République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université 20 Août 1955 de Skikda
Faculté de Technologie - Département de Génie Electrique
Laboratoire de Recherche en Electronique de Skikda

N° : D012118019D

Thèse Présentée en Vue de l'Obtention du Diplôme de

Doctorat ès Sciences

Thème :

Codage Canal et Techniques Efficaces de Décodage Itératif

Spécialité: Electronique - Option: Traitement du Signal

Par :

M. AMAMRA Imed

Dirigé par :

Dr. DEROUICHE Nadir

Soutenue le : 01 / 07 / 2018

Devant le Jury composé de :

Président :	DJEMILI Rafik	Prof. Université 20 Août 1955 de Skikda
Rapporteur :	DEROUICHE Nadir	MCA. Université 20 Août 1955 de Skikda
Examinateur :	BENSLAMA Malek	Prof. Université Frères Mentouri Constantine-1
Examinateur :	LASHAB Mohamed	Prof. Université Larbi Ben Mhidi d'Oum el Bouaghi
Examinateur :	BELMEGUENAI Aissa	MCA. Université 20 Août 1955 de Skikda
Examinateur :	ZEBIRI Chemseddine	MCA. Université Ferhat Abbas Sétif-1

!

سورة هود – 88

إن الحمد لله نحمده سبحانه و تعالى حمدا يليق بجلال وجهه و عظيم سلطانه، فقد سدد الخطى و شرح الصدر و يسر الأمر، فلله الحمد كله و إليه يعود الفضل كله و الصلاة و السلام على اشرف المرسلين سيدنا محمد صلى الله عليه و سلم.

Remerciements

Je tiens à adresser de vifs et sincères remerciements à mon directeur de thèse Mr. N. DEROUICHE. Sa disponibilité, sa générosité intellectuelle, son dynamisme, son objectivité, ses remarques pertinentes, et la confiance qu'il a su m'accorder, m'ont permis de mener à bien mes travaux de thèse. Pendant ces années il n'a cessé de me guider, de m'aider et de m'encourager. Je le remercie de tout cela ainsi que du temps qu'il a su me consacrer.

Je remercie aussi les membres du jury qui m'ont accordé l'honneur d'examiner ce travail, Mr. R. DJEMILI d'avoir présidé le jury, Mr. M. BENSLAMA, Mr. M. LASHAB, Mr. A. BELMEGUENAI et Mr. C. ZEBIRI d'avoir rapporté ce travail et d'avoir participé au jury en tant qu'examinateurs. Leurs remarques et leurs commentaires vont m'être d'une grande utilité.

Je n'oublie pas de dire grand merci à mes collègues et amis, en particulier les membres de l'équipe Technologie de l'Information et de la Communication du laboratoire de recherche LRES.

Je présente mes sentiments de reconnaissance les plus profonds à ma mère, mon défunt père, qui m'ont permis de poursuivre ce long chemin et qui m'ont transmis la passion d'apprendre.

J'adresse mes remerciements à ma femme pour son inestimable soutien et pour tout ce qu'elle a fait pour moi.

ملخص

تلقى اليوم الخوارزميات التكرارية رواجا واسعا في كل مجالات معالجة الإشارة و الاتصالات الرقمية. في نظم الاتصالات الحديثة تحديدا, تستعمل هذه الخوارزميات بالأساس في فك ترميز التربوكود و ترميز التحقق من التكافؤ منخفض الكثافة LDPC, واللذان يعتبران من أصناف رموز تصحيح الأخطاء الأكثر استعمالا لأدائهما العالي و المتميز من ناحية نسبة الخطأ المحتمل.

على الرغم من ذلك, لا تزال الطبيعة التكرارية لخوارزميات فك الترميز, وكذا تعقيدها النسبي, يطرحان تحديا في وجه تطبيقها الميداني, وهذا التحدي مرتبط بالامتثال لمقتضيات التدفق و زمن الاستتار, في النماذج الحديثة المقننة لنظم الاتصالات. كون هذا الإشكال غير محلول كليا في أدبيات الترميز, فقد شكل موضوع بحث في هذه الرسالة.

لقد طرحنا مساهمتان أساسيتان لتحسين فك الترميز التكراري للتربوكود. الأولى تتمثل في معيار الإنهاء الاستباقي لعملية فك الترميز التكرارية, و المصمم بناءا على ملاحظة و تتبع سلوك نسبة التشابه اللوغارتمية, للمخارج البعدية أثناء فك الترميز. سنبين بالمحاكاة على الحاسوب, أن هذه الطريقة المقترحة مقارنة مع طرق مرجعية, تظهر فعالية جيدة في فك الترميز استنادا إلى نسبة الخطأ الثنائي BER بالتوازي مع تقليل معتبر لمتوسط عدد التكرارات.

اقتراحنا الثاني تجسد في تقنية لتحسين أداء و جودة فك الترميز التكراري ذات تصميم مزدوج على مرحلتين. أولا نستعمل اختبار التقارب بتحليل الأنتروبي المتقاطع, و من ثم نستطيع انتقاء الرموز الثنائية غير المتقاربة و وضع معيار إنهاء استباقي فعال. ثانيا نقوم بحساب معامل تعديل يستند إلى مقاربة تكيفية, نطبقه على المعلومة الخارجية للرموز الثنائية غير المتقاربة و المنتقاة سابقا, و ذلك على مستوى كل عنصر فك الترميز للتربوكود. أظهرت نتائج المحاكاة على الحاسوب, أن تطبيق هذا المعيار المقترح مهم و مفيد في نظام إعادة الإرسال الآلي الهجين HARQ و الذي يدمج بين بروتوكول مراقبة الخطأ و ترميز التربوكود, و ذلك بالتقليل من تعقيد فك الترميز إلى حد بعيد. أداء هذه الطريقة يتميز كذلك بتكيفها مع ظروف قناة الإرسال من حيث نسبة الضوضاء في هذه الأخيرة.

الكلمات الدالة : الترميز التلفيفي, فك الترميز التكراري, التربوكود, ترميز LDPC, خوارزمية LogMAP , الأنتروبي المتقاطع, معيار إنهاء استباقي لفك الترميز.

Abstract

Today iterative algorithms are widely used in all areas of signal processing and digital communications. In modern communication systems, iterative algorithms are mainly used in decoding Turbocodes and Low-Density Parity-Check (LDPC) codes, which are two classes of error-correcting codes mainly used for their exceptional error-rate performance. The iterative nature and relative complexity of these decoding algorithms are presenting conceiving challenges to their implementation, while complying with the constraints of throughput and latency, in new standards of telecommunication systems. Being not completely solved in the literature, this problem makes the subject of this thesis.

We present two main contributions for improving the iterative decoding of Turbocodes. The first contribution is an early stopping criterion of the iterative decoding process, which is devised by observing the log-likelihood ratio behavior LLR of the a posteriori outputs during decoding. Simulation results in comparing with some reference methods show that our method achieved an acceptable BER (Bit Error Rate) performance and reduced considerably the average number of iterations.

The second contribution is a technique for improving and optimizing the iterative decoding designed on two phases. First, a bit-level convergence test using the cross-entropy analyses is used to select non converged bits and establish a simple and effective stopping rule. Next, an adaptive approach is used to compute a scaling factor for normalizing the extrinsic information of the previously selected bits. The extra coding gain obtained from this normalization can compensate for the performance degradation of the stopping rule. Simulation results of the proposed criterion show an interesting application in a hybrid automatic retransmission system HARQ, combining an error control protocol and a Turbocode, where the complexity of decoding is considerably reduced. Also in comparison with some previously published stopping rules, this method shows an adaptive termination according to a changing SNR environment.

Keywords : Convolutional coding, iterative decoding, Turbocode, LDPC code, LogMAP algorithm, cross entropy, early stopping criterion.

Résumé

L'utilisation d'algorithmes itératifs est aujourd'hui largement répandue dans tous les domaines du traitement du signal et des communications numériques. Dans les systèmes de communications modernes, les algorithmes itératifs sont principalement utilisés dans le décodage des Turbocodes et des codes LDPC (Low-Density Parity-Check), qui sont deux classes de codes correcteurs d'erreurs utilisés pour leurs performances exceptionnelles en terme de taux d'erreur. La nature itérative des algorithmes de décodage et leur complexité relative présentent encore un défit face à leur implémentation, tout en se conformant au contraintes de débit et de latence, dans les nouvelles normes des systèmes de télécommunication. Ce problème étant non entièrement résolu dans la littérature, fait objet de cette thèse.

Nous avons deux contributions principales pour l'amélioration du décodage itératif des Turbocodes. Notre première contribution est un critère d'arrêt anticipé du processus de décodage itératif, qui est conçu en observant le comportement du rapport de vraisemblance logarithmique LLR des sorties a posteriori lors du décodage. Nous montrons par des simulations que notre méthode en comparaison avec des méthodes de référence, atteint de bonne performance de décodage en terme du taux d'erreur binaire BER, tout en réduisant considérablement le nombre d'itérations moyen.

Notre deuxième contribution est une technique d'amélioration et d'optimisation du décodage itératif conçu sur deux phases. Premièrement, un test de convergence par l'analyse de l'entropie croisée, est utilisé pour sélectionner les bits non-convergents et établir une règle d'arrêt anticipé efficace. Ensuite, une approche adaptative pour le calcule d'un facteur de normalisation est appliquée à l'information extrinsèque des bits non-convergents précédemment sélectionnés, et qui est échangée entre les deux composant-décodeurs du Turbocode. Les résultats de simulations, du critère proposé montre une application intéressante dans un système de retransmission automatique hybride HARQ combinant un protocole de contrôle d'erreur et un Turbocode, où la complexité de décodage est considérablement réduite. Ainsi cette méthode montre une adaptabilité en vers les changements des conditions du canal en terme de rapport signal-à-bruit.

Mots-clefs : Codage convolutif, décodage itératif, Turbocode, code LDPC, algorithme LogMAP, entropie croisée, critère d'arrêt anticipé.

Table des figures

Glossaire

AAL5	ATM Adaptation Layer 5
ACK	Acknowledgement
ARQ	Automatic Repeat reQuest
ARP	Almost Regular Permutation
ATM	Asynchronous Transfer Mode
AUTODIN-II	Automatic Digital Network II
AWGN	Additive White Gaussian Noise
BCH	Bose Chaudhuri Hocquenghem
BCJR	Bahl Cocke Jelinek Raviv
BEC	Binary Erasure Channel
BER	Bit Error Rate
BF	Bit-Flipping
BI-AWGN	Binary Input - Additive White Gaussian Noise
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
CCSDS	Consultative Committee for Space Data Systems
CDMA	Code Division Multiple Access
CE	Cross Entropy
CRC	Cyclic Redundancy Check
DRP	Dithered Relatively Prime
DVB-RCS	Digital Video Broadcasting - Return Channel on Satellite
DVB-RCT	Digital Video Broadcasting - Return Channel on Terrestrial
DVB-S2	Digital Video Broadcasting - Satellite - Second Generation
EG-LDPC	Euclidean Geometry - LDPC
FEC	Forward Error Correction
FER	Frame Error Rate
FPTD	Fully-Parallel Turbo Decoder
HARQ	Hybrid Automatic Repeat reQuest
HDA	Hard-Decision-Aided
HLLR	Histogram of Log-Likelihood Ratio
HSPA	High Speed Packet Access
IP	Intellectual Property core
LDPC	Low-Density Parity-Check
LFSR	Linear Feedback Shift Register
LLR	Logarithmic Likelihood ratio
Log-MAP	Logarithmic Maximum A posteriori Probability
LOS	Line Of Sight
LTE	Long Term Evolution
LUT	Look-Up Table
ML	Maximum Likelihood

MOPS	Million Operations Per Second
MP	Message Passing
NAK	Not AcKnowledged
PCE	Partial Cross-Entropy
PDF	Probability Density Function
PG-LDPC	Projective Geometry - LDPC
QC-LDPC	Quasi-Cyclic LDPC
RSC	Recursive Systematic Convolutional
RTL	Register-Transfer-Level
RTZ	Return To Zero
SCR	Sign Change Ratio
SISO	Soft-Input Soft-Output
SOVA	Soft-Output Viterbi Algorithm
SP	Sum-Product
SNR	Signal to Noise Ratio
SOC	System On Chip
SW-ARQ	Stop and Wait Automatic Repeat request
TC	Turbo Code
TCH/FS-HS-EFS	Traffic Channel/ Full rate for Speech - Half rate for Speech - Enhanced Full rate for Speech
UMTS	Universal Mobile Telecommunications System
VA	Viterbi Algorithm
VHDL	Very high speed integrated circuit Hardware Description Language
WCDMA	Wideband Code Division Multiple Access
WiMAX	Worldwide interoperability for Microwave Access
WLAN	Wireless Local Area Network

SOMMAIRE

Introduction

Aujourd'hui, les télécommunications numériques sont omniprésentes dans nos vies. Quel que soit le moyen physique utilisé pour communiquer, filaire, sans fil, fixe ou mobile, les étapes permettant de transmettre un message d'une source à une destination au travers d'un canal de communication se modélisent de la même façon. Ce schéma de transmission est représenté sur la figure 1.

L'émetteur génère le message qu'il souhaite transmettre, il s'agit d'une suite de symboles binaires. Ce message n'est pas créé aléatoirement, il est donc possible de le compresser afin de réduire la quantité de données à transmettre. Cette étape, appelée codage source, est importante et donne à sa sortie une séquence de symboles binaires $\boldsymbol{u}$. En effet, la transmission de données à un coût, aussi minime qu'il soit, il est indispensable de transmettre la quantité de données minimale.

Le codage canal prend cette séquence de bits et ajoute de la redondance afin de permettre au destinataire de détecter et, éventuellement, corriger les erreurs. Ainsi il produit à sa sortie une séquence codée $\boldsymbol{c}$ appelée mot de code. Chaque symbole de sortie du codeur de canal est transformé en une forme d'onde par le modulateur dite signal modulé qui convient pour la transmission. Quel que soit le type de canal utilisé, des erreurs viendront altérer le message transmis. Ce code doit être adapté au type du canal ainsi qu'à sa probabilité d'erreur.

Du côté récepteur, en applique les différentes opérations, en sens inverse, pour retrouver le contenu du message. Il faut donc démoduler pour produire une sortie discrète ou continue (la séquence reçue $\boldsymbol{r}$), puis décoder (corriger) pour avoir une séquence binaire $\widehat{\boldsymbol{u}}$ appelée séquence d'information estimée avec une certaine probabilité d’erreur. Enfin décompresser pour aboutir au message reçu.

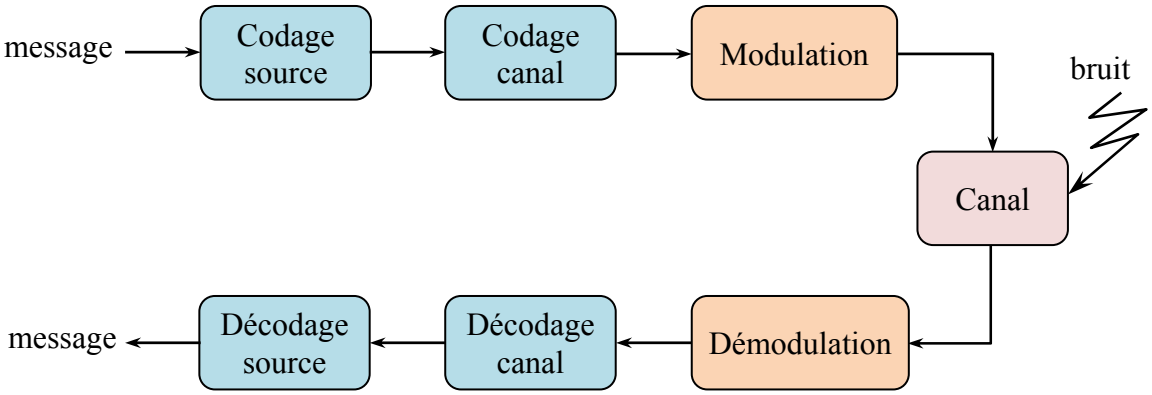

Figure 1. Schéma de transmission numérique

Considéré comme point de repère, le domaine du codage canal a commencé en 1948 avec l'article révolutionnaire de Claude Shannon [1]. Dans lequel il a montré le débit théorique maximal de données pour lesquelles des communications fiables pourraient avoir lieu. Shannon a montré que la probabilité d'erreur d'un canal à bruit blanc gaussien et additif (AWGN) peut être extrêmement faible, à condition que le débit de données soit inférieur ou égal à la capacité du canal. Cette preuve utilisait des mots de code générés aléatoirement et un décodage typique sous-optimal. En utilisant cette méthode, Shannon a été en mesure de démontrer que les régions de décodage pour des mots de code distincts ne se chevauchent que dans une partie négligeable. Malheureusement, la preuve de Shannon ne nous dit pas comment construire des codes qui permettront d'atteindre la capacité du canal.

En raison de leur manque de structure, les codes aléatoires sont notoirement difficiles à décoder. L'ajout d'une structure à un code simplifie considérablement le processus de décodage, mais les codes structurés sont moins performants par rapport à la limite théorique. Ceci est la base du paradoxe de la théorie du codage : *"All codes are good...except those we can think of" J. Wolfowitz.*

Dans les années 1950, les théoriciens ont proposé la première famille de codes correcteurs d'erreurs : les codes en bloc linéaires, à savoir les codes de Hamming [2], de BCH, de Reed-Solomon et de Reed-Muller. Ces codes en bloc ont dominés le domaine du codage canal pendant les premières décennies, où le canal était universellement supposé être un canal symétrique binaire (BSC).

Parallèlement à l'évolution des codes en blocs linéaires, l'invention des codes convolutifs par Elias en 1955 [3] conduit à une approche différente et à l'invention de méthodes de décodage en treillis telles que l'algorithme de Viterbi [4], l'algorithme BCJR (Bahl, Cocke,

Jelinek et Raviv) [5]. Ces deux algorithmes peuvent être facilement adaptés à n'importe quel canal et donc généraliser le concept de correction d'erreur à des canaux généraux pour lesquels l'alphabet de sortie de canal n'est pas nécessairement le même que l'alphabet d'entrée. Nous parlons maintenant de "codage canal" plutôt que de "codage correcteur d'erreur". Puis, le principe des codes concaténés est apparu pour diminuer l'encombrement du décodeur à la réception. Mais il fallut attendre Claude Berrou en 1993 [6,7] pour aboutir à l'invention des 'TurboCodes', qui atteignent des performances très proches de la limite théorique de Shannon. Cette famille de code combine à la fois une concaténation parallèle et un décodage itératif. L'invention des turbocodes a permis de franchir une étape importante et d'aboutir à des codes correcteurs d'erreurs très performants.

Dans la bousculade qui suit pour expliquer la théorie derrière cette performance déroutante, une méthode développée à l'origine par Gallager dans son doctorat [8], connue sous le nom de codage LDPC (Low-Density Parity-Check), a été redécouverte par MacKay [9] et ayant des propriétés comparables à celles des turbocodes. Ces deux méthodes sont devenues les locomotives des normes de communication modernes.

Les algorithmes de décodage de ces codes sont de nature itérative, c'est-à-dire qu'ils doivent effectuer un certain nombre d'itérations avant d'atteindre un degré de confiance satisfaisant en ce qui concerne une trame à décoder. Traditionnellement, les turbodécodeurs standard utilisent un nombre fixe d'itérations par trame décodée. Dans cette thèse, nous proposons quelques règles d'arrêt simples qui peuvent être utilisées pour réduire le nombre moyen d'itérations. Ces techniques offrent un compromis entre le débit et la performance et peuvent fournir une augmentation significative de la vitesse moyenne de décodage tout en maintenant les performances du décodeur sans diminution significative.

Organisation du manuscrit :

Nous commençons donc par rappeler quelques généralités sur les codes linéaires et les différents modèles de canaux, nécessaires à la compréhension des chapitres suivants. Nous rappelons brièvement les concepts de base du décodage itératif, et les principaux codes qui utilisent à ce jour le décodage itératif, à savoir les codes LDPC et les codes concaténés.

Les codes LDPC sont ensuite introduits dans le chapitre 2. Une synthèse des différents algorithmes de construction est également proposée. Ensuite, une étude détaillée sur les

méthodes permettant l'encodage et les algorithmes itératifs de décodage des codes LDPC sera présentée.

Le chapitre 3 présente de façon un peu plus détaillée les codes convolutifs et les turbocodes. Les propriétés de base de leur structure et des distances sont définies. Le concept du treillis, qui est utilisé plusieurs fois tout au long de la thèse, est défini. Cela comprend une description des différents composants d'un turbocodeur, à savoir les codeurs convolutifs RSC constitutifs et l'entrelaceur, ainsi que le concept de décodage itératif, servants de base pour le reste de la thèse. Ce dernier inclut une brève revue de l'algorithme BCJR et ses dérivées adoptés pour le décodage itératif.

Au chapitre 4, les algorithmes du turbodécodeur et les techniques de conception sont discutés, en mettant l'accent sur la réduction du nombre d'itération et la consommation d'énergie. Nous proposons, par la suite, des techniques de décodage dynamique-itératif pour la famille des turbocodes. Deux nouvelles techniques dynamiques-itératives (HLLR, PCE) sont introduites. De plus la deuxième méthode (PCE) est appliquée dans un système HARQ de Type-I, où sa performance est évaluée.

La conclusion générale de ce travail synthétise les différentes idées présentées dans ce document. Les perspectives de travaux futurs sont également décrites.

Chapitre 1

Canaux, Codes et Capacité

Dans ce chapitre, nous introduisons l'hypothèse de notre travail: la communication d'un message numérique sans erreur (ou avec le moins d'erreurs possible) en dépit d'un milieu de communication imparfait.

Tout d'abord, nous présentons des modèles de canaux discrets et sans mémoire et leur description unifiée. Ensuite, nous introduisons les principaux fondements de la théorie de l'information et la transmission de données, ainsi que le deuxième théorème de Shannon du codage de canal. Enfin, nous nous penchons directement dans les principes du concept de la détection et la correction d'erreurs de transmission.

1.1 Modèles de Canaux Radio:

Un système de communication est un moyen de transmission d'informations d'un utilisateur à un autre. Dans un système de communication numérique les données sont transmises sous forme numérique. Le schéma de principe de ce dernier est représenté sur la Fig. 1.1.

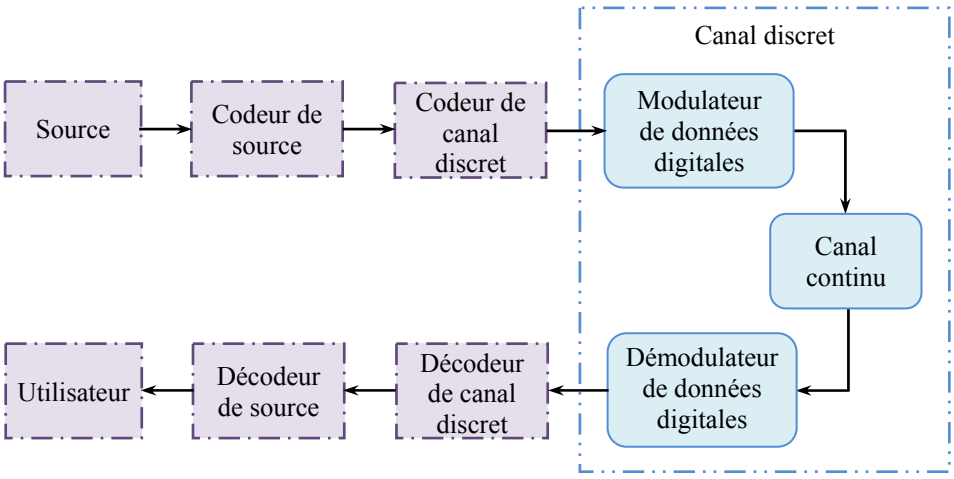

Figure.1.1 Système de communication numérique avec canal discret

Le codage de source est utilisé pour éliminer la redondance de l'information de source pour une transmission efficace. La puissance du signal transmis et la bande passante du canal sont les paramètres clés dans la conception du système de communication numérique.

En utilisant ces paramètres, l'énergie du signal par bit E_b à la densité spectrale de puissance du bruit N_0 est déterminée. Ce rapport est unique pour déterminer la probabilité d'erreur de bit, souvent désigné comme le taux d'erreur binaire (BER). Dans la pratique, pour un E_b/N_0 fixe, un BER acceptable est possible avec le codage de canal. Ceci peut être réalisé en ajoutant des symboles supplémentaires dans le flux d'informations transmises. Ces symboles supplémentaires n'apportent pas de nouvelle information, mais ils permettent au récepteur de détecter et corriger les erreurs, réduisant ainsi la probabilité d'erreur globale.

Pour définir un canal de transmission, on décrit l'ensemble des entrées et des sorties possibles du canal ainsi que le bruit qui peut perturber la transmission.

Le canal discret est le plus simple des canaux de transmission. L'ensemble des entrées et l'ensemble des sorties sont deux alphabets finis X et Y. Le canal est discret sans mémoire si le bruit se modélise comme une probabilité conditionnelle de Y sachant X indépendante du temps. Un canal discret sans mémoire est donc défini par :

- l'alphabet d'entrée X.
- l'alphabet de sortie Y.
- la matrice de transition $\boldsymbol{P_{CH}} = \left(P_{CH\,i,j}\right)$ où $P_{CH\,i,j} = p(y_j|x_i)$.

1.1.1 Le Canal Binaire Symétrique BSC :

Le BSC est un canal discret sans mémoire qui véhicule des symboles binaires à la fois dans l'entrée et la sortie. Il est symétrique car la probabilité de recevoir 0 lorsque 1 est transmis, est la même que la probabilité de recevoir 1 lorsque 0 est transmis.

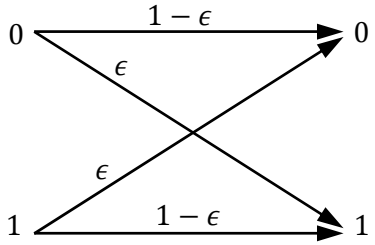

Figure.1.2 Modèle du Canal Binaire Symétrique

Cette probabilité est appelée la probabilité de croisement du canal notée ϵ comme représenté sur la Fig.1.2. La probabilité d'absence d'erreur, à savoir, recevoir le même symbole que celui transmis est $1-\epsilon$. Par conséquent, les probabilités de transition du canal BSC sont :

$$\left.\begin{array}{l} p(y=0|x=0)=1-\epsilon \\ p(y=0|x=1)=\epsilon \\ p(y=1|x=0)=\epsilon \\ p(y=1|x=1)=1-\epsilon \end{array}\right\} \quad (1.1)$$

D'où la matrice de transition $\boldsymbol{P_{CH}} = \begin{bmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{bmatrix}$

Notez qu'on peut toujours supposer que $\epsilon \leq 1/2$. Si ce n'est pas le cas, il suffit de changer, à la sortie du canal, tous les 0 en 1 et les 1 en 0 [10].

1.1.2 Le Canal Binaire à Effacement BEC :

L'effacement est un type spécial d'erreur avec un emplacement connu. Le BEC transmet l'un des deux bits binaires 0 et 1. Cependant, un effacement «*e*» est produit lorsque le récepteur reçoit un bit peu fiable. La sortie du canal BEC est constituée des symboles 0, 1 et *e*, comme illustré sur la Fig.1.3. Le BEC efface un bit avec une probabilité ε, appelée la probabilité du canal à effacement. Ainsi, les probabilités de transition de canal pour le BEC sont les suivantes:

$$\left.\begin{array}{l} p(y=0|x=0)=1-\varepsilon \\ p(y=e|x=0)=\varepsilon \\ p(y=1|x=0)=0 \\ p(y=0|x=1)=0 \\ p(y=e|x=1)=\varepsilon \\ p(y=1|x=1)=1-\varepsilon \end{array}\right\} \quad (1.2)$$

D'où la matrice de transition $\boldsymbol{P_{CH}} = \begin{bmatrix} 1-\varepsilon & \varepsilon & 0 \\ 0 & \varepsilon & 1-\varepsilon \end{bmatrix}$

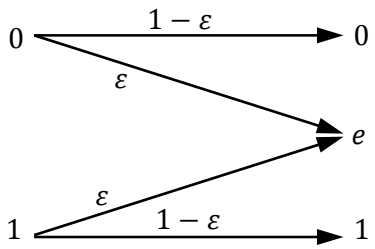

Figure.1.3 Modèle du Canal Binaire à Effacement

On remarquera que la variable aléatoire de sortie du canal Y prend ses valeurs dans un alphabet ternaire {0,1,*e*} où *e* représente l'effacement. Ce canal est souvent utilisé comme modèle pour les réseaux. L'effacement modélise alors les pertes de paquets [10].

1.1.3 Le Canal à Bruit Blanc Additif Gaussien AWGN :

Dans un canal AWGN, le signal est dégradé par le bruit blanc η qui a une densité spectrale constante et une distribution d'amplitude gaussienne. Le canal que nous considérons, et le plus couramment utilisé par les théoriciens de codage. C'est un canal d'entrée binaire avec bruit additif modélisée comme des échantillons d'une distribution de probabilité gaussienne [11].

La distribution gaussienne a une fonction de densité de probabilité donnée par :

$$P_{df}(\eta) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{\eta^2}{2\sigma^2}\right) \tag{1.3}$$

Où σ^2 est la variance d'un processus aléatoire Gaussien.

Le canal à bruit blanc gaussien additif d'entrée binaire (BI-AWGN) peut être décrit par l'équation :

$$y_i = ax_i + \eta_i \tag{1.4}$$

Où $x_i \in \{-1, +1\}$ est le $i^{\text{ème}}$ symbole transmis, y_i est le ième symbole reçu, η_i est le bruit additif échantillonné à partir de η et a est l'atténuation.

Lors de la transmission d'un mot de code binaire sur le canal BI-AWGN, les bits du mot de code $c_i \in \{0,1\}$ peuvent être mappés aux symboles $x_i \in \{-1, +1\}$ par une des deux façons suivantes: $\{0 \to +1\,, 1 \to -1\}$ ou $\{0 \to -1\,, 1 \to +1\}$.

Nous allons utiliser la convention traditionnelle $\{0 \to +1\,, 1 \to -1\}$, parce que l'arithmétique modulo-2 sur {0,1} correspond directement à la multiplication avec $\{-1, +1\}$ lorsque ce mappage traditionnel est utilisé [11].

1.1.4 Le Canal à Evanouissement :

Dans le canal radio, la puissance reçue est affectée par l'atténuation due à la combinaison des effets suivants :

- Affaiblissement de parcours (La perte de trajet): Il est l'atténuation du signal. La puissance reçue par l'antenne de réception diminue lorsque la distance entre l'émetteur et le récepteur augmente. L'atténuation de puissance est proportionnelle à (la distance)$^{\alpha}$, où les valeurs de α varient de 2 à 4. Lorsque la distance varie avec le temps, la perte de trajet varie également.
- Perte due à l'effet de masque (Shadowing) : Elle est due à l'absorption du signal rayonné par les structures dispersives. Il est dérivé d'une variable aléatoire avec une distribution log-normale.
- La perte d'évanouissement : La combinaison de la propagation par trajets multiples, et l'effet Doppler (décalage de fréquence) produit les fluctuations aléatoires de la puissance reçue qui donne les pertes d'évanouissement.

1.1.4.1. Modèles Probabilistes des Canaux à Evanouissement :

a. Canal de Rice :

Lorsque le signal reçu est composé de plusieurs rayons de réflexions et un rayon significatif en ligne de vue (Line Of Sight LOS), l'amplitude de l'enveloppe reçue a une fonction densité de probabilité de Rice qui est donnée dans l'équation (1.5), et l'évanouissement est dit Ricien.

$$P_{df}(x) = \begin{cases} \frac{x}{\sigma^2} exp\left(-\frac{x^2+A^2}{2\sigma^2}\right) I_0\left(\frac{xA}{\sigma^2}\right) & pour \;\; x \geq 0\,, A \geq 0 \\ 0 & autrement \end{cases} \qquad (1.5)$$

Où : x est l'amplitude du signal reçu, I_0 est la fonction de Bessel modifiée d'ordre zéro du premier type, et A représente l'amplitude de crête du signal LOS.

b. Canal de Rayleigh :

L'évanouissement de Rayleigh apparaît quand il y a plusieurs chemins indirects entre l'émetteur et le récepteur et aucun chemin en ligne de vue (LOS). Ce cas représente le pire scénario pour le canal de transmission. L'évanouissement de Rayleigh considère qu'un signal multi-trajet reçu est composé d'un grand nombre d'ondes réfléchies avec des phases et des amplitudes indépendantes et identiquement distribuées. L'enveloppe du signal porteur reçu est de distribution de Rayleigh dans les communications sans fil.

Lorsque l'amplitude de la composante LOS se rapproche de zéro, la PDF de Rice se rapproche d'une PDF de Rayleigh exprimée comme suit:

$$P_{df}(x) = \begin{cases} \frac{x}{\sigma^2} exp\left(-\frac{x^2}{2\sigma^2}\right) & pour \quad x \geq 0 \\ 0 & autrement \end{cases} \tag{1.6}$$

Le canal à bruit blanc gaussien additif et le canal de Rice, qui modélisent un environnement de campagne, se distinguent par une assez bonne performance. Tandis que le canal de Rayleigh, qui décrit mieux l'évanouissement de l'environnement urbain, offre des performances relativement plus mauvaises.

1.2 Capacité d'un Canal et le Deuxième Théorème de Shannon :

Dans un document historique écrit à Bell Labs en 1948 [1], Claude Elwood Shannon a développé une théorie mathématique appelée *théorie de l'information* qui décrit les aspects les plus fondamentaux des systèmes de communication. Cette théorie s'intéresse à la construction et à l'étude de modèles mathématiques à l'aide essentiellement de la théorie des probabilités. Depuis ce premier exposé, la théorie de l'information s'est faite de plus en plus précise et est devenue aujourd'hui incontournable dans la conception de tout système de communication.

1.2.1. Mesure Quantitative de l'Information et l'Entropie :

La sortie d'une source discrète sans mémoire est une séquence de symboles tirées dans un alphabet fini $X = \{x_1, x_2, \dots, x_M\}$. Chaque symbole de la séquence est choisi aléatoirement d'après une loi de probabilité p indépendante du temps. Pour tout symbole x, $p(x)$ est la probabilité pour que ce symbole soit choisi. Il s'agit d'un réel compris entre 0 et 1. On a $\sum_{x \in A} p(x) = 1$. La donnée de $p(x_1)$,..., $p(x_M)$ définit la probabilité discrète p sur X.

Si un symbole x a une probabilité $p(x)$ d'être tiré, son *information propre* est définie par :

$$I(x) = -\log_2 p(x) \tag{1.7}$$

L'information propre s'interprète comme la quantité d'information fournie par la réalisation de cet événement. Notons que l'information propre est toujours positive ou nulle et que, plus un événement est improbable, plus son information propre est grande. A l'inverse, la réalisation d'un événement certain n'apporte aucune information, ce qui semble conforme à l'intuition [12].

La valeur moyenne de l'information propre calculée sur l'ensemble de l'alphabet X revêt une grande importance. Il s'agit donc de l'espérance de la variable aléatoire I. Elle est appelée *entropie* de la source et est notée $H(X)$:

$$H(X) = -\sum_{x \in A} p(x) \log_2 p(x) \tag{1.8}$$

Si une source émet M symboles équiprobables (ou encore avec une loi de probabilité uniforme), son entropie est donc $\log_2 M$. Si $M = 2^k$, son entropie est alors k. Or pour représenter 2^k symboles distincts en binaires, k cases (bits) sont nécessaires.

L'entropie d'une source est quelquefois donnée en *bits/seconde*. Si l'entropie d'une source discrète est H et si les symboles sont émis tous les τ_s secondes, son entropie en *bits/s* est H/τ_s.

On considère un espace probabilisé joint XY où $X = \{x_1, x_2, \dots, x_M\}$ et $Y = \{y_1, y_2, \dots, y_N\}$. L'*information mutuelle* entre les événements $\{X = x\}$ et $\{Y = y\}$ est définie par :

$$I(x; y) = \log_2 \frac{p(x|y)}{p(x)} \tag{1.9}$$

Par définition $p(x, y) = p(x|y)p(y) = p(y|x)p(x)$. Donc :

$$I(x; y) = I(y; x) = \log_2 \frac{p(x,y)}{p(x)p(y)} \tag{1.10}$$

L'information mutuelle $I(x;\ y)$ a les propriétés suivantes :

- $I(x;\ y) > 0$ signifie que si l'un des deux événements se réalise, alors la probabilité de l'autre augmente.
- $I(x;\ y) < 0$ signifie que si l'un des deux événements se réalise, alors la probabilité de l'autre diminue.
- $I(x;\ y) = 0$ signifie que les deux événements sont statistiquement indépendants.

Dans l'espace probabilisé joint XY, la quantité d'information fournie par l'événement $\{X = x\}$ sachant que l'événement $\{Y = y\}$ est réalisé, appelée l'*information propre conditionnelle* de x sachant y, est définie par :

$$I(x|y) = -\log_2 p(x|y) \tag{1.11}$$

L'information mutuelle entre deux événements est donc $I(x;\ y) = I(x) - I(x|y)$.

L'*information mutuelle moyenne* de X et Y dans l'espace probabilisé joint XY est définie par :

$$I(X;Y) = \sum_{x \in X, y \in Y} p(x,y) I(x;y) \tag{1.12}$$

On peut également réécrire l'entropie de l'espace X en fonction de l'information propre par : $H(X) = \sum_{x \in X} p(x) I(x)$.

Enfin l'information propre conditionnelle est une variable aléatoire réelle et nous pouvons définir sa moyenne appelée *entropie conditionnelle* de X sachant Y. Elle est définie sur l'espace de probabilité joint XY [12]:

$$H(X|Y) = -\sum_{x \in X, y \in Y} p(x,y) \log_2 p(x|y) \tag{1.13}$$

On en déduit que $I(X;Y) = H(X) - H(X|Y)$ [12].

1.2.2. Capacité du Canal :

Pour tout canal de transmission on définit une grandeur caractéristique appelée capacité du canal et que l'on peut interpréter comme la quantité maximale d'information pouvant transiter à travers le canal.

On considère un canal discret sans mémoire donné par X, Y, et $\boldsymbol{P_{CH}}$. Lorsqu'on fait varier la distribution d'entrée p_X, l'information mutuelle moyenne $I(X;Y)$ varie. La valeur maximale que l'on peut atteindre s'appelle la capacité de Shannon du canal :

$$C = \max_{p_X} I(X;Y) \tag{1.14}$$

La capacité est donc un réel positif ou nul, inférieur ou égal au log à base 2 du cardinal de X et de Y. La capacité des canaux les plus répondus dans les systèmes de communications sont citées dans le tableau.1.

Nom du Canal	Sa Capacité C
BSC	$1 + \epsilon \log_2 \epsilon + (1-\epsilon)\log_2(1-\epsilon)$
BEC	$1 - \varepsilon$
AWGN	$\frac{1}{2}\log_2(1 + \mathrm{SNR})$

Tableau 1

1.2.3. Théorème de Codage de Canal :

On définit le *rendement* (ou taux) d'un code binaire de cardinal M constitué par des mots de longueur n par :

$$R = \frac{\log_2 M}{n} \tag{1.15}$$

Un code aura un taux 1 lorsque $M = 2^n$, c'est-à-dire lorsqu'aucune redondance n'aura été ajoutée. Un code de taux 1/2 signifie que $M = 2^n/2$, c'est-à-dire que le code double la longueur des séquences binaires de la source [12].

Théorème : Soit C la capacité d'un canal. Si $R < C$ alors il existe un codage qui permet d'avoir une probabilité d'erreur de décodage évanescente lorsque n tend vers l'infini. Inversement, s'il existe un code dont la probabilité d'erreur de décodage est évanescente alors nécessairement $R \leq C$ [13].

Ce théorème comporte deux résultats. Le premier résultat du théorème de codage est appelé *atteignabilité* car on dit qu'un rendement de codage est atteignable lorsque la probabilité de décodage est aussi petite que l'on veut pour n suffisamment grand. Le second est appelé *réciproque* et affirme qu'il ne peut y avoir de codage ayant un rendement supérieur à la capacité ayant une probabilité de décodage évanescente avec n.

Le second théorème de Shannon prédit l'existence de codes dont le rendement peut valoir la capacité, a priori différente de zéro, et qui pourtant produisent une probabilité d'erreur de décodage évanescente. Le théorème affirme donc bien que pour un degré de redondance limité il est possible d'éviter toute ambiguïté de décodage en réception.

1.3 Principe de Détection d'Erreur et Protocoles de Retransmission :

1.3.1. Définition de la fonction du codage :

Le message transmis par la source peut être remplacé par un autre, à condition qu'il en soit déduit d'une certaine manière et réversible. Il n'y a ni création ni destruction d'information, et l'information reste invariante par rapport à l'ensemble des messages qui peuvent être utilisés pour la communiquer. Puisqu'il est possible d'assigner des messages avec

des caractéristiques différentes à la même information, les transformations d'un message initial permettent de l'équiper a priori de propriétés souhaitables.

On peut envisager a priori de transformer un message numérique par codage source, codage canal et cryptographie. Le codage source vise à obtenir une concision maximale. L'utilisation d'un canal est plus coûteuse, plus le message est long, le «coût» ici signifie très généralement l'exigence de ressources limitées, telles que le temps, la puissance ou la bande passante. Pour diminuer ce coût, le codage peut ainsi viser à remplacer le message transmis par la source par le message le plus court possible. Il est nécessaire que le codage soit réversible.

L'objectif du codage de canal est complètement différent, qui est la protection du message contre le bruit de canal. Nous insistons sur la nécessité de prendre en compte le bruit des canaux au point de faire de leur existence une propriété spécifique. Si le résultat de ce bruit est une probabilité d'erreur de symbole incompatible avec la qualité de restitution spécifiée, nous proposons de transformer le message initial par un tel codage afin d'augmenter la sécurité de transmission en présence de bruit. La théorie n'exclut même pas le cas extrême où la qualité spécifiée est l'absence totale d'erreurs.

La possibilité réelle de protéger les messages contre le bruit des canaux n'est pas évidente. Cette protection fera l'objet de l’intérêt général de cette thèse, avec focalisation sur les techniques avancées actuelles et récentes.

Les objectifs de codage source et de codage de canal semblent donc être incompatibles. Ils sont même contradictoires, puisque le codage source augmente la vulnérabilité aux erreurs tout en améliorant la concision.

Remarquons enfin qu'une procédure de codage peut encore avoir une autre fonction, en théorie sans affecter la redondance ou la vulnérabilité aux erreurs: le chiffrement du message, c'est-à-dire le rendre inintelligible pour quiconque sauf son destinataire, en opérant une transformation secrète que lui seul peut inverser. Le déchiffrement, c'est-à-dire la reconstruction du message par un intercepteur indiscret qui ne connaît pas la «clé» spécifiant la transformation permettant de l'inverser, doit être assez difficile pour constituer une impossibilité factuelle. D'autres fonctions impliquent également la cryptographie, par exemple, fournir au message des propriétés permettant d'authentifier son origine (identifier la source sans ambiguïté ou erreur), ou rendre toute détérioration du message par oblitération,

insertion ou substitution de symboles détectables. D'une manière générale, il s'agit de protéger le message contre les indiscrétions ou les détériorations frauduleuses [13].

1.3.2. Détection d'Erreurs par Parité et Protocoles de Retransmission ARQ:

Le schéma fonctionnel représenté sur la figure 1.1 représente un système de communication unidirectionnel. La transmission (ou l'enregistrement) est strictement dans un sens, de l'émetteur au récepteur. Les stratégies de contrôle d'erreur pour un système unidirectionnel doivent utiliser la correction d'erreur directe *Forward Error Correction* (FEC); C'est-à-dire qu'ils utilisent des codes de correction d'erreur qui corrigent automatiquement les erreurs détectées au niveau du récepteur. Des exemples sont des systèmes de stockage numériques, dans lesquels les informations enregistrées sur un support de stockage peuvent être reproduites des semaines voire des mois après son enregistrement; et des systèmes de communication en espace profond (deep-space), où l'équipement de codage relativement simple peut être placé à bord du vaisseau spatial, mais la procédure de décodage beaucoup plus complexe doit être effectuée sur terre. La plupart des systèmes codés utilisés aujourd'hui emploient une certaine forme de FEC, même si le canal n'est pas strictement à sens unique.

1.3.2.1. Description Générale du Protocole de Retransmission ARQ :

Cependant, dans certains cas, un système de transmission peut être bidirectionnel; c'est-à-dire que l'information peut être envoyée dans les deux sens, et l'émetteur agit également comme un récepteur (*tranceiver*) et vice versa. Des exemples de systèmes bidirectionnels sont les réseaux de données et les systèmes de communication par satellite. Les stratégies de contrôle d'erreur pour les systèmes bidirectionnels utilisent la détection d'erreur et la retransmission, appelée *Automatic Repeat reQuest* (ARQ). Dans un système ARQ, lorsque des erreurs sont détectées au niveau du récepteur, une demande est envoyée pour que l'émetteur répète le message, et des demandes de répétition continuent d'être envoyées jusqu'à ce que le message soit correctement reçu [14].

Il existe deux protocoles pour les systèmes ARQ: *stop-and-wait* (SW-ARQ) et l'ARQ en continu. Avec SW-ARQ, l'émetteur envoie un paquet ou mot de code au récepteur et attend un accusé de réception (*acknowledgement*) positif (ACK) ou négatif (NAK) du récepteur. Si un ACK est reçu (aucune erreur détectée), l'émetteur envoie le paquet suivant; cependant, si un NAK est reçu (erreurs détectées), l'émetteur renvoie le paquet précédent. Lorsque le bruit

est persistant, le même paquet peut être retransmis plusieurs fois avant d'être correctement reçu et reconnu. Avec un ARQ en continu, l'émetteur envoie des paquets au récepteur en continu et reçoit des accusés de réception en continu. Lorsqu'un NAK est reçu, l'émetteur commence une retransmission. Il peut revenir au paquet erroné et renvoyer le, plus les N-1 paquets qui le suivent. C'est ce qui est appelé *Go-back-N* ARQ. Sinon, l'émetteur peut simplement renvoyer uniquement les paquets qui sont négativement reconnus. C'est ce qu'on appelle *selective repeat* ARQ.

L'avantage majeur d'un système à détection d'erreur avec un protocole ARQ sur un autre avec FEC est que la détection d'erreur nécessite un équipement de décodage beaucoup plus simple que la correction d'erreur. En outre, l'ARQ est adaptatif dans le sens où l'information est retransmise seulement quand des erreurs se produisent. En revanche, lorsque la probabilité d'erreur dans le canal est élevée, les retransmissions doivent être envoyées plus fréquemment et le débit du système (la vitesse à laquelle les messages nouvellement générés sont correctement reçus) est fortement réduit par l'ARQ. Dans cette situation, une combinaison hybride avec un codage FEC pour les erreurs les plus fréquentes est plus efficace que l'ARQ seule [14].

Le débit (*throughput*) est une mesure de l'efficacité d'un protocole ARQ, défini dans ce travail comme le nombre de paquets acceptés par le nombre total de paquets transmis, si ce n'est pas indiqué autrement (section 4.2).

$$throughput = \frac{\sum paquets\ acceptés}{\sum paquets\ transmis} \tag{1.16}$$

Nous souhaitons que le throughput soit aussi grand que possible, c'est-à-dire approchant 1, mais nous voulons toujours que le taux d'erreur parmi les paquets acceptés soit le plus petit possible, approchant 0. Par conséquent, il ya toujours un compromis entre throughput et taux d'erreur.

Dans un schéma ARQ, après le décodage d'un paquet, soit on réussit et on décode correctement le paquet, cela est exprimé par une probabilité P_c. Ou le paquet reçu est interprété comme un autre mot de code valide, ce qui entraîne une erreur de bloc, qui est exprimée par une probabilité P_e. Sinon, on choisit de ne pas accepter le paquet si la fiabilité de la décision de décodage est inférieure à un seuil, mais plutôt de demander une retransmission. Il en résulte une certaine probabilité de retransmission, P_{ARQ}. Si P_{ARQ} augmente, par

conséquent $P_e + P_c$ diminue, parce que : $P_{ARQ}^i + P_e^i + P_c^i = 1$. Où i représente la i^{ème} retransmission.

Bien qu'on veuille minimiser la probabilité d'erreur P_e, une P_{ARQ} importante entraînera de nombreuses retransmissions, ce qui donnera un throughput très faible [14].

1.3.2.2. Le Protocole de Retransmission Hybride HARQ :

Un protocole de retransmission ARQ hybride (HARQ), utilise un code correcteur d'erreur en conjonction avec le schéma de retransmission. Par conséquent, il essaie de décoder le mot de code reçu et ne demande qu'une retransmission si l'incertitude de la décision de décodage est considérée comme trop élevée, c'est-à-dire si la détection est inférieure à un certain seuil de fiabilité.

Généralement, un code de contrôle de redondance cyclique CRC (Cyclic Redundancy Check) est utilisé pour déterminer si une retransmission est nécessaire ou non. Cela implique que le paquet d'informations est d'abord codé en utilisant le code CRC. Ensuite, il est envoyé au codeur du code de correction d'erreur qui interprétera la séquence d'informations conjointement avec la redondance ajoutée par le codage CRC comme un bloc d'entrée de grande taille.

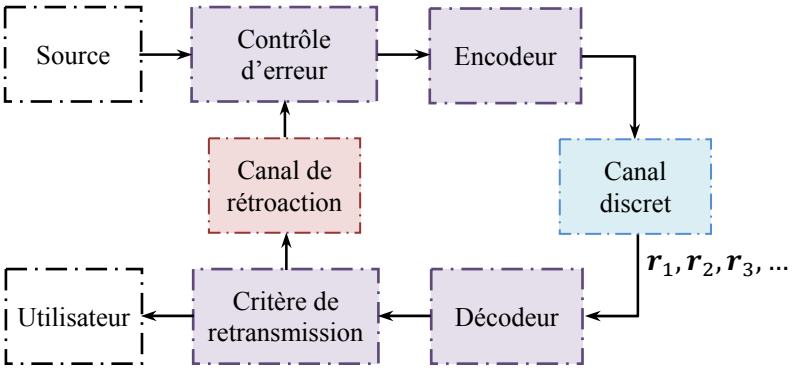

Figure.1.4 : Schéma fonctionnel de base du protocole HARQ

Il existe deux principaux schémas du protocole de retransmission HARQ, notés type-I et type-II. Un protocole HARQ de type I implique que le même message, c'est-à-dire le même contenu de paquet, est envoyé chaque fois que le récepteur demande une retransmission, voir Figure 1.5.

D'autre part, dans un schéma de type II, la première transmission n'inclut habituellement que des bits d'information et certains bits redondants destinés uniquement à la détection

d'erreurs, par ex. un code CRC, comme s'il s'agissait d'un schéma purement ARQ. Si jamais une retransmission est nécessaire, des bits redondants sont envoyés de plus en plus (*Incremental Redundancy*), sauf que cette fois-ci ils sont destinés à la correction d'erreur, rendant à nouveau le schéma de transmission hybride, Figure 1.5.

Dans un sens, le schéma de type II utilise les informations des paquets précédemment reçus. Cela peut aussi être fait dans le schéma de type I par combinaison de paquets (*Packet Combining*) [14].

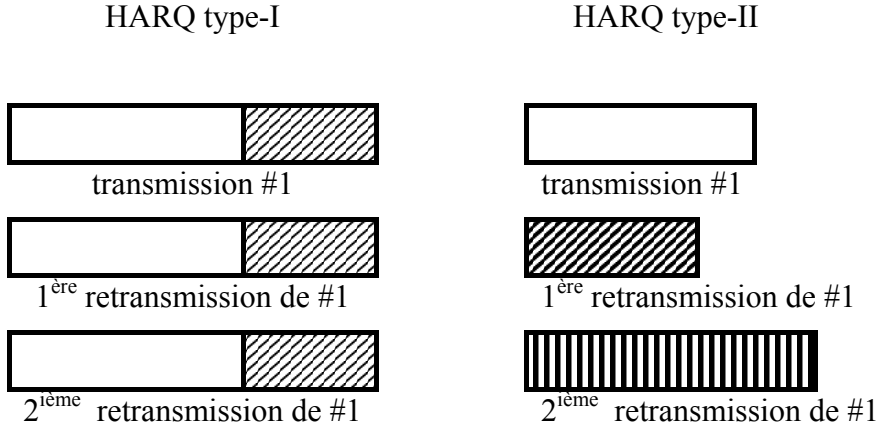

Figure 1.5. Description graphique d'un schéma HARQ de type I et de type II.

Les champs blancs de la figure.1.5 indiquent des bits d'information, tandis que les champs rayés désignent des bits redondants.

1.3.2.3. Codes de Redondance Cyclique CRC :

Pour être efficaces, les codes détecteurs contiennent dans chaque mot de code $\boldsymbol{c}$ les symboles d'information associés $\boldsymbol{u}$. On dit qu'un code (n, k) est *systématique* s'il existe k positions telles que la restriction de tout mot de code $\boldsymbol{c}$ à ces k positions est égale à $\boldsymbol{u}$. Le code est dit aussi *séparable* car on peut extraire directement les symboles d'information $\boldsymbol{u} = [u_{k-1}, \ldots, u_0]$ du mot de code $\boldsymbol{c}$. Souvent, ces symboles d'information sont mis dans l'ordre aux premières positions. Le principe du codage systématique est alors de calculer les $r = n - k$ symboles de redondance $[c_{n-1}, \ldots, c_k]$. La généralisation à un code systématique sur k autres positions est directe [15].

Dans un code systématique, les symboles de redondance sont appelés *symboles de parité* ou *symboles de contrôle*. En effet, la détection d'erreurs se ramène à un *contrôle de parité* (ou *checksum*).

On associe r symboles de redondance à chaque mot $\boldsymbol{u} = [u_{k-1}, \ldots, u_0]$ source tel que le mot de code $\boldsymbol{c} = [u_{k-1}, \ldots, u_0, c_{k+r-1}, \ldots, c_k]$ vérifie un prédicat $f(\boldsymbol{c})$ (la parité) avec f facile à calculer.

Les codes de redondance cyclique, appelés CRC (*Cyclic Redundancy Check*), sont très utilisés dans les réseaux informatiques. Dans ces codes, un mot binaire $\boldsymbol{u} = [u_{k-1}, \ldots, u_0] \in \{0,1\}^k$ est représenté par un polynôme $P_u(X) = \sum_{i=0}^{k-1} u_i X^i$ qui est défini sur le champ de Galois $\mathbb{GF}(2)$. Par exemple, $\boldsymbol{u} = [10110]$ est représenté par le polynôme $P_u = X^4 + X^2 + X$.

Un code CRC est caractérisé par un *polynôme générateur* P_g de degré r : $P_g(X) = X^r + \sum_{i=0}^{r-1} g_i X^i$ dans $\mathbb{GF}(2)$. Le mot source binaire $\boldsymbol{u} = [u_{k-1}, \ldots, u_0]$ associé au polynôme $P_u(X)$, est codé par le mot de code binaire $\boldsymbol{c} = [c_{n-1}, \ldots, c_0] = [u_{k-1}, \ldots, u_0, c_{r-1}, \ldots, c_0]$où $[c_{r-1}, \ldots, c_0]$est la représentation du reste de la division euclidienne de $X^r P_u$ par P_g. La multiplication de P_u par X^r se traduisant par un décalage de bits, le polynôme $P_c(X) = \sum_{i=0}^{n-1} c_i X^i$ vérifie donc [15]:

$$P_c = X^r P_u + \left(X^r P_u \; modulo \; P_g\right) \tag{1.17}$$

Exemple : Avec le polynôme générateur $P_g = X^2 + 1$, le mot $\boldsymbol{u} = [10110]$ est codé en ajoutant les bits de redondance du polynôme :

$$\begin{aligned}(X^r P_u) \; mod \; (P_g) &= \left((X^4 + X^2 + X)X^2\right) mod (X^2 + 1) \\ &= (X^6 + X^4 + X^3) mod (X^2 + 1) \\ &= X\end{aligned}$$

Qui correspond aux bits de redondance $[c_1, c_0] = [10]$. Le mot codé est donc $\boldsymbol{c} = [1011010]$.

Le codage CRC se ramène donc à implémenter le calcul du reste d'une division euclidienne. Cette opération peut être efficacement réalisée par l'implémentation matérielle de l'algorithme standard en utilisant des registres à décalage linéaire LFSR (Voir figure 1.6) (*Linear Feedback Shift Register*) [10,16].

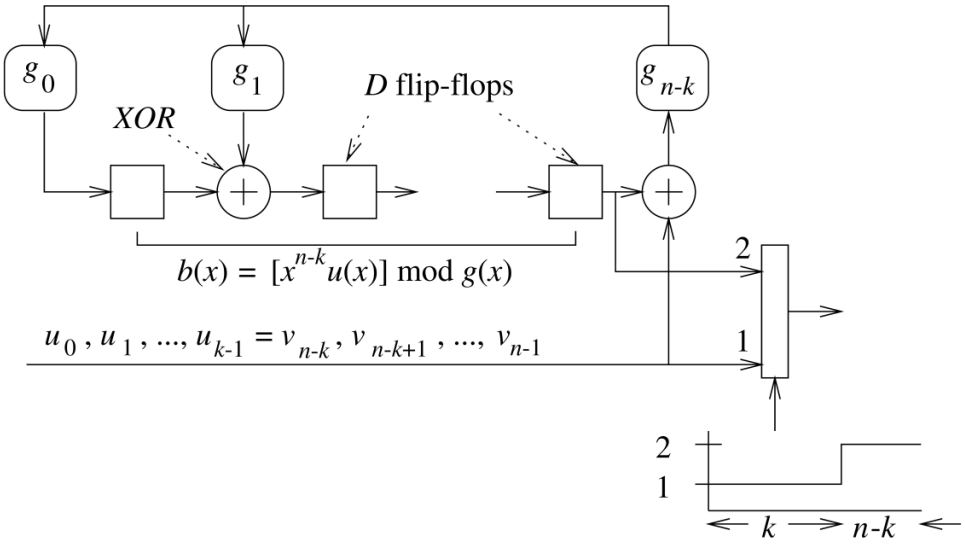

Figure 1.6. Structure général du codeur CRC avec un registre LFSR

Pour le décodage, on remarque que dans $\mathbb{GF}(2)$ on a aussi $P_c = X^r P_u - \left(X^r P_u \; modulo \; P_g\right)$, le polynôme P_c est donc un multiple de P_g. Cette propriété permet un décodage rapide, lui aussi basé sur une division euclidienne de polynômes et réalisable avec un registre LFSR. A la réception de $\boldsymbol{c'} = [c'_{n-1}, \ldots, c'_0]$, il suffit de calculer le reste $R(X)$ du polynôme $P_{c'}$ par le polynôme P_g. Si ce reste est nul, le mot reçu est un mot de code et l'on ne détecte pas d'erreurs. Sinon, si $R(X) \neq 0$, il y a eu erreur lors de la transmission [15].

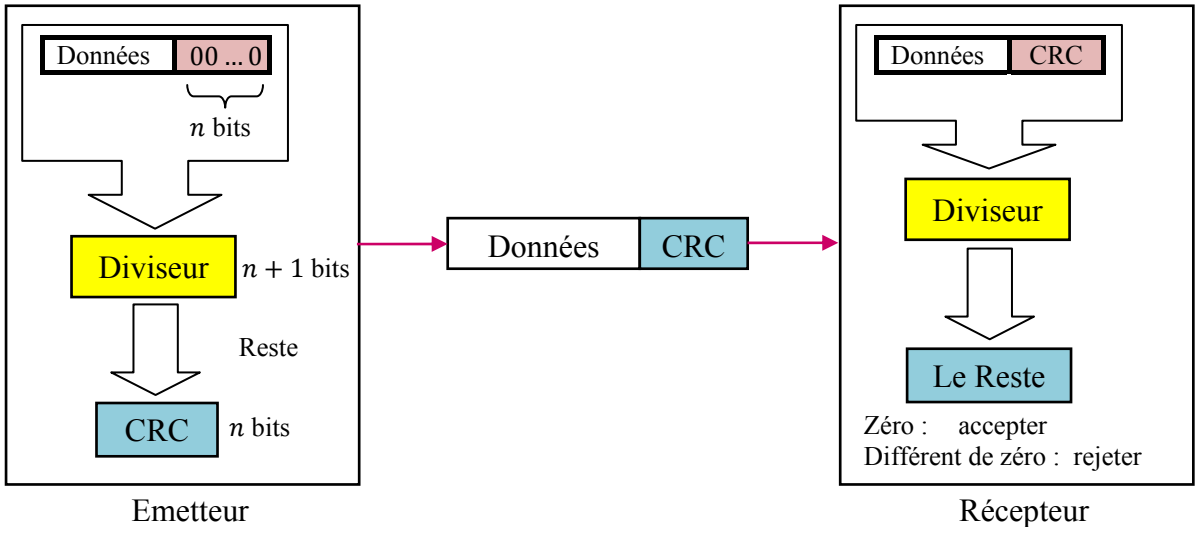

Figure 1.7 Génération des bits de redondance CRC et calcul du syndrome de parité

Les codes CRC sont très utilisés dans les réseaux. Outre leurs bonnes propriétés, ils sont choisis aussi pour l'efficacité du codage et décodage. Le tableau 1.2 donne quelques exemples

normalisés; la normalisation concerne non seulement la valeur du polynôme mais aussi les algorithmes de codage et décodage [15].

Nom	Générateur	Factorisation	Exemples d'utilisation
TCH/FS-HS-EFS	$X^3 + X + 1$	irréductible	GSM transmission de voix
GSM TCH/EFS	$X^8 + X^4 + X^3 + X^2 + 1$	irréductible	GSM pré-codage canal à plein taux
CRC-8	$X^8 + X^7 + X^4 + X^3 + X + 1$	$(X + 1)(X^7 + X^3 + 1)$	GSM 3ème génération
CRC-16 X25-CCITT	$X^{16} + X^{12} + X^5 + 1$	$(X + 1)(X^{15} + X^{14} + X^{13} + X^{12} + X^4 + X^3 + X^2 + X + 1)$	Protocole X25-CCITT ; contrôle trames PPP FCS-16 (RFC-1662)
CRC-24	$X^{24} + X^{23} + X^{18} + X^{17} + X^{14} + X^{11} + X^{10} + X^7 + X^6 + X^5 + X^4 + X^3 + X + 1$	$(X + 1)(X^{23} + X^{17} + X^{13} + X^{12} + X^{11} + X^9 + X^8 + X^7 + X^5 + X^3 + 1)$	Communication UHF et satellites (SATCOM) ; messages OpenPGP (RFC-2440)
CRC-24 (3GPP)	$X^{24} + X^{23} + X^6 + X^5 + 1$	$(X + 1)(X^{23} + X^5 + 1)$	GSM 3ème génération
CRC-32 AUTODIN-II	$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$	irréductible	IEEE-802.3 (Ethernet), ATM AAL5, trames PPP FCS-32 (RFC-1662) ; contrôle d'intégrité des fichiers ZIP et RAR

Tableau 1.2 Exemples de codes CRC normalisés

1.4 Principe de la Correction d'Erreur Directe FEC et le Décodage Itératif :

Nous rappelons quelques généralités sur les codes linéaires avant de détailler les principes du décodage itératif.

1.4.1. Généralités sur les Codes Linéaires :

Soit $\mathbb{F}_q$ le corps à q éléments. Un code *linéaire* $\mathcal{C}$ de longueur n et de dimension k sur $\mathbb{F}_q$ est un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k. Un tel code est noté $(n, k)_q$. Tous les codes que l'on considèrera par la suite sont définis sur le corps à deux éléments, $\mathbb{F}_2$. Ces codes sont dits binaires et seront simplement notés (n, k).

Le *rendement* R d'un code $(n, k)_q$ est la proportion d'information contenue dans un mot de code : $R = \frac{k}{n}$.

En plus de sa longueur et de sa dimension, la capacité de correction d'un code est importante afin d'utiliser un code adapté à la situation considérée.

La *distance minimale* d_{min} d'un code $\boldsymbol{\mathcal{C}}(n,k)_q$ est la plus petite distance, au sens de la distance de Hamming, entre deux mots distincts du code $\boldsymbol{\mathcal{C}}$. Pour les codes linéaires, cette distance est égale au plus petit poids de Hamming des mots non-nuls appartenant au code $\boldsymbol{\mathcal{C}}$.

Soit $\boldsymbol{\mathcal{C}}$ un code $(n,k)_q$ et d_{min} sa distance minimale, la capacité de correction de $\boldsymbol{\mathcal{C}}$ est égale à $\frac{d_{min}-1}{2}$. Cette capacité correspond au nombre d'erreurs que le code est capable de corriger dans tous les cas.

Afin d'associer à tout mot d'information $\boldsymbol{u} = (u_0, \ldots, u_{k-1})$ le mot de code, noté $\boldsymbol{c} = (c_0, \ldots, c_{n-1})$, correspondant, la matrice génératrice du code est utilisée.

Une *matrice génératrice* $\boldsymbol{G}$ d'un code linéaire $\boldsymbol{\mathcal{C}}(n,k)_q$ est une matrice de taille $k \times n$ à coefficients dans $\mathbb{F}_q$. Les lignes de cette matrice forment une base de $\boldsymbol{\mathcal{C}}$. Une méthode de codage des codes linéaires consiste à effectuer le produit $\boldsymbol{uG}$. Quant au décodage, il peut utiliser la matrice de parité.

La *matrice de parité* $\boldsymbol{H}$ d'un code linéaire $\boldsymbol{\mathcal{C}}(n,k)_q$ est une matrice de n colonnes et de rang $n - k$ à coefficients dans $\mathbb{F}_q$ telle que pour tout mot de code $\in \boldsymbol{\mathcal{C}}$, on a : $\boldsymbol{Hc}^T = \boldsymbol{0}$. Le code engendré par la matrice $\boldsymbol{H}$ est un code $(n, n-k)_q$ appelé code dual de $\boldsymbol{\mathcal{C}}$, il est noté $\boldsymbol{\mathcal{C}}^{\perp}$. Par définition, une matrice de parité $\boldsymbol{H}$ d'un code linéaire $\boldsymbol{\mathcal{C}}$ engendré par la matrice génératrice $\boldsymbol{G}$ est telle que $\boldsymbol{GH}^T = \boldsymbol{0}$. La forme de la matrice $\boldsymbol{G}$ peut être utilisée pour déterminer $\boldsymbol{H}$.

La matrice génératrice $\boldsymbol{G}$ est dite *systématique* si elle est de la forme $(\boldsymbol{I}_k|\boldsymbol{A})$, où $\boldsymbol{I}_k$ représente la matrice identité de taille $k \times k$. Un codage effectué par une telle matrice est dit systématique [17].

Dans un codage systématique, le mot de code associé au mot d'information $\boldsymbol{u}$ contient $\boldsymbol{u}$ sur ses k premiers indices. La matrice de parité $\boldsymbol{H}$ se déduit de la matrice génératrice systématique $\boldsymbol{G} = (\boldsymbol{I}_k|\boldsymbol{A})$ par $\boldsymbol{H} = (\boldsymbol{A}^T|\boldsymbol{I}_{n-k})$.

1.4.2. Rapport de Vraisemblance Logarithmique :

Soit U une variable aléatoire dans $\mathbb{GF}(2)$ d'éléments binaires $\{0,1\}$. Considérons une transmission non-codée utilisant la modulation BPSK (*Binary Phase Shift Keying*) sur le canal AWGN. Un bit d'information $u = 0$ est transmis comme $x = +1$ et un bit d'information $u = 1$ est transmis comme $x = -1$. L'observation à la sortie du canal bruité est donnée par $y = x + \eta$, où η représente l'échantillon de bruit ayant une distribution gaussienne centrée

d'écart-type σ. Soit $P(u=0)$ (respectivement $P(u=1)$) la probabilité a priori que le digit émis soit un 0 (respectivement un 1).

Le rapport de vraisemblance logarithmique ou *Log-Likelihood Ratio* (LLR) de la variable aléatoire binaire U, noté $L(u)$ et abrégé *L-value*, est défini par [18]:

$$L(u) = \ln \frac{P(u=0)}{P(u=1)} \tag{1.18}$$

$P(x)$ est la probabilité que la variable X prend la valeur x qui peut être calculer par :

$$P(x = \pm 1) = \left(\frac{e^{-\frac{L(x)}{2}}}{1 + e^{-L(x)}} \right) e^{\frac{xL(x)}{2}} \tag{1.19}$$

Les probabilités a posteriori ou vraisemblances sont calculées à partir du théorème de Bayes comme suit :

$$\begin{aligned} P(u=0|y) &= \frac{P(y|u=0)P(u=0)}{P(y)} \\ P(u=1|y) &= \frac{P(y|u=1)P(u=1)}{P(y)} \end{aligned} \tag{1.20}$$

Elles peuvent être considérées comme un raffinement de la connaissance a priori sur la valeur du digit transmis fourni par l'observation du canal. La décision ferme optimale $\hat{u}$ au sens du maximum a posteriori (MAP) est alors la suivante :

$$\hat{u} = \begin{cases} 0 & si\ P(u=0|y) > P(u=1|y) \\ 1 & sinon \end{cases} \tag{1.21}$$

La décision souple est définie par le rapport de vraisemblance logarithmique :

$$L(u|y) = \ln \frac{P(u=0|y)}{P(u=1|y)} \tag{1.22}$$

En utilisant l'équation (1.20), la décision souple peut être décomposée en :

$$L(u|y) = L_c(y) + L_a(u) \tag{1.23}$$

Où $L_c(y) = \ln \frac{P(y|u=0)}{P(y|u=1)}$ est le rapport de vraisemblance logarithmique du canal et $L_a(u) = \ln \frac{P(u=0)}{P(u=1)}$ est le rapport de vraisemblance logarithmique a priori. Le critère de décision au sens du MAP peut alors s'écrire :

$$\hat{u} = \begin{cases} 0 & si\ \ L(u|y) > 0 \\ 1 & sinon \end{cases} \tag{1.24}$$

Il s'ensuit que le signe de la décision souple détermine la décision ferme et que la valeur absolue de la décision souple détermine la fiabilité de cette décision. En particulier, avec le bruit gaussien, le rapport de vraisemblance logarithmique du canal a pour expression :

$$L_c(y) = \ln \frac{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-1)^2}{2\sigma^2}}}{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y+1)^2}{2\sigma^2}}} = \frac{2}{\sigma^2} y \tag{1.25}$$

Pour un système de transmission codé, nous montrerons dans le chapitre 3 que si l'information a priori est distribuée de manière indépendante, la sortie souple d'un décodeur peut s'écrire sous la forme :

$$L(u|y) = L_c(y) + L_a(u) + L_e(u) \tag{1.26}$$

Où $L_e(u)$ est le rapport de vraisemblance logarithmique *extrinsèque* représentant la connaissance acquise grâce au processus de décodage [18].

1.4.3. Décodage Itératif :

Considérons le code à deux dimensions illustré par la Figure.1.8. Les données u sont réparties dans un tableau à k_1 colonnes et k_2 lignes. Un code horizontal génère le bloc de parité noté p^- et un code vertical génère le bloc de parité noté $p^|$. Chaque mot de code horizontal et vertical, est de longueur n_1 et n_2 respectivement, et correspond à k_1 et k_2 bits informatifs [18].

Soit L_e^- , $L_e^|$ les rapports de vraisemblance logarithmiques extrinsèques obtenus à partir d'un décodage MAP horizontal et vertical respectivement. Nous supposerons que les données sont équiprobables, donc $L_a(u) = 0$ avant qu'un décodage n'aie eu lieu. L'algorithme de décodage itératif de ce code se déroule de la manière suivante :

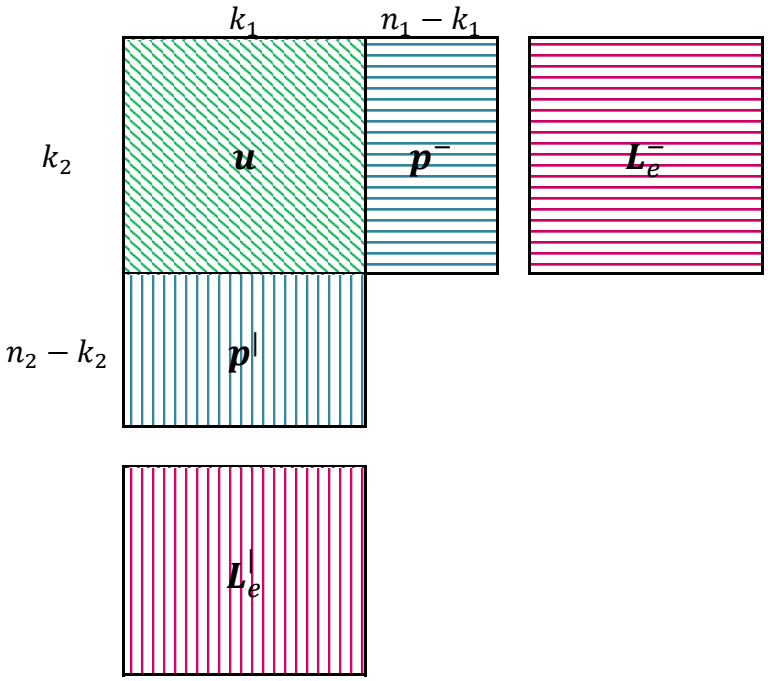

Figure 1.8. Schéma de code à deux dimensions pour décodage itératif

1) Initialiser l'information a priori à $L_a(u) = 0$.
2) Décoder horizontalement et produire l'information extrinsèque comme suit :

$$L_e^{-}(\hat{u}) = L(\hat{u}) - L_c(y) - L_a(u) \tag{1.27}$$

3) Choisir $L_a(u) = L_e^{-}(\hat{u})$.
4) Décoder verticalement et produire l'information extrinsèque comme suit :

$$L_e^{|}(\hat{u}) = L(\hat{u}) - L_c(y) - L_a(u) \tag{1.28}$$

5) Choisir $L_a(u) = L_e^{|}(\hat{u})$.
6) Si suffisamment d'itérations se sont produites pour conduire à une décision ferme fiable, aller à 7; sinon retourner à 2.
7) Prendre des décisions fermes sur les données à partir du signe de :

$$L(\hat{u}) = L_c(y) + L_e^{-}(\hat{u}) + L_e^{|}(\hat{u}) \tag{1.29}$$

Notons que les décodeurs constitutifs échangent seulement le terme extrinsèque parce qu'il correspond à la nouvelle information acquise durant l'étape de décodage courante. Il est commode d'interpréter l'information extrinsèque comme de la diversité qui améliore les décisions souples à chaque étape de décodage [18].

Chapitre **2**

Codes de Contrôle de Parité à Faible Densité LDPC

2.1. Introduction :

Les codes de contrôle de parité à faible densité (Low Density Parity Check), sont une classe de codes correcteurs d'erreurs et ont étés proposés par Gallager [8] dans sa thèse de doctorat en 1962, soit douze ans après l'introduction des codes de correction d'erreurs par Hamming dans son article publié en 1950 [2].

Les codes de Hamming et les codes LDPC sont des codes en blocs : les messages sont divisés en blocs à coder par l'émetteur et décodés de manière similaire en tant que blocs séparés par le récepteur. Les codes de Hamming sont courts et très structurés avec une capacité de correction d'erreur fixe connue. Alors que les codes LDPC sont le contraire, généralement long et souvent construit pseudo-aléatoirement avec seulement une notion probabiliste de leur performance de correction d'erreur attendue.

Ce deuxième chapitre introduit les codes Low Density Parity Check (LDPC). Tout d'abord les principales définitions et quelques notations seront introduites. Dans une deuxième partie la représentation graphique des codes LDPC par le graphe de Tanner, et plus particulièrement la taille des cycles et les propriétés de la distance minimale seront brièvement décrites. La troisième partie de ce chapitre présentera leur principe de construction, d'optimisation, et quelques méthodes de construction des plus répandues dans la littérature. La méthode d'encodage à complexité linéaire est discutée, suivie par l'introduction des algorithmes de décodage itératifs à l'aide d'un algorithme de décision ferme (*bit-flipping*), de sorte que le sujet est développé d'abord sans référence à la théorie des probabilités. Par la suite, l'algorithme de décodage à entrée-sortie souple (*sum-product*) est présenté.

2.2. Définition et propriétés :

Dans les communications numériques, un mot de données est une séquence d'utilisateur de k bits. Un code de bloc linéaire $\boldsymbol{\mathcal{C}}(n,k)$ avec longueur de mot de données k et longueur de mot de code (bloc) n sur le champ binaire $\mathbb{GF}(2)$ peut être considéré comme un sous-espace de dimension-k de l'espace vectoriel $\mathcal{V}_n$ de dimension-n sur le champ binaire $\mathbb{GF}(2)$.

Il existe 2^k mots de données $\boldsymbol{u} = [u_0, u_1, \dots, u_{k-1}]$ dans le code $\boldsymbol{\mathcal{C}}$, et chacun d'eux correspond à un mot de code unique $\boldsymbol{c} = [c_0, c_1, \dots, c_{n-1}]$. Notons que $\boldsymbol{u}$ et $\boldsymbol{c}$ sont des vecteurs lignes, et non pas des colonnes. Comme $\boldsymbol{\mathcal{C}}$ est un sous-espace de dimension k, il existe k vecteurs linéairement indépendants $\boldsymbol{g}_0, \boldsymbol{g}_1, \dots, \boldsymbol{g}_{k-1}$ qui engendre $\boldsymbol{\mathcal{C}}$. Par conséquent, chaque mot de code peut être écrit comme une combinaison linéaire de ces vecteurs indépendants :

$$\boldsymbol{c} = \boldsymbol{uG} \tag{2.1}$$

Où $\boldsymbol{G} = [\boldsymbol{g}_0, \boldsymbol{g}_1, \dots, \boldsymbol{g}_{k-1}]^T$ est la matrice génératrice de dimension $(k \times n)$. L'espace nul du sous-espace $\boldsymbol{\mathcal{C}}$, c'est-à-dire le sous-espace orthogonal de $\boldsymbol{\mathcal{C}}$ de dimension $m = n - k$, peut être engendré par m vecteurs linéairement indépendants $\boldsymbol{h}_0, \boldsymbol{h}_1, \dots, \boldsymbol{h}_{m-1}$. Chaque mot de code $\boldsymbol{c}$ vérifie les équations de contrôle de parité $\forall i$: $\boldsymbol{c} \cdot \boldsymbol{h}_i^T = 0$. La matrice de vérification de parité $\boldsymbol{H}$ de dimension $(m \times n)$ rassemble ces $\boldsymbol{h}_i$ en format matriciel comme $\boldsymbol{H} = [\boldsymbol{h}_0, \boldsymbol{h}_1, \dots, \boldsymbol{h}_{m-1}]^T$ et donc tous les mots de codes satisfont à l'équation [19] :

$$\boldsymbol{c} \cdot \boldsymbol{H}^T = 0 \tag{2.2}$$

Un code de contrôle de parité de faible densité (LDPC) est un code en bloc binaire linéaire long, dont la matrice de vérification de parité $\boldsymbol{H}$ a une faible densité de 1s. Plus précisément, $\boldsymbol{H}$ est une matrice creuse et contient un petit nombre fixe w_c de '1's dans chaque colonne et un petit nombre fixe w_r de '1's dans chaque ligne. Ce caractère creux (*sparsity*) de la matrice de parité rend la complexité du décodage faible et mène à des implémentations simples. Si la longueur du mot de code est n, on dit que $\boldsymbol{H}$ caractérise un code LDPC (n, w_c, w_r). Ces codes peuvent être appelés codes LDPC *réguliers* pour les distinguer des codes irréguliers dont les valeurs de w_c et w_r ne sont pas constantes. La matrice $\boldsymbol{H}$ d'un code *irrégulier* a approximativement w_r 1s dans chaque ligne et w_c 1s dans chaque colonne.

En comptant le nombre de 1 dans $\boldsymbol{H}$, il en résulte que $nw_c = mw_r$. Le taux du code défini par $\boldsymbol{H}$ est [20]:

$$R = \frac{k}{n} = \frac{n-m}{n} = 1 - \frac{w_c}{w_r} \quad (2.3)$$

Notons que l'équation (3) n'est vraie que si la matrice $\boldsymbol{H}$ est de rang complet (*full rank*). Si $\boldsymbol{H}$ n'est pas de rang complet, le taux du code est $R = \frac{n-rang(\boldsymbol{H})}{n}$.

2.3. Représentation par le Graphe de Tanner :

On peut associer au code LDPC, un graphe appelé le graphe de Tanner. Les mots $\boldsymbol{c}$ du code défini par $\boldsymbol{H}$ sont les mots binaires dont les n bits vérifient simultanément les m équations de parité. Ce système d'équations linéaires est représenté graphiquement dans la figure 2.1.

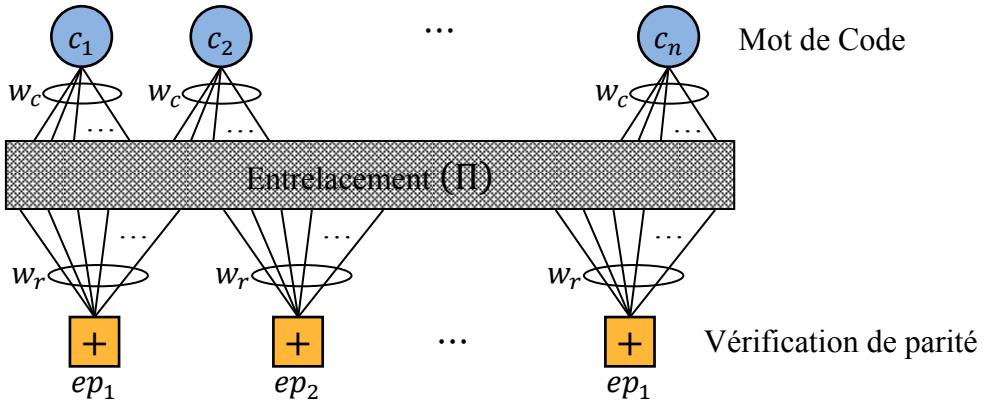

Figure 2.1. Graphe bipartite de Tanner d'un code LDPC régulier

Une telle représentation est appelée *graphe bipartite* du code. Dans ce graphe, des branches sont reliées à deux nœuds appartenant à deux classes différentes :

- La première classe de nœuds appelés *nœuds de variable*, correspond aux bits des mots de code ($c_j,\ j \in \{1, \dots, n\}$), et donc aux colonnes de $\boldsymbol{H}$.
- La seconde classe de nœuds, appelés *nœuds de parité*, correspond aux équations de parité ($ep_i, i \in \{1, \dots, m\}$), et donc aux lignes de $\boldsymbol{H}$.

Ainsi, à chaque branche reliant un nœud de variable c_j à un nœud de parité ep_i correspond le '1' qui se situe à l'intersection de la j-ième colonne et de la i-ième ligne de la matrice de contrôle de parité $\boldsymbol{H}$ [21].

Un *cycle* sur un graphe de Tanner est un chemin sur le graphe qui permet de partir d'un nœud et de revenir à ce même nœud sans passer par la même branche, et qui contient d'autres nœuds

pas plus d'une seule fois. La taille d'un cycle est donnée par le nombre de branches qu'il contient. Le graphe étant bipartite, la taille des cycles est paire.

En anglais, la taille du cycle le plus court g dans un graphe est appelée '*girth*'. La présence de cycles dans le graphe pourra dégrader les performances de décodage par un phénomène d'auto-confirmation lors de la propagation des messages [21].

Les cycles, en particulier les cycles courts dans le graphe de Tanner, conduisent à un décodage inefficace et empêchent l'algorithme de propagation de croyance de converger vers le résultat de décodage optimal, taxant ainsi la performance des décodeurs LDPC.

Gallager [8] a montré que le nombre d'itérations indépendantes au décodage est proportionnel au girth g du graphe de Tanner. Une seconde raison qui affecte la performance du décodeur LDPC concerne la distance minimale du code. Tanner [22] obtint une limite inférieure sur la distance minimale d_{min}. Cette limite inférieure est proportionnelle avec le girth g du code ; par conséquent, les codes LDPC à grand girth sont préférés.

Pour augmenter la valeur de g et éviter les cycles courts, la matrice de parité H doit être suffisamment creuse. Cela signifie que la longueur du mot de code n doit être suffisamment grande. En fait, Gallager fournit une limite inférieure vague sur la longueur de bloc n pour une valeur de g donnée d'un code LDPC régulier. Bien sûr, cette limite inférieure est juste que, même si n dépasse la limite pour un g donné, il pourrait ne pas exister un code LDPC régulier avec cette longueur de mot de code et la valeur du girth voulue. Pour une matrice de parité $\boldsymbol{H}$ avec poids de colonne w_c et poids de ligne w_r, la limite sur la longueur de bloc n est [19] :

$$n \geq \begin{cases} 1 + \sum_{2}^{j+1} w_c(w_c-1)^{i-2}(w_r-1)^{i-1} & si: g = 4j+2 \\ \sum_{1}^{j} w_r(w_c-1)^{i-1}(w_r-1)^{i-1} & si: g = 4j \end{cases} \tag{2.4}$$

Par exemple, pour construire un code LDPC $(n, 3, 12)$ avec $g = 12$, la longueur de ses mots de codes n devrait satisfaire $n \geq 6084$.

2.4. Construction de codes LDPC :

La performance des codes LDPC dépend en grande partie de la structure du code, c'est-à-dire de la construction de la matrice de parité $\boldsymbol{H}$. Le problème de construction de code LDPC est un problème d'optimisation qui s'effectue généralement en trois étapes [21] :

- Optimisation *a priori* des profils d'irrégularité des nœuds de parité et de variable.
- Construction de matrices $\boldsymbol{H}$ de taille adéquate respectant les profils d'irrégularité et maximisant la longueur des cycles (maximum g possible).
- Eventuellement, choix ou rejet des codes sur le critère de la distance minimale ou sur les performances calculées par simulation.

Un grand nombre de techniques de conception sont proposées dans la littérature sous des critères de conception différentes : Spécifiquement conçus pour les codes LDPC régulier ou irrégulier, performances proche de la capacité de Shannon, encodage et décodage efficaces, un bas niveau de plancher d'erreurs (error floor), ressources mémoire réduites, … etc. Ces techniques de constructions peuvent être classées sous deux rubriques principales:

- *Constructions Aléatoires* : elles sont basées sur la génération d'une matrice de parité remplie aléatoirement par 0 et 1, et telle que les propriétés LDPC sont satisfaites. En particulier, après avoir choisi les paramètres n, R et w_c pour les codes réguliers, les poids des lignes et des colonnes de $\boldsymbol{H}$ doivent être exactement w_r et w_c , respectivement, et qui soit petits par rapport au nombre de colonnes et lignes. Des contraintes supplémentaires peuvent être incluses : par exemple, le nombre de 1 en commun entre deux colonnes quelconques (ou deux lignes) ne doit pas dépasser un (cette contrainte empêche les cycles court de 4 branches où $g = 4$).
  En général, les codes construits aléatoirement sont bons si n est assez grand, mais leur performance peut ne pas être satisfaisante pour les valeurs modérés de n. De plus, ils ne sont généralement pas assez structurés pour permettre un encodage simple.
- *Constructions Algébriques* : ces méthodes produisent des codes LDPC pouvant se prêter à un décodage plus facile que les codes de construction aléatoire. De plus, pour des valeurs de n modérés, la probabilité d'erreur de codes algébriques bien conçus peut être plus faible.

Dans cette section, nous présentons quelques-unes des plus populaires et importantes.

2.4.1. Codes de Gallager :

La construction originale utilisée par Gallager en 1963 est une méthode pour la construction aléatoire de $\boldsymbol{H}$ à partir des contrainte (n, w_c, w_r) d'un code LDPC régulier.

La construction de la matrice de contrôle de parité $\boldsymbol{H}$ d'un code Gallager (n, w_c, w_r) est réalisée par les étapes suivantes [23] :

1) $\boldsymbol{H}$ est constituée de w_c sous-matrices $\boldsymbol{H}_i$ tel que $i = 1, \cdots, w_c$:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_2 \\ \vdots \\ \boldsymbol{H}_{w_c} \end{bmatrix} \tag{2.5}$$

2) $\boldsymbol{H}_i$ une sous-matrice de dimension $n/w_c \times n$ qui contient un seul '1' dans chaque colonne et w_r '1' dans chaque ligne.
3) Les w_r '1' de la i^{ème} ligne de $\boldsymbol{H}_1$ se trouvent dans les colonnes $(i-1)w_r + 1$ jusqu'à iw_r . C'est-à-dire que chaque ligne est obtenue par une rotation de la ligne immédiatement précédente par w_r positions vers la droite.
4) Les autres $w_c - 1$ sous-matrices soit $\boldsymbol{H}_2, \boldsymbol{H}_3, \cdots, \boldsymbol{H}_{w_c}$ sont des versions permutées pseudo-aléatoirement de $\boldsymbol{H}_1$, et sont obtenues par permutation aléatoire (avec des probabilités égales) des colonnes de $\boldsymbol{H}_1$.

Quoique la construction des codes LDPC de Gallager est simple et donne de bonnes performances, cette méthode n'a pas de structure pour un encodage rapide. Le taux du code de conception peut ne pas être le taux réel, car $\boldsymbol{H}$ n'est pas forcement de rang complet. Aussi aucune garantie que les petits cycles ne soient pas présents dans le graph de Tanner.

2.4.2. Codes Quasi-Cyclique (QC- LDPC) :

Une autre famille de codes permet un encodage simple : ces codes font partie de la famille des codes Quasi-Cyclique (QC). Ils peuvent être simplement encodés en utilisant des registres à décalage LFSR avec une complexité linéairement proportionnelle au nombre de bits de parité pour le codage en série et à la longueur de code pour le codage parallèle. Ils ont également, en raison de leur symétrie cyclique, des avantages dans la mise en œuvre.

Les codes QC-LDPC ont une matrice de contrôle de parité qui se compose de blocs carrés qui sont soit des circulants de rang complet, soit des matrices nulles.

Un circulant est une matrice carrée dans laquelle chaque ligne est le décalage cyclique (rotation vers la droite) de la ligne au-dessus, et la première ligne est le décalage cyclique de la dernière ligne. Chaque colonne d'un circulant est le décalage cyclique descendant de la

colonne à sa gauche et la première colonne est le décalage cyclique de la dernière colonne. Ainsi, un circulant est entièrement caractérisé par sa première ligne ou colonne, qui est appelée générateur du circulant.

Pour les codes QC-LDPC, un circulant $\boldsymbol{P}$ de dimension $L \times L$ sur $\mathbb{GF}(2)$ est généralement fait pour être de rang complet et ses éléments sont exprimés par :

$$p_{i,j} = \begin{cases} 1, & si\ i+1 \equiv j\ mod\ L \\ 0, & autrement \end{cases} \tag{2.6}$$

Notons que $\boldsymbol{P}^i$ est la matrice de permutation du circulant qui déplace la matrice d'identité $\boldsymbol{I}$ vers la droite par i fois pour tout entier i, $0 \leq i \leq L$. Soit la matrice nulle de dimension $L \times L$ que l'on notera par $\boldsymbol{P}^\infty$ pour des raisons de simplicité. Par exemple, $\boldsymbol{P}^1 = \boldsymbol{P}$ est donné par :

$$\boldsymbol{P} = \begin{bmatrix} 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & & 1 \\ 1 & 0 & 0 & & 0 \end{bmatrix} \tag{2.7}$$

Soit $\boldsymbol{H}_{QC}$ la matrice de contrôle de parité $w_c L \times w_r L$ construite comme suit :

$$\boldsymbol{H}_{QC} = \begin{bmatrix} \boldsymbol{P}^{\alpha_{11}} & \boldsymbol{P}^{\alpha_{12}} & \cdots & \boldsymbol{P}^{\alpha_{1(w_r-1)}} & \boldsymbol{P}^{\alpha_{1w_r}} \\ \boldsymbol{P}^{\alpha_{21}} & \boldsymbol{P}^{\alpha_{22}} & \cdots & \boldsymbol{P}^{\alpha_{2(w_r-1)}} & \boldsymbol{P}^{\alpha_{2w_r}} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \boldsymbol{P}^{\alpha_{w_c1}} & \boldsymbol{P}^{\alpha_{w_c2}} & \cdots & \boldsymbol{P}^{\alpha_{w_c(w_r-1)}} & \boldsymbol{P}^{\alpha_{w_cw_r}} \end{bmatrix} \tag{2.8}$$

Où : $\alpha_{ij} \in \{0, 1, \cdots, L-1, \infty\}$.

Le code $\boldsymbol{\mathcal{C}}$, issu de la matrice de contrôle de parité $\boldsymbol{H}_{QC}$ définissant un QC-LDPC, est quasi-cyclique dans le sens où : $\boldsymbol{c} = (\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{w_r-1}) \in \boldsymbol{\mathcal{C}}$ implique que $(T^i\boldsymbol{c}_0, T^i\boldsymbol{c}_1, \ldots, T^i\boldsymbol{c}_{w_r-1}) \in \boldsymbol{\mathcal{C}}$ pour tout i, $0 \leq i \leq L-1$, où $T^i\boldsymbol{c}_l \equiv (c_{l,i}, c_{l,i\oplus 1}, \ldots, c_{l,i\oplus L-1})$, ici le symbole $\oplus$ désigne l'addition modulo-L.

Dans les codes QC-LDPC, si les emplacements des 1s dans la première ligne de l'i^{ème} rangée de blocs $\boldsymbol{H}_i \equiv [\boldsymbol{P}^{\alpha_{i1}}, \boldsymbol{P}^{\alpha_{i2}}, \cdots, \boldsymbol{P}^{\alpha_{iw_r}}]$ sont donnés, les emplacements des autres 1s dans $\boldsymbol{H}_i$ sont déterminés de façon unique. Ainsi, la mémoire requise pour stocker la matrice de contrôle de parité $\boldsymbol{H}_{QC}$ du code QC-LDPC peut être réduite d'un facteur $1/L$, par rapport aux codes LDPC construits de façon aléatoire.

Le code QC-LDPC peut être régulier ou irrégulier en fonction du choix des α_{ij} de $\boldsymbol{H}_{QC}$. Lorsque $\boldsymbol{H}_{QC}$ ne contient pas de sous-matrice nulle, c.à.d. $\alpha_{ij} \neq \infty$, c'est un code LDPC régulier avec poids de colonne w_c et poids de ligne w_r. Sinon, il s'agit d'un code LDPC irrégulier [23].

2.4.3. Les 'Array codes' LDPC :

Les 'Array codes' LDPC sont des codes LDPC structurés basés sur des 'Array codes' qui sont des codes bidimensionnels proposés pour détecter et corriger des erreurs de type *Burst error*.

Les 'Array codes' sont proposés par Fan dans [24]. Ils peuvent être considérés comme des codes QC-LDPC réguliers. Un 'Array code' LDPC est construit par des sous-matrices de permutation ou circulants $\boldsymbol{P}^i$ de dimension $q \times q$. Pour un nombre premier q et des entiers positifs $j \leq k \leq q$, la matrice de contrôle de parité du 'Array code' LDPC régulier est définie par:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{P}^1 & \cdots & \boldsymbol{P}^{j-1} & \cdots & \boldsymbol{P}^{k-1} \\ \boldsymbol{I} & \boldsymbol{P}^2 & \cdots & \boldsymbol{P}^{2(j-1)} & \cdots & \boldsymbol{P}^{2(k-1)} \\ \vdots & \vdots & & \vdots & & \vdots \\ \boldsymbol{I} & \boldsymbol{P}^{j-1} & \cdots & \boldsymbol{P}^{(j-1)(j-1)} & \cdots & \boldsymbol{P}^{(j-1)(k-1)} \end{bmatrix} \tag{2.9}$$

Ainsi, le 'Array code' LDPC est un code QC-LDPC régulier avec $L = q$, la taille du mot de code $n = kq$ et $m = jq$ où les poids de colonne et de ligne du 'Array code' LDPC sont respectivement $w_c = j$ et $w_r = k$. Les matrices de contrôle de parité issues de cette construction ont toujours un girth $g \geq 6$ pour $j \geq 3$ [24].

Pour un encodage efficace des 'Array codes' LDPC, une matrice de contrôle de parité modifiée définissant un code irrégulier, a été proposée sous la forme suivante :

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{P}^1 & \cdots & \boldsymbol{P}^{j-2} & \cdots & \boldsymbol{P}^{k-2} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{P}^{2(j-3)} & \cdots & \boldsymbol{P}^{2(k-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} & \cdots & \boldsymbol{P}^{(j-1)(k-j)} \end{bmatrix} \tag{2.10}$$

En raison de la forme triangulaire supérieure de $\boldsymbol{H}$, l'encodage peut être réalisé efficacement, c'est-à-dire que la complexité de l'encodage augmente linéaire avec la longueur du mot de code.

Un avantage particulier en conséquence de la structure de $\boldsymbol{H}$, c'est qu'il n'y a pas de cycle de longueur 4 dans le graphe de Tanner correspondant. Ainsi, les 'Array codes' LDPC modifiés ont des niveaux de plancher d'erreur (error floor) très bas [23].

2.4.4. Construction de MacKay-Neal :

Une autre méthode de construction pseudo-aléatoire très commune pour les codes LDPC a été proposée par MacKay et Neal. Dans cette méthode, les codes LDPC peuvent être construits par un logiciel de recherche informatique d'une manière pseudo-aléatoire sur la base d'un ensemble de directives satisfaisant aux conditions données dans la section 2.2. Cette construction résulte en un ensemble de codes aléatoires qui se sont révélés contenir de bons codes LDPC.

Supposons qu'il soit souhaitable de construire un code LDPC de longueur n avec un taux $R = k/n$. Afin de construire une matrice de contrôle de parité pour ce code désiré, nous avons besoin de choisir un poids de colonne approprié w_c et un nombre approprié m de lignes. Il est clair que m doit être au moins égal à $n - k$, le nombre de symboles de contrôle de parité du code désiré. Dans la construction sur ordinateur, m est habituellement choisi égal à $n - k$. Pour que $\boldsymbol{H}$ ait un poids de ligne constant w_r, la condition suivante [14] doit être maintenue. Sinon, $\boldsymbol{H}$ ne peut pas avoir de poids constant.

$$w_c n = w_r(n - k) \tag{2.11}$$

Dans ce cas, nous essayons simplement de garder tous les poids des lignes proches de w_r. Si n est divisible par $n - k$, il résulte de l'équation (2.11) que w_r est un multiple de w_c; C'est-à-dire $w_r = w_c n/(n - k)$. Dans ce cas, on peut construire une matrice LDPC régulière avec poids de colonnes w_c et poids de lignes w_r. Si n n'est pas divisible par $n - k$, on divise $w_c n$ par $n - k$ et on obtient :

$$w_c n = w_r(n - k) + b \tag{2.12}$$

Où w_r et b sont le quotient et le reste, respectivement, avec $0 < b < n - k$. L'équation (2.12) peut être réarrangée comme suit:

$$w_c n = w_r(n - k - b) + (w_r + 1)b \tag{2.13}$$

Ce qui suggère que l'on peut construire une matrice de parité $\boldsymbol{H}$ avec deux poids de rangées, w_r et $w_r + 1$, respectivement. Pour des raisons de commodité, $\boldsymbol{H}$ est construite de

telle sorte que ses b-lignes supérieures ont un poids $w_r + 1$, et que ses $(n - k - b)$-lignes inférieures ont un poids w_r.

La construction de $\boldsymbol{H}$ est effectuée étape par étape. A chaque étape, une colonne est ajoutée à une matrice partiellement formée. Chaque colonne ajoutée doit satisfaire à certaines contraintes. Pour $1 \leq i \leq n$, au niveau de la ième étape, on choisit une colonne binaire de $(n - k)$-uplet de poids w_c comme colonne candidate $\boldsymbol{h}_i$ et on l'ajoute à la matrice de contrôle de parité partielle :

$$\boldsymbol{H}_{i-1} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_{i-1}] \tag{2.14}$$

obtenue à l'étape $(i - 1)$ pour former la matrice de parité suivante $\boldsymbol{H}_i$.

Pour $i = 1$, $\boldsymbol{H}_0$ est simplement la matrice nulle. Pour ajouter la colonne $\boldsymbol{h}_i$ à $\boldsymbol{H}_{i-1}$, les contraintes suivantes doivent être satisfaites [14]:

1. Choisir $\boldsymbol{h}_i$ au hasard parmi les $(n - k)$-uplets binaires restants qui ne sont pas utilisés dans $\boldsymbol{H}_{i-1}$ et qui n'ont pas été rejetés précédemment.
2. Vérifier si $\boldsymbol{h}_i$ a plus d'un composant '1' en commun avec n'importe quelle colonne dans $\boldsymbol{H}_{i-1}$. Sinon, passer à l'étape 3; autrement, rejeter $\boldsymbol{h}_i$ et retourner à l'étape 1 pour choisir une autre colonne candidate.
3. Ajouter $\boldsymbol{h}_i$ à $\boldsymbol{H}_{i-1}$ pour former une matrice de parité partielle $\boldsymbol{H}_i$ temporaire. Vérifier les poids des lignes de $\boldsymbol{H}_i$. Si toutes les b-lignes supérieures de $\boldsymbol{H}_i$ ont des poids inférieurs ou égaux à $w_r + 1$ et que toutes les $(n - k - b)$-lignes inférieures de $\boldsymbol{H}_i$ ont des poids inférieurs ou égaux à w_r, alors ajouter définitivement $\boldsymbol{h}_i$ à $\boldsymbol{H}_{i-1}$ pour former $\boldsymbol{H}_i$ et passer à l'étape 1 pour poursuivre le processus de construction. Si l'une des b-lignes supérieures de $\boldsymbol{H}_i$ a un poids dépassant $w_r + 1$, ou que l'une des $(n - k - b)$-lignes inférieures de $\boldsymbol{H}_i$ a un poids supérieur à w_r, rejetez $\boldsymbol{h}_i$ et passez à l'étape 1 pour choisir une autre colonne candidate.

Le processus de construction pas à pas se poursuit jusqu'à ce qu'une matrice de vérification de parité $\boldsymbol{H}$ avec n colonnes soit formée. Si $b = 0$, $\boldsymbol{H}$ est une matrice régulière avec des poids de lignes et de colonnes w_r et w_c, respectivement. Si $b \neq 0$, alors $\boldsymbol{H}$ a deux poids de lignes, $w_r + 1$ et w_r. Pour n, k et w_c donnés, il est possible que tous les $(n - k)$-uplets soient utilisés ou rejetés avant que $\boldsymbol{H}$ ne soit formé. Pour réduire cette possibilité, on doit choisir n, k et w_c de telle sorte que le nombre total de $(n - k)$-uplets binaires, $\binom{n-k}{w_c} =$

$\frac{(n-k)!}{w_c!(n-k-w_c)!}$ soit beaucoup plus grand que la longueur de code n, ou on peut assouplir les contraintes de poids des lignes à l'étape 3 pour permettre plusieurs poids de ligne. Cela réduit également la probabilité qu'une colonne candidate choisie, qui satisfait la contrainte à l'étape 2, soit rejetée à l'étape 3. Bien sûr, nous pouvons redémarrer le processus de construction en choisissant une autre séquence de colonnes candidates.

Si le rang sur les lignes de $\boldsymbol{H}$ est exactement $n-k$, l'espace nul de $\boldsymbol{H}$ donne un code LDPC(n,k) avec un taux exactement $R=\frac{k}{n}$. Si le rang de $\boldsymbol{H}$ est inférieur à $n-k$, alors l'espace nul donne un code LDPC(n,k') avec $k'>k$ et $\frac{k'}{n}>\frac{k}{n}$. On peut aisément montrer (voir équation (2.3)) que le taux R du code construit est borné par $R \leq 1-\frac{w_c}{w_r}$.

La contrainte à l'étape 2 de la sélection de colonne assure que le graphe de Tanner du code ne contient pas de cycle de longueur 4. Donc, le 'girth' du graphe de Tanner est au moins 6.

La construction précédente est efficace seulement pour les valeurs relativement petites de w_c, habituellement 3 ou 4. Pour des valeurs de w_c plus grandes, la satisfaction des contraintes aux étapes 2 et 3 peut être coûteuse en termes de calcul.

Le code construit n'est pas unique, parce qu'à l'étape 1 de la construction une colonne est choisie aléatoirement parmi les $(n-k)$-uplets binaires disponibles restants. La construction donne un ensemble de codes LDPC aléatoires. Avec cette construction, il est très difficile de déterminer la distance minimale du code construit. Pour de petites valeurs de w_c, 3 ou 4, la limite inférieure sur la distance minimale est $w_c + 1$.

2.4.5. Autres méthodes et comparatif :

En plus des méthodes algébriques basées sur les matrices de permutation, nous avons des constructions basées sur les géométries finies qui sont introduites par Kou et al dans [25]. Les codes obtenus avec les géométries finies, ne contiennent pas de cycle de longueur 4. De plus, ces codes ont de bonnes distances minimales et des performances à quelques dixièmes de dB de la limite théorique de Shannon. La construction de ces codes se base sur les points et lignes de la géométrie projective et de la géométrie euclidienne sur les corps finis. Nous avons deux types de code basés sur la géométrie euclidienne qui sont dénotés par Type-I EG-LDPC Codes et Type-II EG-LDPC Codes et deux autres types de code basés sur la géométrie

projective qui sont dénoté par Type-I PG-LDPC Codes, Type-II PG-LDPC Codes. Pour plus de détails sur ces types le lecteur est invité à voir les références [15, 25]. Mais il faut noter que toutes les géométries finies peuvent être mises en forme soit cyclique ou quasi-cyclique; ce qui permet d'avoir un encodage en temps linéaire avec des registres de décalage et peu d'espace mémoire [24].

Le tableau suivant résume les propriétés de base de chaque méthode de construction présentée et aussi citée ici [26].

Tableau 2.1 Propriétés des méthodes de construction des codes LDPC

	Propriétés de la construction					
Méthode de construction	Simplicité	Girth $g > 4$ garanti	$\boldsymbol{H}$ de rang complet garanti	Encodage rapide	Profil d'irrégularité	Performance
Gallager	√					***
Quasi-Cycliques				√	√	**
Array Codes		√	√			*
Array Codes modifiés		√	√	√		*
MacKay-Neal		√			√	***
Géométries finies		√		√		*

2.5. Encodage des codes LDPC :

Comme avec n'importe quel code en bloc linéaire, un code LDPC peut être encodé simplement en multipliant le vecteur correspondant aux bits d'information que l'on souhaite coder avec la matrice génératrice, c.à.d. $\boldsymbol{c} = \boldsymbol{uG}$.

Étant donné que les codes LDPC sont généralement définis par leur matrice de contrôle de parité, plutôt que leur matrice génératrice, l'introduction d'un moyen d'encodage en utilisant uniquement de la matrice de contrôle de parité serait utile. Rappelons que tous les mots de code satisfont à l'équation de vérification de parité $\boldsymbol{c} \cdot \boldsymbol{H}^T = 0$.

L'encodage d'un code LDPC peut se révéler relativement complexe si la matrice $\boldsymbol{H}$ n'a pas de structure particulière. Il existe des solutions génériques d'encodage, dont un algorithme de complexité en $\mathcal{O}(n)$, nécessitant un prétraitement complexe sur la matrice $\boldsymbol{H}$. Une autre solution consiste à construire directement la matrice $\boldsymbol{H}$ de façon à obtenir un code

systématique très simple à encoder. C'est notamment cette solution qui a été adoptée pour le code du standard DVB-S2 de transmission numérique de télévision par satellite [21].

2.5.1. Encodage par matrice génératrice :

Une transformation de $\boldsymbol{H}$ en matrice systématique $\boldsymbol{H}_{sys}$ est possible, par exemple avec l'algorithme d'élimination gaussienne. Cette technique, relativement simple, a cependant un inconvénient majeur : la matrice génératrice $\boldsymbol{G}_{sys}$ du code systématique n'est généralement pas creuse. La complexité d'encodage augmente rapidement en $\mathcal{O}(n^2)$, ce qui rend cette opération trop complexe pour des codes de taille usuelle [21].

On peut dériver une matrice génératrice $\boldsymbol{G}$ à partir de la matrice de contrôle de parité $\boldsymbol{H}$ au moyen de l'élimination gaussienne en arithmétique modulo-2. Puisque la matrice $\boldsymbol{G}$ est générée une fois pour une matrice de contrôle de parité, elle est utilisable dans tout le codage des messages.

Maintenant, si nous trouvons une décomposition $\boldsymbol{H}_{sys}$ de $\boldsymbol{H}$ en utilisant des permutations sur les colonnes :

$$\boldsymbol{H}_{sys} = [\boldsymbol{H}_p \, , \boldsymbol{H}_s] \tag{2.15}$$

Tel que $\boldsymbol{H}_p$ de $m \times m$ est inversible et $\boldsymbol{H}_s$ de $m \times k$, on peut partitionner le vecteur de code $\boldsymbol{c}$ de $1 \times n$, en une partie systématique $\boldsymbol{c}_s$ et une de parité $\boldsymbol{c}_p$, comme $\boldsymbol{c} = [\boldsymbol{c}_p \, , \boldsymbol{c}_s]$.

Ensuite, en combinant l'équation de vérification de parité avec la décomposition ci-dessus, on obtient :

$$\boldsymbol{c}_p \boldsymbol{H}_p^T + \boldsymbol{c}_s \boldsymbol{H}_s^T = 0 \tag{2.16}$$

Les vecteurs $\boldsymbol{c}_p$ et $\boldsymbol{c}_s$ sont liés par $\boldsymbol{c}_p = \boldsymbol{c}_s \boldsymbol{P}$, où $\boldsymbol{P}$ est la matrice de coefficients de $k \times m$. Pour tout message non nul $\boldsymbol{c}_s$, la matrice de coefficients du codes LDPC satisfait la condition :

$$\boldsymbol{P} \boldsymbol{H}_p^T + \boldsymbol{H}_s^T = 0 \tag{2.17}$$

En résolvant l'équation (17) pour la matrice $\boldsymbol{P}$, on obtient :

$$\boldsymbol{P} = \boldsymbol{H}_s^T \left(\boldsymbol{H}_p^T\right)^{-1} \tag{2.18}$$

Où $\left(\boldsymbol{H}_p^T\right)^{-1}$ est la matrice inverse de $\boldsymbol{H}_p^T$, qui est naturellement définie en arithmétique en modulo-2. Enfin, la matrice génératrice du code LDPC est définie par :

$$\boldsymbol{G}_{sys} = [\boldsymbol{P}\,, \boldsymbol{I}_k] = \left[\boldsymbol{H}_s^T\left(\boldsymbol{H}_p^T\right)^{-1}, \boldsymbol{I}_k\right] \tag{2.19}$$

Où $\boldsymbol{I}_k$ est la matrice identité $k \times k$.

2.5.2. Encodage à complexité linéaire :

Richardson et Urbanke ont proposé une solution permettant un encodage quasi-linéaire de complexité en $\mathcal{O}(n)$, ainsi que des algorithmes, pouvant être qualifiés littéralement de gourmands, qui permettent d'effectuer un prétraitement de la matrice de contrôle de parité $\boldsymbol{H}$. Le but du prétraitement est de mettre $\boldsymbol{H}$ sous une forme pseudo triangulaire inférieure, comme illustrée dans la figure 2.2, en utilisant uniquement des permutations de lignes ou de colonnes. Cette matrice est composée de six sous-matrices, toujours creuses, notées $\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$,$\boldsymbol{D}$,$\boldsymbol{E}$ et d'une matrice $\boldsymbol{T}$ triangulaire inférieure [21].

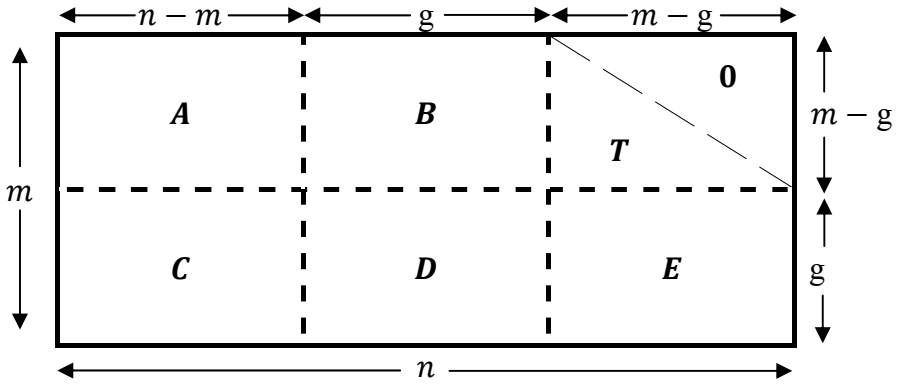

Figure 2.2. Représentation sous forme pseudo triangulaire inférieure de la matrice $\boldsymbol{H}$

La procédure par étapes d'un encodage efficace du codage LDPC est la suivante [10]:

Etape 1: En effectuant des permutations de lignes et de colonnes, la matrice de contrôle de parité non-singulière $\boldsymbol{H}$, doit être amenée dans une forme triangulaire inférieure, notée $\boldsymbol{H}_t$:

$$\boldsymbol{H}_t = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{E} \end{bmatrix} \tag{2.20}$$

Avec un écart g aussi petit que possible. Où $\boldsymbol{A}$ est de taille $(m-\mathrm{g})\times(n-m)$, $\boldsymbol{B}$ de $(m-\mathrm{g})\times\mathrm{g}$, $\boldsymbol{C}$ de $\mathrm{g}\times(n-m)$, $\boldsymbol{D}$ de $\mathrm{g}\times\mathrm{g}$, $\boldsymbol{E}$ de $\mathrm{g}\times(m-\mathrm{g})$ et $\boldsymbol{T}$ de $(m-\mathrm{g})\times(m-\mathrm{g})$. Aussi $\boldsymbol{T}$ est triangulaire inférieure avec des '1' le long de la diagonale.

Etape 2: On effectue une multiplication à droite de $\boldsymbol{H}_t$ par la matrice $\begin{bmatrix} \boldsymbol{I}_{m-\mathrm{g}} & \boldsymbol{0} \\ -\boldsymbol{E}\boldsymbol{T}^{-1} & \boldsymbol{I}_{\mathrm{g}} \end{bmatrix}$:

$$\begin{bmatrix} \boldsymbol{I}_{m-\mathrm{g}} & \boldsymbol{0} \\ -\boldsymbol{E}\boldsymbol{T}^{-1} & \boldsymbol{I}_{\mathrm{g}} \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{E} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C} & -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B}+\boldsymbol{D} & \boldsymbol{0} \end{bmatrix} \tag{2.21}$$

Etape 3: Obtenir $\boldsymbol{c}_{p1}$ en utilisant l'équation suivante :

$$\boldsymbol{c}_{p1}{}^{T} = -\boldsymbol{\Phi}^{-1}(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C})\boldsymbol{c}_{s}{}^{T} \tag{2.22}$$

Où $\boldsymbol{\Phi} = -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B}+\boldsymbol{D}$ et $\boldsymbol{c}_s$ est le vecteur du message d'information. Afin de vérifier que le $\boldsymbol{\Phi}$ est non-singulière, il doit être assuré en effectuant des permutations de colonne davantage.

Etape 4 : Obtenir $\boldsymbol{c}_{p2}$ en utilisant l'équation suivante :

$$\boldsymbol{c}_{p2}{}^{T} = -\boldsymbol{T}^{-1}\left(\boldsymbol{A}\boldsymbol{c}_{s}{}^{T}+\boldsymbol{B}\boldsymbol{c}_{p1}{}^{T}\right) \tag{2.23}$$

Etape 5 : Formez le vecteur du mot de code $\boldsymbol{c}$ comme :

$$\boldsymbol{c} = [\boldsymbol{c}_s\,,\boldsymbol{c}_{p1}\,,\boldsymbol{c}_{p2}] \tag{2.24}$$

$\boldsymbol{c}_{p1}$ contient les premiers g bits de parité et $\boldsymbol{c}_{p2}$ contient les $(m-\mathrm{g})$ bits de parité restants.

De nombreuses opérations coûteuses en temps peuvent être faites une fois pour toute dans un prétraitement. Toutes les opérations répétées au cours de l'encodage ont une complexité en $\mathcal{O}(n)$ sauf la multiplication de $(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C})\boldsymbol{c}_{s}{}^{T}$ par la matrice carrée $-\boldsymbol{\Phi}^{-1}$ de taille $\mathrm{g}\times\mathrm{g}$ qui, après inversion, n'est plus creuse d'où une complexité en $\mathcal{O}(\mathrm{g}^2)$. Richardson et Urbanke ont montrés que l'on peut obtenir une valeur de g égale à une faible fraction de n ($\mathrm{g}=\alpha n$) où α est un coefficient suffisamment faible pour que $\mathcal{O}(\mathrm{g}^2) \ll \mathcal{O}(n)$ pour des valeurs suffisamment grandes de n [21].

2.6. Décodage des codes LDPC :

La classe d'algorithmes utilisés pour décoder les codes LDPC sont appelés collectivement des algorithmes à passage de messages MP (*message-passing*), car leur fonctionnement peut

être expliqué par le passage des messages le long des branches du graphe de Tanner. Chaque nœud du graphe de Tanner fonctionne de manière isolée, ayant accès uniquement aux messages des branches avec lesquelles il est raccordé. Les algorithmes MP sont un type d'algorithme de décodage itératif où les messages passent en arrière et en avant, entre les nœuds de variable et les nœuds de parité, itérativement jusqu'à ce qu'un résultat soit atteint (ou que le processus soit arrêté). Ces algorithmes sont nommés selon le type de message transmis ou selon le type d'opération effectuée sur les nœuds.

Dans certains algorithmes, tels que le décodage ferme dit *bit-flipping*, les messages sont binaire et dans d'autres, tels que l'algorithme de décodage à propagation de la croyance BP ou somme-produit, les messages sont des probabilités (ou leurs rapports de vraisemblance logarithmique LLR) qui représentent un niveau de croyance sur la valeur des bits du mot de code.

Dans cette section nous présenterons les deux principaux algorithmes à savoir le *bit-flipping* et le *sum-product* (connu aussi par *Belief Propagation*). Dans le premier le principe itératif de l'algorithme MP est introduit de façon simplifié et qui constituera le point de départ pour présenter le deuxième algorithme de décodage itératif souple des codes LDPC auquel nous nous intéressons.

2.6.1. L'algorithme Bit-Flipping (BF) :

Bit-flipping est le nom donné à l'algorithme de décodage par passage de message MP à entrée et sortie ferme (ou dure) pour les codes LDPC. Une décision binaire (ferme) sur chaque bit reçu est effectuée par le détecteur et transmise au décodeur. Pour l'algorithme *bit-flipping*, les messages transmis le long des branches du graphe de Tanner sont également binaires : un nœud de variable envoie un message déclarant s'il s'agit d'un '1' ou d'un '0', et chaque nœud de parité envoie un message à chaque nœud de variable qui lui-est connecté, déclarant sa décision sur la valeur de ce bit (sur la base des informations disponibles au nœud de parité), ou un message indiquant si son équation de contrôle de parité est satisfaite.

L'algorithme de décodage *bit-flipping* (par basculement de bits) consiste essentiellement à corriger les bits du vecteur reçu $\boldsymbol{r}$ en observant la sortie de l'ensemble des équations de parité. En premier lieu, on calcule le syndrome $\boldsymbol{s}$ du vecteur reçu. Si celui-ci est nul, on suppose que le vecteur reçu est correct et on le décode avec le message correspondant $\tilde{\boldsymbol{u}}$. Dans le cas contraire, le syndrome étant différent de zéro, il y a donc des bits en erreur.

L'algorithme consiste à faire basculer les bits reçus causant le plus grand nombre d'erreur de parité. Le syndrome est à nouveau calculé et les bits reçus inversés à nouveau jusqu'à ce que l'on obtienne un syndrome nul, et donc un mot de code valide qui peut alors être décodé en un message, ou lorsque l'algorithme a atteint un nombre maximal prédéterminé d'itérations (si le syndrome diffère de zéro, l'algorithme rapporte une erreur de décodage) [11].

Phase 1 : Calcul du syndrome : $\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^T$. Si $\boldsymbol{s} = \boldsymbol{0}$, l'algorithme se termine : le vecteur reçu est un mot de code valide, $\tilde{\boldsymbol{c}}$, qui peut être décodé en un message $\tilde{\boldsymbol{u}}$.

Le décodeur LDPC s'arrête immédiatement lorsqu'un mot de code valide a été trouvé en vérifiant si les équations de contrôle de parité sont satisfaites. C'est-à-dire que le syndrome $\boldsymbol{s} = \boldsymbol{0}$ est un critère d'arrêt pour l'algorithme *bit-flipping*. Ceci est vrai pour tout décodage à passage de message des codes LDPC et présente deux avantages importants. D'une part, des itérations supplémentaires sont évitées une fois qu'une solution a été trouvée et, d'autre part, l'insuccès à converger vers un mot de code est toujours détecté.

Phase 2 : Déterminer pour chacun des n bits reçus le nombre d'équations de parité en erreur : $\{f_1, f_2, \dots, f_n\}$.

Phase 3 : Déterminer l'ensemble $\mathcal{S}$ des bits reçus pour lesquels on obtient un maximum d'équations de parité en erreur, i.e. $max[f_1, f_2, \dots, f_n]$.

Phase 4 : Faire basculer l'ensemble des bits reçus dans $\mathcal{S}$ et retourner à la première étape (calcul de syndrome).

Exemple 1:

La matrice de parité $\boldsymbol{H}$ d'un code LDPC de longueur $n = 12$ et $m = 6$, de poids de lignes $w_r = 6$ et de colonnes $w_c = 3$ est donnée par [11]:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad (2.25)$$

Le graphe de Tanner de ce code LDPC est :

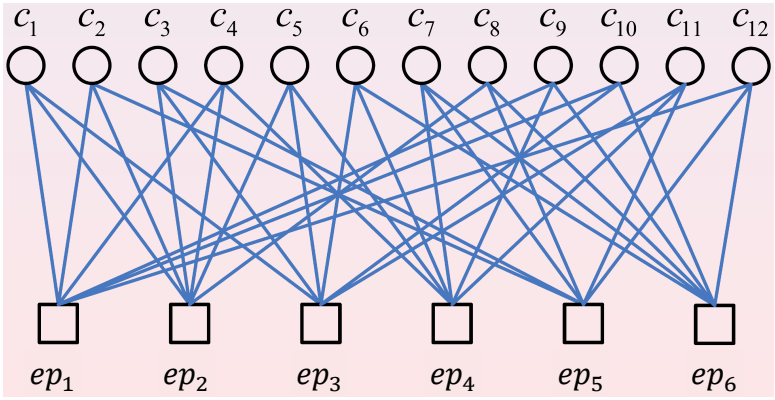

Figure 2.3. Graphe de Tanner du code LDPC de l'exemple 1

Supposons que l'on reçoive le vecteur $\boldsymbol{r} = [1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0]$. Le calcul du syndrome donne $\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^T = [0\ 1\ 0\ 1\ 1\ 1]$. Ce qui indique que le vecteur reçu est en erreur. On détermine alors pour chacun des $n = 12$ bits le nombre d'équations de parité en erreur :

$$\{f_1 = 1, f_2 = 2, f_3 = 2, f_4 = 2, f_5 = 2, f_6 = 2, f_7 = 3, f_8 = 3, f_9 = 2, f_{10} = 1, f_{11} = 2, f_{12} = 2\}$$

L'ensemble $\mathcal{S}$ des bits reçus conduisant à un maximum d'équations de parité en erreur est : $\mathcal{S} = [c_7, c_8]$. On fait alors basculer les bits reçus c_7, c_8 : $\boldsymbol{r} = [1\ 0\ 0\ 0\ 0\ 1\ \underline{0}\ \underline{1}\ 1\ 0\ 0\ 0]$.

Le calcul du syndrome $\boldsymbol{s} = [0\ 0\ 0\ 0\ 1\ 1]$. Le syndrome étant différent de zéro, on effectue une seconde itération de l'algorithme de basculement de bits.

Les erreurs dans les équations de parité pour chaque bit reçu sont cette fois-ci :

$$\{f_1 = 0, f_2 = 1, f_3 = 1, f_4 = 0, f_5 = 0, f_6 = 1, f_7 = 2, f_8 = 2, f_9 = 1, f_{10} = 1, f_{11} = 1, f_{12} = 2\}$$

Et $\mathcal{S} = [c_7, c_8, c_{12}]$, alors on fait basculer les bits c_7, c_8, et c_{12} et on obtient $\boldsymbol{r} = [1\ 0\ 0\ 0\ 0\ 1\ \underline{1}\ \underline{0}\ 1\ 0\ 0\ \underline{1}]$.

Le calcul du syndrome donne $\boldsymbol{s} = [1\ 1\ 0\ 1\ 0\ 0]$. Celui-ci étant toujours différent de zéro, on effectue une troisième itération.

$$\{f_1 = 2, f_2 = 2, f_3 = 1, f_4 = 3, f_5 = 2, f_6 = 1, f_7 = 1, f_8 = 1, f_9 = 2, f_{10} = 1, f_{11} = 1, f_{12} = 1\}$$

Donc $\mathcal{S} = [c_4]$. On ne fait basculer que le quatrième bit du vecteur reçu $\boldsymbol{r} = [1\ 0\ 0\ \underline{1}\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1]$, et le calcul du syndrome est cette fois-ci nul $\boldsymbol{s} = [0\ 0\ 0\ 0\ 0\ 0]$.

Le vecteur $\boldsymbol{r}$ est maintenant un mot de code valide, $\tilde{\boldsymbol{c}}$, (idéalement le bon) qui peut être maintenant décodé en un message.

Le tableau suivant récapitule les étapes de cet exemple de façon plus compacte.

Tableau 2.2. Etapes du décodage BF pour l'exemple 1

Mot de code reçu $\boldsymbol{r}$	[1 0 0 0 0 1 1 0 1 0 0 0]

Décodage avec l'algorithme Bit-Flipping																
Phase 1			**Phase 2**											**Phase 3**	**Phase 4**	
Syndrome $\boldsymbol{s}$	Erreurs	Itér.	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	$\mathcal{S}$	Mot de code $\tilde{\boldsymbol{c}}$
[0 1 0 1 1 1]	Oui	1	1	2	2	2	2	2	**3**	**3**	2	1	2	2	$[c_7, c_8]$	[1 0 0 0 0 1 01 1 0 0 0]
[0 0 0 0 1 1]	Oui	2	0	1	1	0	0	1	**2**	**2**	1	1	1	**2**	$[c_7, c_8, c_{12}]$	[1 0 0 0 0 1 10 1 0 0 1]
[1 1 0 1 0 0]	Oui	3	2	2	1	**3**	2	1	1	1	2	1	1	1	$[c_4]$	[1 0 0 1 0 1 1 0 1 0 0 1]
[0 0 0 0 0 0]	Non	-														↓
											Mot de code corrigé $\tilde{\boldsymbol{c}}$					[**1 0 0 1 0 1 1 0 1 0 0 1**]

2.6.2. L'algorithme Sum-Product (SP) :

L'algorithme somme-produit est un algorithme de passage de message à décision souple. Il est semblable à l'algorithme *bit-flipping* décrit dans la section précédente, mais les messages représentant chaque décision (si la valeur de bit est 1 ou 0) sont maintenant des probabilités. Alors que le décodage BF accepte une première décision ferme sur les bits reçus en entrée, l'algorithme de SP est un algorithme de décision souple qui accepte la probabilité pour chaque bit reçu en tant qu'entrée. A la sortie du canal, les probabilités de bits reçues, sont également appelées probabilités a priori des bits reçus, car elles sont connues à l'avance avant que le décodeur LDPC ne soit opérationnel. Pour le décodeur SP, les informations extrinsèques transmises entre les nœuds sont également données comme des probabilités plutôt que des décisions fermes.

Dans ce qui suit, nous utilisons la notation B_j pour représenter l'ensemble des nœuds de variables connectés au $j^{\text{ième}}$ nœud de parité du graphe de Tanner correspondant à la matrice $\boldsymbol{H}$. De même, nous utilisons la notation A_i pour représenter l'ensemble des nœuds de parités connectés à l'$i^{\text{ème}}$ nœud de variable.

Si on considère le code LDPC de l'exemple 1, on a :

j	1	2	3	4	5	6
B_j	[1,2,4,9,10,12]	[1,2,3,4,5,8]	[1,3,5,6,10,11]	[4,5,6,7,9,11]	[2,3,7,8,11,12]	[6,7,8,9,10,12]

i	1	2	3	4	5	6	7	8	9	10	11	12
A_i	[1,2,3]	[1,2,5]	[2,3,5]	[1,2,4]	[2,3,4]	[3,4,6]	[4,5,6]	[2,5,6]	[1,4,6]	[1,3,6]	[3,4,5]	[1,5,6]

Le message extrinsèque $E_{j,i}$ du nœud de parité j à un nœud de variable i auquel il est connecté, est l'avis du nœud de parité j sur la probabilité que $c_i = 1$, sur la base des informations disponibles au nœud de parité j. C'est-à-dire que $E_{j,i}$ donne la probabilité que $c_i = 1$, ce qui implique que l'équation de contrôle de parité j soit satisfaite. Notons que $E_{j,i}$ n'est pas défini si le bit c_i n'est pas inclus dans l'équation de contrôle de parité j, car aucune information extrinsèque n'est transmise entre les nœuds i et j dans ce cas.

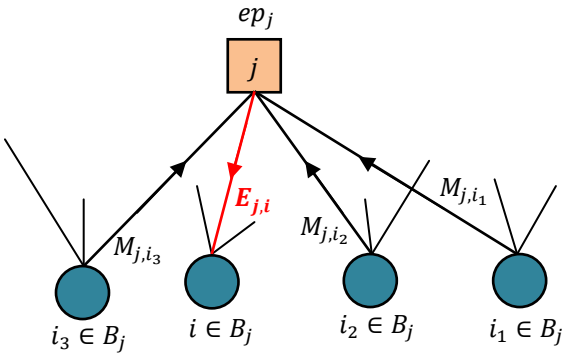

Figure 2.4. Mise à jour des messages $E_{j,i}$ se propageant d'un nœud de parité vers un nœud de variable

La probabilité qu'une équation de contrôle de parité soit satisfaite si $c_i = 1$, est la probabilité qu'un nombre impair de bits dans cette équation de vérification de parité soit '1' à l'exclusion de c_i. Gallager a démontré dans [8] que cette probabilité est :

$$P_{j,i}^{ext} = \frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j, i' \neq i} \left(1 - 2P_{j,i'}\right) \tag{2.26}$$

Où $P_{j,i'}$ est l'estimation actuelle de la probabilité que $c_i = 1$ disponible au nœud de parité j. La probabilité que l'équation de contrôle de parité soit satisfaite si $c_i = 0$ est donc $1 - P_{j,i}^{ext}$.

Pour une variable binaire x il est facile de trouver $p(x = 1)$ étant donné $p(x = 0)$, puisque $p(x = 1) = 1 - p(x = 0)$, et donc il suffit de stocker une seule valeur de probabilité pour x. Les rapports de vraisemblance logarithmique LLR (équation (1.18)) sont utilisés pour représenter les métriques d'une variable binaire par une valeur unique. La conversion des LLRs en probabilités est :

$$p(x = 1) = \frac{e^{-L(x)}}{1 - e^{-L(x)}} \quad , \quad p(x = 0) = \frac{e^{L(x)}}{1 + e^{L(x)}} \tag{2.27}$$

L'avantage de la représentation logarithmique des probabilités est que, lorsque les probabilités doivent être multipliées, les rapports de vraisemblance logarithmique doivent seulement être ajoutés, cela peut réduire la complexité du décodeur SP.

Le LLR de l'information extrinsèque depuis le nœud de parité j au nœud de variable i est alors :

$$E_{j,i} = L\left(P_{j,i}^{ext}\right) = \ln \frac{1 - P_{j,i}^{ext}}{P_{j,i}^{ext}} \tag{2.28}$$

En remplaçant (2.26) dans (2.28) on obtient :

$$E_{j,i} = \ln \frac{\frac{1}{2} + \frac{1}{2}\prod_{i' \in B_j, i' \neq i}\left(1 - 2P_{j,i'}\right)}{\frac{1}{2} - \frac{1}{2}\prod_{i' \in B_j, i' \neq i}\left(1 - 2P_{j,i'}\right)} \tag{2.29}$$

En mettant $M_{j,i'} = L\left(P_{j,i'}\right) = \ln \frac{1-P_{j,i'}}{P_{j,i'}}$, et en utilisant la relation $\tanh\left(\frac{1}{2}\ln\left(\frac{1-p}{p}\right)\right) = 1 - 2p$, l'équation (2.29) s'écrit à nouveau sous la forme suivante :

$$E_{j,i} = \ln \frac{1 + \prod_{i' \in B_j, i' \neq i} \tanh\left(\frac{M_{j,i'}}{2}\right)}{1 - \prod_{i' \in B_j, i' \neq i} \tanh\left(\frac{M_{j,i'}}{2}\right)} \tag{2.30}$$

Par ailleurs, en utilisant la relation $2\tanh^{-1} p = \ln\left(\frac{1+p}{1-p}\right)$, l'expression de $E_{j,i}$ devient :

$$E_{j,i} = 2\tanh^{-1}\left(\prod_{i' \in B_j, i' \neq i} \tanh\left(\frac{M_{j,i'}}{2}\right)\right) \tag{2.31}$$

Chaque nœud de variable a accès au LLR d'entrée $R_i = L(r_i)$, et aux LLR de chaque nœud de parité qui lui-sont connectés. Le LLR total de l'$i^{\text{ème}}$ bit est la somme de ces LLRs :

$$L_i = L(P_i) = R_i + \sum_{j \in A_i} E_{j,i} \tag{2.32}$$

Toutefois, les messages envoyés par les nœuds de variable aux nœuds de parité, $M_{j,i}$, ne sont pas la valeur LLR complète pour chaque bit. Pour éviter de renvoyer à chaque nœud de parité l'information qu'il possède déjà, le message envoyé de l'$i^{\text{ème}}$ nœud de variable au $j^{\text{ième}}$

nœud de parité est la somme dans l'équation (2.32), sans la composante $E_{j,i}$ venant juste du $j^{\text{ième}}$ nœud de parité. C'est-à-dire :

$$M_{j,i} = R_i + \sum_{j' \in A_i,\, j' \neq j} E_{j',i} \tag{2.33}$$

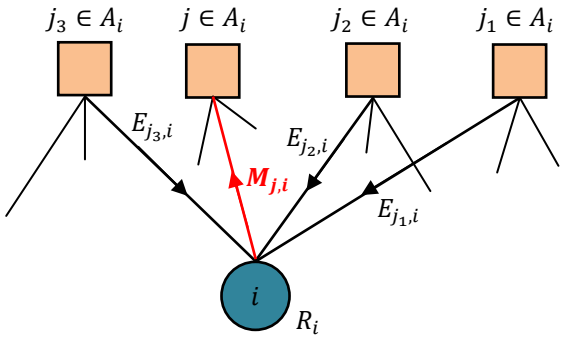

Figure 2.5. Mise à jour des messages $M_{j,i}$ se propageant d'un nœud de variable vers un nœud de parité

Pour réduire la complexité d'implémentation du décodeur, l'algorithme SP peut être modifié de manière à remplacer le terme produit dans l'équation (2.31) par une somme, du fait que la fonction tanh est monotone et impaire.

En mettant $M_{j,i'} = \alpha_{j,i'}\beta_{j,i'}$, où $\alpha_{j,i'} = sign(M_{j,i'})$ et $\beta_{j,i'} = |M_{j,i'}|$, alors l'équation (2.31) est récrite comme suit :

$$E_{j,i} = \prod_{i' \in B_j, i' \neq i} (\alpha_{j,i'}) \, 2\tanh^{-1}\left(\prod_{i' \in B_j, i' \neq i} \tanh\left(\frac{\beta_{j,i'}}{2}\right)\right) \tag{2.34}$$

Pour remplacer le produit par une somme on utilise $\ln^{-1}(\ln(x)) = x$ dans l'équation (2.34), ainsi on a :

$$E_{j,i} = \prod_{i' \in B_j, i' \neq i} (\alpha_{j,i'}) \, 2\tanh^{-1}\left(\ln^{-1}\left(\ln\left(\prod_{i' \in B_j, i' \neq i} \tanh\left(\frac{\beta_{j,i'}}{2}\right)\right)\right)\right)$$

$$= \prod_{i' \in B_j, i' \neq i} (\alpha_{j,i'}) \, 2\tanh^{-1}\left(\ln^{-1}\left(\sum_{i' \in B_j, i' \neq i} \ln\left(\tanh\left(\frac{\beta_{j,i'}}{2}\right)\right)\right)\right) \tag{2.35}$$

On va introduire la fonction $\phi(x)$, qui a la propriété mathématique $\phi^{-1} = \phi$ pour $x > 0$, définie par la relation suivante :

$$\phi(x) = -\ln\left(\tanh\frac{x}{2}\right) = \ln\frac{e^x + 1}{e^x - 1} \tag{2.36}$$

On obtient finalement :

$$E_{j,i} = \prod_{i' \in B_j, i' \neq i} (\alpha_{j,i'})\ \phi\left(\sum_{i' \in B_j, i' \neq i} \phi\left(\beta_{j,i'}\right)\right) \tag{2.37}$$

Le produit des signes peut être calculé en utilisant l'addition modulo-2 des décisions fermes sur chaque $M_{j,i'}$, alors que la fonction ϕ peut être mise en œuvre facilement à l'aide d'une table de consultation (lookup table) [11].

L'algorithme SP calcule itérativement une approximation de la valeur MAP L_i pour chaque bit de code c_i. Cependant, les probabilités a posteriori calculées par le décodeur SP sont des probabilités MAP exactes seulement si le graphe de Tanner est sans cycle. En bref, l'information extrinsèque obtenue à partir d'une équation de contrôle de parité dans la première itération est indépendante de l'information a priori du bit en question (elle dépendra bien entendu de l'information a priori des autres bits du mot de code). L'information extrinsèque fournie au nœud de variable i dans les itérations suivantes reste indépendante de l'information a priori initiale pour c_i jusqu'à ce que cette dernière soit retournée au nœud de variable i via un cycle dans le graphe de Tanner.

Le fonctionnement détaillé de l'algorithme SP est récapitulé par le pseudo-code suivant :

Algorithme de décodage Sum-Product (SP)

Input : $\boldsymbol{r} = (r_1, r_2, \dots, r_n) \in \mathcal{R}^n$ ▷ Mot de code reçu

($\mathcal{R}$ est l'alphabet de la sortie du canal)

Output : $\tilde{\boldsymbol{c}} = (\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_n) \in \{0,1\}^n$ ▷Mot de code estimé

Initialisation :

for all $i = 1, \dots, n$ **do** $R_i = \ln\left(\frac{P(c_i=0|r_i)}{P(c_i=1|r_i)}\right)$;

for all $i = 1, \dots, n$ and $j \in A_i$ **do** $M_{j,i} = R_i$ (équ.(2.33) avec $E_{j,i} = 0$)

$iter = 1$

Iteration Loop :

for all $j = 1, \dots, m$ and $i \in B_j$ **do** ▷Messages des nœuds de parité-vers-variable

$$E_{j,i} = \prod_{i' \in B_j, i' \neq i} (\alpha_{j,i'}) \; \phi\left(\sum_{i' \in B_j, i' \neq i} \phi\left(\beta_{j,i'}\right) \right)$$

for all $i = 1, \dots, n$ and $j \in A_i$ **do** ▷Messages des nœuds de variable-vers-parité

$$M_{j,i} = R_i + \sum_{j' \in A_i,\, j' \neq j} E_{j',i}$$

for all $i = 1, \dots, n$ **do** ▷Information a posteriori

$$L_i = R_i + \sum_{j \in A_i} E_{j,i}$$

for all $i = 1, \dots, n$ **do** ▷Décision ferme

$$\tilde{c}_1 = \frac{1}{2}\left(1 - sign(L_i)\right)$$

if $\tilde{\boldsymbol{c}}$ est un mot de code **or** $iter = It_{max}$ **then** : terminer iteration loop

$iter = iter + 1$

End Iteration Loop

Chapitre **3**

Les Codes Convolutifs et les Turbocodes

Dans ce chapitre, les principaux algorithmes de décodage convolutif, à savoir les algorithmes Viterbi et Maximum a Posteriori (MAP) sont étudiés. En particulier, nous fournissons l'arrière-plan requis sur les représentations graphiques des codes en utilisant le diagramme en treillis et le diagramme de transitions d'états. Par la suite, nous présentons les structures de l'encodeur / décodeur de turbocodes. Les principaux algorithmes de décodage des turbocodes, à savoir l'algorithme MAP et les variantes Log-MAP et MaxLog-MAP, sont également introduits dans ce chapitre.

3.1. Introduction :

Le codage probabiliste est la ligne de développement alternative qui a été directement inspirée par l'approche probabiliste de Shannon en matière de codage. Alors que la théorie du codage algébrique vise à trouver des codes spécifiques qui maximisent la distance minimale d pour (n, k) donnés, le codage probabiliste est beaucoup plus concerné par la recherche de classes de codes qui optimisent la performance moyenne en fonction de la complexité du codage et du décodage.

Les décodeurs probabilistes utilisent généralement des informations de décision souple (fiabilité), à la fois comme entrées (à partir des sorties du canal), et aux étapes intermédiaires du processus de décodage. Les schémas de codage classiques qui relèvent de cette classe comprennent les codes convolutifs, les codes de produits, les codes concaténés, la modulation codée en treillis et le décodage des codes de blocs en treillis.

Pendant de nombreuses années, la concurrence entre les approches algébriques et probabilistes a été établie comme une concurrence entre les codes en blocs et les codes convolutifs. Le codage convolutif a été motivé dès le début par l'objectif d'optimiser le compromis de la performance par rapport à la complexité, qui, sur le canal BI-AWGN, implique nécessairement des décisions souples et un décodage quasi optimal. En pratique, la

plupart des systèmes de codage canal ont utilisé des codes convolutifs. Les codes modernes qui s'approchent de la capacité sont le fruit ultime de cette ligne de développement [27].

C'est en 1955 que Peter Elias introduit la notion de code convolutif. La linéarité est la seule propriété algébrique des codes convolutifs qui est partagée avec les codes en blocs algébriques. La structure introduite par Elias a ensuite été comprise comme la structure dynamique d'un procédé Markovien à états-fini, discret dans le temps, de k-entrées, et n-sorties. Un code convolutif se caractérise par son taux de codage $R = k/n$, où k et n sont généralement de petits nombres entiers, et par le nombre de ses états, qui est souvent lié directement à la complexité du décodage [27].

Depuis, la communauté des chercheurs, dans le domaine de la théorie de l'information et du codage, a été motivée par le problème de trouver des algorithmes et méthodes pour pouvoir décoder près de la capacité avec une performance quasi optimale et une complexité faisable. Jack Wozencraft a développé en 1961 l'algorithme de décodage séquentiel '*Sequential decoding*'. Par la suite, en 1963, Jim Massey a proposé une méthode de décodage très simple pour les codes convolutifs, appelée décodage de seuil '*Threshold decoding*'. En 1967, Andy Viterbi [4] a présenté ce que l'on a appelé l'algorithme de Viterbi '*Viterbi Algorithm*' (VA) comme un algorithme de décodage asymptotiquement optimal pour les codes convolutifs, afin de prouver des limites d'erreur exponentielles. Il a été rapidement reconnu que l'algorithme de Viterbi était en fait un algorithme de décodage optimal.

En 1993, Berrou et co-auteurs [6] ont choqué la communauté de recherche dans le domaine du codage par la conception d'un système de codage appelé '*Turbo Codes*' (TC) qui a permis un saut quantique dans la performance des codes sur les canaux généraux. Ils ont obtenu de très bonnes performances d'erreur dans une faible marge de la capacité du canal, ce qui avait été pensé impossible avec les systèmes pratiques et avec une complexité modérée par la plupart des théoriciens du codage [28].

Dans la course effrénée, suite à ce développement, pour expliquer la théorie derrière cette performance incroyable, une méthode initialement développée par Gallager dans sa thèse de doctorat, connue sous le nom de codage LDPC (LowDensity Parity-Check) a été redécouverte par MacKay [9] et a montré qu'elle avait des propriétés comparables. Ces deux méthodes sont devenues les bourreaux du travail des normes de communication modernes, avec des arguments sur les avantages techniques de l'un sur l'autre, la plupart du temps obscurci par des intérêts commerciaux et de normalisation de qui argumente [28].

3.2. Structure et représentation mathématiques des codes convolutifs:

Les codes convolutifs sont la deuxième grande classe de codes correcteurs d'erreur en plus des codes en blocs. Pour un code convolutif à chaque instant t, l'encodeur délivre un bloc de n symboles binaires $\boldsymbol{c}_t = (c_t^{(1)}, c_t^{(2)}, \dots, c_t^{(n)})$. Ce bloc est une fonction du bloc de k symboles d'information $\boldsymbol{u}_t = (u_t^{(1)}, u_t^{(2)}, \dots, u_t^{(k)})$ présents à son entrée et de m blocs précédents. Les codes convolutifs par conséquent introduisent un effet mémoire d'ordre m.

La quantité $v = m + 1$ s'appelle la longueur de contrainte du code et le rapport $R = k/n$ s'appelle le taux du code. Si k symboles d'information à l'entrée du codeur sont explicitement trouvés dans le bloc codé $\boldsymbol{c}_t$, alors le code est systématique, c'est-à-dire :

$$\boldsymbol{c}_t = (u_t^{(1)}, u_t^{(2)}, \dots, u_t^{(k)}, c_t^{(k+1)}, c_t^{(k+2)}, \dots, c_t^{(n)}) \tag{3.1}$$

Le diagramme général d'un codeur convolutif (k, n, m) est représenté sur la figure 3.1 [13].

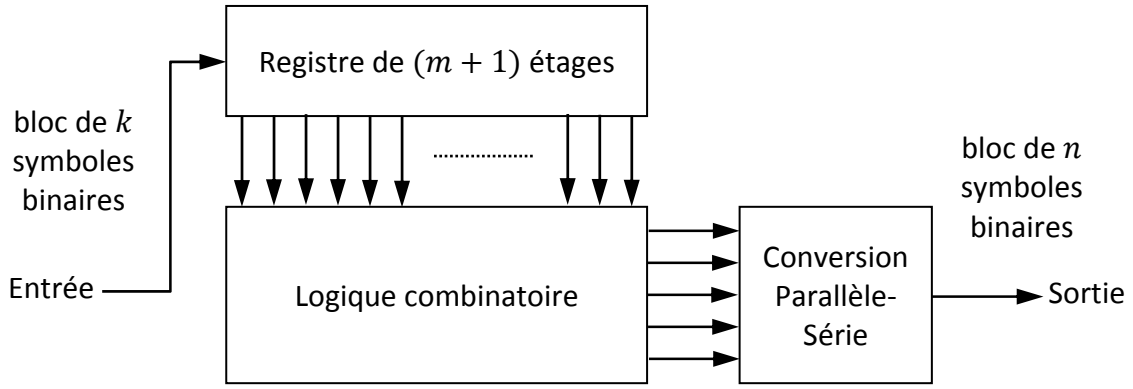

Figure 3.1. Diagramme général d'un codeur convolutif (k, n, m)

Un code convolutif (n, k, m) peut être implémenté avec un circuit séquentiel linéaire à k-entrée et n-sortie avec une mémoire d'entrée m. Typiquement, n et k sont de petits nombres entiers avec $k < n$, mais l'ordre de mémoire m doit être rendu grand pour atteindre de faibles probabilités d'erreur. En effet, outre la redondance, la mémoire est un paramètre important pour la capacité de correction d'erreur d'un code. Dans le cas spécial et important lorsque $k = 1$, la séquence d'informations n'est pas divisée en blocs et peut être traitée en continu.

3.2.1. Représentation matricielle :

Soit l'entrée du codeur convolutif (séquence d'information) une séquence de k-uplets binaires $\boldsymbol{u} = \boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_t, \dots$. La sortie (séquence de code) est une séquence de n-uplets binaires $\boldsymbol{c} = \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_t, \dots$. La relation entre la séquence d'informations et la séquence de code est déterminée par l'équation :

$$\boldsymbol{c} = \boldsymbol{u} \cdot \boldsymbol{G} \tag{3.2}$$

Où :

$$\boldsymbol{G} = \begin{bmatrix} G_0 & G_1 & \cdots & G_m & & \\ & G_0 & G_1 & \cdots & G_m & \\ & & G_0 & G_1 & \cdots & G_m \\ & & & \cdots & \cdots & \cdots \end{bmatrix} \tag{3.3}$$

$\boldsymbol{G}$ est une matrice génératrice semi-infinie, où les sous-matrices G_l , $(l = 0,1, \dots, m)$ sont des matrices $k \times n$ binaires dont les entrées sont :

$$G_l = \begin{bmatrix} g_{1,l}^{(1)} & g_{1,l}^{(2)} & \cdots & g_{1,l}^{(n)} \\ g_{2,l}^{(1)} & g_{2,l}^{(2)} & \cdots & g_{2,l}^{(n)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,l}^{(1)} & g_{k,l}^{(2)} & \cdots & g_{k,l}^{(n)} \end{bmatrix} \tag{3.4}$$

On appelle $\boldsymbol{g}_i^{(j)} = \left(g_{i,0}^{(j)}, g_{i,1}^{(j)}, \dots, g_{i,m}^{(j)}\right)$ les séquences génératrices.

L'arithmétique dans l'équation (3.2) est effectuée sur le champ binaire $\mathbb{GF}(2)$, et les parties laissées vierges dans la matrice génératrice $\boldsymbol{G}$ sont supposées être remplies de zéros [29].

Notons que les codes convolutifs sont bien adaptés au codage des transmissions avec un flux continu de données. En effet, les séquences de données à coder peuvent avoir une longueur quelconque [29].

Exemple 1 :

La figure 3.2 montre un codeur convolutif (2,1,3) non-systématique de rendement $R = \frac{k}{n} = \frac{1}{2}$ et d'une longueur de contrainte $\upsilon = m + 1 = 4$. La séquence d'information $\boldsymbol{u} =$

$u_0, u_1, u_2, \dots, u_t, \dots$ est convertie en deux séquences codées $\boldsymbol{c}^{(1)} = c_0^{(1)}, c_1^{(1)}, c_2^{(1)}, \dots, c_t^{(1)}, \dots$ et $\boldsymbol{c}^{(2)} = c_0^{(2)}, c_1^{(2)}, c_2^{(2)}, \dots, c_t^{(2)}, \dots$

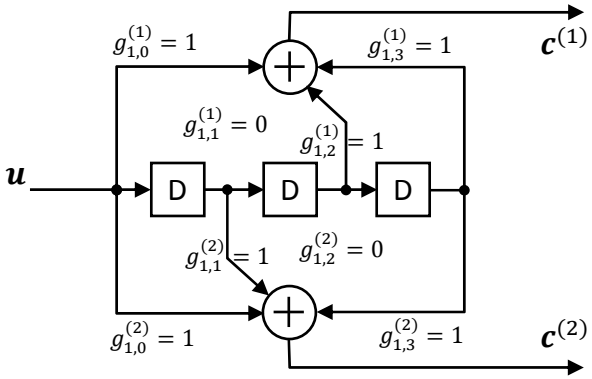

Figure 3.2. Codeur convolutif de rendement $R = \frac{1}{2}$

Les bits $c_t^{(1)}$ et $c_t^{(2)}$ à la sortie du codeur de la figure 3.2 à l'instant *t* sont :

$$c_t^{(1)} = u_t + u_{t-2} + u_{t-3}$$

$$c_t^{(2)} = u_t + u_{t-1} + u_{t-3}$$

Les sous-matrices $G_i, i = 0,1,2,3$, sont :

$$G_0 = \begin{pmatrix} g_{1,0}^{(1)} & g_{1,0}^{(2)} \end{pmatrix} = (1 \quad 1)$$
$$G_1 = \begin{pmatrix} g_{1,1}^{(1)} & g_{1,1}^{(2)} \end{pmatrix} = (0 \quad 1)$$
$$G_2 = \begin{pmatrix} g_{1,2}^{(1)} & g_{1,2}^{(2)} \end{pmatrix} = (1 \quad 0)$$
$$G_3 = \begin{pmatrix} g_{1,3}^{(1)} & g_{1,3}^{(2)} \end{pmatrix} = (1 \quad 1)$$

Ainsi, les séquences génératrices s'obtiennent comme :

$$\boldsymbol{g}^{(1)} = (1 \quad 0 \quad 1 \quad 1)$$
$$\boldsymbol{g}^{(2)} = (1 \quad 1 \quad 0 \quad 1)$$

Donc on obtient la matrice génératrice suivante :

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

Dans le cas général, un codeur convolutif code k séquences d'information :

$$\begin{aligned} \boldsymbol{u}^{(1)} &= u_0^{(1)}, u_1^{(1)}, u_2^{(1)}, \dots, u_t^{(1)}, \dots \\ \boldsymbol{u}^{(2)} &= u_0^{(2)}, u_1^{(2)}, u_2^{(2)}, \dots, u_t^{(2)}, \dots \\ \vdots &= \vdots \\ \boldsymbol{u}^{(k)} &= u_0^{(k)}, u_1^{(k)}, u_2^{(k)}, \dots, u_t^{(k)}, \dots \end{aligned} \tag{3.5}$$

En n séquences codées :

$$\begin{aligned} \boldsymbol{c}^{(1)} &= c_0^{(1)}, c_1^{(1)}, c_2^{(1)}, \dots, c_t^{(1)}, \dots \\ \boldsymbol{c}^{(2)} &= c_0^{(2)}, c_1^{(2)}, c_2^{(2)}, \dots, c_t^{(2)}, \dots \\ \vdots &= \vdots \\ \boldsymbol{c}^{(n)} &= c_0^{(n)}, c_1^{(n)}, c_2^{(n)}, \dots, c_t^{(n)}, \dots \end{aligned} \tag{3.6}$$

à l'aide des k registres à décalage de longueur : $m_1, m_2, \dots, m_k$. La mémoire totale m d'un codeur convolutif est donnée par :

$$m = \sum_{i=1}^{k} m_i \tag{3.7}$$

La longueur de contrainte est égale au nombre maximal de bits de sortie pouvant être affectés par un bit à l'entrée d'un codeur convolutif [30]:

$$v = \max_{i=1,\dots,k} (1 + m_i) \tag{3.8}$$

3.2.2. Représentation polynômiale :

On peut représenter les séquences d'information, les séquences codées et la matrice génératrice d'un code convolutif sous forme polynômiale. On utilise comme variable un opérateur de retard D. Ainsi une séquence d'information peut s'exprimer par $\boldsymbol{u}^{(i)}(D) = u_0^{(i)} + u_1^{(i)} D + u_2^{(i)} D^2 + \cdots + u_t^{(i)} D^t + \cdots$ avec $i = 1, \dots, k$. Alors qu'une séquence codée devient $\boldsymbol{c}^{(j)}(D) = c_0^{(j)} + c_1^{(j)} D + c_2^{(j)} D^2 + \cdots + c_t^{(j)} D^t + \cdots$ avec $j = 1, \dots, n$.

Quant à la matrice génératrice G(D), donnée par l'ensemble des polynômes générateurs $g_i^{(j)}(D)$ issus des séquences génératrices, elle s'écrit [29][30]:

$$\boldsymbol{G}(D) = \begin{bmatrix} \boldsymbol{g}_1^{(1)}(D) & \boldsymbol{g}_1^{(2)}(D) & \cdots & \boldsymbol{g}_1^{(n)}(D) \\ \boldsymbol{g}_2^{(1)}(D) & \boldsymbol{g}_2^{(2)}(D) & \cdots & \boldsymbol{g}_2^{(n)}(D) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{g}_k^{(1)}(D) & \boldsymbol{g}_k^{(2)}(D) & \cdots & \boldsymbol{g}_k^{(n)}(D) \end{bmatrix} \tag{3.9}$$

$$\boldsymbol{g}_i^{(j)}(D) = g_{i,0}^{(j)} + g_{i,1}^{(j)} D + g_{i,2}^{(j)} D^2 + \cdots + g_{i,m}^{(j)} D^m \tag{3.10}$$

Ainsi le codage convolutif des séquences d'information $\boldsymbol{u}^{(i)}(D)$ est obtenu par :

$$\begin{aligned} \boldsymbol{c}(D) &= [\boldsymbol{c}^{(1)}(D) \quad \boldsymbol{c}^{(2)}(D) \quad \cdots \quad \boldsymbol{c}^{(n)}(D)] = \boldsymbol{u}(D)\boldsymbol{G}(D) \\ &= [\boldsymbol{u}^{(1)}(D) \quad \boldsymbol{u}^{(2)}(D) \quad \cdots \quad \boldsymbol{u}^{(k)}(D)] \begin{bmatrix} \boldsymbol{g}_1^{(1)}(D) & \boldsymbol{g}_1^{(2)}(D) & \cdots & \boldsymbol{g}_1^{(n)}(D) \\ \boldsymbol{g}_2^{(1)}(D) & \boldsymbol{g}_2^{(2)}(D) & \cdots & \boldsymbol{g}_2^{(n)}(D) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{g}_k^{(1)}(D) & \boldsymbol{g}_k^{(2)}(D) & \cdots & \boldsymbol{g}_k^{(n)}(D) \end{bmatrix} \end{aligned} \tag{3.11}$$

Exemple 1 (suite):

La matrice génératrice du code de l'exemple.1 est de la forme :

$$\boldsymbol{G}(D) = (\boldsymbol{g}^{(1)}(D) \quad \boldsymbol{g}^{(2)}(D)) = (1 + D^2 + D^3 \quad 1 + D + D^3)$$

Ces polynômes générateurs peuvent aussi se résumer par la suite de leurs coefficients, respectivement (1011) et (1101), généralement notée en représentation octale, respectivement $(13)_{octal}$ et $(15)_{octal}$. □

Une classe de codes convolutifs est la classe des codes systématiques. Dans un code systématique, les premières *k* séquences de sortie sont des répliques exactes des *k* séquences d'entrée. Une sous-classe importante de codes convolutifs systématiques est celle des codes convolutifs systématiques récursifs 'Recursive Systematic Convolutional Code' (RSC). Ils jouent un rôle particulier dans les codes concaténés en parallèles, ainsi que dans les codes convolutifs couplés. Définir les polynômes générateurs d'un code systématique récursif est moins évident.

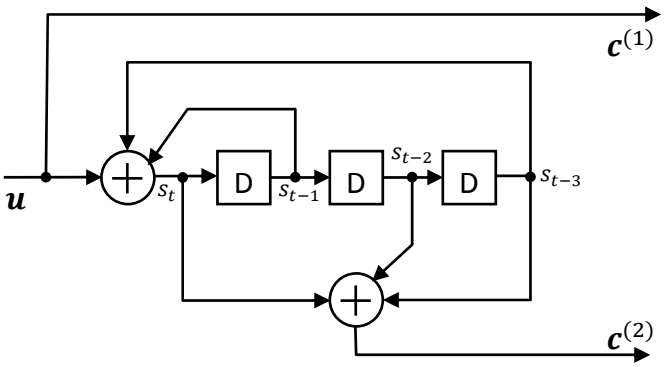

Figure 3.3. Codeur RSC de rendement $R = \frac{1}{2}$

Exemple 2 :

Considérons l'exemple de la figure 3.3. Le premier polynôme générateur est trivial puisque le code est systématique. Pour identifier le second, il faut noter que :

$$s_t = u_t + s_{t-1} + s_{t-3} \implies u_t = s_t + s_{t-1} + s_{t-3}$$
$$c_t^{(2)} = s_t + s_{t-2} + s_{t-3}$$

Ce résultat peut être reformulé en introduisant la forme polynômiale :

$$\boldsymbol{u}(D) = \boldsymbol{s}(D)(1 + D + D^3) \implies \boldsymbol{s}(D) = \frac{\boldsymbol{u}(D)}{1 + D + D^3}$$
$$\boldsymbol{c}^{(1)}(D) = \boldsymbol{u}(D)$$
$$\boldsymbol{c}^{(2)}(D) = \boldsymbol{s}(D)(1 + D^2 + D^3) = \frac{1 + D^2 + D^3}{1 + D + D^3}\boldsymbol{u}(D)$$

Ce qui est équivalent à :

$$\boldsymbol{c}(D) = [\boldsymbol{c}^{(1)}(D) \quad \boldsymbol{c}^{(2)}(D)] = \boldsymbol{u}(D)[\boldsymbol{g}^{(1)}(D) \quad \boldsymbol{g}^{(2)}(D)]$$

Où $\boldsymbol{g}^{(1)}(D) = 1$ et $\boldsymbol{g}^{(2)}(D) = \frac{1+D^2+D^3}{1+D+D^3}$.

Donc on obtient la matrice génératrice du code convolutif RSC sous la forme polynômiale ainsi :

$$\boldsymbol{G}(D) = \left[1 \quad \frac{1 + D^2 + D^3}{1 + D + D^3}\right]$$

3.2.3. Représentation graphique :

Les codes générés par des codeurs convolutifs peuvent être représentés graphiquement selon trois modèles : l'arbre, le treillis et le diagramme d'états. La première représentation graphique est la moins pertinente, pour cela elle n'est pas abordée dans la suite du chapitre.

3.2.3.1. Diagramme d'états :

Un codeur convolutif a un nombre fini m de mémoire et, par conséquent, un nombre fini 2^m d'états de mémoire $\boldsymbol{s}$ et donc le codeur peut être réalisé comme un automate d'état fini.

Le résultat $\boldsymbol{c}_t$ du codeur à l'instant t dépend de l'état de mémoire $\boldsymbol{s}_t$ et du bloc d'informations $\boldsymbol{u}_t$. Au moment $t+1$, le codeur est dans l'état $\boldsymbol{s}_{t+1}$. Chaque modification d'état $\boldsymbol{s}_t \rightarrow \boldsymbol{s}_{t+1}$ est associée à une certaine séquence d'entrée et à une séquence de sortie. Le graphique avec tous les états possibles peut être présenté comme dans l'exemple suivant.

Le codeur convolutif vu précédemment dans la figure.3.3, est défini par la matrice génératrice sous la forme polynômiale $\boldsymbol{G}(D) = \left[1 \quad \frac{1+D^2+D^3}{1+D+D^3}\right]$. Sa mémoire étant de $m = 3$ cellules, son diagramme d'état comprendra $2^3 = 8$ états.

Les bits d'information étant appliqués à l'entrée du codeur, un à la fois, il en résulte deux transitions possibles à chaque instant. A chaque transition correspond aussi 2 bits codés. La figure 3.4 montre le diagramme d'états avec ses transitions ainsi que les bits d'information et les bits codés [21,30].

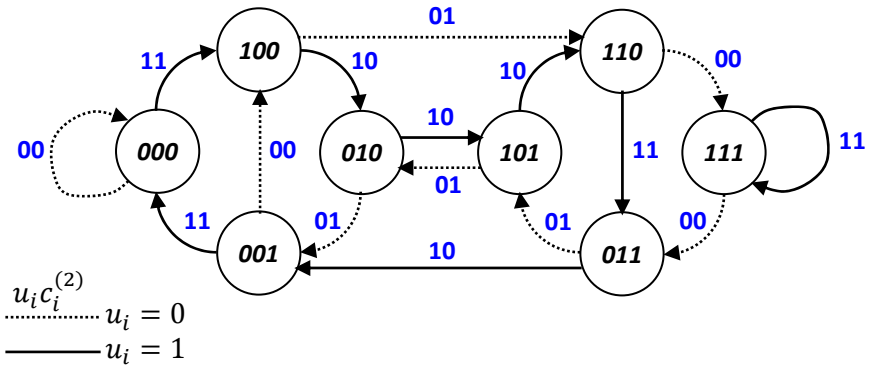

Figure 3.4. Diagramme d'état du codeur convolutif RSC de l'exemple 2 [21].

3.2.3.2. Treillis d'un code :

La représentation la plus courante d'un code convolutif est le diagramme en treillis. Il est d'une importance majeure aussi bien pour la définition des propriétés d'un code que pour son décodage, comme nous le verrons dans la suite du chapitre.

A un instant t, l'état d'un codeur convolutif peut prendre 2^m valeurs différentes. Chacune de ces valeurs possibles est représentée par un nœud. A chaque instant t est associée une colonne d'états-nœuds et en fonction de l'entrée $\boldsymbol{u}_t$, le codeur transite d'un état $\boldsymbol{s}_t$ à un autre $\boldsymbol{s}_{t+1}$ en délivrant les bits codés. Cette transition entre deux états est représentée par un arc entre les deux nœuds associés et étiqueté avec les sorties du codeur. Dans le cas d'un code binaire, la transition sur une entrée à '0' (respectivement '1') est représentée par un trait pointillé (respectivement plein). La succession des états $\boldsymbol{s}_t$ jusqu'à l'instant t' est représentée par les différents chemins entre l'état de départ et les différents états possibles à l'instant t'[21].

Illustrons ceci par l'exemple du codeur systématique RSC de la figure 3.3. En faisant l'hypothèse que l'état de départ $\boldsymbol{s}_0$ est l'état (000) :

- si $u_1 = 0$ alors l'état suivant, $\boldsymbol{s}_1$, est aussi (000). La transition sur une entrée à '0' est représentée par un trait pointillé et étiquetée dans ce premier cas par '00', la valeur des sorties du codeur.
- si $u_1 = 1$, alors l'état suivant, $\boldsymbol{s}_1$, est (100). La transition sur une entrée à '1' est représentée par un trait plein et étiquetée ici par '11'.
- Il faut ensuite envisager les quatre transitions possibles : de $\boldsymbol{s}_1 = (000)$ si $u_2 = 0$ ou $u_2 = 1$, et de $\boldsymbol{s}_1 = (100)$ si $u_2 = 0$ ou $u_2 = 1$.

En itérant cette construction, on aboutit à la représentation, en figure 3.5, de toutes les successions possibles d'états à partir de l'état de départ jusqu'à un instant désiré t' (ici $t' = 5$). Une section complète du treillis suffit à caractériser le code.

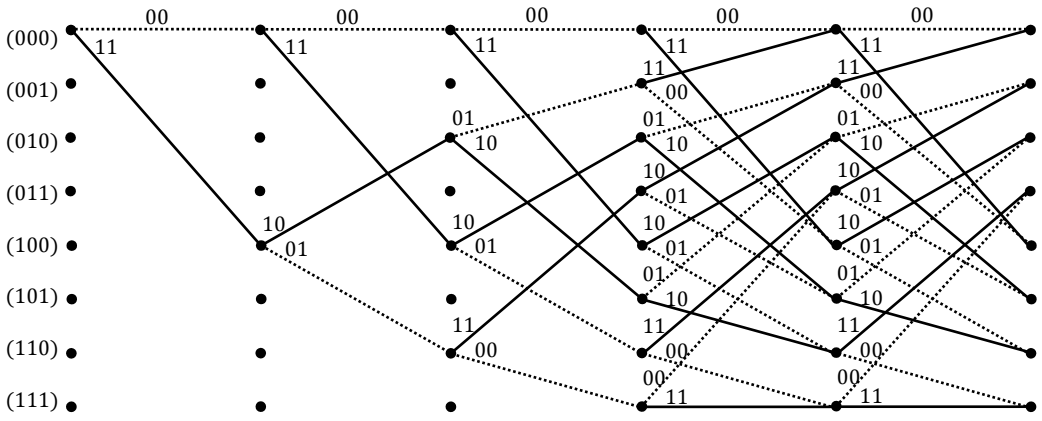

Figure 3.5. Diagramme en treillis du codeur convolutif RSC de l'exemple 2.

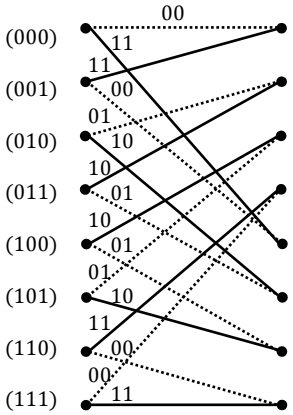

Figure 3.6. Section complète du treillis du codeur convolutif RSC de l'exemple 2.

3.2.3.3. Fermeture de treillis

Les codes convolutifs sont naturellement adaptés aux applications de diffusion où le message transmis est de longueur infinie. Cependant, la plupart des systèmes de télécommunications utilisent des transmissions par trames indépendantes, avec des longueurs K très courtes (quelques dizaines de bits) dans certains cas. Dans le processus de décodage des codes convolutifs, le décodeur doit connaître l'état initial et final du codeur pendant le décodage du bloc. Puisque la structure du codeur convolutif appelle un registre à décalage, l'état initial peut être facilement résolu en réinitialisant le registre. Connaître l'état final n'est pas si simple car sa valeur dépend du message encodé. Les techniques utilisées pour connaître ces états sont généralement appelées *fermeture de treillis*. Elles consistent généralement à forcer les états initial et final à des valeurs connues du décodeur (en général zéro). Plusieurs solutions peuvent être envisagées pour la fermeture des treillis:

Troncature directe (Truncated): La possibilité la plus évidente est d'arrêter le processus de codage lorsque tous les symboles du bloc d'information ont été appliqués à l'entrée du codeur. Ainsi, l'état final est inconnu du côté du décodeur, et la performance de correction d'erreur du code est dégradée en raison d'une plus grande densité d'erreurs de décodage à la fin des blocs (Effet de bord). Cette dégradation est plus forte pour les blocs courts que pour les blocs longs et pourrait être admissible dans certaines applications. Il convient de noter que la troncature directe des treillis a un impact plus négatif sur le taux d'erreur de bloc (FER ou PER) que sur le BER [31].

Terminaison à zéro (Zero-tail): La solution standard pour éviter cette dégradation des performances à la fin du bloc d'informations consiste à ajouter m bits supplémentaires ou des *tail-bits* (de queue) au bloc d'informations afin que l'encodeur récupère l'état zéro, m étant la mémoire du code. Dans le cas de codes convolutifs non récursifs, l'injection de m bits à zéro à la fin du bloc force l'encodeur à terminer à l'état zéro. Tout se passe comme si le bloc codé était de longueur $K + m$. avec $u_{K+1} = u_{K+2} = \cdots = u_{K+m} = 0$. Le rendement de codage est légèrement diminué par la transmission des bits de fermeture. Cependant, compte tenu de la taille des blocs généralement transmis cette dégradation du rendement est bien souvent négligeable.

Dans le cas des codes récursifs, il est aussi possible d'injecter un zéro à l'entrée du registre. La figure 3.7 illustre une méthode simple pour contourner ce problème. En effet, s_t est le résultat d'une somme modulo 2 de deux symboles identiques.

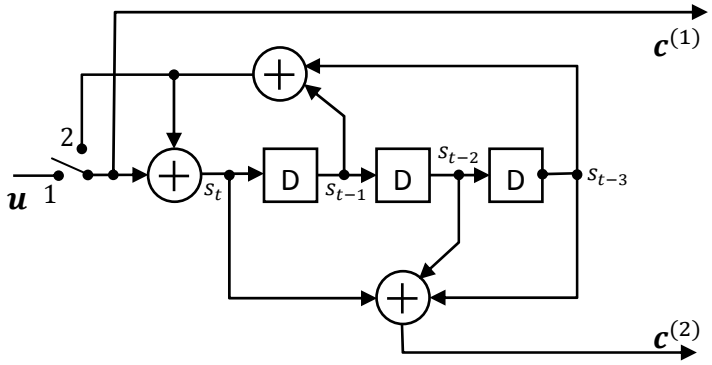

Figure 3.7. Codeur convolutif RSC permettant une terminaison à l'état 0 en m périodes.

Fermeture circulaire (Tail-biting) : Cette technique appelée '*tail-biting*', a été introduite par Howard H. MA et Jack K. WOLF en 1986 [32] pour résoudre le problème de la terminaison du treillis sans introduire d'effets de bord. Elle consiste à rendre le treillis de décodage

circulaire, c'est-à-dire à faire en sorte que l'état de départ et l'état final du codeur soient identiques. Cet état est alors appelé état de circulation.

Parmi les différentes techniques visant à transformer un code convolutif en code en bloc, le tail-biting, est la meilleure méthode de terminaison pour les codes RSC et les turbocodes. Tout d'abord, aucun bit supplémentaire ne doit être ajouté et transmis; ainsi, le rendement de codage est préservé et l'efficacité spectrale de la transmission n'est pas réduite. Ensuite, le tail-biting n'induit aucun effet secondaire dans le message. Par conséquent, tous les bits d'information sont protégés de la même manière par le code convolutif et la propriété circulaire empêche l'apparition de mots de code tronqués à faible poids car le treillis peut être considéré comme de longueur infinie.

L'idée fondamentale derrière le tail-biting est que le codeur est contrôlé de telle sorte qu'il commence et termine l'encodage dans le même état, c'est-à-dire $\boldsymbol{s}_0 = \boldsymbol{s}_N$, où la longueur du message d'information est Nk bits. Dans la représentation en treillis d'un code de fermeture circulaire, seuls les chemins qui commencent et se terminent dans le même état sont des mots de code valides [31].

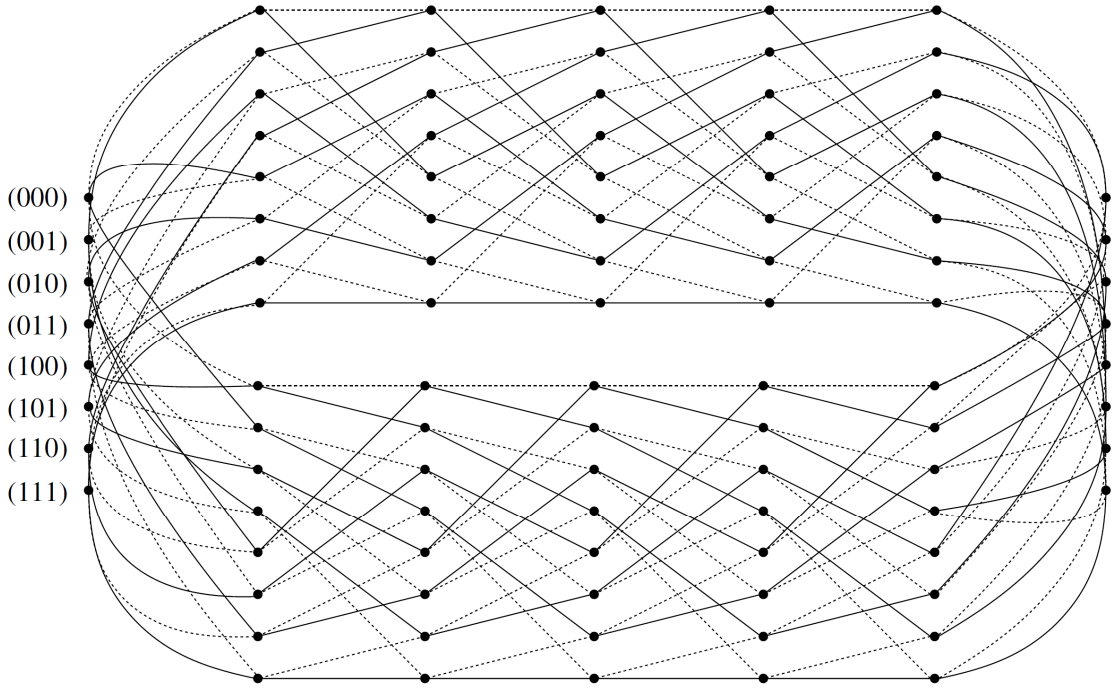

Figure 3.8. Treillis d'un code convolutif RSC circulaire à 8 états [21].

Pour les encodeurs convolutifs non-récursifs, il est facile de satisfaire à la condition de l'état de circulation du tail-biting $\boldsymbol{s}_0 = \boldsymbol{s}_N$, puisque l'état final $\boldsymbol{s}_N$ dépend uniquement des m

derniers k-uplets d'entrée $\boldsymbol{u}_{N-m}, \boldsymbol{u}_{N-m+1}, \dots, \boldsymbol{u}_{N-1}$ à l'encodeur. Pour les encodeurs récursifs, la situation est plus compliquée. Dans ce cas, l'état final $\boldsymbol{s}_N$ dépend de l'ensemble de la séquence d'information $\boldsymbol{u} = \boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_{N-1}$. Ainsi, nous devons calculer pour une séquence d'information donné $\boldsymbol{u}$ l'état initial $\boldsymbol{s}_0$ qui mènera au même état après N cycles [33].

En utilisant la représentation de l'espace d'état, il est possible d'établir une relation entre l'état $\boldsymbol{s}_{t+1}$ du codeur à un instant $t+1$, son état $\boldsymbol{s}_t = [s_t^{(1)}, s_t^{(2)}, \dots, s_t^{(m)}]^T$ et la donnée entrante $\boldsymbol{u}_t = [u_t^{(1)}, u_t^{(2)}, \dots, u_t^{(k)}]^T$ à un instant t :

$$\boldsymbol{s}_{t+1} = \boldsymbol{A}\boldsymbol{s}_t + \boldsymbol{B}\boldsymbol{u}_t \tag{3.12}$$

$$\boldsymbol{c}_t = \boldsymbol{C}\boldsymbol{s}_t + \boldsymbol{D}\boldsymbol{u}_t \tag{3.13}$$

Où $\boldsymbol{A} \in GF(2)^{m\times m}$ est la matrice d'état qui détermine la manière dont les m éléments de mémoire sont connectés. $\boldsymbol{B} \in GF(2)^{m\times k}$ est la matrice d'entrée, elle détermine comment les entrées entraînent les éléments de mémoire. La matrice $\boldsymbol{C} \in GF(2)^{n\times m}$ détermine comment la sortie dépend de l'état actuel. Enfin, la matrice $\boldsymbol{D} \in GF(2)^{n\times k}$ détermine comment les entrées actuelles contribuent à la sortie [34].

La solution complète de (3.12) est calculée par la superposition de la solution d'une séquence d'entrée de zéros $\boldsymbol{s}_t^{[zi]}$ et la solution d'un état initial zéro $\boldsymbol{s}_t^{[zs]}$:

$$\boldsymbol{s}_t = \boldsymbol{s}_t^{[zi]} + \boldsymbol{s}_t^{[zs]} = \boldsymbol{A}^t\boldsymbol{s}_0 + \sum_{j=0}^{t-1} \boldsymbol{A}^{(t-1)-j}\boldsymbol{B}\boldsymbol{u}_j \tag{3.14}$$

Si nous exigeons que l'état à l'instant $t = N$ soit égal à l'état initial $\boldsymbol{s}_0$ (l'état de circulation $\boldsymbol{s}_c = \boldsymbol{s}_0$), on obtient de (3.14) l'équation :

$$(\boldsymbol{A}^N + \boldsymbol{I}_m)\boldsymbol{s}_0 = \boldsymbol{s}_N^{[zs]} \tag{3.15}$$

$\boldsymbol{I}_m \in GF(2)^{m\times m}$ est la matrice identité.

Pourvu que la matrice $(\boldsymbol{A}^N + \boldsymbol{I}_m)$ soit inversible, l'état initial $\boldsymbol{s}_0$ correct peut être calculé en sachant la réponse $\boldsymbol{s}_N^{[zs]}$ (état final) de l'encodeur à l'état initial nul [33].

En conclusion, s'il existe, l'état de circulation s'obtient en deux étapes. La première consiste à coder la trame de k bits en entrée après avoir initialisé le codeur à l'état zéro et à

conserver l'état de terminaison. La seconde se résume à déduire l'état de circulation du précédent état de terminaison et d'une table (obtenue par l'inversion de $(\boldsymbol{A}^N + \boldsymbol{I}_m)$).

3.3. Distance libre et spectre de distances :

Un code étant exploité pour ses capacités de correction, il faut être en mesure de pouvoir les estimer pour choisir un code parmi d'autres de manière judicieuse, en fonction de l'application visée. Parmi les mauvais choix possibles, les *codes catastrophiques*.

Il existe des codes convolutifs appelés ***codes catastrophiques*** pour lesquels un nombre fini d'erreurs à l'entrée du décodeur peut produire un nombre infini d'erreurs en sortie du décodeur, ce qui justifie bien leur appellation. Une propriété majeure de ces codes est qu'il existe au moins une séquence d'informations de poids infini qui, à la sortie du codeur, engendre une séquence codée de poids fini. Il ressort de ce commentaire que les codes systématiques ne peuvent donc pas être catastrophiques [13].

Cependant le choix d'un code convolutif ne peut se résumer à la question « *est-il catastrophique ?* ». En exploitant les représentations graphiques, les propriétés et les performances des codes peuvent être comparées.

3.3.1. Distance libre d'un code convolutif :

La ***distance libre*** d_{free} d'un code convolutif est la ***distance minimale de Hamming*** entre toutes les paires $(\boldsymbol{c}_1, \boldsymbol{c}_2)$ des séquences de codes avec terminaison à zéro du treillis, sachant que $\boldsymbol{c}_1 = \boldsymbol{c}_{1,0}, \boldsymbol{c}_{1,1}, \boldsymbol{c}_{1,2}, \dots, \boldsymbol{c}_{1,N}$,et $\boldsymbol{c}_2 = \boldsymbol{c}_{2,0}, \boldsymbol{c}_{2,1}, \boldsymbol{c}_{2,2}, \dots, \boldsymbol{c}_{2,N}$.

$$d_{free} = \min_{\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C}_N} \sum_{i=0}^{N-1} d_H(\boldsymbol{c}_{1,i}, \boldsymbol{c}_{2,i}) \tag{3.16}$$

Étant donné que les codes convolutifs sont linéaires, la distance libre d'un code convolutif est le poids minimal de Hamming des mots de code non-nuls terminés à l'état zéro. Comme les mots de code terminés doivent commencer et se terminer à l'état zéro, la distance libre peut être définie comme le poids de Hamming minimal des séquences de code $\boldsymbol{c} = \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_N$ sur tous les chemins de treillis qui commencent et se terminent à l'état zéro.

$$d_{free} = \min_{\boldsymbol{c} \in \mathcal{C}_N, \boldsymbol{c} \neq \boldsymbol{0}} \sum_{i=0}^{N-1} w_H(\boldsymbol{c}_i) \tag{3.17}$$

Avec : $\mathcal{C}_N = \{\boldsymbol{c} \in \mathcal{C} | \boldsymbol{s}_0 = \boldsymbol{0},\ \boldsymbol{s}_i \neq \boldsymbol{0}, pour\ 1 \leq i \leq N-1,\ \boldsymbol{s}_N = \boldsymbol{0}\}$

Contrairement aux codes en blocs, de grandes distances minimales (donc de faibles probabilités d'erreur) ne sont pas obtenues pour les codes convolutifs en augmentant la longueur du code. Au lieu de cela, d_{free} peut être augmenté en augmentant la longueur de contrainte v, augmentant ainsi le nombre total d'états du codeur et augmentant par conséquent le nombre d'états par lesquels un chemin de treillis doit passer avant de pouvoir revenir à l'état zéro [21,29].

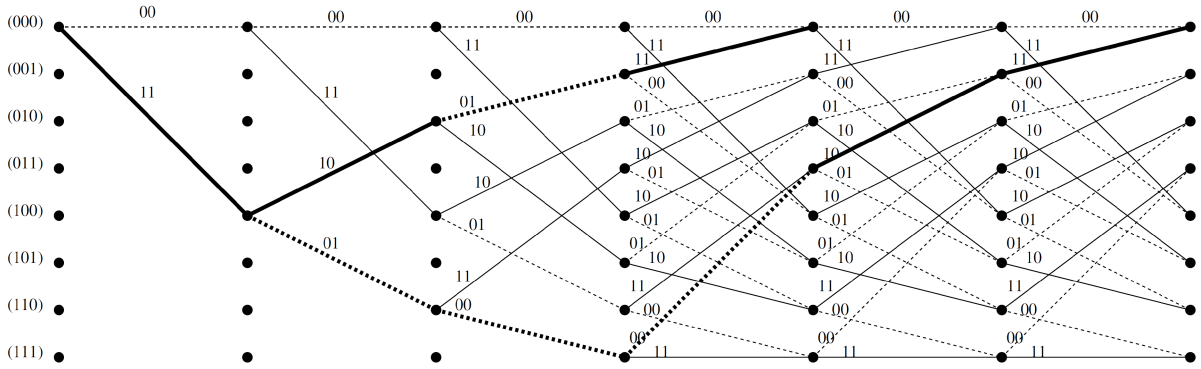

Figure 3.9. Séquences de code (en gras) définissant la distance libre du code de matrice génératrice $\boldsymbol{G}(D) = \left[1 \quad \frac{1+D^2+D^3}{1+D+D^3}\right]$

3.3.2. Spectre des distances et Fonction de transfert :

L'énumération des distances, au delà de la distance minimale de Hamming, et de leurs poids est nécessaire pour évaluer finement la probabilité d'erreur. Cette énumération porte le nom de spectre des distances. Par définition, tous les chemins possibles du diagramme de transitions d'états, qui commencent et se terminent dans l'état zéro tout en supprimant la transition *selfloop* à cet état, déterminent le spectre de distance d'un codeur convolutif [30].

Ce spectre de distance est important, puisque le pouvoir de correction d'un code dépend de toutes ces séquences qui commencent et se terminent à l'état zéro (séquence *Return To Zero*), que l'on va considérer suivant l'ordre croissant de leurs poids. Il est possible d'établir la fonction de transfert du code. Celle-ci s'obtient à partir du diagramme de transitions d'états dans lequel l'état de départ ($\boldsymbol{s}_0 = \boldsymbol{0}$) est scindé en deux états $\boldsymbol{s}_0^{in}$ et $\boldsymbol{s}_0^{out}$, qui ne sont autres que l'état de départ et l'état d'arrivée de toute séquence *RTZ*.

Nous allons maintenant obtenir une expression de la fonction de transfert dont l'expansion donne le spectre de distance de tous ces chemins. La méthode, qui est due à Viterbi, est mieux expliquée par un exemple [21].

Illustrons le calcul de la fonction de transfert par le cas du code RSC de la figure 3.3, dont le diagramme de transitions d'états est de nouveau représenté en figure 3.10.

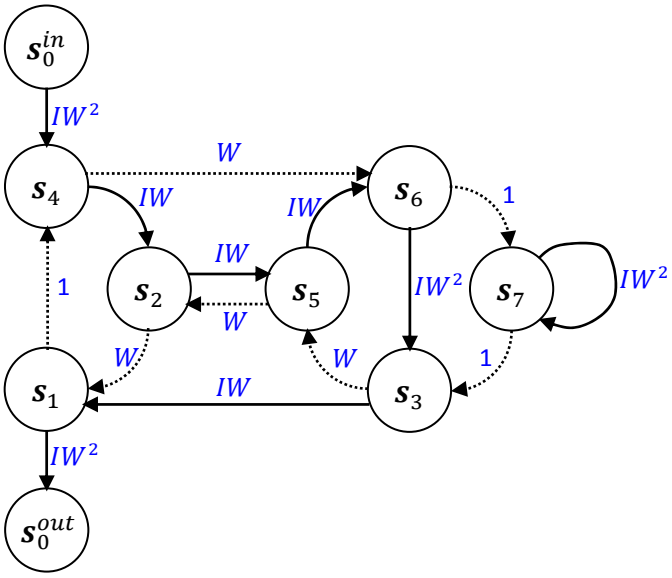

Figure 3.10. Diagramme d'états du codeur RSC modifiée pour le calcul de la fonction de transfert

Chaque transition possède une étiquette $I^i W^j$, où j est le poids de la séquence codée et i celui de la séquence en entrée du codeur. Dans notre exemple, i peut prendre la valeur 0 ou 1 selon le bit en entrée du codeur à chaque transition et j varie entre 0 et 2, puisque 4 symboles codés sont possibles (00, 01, 10, 11), de poids compris entre 0 et 2.

La fonction de transfert du code $\mathcal{T}(I, W)$ est alors définie par :

$$\mathcal{T}(I, W) = \frac{\boldsymbol{s}_0^{out}}{\boldsymbol{s}_0^{in}} \tag{3.18}$$

Pour établir cette fonction, il faut résoudre le système d'équations issu des relations entre les huit états $(\boldsymbol{s_1}, \boldsymbol{s_2}, \ldots, \boldsymbol{s_7}, \boldsymbol{s_0^{out}})$ en fonction de $\boldsymbol{s_0^{in}}$. De la figure 3.10 on peut écrire :

$$\begin{aligned}
\boldsymbol{s}_1 &= W\boldsymbol{s}_2 + IW\boldsymbol{s}_3 \\
\boldsymbol{s}_2 &= IW\boldsymbol{s}_4 + W\boldsymbol{s}_5 \\
\boldsymbol{s}_3 &= IW^2\boldsymbol{s}_6 + \boldsymbol{s}_7 \\
\boldsymbol{s}_4 &= IW^2\boldsymbol{s}_0^{in} + \boldsymbol{s}_1 \\
\boldsymbol{s}_5 &= IW\boldsymbol{s}_2 + W\boldsymbol{s}_3 \\
\boldsymbol{s}_6 &= W\boldsymbol{s}_4 + IW\boldsymbol{s}_5 \\
\boldsymbol{s}_7 &= \boldsymbol{s}_6 + IW^2\boldsymbol{s}_7
\end{aligned} \tag{3.19}$$

Et :

$$\boldsymbol{s}_0^{out} = IW^2\boldsymbol{s}_1 \tag{3.20}$$

Le système d'équations (3.19) peut être réécrit sous forme matricielle comme :

$$\begin{bmatrix} 1 & -W & -IW & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -IW & -W & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -IW^2 & -1 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -IW & -W & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -W & -IW & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1-IW^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ \boldsymbol{s}_3 \\ \boldsymbol{s}_4 \\ \boldsymbol{s}_5 \\ \boldsymbol{s}_6 \\ \boldsymbol{s}_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ IW^2\boldsymbol{s}_0^{in} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.21}$$

À l'aide d'un outil de calcul formel pour résoudre le système s'équations linéaires, il est aisé d'aboutir au résultat suivant :

$$\boldsymbol{s}_1 = \frac{(-I^7+2I^5-I^3)W^{10}+(I^6-3I^4+I^2)W^8+(-I^5+4I^3-I)W^6-2I^2W^4}{(I^6-2I^4+I^2)W^8+(-I^5+4I^3-I)W^6+(I^4-6I^2+1)W^4+5IW^2-1}\boldsymbol{s}_0^{in} \tag{3.22}$$

En combinant (3.20) et (3.22), nous trouvons :

$$\mathcal{T}(I,W) = \frac{\boldsymbol{s}_0^{out}}{\boldsymbol{s}_0^{in}} = \frac{(-I^8+2I^6-I^4)W^{12}+(I^7-3I^5+I^3)W^{10}+(-I^6+4I^4-I^2)W^8-2I^3W^6}{(I^6-2I^4+I^2)W^8+(-I^5+4I^3-I)W^6+(I^4-6I^2+1)W^4+5IW^2-1} \tag{3.23}$$

La fonction de transfert $\mathcal{T}(I,W)$ peut ensuite être développée en série de puissances donnée comme suit [21,30]:

$$\begin{aligned}
\mathcal{T}(I,W) = \quad & 2I^3W^6 \\
& +(I^6+8I^4+I^2)W^8 \\
& +(8I^7+33I^5+8I^3)W^{10} \\
& +(I^{10}+47I^8+145I^6+47I^4+I^2)W^{12} \\
& +(14I^{11}+254I^9+649I^7+254I^5+14I^3)W^{14} \\
& +\cdots
\end{aligned} \tag{3.24}$$

Cette écriture permet d'observer que deux séquences *RTZ* de poids 6 sont produites par deux séquences en entrée de poids 3, que des séquences *RTZ* de poids 8 sont produites par une séquence de poids 6, par huit séquences de poids 4 et par une séquence de poids 2, etc.

La notion de fonction de transfert est très efficace pour étudier les performances d'un code convolutif. Une version dérivée est par ailleurs essentielle à la classification des codes vis-à-vis de leurs performances. Il s'agit du spectre de distance $\omega(d)$ dont la définition est la suivante :

$$\left(\frac{\partial \mathcal{T}(I,W)}{\partial I}\right)_{I=1} = \sum_{d=d_{free}}^{\infty} \omega(d) W^d \tag{3.25}$$

Par exemple, les premiers termes du spectre du code systématique récursif RSC, obtenus à partir de (3.24), sont présentés dans le tableau 3.1. Ce spectre est essentiel pour l'estimation des performances des codes en termes de calcul de probabilité d'erreur.

d	6	8	10	12	14	……
$\omega(d)$	6	40	245	1446	8295	……

Tableau 3.1. Premiers termes du spectre de distance du code RSC de matrice génératrice

$$\boldsymbol{G}(D) = \left[1 \quad \frac{1+D^2+D^3}{1+D+D^3}\right]$$

3.4. Le décodage des codes convolutifs :

Un avantage des codes convolutifs est la disponibilité d'algorithmes de décodage très efficaces fonctionnant sur le diagramme de treillis du code. Deux algorithmes sont principalement utilisés, soit l'algorithme de Viterbi et l'algorithme MAP ou BCJR (doit son nom à ses auteurs Bahl, Cocke, Jelinek et Raviv) publiés dans [4] et [5] respectivement. Ce dernier sert essentiellement dans des algorithmes de décodage itératif.

L'algorithme de Viterbi est un décodeur au sens du maximum de vraisemblance (ML). Chaque chemin dans le treillis correspond à un mot de code, et ainsi le décodeur ML (qui trouve le mot de code le plus probable) recherche le chemin le plus probable dans le treillis connaissant le mot reçu en sortie d'un canal sans mémoire. Pour cela, les branches du treillis du code convolutif sont pondérées par leurs probabilités relatives au mot reçu.

En revanche dans l'algorithme MAP (maximum a posteriori probability), chaque branche du treillis peut correspondre à une entrée particulière: le décodeur MAP au niveau du bit, qui recherche l'information la plus probablement codée et non le mot de code le plus probable, calcule la probabilité de chaque branche de treillis.

Le décodeur reçoit une version bruité du mot de code transmis à partir duquel il détermine l'information. Nous désignerons l'information original par :

$$\begin{aligned} \boldsymbol{u} &= [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_T] = [u_1, u_2, \dots, u_K] \\ &= \left[\mathrm{u}_1^{(0)}, \mathrm{u}_1^{(1)}, \dots, \mathrm{u}_1^{(k-1)}, \mathrm{u}_2^{(0)}, \dots, \mathrm{u}_2^{(k-1)}, \dots, \mathrm{u}_T^{(0)}, \dots, \mathrm{u}_T^{(k-1)}\right] \end{aligned} \tag{3.26}$$

Soit le nombre de bits dans l'information $K^* = kT$, où T est le nombre total de transitions d'état de l'encodeur et k le nombre de bits d'entrée par transition d'état.

Le mot de code résultant est désigné par :

$$\begin{aligned} \boldsymbol{c} &= [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{T+m}] = [c_1, c_2, \dots, c_N] \\ &= \left[\mathrm{c}_1^{(0)}, \mathrm{c}_1^{(1)}, \dots, \mathrm{c}_1^{(n-1)}, \mathrm{c}_2^{(0)}, \dots, \mathrm{c}_2^{(n-1)}, \dots, \mathrm{c}_{T+m}^{(0)}, \dots, \mathrm{c}_{T+m}^{(n-1)}\right] \end{aligned} \tag{3.27}$$

Où le nombre de bits dans le mot de code, $N = n(T + m)$, avec n le nombre de bits par mot de code par transition d'état, et m transitions pour une terminaison à zéro du treillis (zero-tail).

Nous allons désigner les valeurs bruités reçues d'un canal sans mémoire à entrée binaire (souvent AWGN) par :

$$\begin{aligned} \boldsymbol{r} &= [\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_{T+m}] = [r_1, r_2, \dots, r_N] \\ &= \left[\mathrm{r}_1^{(0)}, \mathrm{r}_1^{(1)}, \dots, \mathrm{r}_1^{(n-1)}, \mathrm{r}_2^{(0)}, \dots, \mathrm{r}_2^{(n-1)}, \dots, \mathrm{r}_{T+m}^{(0)}, \dots, \mathrm{r}_{T+m}^{(n-1)}\right] \end{aligned} \tag{3.28}$$

3.4.1. L'algorithme de Viterbi (ML) :

L'avantage de l'algorithme de Viterbi est qu'il n'est pas nécessaire de calculer les probabilités des branches le long du chemin complet pour tous les mots de code possibles. À chaque nœud du treillis, le meilleur chemin jusqu'ici, appelé chemin survivant (*survivor path*) pour ce nœud, est stocké et les chemins restants sont rejetés. Cela réduit considérablement la complexité de trouver le mot de code à vraisemblance maximale (ML) pour les codes convolutifs.

Le décodeur ML sortira le mot de code estimé $\tilde{\boldsymbol{c}}$ qui maximise la vraisemblance logarithmique de la fonction $\log p(\boldsymbol{r}|\boldsymbol{c})$ sur tous les mots de code possibles $\boldsymbol{c}$ sur un treillis de $T+m$ transitions d'état. Puisque nous avons un code convolutif sur un canal sans mémoire :

$$p(\boldsymbol{r}|\boldsymbol{c}) = \prod_{i=1}^{T+m} p(\mathbf{r}_i|\mathbf{c}_i) = \prod_{i=1}^{N} p(r_i|c_i) \tag{3.29}$$

Il s'ensuit que :

$$\log p(\boldsymbol{r}|\boldsymbol{c}) = \sum_{i=1}^{T+m} \log p(\mathbf{r}_i|\mathbf{c}_i) = \sum_{i=1}^{N} \log p(r_i|c_i) \tag{3.30}$$

Où $p(r_i|c_i)$ est une probabilité de transition du canal. Ceci est une règle de décodage de probabilité d'erreur minimale lorsque tous les mots de code sont équiprobables.

La fonction de vraisemblance logarithmique $\log p(\boldsymbol{r}|\boldsymbol{c})$ notée par $M(\boldsymbol{r}|\boldsymbol{c})$ est appelée la métrique associée au chemin (mot de code) $\boldsymbol{c}$. Les termes $\log p(\mathbf{r}_i|\mathbf{c}_i)$ dans l'équation (3.30) sont appelés métriques de branches et sont notés $M(\mathbf{r}_i|\mathbf{c}_i)$, tandis que les termes $\log p(r_i|c_i)$ sont appelés métriques de bits et sont notés $M(r_i|c_i)$. Donc on peut écrire la métrique du chemin $M(\boldsymbol{r}|\boldsymbol{c})$ comme :

$$M(\boldsymbol{r}|\boldsymbol{c}) = \sum_{i=1}^{T+m} M(\mathbf{r}_i|\mathbf{c}_i) = \sum_{i=1}^{N} M(r_i|c_i) \tag{3.31}$$

On peut maintenant exprimer la métrique partielle d'un chemin pour ses t premières branches par :

$$M([\boldsymbol{r}|\boldsymbol{c}]) = \sum_{i=1}^{t} M(\mathbf{r}_i|\mathbf{c}_i) = \sum_{i=1}^{nt} M(r_i|c_i) \tag{3.32}$$

L'algorithme suivant, lorsqu'il est appliqué à la séquence reçue $\boldsymbol{r}$, trouve le chemin à travers le treillis avec la plus grande métrique, c'est-à-dire le chemin de vraisemblance maximum (mot de code). L'algorithme traite $\boldsymbol{r}$ d'une manière récursive. A chaque unité de temps, il ajoute 2^k métriques de branche à chaque métrique de chemin précédemment mémorisée (l'opération addition), il compare les métriques de tous les 2^k chemins entrant dans chaque état (opération de comparaison) et sélectionne le chemin survivant avec la plus grande métrique (l'opération de sélection). Le survivant à chaque état est ensuite mémorisé

avec sa métrique. Dans le cas de métriques identiques pour deux chemins dans un nœud, le chemin du survivant est choisi au hasard. Lorsque le treillis est terminé, il n'y aura qu'un seul chemin survivant final à l'instant $T+m$. Ce chemin correspond au mot de code de vraisemblance maximum ML.

Etapes de l'algorithme de Viterbi [11,14]:

Etape1 : En commençant à l'indice de temps $t=m$, calculez la métrique partielle pour le chemin unique entrant dans chaque état. Mémorisez le chemin survivant et sa métrique pour chaque état.

Etape 2 : Augmentez t de 1. Calculez la métrique partielle pour tous les 2^k chemins entrant dans un état en ***ajoutant*** la métrique de branche entrant dans cet état à la métrique du chemin survivant connecté à l'état précédent de cette branche. Pour chaque état, ***comparez*** les métriques de tous les 2^k chemins entrant dans cet état, ***sélectionnez*** le chemin survivant, celui qui possède la métrique la plus grande, stockez-le avec sa métrique et éliminez tous les autres chemins.

Etape 3 : Si $t<T+m$, répétez l'étape 2. Sinon, arrêtez.

Le calcul effectué par l'algorithme de Viterbi est basé essentiellement sur les opérations d'addition, comparaison et sélection de l'étape 2. En pratique, la séquence d'informations correspondant au chemin survivant à chaque état, plutôt que le chemin survivant lui-même, est stockée dans les étapes 1 et 2, éliminant ainsi la nécessité d'inverser le mot de code estimé $\tilde{\boldsymbol{c}}$ pour récupérer la séquence d'informations estimée $\tilde{\boldsymbol{u}}$ lorsque l'algorithme se termine.

Il y a 2^m survivants en allant de l'indice de temps m jusqu'à l'indice T, un pour chacun des 2^m états. Après l'indice de temps T, il y a moins de survivants, puisqu'il y a moins d'états alors que le codeur retourne à l'état nul. Enfin, à l'indice de temps $T+m$, il n'y a qu'un seul état, l'état nul, et donc un seul survivant, et l'algorithme se termine. Ce dernier survivant est le chemin du maximum de vraisemblance.

3.4.2. L'Algorithme MAP :

L'algorithme MAP permet de calculer la probabilité a posteriori de chaque bit d'information ou de chaque symbole transmis et le décodeur correspondant sélectionne à chaque instant le bit ou le symbole le plus probable. La portée de cette méthode de décodage

est restée confidentielle jusqu'à la découverte des turbocodes car elle n'apporte pas d'amélioration de performance notable par rapport à l'algorithme de Viterbi pour le décodage des codes convolutifs et s'avère plus complexe à mettre en œuvre. En revanche la situation a changé en 1993 car le décodage des turbocodes fait appel à des décodeurs élémentaires à sorties pondérées ou souples et l'algorithme MAP, contrairement à l'algorithme de Viterbi, permet d'associer naturellement une pondération à chaque décision [21].

Rappelons qu'un symbole d'information entrant est constitué de k symboles binaires, $k = 1$ pour le cas binaire (notre cas), c.à.d. que le bloc d'information est d'une longueur $K = kT = T$. Considérons que le décodeur reçoit une séquence bruitée, ayant une longueur N.

Soit S_t l'état de l'encodeur à l'instant t. Le bit u_t est associé à la transition de l'instant $t-1$ à l'instant t. Les états de treillis au niveau $t-1$ et au niveau t sont respectivement indexés par les entiers s' et s. De plus, la transition $S_{t-1} \rightarrow S_t$ dépend seulement du symbole d'entrée u_t.

L'algorithme MAP donne, pour chaque bit décodé u_t, la probabilité que ce bit soit +1 ou -1, sachant la séquence de symboles reçue $\boldsymbol{r}$. Cela revient à trouver le logarithme de rapport de vraisemblance LLR (Log-Likelihood Ratio) a posteriori $L(u_t|\boldsymbol{r})$, où [35]:

$$L(u_t|\boldsymbol{r}) = \ln\left(\frac{P(u_t = +1|\boldsymbol{r})}{P(u_t = -1|\boldsymbol{r})}\right) \tag{3.33}$$

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

$$L(u_t|\boldsymbol{r}) = \ln\left(\frac{P(u_t = +1, \boldsymbol{r})}{P(u_t = -1, \boldsymbol{r})}\right) \tag{3.34}$$

Considérons maintenant la figure 3.6 montrant les transitions possibles pour le code RSC de $m = 3$ montré dans la figure 3.3. Pour ce code, il y a huit états de l'encodeur, et puisque nous considérons un code binaire, dans chaque état de l'encodeur deux transitions sont possibles, en fonction de la valeur de ce bit. Une de ces transitions est associée au symbole d'entrée -1 représenté par une ligne continue (équivalent au bit 1), tandis que l'autre correspond au symbole d'entrée +1 affiché par des pointillés (équivalent au bit 0). On peut voir sur la figure 3.6 que si l'état précédent S_{t-1} et l'état actuel S_t sont connus, alors la valeur du bit d'entrée u_t, qui a provoqué la transition entre ces deux états, sera connue. D'où la probabilité que $u_t = +1$ est égale à la probabilité que la transition de l'état précédent S_{t-1} à

l'état actuel S_t, est une de l'ensemble des huit transitions possibles lorsque $u_t = +1$ (les transitions montrées avec des pointillés). Cet ensemble de transitions est mutuellement exclusif (c'est-à-dire qu'une seule d'entre elles aurait pu se produire au niveau de l'encodeur), et donc la probabilité que l'une d'entre elles se produise est égale à la somme de leurs probabilités individuelles. Par conséquent, nous pouvons réécrire l'équation (3.34) comme [35]:

$$L(u_t|\boldsymbol{r}) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_t=+1} P(S_{t-1}=s', S_t=s, \boldsymbol{r})}{\sum_{(s',s)\Rightarrow u_t=-1} P(S_{t-1}=s', S_t=s, \boldsymbol{r})}\right) \tag{3.35}$$

Où $(s', s) \Rightarrow u_t = +1$ est l'ensemble des transitions de l'état précédent $S_{t-1} = s'$ à l'état actuel $S_t = s$ qui peut se produire si le bit d'entrée $u_t = +1$, et de même pour $(s', s) \Rightarrow u_t = -1$. Pour plus de brièveté nous écrirons $P(S_{t-1} = s', S_t = s, \boldsymbol{r})$ comme $P(s', s, \boldsymbol{r})$.

En utilisant la loi de Bayes de $P(a, b) = P(a|b)P(b)$. En supposant un canal de transmission sans mémoire, alors la future séquence reçue $\boldsymbol{r}_{j>t}$ ne dépendra que de l'état actuel s et non pas de l'état précédent s' ou des séquences présentes et précédentes reçus du canal $\boldsymbol{r}_t$ et $\boldsymbol{r}_{j<t}$. La probabilité conjointe $P(s', s, \boldsymbol{r})$ peut être écrite comme le produit de trois probabilités indépendantes [18]:

$$\begin{aligned} P(s', s, \boldsymbol{r}) &= P\left(s', \boldsymbol{r}_{j<t}\right) \cdot P(s, \boldsymbol{r}_t|s') \cdot P\left(\boldsymbol{r}_{j>t}|s\right) \\ &= \underbrace{P\left(s', \boldsymbol{r}_{j<t}\right)}_{=\alpha_{t-1}(s')} \cdot \underbrace{P(s|s') \cdot P(\boldsymbol{r}_t|\{s', s\})}_{\cdot\gamma_t(s', s)} \cdot \underbrace{P\left(\boldsymbol{r}_{j>t}|s\right)}_{\cdot\beta_t(s)} \end{aligned} \tag{3.36}$$

$\alpha_{t-1}(s')$: la probabilité que le treillis soit à l'état s' à l'instant $t-1$ et que la séquence reçue jusqu'à ce point soit $\boldsymbol{r}_{j<t}$, ou tout simplement la métrique de nœud dans le sens *Aller (forward)*.

$\beta_t(s)$: la probabilité que la future séquence reçue sera $\boldsymbol{r}_{j>t}$, étant donné que le treillis est dans l'état s à l'instant t, ou tout simplement la métrique de nœud dans le sens *Retour (backward)*.

$\gamma_t(s', s)$: la probabilité que le treillis passe à l'état s et la séquence reçue pour cette transition est $\boldsymbol{r}_t$, étant donné que le treillis était dans l'état s' à l'instant $t-1$, ou tout simplement la métrique de branche du treillis.

Les probabilités récurrentes des métriques de nœuds dans le sens *Aller* et *Retour* de l'algorithme MAP sont calculées de la manière suivante [18] :

$$\begin{aligned} \alpha_t(s) &= \sum_{s'} \gamma_t(s', s)\alpha_{t-1}(s') \\ \beta_{t-1}(s') &= \sum_{s} \gamma_t(s', s)\beta_t(s) \end{aligned} \tag{3.37}$$

Ainsi, une fois que les valeurs de $\gamma_t(s', s)$ sont connues, les valeurs de $\alpha_t(s)$ et $\beta_{t-1}(s')$ peuvent être calculées récursivement. En supposant que le treillis a l'état initial $S_0 = 0$, de plus il est terminé à zéro (zero-tail), donc les conditions initiales de cette récursivité sont :

$\alpha_0(S_0 = 0) = \beta_N(S_N = 0) = 1$
$\alpha_0(S_0 = s) = \beta_N(S_N = s) = 0 \quad pour\ tous\ s \neq 0$

Chaque fois qu'une transition entre s' et s existe, les probabilités de transition de branche sont données par :

$$\begin{aligned} \gamma_t(s', s) &= P(\boldsymbol{r}_t|\{s', s\})P(s|s') \\ &= P(\boldsymbol{r}_t|\boldsymbol{c}_t)P(u_t) \end{aligned} \tag{3.38}$$

En utilisant la définition du LLR (équation (1.19)), on a :

$$\begin{aligned} P(u_t = \pm 1) &= \left(\frac{e^{-\frac{L(u_t)}{2}}}{1 + e^{-L(u_t)}}\right) e^{\frac{u_t L(u_t)}{2}} = A_t e^{\frac{u_t L(u_t)}{2}} \\ P(\boldsymbol{r}_t|\boldsymbol{c}_t) &= B_t \exp\left(\frac{1}{2} L_c r_t^{(0)} u_t + \frac{1}{2}\sum_{i=1}^{n-1} L_c r_t^{(i)} c_t^{(i)}\right) \end{aligned} \tag{3.39}$$

Les termes A_t et B_t sont équivalent pour toutes les transitions depuis le nœud $t-1$ au nœud t, et s'annulent donc dans le rapport de l'équation (3.35). Par conséquent, la métrique de transition de branche à utiliser dans (3.37) se réduit à l'expression [18,35]:

$$\gamma_t(s', s) = \exp\left(\frac{1}{2} u_t \left(L_c r_t^{(0)} + L(u_t)\right)\right) \gamma_t^e(s', s) \tag{3.40}$$

Avec : $\gamma_t^e(s', s) = \exp\left(\frac{1}{2}\sum_{i=1}^{n-1} L_c r_t^{(i)} c_t^{(i)}\right)$.

On remplaçant (3.40) et (3.36) dans (3.35), le $L(u_t|\boldsymbol{r})$ s'écrira donc par :

$$L(u_t|\boldsymbol{r}) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_t=+1}\alpha_{t-1}(s')\gamma_t(s',s)\beta_t(s)}{\sum_{(s',s)\Rightarrow u_t=-1}\alpha_{t-1}(s')\gamma_t(s',s)\beta_t(s)}\right)$$
$$= L_c r_t^{(0)} + L(u_t) + \ln\left(\frac{\sum_{(s',s)\Rightarrow u_t=+1}\alpha_{t-1}(s')\gamma_t^e(s',s)\beta_t(s)}{\sum_{(s',s)\Rightarrow u_t=-1}\alpha_{t-1}(s')\gamma_t^e(s',s)\beta_t(s)}\right) \quad (3.41)$$

Ainsi, nous avons montré que l'algorithme MAP pour les codes systématiques, a la structure de l'équation (1.26).

Un décodeur qui implémente le critère MAP symbole par symbole, calcule pour tous les bits d'information la valeur binaire $\hat{u}_t$ qui maximise la probabilité a posteriori $P(u_t = +1|\boldsymbol{r})$. Par conséquent, un décodeur MAP qui procède symbole par symbole minimise réellement le taux d'erreurs binaire. Ainsi, nous pouvons formuler la règle de décodage MAP symbole par symbole en termes de rapport de vraisemblance $L(\hat{u}_t) = L(u_t|\boldsymbol{r})$, et obtenir :

$$\hat{u}_t = \begin{cases} 0 & si\ \ L(\hat{u}_t) \geq 0 \\ 1 & si\ \ L(\hat{u}_t) < 0 \end{cases} \quad (3.42)$$

Nous observons que la décision ferme de décodage MAP du symbole peut être basée sur le signe de la valeur LLR, et la quantité $|L(\hat{u}_t)|$ est la fiabilité de cette décision sous forme de la vraisemblance conditionnellement à l'entrée $\boldsymbol{r}$.

3.5. Les Turbocodes :

Une classe de codes correcteurs d'erreur décodables itérativement, différente des codes LDPC qui ont été présentés dans le chapitre précédent, est les Turbocodes. Le reste de ce chapitre, nous considérons cette classe importante, en présentant les structures des encodeurs et décodeurs turbo ainsi que leurs algorithmes de décodage itératif.

Le codage turbo est basé sur deux idées fondamentales : la conception d'un code avec des propriétés aléatoires et la conception d'un décodeur qui exploite le décodage itératif facile à mettre en œuvre. Les turbocodes ont une performance exceptionnellement bonne, en particulier avec des taux d'erreur de bit modérés et pour des longueurs de blocs importantes.

3.5.1. Principes fondamentaux du turbocodeur :

Le turbocodeur implique une concaténation parallèle d'au moins deux codes convolutifs systématiques récursifs élémentaires (RSC) séparés par un entrelaceur Π. La séquence d'information est codée deux fois, avec un entrelaceur entre les deux codeurs servant à rendre

les deux séquences de données codées approximativement statistiquement indépendantes l'une de l'autre.

La figure 3.11 représente un turbocode dans sa version la plus classique. Le message binaire d'entrée est codé dans son ordre naturel et dans un ordre entrelacé (permuté) par deux codeurs convolutifs récursifs RSC1 et RSC2. Sur la figure, les encodeurs RSC sont identiques mais cette propriété n'est pas indispensable. Le taux de codage global R du turbo code est $R = \frac{1}{3}$. Pour obtenir des taux plus élevés, la technique la plus courante consiste à poinçonner (perforer), c'est-à-dire à ne pas transmettre, une partie des symboles codés, le plus souvent des symboles de redondance $\boldsymbol{p}^{(1)}$ et $\boldsymbol{p}^{(2)}$.

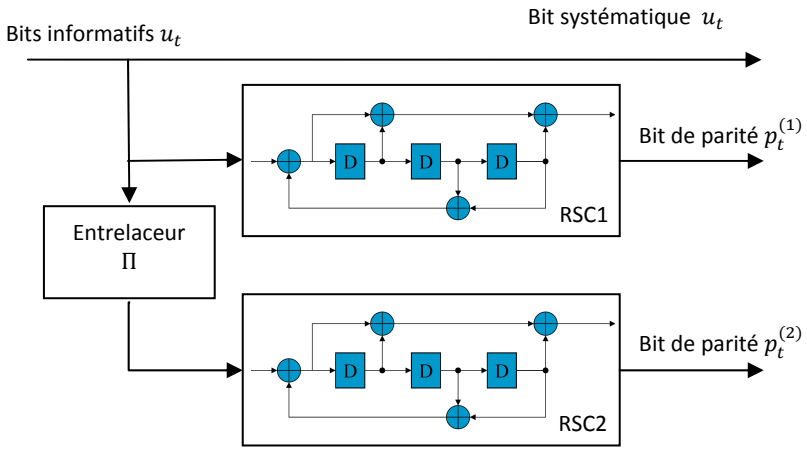

Figure 3.11. La structure du turbocodeur : une concaténation parallèle de deux codeurs RSC $\left(\boldsymbol{G}(D) = \left[1 \quad \frac{1+D+D^3}{1+D^2+D^3}\right]\right)$ séparés par un entrelaceur Π

Le message binaire d'entrée $\boldsymbol{u}$ de longueur K est codé directement par le premier codeur RSC1, ce qui produit les bits de parité $\boldsymbol{p}^{(1)}$; cependant, il est entrelacé avant d'être codé par le second codeur convolutif, ce qui produit les bits de parité $\boldsymbol{p}^{(2)}$. Les codes turbo sont des codes systématiques; ainsi les bits $\boldsymbol{u}$, $\boldsymbol{p}^{(1)}$ et $\boldsymbol{p}^{(2)}$ sont tous transmis et le mot de code turbo est :

$$\boldsymbol{c} = \left[u_1, p_1^{(1)}, p_1^{(2)}, u_2, p_2^{(1)}, p_2^{(2)}, \dots, u_K, p_K^{(1)}, p_K^{(2)}\right] \tag{3.43}$$

Les tout premiers turbocodes ont été conçus pour coder et décoder des séquences de données continues. Cependant, la plupart des systèmes de télécommunication utilisent des transmissions en bloc indépendantes, avec des longueurs très grandes ou modérées et même très courtes dans certains cas. Dans le processus de décodage des codes convolutifs, le décodeur doit connaître l'état initial et l'état final du codeur lors du décodage du bloc. Puisque

la structure du codeur convolutif appelle un registre à décalage, l'état initial peut être facilement fixé en réinitialisant le registre. Connaître l'état final n'est pas si facile puisque sa valeur dépend du message codé. Les différentes techniques permettant de résoudre ce problème sont appelées techniques de terminaison de treillis (voir section 3.2.3.3). Pour les turbocodeurs, les treillis des deux composants RSC doivent être terminés. Plusieurs solutions peuvent être envisagées :

Troncature directe : Dégradation de la performance globale de correction d'erreur du code en raison d'une plus grande densité d'erreurs de décodage à l'extrémité du bloc. Cette dégradation est plus forte pour les blocs courts que pour les blocs longs et pourrait être admissible dans certaines applications.

Terminaison à zéro : pour les turbocodes, le treillis d'un ou des deux composants RSC peut être terminé en utilisant des bits de fermeture (tail-bits). Les normes CCSDS, UMTS et LTE ont adopté cette technique. Un premier inconvénient lorsque la terminaison à zéro est utilisée par le turbocode est que les bits de fermeture utilisés pour terminer le treillis d'un composant RSC ne sont pas codés par l'autre composant RSC. En d'autres termes, ils ne sont pas codés en turbo. Ils sont donc moins protégés que les bits de données réguliers. Ceci conduit, mais à un degré moindre, à des inconvénients similaires à la troncation directe. De plus, les bits de fermeture doivent être transmis. Ceci entraîne une diminution du taux de codage et de l'efficacité spectrale, heureusement non significative, à l'exception de très courtes longueurs de blocs d'information [31].

Terminaison automatique du treillis: L'entrelaceur peut être conçu pour que le turbocodeur termine toujours à l'état zéro, à condition que l'auto-concaténation soit appliquée : la séquence de données entrelacée doit être directement codée derrière la séquence non entrelacée, sans initialiser l'état du codeur entre ces deux séquences. Contrairement à la technique de terminaison à zéro, cette méthode n'introduit aucun effet secondaire et ne nécessite pas la transmission de bits supplémentaires. Malheureusement, elle impose de fortes contraintes sur la conception de l'entrelaceur.

Terminaison circulaire (tail-biting) : c'est la meilleure méthode de terminaison pour les turbocodes. Premièrement, aucun bit supplémentaire ne doit être ajouté et transmis; ainsi, il n'y a pas de perte de débit et l'efficacité spectrale de la transmission n'est pas réduite. Ensuite, le tail-biting n'induit aucun effet secondaire dans le message. Par conséquent, tous les bits d'information sont protégés de la même manière par le turbocode et la propriété circulaire

empêche l'apparition de mots de code tronqués de faible poids puisque le treillis peut être considéré comme ayant une longueur infinie. De plus, comme aucune position particulière dans le treillis ne doit être prise en compte, la conception de l'entrelacement est simplifiée. En raison de sa commodité particulière pour les turbocodes, cette technique de terminaison a été adoptée dans plusieurs normes de télécommunication récentes telles que DVB-RCS, DVB-RCT et WiMAX [31].

Les arguments en faveur de ce schéma de codage sont les suivants [21]:

- Un décodeur de code convolutif est vulnérable aux erreurs survenant en paquets. Coder le message deux fois, suivant deux ordres différents (avant et après entrelacement), c'est rendre peu probable l'apparition simultanée de paquets d'erreurs à l'entrée des décodeurs de RSC1 et de RSC2. Si des erreurs groupées surviennent à l'entrée du décodeur de RSC1, l'entrelacement les disperse dans le temps et elles deviennent des erreurs isolées, aisément corrigibles, pour le décodeur de RSC2. Le raisonnement est également réciproque pour les paquets d'erreurs à l'entrée du décodeur de RSC2.
- La concaténation parallèle conduit à un rendement de codage plus élevé que celui de la concaténation série. La concaténation parallèle est donc plus favorable lorsque des rapports signal à bruit proches des limites théoriques sont considérés, avec des taux d'erreurs visés moyens.
- Les codes élémentaires RSC sont de petits codes : codes à 16, 8, voire 4 états. Le décodage, même s'il met en œuvre des traitements probabilistes répétés, reste de complexité raisonnable.

Le tableau 3.2 recense les applications industrielles connues à nos jours qui utilisent les turbocodes en pratique [21].

La manière dont on effectue l'entrelacement est importante lorsque le taux d'erreurs binaires cible est inférieur à 10^{-5} environ. Au-dessus de cette valeur, la performance est peu sensible à l'entrelacement, à condition bien sûr que celle-ci respecte au moins le principe de dispersion (cela peut être par exemple un entrelaceur régulier). Pour un taux d'erreurs cible faible ou très faible, la performance est dictée par la distance minimale du code et celle-ci est très dépendante de l'entrelacement Π [21].

Tableau 3.2. Les applications standardisées des turbocodes convolutifs.

Application	Turbocode	Terminaison	Polynômes	Rendements
CCSDS (espace lointain)	Binaire, 16 états	Tail-bits	23, 33, 25, 37	1/6, 1/4, 1/3, 1/2
UMTS, CDMA2000 (mobile 3G)	Binaire, 8 états	Tail-bits	13, 15, 17	1/4, 1/3, 1/2
DVB-RCS (Return Channel on Satellite)	Double binaire, 8 états	Circulaire	15, 13	de 1/3 à 6/7
DVB-RCT (Return Channel on Terrestrial)	Double binaire, 8 états	Circulaire	15, 13	1/2, 3/4
Inmarsat (M4)	Binaire, 16 états	Aucune	23, 35	1/2
Eutelsat (skyplex)	Double binaire, 8 états	Circulaire	15, 13	4/5, 6/7
IEEE 802.16 (WiMAX)	Double binaire, 8 états	Circulaire	15, 13	de 1/2 à 7/8

Le motif de poinçonnage doit être le plus régulier possible, à l'image de ce qui se pratique pour les codes convolutifs classiques. Toutefois, il peut être avantageux d'avoir un motif de poinçonnage faiblement irrégulier quand on recherche de très faibles taux d'erreurs et lorsque la période de poinçonnage est un diviseur de la période du polynôme générateur de récursivité ou de parité. Le poinçonnage s'effectue classiquement sur les symboles de redondance.

3.5.2. L'entrelacement :

Quoi que nous l'appelions, l'entrelacement ou la permutation, la technique qui consiste à étaler les données dans le temps a toujours été très utile dans les communications numériques. L'idée initiale et fondamentale d'introduire un entrelaceur dans la structure concaténée parallèle des turbocodes était de lutter efficacement contre l'occurrence d'éclat d'erreurs (burst errors), sur au moins l'une des dimensions du code composite.

Un entrelacement avec de bonnes propriétés de dispersion évite que deux bits proches l'un de l'autre avant d'être entrelacés (c'est-à-dire dans l'ordre naturel) restent proches après l'entrelacement. Ces aspects indésirables provoquent une dégradation des performances car elles introduisent une certaine corrélation dans le processus de décodage itératif [31]. Les

propriétés de dispersion d'un entrelacement peuvent être mesurées à travers sa distance spatiale cumulée, définie comme :

$$S_{min} = \min_{j_1, j_2 | j_1 \neq j_2} (|j_1 - j_2| + |\Pi(j_1) - \Pi(j_2)|) \qquad (3.44)$$

Pour $j_1, j_2 = 1, \ldots, K$. Où K est la longueur de l'entrelaceur et Π désigne la fonction de permutation qui associe l'indice $\Pi(j)$ dans l'ordre entrelacé à un indice i dans l'ordre naturel.

3.5.2.1. Permutation régulière :

L'implémentation conventionnelle de la permutation régulière implique l'écriture des données dans une rangée en mémoire tout en les codant avec le premier codeur, et en les lisant par colonnes pour la deuxième étape de codage avec le deuxième codeur. Ce processus de permutation suppose que $K = N_r \times N_c$, voir figure 3.12-a.

Ensuite, une autre forme de permutation régulière (décalage circulaire) consiste à écrire les données dans une mémoire linéaire de l'adresse $i = 0$ à l'adresse $i = K - 1$ et à les lire aux adresses :

$$i = \Pi(j) = Pj \; mod(K) \qquad \text{pour } j = 0,1, \ldots, K-1 \qquad (3.45)$$

Comme représentée sur la figure 3.12-b, le bloc de données est considéré comme un cercle, les deux extrémités étant adjacentes dans l'ordre naturel ($i = 0 \; et \; i = K - 1$) et dans l'ordre entrelacé ($j = 0 \; et \; j = K - 1$). Cela rend le décalage circulaire bien adapté pour les codes convolutifs de terminaison circulaire. La contrainte de conception de ce type d'entrelaceur est que P et K doivent être relativement premiers.

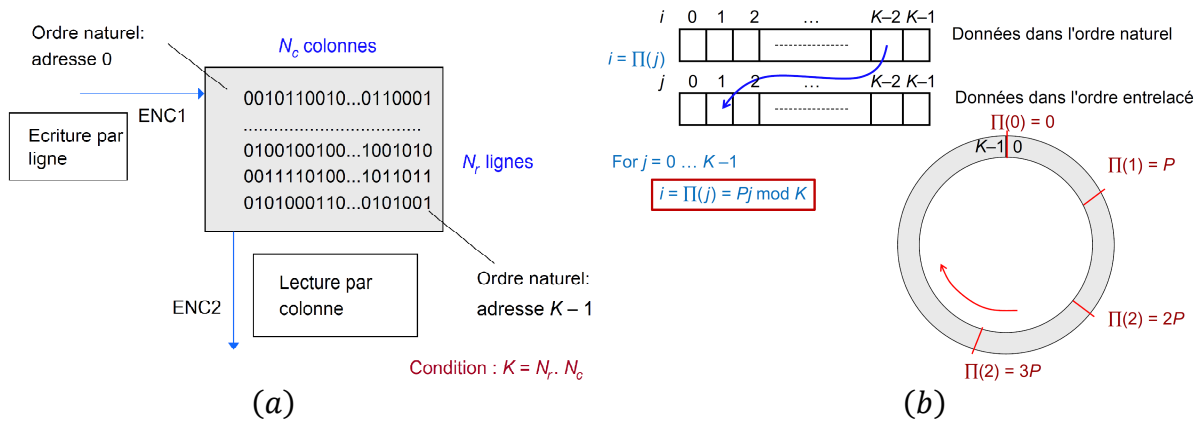

Figure 3.12. La Permutation régulière : a) forme rectangulaire, b) forme circulaire

La permutation régulière est capable d'atteindre la valeur maximale de la distance spatiale cumulée $S_{min} = \sqrt{2K}$. Elle est donc appropriée pour les turbocodes dans le sens qu'ils minimisent la corrélation entre les informations extrinsèques fournies par chaque décodeur [21,31].

3.5.2.2. Permutation irrégulière :

Les permutations irrégulières ont fait l'objet de nombreuses publications ou de plusieurs chapitres d'ouvrages jusqu'à ce jour. Nous présentons ici à titre d'exemple un 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*) ou DRP (*Dithered Relatively Prime*) [21].

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.

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. Elle permet en retour de supprimer les motifs d'erreurs composés.

Le modèle mathématique associé à cette permutation presque régulière, sous sa forme circulaire, est une extension de (3.45) :

$$i = \Pi(j) = Pj + Q(j) \ \ mod(K) \qquad \text{pour } j = 0,1,\dots,K-1 \tag{3.46}$$

Si l'on choisit : $Q(j) = A(j)P + B(j)$, où les entiers positifs $A(j)$ et $B(j)$ sont périodiques, de cycle C (diviseur de K), alors ces grandeurs correspondent à des décalages positifs appliqués respectivement avant et après la permutation régulière [21,31].

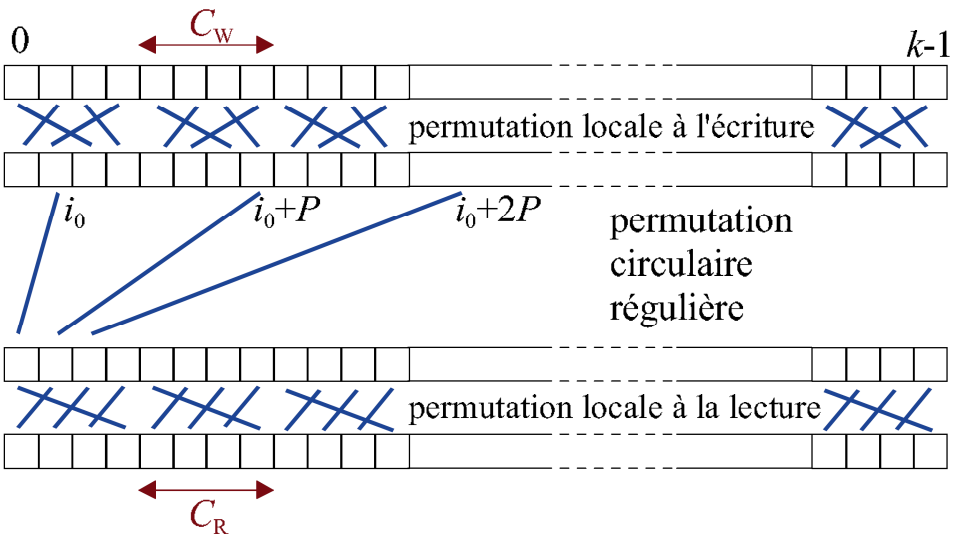

Figure 3.13. Permutation de type DRP

3.5.3. Le décodage des turbocodes :

En sortie de canal, trois vecteurs bruitées $\boldsymbol{r}^{(0)} = \tilde{\boldsymbol{u}}$, $\boldsymbol{r}^{(1)}$ et $\boldsymbol{r}^{(2)}$ sont observées. Afin de retrouver à partir de ces vecteurs l'information $\hat{\boldsymbol{u}}$, le décodage suit le schéma représenté sur la figure 3.14. Les grandeurs considérées à chaque nœud du montage sont des LLR, les opérations de décodage étant effectuées dans le domaine logarithmique. Deux décodeurs convolutifs sont placés en parallèle afin d'utiliser simultanément les informations données par chacun d'eux. Il s'agit donc d'un décodage itératif au sens où la boucle enchaînant décodage de $\mathcal{C}_1$, application de Π, décodage de $\mathcal{C}_2$ et désentrelacement Π^{-1} est répétée plusieurs fois.

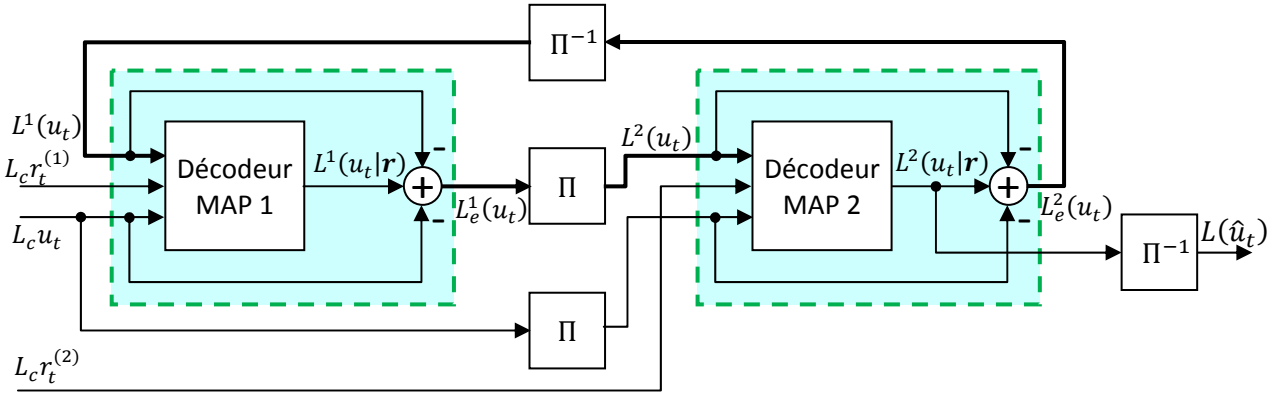

Figure 3.14. Schéma de décodage d'un turbocode (Turbodécodeur)

Comme on le voit sur la figure, chaque décodeur prend trois entrées: les bits de sortie de canal systématiques $\tilde{\boldsymbol{u}}$, les bits de parité transmis par le codeur RSC associé $\boldsymbol{r}^{(i)}, i = 1,2$, et les informations provenant du décodeur de l'autre composant sur les valeurs probables des bits

concernés. Cette information provenant de l'autre décodeur est appelée information a priori. Les composants décodeurs doivent également fournir ce qu'on appelle les sorties souples (Soft-output) pour les bits décodés sous forme de LLR. La polarité du LLR détermine le signe du bit, tandis que son amplitude quantifie la probabilité d'une décision correcte. Deux types de décodeurs sont adéquats, soit l'algorithme SOVA (Soft-Output Viterbi Algorithm) proposé par Hagenauer et Höher [36] et l'algorithme MAP (Maximum A posteriori Probability) de Bahl et al. [5], qui sont décrits aux sections 3.4.1 et 3.4.2, respectivement. Dans ce travail on n'utilisera que le deuxième type pour des raisons de performances et de complexité, ainsi que pour sa popularité dans les applications pratiques actuelles citées précédemment.

En utilisant l'équation 3.41 et en rappelant que dans le numérateur nous avons $u_t = +1$ pour tous les termes de la somme, alors que dans le dénominateur nous avons $u_t = -1$, nous pouvons la réécrire comme :

$$L(u_t|\boldsymbol{r}) = L_c u_t + L(u_t) + L_e(u_t) \tag{3.47}$$

Où : $L_e(u_t) = \ln\left(\frac{\sum_{(s',s)\Rightarrow u_t=+1} \alpha_{t-1}(s')\gamma_t^e(s',s)\beta_t(s)}{\sum_{(s',s)\Rightarrow u_t=-1} \alpha_{t-1}(s')\gamma_t^e(s',s)\beta_t(s)}\right)$

Nous donnons ci-dessous les définitions des termes a priori, extrinsèque et a posteriori, que nous utilisons tout au long de cette section.

L'information **a priori** relative à un bit est une information connue avant le début du décodage, à partir d'une source autre que la séquence reçue ou les contraintes du code. Elle est également appelée information intrinsèque pour faire contraste avec l'information extrinsèque.

L'information **extrinsèque** fournie par un décodeur relative à un bit u_t**,** est l'information basée sur la séquence reçue et sur l'information a priori, mais excluant les bits systématiques reçus et l'information a priori $L(u_t)$ relative au bit u_t. Typiquement, le composant décodeur fournit cette information en utilisant les contraintes imposées à la séquence transmise par le code utilisé.

L'information **a posteriori** relative à un bit est l'information que le décodeur génère en prenant en compte toutes les sources d'informations disponibles concernant u_t. C'est le LLR a posteriori, c'est-à-dire $L(u_t|\boldsymbol{r})$, que l'algorithme MAP génère en sortie [35].

Dans le turbodécodeur, les deux composants décodeurs partagent leurs informations extrinsèques sur les bits de message en tant qu'informations a priori introduites dans chaque

décodeur. La principale différence pour les décodeurs MAP à l'intérieur d'un turbodécodeur est la source de leurs informations a priori. La deuxième différence dans un turbodécodeur est que chaque décodeur MAP est utilisé plus d'une fois. L'information extrinsèque est mise à jour et transmise entre les décodeurs sur plusieurs itérations.

Une itération du turbodécodage consiste en une application de l'algorithme MAP pour chaque composant décodeur. Un turbodécodeur peut utiliser n'importe quel nombre d'itérations. Cependant, augmenter le nombre d'itérations augmente le temps de décodage.

Le turbodécodeur dont le schéma bloc est illustré dans la figure 3.14, a comme entrée $L_c u_t$, $L_c r_t^{(1)}$ et $L_c r_t^{(2)}$, les LLR des séquences $\boldsymbol{r}^{(0)} = \widetilde{\boldsymbol{u}}$, $\boldsymbol{r}^{(1)}$ et $\boldsymbol{r}^{(2)}$ en sortie du canal respectivement.

Dans la première itération, $l = 1$, le décodeur DEC1 n'a que $L_c u_t$, et $L_c r_t^{(1)}$ à son entrée, et pas d'information a priori ($L_1^1(u_t) = 0$). A la sortie il donne l'information a posteriori $L_1^1(u_t|\boldsymbol{r})$. La nouvelle information extrinsèque qu'il a créé à propos des bits u_t de la première itération, indiquée par l'indice 1, est donc $L_{e,1}^1(u_t) = L_1^1(u_t|\boldsymbol{r}) - L_c u_t$.

Les entrées du décodeur DEC2 provenant du canal sont $\Pi(L_c u_t)$ et $L_c r_t^{(2)}$, les valeurs $L_c u_t$ étant entrelacées pour être dans le même ordre que celui des bits de message lorsqu'ils sont entrés dans le codeur RSC2.

Pour l'itération l, le décodeur DEC2 utilisera les informations extrinsèques provenant du premier décodeur, $L_{e,l}^1(u_t)$ après entrelacement, en tant qu'information a priori supplémentaire $L_l^2(u_t) = \Pi\left(L_{e,l}^1(u_t)\right)$ sur les bits du message u_t.

Le décodeur DEC2 sortira les LLR a posteriori qu'il calcule pour les bits du message en tant que $L_l^2(u_t|\boldsymbol{r})$. La nouvelle information extrinsèque créée par le décodeur DEC2 sur les bits de message à l'itération l est donc :

$$L_{e,l}^2(u_t) = L_l^2(u_t|\boldsymbol{r}) - \Pi(L_c u_t) - L_l^2(u_t) \tag{3.48}$$

A partir de la seconde itération, le décodeur DEC1 répétera l'algorithme MAP mais maintenant avec une information a priori provenant de information extrinsèque du décodeur DEC2 créée à l'itération précédente. Ainsi, dans la $l^{\text{ème}}$ itération, l'information extrinsèque

provenant du décodeur DEC2 est désentrelacée de manière à être dans le même ordre que les bits de message de DEC1 : $L_l^1(u_t) = \Pi^{-1}\left(L_{e,l-1}^2(u_t)\right)$.

La nouvelle information extrinsèque sur les bits du message fournie par le décodeur DEC1 est ainsi :

$$L_{e,l}^1(u_t) = L_l^1(u_t|\boldsymbol{r}) - L_c u_t - L_l^1(u_t) \tag{3.49}$$

Notons que les rapports de vraisemblance logarithmique du canal restent inchangés tout au long du turbodécodage, et seules les informations extrinsèques changent à chaque itération.

Toutes les démonstrations et le développement mathématique de l'algorithme MAP (déjà décrit à la section 3.4.2) peuvent être consultés avec plus de détails dans les références [41,43]

Le turbodécodeur peut être arrêté après un nombre fixe d'itérations I_{max} ou lorsque des critères prédéterminés ont été atteints (nous les traiterons en détails dans le chapitre 4). La décision finale peut être prise sur la base de la sortie de l'un ou l'autre décodeur MAP ou la moyenne des deux sorties.

Décodage itératif des Turbocodes avec l'algorithme MAP

Input : $\boldsymbol{r} = \left[u_1, r_1^{(1)}, r_1^{(2)}, \dots, u_K, r_K^{(1)}, r_K^{(2)}\right] \in \mathcal{R}^n$ ▷Mot de code reçu

Output : $\widehat{\boldsymbol{u}} = (\hat{u}_1, \hat{u}_2, \dots, \hat{u}_K) \in \{0,1\}^K$ ▷Mot d'information estimé

procedure Turbo(treillis1,treillis2, Π, I_{max}, $L_c\boldsymbol{u}$, $L_c\boldsymbol{r}^{(1)}$, $L_c\boldsymbol{r}^{(2)}$)

$L_e^2(\boldsymbol{u}) = zeros$

Iterations Loop :

for all $l = 1, \dots, I_{max}$ **do** ▷Pour chaque itération l

$L^1(\boldsymbol{u}) = \Pi^{-1}\left(L_e^2(\boldsymbol{u})\right)$

$L^1(\boldsymbol{u}|\boldsymbol{r}) = MAPdec(\text{treillis1}, L_c\boldsymbol{u}, L_c\boldsymbol{r}^{(1)}, L^1(\boldsymbol{u}))$ ▷Décodeur DEC1

$L_e^1(\boldsymbol{u}) = L^1(\boldsymbol{u}|\boldsymbol{r}) - L_c\boldsymbol{u} - L^1(\boldsymbol{u})$

$L^2(\boldsymbol{u}) = \Pi\left(L_e^1(\boldsymbol{u})\right)$

$L^2(\boldsymbol{u}|\boldsymbol{r}) = MAPdec(\text{treillis2}, \Pi(L_c\boldsymbol{u}), L_c\boldsymbol{r}^{(2)}, L^2(\boldsymbol{u}))$ ▷Décodeur DEC2

$L_e^2(\boldsymbol{u}) = L^2(\boldsymbol{u}|\boldsymbol{r}) - \Pi(L_c\boldsymbol{u}) - L^2(\boldsymbol{u})$

End Iterations Loop

$L(\widehat{\boldsymbol{u}}) = \Pi^{-1}\left(L^2(\boldsymbol{u}|\boldsymbol{r})\right)$

$\widehat{\boldsymbol{u}} = sign\left(L(\widehat{\boldsymbol{u}})\right)$

3.5.4. Modification et simplification de l'algorithme MAP :

Le décodage suivant l'algorithme MAP requiert un grand nombre d'opérations, dont des calculs d'exponentielles et des multiplications. Pour éviter ces calculs mathématiques complexes, la réécriture de l'algorithme de décodage dans le domaine logarithmique s'avère très utile pour simplifier les traitements.

3.5.4.1. L'approximation Max-Log-MAP :

La technique Max-Log-MAP a simplifié l'algorithme MAP en transférant les récurrences de l'équation (3.37) dans le domaine logarithmique et en invoquant une approximation dans le but de réduire considérablement la complexité d'implémentation associée. En raison de cette approximation, les performances de l'algorithme Max-Log-MAP sont sous-optimales.

L'algorithme MAP calcule les LLR de $L(u_t|\boldsymbol{r})$ a posteriori en utilisant l'équation (3.41). Pour ce faire, il faut les valeurs suivantes :

- les valeurs $\alpha_{t-1}(s')$, qui sont calculées par récurrence directe en utilisant l'équation (3.37).
- les valeurs $\beta_t(s)$, qui sont calculées par récurrence inverse en utilisant l'équation (3.37).
- les probabilités de transition de branche $\gamma_t(s', s)$, calculées à l'aide de l'équation (3.40).

L'algorithme Max-Log-MAP simplifie cela en transférant ces équations dans le domaine logarithmique, puis en utilisant l'approximation :

$$\ln\left(\sum_i e^{x_i}\right) \approx \max_i x_i \tag{3.50}$$

Alors, on défini les nouveaux métriques $A_t(s)$, $B_t(s)$ et $\Gamma_t(s', s)$ comme :

$$A_t(s) = \ln\big(\alpha_t(s)\big) \ , \ \ B_t(s) = \ln\big(\beta_t(s)\big) \ et \ \ \Gamma_t(s', s) = \ln\big(\gamma_t(s', s)\big) \tag{3.51}$$

Donc l'équation (3.37) des probabilités récurrentes des métriques de nœuds $\alpha_t(s)$ et $\beta_{t-1}(s')$ peut être réécrite ainsi :

$$\begin{aligned} A_t(s) &= \ln\left(\sum_{s'} \gamma_t(s',s)\alpha_{t-1}(s')\right) \\ &= \ln\left(\sum_{s'} e^{\Gamma_t(s',s)+A_{t-1}(s')}\right) \\ &\approx \max_{s'}\left(\Gamma_t(s',s) + A_{t-1}(s')\right) \end{aligned} \tag{3.52}$$

$$\begin{aligned} B_{t-1}(s') &= \ln\left(\sum_{s} \gamma_t(s',s)\beta_t(s)\right) \\ &= \ln\left(\sum_{s} e^{\Gamma_t(s',s)+B_t(s)}\right) \\ &\approx \max_{s}\left(\Gamma_t(s',s) + B_t(s)\right) \end{aligned} \tag{3.53}$$

On voit bien que le passage au domaine logarithmique élimine les exponentiels et permet de remplacer les multiplications par des additions et la somme par une simple comparaison. Ce qui réduit considérablement la complexité d'implémentation du turbodécodeur.

En utilisant l'équation (3.40), le calcule des métriques de branches logarithmique $\Gamma_t(s',s)$ dans les deux équations précédentes est obtenu par :

$$\begin{aligned} \Gamma_t(s',s) &= \ln\left(e^{\frac{1}{2}u_t\left(L_c r_t^{(0)}+L(u_t)\right)+\frac{1}{2}\sum_{i=1}^{n-1} L_c r_t^{(i)} c_t^{(i)}}\right) \\ &= \frac{1}{2}u_t L(u_t) + \frac{1}{2}\sum_{i=0}^{n-1} L_c r_t^{(i)} c_t^{(i)} \end{aligned} \tag{3.54}$$

La métrique de branche est donc équivalente à celle utilisée dans l'algorithme de Viterbi, avec l'ajout du LLR de l'information a priori $u_t L(u_t)$.

Enfin, à partir de l'équation (3.41), nous pouvons écrire pour les LLR a posteriori $L(u_t|\boldsymbol{r})$ que l'algorithme Max-Log-MAP calcule :

$$\begin{aligned} L(u_t|\boldsymbol{r}) &= \ln\left(\frac{\sum_{(s',s)\Rightarrow u_t=+1} e^{A_{t-1}(s')+\Gamma_t(s',s)+B_t(s)}}{\sum_{(s',s)\Rightarrow u_t=-1} e^{A_{t-1}(s')+\Gamma_t(s',s)+B_t(s)}}\right) \\ &\approx \max_{(s',s)\Rightarrow u_t=+1}\left(A_{t-1}(s') + \Gamma_t(s',s)+B_t(s)\right) - \max_{(s',s)\Rightarrow u_t=-1}\left(A_{t-1}(s') + \Gamma_t(s',s) + B_t(s)\right) \end{aligned} \tag{3.55}$$

L'algorithme Max-Log-MAP peut être résumé comme suit. Les récurrences dans les sens directe et inverse, similaires à la récurrence directe utilisée dans l'algorithme de Viterbi, sont utilisées pour calculer $A_t(s)$ et $B_t(s)$en utilisant les équations (3.52) et (3.53) respectivement.

La métrique de branche logarithmique $\Gamma_t(s', s)$ utilisée est donnée par l'équation (3.54). Une fois que les récurrences ont été effectuées, les LLR a posteriori peuvent être calculés en utilisant l'équation (3.55). Ainsi, la complexité de l'algorithme Max-Log-MAP n'est pas très supérieure à celle de l'algorithme de Viterbi.

Le calcul de $L(u_t|\boldsymbol{r})$ à partir des valeurs $A_{t-1}(s')$, $B_t(s)$ et $\Gamma_t(s', s)$ nécessite, pour chaque bit, deux additions pour chacune des 2×2^m transitions à chaque étage du treillis, deux maximisations et une soustraction.

3.5.4.2. L'algorithme Log-MAP :

L'algorithme Max-Log-MAP donne une légère dégradation des performances par rapport à l'algorithme MAP en raison de l'approximation de l'équation (3.50). Lorsqu'il est utilisé pour le décodage itératif des turbocodes, cette dégradation se traduit par une baisse des performances d'environ 0.35 dB.

Cependant, l'approximation de l'équation (3.50) peut être rendue exacte en utilisant le logarithme Jacobien noté généralement par l'opérateur $max^*(.)$:

$$max^*(x, y) \equiv \ln(e^x + e^y) = \max(x, y) + \ln\left(1 + e^{-|y-x|}\right) \qquad (3.56)$$

Où le terme $\ln\left(1 + e^{-|y-x|}\right)$ peut être considéré comme un correctif, qui peut facilement être calculé en utilisant une table de consultation LUT (look-up table). C'est alors la base de l'algorithme Log-MAP proposé par Robertson et al. [37].

En suivant les mêmes étapes de l'algorithme Max-Log-MAP, L'algorithme Log-MAP calcule les valeurs de $A_t(s)$ et $B_t(s)$ en utilisant une récurrence directe et inverse respectivement. Cependant, la maximisation dans les équations (3.52) et (3.53) est complétée par le terme de correction de l'équation (3.56). Cela signifie que les valeurs exactes de $A_t(s)$ et $B_t(s)$ sont calculées.

De même, l'approximation dans l'équation (3.55) donnant les LLR a posteriori $L(u_t|\boldsymbol{r})$ peut être éliminée en utilisant le logarithme Jacobien.

Cependant, comme expliqué précédemment, il y aura 2^m transitions à considérer dans chacune des maximisations de l'équation (3.55). Nous devons donc généraliser l'équation (3.56) pour faire face à plus de deux termes. Ceci est fait en imbriquant les opérations $max^*(x_{i+1}, x_i)$ du logarithme Jacobien comme :

$$\ln\left(\sum_{i=1}^{I} e^{x_i}\right) = max^*\left(x_I, max^*\left(x_{I-1}, \dots, max^*(x_3, max^*(x_2, x_1))\right)\right) \qquad (3.57)$$

Le terme de correction $\ln(1 + e^{-|y-x|})$ n'a pas besoin d'être calculé pour chaque valeur de $|y - x|$, mais peut être stocké dans une table LUT. Robertson et al. [37] ont trouvé qu'une telle table de consultation ne doit contenir que huit valeurs pour $|y - x|$, comprises entre 0 et 5. Cela signifie que l'algorithme Log-MAP n'est que légèrement plus complexe que l'algorithme Max-Log-MAP, mais il donne exactement la même performance que l'algorithme MAP. Par conséquent, il s'agit d'un algorithme très intéressant à utiliser dans les composants décodeurs d'un turbodécodeur itératif.

3.6. Analyse des performances des turbocodes :

Les deux principales méthodes pour évaluer la performance des turbocodes sont l'analyse théorique et la simulation par ordinateur. L'analyse théorique d'un turbocode est très difficile en raison de la structure du schéma de codage. Quelques articles [38-41] ont analysé les turbocodes avec cette approche théorique. Cependant, leurs résultats ne correspondent pas (trop optimistes) aux résultats de la simulation par ordinateur. De plus, les analyses théoriques présentées dans ces articles ne sont pas clairement décrites et sont donc difficiles à suivre. En outre, l'approche théorique nécessite souvent un temps d'exécution de l'ordinateur énorme pour trouver la répartition complète du poids des mots de code du turbocode.

Dans cette thèse, en raison des difficultés présentées dans l'analyse théorique, la performance des turbo-codes est évaluée par simulation informatique. MATLAB est utilisé pour construire le code informatique d'un turbocode, et les simulations ont été effectuées sur des ordinateurs d'architecture récente avec des ressources de calcul moyennes.

Nous présentons dans ce paragraphe les taux d'erreurs par bit BER (Bit Error Rate) en fonction du rapport $\frac{E_b}{N_0}$ pour différents turbocodes et en fonction de plusieurs paramètres. Nous montrons qu'il existe de nombreux paramètres, dont certains sont liés, qui affectent la performance des turbocodes. Certains de ces paramètres sont :

- Le nombre d'itérations de décodage utilisées.
- La longueur de bloc d'information ou la latence des données d'entrée
- Les polynômes générateurs et les longueurs de contraintes des codes constituants

Les taux d'erreurs sont calculés par une simulation de Monte-Carlo sur un canal AWGN en appliquant le décodage itératif.

Le tableau suivant récapitule tous les principaux paramètres de simulation, qui seront utilisés pour obtenir les différents résultats, sauf indication contraire.

Tableau 3.3. Paramètres standard du turbocodeur et du turbodécodeur utilisés

Canal	Canal à Bruit Blanc Additif Gaussien AWGN
Modulation	Binary Phase Shift Keying BPSK
Composant Codeur RSC	$n = 2$, $k = 1$, $\upsilon = 3$, $\boldsymbol{G}(D) = \left[1 \quad \frac{1+D^2}{1+D+D^2}\right]$
Entrelaceur	Régulier $N = 1024\ bits$
Poinçonnage	$R = \frac{1}{3}$
Composant décodeur	Décodeur LogMAP
Nombre d'itérations	$I_{max} = 10$

La figure 3.15 montre l'influence du nombre d'itérations de décodage sur le taux d'erreur binaire du turbocode. Au fur et à mesure que le nombre d'itérations utilisées par le turbodécodeur augmente, la décroissance du BER est nettement améliorée. Cependant, au delà de huit itérations, il y a peu d'amélioration en utilisant d'autres itérations. Par exemple, on peut voir sur la figure.3.15 que l'utilisation de huit itérations au lieu de six donne une amélioration d'environ 0.1dB. On en conclut que le gain entre deux itérations successives diminue rapidement en fonction du nombre d'itérations réalisé.

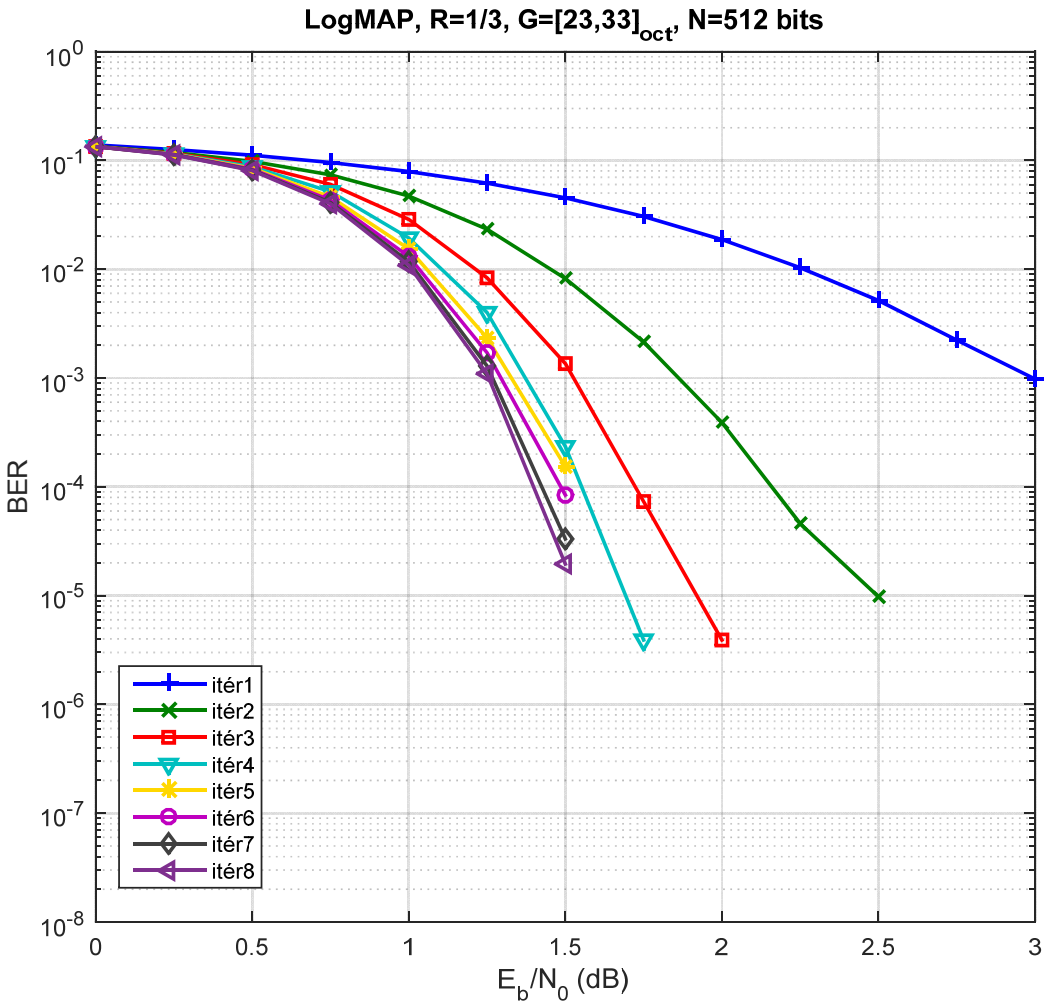

Figure 3.15. Comparaison de BER en fonction du nombre d'itérations de décodage

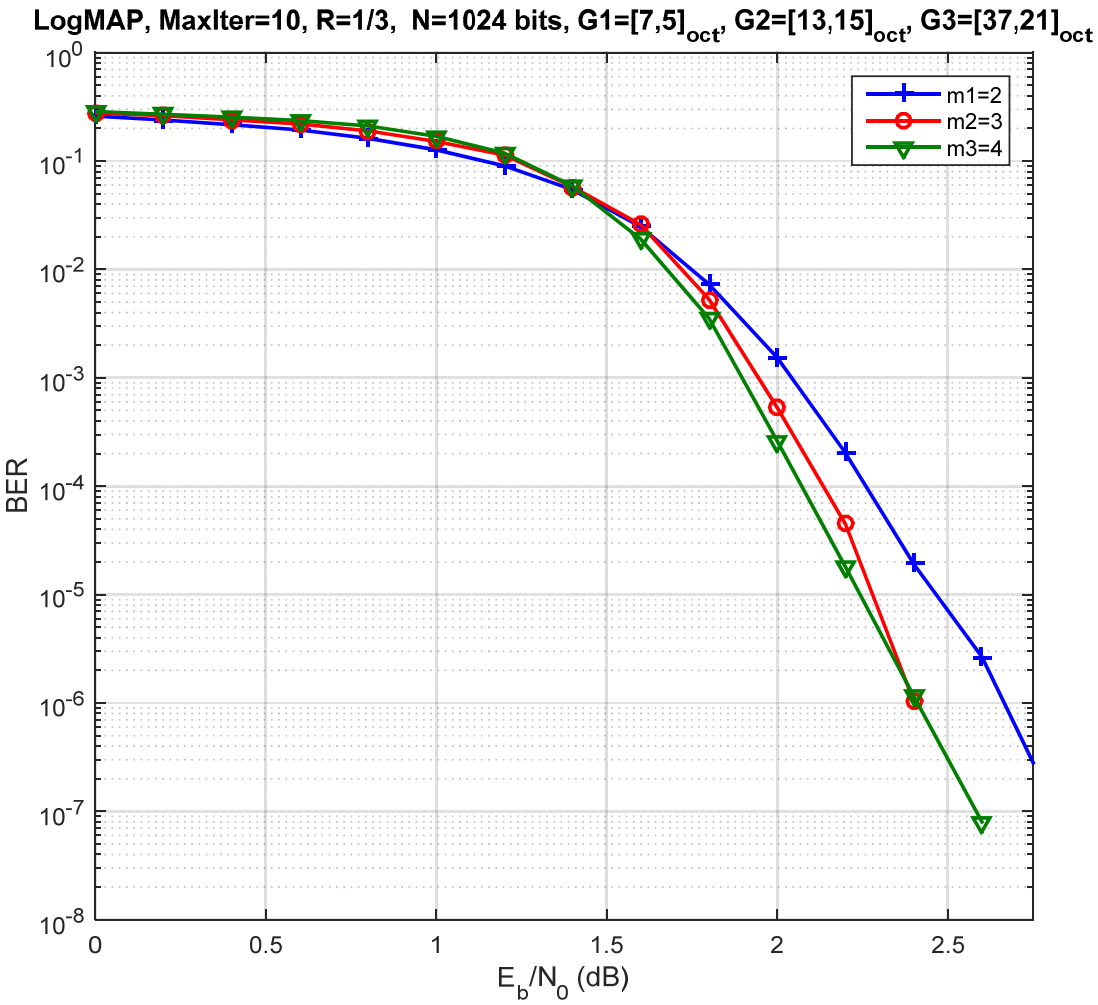

Figure 3.16. Comparaison des performances des turbocodes avec différente longueurs de contrainte ($\upsilon = 3\,,4\,,5$)

De façon surprenante, le choix du codeur RSC constitutif, et en particulier sa longueur de contrainte υ, n'influence pas significativement les performances des turbocodes à faibles rapports signal sur bruit et modérément faible à des rapports plus élevés. Pour cette raison, les turbocodes utilisent généralement des codes constituants RSC simples avec une longueur de contrainte $3 \leq \upsilon \leq 5$.

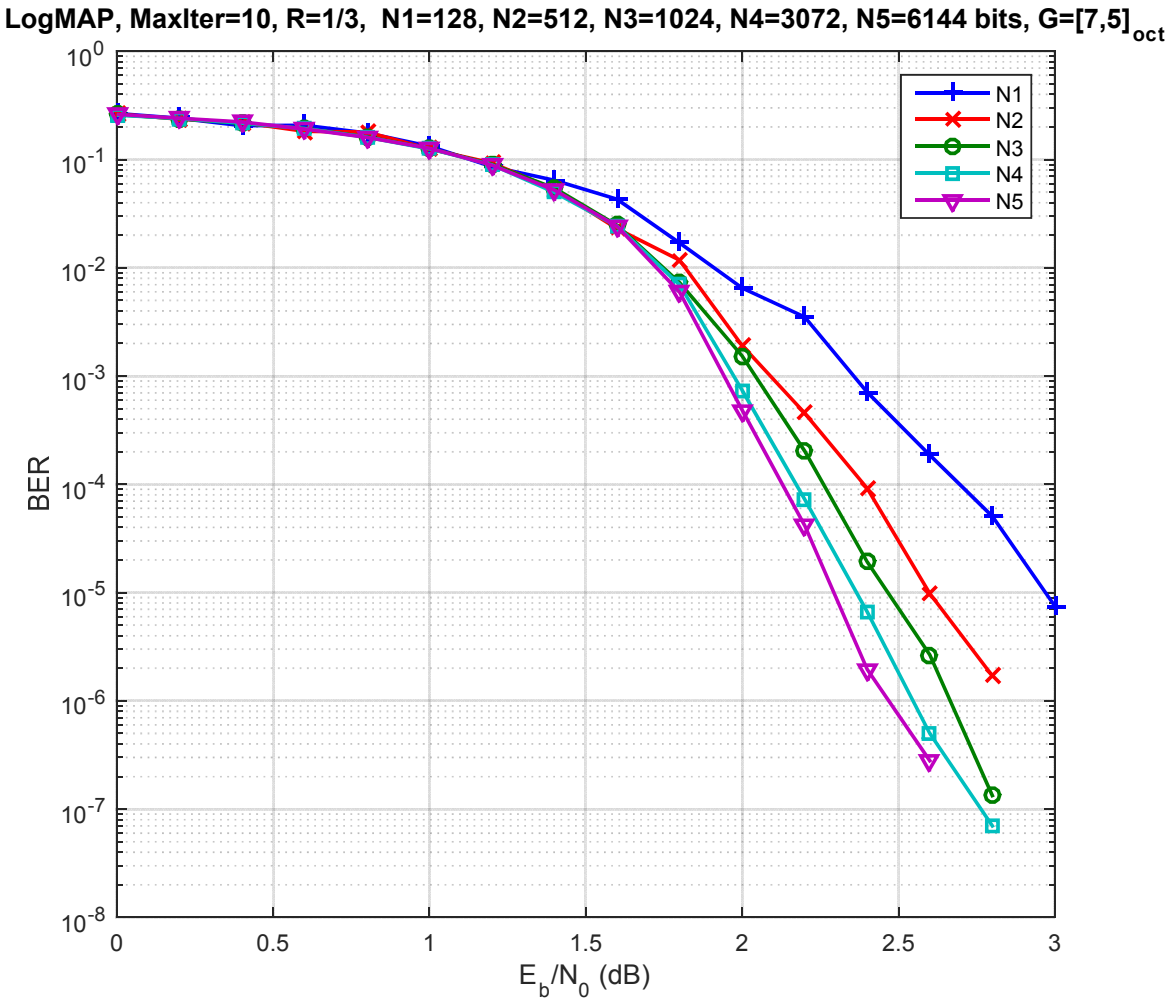

Figure 3.17. Effet de la longueur de trame sur les performances BER du turbocodage.

Considérons maintenant la performance de différents turbocodes en faisant varier la longueur du code. Dans la figure 3.17, nous présentons le taux d'erreur binaire (BER) des turbocodes utilisant le même codeur convolutif RSC avec différentes longueurs d'entrelacement N. L'augmentation de la longueur du code entraîne une amélioration des performances pour une large plage du taux d'erreur binaire. Ceci confirme le fait que l'augmentation de la longueur de l'entrelaceur pour un turbocode donné, permet de luter contre la corrélation des bits erronés ayants des positions très proches dans la séquence des mots de codes, et donc le processus de décodage itératif conduit à de meilleures performances.

Notons donc que la performance d'un turbocode est fortement liée à la taille de l'entrelaceur. En effet, la taille de ce dernier est l'un des facteurs les plus influents affectant la performance des turbocodes. Il semblerait souhaitable d'utiliser la longueur de l'entrelaceur la plus grande possible; cependant, des entrelaceurs plus grands introduisent des latences de décodage plus longues et nécessitent plus de mémoire dans le décodeur. Ainsi, la taille de

l'entrelaceur doit être choisie pour correspondre aux exigences du système en termes du taux d'erreur binaire, de temps de latence et de mémoire du décodeur.

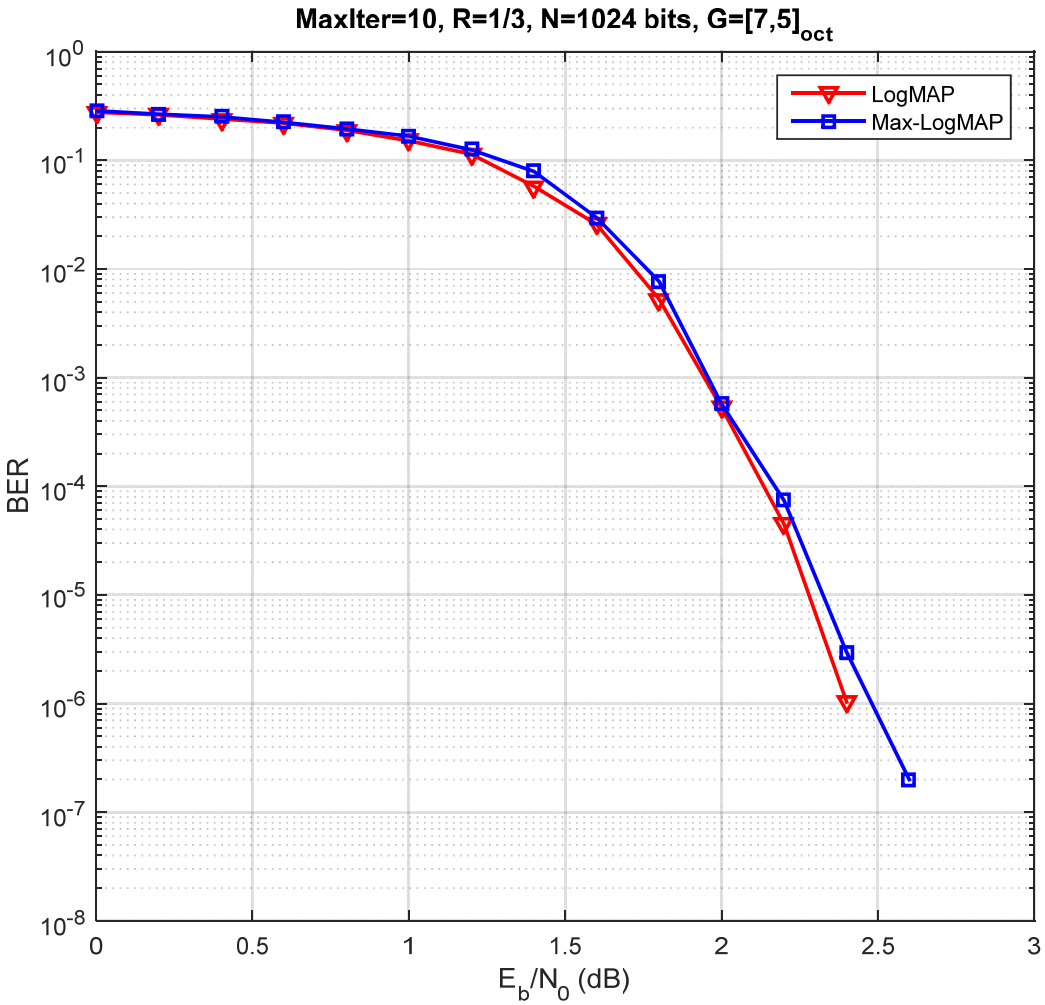

Figure 3.18. Comparaison des performances de l'algorithme LogMAP et l'approximation Max-LogMAP

La figure 3.18 compare les algorithmes LogMAP et Max-LogMAP pour un turbodécodeur d'une longueur de trame de 1024 bits. On peut voir sur cette figure que le Max-LogMAP donne une dégradation des performances par rapport à l'algorithme LogMAP. A un taux d'erreur de 10^{-4}, cette dégradation est d'environ 0.05 dB.

Chapitre **4**

Techniques Dynamiques pour l'Amélioration du Décodage Itératif

4.1. Introduction :

Au cours des deux dernières décennies, la communication sans fil a été révolutionnée par les codes correcteurs d'erreur qui bénéficient d'algorithmes de décodage itératifs.

Les turbocodes sont largement utilisés dans les systèmes actuels (candidat pour de futures normes) de télécommunication sans fil, tels que les normes de téléphonie mobile 3G et 4G (HSPA, LTE), la norme de réseau métropolitain sans fil IEEE 802.16 (WiMAX) et de nombreux autres standards. Classiquement, l'algorithme Logarithmique BCJR (LogMAP) est employé pour le décodage itératif de ces codes convolutifs. Pendant ce temps, la norme WiFi (IEEE 802.11b) pour les réseaux locaux sans fil (WLAN) a adopté les codes LDPC, qui peuvent fonctionner sur la base de l'algorithme Sum-Product.

En raison de leur forte capacité de correction d'erreurs, ces codes de canal sophistiqués ont facilité une communication fiable aux débits de transmission qui s'approchent étroitement de la capacité du canal sans fil. Cependant, le débit de transmission réalisable est limité par le débit de traitement de l'algorithme de décodage itératif, si une opération en temps réel est requise. De plus, la latence de traitement de l'algorithme de décodage itératif impose une limite à la latence de bout en bout. Par conséquent, il existe une demande pour des algorithmes de décodage itératifs ayant des débits de traitement multi-gigabits et des latences de traitement ultra-faibles.

4.2. Approches des techniques de décodage itératif :

Ainsi, le turbodécodage est une tâche relativement complexe, exigeant une quantité considérable de ressources matériels et d'énergie de l'ensemble du terminal mobile (récepteur). L'implémentation efficace du turbodécodeur, qui maximise le débit du traitement et minimise la latence, sans dégrader les performances requises, est devenue un champ fertile pour la communauté des chercheurs dans le domaine de la théorie de l'information et du codage. Ce problème à était exploré sur plusieurs niveaux et on peut classer les travaux d recherches selon trois approches principales : niveau algorithmique, niveau architectural matériel et technologie d'implémentation, et les techniques dynamiques-itératives.

4.2.1. Approche algorithmique :

Dans un premier temps, le défit était de ramener la complexité des calculs de l'algorithme MAP à un niveau permettant l'implémentation du turbodécodeur sur un système de communication en temps réel. C'est la conception au niveau algorithmique. Les principaux aboutissements ont donnés deux variantes de complexité plus faible que l'algorithme de décodage MAP original à savoir les versions Max-Log-MAP et Log-MAP [37]. Mais avec la demande croissante de décodeurs qui répondent à des exigences plus gourmandes, par les nouvelles normes de systèmes de communication sans fil, ces simplifications ne sont plus satisfaisantes.

4.2.2. Approche architecturale et technologie d'implémentation :

Pour obtenir des implémentations de décodeurs itératives efficaces, l'espace de conception du système doit être exploré en mettant l'accent sur la partie dépendante de l'implémentation. Les décisions de conception sont évaluées en fonction de la complexité, le débit (throughput) et la consommation d'énergie.

Bien que le décodage itératif soit significativement moins complexe que le décodage optimal (algorithme à pile et décodage séquentiel), il reste une tâche de calcul complexe en raison de l'utilisation itérative de composants décodeurs coûteux. Même l'utilisation de l'algorithme sous-optimal Max-LogMAP [37,42] pour composant de décodage du code, entraîne des besoins de performance de calcul considérablement élevés. Les estimations de complexité révèlent environ 1500 MOPS pour un débit de données utilisateur de 2 Mbps, en supposant que la longueur de contrainte du code convolutif est $v = 3$ et cinq itérations [43].

Au cours des dernières années, un certain nombre d'architectures et de techniques de conception ont été proposées par les chercheurs pour réduire la consommation d'énergie, améliorer les performances de décodage, réduire la latence et augmenter le débit des turbodécodeurs. Ce qui suit est un bref aperçu de certains des travaux publiés pour atteindre de tels objectifs.

Dans [44,45], une suite d'algorithmes MAP pour le turbodécodage avec des compromis de qualité énergétique pour les canaux AWGN et à évanouissement sont présentés. Ces algorithmes sont dérivés en appliquant des techniques d'approximation telles que l'élagage du treillis, la réduction du nombre d'états, la normalisation des informations extrinsèques, l'application d'une fenêtre glissante et la fin anticipée de l'algorithme MAP.

Dans [46], un turbodécodeur, basé sur l'algorithme LogMAP à virgule fixe (12 bits), est conçu avec la méthode de description des architectures microélectroniques RTL (Register-Transfer-Level), et simulé en utilisant le langage VHDL. Le modèle RTL implémenté est vérifié en comparant ses paramètres avec ceux obtenus à partir d'une implémentation en langage C du même turbo décodeur.

Dans [47], un turbodécodeur LogMAP de base-4 (radix-4) pour des terminaux de données mobiles 3G à haut débit est décrit. Le noyau du processeur LogMAP traite deux symboles reçus par cycle d'horloge en utilisant une architecture radix-4 fenêtrée, doublant le débit par rapport à une architecture similaire de radix-2 pour une fréquence d'horloge donnée. La puce est fabriquée en CMOS à 0.18μm, cadencé à une fréquence d'horloge crête de 145MHz à 1.8V et dissipe 956 mW lors du décodage des flux de données HSDPA (High Speed Downlink Packet Access) à un débit continu de 10.8 Mbps. Le décodeur a un taux de codage R=1/3 et une efficacité énergétique de 10 nJ/bit/itération lorsqu'il fonctionne à 24 Mbps (145MHz). Le noyau du décodeur est réalisé sur 14,5 mm^2 et contient 410k portes logiques et 0.45Mb de SRAM.

Les systèmes de communication actuels peuvent incorporer un niveau élevé d'intégration de plusieurs modules sur une seule puce, c'est-à-dire System On Chip (SOC). Ainsi, un turbocodec (turbocodeur et turbodécodeur) peut être implémenté en tant que noyaux IP (Intellectual Property core) logiciel (Soft Core) ou matériel (Hard Core) pouvant être intégrés comme blocs constituants, aux côtés d'autres composants, tels que filtres et démodulateurs, sur une mono-puce d'un système de communication. Ces noyaux IP présentent une grande

flexibilité de conception pour s'adapter aux diverses applications et aux besoins des clients. Des exemples de noyaux turbocodec peuvent être trouvés dans [48].

4.2.3. Approche par techniques dynamiques-itératives :

Les turbodécodeurs sont de nature itérative, c'est-à-dire qu'ils doivent effectuer un certain nombre d'itérations avant d'atteindre un degré de confiance satisfaisant en ce qui concerne une séquence à décoder. Au début, les turbodécodeurs standards utilisaient un nombre fixe d'itérations, ce qui est déterminé par la dégradation du bruit dans le cas le plus défavorable. Cependant, la plupart des mots de code ne sont pas corrompus par le pire des cas et nécessitent donc moins d'itérations pour converger.

En fait, à partir des figures 3.15 à 3.18, le décodeur peut corriger certains blocs de données reçus avant d'atteindre l'itération finale I_{max}. Il n'est pas nécessaire de dépenser le même nombre d'itérations I_{max} pour décoder chaque bloc de données reçu. Pendant le turbodécodage de la plupart des blocs de données erronés, les valeurs souples LLR correspondantes à l'information a posteriori convergeront après certaines itérations; donc le gain de performance dans le processus itératif dans le restant des itérations est très insignifiant. Si les données souffrent d'un bruit plus faible, il existe une forte probabilité que l'itération liée à la convergence soit inférieure à la valeur I_{max} prédéfinie. Le processus de décodage entre la borne inférieure de convergence et I_{max}, est considéré comme un gaspillage de temps et de calcul. Pour éviter trop d'itérations redondantes au décodage, nous terminerons le décodage itératif de manière dynamique par des *critères d'arrêt anticipé* (early stopping criteria).

Un critère d'arrêt est une technique de décodage dynamique-itératif afin de déterminer si l'algorithme de décodage a convergé pour que les itérations puissent être terminées. Un critère d'arrêt correctement conçu réduit le nombre moyen d'itérations, tout en conservant la même performance en terme de probabilité d'erreur sur les bits BER.

4.3. Critères d'arrêt anticipé – Etat de l'art :

Dans cette section, nous proposons quelques "règles d'arrêt" simples et efficaces qui peuvent être utilisées pour réduire le nombre moyen d'itérations. Ces techniques offrent un compromis entre le débit (throughput) et la performance et peuvent fournir une augmentation

significative du throughput moyen de décodage tout en préservant les performances du décodeur.

De nombreux critères ont été suggérés dans la littérature pour arrêter davantage de nouvelles itérations de décodage lors de la satisfaction de certaines conditions, réduisant ainsi considérablement la dissipation de puissance et/ou augmentant le débit moyen de transmission. Les critères d'arrêt peuvent être classés principalement en trois groupes :

- critères de décision ferme (Hard-decision criteria)
- critères de décision souple (Soft-decision criteria)
- critères ad-hoc reposant sur des informations secondaires (CRC criteria)

Les deux premières sont des techniques d'arrêt basées sur l'amélioration de la qualité du canal. Alors que le troisième groupe s'appui soit sur le calcul des valeurs de contrôle CRC, réalisable à l'aide d'un code de détection d'erreurs qui est le cas échéant, soit en particulier sur un critère irréalisable supposant une connaissance préalable de l'information transmise mais utile en tant que référence de performance (Benchmark) qui est nommé GENIE.

Certaines de ces techniques sont discutées comme suit :

4.3.1. Le Critère de l'Entropie Croisée CE (Cross-Entropy) :

Une fois que le turbodécodage a convergé, les probabilités a posteriori $L(u_t|\boldsymbol{r})$ de chaque composant décodeur (DEC1, DEC2) devraient être en accord. D'où une mesure utile de la convergence du décodeur est la similarité de ces distributions. Une façon de mesurer la similarité de deux fonctions de distribution de probabilité est de calculer la divergence de Kullback-Leibler également connue sous le nom de l'entropie croisée.

L'entropie croisée CE est une mesure de la différence entre deux distributions de probabilité, P la distribution réelle et Q la distribution de référence pour la variable aléatoire X. Lorsque P et Q sont des distributions discrètes, l'entropie croisée CE est donné par :

$$D(P \parallel Q) = \sum_j p_j \log_2 \frac{p_j}{q_j} \tag{4.1}$$

Si les distributions sont les mêmes, l'entropie croisée CE est nul. Autrement-dit plus elles sont proches, plus la valeur de $D(P \parallel Q)$ est petite.

Hagenauer et al. [18] a pris l'idée du critère de l'entropie croisée pour détecter la convergence entre les sorties de chaque composant décodeur du turbodécodeur. En d'autres termes, le critère d'entropie croisée (CE) est utilisé pour mesurer la proximité des deux distributions de sortie $L_l^1(u_t|\boldsymbol{r})$ et $L_l^2(u_t|\boldsymbol{r})$ des deux composants décodeurs [49] (équations (3.48) et (3.49)) soumis à une séquence de données indépendante u_t. A l'itération l, l'entropie croisée $T(l)$ de $L_l^2(u_t|\boldsymbol{r})$ par rapport à la distribution de référence $L_l^1(u_t|\boldsymbol{r})$ est [18]:

$$T(l) \approx \sum_t \frac{\left|\Delta L_{e,l}(u_t)\right|^2}{\exp(|L_l^1(u_t|\boldsymbol{r})|)} \tag{4.2}$$

Sachant que $\Delta L_{e,l}(u_t) = L_{e,l}^2(u_t) - L_{e,l}^1(u_t)$.

En pratique, nous déterminons que la convergence s'est produite lorsque l'entropie croisée devient suffisamment petite. Il est pris comme critère d'arrêt si :

$$T(l) < \text{un seuil} \tag{4.3}$$

Dans [18], il est montré qu'on peut prendre le seuil d'arrêt du décodage itératif dans la gamme de $10^{-2}T(1)$ à $10^{-4}T(1)$. Cette méthode est complexe à mettre en œuvre en raison des opérations non linéaires impliquées et de l'utilisation et le stockage de nombres réels.

4.3.2. Le Critère du Rapport de Changement de Signe SCR :

Le critère SCR (Sign Change Ratio), suggéré dans [50], comme une simplification du critère CE, est obtenu comme suit.

Soit $C(l)$ le nombre de changements de signe entre $L_{e,l}^2(u_t)$, $t = 0,1,\ldots,K-1$ et $L_{e,l-1}^2(u_t)$. Expérimentalement, et en effectuant une approximation sur l'expression de $T(l)$, il a été trouvé que si :

$$C(l) < \varepsilon K \tag{4.4}$$

Où ε est typiquement compris entre 0.005 et 0.03, le décodage itératif peut être terminé et que la performance de ce critère d'arrêt SCR est similaire à celle du critère d'entropie croisée.

Cette méthode est beaucoup plus simple que la méthode CE, mais implique des opérations sur des nombres réels et le stockage en mémoire de nombres entiers.

4.3.3. Le critère HDA (Hard-Decision-Aided) :

Egalement suggéré dans [50], cette méthode compare les décisions dures (c'est-à-dire les bits décodés) des itérations courantes et précédentes, si la relation suivante est satisfaite, le décodage itératif est terminé.

$$\hat{u}_t^1(l) = \hat{u}_t^2(l) \ , \quad \forall t, \ 0 \leq t \leq K-1 \tag{4.5}$$

Avec : $\hat{u}_t^i(l) = sign\left(L_l^i(u_t|\boldsymbol{r})\right)$

Cette méthode implique uniquement des opérations binaires et le stockage de K-bits (c'est-à-dire, des décisions dures de l'itération précédente), où K est la longueur de bloc d'information. Cette méthode est également utilisée dans les turbodécodeurs des normes LTE-advanced de 3GPP et IEEE-802.16m.

On peut voir que les critères d'arrêt suggérées dans [51,52] sont directement/indirectement extraits des critères SCR et HDA (c'est-à-dire, test d'informations extrinsèques, LLR et/ou décisions dures).

4.3.4. Le critère CRC :

Il s'agit d'une règle d'arrêt fondée sur la détection de séquences décodées erronées à l'aide d'un code de redondance cyclique externe (CRC) appliqué à des bits décodés après la décision dure [53]. Un code de détection d'erreur séparé, tel qu'un code CRC, peut être concaténé en tant que code externe avec un turbocode interne afin de marquer les séquences erronées du turbodécodage. La condition d'arrêt avec cette règle est satisfaite chaque fois que le syndrome du code CRC est nul. Cette technique des sommes de contrôle CRC fait partie de la norme sans fil 3G-WCDMA.

4.4. Proposition de nouveaux critères d'arrêt du décodage itératif :

Dans cette section, deux nouvelles techniques d'arrêt anticipé du décodage itératif, de faible complexité, sont proposées pour réduire la consommation d'énergie du turbodécodeur basé sur l'algorithme Log-MAP et/ou augmenter le débit moyen de transmission. Les performances et la complexité des nouvelles techniques sont comparées à celles d'autres techniques dynamiques-itératives représentants l'état de l'art.

4.4.1. Le critère d'arrêt HLLR :

4.4.1.1. L'histogramme LLR comme une mesure de convergence :

Selon le principe du décodage itératif, le LLR de l'entrée a priori d'un composant décodeur provient du LLR de la sortie extrinsèque de l'autre composant. L'observation des LLRs d'un composant décodeur permet de caractériser la convergence du turbodécodeur vers la trame décodée finale.

Dans un algorithme de décodage Soft-In-Soft-Out (LogMAP), le LLR a posteriori en sortie d'un composant décodeur peut être modélisé par une variable aléatoire gaussienne indépendante, où sa variance correspond à la variance du canal AWGN, et sa valeur moyenne au bit systématique transmis. C'est le signe de LLR qui détermine le symbole binaire, négatif pour le bit 0 et positif pour le bit 1, et la valeur absolue quantifie la certitude de la décision dure du décodeur. Ainsi, lorsque les valeurs LLR sont proches de zéro, la décision du décodeur est ambiguë.

Les histogrammes des deux sorties a posteriori $\left(L_l^1(u_t|\boldsymbol{r})\ ,\ L_l^2(u_t|\boldsymbol{r})\right)$ des composants décodeur sont représentés sur la Figure.4.1 en prenant l'indice d'itération comme paramètre. Il est évident que l'histogramme converge vers la distribution gaussienne bilatérale centrée sur deux moyennes, qui représentent les deux symboles binaires, de façon proportionnelle au nombre d'itérations.

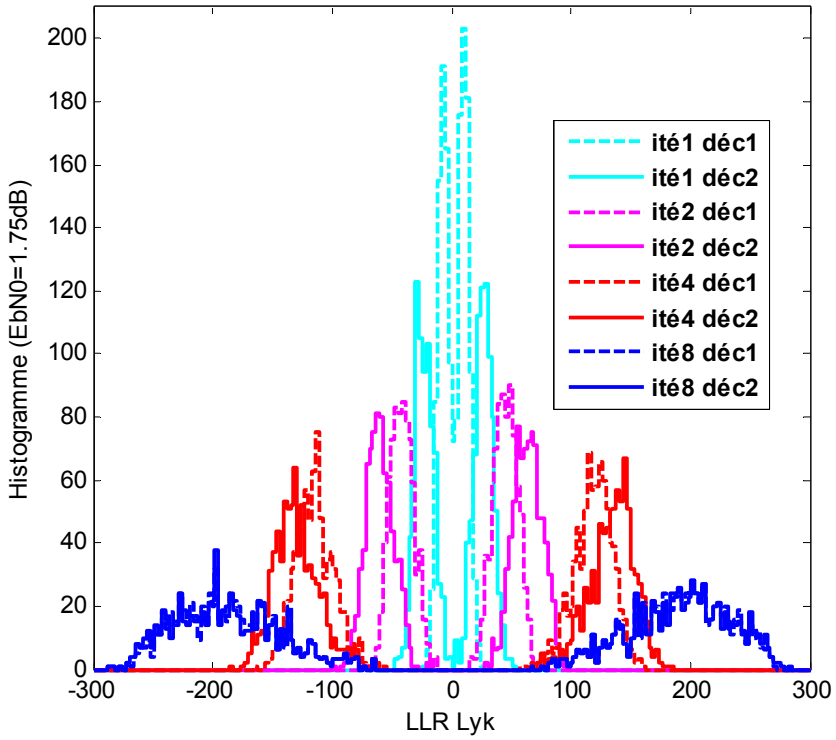

Figure 4.1. Convergence de l'histogramme de LLR a posteriori pendant les itérations

En observant le nombre de bits erronés par trame aux sorties des deux composants décodeur, pour des taux SNR élevées et après chaque itération, on peut conclure que la plupart des trames sont correctement décodées (zéro bits erronés) juste après quelques itérations sans atteindre la limite supérieure des itérations I_{max}.

D'autre part, pour des taux SNR faibles, la limite supérieure des itérations n'est pas suffisante pour atteindre zéro bits erronés, de plus, des itérations supplémentaires n'améliorent pas la situation et les valeurs LLR sont saturées.

4.4.1.2. Le critère HLLR d'arrêt anticipé proposé :

De la discussion de la section précédente, il a été montré que l'histogramme du LLR de sortie donne une information importante sur la convergence du processus de décodage itératif dans les deux situations du SNR. Par conséquent, nous proposons un nouveau critère d'arrêt basé sur la métrique HLLR comme suit.

En turbodécodage, l'amplitude du LLR est un indice quantifiant la fiabilité de la décision du décodage de l'information. Nous divisons donc l'intervalle de variation du LLR en sous-intervalles que nous appelons les bacs qui sont centrés autour de B_j. Cependant, en comptant les valeurs des LLRs appartenant à chaque bac $HLLR(j)$ pour tout les bits du bloc

d'information à décoder après chaque demi-itération (après chaque sortie d'un composant décodeur), on peut observer leur changement au cours des itérations ou demi-itérations.

Soit w la largeur des bacs, on calcule le nombre de bacs (un nombre impair) par :

$$nomBac = 2J + 1 \tag{4.6}$$

Avec J définie par :

$$J = \left\lfloor \frac{1 + \log_2 K}{w} \right\rfloor \tag{4.7}$$

Avec K la taille du bloc à décoder.

Ensuite, les intervalles des variations de LLRs sont centrés autour de B_j obtenu par :

$$B_j = jw \quad , \quad j \in \mathbb{Z} : -J \leq j \leq J \tag{4.8}$$

Soit aussi la fonction indicatrice $FI(x)$, qui vaut 1 lorsque son argument est vrai, définie ainsi :

$$FI(x) = \begin{cases} 1 & si -\frac{1}{2} \leq x < \frac{1}{2} \\ 0 & sinon \end{cases} \tag{4.9}$$

L'histogramme des rapports de vraisemblance logarithmique est obtenu par :

$$HLLR(j) = \sum_{t=0}^{K-1} FI\left(\frac{L_l^i(u_t|\boldsymbol{r}) - B_j}{w}\right) \tag{4.10}$$

On note ici que : $i = 1{,}2$ l'indice des composants décodeurs (DEC1 et DEC2), et $l = 1{,}2, \ldots, I_{max}$ l'indice d'itération.

Le nombre et la taille des bacs sont importants en termes de complexité d'implémentation matérielle ou d'efficacité du critère d'arrêt, notamment le bac central ($HLLR(j)$).

Clairement, il est évident que le décodage est complètement correct et peut être terminé lorsque le bac central de l'histogramme $HLLR(0)$ est nul, ce qui signifie qu'il n'y a pas de valeurs ambiguës de LLR. Par conséquent, itérer davantage le turbodécodeur ne fera qu'amplifier les valeurs absolues des LLRs.

Lorsque la situation de décodage insoluble est rencontrée, les bacs des extrémités $HLLR(\pm J)$ restent nuls et le bac central $HLLR(0)$ ne descend pas sous une certaine valeur et oscille autour d'elle. Dans ce cas, le décodage peut être terminé après cette observation, précisément lorsque la valeur du bac central est toujours supérieure à 4% de la longueur de la trame et décroît avec moins de 5% de sa valeur précédente après les trois premières itérations. Le décodeur peut à ce moment envoyer une demande de retransmission de données ARQ à l'émetteur.

4.4.1.3. Résultats de simulation :

Dans cette section, nous étudions les performances du critère proposé en termes de taux d'erreur binaire et de nombre moyen d'itérations, en fonction de $\frac{E_b}{N_0}$, le rapport signal-à-bruit au niveau des bits de l'information. Les simulations suivantes sont effectuées sur un canal AWGN, avec une modulation BPSK. Nous avons utilisé un turbocode de taux $R = \frac{1}{3}$, dont le polynôme générateur est $\boldsymbol{G} = \left[1 \ , \ \frac{1+D^2}{1+D+D^2}\right]$, et avec un entrelaceur non régulier aléatoire.

Pour chaque valeur E_b/N_0, nous avons transmis au moins 1000 blocs de données, chacun de longueur $K = 1568$ bits. Le nombre maximal d'itérations I_{max} a été fixé à 10.

Pour les bacs du HLLR, nous avons utilisé différentes largeurs $w = 3,4,5,6,7, et\ 9$ sur l'intervalle $[-3w\,, 3w]$. Ces valeurs ont été choisies après plusieurs essais et tests en simulation, dans le but de couvrir une gamme raisonnable de taux d'erreur de trame non détectés.

Les figures 4.2-a et 4.2-b montrent des histogrammes types $HLLR(j)$ et leurs évolutions en fonction des itérations sous de bonnes conditions de canal (c'est-à-dire, des E_b/N_0 élevés) et sous des conditions de canal médiocres (c'est-à-dire à des E_b/N_0 faibles) respectivement.

On appel ces deux cas :

- Une trame (bloc d'information) soluble, ce qui veut dire que la trame correcte peut être obtenu après quelques itérations, et d'autres itérations sont redondantes.
- Une trame insoluble, où sa version correcte ne peut pas être atteinte après plusieurs itérations; par conséquent, une grande quantité d'énergie est consommée, sans décoder la trame correctement.

Nous constatons que la certitude de décision LLR des bits décodés au fil des itérations est de plus en plus forte, et le nombre de bits décodés avec incertitude est zéro ou presque après les premières itérations. De l'autre coté, de la trame insoluble, nous constatons que les valeurs de la certitude de décision LLR des bits décodés restent proches de zéro (décision ambigüe) pendant plus de trois itérations.

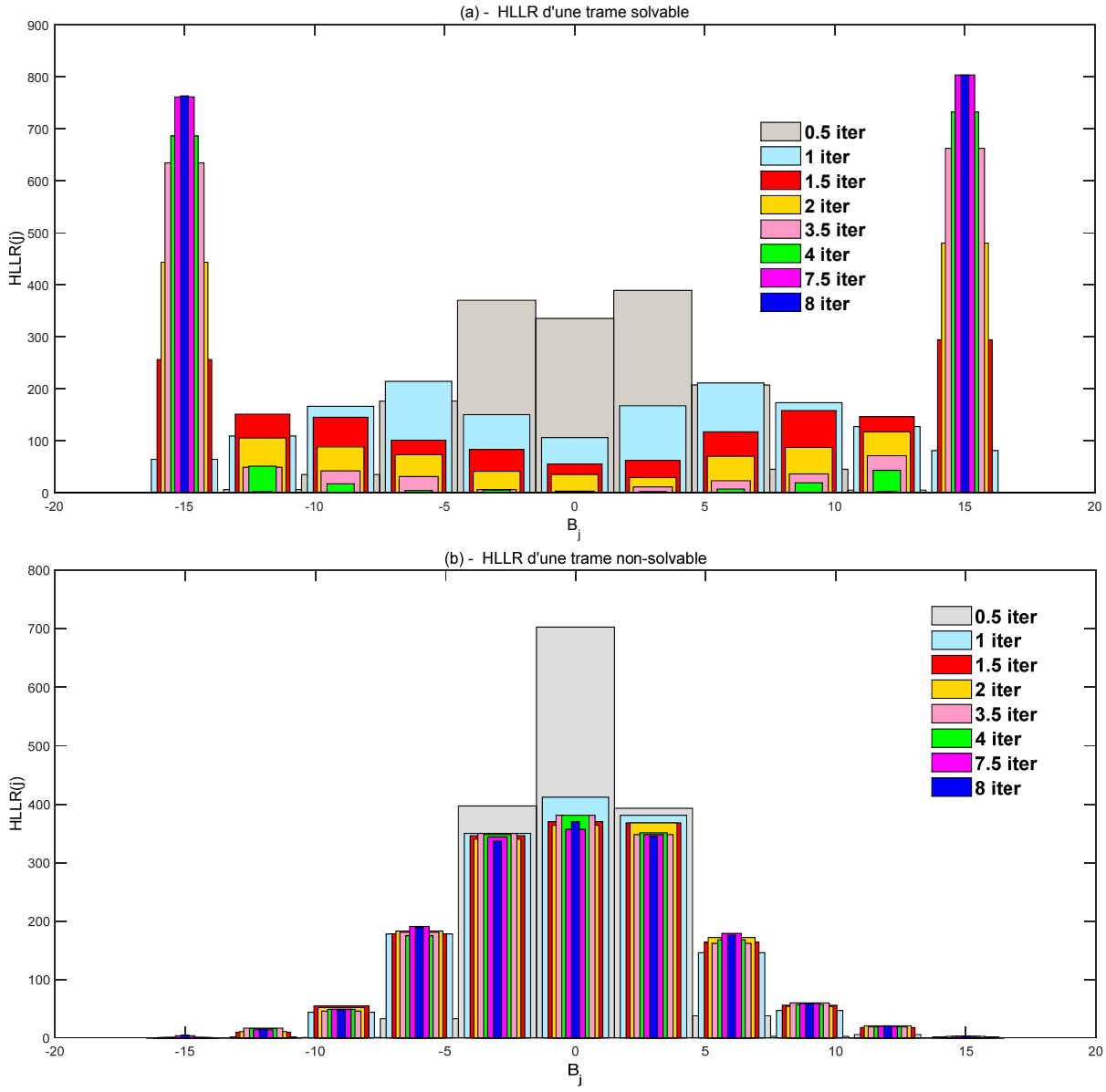

Figure 4.2. Histogrammes types $HLLR(j)$: (a) trame soluble, (b) trame insoluble

Pour comparer le critère d'arrêt proposé avec les précédents, nous avons choisi deux autres critères: 1) GENIE qui est utile pour établir un benchmark de performance imbattable par rapport auquel d'autres critères sont mesurés. 2) HDA (Hard Decision Aided) [50] qui est une règle bien connue et qui tente de détecter les séquences décodées correctement en

évaluant les décisions durs des bits à la fin de chaque itération par rapport à l'itération précédente.

La figure.4.3 montre les résultats du taux d'erreur binaire BER en fonction de E_b/N_0. Toutes les méthodes simulées ont presque les mêmes performances en ce qui concerne le BER dans la zone caractérisée par des conditions de canal médiocres. Lorsque E_b/N_0 augmente, une légère dégradation des performances peut être constatée par rapport à la courbe de référence GENIE, ce qui est acceptable pour la plupart des systèmes de communication (BER $<10^{-4}$).

On remarque aussi que la valeur de w la largeur des bacs, est un paramètre important pour le critère proposé HLLR. Il est directement lié au seuil de normalisation de la fonction indicatrice dans l'équation (4.10), ce qui mène à recenser un grand nombre de LLR a priori sur un intervalle plus large du bac central de l'histogramme $HLLR(0)$. On explique ce comportement par une décision du critère d'arrêt anticipé plus strict envers les valeurs ambiguës.

Ces déductions sont clairement expliquées par les résultats de la figure.4.4. Dans cette dernière un deuxième angle de vue pour évaluer les performances de notre critère d'arrêt est représenté. Dans cette simulation, le nombre moyen d'itérations le long de toutes les trames décodées est calculé par rapport à E_b/N_0. Le critère HLLR proposé réalise une performance très proche de celle du benchmark GENIE. De plus, lorsque le rapport signal-à-bruit augmente, HLLR fonctionne mieux que HDA, par exemple à $E_b/N_0 = 0.5dB$, la méthode HLLR9 réalise un nombre moyen d'itérations inférieur à celui atteint par HDA avec une différence d'environ 0.835 itération. Cette différence est très importante si elle est interprétée comme une réduction de la complexité et une économie de l'énergie d'environ 25,14% en éliminant les opérations inutiles. Bien sûr, c'est le coût de la légère dégradation de la performance en BER d'environ -1.2133×10^{-5}.

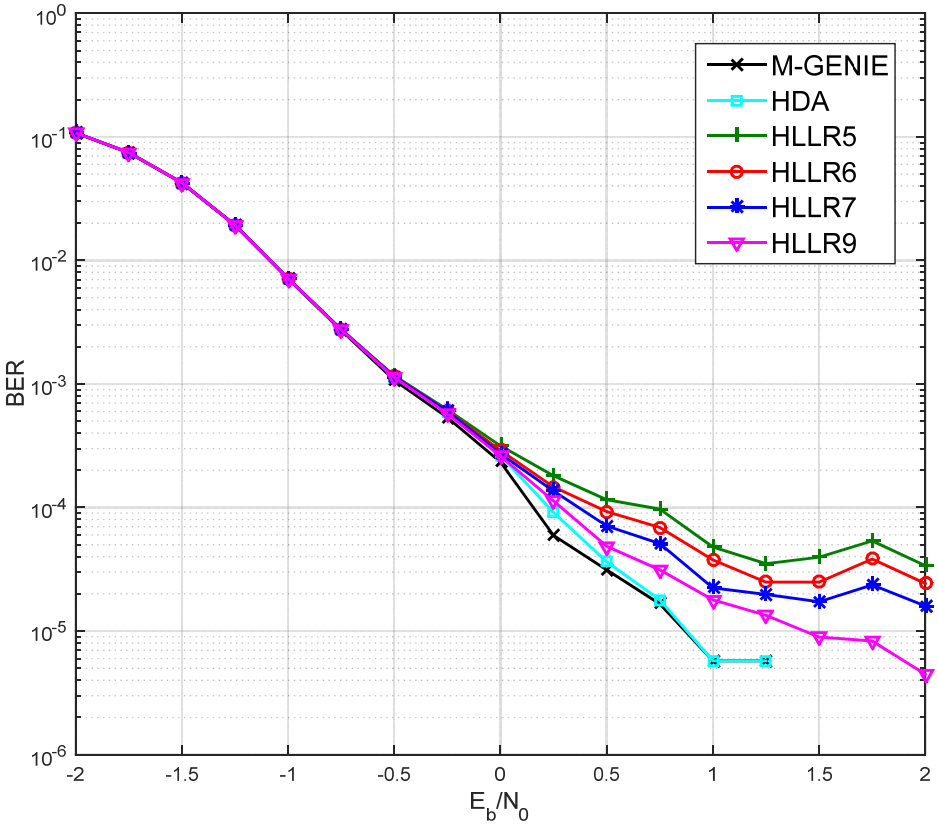

Figure 4.3. BER de GENIE, HDA et HLLR (pour $w = 5, 6, 7, 9$)

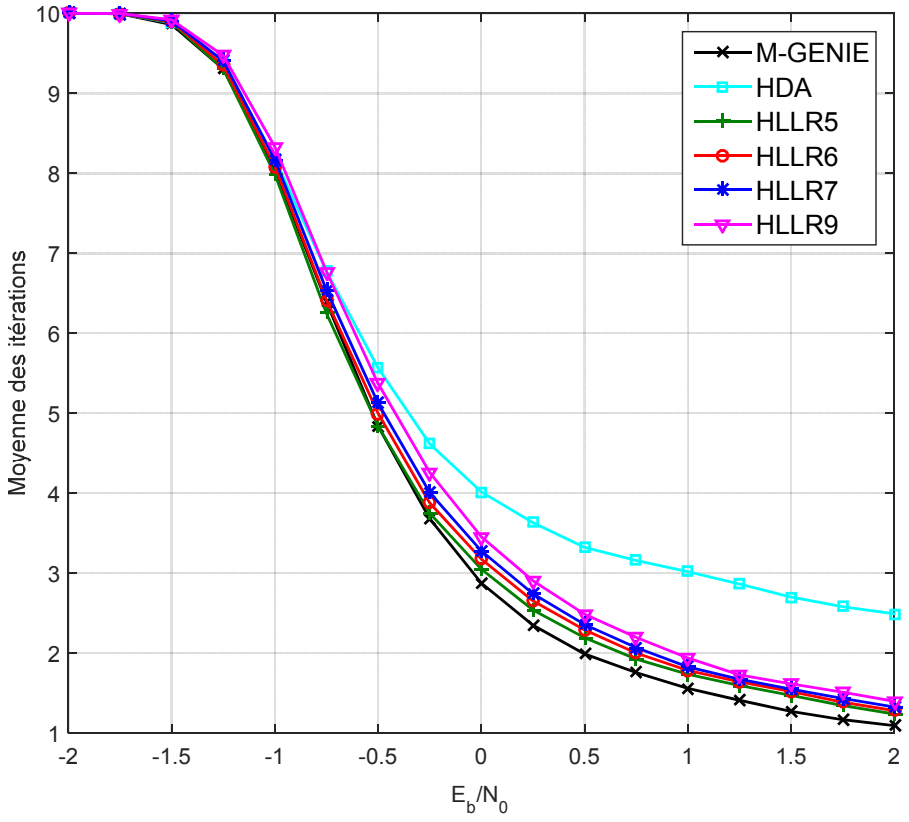

Figure 4.4. Moyenne des itérations de GENIE, HDA et HLLR (pour $w = 5, 6, 7, 9$)

4.4.2. Critère d'arrêt de l'entropie croisée partiel avec normalisation adaptative PCE :

Dans la littérature, de nombreux critères d'arrêt anticipé ont été introduits [49, 53-70] pour réduire la complexité et la consommation d'énergie en mettant fin au processus de décodage itératif dès que le résultat d'une itération $l < I_{max}$ de décodage est suffisamment fiable et qu'une amélioration supplémentaire est peu probable dans le restant des itérations.

Bien que beaucoup d'entre eux soient efficaces pour les trames solubles avec des rapports signal-à-bruit (E_b/N_0) élevés, ils échouent pour les trames insolubles où le décodeur n'a pas la capacité de décoder et les itérations ultérieures ne sont qu'une perte de temps et d'énergie, ce qui se répercute sur la latence de décodage. De plus, certaines techniques sont contraintes par une itération complète pour décider de terminer ou non le décodage itératif, de sorte que le nombre d'itérations minimum contrôlable est 1 ou 2 itérations.

Dans la transmission en temps réel, la fiabilité peut être améliorée en combinant des codes de correction d'erreurs directe (FEC) avec des protocoles de retransmission (ARQ). Cette combinaison est connue sous le nom de protocole de retransmission hybride (HARQ). Cependant, de nombreux efforts de recherche concernent surtout les turbo-codes combinés dans les protocoles HARQ. Lorsque les mots de code ne sont pas décodés avec succès au niveau du récepteur, un message NAK (Negative Acknowledgement) est renvoyé à l'émetteur pour demander la retransmission. Les échecs de décodage fréquents impliquent que le décodeur atteigne le nombre d'itérations I_{max} plusieurs fois, et par conséquent, une grande partie de l'énergie est gaspillée si une trame est retransmise plusieurs fois.

Il est un peu ironique de consommer une grande part d'énergie et de temps sur les échecs de décodage, quoiqu'il est difficile de détecter ces mauvaises trames avant le décodage en raison de la difficulté de prévoir le caractère défaillant du signal reçu. Dans ce cas, il est important de concevoir un critère efficace pour terminer le décodage itératif qui peut à la fois, satisfaire aux performances des critères d'arrêt conventionnels sous de bonnes conditions de canal, et aussi terminer prématurément le décodage itératif sous des conditions de canal médiocres. Le temps de décodage restant après cet arrêt prématuré, laisse un temps précieux, pendant lequel intervient un système HARQ, économisant ainsi l'énergie et augmentant le débit moyen de décodage.

Dans cette section, il est supposé que le décodeur peut rencontrer la situation insoluble où il y a peu d'informations pour qu'il puisse aboutir correctement à la trame transmise. Dans ce cas, il est important de dériver un critère d'arrêt anticipé du processus de décodage itératif qui soit efficace dans une telle situation, afin d'éviter ces calculs inutiles et gagner un temps de décodage.

De plus, on a combiné une technique de normalisation avec le critère d'arrêt proposé, compensant l'éventuelle perte de performance due à l'arrêt anticipé du décodage itératif. La normalisation est donc effectuée avec un facteur de correction adaptatif qui ajuste les informations de fiabilité trop optimistes dans le décodeur SISO sous des conditions de canal médiocres.

4.4.2.1. Etat de convergence individuel des bits basé sur l'entropie croisée :

Sélection de bits avec un test de convergence de l'entropie croisée :

Soit $\boldsymbol{u} = \{u_t,\ 1 \leq t \leq K\}$ une trame de données transmise de longueur K. Le turbodécodeur (utilisant l'algorithme logMAP) donne en sortie les estimations $\widehat{\boldsymbol{u}} = \{\hat{u}_t,\ 1 \leq t \leq K\}$.

En supposant que les LLRs a posteriori des bits d'information sont statistiquement indépendants après chaque itération. Il a été montré dans [18], [54], [55] que l'entropie croisée $T(l)$ de la trame en sortie du turbodécodeur, donnée par l'équation (4.2), n'est que la somme sur le long de la trame de l'entropie croisée $T^{(l,i)}(u_t)$ du bit u_t.

On admettant le changement de notation $T_{block}^{(l,i)} = T(l)$, l'entropie croisée de la trame et du bit u_t sont respectivement :

$$T_{block}^{(l,i)} = T(l) = \sum_{t=1}^{K} T^{(l,i)}(u_t) \tag{4.11}$$

$$T^{(l,i)}(u_t) \approx \frac{\left|\Delta L_{e,l}(u_t)\right|^2}{\exp\left(\left|L_l^i(u_t|\boldsymbol{r})\right|\right)} \tag{4.12}$$

Avec i l'indice du composant décodeur LogMAP ($i = 1,2$).

Les résultats de simulation sur ordinateur [54,55] ont montrés qu'une fois la sortie du turbodécodeur après $l^{\text{ème}}$ itération converge, $T_{block}^{(l,i)}$ diminue d'un facteur entre 10^{-2} à 10^{-4} par

rapport à sa valeur après la première itération $T_{block}^{(1,i)}$. Il est donc raisonnable de l'utiliser comme critère d'arrêt du décodage itératif :

$$T_{block}^{(l,i)} < 10^{-3} T_{block}^{(1,i)} \tag{4.13}$$

Notons que l'équation (4.13) compare essentiellement l'état de convergence moyenné sur tous les bits de la trame et ne reflète pas l'état de convergence des bits individuels. Un autre critère basé sur l'état de convergence des bits individuels est donné par [55] :

$$T^{(l,i)}(u_t) < 10^{-3} T^{(1,i)}(u_t) \tag{4.14}$$

Les bits de la trame qui satisfont à l'équation (4.14) sont classés, des bits convergents et les autres restant des non-convergents. Ainsi, les estimations des bits d'information à la sortie peuvent être réécrites en deux parties, à savoir l'ensemble des bits non-convergents $\widehat{\boldsymbol{u}}_{\Omega}$ et celui des bits convergents $\widehat{\boldsymbol{u}}_{\overline{\Omega}}$ où Ω, $\overline{\Omega}$ représente les ensembles des indices définis par :

$$\begin{aligned} \Omega &\equiv \left\{ j | T^{(l,i)}\left(u_j\right) \geq 10^{-3} T^{(1,i)}\left(u_j\right), 1 \leq j \leq K \right\} \\ \overline{\Omega} &\equiv \{ 1 \leq j \leq K \} - \Omega \end{aligned} \tag{4.15}$$

4.4.2.2. Dérivation du critère d'arrêt basé sur l'ensemble des bits non-convergents :

Sur la figure.4.5, l'évolution des parties convergente et non-convergente de l'entropie croisée, avec la progression du décodage itératif (en fonction de l), est représentée sous des conditions de canal différentes (E_b/N_0 faible/moyen/élevé).

Cette illustration fournit un aperçu utile sur le comportement du décodage itératif. On peut voir sur la figure que l'amplitude de l'entropie croisée de la partie convergente est négligeable par rapport à la partie non convergente. De plus, la pente de ce dernier est décroissante (monotone). Ainsi, la vitesse de convergence du décodage itératif est quasiment basée sur l'ensemble des bits non-convergents, qui sont déterminants dans un critère d'arrêt anticipé, et donc il est plus judicieux de s'intéresser plus à l'entropie croisée partielle que l'on notera $T_{\Omega}^{(l,i)}$ c.à.d. l'entropie croisée de l'ensemble non-convergent.

Ainsi, le critère d'arrêt basé sur l'entropie de la trame entière de l'équation (4.13) est principalement déterminée par l'entropie croisée individuelle des bits non-convergents qui est la somme de l'équation (4.11) mais sur l'ensemble Ω, soit :

$$T_{\Omega}^{(l,i)} = \sum_{j \in \Omega} T^{(l,i)}(u_j) \tag{4.16}$$

En outre, il a été constaté [56] que les rapports de vraisemblance logarithmique de sortie des composants décodeurs LogMAP ne prédisent pas correctement la probabilité a posteriori de la décision dure pour les conditions de canal médiocres. Ces informations de fiabilité de la sortie du décodeur sont trop optimistes, et la sortie peut être considérée comme multipliée par un facteur, qui dépend du taux d'erreur binaire (BER) actuel. Pour se rapprocher du vrai rapport de vraisemblance logarithmique LLR, la sortie doit être normalisée. Cette correction ou mise à l'échelle des LLRs est nécessaire pour améliorer les performances.

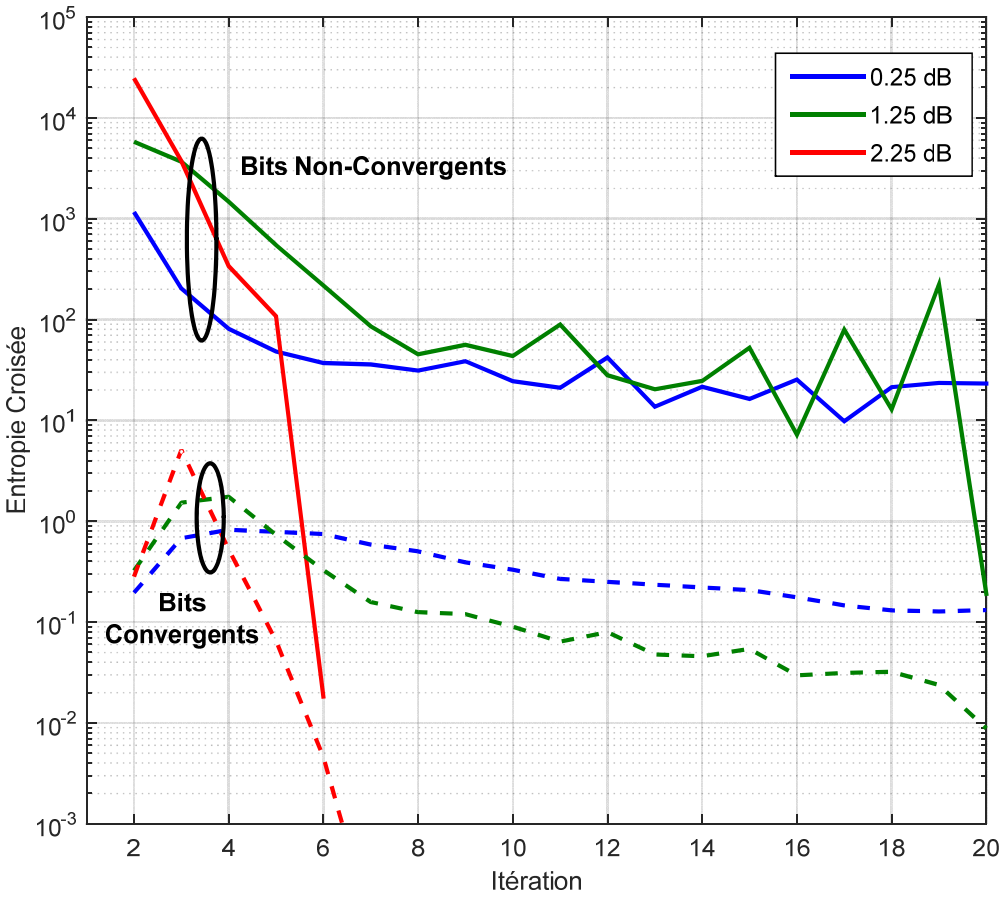

Figure 4.5. Evolution de l'entropie croisée sur l'ensemble convergent $\overline{\Omega}$ et non-convergent Ω en fonction des itérations

En conséquence, nous avons choisi de normaliser les LLRs trop optimistes des bits non convergents, par un facteur adaptatif qui aura un effet compensateur de la faible dégradation des performances du critère d'arrêt proposé. Ainsi, à chaque itération, nous avons multiplié les informations extrinsèques aux sorties des décodeurs LogMAP par un facteur de normalisation noté $C_{NSD}^{(l,i)}$:

$$\tilde{L}_l^1(u_j) = C_{NSD}^{(l-1,1)}\, \Pi^{-1}\left(\tilde{L}_{e,l-1}^2(u_j)\right) \tag{4.17}$$

$$\tilde{L}_l^2(u_j) = C_{NSD}^{(l,2)}\, \Pi\left(\tilde{L}_{e,l}^1(u_j)\right) \tag{4.18}$$

Avec : $j \in \Omega$, ainsi que $\tilde{L}_l^1(u_j)$, $\tilde{L}_{e,l}^1(u_j)$, $\tilde{L}_l^2(u_j)$ et $\tilde{L}_{e,l}^2(u_j)$ sont les versions normalisées de $L_l^1(u_j)$, $L_{e,l}^1(u_j)$, $L_l^2(u_j)$ et $L_{e,l}^2(u_j)$ respectivement pour les bits non-convergents seulement.

Les facteurs de normalisation sont calculés en considérant les bits de la trame entière en estimant le nombre de différences de signe NSD (Number of Sign Difference) entre la sortie extrinsèque et l'entrée a priori de la trame d'information à la $l^{\text{ème}}$ itération. Ils sont donnés par :

$$C_{NSD}^{(l,1)} = 1 - \frac{NSD^{(l-1,2)}(\hat{\boldsymbol{u}})}{K} \tag{4.19}$$

$$C_{NSD}^{(l,2)} = 1 - \frac{NSD^{(l,1)}(\hat{\boldsymbol{u}})}{K} \tag{4.20}$$

Nous aboutissons enfin à la règle d'arrêt anticipé de l'entropie croisée modifiée que l'on appelle entropie croisée partielle PCE (Partial Cross-Entropy). Ce critère se concentre uniquement sur les bits non-convergents. Les particularités adaptatives de normalisation amélioreront la capacité du critère d'arrêt proposé pour détecter non seulement les trames solubles dans des conditions de canal favorables, mais aussi pour arrêter le décodage itératif des trames non-solubles sous des conditions de canal défavorables.

Le critère d'arrêt anticipé de l'entropie croisée partielle proposé basé sur la sélection et normalisation de bits non-convergents est donné par :

$$T_{\Omega}^{(l,i)} < 10^{-3} T_{block}^{(1,i)} \tag{4.21}$$

4.4.2.3. Résultats de simulation et discussion :

Les différents paramètres de simulation utilisés dans ce critère (PCE) sont identiques à ceux du critère HLLR pour mettre en comparaison les résultats obtenus. Nous avons utilisé un turbocode de taux $R = \frac{1}{3}$, dont le polynôme générateur est $\boldsymbol{G} = \left[1\ ,\ \frac{1+D^2}{1+D+D^2}\right]$, et avec un entrelaceur non régulier aléatoire pour la transmission sur un canal AWGN, avec une modulation BPSK. Nous transmettons au moins $B_K = 1000$ trames chacune de taille $K = 1568$ bits, et $I_{max} = 10$.

La performance du critère PCE proposé est évaluée en termes de taux d'erreurs binaires BER et de nombre moyen d'itérations. Les résultats obtenus sont mis en comparaison avec les critères HLLR, HDA, CE, NSD et le benchmark GENIE. De plus le critère PCE est appliqué dans un système HARQ de Type-I qui utilise un maximum de retransmissions $Rq_{max} = 4$.

La performance du système HARQ est évaluée en termes de débit (throughput). Rappelons que le débit, dans un système de codage de canal avec protocole HARQ, est défini comme le nombre moyen de trames acceptées dans le temps nécessaire pour transmettre une trame.

$$Throughput = \frac{1}{B_K}\sum_{i=1}^{B_K}\frac{1}{1+Rq(i)} \tag{4.22}$$

Nous proposons une nouvelle formule pour mesurer le débit du système HARQ-Type-I plus adapté au décodage itératif utilisant un critère d'arrêt anticipé. Cette nouvelle formule mesure mieux la capacité du critère d'arrêt dans un protocole HARQ a augmenter le débit moyen du système en profitant du temps gagné par l'arrêt anticipé du turbodécodeur, c.à.d. prendre en compte le nombre moyen d'itérations pour décoder la trame reçue.

Ce débit qu'on note $Throughput_{PCE}$ est calculé comme suit, sachant que $Itr(i,j)$ est le nombre d'itérations atteint par la trame i durant la retransmission j.

$$Throughput_{PCE} = \frac{1}{B_K}\sum_{i=1}^{B_K}\frac{I_{max}}{\sum_{j=1}^{Rq_{max}+1} Itr(i,j)} \tag{4.23}$$

L'équation (4.23) donne une mesure conjointe du temps économisé par le critère d'arrêt et du temps consommé par la retransmission, ce qui est plus précis et montre les performances réelles du système.

Le nombre de différences de signe NSD entre la sortie extrinsèque $L_{e,l}^{i}(u_t)$ et l'entrée a priori $L_l^i(u_t)$, c'est-à-dire les bits sur lesquels les deux composants décodeurs sont en désaccord, est utilisé comme seuil de retransmission, mais cela seulement lorsque le décodage itératif est terminé par le critère d'arrêt PCE.

La figure.4.6 montre le BER tracé en fonction de E_b/N_0 pour un turbodécodeur log-MAP. En bleu le turbodécodage conventionnel avec un nombre d'itération fixe $I_{max} = 10$, et

en rouge utilisant une normalisation adaptative partielle. Comme on peut le voir, le BER a été amélioré au moyen de la normalisation adaptative partielle.

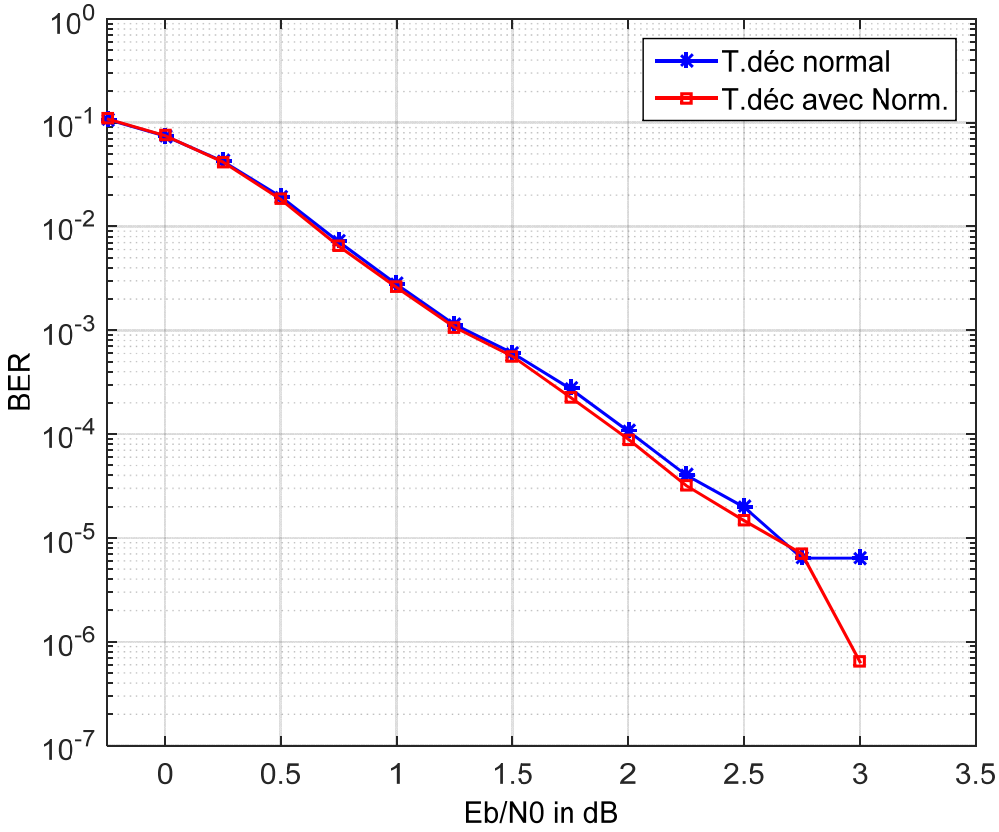

Figure 4.6. Gain en BER du turbodécodage avec normalisation adaptative

Pour observer les limites empiriques réalisables du critère proposé, les simulations pour plusieurs approches présentées précédemment sont effectuées, avec la courbe GENIE fournie comme référence.

Les figures 4.7 et 4.8 illustrent les résultats de simulation du taux d'erreur binaire BER et du nombre d'itérations moyen en fonction de E_b/N_0 pour les critères d'arrêt HDA, NSD, HLLR, CE, et le critère proposé PCE. Ce dernier a été réalisé avec le seuil 10^{-3} pour la sélection des bits non-convergents et le seuil 10^{-2} pour arrêter le décodage itératif. Les cinq techniques sont appliquées à un système de turbocodeur/turbodécodeur modélisé dans l'environnement de programmation MATLAB, avec environ 10 Mbits comme entrée de codeur et un modèle de canal AWGN.

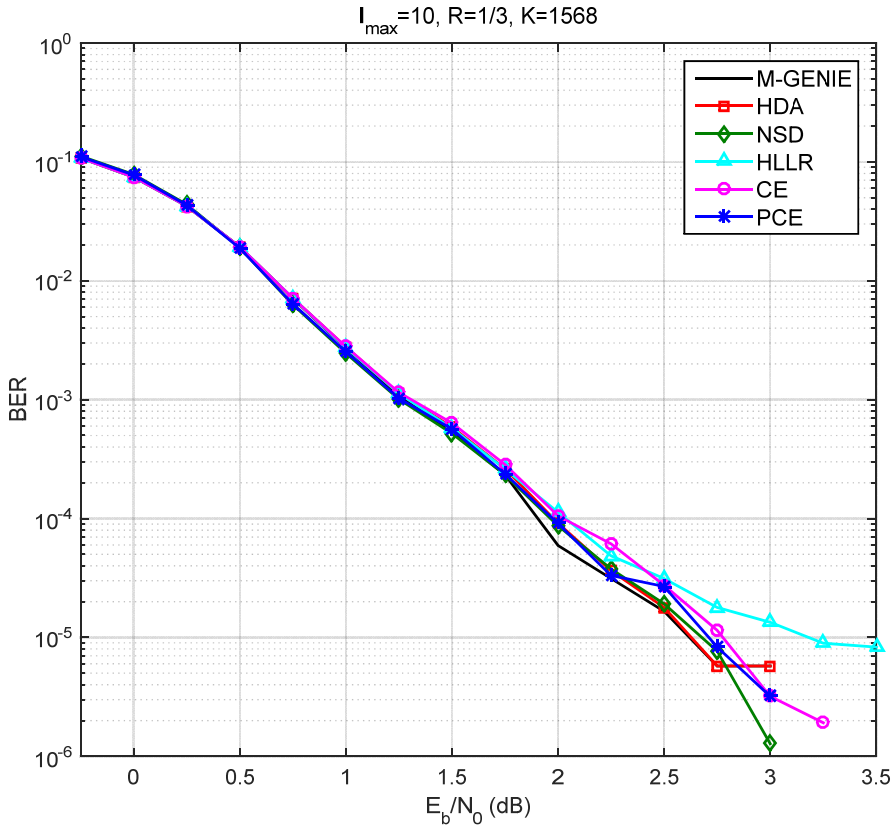

Figure 4.7. Performances de décodage en BER pour les différents critères simulés

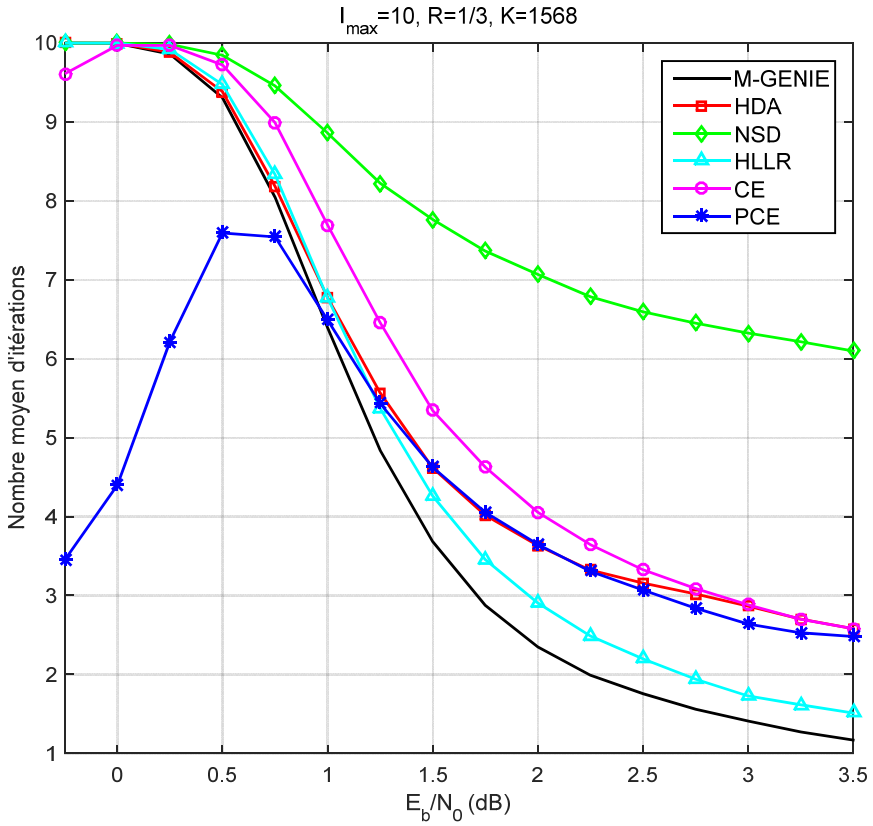

Figure 4.8. Comparaison du nombre moyen d'itérations des critères proposés avec ceux de la littérature

Sur la figure.4.7, les performances en BER des critères simulés sont proches les unes des autres, que ce soit dans des situations de E_b/N_0 faibles ou élevées. On a donc vérifié que le critère PCE est proche des performances de décodage à d'autres techniques dynamiques itératives à faible complexité.

A partir de la figure.4.7, on constate que la performance du turbodécodeur utilisant le nouveau critère est proche de celle du turbodécodeur utilisant la méthode (NSD ou HDA), et meilleure que celle du turbodécodeur utilisant la méthode (CE ou HLLR).

Comme observé à partir du graphe de la figure.4.8, le critère PCE est nettement supérieur dans la réduction d'itération par rapport aux techniques NSD, HDA, et CE. Il est clair aussi qu'il présente un avantage sur les autres critères dans la région de faible rapport signal-à-bruit, c'est-à-dire que le turbo décodeur fonctionne avec très peu d'itérations sans perte significative de performance BER comme on peut le vérifier sur la figure.4.7.

D'autre part, dans la plupart des critères comparés, le turbodécodeur atteint le maximum d'itérations dans des conditions de canal médiocre (faible rapport signal-à-bruit), ce qui est considéré comme une dépense énergétique et un retard inutiles, avant de demander une retransmission, sauf pour le critère proposé où il réduit considérablement le nombre d'itérations requises.

On observe également que le nombre d'itérations moyen donné par le critère PCE est inférieur d'une fraction d'itération de celui donné par les critères (HDA, CE) et plus élevé d'une itération presque que celui donné par le premier critère proposé HLLR.

De toute évidence, la charge de calcul ajoutée, par le critère d'arrêt PCE au décodage itératif, est significativement inférieure à celle des itérations superflues économisées sur un décodage itératif complet.

Dans les simulations suivantes, nous considérons un E_b/N_0 fixe pour toutes les retransmissions, qui est équivalent à sa valeur à la première transmission.

Un système HARQ offre la plupart des améliorations à des niveaux de bruit élevés, puisque la probabilité d'une retransmission est élevée, et donc le gain. Puisque le critère d'arrêt PCE a de bonnes performances pour des niveaux de bruit élevés, là où nous prévoyons de fonctionner, nous utiliserons principalement ce critère pour demander une retransmission.

Le turbocode de taux R=1/3 basé sur deux codes convectifs de polynôme générateur $\boldsymbol{G} = \left[1 \ , \ \frac{1+D^2}{1+D+D^2}\right]$ avec un entrelaceur aléatoire, est utilisé conjointement avec le protocole HARQ de type-I. Le nombre maximal autorisé d'itérations est de 10 et le nombre maximal autorisé de retransmissions est de 4.

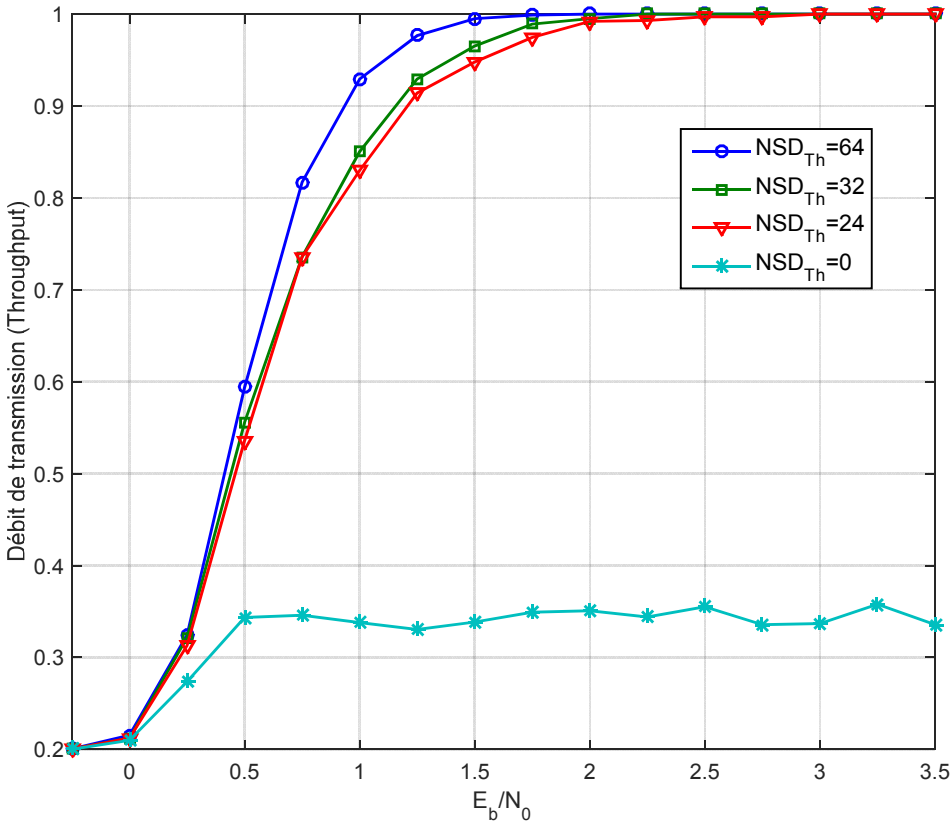

Figure 4.9. Débit du système HARQ en utilisant le critère PCE selon l'équation (4.22).

La figure.4.9 illustre le débit de transmission du système HARQ avec critère d'arrêt anticipé PCE et pour différentes valeurs du seuil de retransmission basé sur le NSD. Ce débit est calculé selon l'équation (4.22). Nous notons que le débit du système régulier, sans protocole de retransmission ni critère d'arrêt anticipé est égal à 1 selon la même équation.

Pour les faibles niveaux de bruit, la probabilité d'une retransmission est très proche de zéro dans cette zone, c'est pourquoi toutes les courbes tendent vers 1 qui est la courbe régulière sans HARQ. Pour les niveaux de bruit très élevés, les courbes des différents seuils sont également superposées. Ceci est dû au fait que la probabilité d'une retransmission tend vers 1. Puisque pratiquement toutes les transmissions sont rejetées, il n'y a rien à gagner d'une retransmission. Cependant, la zone entre ces valeurs extrêmes est la zone de travail des systèmes HARQ et un gain notable peut être observé. Dans le cas du seuil $NSD_{Th} = 0$, les résultats sont en dehors de la plage acceptable car une correspondance parfaite entre les deux

informations extrinsèques des composants décodeurs est difficile, et le nombre maximal de retransmissions est toujours atteint même si la trame décodée est sans erreurs.

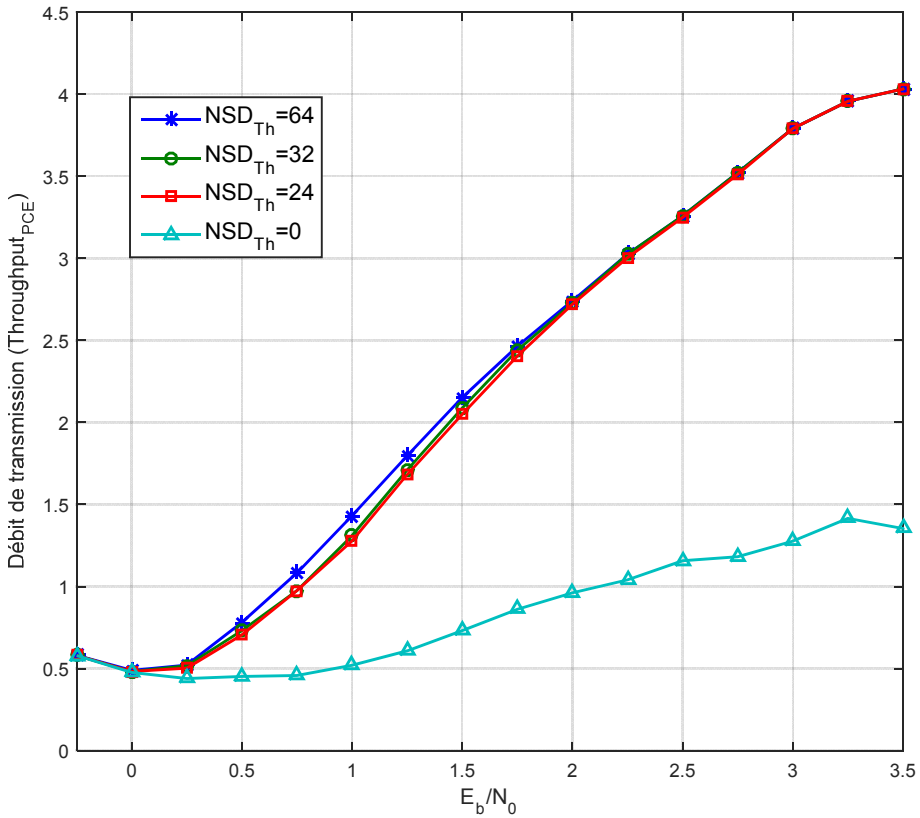

Figure 4.10. Débit du système HARQ en utilisant le critère PCE selon l'équation (4.23).

En considérant l'avantage de l'arrêt anticipé réussi du décodage itératif en moins de la durée requise par décodage itératif normal, on peut exploiter ce gain de temps pour commencer à décoder une nouvelle trame reçue. L'effet cumulatif de ces portions de temps, en particulier dans de bonnes conditions de canal, entraîne une augmentation impressionnante du débit global de transmission du système.

Ceci est démontré par les résultats de simulation illustrés sur la figure.4.10 en calculant le débit avec l'équation (4.23) pour prendre en compte cette propriété. Nous observons sous des conditions de bruit faible, que le système HARQ a la possibilité d'augmenter son débit jusqu'à quatre fois la capacité nominale. Même dans des conditions de bruit élevé, le débit total est maintenu à plus de la moitié de la capacité nominale.

4.5. Conclusion :

Dans ce chapitre, deux techniques améliorant le turbodécodage, économes en énergie, offrant un débit plus élevé et un temps de latence moindre, ont été proposées. L'algorithme de décodage Log-MAP a été choisi pour la mise en œuvre du décodeur SISO constitutif en raison de son compromis performance-complexité optimisé. Deux critères d'arrêt anticipé du décodage itératif ont été appliqués pour réduire le nombre d'itérations du turbodécodeur. Les résultats du premier critère HLLR ont montré de bonnes performances de décodage avec réduction importante du nombre d'itérations moyen.

La deuxième approche impliquait l'application d'un nouveau critère dynamique-itératif de faible complexité (PCE) pour réduire le nombre d'itérations de décodage dans de bonnes ou mauvaises conditions de canal. La majorité des techniques dynamiques-itératives de la littérature ne sont efficaces que dans de bonnes conditions de canal. Le nouveau critère d'arrêt anticipé du décodage itératif PCE s'est avéré supérieur en termes de réduction d'itération.

En outre, un turbodécodeur appliquant la nouvelle technique PCE, combinée avec un protocole de retransmission HARQ de type-I, a été modélisé dans l'environnement de programmation MATLAB, puis simulé. La combinaison du turbodécodage PCE-HARQ permet au système de bénéficier de la réduction de la complexité du décodage itératif et d'améliorer jusqu'à quatre fois le débit de transmission global grâce à une stratégie efficace de détection et de correction d'erreur.

Conclusion et Perspectives

Dans un système de communications numériques, il s'avère que les codes correcteurs d'erreurs sont des techniques efficaces pour lutter contre le bruit introduit lors de la transmission du message. Les turbocodes ont été adoptés par de nombreux standards de télécommunications pour leurs performances qui sont très proches de la limite de Shannon. L'utilisation des turbocodes pose néanmoins de nouvelles contraintes dues au décodage itératif. Parmi ces contraintes, la maîtrise du nombre d'itérations adéquat dans les composants décodeurs LogMAP, qui est primordiale surtout avec la demande croissante de débit par les nouvelles normes des systèmes de communication sans fil.

Le décodage itératif atteint de bonnes performances à mesure que le nombre d'itérations augmente avec le processus de décodage. Cependant, trop d'itérations provoque un fardeau de calcul (consommation de l'énergie) et de latence. Les critères d'arrêt anticipé traitent ce problème, en trouvant un compromis acceptable entre performance et complexité et en réduisant le nombre d'itérations.

Notre travail a abouti à la proposition de deux nouveaux critères d'arrêt anticipé présentés dans cette thèse. Ils ont été testés à partir de données bruitées afin de mettre en évidence leurs capacités et leurs limites. Le premier critère HLLR présenté dans cette thèse est basé sur un concept simple et peut être introduit avec des calculs minimaux dans la mise en œuvre. Nous avons focalisé cette règle pour réduire considérablement le nombre moyen d'itérations par rapport au critère HDA classique, au désagrément d'une perte de performance négligeable. La nouvelle méthode n'a pas besoin de mémoires supplémentaires pour sauvegarder les calculs entre les itérations, car elle fonctionne avec quelques paramètres et peut arrêter le décodage itératif par pas de demi-itération.

Dans la deuxième technique, un nouveau critère d'arrêt, PCE, est proposé. Il est basé sur l'identification de la convergence des bits lors du décodage itératif via l'entropie croisée, et l'amélioration de l'algorithme LogMAP avec la normalisation adaptative. Dans un premier temps, les bits sont classés en termes de leurs états de convergence après la première itération complète. Ensuite, une correction adaptative partielle et un critère d'arrêt d'entropie croisée partielle sont appliqués aux bits non convergents.

Les simulations montrent que le nombre moyen d'itérations atteint par le critère PCE proposé surpasse les autres critères discutés. Il peut terminer le décodage itératif dans la réception avec un faible bruit, où l'information d'entrée est très fiable, ou même dans la réception contaminée par un bruit élevé, où le décodeur n'a pas la capacité de décoder correctement.

Le turbodécodage utilisant le critère PCE, combiné au protocole HARQ de type-I, permet au système de bénéficier d'une complexité de calcul très réduite et d'améliorer le débit de transmission global qui atteint jusqu'à quatre fois le débit nominal, grâce à cette stratégie efficace de détection et de correction des erreurs.

Les critères proposés ne sont pas limités à l'algorithme LogMAP et donc la perspective inclut la possibilité de généraliser à d'autres algorithmes de décodage itératifs dans ce contexte [75-83]. Il serait intéressant d'exploiter ces critères dans les codes LDPC pour repousser les limites de leurs décodeurs itératifs.

Le débit de transmission réalisé par le critère PCE, peut être augmenté à des débits plus élevés en appliquant, au lieu de l'algorithme LogMAP, des algorithmes plus parallélisés au turbo-décodeur comme l'algorithme FPTD [71] (au détriment de plus de surface de silicium et de consommation d'énergie).

La majorité de ces résultats ont donné lieu à des publications dans des conférences internationales [72,73], une publication dans un article de revue internationale [74].

Publications

[1] I. Amamra, N. Derouiche, "A stopping criteria for turbo decoding based on the LLR histogram," *16th IEEE Mediterranean Electrotechnical Conference*, Yasmine Hammamet, 2012, pp. 699-702.

[2] I. Amamra, N. Derouiche, "New Stopping Criterion of Iterative Turbo Decoding for HARQ Systems," *International Conference on Applied Analysis and Mathematical Modeling (ICAAMM 2015)*, Istanbul, Turkey 2015, pp. 149.

[3] I. Amamra, N. Derouiche, "Enhancement of iterative turbo decoding for HARQ systems," *ICTACT Journal on Communication Technology*, vol. 7, no. 2, pp. 1295-1300, Jun. 2016.

Bibliographie

[1] C. Shannon, "A Mathematical Theory of Communication", *Bell System Technical Journal*, vol. 27, no. 4, pp. 623-656, 1948.

[2] R. Hamming, "Error Detecting and Error Correcting Codes", *Bell System Technical Journal*, vol. 29, no. 2, pp. 147-160, 1950.

[3] P. Elias, "Error-free Coding", *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 29-37, 1954.

[4] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260-269, 1967.

[5] L. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate ", *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284-287, 1974.

[6] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes", *Proceedings of ICC '93 - IEEE International Conference on Communications,* Geneva, 1993, pp. 1064-1070 vol.2.

[7] C. Berrou et A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes", *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261-1271, 1996.

[8] R. G. Gallager, “Low-Density Parity-Check Codes,” M.I.T. Press, Cambridge, Massachusetts, 1963.

[9] D. MacKay, "Good error-correcting codes based on very sparse matrices", *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399-431, 1999.

[10] K. D. Rao, *Channel Coding Techniques for Wireless Communications*. s.l.: Springer-Verlag, 2015.

[11] S. J. Johnson, *Iterative Error Correction: Turbo, Low-density Parity-check and Repeat-accumulate Codes*. Cambridge University Press, 2009.

[12], D. Le Ruyet, et M. Pischella. *Digital Communications 1: Source and Channel Coding*. John Wiley & Sons, 2015.

[13] E. A. Glavieux, *Channel Coding in Communication Networks: From Theory to Turbocodes*. John Wiley & Sons, 2007.

[14] S. Lin et D. J. Costello, *Error control coding: fundamentals and applications*. Upper Saddle River, NJ: Pearson Education, 2004.

[15] J.-G. Dumas, *Théorie des codes: compression, cryptage, correction*. Paris: Dunod, 2007.

[16] R. H. Morelos-Zaragoza, *The Art of error correcting coding*. Chichester: Wiley, 2006.

[17] A. Tixier et J.-P. Tillich, "Reconnaissance de codes correcteurs," Thèse de doctorat: Informatique: Paris 6, 2015.

[18] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, 1996.

[19] J. M. F. Moura, Jin Lu and Haotian Zhang, "Structured low-density parity-check codes," in *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 42-55, Jan. 2004.

[20] E. Biglieri, *Coding for wireless channels*. New York, NY: Springer, 2005.

[21] C. Berrou, *Codes et turbocodes*. New York: Springer-Verlag France, 2007.

[22] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[23] T. Ohtsuki, "LDPC Codes in Communications and Broadcasting," *IEICE Transactions on Communications*, vol. E90-B, no. 3, pp. 440–453, Jan. 2007.

[24] J. L. Fan. "Array codes as low-density parity-check codes," *Proceedings of 2nd Symposium on Turbo Codes*, pp 543–546, Brest, France, Sept. 2000.

[25] Y. Kou, S. Lin and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," in *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711-2736, Nov 2001.

[26] A. Balatsoukas-Stimming "Analysis and Design of LDPC Codes for the Relay Channel" Thèse d'ingéniorat, Technical University of Crete, Department of Electronic and Computer Engineering, 2010.

[27] D. J. Costello and G. D. Forney, "Channel coding: The road to channel capacity," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1150–1177, 2007.

[28] E. Ankan, N. U. Hassan, M. Lentmaier, G. Montorsi, and J. Sayir, "Challenges and some new directions in channel coding," *Journal of Communications and Networks*, vol. 17, no. 4, pp. 328–338, 2015.

[29] Slim Chaoui "Convolutional Coupled Codes" Thèse de doctorat, Université technique de Darmstadt, Département de génie électrique et technologie de l'information, 2003.

[30] R. Johannesson et K. S. Zigangirov, *Fundamentals of convolutional coding*, 2nd ed. Wiley-IEEE Press, 2015.

[31] E. Biglieri, D. Declerq, et M. Fossorier, *Channel coding: theory, algorithms, and applications*, 1st ed. Oxford: Academic Press, 2014.

[32] H. Ma and J. Wolf, “On Tail Biting Convolutional Codes, ” in *IEEE Transactions on Communications*, vol. 34, no. 2, pp. 104-111, Feb 1986.

[33] C. Weiss, C. Bettstetter, S. Riedel et D. J. Costello, "Turbo decoding with tail-biting trellises," *1998 URSI International Symposium on Signals, Systems, and Electronics. Conference Proceedings (Cat. No.98EX167)*, Pisa, Italy, 1998, pp. 343-348.

[34] C. Fragouli, R.D. Wesel, “Convolutional codes and matrix control theory, ” *7th International Conference on Advances in Communications and Control*, Athens, Greece, June 28-July 2, 1999.

[35] L. Hanzo, T. H. Liew, B. L. Yeap, R. Y. S. Tee, S. X. Ng, *Turbo coding, turbo equalisation and space-time coding: exit-chart aided near-capacity designs for wireless channels*, 2nd ed. Wiley-IEEE Press, 2011.

[36] J. Hagenauer and P. Hoeher, “A Viterbi algorithm with soft-decision outputs and its applications,” *Global Telecommunications Conference and Exhibition 'Communications Technology for the 1990s and Beyond' (GLOBECOM), 1989. IEEE*, Dallas, TX, 1989, pp. 1680-1686 vol.3.

[37] P. Robertson, E. Villebrun and P. Hoeher, “A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain,” *IEEE International Conference on Communications, ICC '95 Seattle, 'Gateway to Globalization'*, Seattle, WA, 1995, pp. 1009-1013 vol.2.

[38] S. Benedetto et G. Montorsi, “Unveiling turbo codes: some results on parallel concatenated coding schemes,” in IEEE Transactions on Information Theory, vol. 42, no. 2, pp. 409-428, Mar 1996.

[39] L. C. Perez, J. Seghers et D. J. Costello, “A distance spectrum interpretation of turbo codes,” in IEEE Transactions on Information Theory, vol. 42, no. 6, pp. 1698-1709, Nov. 1996.

[40] H. El Gamal et A. R. Hammons, “Analyzing the turbo decoder using the Gaussian approximation,” in *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 671-686, Feb 2001.

[41] S. T. Brink, “Convergence behavior of iteratively decoded parallel concatenated codes,” *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, 2001.

[42] P. Robertson, P. Hoeher, et E. Villebrun, “Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding,” *European Transactions on Telecommunications*, vol. 8, no. 2, pp. 119–125, 1997.

[43] F. Berens, A. Worm, H. Michel et N. Wehn, “Implementation aspects of turbo-decoders for future radio applications,” *Gateway to 21st Century Communications Village. VTC 1999-Fall. IEEE VTS 50th Vehicular Technology Conference*, Amsterdam, 1999, pp. 2601-2605 vol.5.

[44] J. Kaza and C. Chakrabarti, “Design and implementation of low-energy turbo decoders,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 9, pp. 968–977, 2004.

[45] Y. Lin, S. Mahlke, T. Mudge, C. Chakrabarti, A. Reid, et K. Flautner, “Design and Implementation of Turbo Decoders for Software Defined Radio,” *2006 IEEE Workshop on Signal Processing Systems Design and Implementation, Banff, Alta., 2006, pp. 22-27.*.

[46] Yanhui Tong, T. H. Yeap et J. Y. Chouinard, “VHDL implementation of a turbo decoder with log-MAP-based iterative decoding,” *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 4, pp. 1268-1278, Aug. 2004.

[47] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett and C. Nicol, “A 24Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless,” *2003 IEEE International Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC.*, San Francisco, CA, USA, 2003, pp. 150-484 vol.1.

[48] C.-C. Wong and H.-C. Chang, *Turbo decoder architecture for beyond-4G applications*. New York: Springer, 2014.

[49 67] F. Li et A. Wu, “On the new stopping criteria of iterative turbo decoding by using decoding threshold”, *IEEE Transactions on Signal Processing,* vol. 55, no. 11, pp. 5506-5516, 2007.

[50] R. Y. Shao, Shu Lin et M. P. C. Fossorier, “Two simple stopping criteria for turbo decoding, ” *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1117-1120, Aug 1999.

[51] Zhongfeng Wang, H. Suzuki et K. K. Parhi, “VLSI implementation issues of TURBO decoder design for wireless applications,” *1999 IEEE Workshop on Signal Processing Systems. SiPS 99. Design and Implementation*, Taipei, 1999, pp. 503-512.

[52] H. Suzuki, Z. Wang et K. K. Parhi, “A K=3, 2 Mbps low power turbo decoder for 3rd generation W-CDMA systems,” *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference*, Orlando, FL, 2000, pp. 39-42.

[53] K. R. Narayanan et G. L. Stuber, “A novel ARQ technique using the turbo coding principle,” in *IEEE Communications Letters*, vol. 1, no. 2, pp. 49-51, March 1997.

[54] E.-H. Lu, Y.-N. Lin, et W.-W. Hung, “Improvement of turbo decoding using cross-entropy,” *Computer Communications*, vol. 32, no. 6, pp. 1034–1038, 2009.

[55] J. Wu, Z. Wang et B. R. Vojcic, “Partial iterative decoding for binary turbo codes via cross-entropy based bit selection,” *IEEE Transactions on Communications*, vol. 57, no. 11, pp. 3298-3306, Nov. 2009.

[56] L. Papke, P. Robertson et E. Villebrun, “Improved decoding with the SOVA in a parallel concatenated (Turbo-code) scheme,” *IEEE International Conference on Communications, ICC '96, Conference Record, Converging Technologies for Tomorrow's Applications.* Dallas, TX, 1996, pp. 102-106 vol.1.

[57] B. Shin, S. Kim, H. Park, J. No et D. Shin, “New stopping criteria for iterative decoding of LDPC codes in H-ARQ systems”, *International Journal of Communication Systems*, vol. 26, no. 11, pp. 1475-1484, 2013.

[58] P. Haase et H. Rohling, “Truncated turbo decoding with reduced computational complexity”, *Wireless Personal Communications*, vol. 20, no. 2, pp. 101-112, 2002.

[59] Q. Luo, R. Hoshyar-Jabal-Kandi et P. Sweeney, “Cross-entropy-based method to analyse iterative decoding”, *IET Commun.*, vol. 2, no. 1, p. 113-120, 2008.

[60] Z. Ma, P. Fan, W. Mow et Q. Chen, “A joint early detection-early stopping scheme for short-frame turbo decoding,” *AEU-International Journal of Electronics and Communications*, vol. 65, no. 1, pp. 37-43, 2011.

[61] A. Hunt, S. Crozier, K. Gracie et P. Guinand, “A completely safe early-stopping criterion for max-log Turbo code decoding,” *4th International Symposium on Turbo Codes & Related Topics; 6th International ITG-Conference on Source and Channel Coding*, Munich, Germany, 2006, pp. 1-6.

[62] A. Matache, S. Dolinar et F. Pollara, “Stopping rules for turbo decoders,” *Jet Propulsion Lab*, Pasadena, CA, JPL TMO Progress Rep. 42-142, Aug. 15, 2000.

[63] A. Heim et U. Sorger, “Turbo decoding: Why stopping-criteria do work,” *2008 5th International Symposium on Turbo Codes and Related Topics*, Lausanne, 2008, pp. 255-259.

[64] M. Rovini et A. Martinez, “Efficient stopping rule for turbo decoders,” *Electronics Letters*, vol. 42, no. 4, p. 235, 2006.

[65] L. Huang, Q. T. Zhang et L. L. Cheng, “Information Theoretic Criterion for Stopping Turbo Iteration,” *IEEE Transactions on Signal Processing*, vol. 59, no. 2, pp. 848-853, Feb. 2011.

[66] R. D. Souza, A. G. D. Uchôa, et M. E. Pellenz, “A novel hybrid ARQ scheme using turbo codes and diversity combining,” *AEU - International Journal of Electronics and Communications*, vol. 64, no. 11, pp. 1078–1081, 2010.

[67] Esam A. Obiedat et Lei Cao, "Turbo Decoder for Low-Power Ultrawideband Communication Systems," *International Journal of Digital Multimedia Broadcasting*, vol. 2008, Article ID 897069, 7 pages, 2008.

[68] A. Shibutani, H. Suda et F. Adachi, "Complexity reduction of turbo decoding," *Proceedings of the 50th IEEE Vehicular Technology Conference*, vol. 3, pp. 1570-1574, Amsterdam, The Netherlands, 1999.

[69] K. Gracie, A. Hunt et S. Crozier, "Performance of Turbo Codes using MLSE-Based Early Stopping and Path Ambiguity Checking for Inputs Quantized to 4 Bits," *4th International Symposium on Turbo Codes & Related Topics; 6th International ITG-Conference on Source and Channel Coding*, Munich, Germany, 2006, pp. 1-6.

[70] A. Savin, L. Trifina, et M. Andrei, "Threshold Based Iteration Stopping Criterion for Turbo Codes and for Scheme Combining a Turbo Code and a Golden Space-Time Block Code," *Advances in Electrical and Computer Engineering*, vol. 14, no. 1, pp. 139–142, 2014.

[71] R. G. Maunder, "A Fully-Parallel Turbo Decoding Algorithm," *IEEE Transactions on Communications*, vol. 63, no. 8, pp. 2762-2775, Aug. 2015.

[72] I. Amamra et N. Derouiche, "A stopping criteria for turbo decoding based on the LLR histogram," *16th IEEE Mediterranean Electrotechnical Conference*, Yasmine Hammamet, 2012, pp. 699-702.

[73] I. Amamra et N. Derouiche, "New Stopping Criterion of Iterative Turbo Decoding for HARQ Systems," *International Conference on Applied Analysis and Mathematical Modeling (ICAAMM 2015)*, Istanbul, Turkey 2015, pp. 149.

[74] I. Amamra et N. Derouiche, "Enhancement of iterative turbo decoding for HARQ systems," *ICTACT Journal on Communication Technology*, vol. 7, no. 2, pp. 1295-1300, Jun. 2016.

[75] A. J. Felstrom et K. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2181–2191, 1999.

[76] T. Richardson et R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[77] J. Anderson and S. Hladik, "Tailbiting MAP decoders," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 297–302, 1998.

[78] N. Y. Yu, "Performances of Punctured Tail-Biting Convolutional Codes Using Initial State Estimation," *2008 IEEE 68th Vehicular Technology Conference*, Calgary, BC, 2008, pp. 1-5.

[79] W. Zhang, M. Lentmaier, K. S. Zigangirov et D. J. Costello, “Braided Convolutional Codes: A New Class of Turbo-Like Codes,” *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 316-331, Jan. 2010.

[80] C. Hager, H. D. Pfister, A. G. I. Amat, et F. Brannstrom, “Density Evolution for Deterministic Generalized Product Codes on the Binary Erasure Channel at High Rates,” *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4357–4378, 2017.

[81] D. J. Costello, M. Lentmaier, et D. G. M. Mitchell, “New perspectives on braided convolutional codes,” *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Brest, 2016, pp. 400-405.

[82] S. Moloudi, M. Lentmaier, et A. G. I. Amat, “Spatially coupled turbo codes,” *International Symposium on Turbo Codes & Iterative Information Processing (ISTC), 2014*, Bremen, Germany, pp. 883-887.

[83] M. Lentmaier, I. Andriyanova, N. Hassan, G. Fettweis, “Spatial Coupling - A way to Improve the Performance and Robustness of Iterative Decoding,” *European Conference on Networks and Communications (EuCNC)*, Paris, France, 2015.