Titre: Title:	Terminaison des codes convolutionnels récursifs doublement-orthogonaux
Auteur: Author:	Rafaa Liwa Layouni
Date:	2013
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Layouni, R. L. (2013). Terminaison des codes convolutionnels récursifs doublement-orthogonaux [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1197/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/1197/
Directeurs de recherche: Advisors:	Christian Cardinal, & David Haccoun
Programme: Program:	génie électrique

UNIVERSITÉ DE MONTRÉAL

TERMINAISON DES CODES CONVOLUTIONNELS RÉCURSIFS DOUBLEMENT ORTHOGONAUX

RAFAA LIWA LAYOUNI

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

AOÛT 2013

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

TERMINAISON DES CODES CONVOLUTIONNELS RÉCURSIFS DOUBLEMENT ORTHOGONAUX

présenté par : LAYOUNI Rafaa Liwa

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. NERGUIZIAN Chahé, Ph.D., président

M. CARDINAL Christian, Ph.D., membre et directeur de recherche

M. HACCOUN David, Ph.D., membre et codirecteur de recherche

M. FRIGON Jean-François, Ph.D., membre

À tous mes professeurs…

REMERCIEMENTS

Je tiens ici à exprimer toute ma reconnaissance aux personnes qui m'ont aidé, encouragé et soutenu pendant ces années de préparation à la Maîtrise.

Premièrement, j'aimerais remercier mon directeur de recherche, Monsieur Christian Cardinal, ainsi que mon co-directeur de recherche, Monsieur David Haccoun, pour leur appui et présence durant toutes les étapes de ce mémoire ainsi que leurs conseils et commentaires constructifs sans lesquels ce travail n'aurait pu voir le jour. Je leur suis tout particulièrement reconnaissant pour la confiance qu'ils m'ont accordé et pour les moyens financiers et logistiques qu'ils ont mis à ma disposition.

J'aimerais aussi remercier tous les membres du jury qui me font l'honneur de juger ce travail.

À mes sœurs, à mes frères et à mes parents merci pour vos encouragements.

Je tiens aussi à remercier tous mes collègues du laboratoire pour toutes ces discussions quotidiennes enrichissantes lors de ces deux dernières années. Je pense particulièrement à Xavier, Éric, Xuhua, Wael et Pierre.

Finalement, merci Emna pour ta grande patience et pour tes encouragements pendant les moments les plus difficiles.

RÉSUMÉ

Le travail de recherche présenté dans ce mémoire est un prolongement des travaux précédemment entamés sur les codes convolutionnels récursifs doublement orthogonaux (RCDO). Ces codes correcteurs d'erreur ont pour objectif de corriger les erreurs se produisant lors de la transmission, dans un canal bruité, de l'information entre une source et un destinataire. À ce jour, ces codes RCDO ont été seulement utilisés dans le contexte d'une transmission en continu et sont uniquement appliqués à des trames de longueurs indéfinies. Cependant, il devient indispensable de considérer le problème lié à la terminaison des codes convolutionnels RCDO lorsque nous envisageons de les utiliser au sein des systèmes de communication basés sur une transmission par paquets tels que WiMax, Ethernet, LTE, etc. Le problème de la terminaison est un problème ouvert en général pour tous les codes convolutionnels récursifs et consiste à ajouter une séquence de bits spécifique à la fin de la trame de sorte que le codeur RCDO reconverge vers l'état initial. Typiquement, cela implique que tous les éléments de délai composant les registres à décalage du codeur possèdent la valeur zéro à la fin de la transmission. Bien que la terminaison des codes RCDO entraîne une légère perte du taux de codage, des améliorations de protection contre les erreurs à la fin de la trame sont effectuées.

Les objectifs de ce mémoire sont multiples. Dans un premier temps, ce travail permet d'étudier et de résoudre le problème de terminaison des codeurs RCDO. Nous avons proposé une nouvelle technique de terminaison capable de trouver les bits de terminaison qui permettent de reconduire le codeur RCDO vers l'état initial toute en assurant une complexité raisonnable associée à la génération de la séquence de terminaison. Cette technique nous a permis de réduire le plus possible la longueur de la séquence de terminaison à environ celle de la mémoire du codeur. À partir de la technique de terminaison proposée, il devient possible de définir les conditions de la terminaison qu'on doit imposer aux connexions du codeur dans le but de générer des codes RCDO terminables. En tenant compte des conditions de terminaison ainsi que des conditions de double orthogonalité, nous avons aussi proposé un nouvel algorithme de recherche capable de construire une architecture efficace des codeurs RCDO multi-registres terminables. Ce programme nous a permis de réduire la mémoire effective des codeurs RCDO, ce qui implique la

minimisation de la perte de taux de codage causée par la terminaison ainsi que la diminution de la latence et de la complexité lors du codage et du décodage.

Les résultats de simulations présentés dans ce mémoire sont prometteurs, car la terminasion des codes RCDO a permis une amélioration supplémentaire des performances d'erreurs à la fin de la trame, mais ces avantages sont obtenus au prix d'une légère perte du taux de codage causée par la transmission d'une séquence additionnelle (i.e. les bits de terminasion). Toutefois, les performances d'erreur offertes par les codes RCDO terminés sont bien meilleures que les résultats obtenus par les codes en blocs LDPC. Nous avons aussi comparé les performances des codes RCDO terminables utilisés dans le contexte d'une transmission en continu, c'est-à-dire lorsqu'ils sont appliqués à des trames de longueurs infinies, à celles d'un nombre de codes LDPC en blocs et convolutionnels que l'on retrouve dans la littérature. Les résultats obtenus avec les codes RCDO terminables dépassent largement les performances d'erreur des codes en blocs LDPC, et se comparent favorablement aux codes convolutionnels LDPC.

ABSTRACT

This thesis presents an extension of a previous work started on Recursive Convolutional Doubly-Orthogonal (RCDO) codes. These error correcting codes are designed to detect and/or to correct errors caused by the corruptive channel noise during the transmission of data from a source to a destination. So far, RCDO codes were only decoded in a streaming fashion and were only applied to frames of indefinite lengths. However, it is essential to consider the termination for convolutional RCDO codes when used in packet-based communication systems such as WiMax, Ethernet and LTE. The termination problem remains an open and complex problem for all recursive convolutioanl codes and consists of adding a well-defined sequence of tail bits at the end of the frame so that the RCDO encoder can converge to the initial state, usually the all-zero state, which denotes that all the next information and parity bits are zero. Although the termination of RCDO codes causes a small loss of the coding rate, this can facilitate a good error performance over the last bits of a frame.

This thesis has several objectives. First of all, this work studies and solves the RCDO encoder termination problem. In fact, for convolutional codes, the feedforward encoders can be terminated by simply injecting a sequence of zeros to their inputs. However, this is not a trivial problem for RCDO encoders due to their particular recursive structure. A termination sequence, whose component depends on the encoder state, can be determined by solving a system of linear equations over $GF(2)$. For that purpose, we propose a novel termination algorithm for generating, with a reasonable complexity, the tail bits that takes the encoder back to a known state. Moreover, this algorithm allows to minimize the length of the termination sequence to approximately the code memory. From the proposed termination algorithm, it becomes possible to define a number of additional conditions, called *termination conditions*, which must be imposed on the encoder's connexions in order to make the RCDO encoder terminable. Taking into consideration the terminations conditions, along with the code doubly orthogonal conditions, we also propose a new searching algorithm capable of building an efficient architecture of multi shift registers terminable RCDO encoders. It follows that this program allows to reduce considerably the code memory, which involves minimizing the rate loss caused by termination, the decoding delay and the processor complexity.

Simulation results presented in this thesis are promising, as the terminated RCDO codes can provide an extra performance gain over the last bits of a frame at the cost of a slight rate loss due to the transmission of the additional tail bits. Despite these disadvantages, error performances offered by terminated RCDO are better those obtained by the LDPC bloc codes. Finally, we compared the performances of terminable RCDO codes to LDPC convolutional codes and LDPC bloc codes. Computer simulations showed that, for equal processor complexity and storage requirements, terminable RCDO codes have error performances comparable to those of LDPC convolutional codes. However, for an equal processor complexity, terminable RCDO codes outperformed significantly the LDPC bloc codes.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

CDO	Convolutional Doubly-Orthogonal codes
RCDO	Recursive Convolutional Doubly-Orthogonal codes
LDPC-B	Low-Density Parity-Check Block codes
LDPC-C	Low-Density Parity-Check convolutional codes
LTE	Long-Term Evolution
GF	Galois Field
QoS	Quality of Service
AWGN	Additive White Caussian Noise
BPSK	Binary Phase Shift Keying
SOVA	Soft Output Viterbi Algorithm
BCJR	Algorithme de décodage de Bahl-Cocke-Jelinek-Raviv
CSO	Convolutional Simply-Orthogonal codes
S-CDO	Simplify-Convolutional Doubly-Orthogonal codes
PEG	Algorithme Progressive Edge Growth
BP	AlgorithmeBelief Propagation
LRV	Logarithme du Rapport de Vraisemblance
MAC	Media Access Control
FER	Frame Error Rate
BER	Binary Error Rate
FIFO	First-In First-Out
FSM	Finite State Machine

E_b	Énergie d'un bit (joule)
B	Largeur de bande (Hz)
C	Capacité de Shannon
$\frac{E_b}{N_0}$	Rapport signal sur bruit
$\boldsymbol{v}_{\boldsymbol{i}}^{(k)}$	Symbole binaire à la k −ième sortie d'un codeur RCDO à l'instant i
$r = b/c$	Taux de codage d'un codeur convolutionnel ayant b entrées et c sorties
$\alpha_{i,j}$	Connexion entre le i −ième symbole à la sortie d'un codeur RCDO et le j −ième registre à décalage composant le codeur
m_s	Mémoire du plus grand registre à décalage composant un codeur convolutionnel
$\mathbf{H}^{\mathrm{T}}$	Matrice de contrôle binaire associée à un code LDPC
$\mathbf{H}^{\mathrm{T}}(D)$	Matrice de contrôle de parité associée à un code RCDO
$\mathbf{v}$	Vecteur à composantes binaires représentant un mot de code
$\mathrm{m}_v^{(l)}$	Message à la l −ième itération qui correspond au nœud v
$\mathrm{m}_{vc}^{(l)}$	Message échangé à la l −ième itération entre le nœud v et le nœud c du graphe de Tanner
$\lceil . \rceil$	Fonction plafond

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

Ces dernières années, la demande pour des systèmes de communications numériques performants et rapides a énormément augmentée. La démocratisation des moyens de communication modernes comme l'Internet et les téléphones cellulaires intelligents, et les recherches diversifiées dans les domaines de la théorie de l'information et de la microélectronique ont renforcé cette tendance.

Le but principal d'un système de communication numérique est de transmettre des informations à partir d'une ou plusieurs sources à une ou plusieurs destinations avec une grande fiabilité et une qualité de service (QoS) donnée. Cependant, à cause de la présence du bruit dans les canaux de communication, l'information reçue, en général diffère de celle transmise. La tolérance aux erreurs de transmission à des débits élevés devient donc un critère important pour améliorer la qualité de service. En effet, la qualité d'une transmission numérique dépend principalement de la probabilité d'occurrence d'erreurs dans les symboles transmis. Cette probabilité d'erreur étant fonction du rapport signal à bruit, une des solutions pour améliorer la qualité de transmission est d'augmenter la puissance d'émission. Malheureusement, cette solution est souvent très couteuse en termes d'énergie et de technologie, ce qui rend sa réalisation souvent impossible. Une alternative pour protéger l'intégrité du message transmis, consiste à ajouter aux informations utiles, des informations redondantes selon une règle bien déterminée. La génération de la redondance par le codeur à l'émission permet à l'aide d'un algorithme de décodage à la réception de détecter et/ou corriger un nombre fini d'erreurs introduites par le bruit du canal de transmission. Cette technique est connue sous le nom "codage de canal" dont le père fondateur est Claude Shannon [1] [2].

La théorie de l'information établie à la fin des années 1940, par Shannon a révolutionné le monde des télécommunications en donnant naissance à un théorème fondamental, celui du codage de canal. Ce dernier affirme que, tant que le taux de transmission des symboles d'information est inférieur à une certaine limite appelée capacité du canal, il est théoriquement possible d'obtenir

une probabilité d'erreur arbitrairement faible, à la condition d'utiliser un code correcteur d'erreur approprié [1] [2]. Malgré les résultats présentés par Shannon, aucune indication n'est fournie sur la technique du codage qui permet d'atteindre cette limite promise par la théorie. Une vraie course s'est alors engagée dans le but de trouver des codes correcteurs d'erreurs plus ou moins complexes permettant de s'approcher de façon pratique de la limite donnée par la capacité du canal.

Ce n'est qu'en 1993 que les chercheurs Berrou, Glavieux et Thitimajshima inventèrent un système de codage et de décodage appelé codage Turbo [3] qui, pour la première fois, permettait d'atteindre à quelques fractions de dB près la limite théorique de transmission prédite par Shannon [4]. Cette révolution a ouvert de nombreuses nouvelles pistes de recherche dans le domaine du codage correcteur d'erreurs et plus généralement dans les systèmes de communications numériques. Cette avancée a eu pour conséquence la redécouverte des codes en blocs LDPC (Low-Density Parity-check) initialement introduits par Gallager [5] en 1962, mais qui sont restés dans l'oubli pendant presque 35 ans à cause de l'état des technologies de l'époque qui ne permettait pas de traiter d'une manière pratique les algorithmes décrits. En 1996, encouragé par le contexte qui a suivi la découverte du principe turbo, Mackay *et al.* [6] redécouvrent les codes LDPC de Gallager. Grâce aux décodages itératifs des codes Turbo et des codes LDPC, ces deux techniques ont permis d'offrir d'excellentes performances d'erreurs. Cependant, ces résultats ont été obtenus aux prix d'une énorme complexité matérielle engendrée par ces types de codeurs et décodeurs et d'un important délai associé au décodage. Dans le but de contourner ces inconvénients, les professeurs Gagnon, Haccoun, Cardinal et Batani ont proposé en 1997 [7] [8] [41], une procédure efficace et pratique de codage et de décodage basée sur des propriétés algébriques de codes convolutionnels. Cette invention, a donnée naissance aux codes convolutionnels doublement orthogonaux (CDO) qui peuvent notamment être décodés par un décodeur à seuil itératif. Cette nouvelle classe de codes offre un compromis intéressant, entre la complexité matérielle et la performance de correction d'erreurs, à celui proposé par les codes Turbo et LDPC. Dans ce mémoire nous nous intéressons en particulier aux codes convolutionnels récursifs doublement orthogonaux (RCDO) [9]. En effet, une étude récente montre que ces codes RCDO sont les plus attrayants puisqu'ils procurent des performances d'erreur s'approchant à quelques dixièmes de décibel la limite de Shannon [10].

1.1 Éléments de problématique du codage correcteur d'erreur

De nos jours, un bon nombre de systèmes de communication filaire et sans fil utilise des protocoles de communications basés sur le format d'une trame Ethernet. L'application des codes en blocs, tels que les codes LDPC-B (Low Density Parity Check-Bloc Codes), à de tels systèmes pose plusieurs défis. Par exemple dans les normes comme WiMax ou LTE, comment pouvons-nous tenir compte d'une trame de taille aléatoire dans un bloc fixe du code LDPC-B? Que pouvons-nous faire si la trame Ethernet à transmettre est légèrement plus grande qu'un multiple de la taille du bloc? Voici quelques questions qui ont été considérées dans notre travail.

Pour répondre à nos questions de recherches, nous pouvons rajouter des bits de bourrage à la fin de la trame jusqu'à atteindre la taille du bloc du code utilisé. Cette technique présente néanmoins un inconvénient majeur, qui tient à la perte du taux effectif de transmission. Dans ce mémoire, nous avons considéré les codes convolutionnels puisqu'ils peuvent coder et décoder de manière continue des trames de longueurs arbitraires. Ceci rend les codes convolutionnels RCDO des candidats appropriés pour les applications de streaming et les systèmes de communication basés sur une transmission par paquets de longueurs variables et finies. Cependant, cela apporte une exigence supplémentaire lorsque les codes convolutionnels sont appliqués à des trames de longueurs spécifiques, car le codeur doit être ramené à un état connu (par exemple l'état zéro) pour traiter de façon fiable, au décodage, la queue des trames transmises. Cette technique est connue sous le nom "terminaison". Pour les codes convolutionnels non récursifs, la terminaison consiste en une séquence de zéros permettant de converger à l'état zéro. En revanche, le code récursif convolutionnel ne peut pas être terminé d'une manière aussi simple à cause de la récursivité et de la structure multi-registres du codeur. Dans ce cas, la séquence de terminaison, dont les composants dépendent de l'état de l'encodeur, s'obtient par la résolution d'un système d'équation linéaire dans $GF(2)$.

La problématique de notre travail de recherche demeure dans la recherche des codes convolutionnels récursifs multi-registres doublement orthogonaux RCDO qui soient terminables, dont les encodeurs peuvent convergés vers un état connu. Les codes obtenus jusqu'à présent offrent de bonnes performances d'erreur, mais sont non terminables, ce qui constitue un obstacle majeur à leur utilisation dans des systèmes de communication basés sur une transmission par paquets. C'est ce problème a motivé la recherche présentée dans ce mémoire.

1.2 Objectifs de recherche

Ce mémoire de recherche consiste tout particulièrement à trouver la séquence de bits adéquate qui permet à l'encodeur de reconverger vers un état connu par le décodeur. Ainsi, les codes RCDO terminés pourraient être de bons candidats pour être appliqués dans tous systèmes de communications basés sur une transmission par paquets comme dans les normes Ethernet, WiMax et LTE.

Ce projet de recherche a pour objectif d'analyser et de déterminer l'ensemble des conditions que les codes RCDO doivent respecter pour pouvoir être terminés par une séquence de bits de longueur avoisinant à celle de la mémoire (span) du code. Une fois ces conditions établies, nous construirons des codes RCDO terminables tout en réduisant le plus possible la mémoire du code. Ensuite, nous introduisons une méthode algébrique qui permet de trouver la séquence de terminaison afin de ramener l'encodeur à un état connu par le décodeur.

1.3 Contributions

Les contributions apportées par ce travail de recherche sont les suivantes :

1. Établissement d'un lien entre la forme polynômiale et la forme binaire des matrices de contrôle de parité définissant les codes RCDO.
2. Résolution mathématique du problème de la terminaison des codeurs multi-registres RCDO.
3. Conception de programmes permettant la génération de la séquence de terminaison.
4. Analyse de la complexité liée à la génération des bits de la terminaison.
5. Élaboration de la définition des conditions de la terminaison permettant de construire des codes RCDO terminables.
6. Modification du programme existant pour la recherche des codes RCDO terminables réguliers et irréguliers.
7. Comparaison des codes RCDO terminables aux codes LDPC en blocs et aux codes LDPC convolutionnels.

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

1.4 Organisation du mémoire

Ce mémoire comporte six chapitres. Après ce chapitre d'introduction, le Chapitre 2 présente une revue bibliographique dans laquelle nous exposons les différentes parties constituant la chaîne de transmission numérique. Nous nous intéressons aussi à un certain types codes correcteurs d'erreurs modernes décodés par un algorithme de décodage itératif à passage de message dont l'architecture du décodeur est brièvement présentée. Au Chapitre 3, nous analysons les codes convolutionnels récursifs multi-registres doublement orthogonaux RCDO, et nous établissons le lien entre la forme polynômiale et la forme binaire des matrices de contrôle de parité définissant le code. Au Chapitre 4, la problématique de terminaison des codeurs convolutionnels a été discutée en détail, et le problème de la terminaison des codes RCDO a été résolu algébriquement. Ensuite, nous définissons les conditions de terminaison et proposons un nouvel algorithme permettant de construire des codeurs RCDO terminables. Au Chapitre 5, nous étudions les performances des codes RCDO terminés lors d'une transmission par paquets. De plus, nous comparons les performances d'erreur des codes RCDO terminables lors d'une transmission en continu aux codes LDPC en blocs et aux codes LDPC convolutionnels. Finalement, le dernier chapitre résume l'ensemble des travaux effectués et propose certaines ouvertures pour lesquelles les codes RCDO terminés pourraient être utilisés.

CHAPITRE 2

REVUE BIBLIOGRAPHIQUE

Le principe de base d'un système de communication numérique est de transmettre un message à partir d'un émetteur et de pouvoir le récupérer correctement au récepteur. À cause des bruits et interférences provenant du canal, les systèmes de communication actuels utilisent des techniques de codage de canal protégeant la fiabilité du message transmis. Il existe deux grandes techniques de codage correcteur d'erreur : le codage en bloc et le codage convolutionnel.

Dans un premier temps, ce chapitre présente les différentes parties constituant une chaîne de transmission ainsi que quelques notions et concepts reliés au codage de canal dans un système de communication numérique. Après cette présentation, nous nous intéresserons aux codes correcteurs d'erreurs utilisés conjointement avec des décodeurs itératifs tels que, les codes LDPC [5], les codes convolutionnels orthogonaux [20] [41] et les codes Turbo [3] [42]. Ces derniers sont décodés par un algorithme itératif à passage de messages dont l'architecture du décodeur sera brièvement abordée à la fin de ce chapitre.

2.1 Système de communication numérique

Une chaîne de transmission modélise les différentes étapes permettant le transfert d'information à partir d'une source vers un destinataire. À la Figure 2.1, nous présentons le schéma en bloc utilisé pour modéliser le système de communication numérique que nous avons considéré dans ce mémoire.

La première partie d'un système de communication numérique consiste en une opération nommée codage de source. Ce codage sert principalement à éliminer la redondance présente dans la séquence issue d'une source. Cette compression permet d'avoir moins de données à transmettre, d'où un gain de temps. Cet aspect ne sera cependant pas traité dans ce mémoire. Ensuite, la séquence binaire qui représente l'information compressée, est encodée par le codeur de canal, qui la transforme en une autre séquence binaire. Cette opération, appelée aussi codage détecteur/correcteur d'erreur, a pour rôle d'ajouter de la redondance contrôlée dans le message à

transmettre pour permettre à la réception de détecter et éventuellement corriger les erreurs introduites lors de la transmission. C'est ce type de codage qui va nous intéresser en particulier dans la suite de ce mémoire.

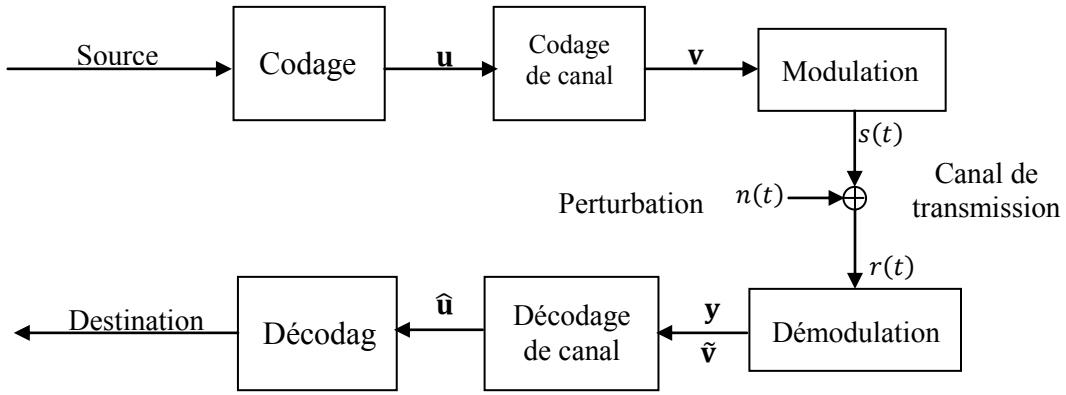


FIGURE 2-1: Schéma bloc représentant un système de communication numérique.

Soit $\mathbf{u} = (u_0, u_1, \dots, u_i, \dots)$ la séquence de symboles binaires issue du codeur de source, où chaque bit d'information $u_i \in \{0, 1\}$ est d'énergie E_b, et $i = 0, 1, 2, \dots$ est l'instant de transmission de chaque bit. Nous supposons que tous les bits sont équiprobables tels que $P(u_i = 0) = P(u_i = 1) = 1/2$ .Cette séquence est ensuite transmise vers un codeur de canal de taux de codage r. Ce codeur ajoute, selon un certain nombre de règles, un ou plusieurs bits de redondance pour chaque bit d'information pour former ainsi la séquence binaire codée $\mathbf{v} = (v_0, v_1, \dots, v_i, \dots)$. Cette séquence est ensuite transmise vers un modulateur qui sert à transformer les symboles codés en signaux analogiques exprimés en fonction du temps $s(t)$ avant de les transmettre dans le canal. Dans ce mémoire, nous considérons uniquement un modulateur BPSK qui n'est rien d'autre qu'une modulation antipodale, faisant correspondre les bits de valeur '0' à un niveau de tension négative et les bits de valeur '1' à un niveau de tension positive. Les symboles $s_i(t)$ à la sortie du modulateur BPSK sont calculés par l'équation suivante $s_i(t) = x_i f(t) = (1 - 2v_i) f(t)$ où $f(t)$ représente une fonction d'énergie $E_s = r \cdot E_b$ qui est définie sur le support $0 \leq t \leq T$. Ces symboles sont ensuite transmis vers le canal. Le canal considéré est binaire symétrique sans mémoire à bruit blanc additif gaussien (AWGN). Par conséquent, $n_i(t)$ est un processus aléatoire gaussien de moyenne nulle et de densité de puissance spectrale

unilatérale N_0 Watts/Hz. Le signal reçu en sortie du canal $r_i(t) = s_i(t) + n_i(t)$ est démodulé par un filtre adapté normalisé en produisant en sortie une valeur réelle non quantifiée y_i exprimée par :

$$y_i = \sqrt{r\,E_b}x_i + n_i \tag{2.1}$$

où $x_i = (1 - 2v_i)$ est le symbole antipodale émis et n_i est une variable aléatoire gaussienne centrée de variance $\sigma^2 = N_0/2$. Dans le cas d'une quantification ferme, le symbole antipodale à la sortie du démodulateur à l'instant i peut être défini comme suit :

$$\tilde{v}_i = v_i \oplus e_i \tag{2.2}$$

où e_i représente l'erreur qui affecte le symbole codé v_i et où $\oplus$ représente l'opérateur logique OU exclusif (XOR). Sur la Figure 2.2, la probabilité que le récepteur reçoive un symbole à 1 sachant qu'un symbole 0 à été transmis ($e_i = 1$) est égale à la probabilité de transition p. Lorsqu'une modulation BPSK est utilisée pour transmettre l'information dans un canal AWGN, la probabilité de transition p est égale à :

$$p = Q\left(\sqrt{\frac{2\,r\,E_b}{N_0}}\right) \tag{2.3}$$

Une fois la démodulation achevée, les sorties non quantifiées $\mathbf{y} = (y_0, y_1, \ldots, y_i, \ldots)$ ou fermes $\tilde{\mathbf{v}} = (\tilde{v}_0, \tilde{v}_1, \ldots, \tilde{v}_i, \ldots)$ seront fournies au décodeur de canal afin de retrouver la séquence initiale $\mathbf{u}$. En effet, il utilisera les symboles de redondances, appelés aussi les symboles de parités, générés par le codeur de canal pour fournir en sortie une séquence décodée $\hat{\mathbf{u}} = (\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_i, \ldots)$ qui correspond à la décision final faite par le décodeur.

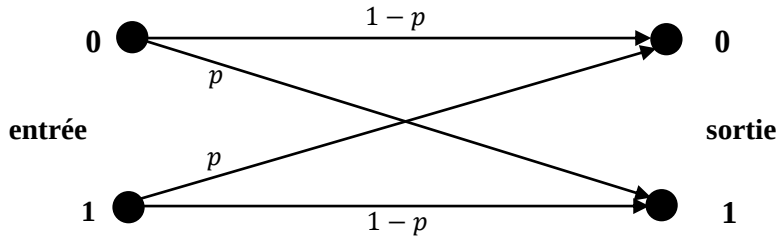


FIGURE 2-2 : Canal binaire symétrique sans mémoire.

L'une des bornes établie par Shannon est celle liée à la capacité théorique de transmission dans un canal AWGN de moyenne nulle et de densité spectrale unilatérale égale à N_0. Shannon a montré que pour ce type de canal la capacité théorique bits/s est égale à [1] :

$$C = B \log_2(1 + SNR) \text{ bits/s} \tag{2.4}$$

avec le rapport signal à bruit :

$$SNR = \frac{D_b \cdot E_b}{B \cdot N_0} \tag{2.5}$$

où B (Hz) représente la largeur de bande du canal de transmission, et D_b (bits/s) désigne le débit binaire. La relation (2.4) indique qu'il est possible théoriquement, de transmettre sans erreur l'information à un taux de transmission inférieur ou égal à C en utilisant un code correcteur d'erreur approprié. Par contre, si le débit de transmission est supérieur à la capacité du canal alors, il n'existe aucune technique de codage correcteur d'erreurs qui assure une transmission fiable avec une probabilité d'erreur arbitrairement faible. À partir des relations (2.4) et (2.5), nous pouvons en déduire le rapport signal à bruit minimal $(E_b/N_0)_{min}$, à partir duquel, il est théoriquement possible de transmettre sans erreur. Le Tableau 2.1 fournit la valeur du rapport (E_b/N_0) minimum lorsqu'on envisage l'utilisation des codes correcteurs d'erreurs de taux de codage $r = 0.1, 0.25, 0.33, 0.5, 0.66, 0.75$ et 1 pour un canal AWGN à sortie non quantifiée et une modulation BPSK [9]. À titre d'exemple, il est théoriquement possible de construire un code correcteur d'erreur de taux de codage $r = 1/2$ capable de transmettre l'information d'une manière fiable pour des valeurs de (E_b/N_0) supérieures à 0.18 dB. En se référant au Tableau 2.1, il n'existe aucune technique de codage capable de faire tendre la probabilité d'erreur vers zéro pour un canal très bruité où le rapport signal sur bruit (E_b/N_0) est inférieur à -1.6 dB. Les limites de Shannon présentées au Tableau 2.1 sont généralement utilisées pour comparer les systèmes de codage entre eux.

TABLEAU 2.1: Limite des valeurs de E_b/N_0 pour plusieurs taux de codage r lors d'une transmission dans un canal AWGN avec une modulation BPSK.

Taux de codage r	$r \to 0$	0.1	0.25	0.33	0.5	0.66	0.75	1
$(E_b/N_0)_{min}$ en dB	-1.6	-1.28	-0.79	-0.5	0.18	1.02	1.62	7.86

2.2 Définition et paramétrisation des codes correcteurs d'erreurs

2.2.1 Codes en blocs

Le codage en blocs consiste à segmenter la trame d'information en plusieurs blocs de taille fixe k, pour ensuite transformer chacun des blocs d'informations $\mathbf{u} = (u_1, u_2, \dots, u_k)$ en un mot de code $\mathbf{v} = (v_1, v_2, \dots, v_n)$ de taille n, $n > k$, en utilisant une loi linéaire. Le taux de codage ainsi obtenu à la sortie du codeur est donné par :

$$r = k/n \text{ (bits/symbole)} \tag{2.6}$$

où k représente le nombre de bits acceptés en entrée du codeur et n désigne le nombre de bits générés en sortie du codeur.

Un code en blocs linéaire $C(n, k)$, $n > k$, est dit systématique si le mot de code généré à la sortie de l'encodeur se compose, comme illustré dans la Figure 2.3, de :

- k symboles de la séquence d'information sont transmis tels quels,
- $(n - k)$ symboles calculés à partir d'une combinaison linéaire d'un nombre prédéterminé des symboles d'information. Il s'agit des symboles de parité ou de redondance.

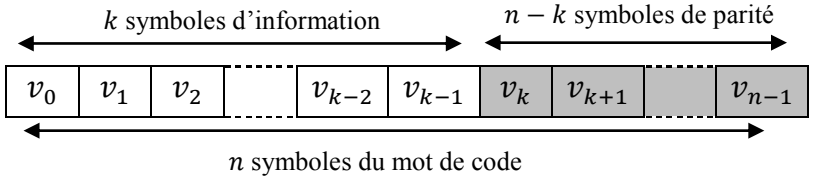


FIGURE 2-3: Représentation d'un mot de code pour un code systématique.

Un code en bloc $C(n, k)$, $n > k$, peut être défini au moyen d'une matrice binaire $\boldsymbol{G}$ à k lignes et n colonnes appelée *matrice génératrice*. Cette matrice est construite de façon à ce que toute combinaison des mots de code donne un autre mot de code, et par conséquent le code est linéaire. Ainsi, à partir de $\boldsymbol{G}$, l'opération générale d'encodage peut être représentée sous forme matricielle de la façon suivante :

$$\mathbf{v} = \mathbf{u} \cdot \boldsymbol{G} \tag{2.7}$$

où $\mathbf{u}$ est le message d'information de longueur k et $\mathbf{v}$ est le mot de code de longueur n généré. Il existe une matrice de parité $\boldsymbol{H}$ de dimensions $(n-k) \times n$ qui est associée au code telle que :

$$\boldsymbol{G} \cdot \boldsymbol{H}^{\boldsymbol{T}} = \mathbf{0} \tag{2.8}$$

La matrice $\boldsymbol{H}$ sert au niveau du décodage, à détecter et corriger les erreurs de transmission en vérifiant si le mot de code reçu, $\mathbf{v}$, est valide. Pour cela, il faut satisfaire :

$$S(\mathbf{v}) = \mathbf{v} \cdot \boldsymbol{H}^{\boldsymbol{T}} = \mathbf{u} \cdot \boldsymbol{G} \cdot \boldsymbol{H}^{\boldsymbol{T}} = \mathbf{0} \tag{2.9}$$

où le vecteur $S(\mathbf{v})$ est appelé syndrôme de $\mathbf{v}$. Un syndrôme nul implique que la séquence reçue est un mot de code valide mais ne peut garantir qu'il s'agisse du mot de code émis. Par ailleurs, un mot de code peut très bien se transformer à un autre en fonction du motif d'erreurs. Par exemple, si $\tilde{\mathbf{v}}$ est la séquence reçue, $\mathbf{e}$ l'erreur qui affecte le mot de code $\mathbf{v}$ émis, alors $\tilde{\mathbf{v}} = \mathbf{v} \oplus \mathbf{e}$ et de là $S(\tilde{\mathbf{v}}) = \mathbf{e} \cdot \boldsymbol{H}^{\boldsymbol{T}}$. Ceci implique que le syndrôme ne dépend que de la séquence d'erreur du canal.

2.2.2 Codes convolutionnels

Une deuxième technique de correction d'erreurs intitulée le codage convolutionnel a été inventée en 1955 au MIT (Massachussetts Institute of Technology) par Elias [13]. Cette technique constitue une classe extrêmement flexible et efficace de codage correcteur d'erreurs. Les codes convolutionnels demeurent les plus utilisés dans les systèmes de communications fixes et mobiles. Les encodeurs convolutionnels encodent les bits d'information de façon continue et génèrent en sortie des séquences de symboles codés. Les codes convolutionnels se distinguent des codes en blocs, car chaque symbole de c éléments en sortie de l'encodeur dépend non seulement des b bits d'information à l'entrée, mais aussi d'un certain nombre de bits d'information précédents.

À la Figure 2.4, la structure du codeur convolutionnel systématique multi-registres est constituée de $(c-b)$ registres à décalage auxquels b bits d'information sont connectés. De plus, nous définissons deux paramètres importants du code convolutionnel employé, *la mémoire du code* dénoté m_s qui correspond au nombre le plus grand de cellules de tous les registres à décalage et de *l'ensemble des connexions A* reliant les symboles d'entrée du codeur aux éléments de délais des registres à décalage.

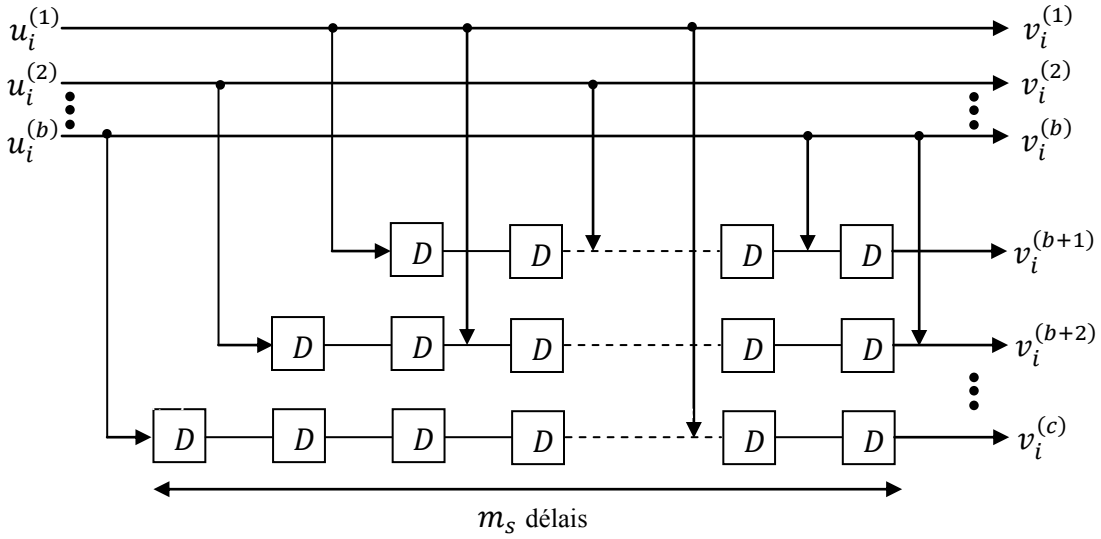


FIGURE 2-4 : Structure d'un codeur convolutionnel systématique ayant un taux de codage $r = b/c$.

Deux principales caractéristiques doivent être prises en compte pour pouvoir générer de bons codes convolutionnels : maximiser la distance minimale du code et décoder avec une complexité raisonnable. Dans la section suivante, nous présentons brièvement les codes Turbo.

2.3 Codes Turbo

Dans cette section, nous présentons les concepts fondamentaux du codage et décodage itératif des codes turbo. Ces derniers sont générés à l'aide d'un ensemble de codeurs convolutionnels séparés par des blocs d'entrelacements. Cette structure a été originalement introduite en 1993 par C. Berrou, A. Glavieux, et P. Thitimasjshima [3]. Le premier code Turbo, illustré dans la Figure 2.5, comporte deux codeurs convolutionnels montés en parallèle et séparés par un entrelaceur aléatoire. Les symboles de sortie du codeur sont composés de bits d'informations, de symboles codés issus du premier codeur convolutionnel et de symboles codés à partir des bits d'information permutés par l'entrelaceur puis générés par le deuxième codeur convolutionnel. La permutation des bits d'information effectuée par l'entrelaceur joue un rôle majeur, puisqu'elle permet d'augmenter la distance libre du code [14].

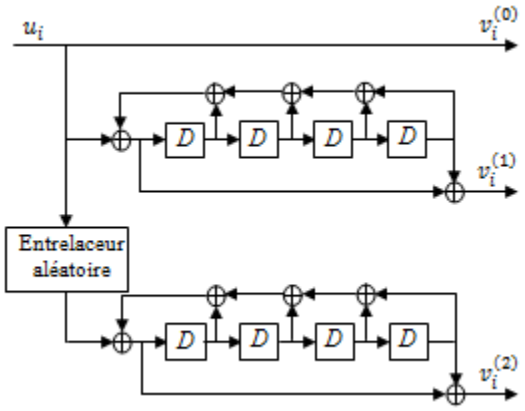


FIGURE 2-5 : Schéma de codeur Turbo ayant un taux de codage $r = 1/3$ [3].

À la Figure 2.6, nous illustrons un schéma bloc d'un décodeur itératif Turbo qui se compose de deux décodeurs, des entrelaceurs identiques à celui utilisé par le codeur et un délaceur qui effectue l'opération inverse de l'entrelaceur. Le principe de décodage itératif repose sur le fait de répéter plusieurs fois l'échange d'information dite extrinsèque entre les deux décodeurs. Parmi les algorithmes utilisés par le décodage Turbo, citons SOVA [16] et BCJR [17], où ce dernier effectue une estimation qui maximise la probabilité *a posteriori* (MAP) des symboles d'information. Dans la première itération, le premier décodeur calcule une valeur extrinsèque à partir des symboles reçus du canal associés aux sorties du premier codeur. Le deuxième décodeur utilisera cette valeur après avoir été entrelacée ainsi que les sorties pondérées du canal associées aux sorties du deuxième codeur afin de calculer une deuxième valeur extrinsèque qui sera retournée au premier décodeur par l'intermédiaire du délaceur. Ce cycle représente une itération et peut être répété autant de fois que le nombre d'itérations maximal effectuées par le décodeur. L'intérêt de la présence de l'entrelaceur dans ce processus est d'assurer l'indépendance entre les informations extrinsèques échangées d'une itération à une autre, ce qui améliorera l'estimation sur les symboles d'information. Le succès des codes Turbo dépend en grande partie du type et de la taille de l'entrelaceur qui est utilisé [9]. Généralement, plus la taille de l'entrelaceur est grande (plusieurs milliers de bits), plus les performances d'erreurs augmentent [15]. En contrepartie, une taille élevée de l'entrelaceur génère un délai important lors de l'encodage et du décodage de la séquence d'information. Or, ceci peut représenter un inconvénient pour certains systèmes de communications sensibles au délai.

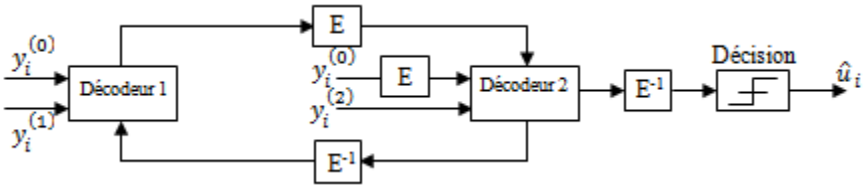


FIGURE 2-6 : Schéma bloc du décodeur itératif Turbo ayant un taux de codage $r = 1/3$.

À la suite de la découverte des codes Turbo, de nombreuses voies de recherche dans le domaine du codage canal ont été ouvertes et plusieurs travaux ont été redécouverts. Par exemple, les codes LDPC en blocs initialement proposés par Gallager [5] ont été redécouverts par Luby *et al.* en 1997 [18]. Ce bouleversement apporté par le décodage itératif turbo a également permis de présenter une nouvelle technique de décodage inspirée du concept d'orthogonalité des codes convolutionnels [41]. Cette nouvelle structure est proposée dans le but de réduire la complexité de décodage Turbo et de diminuer la latence lors de l'encodage et de décodage causée par l'entrelaceur [27] [19].

2.4 Codes convolutionnels orthogonaux

2.4.1 Codes convolutionnels simplement orthogonaux

Dans cette partie, nous introduisons les codes convolutionnels simplement orthogonaux (CSO) proposés par Massey [20]. La Figure 2.7 présente un codeur convolutionnel systématique de taux de codage $r = 1/2$. Un tel codeur peut être décrit en utilisant l'ensemble des positions de connexions $A = \{\alpha_1, \dots, \alpha_{|A|}\}$ où $|A|$ représente le nombre de ces connexions et la valeur α_k représente la k—ième connexion entre le symbole d'entrée du codeur et le α_k—ième élément de délai du registre à décalage, $\alpha_k \in \mathbb{N}$, $k = \{1, \dots, |A|\}$ et $\alpha_{|A|} = m_s$. Avec cet ensemble, nous pouvons représenter les symboles de parité $v_i^{(2)}$ à la sortie du codeur convolutionnel aux instants $i = 0,1, \dots$ en fonction des éléments α_k, $k = 1,2, \dots, |A|$, selon la relation (2.10) :

$$v_i^{(2)} = \sum_k^{|A|} v_{i-\alpha_k}^{(1)} \tag{2.10}$$

où $v_i^{(1)}$représente le bit d'information à l'instant i et la somme dans l'équation (2.10) représente une somme modulo-2. Massey a montré que les codes convolutionnels systématiques peuvent utiliser un simple décodeur à seuil si l'ensemble A du code respecte la définition suivante.

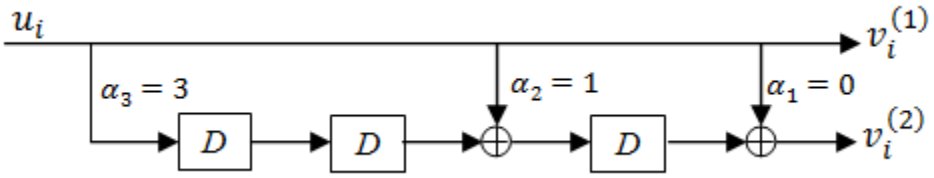


FIGURE 2-7 : Codeur convolutionnel systématique de taux de codage $r = 1/2$, $m_s = 3$.

> **Définition 2.1** : *Un code convolutionnel systématique est simplement orthogonal (CSO) si la position des connexions du codeur* $A = \{\alpha_1, \dots, \alpha_{|A|}\}$ *est telle que :*
> *Les différences* $(\alpha_j - \alpha_k)$ *sont distinctes,* $j \neq k$, $\alpha_j, \alpha_k \in \mathbb{N}$, j *et* $k \in \{1,2, \dots, |A|\}$.

Nous vérifions que le code $A = \{0,1,3\}$ de la Figure 2.7 est bien un code convolutionnel simplement orthogonal car les différences $(\alpha_j - \alpha_k)$ sont distinctes, tel que montré au Tableau 2.2. Nous mentionnons que le nombre de comparaisons à effectuer pour vérifier la condition de simple orthogonalité est égal à $|A|(|A| - 1)/2$.

TABLEAU 2.2: Vérification de l'orthogonalité du codeur de la Figure 2.7.

j	k	$(\alpha_j - \alpha_k)$
3	1	3
3	2	2
2	1	1

La définition de simple d'orthogonalité permet, en effet, d'effectuer un décodage de symbole à l'aide d'un ensemble d'équations de syndrômes indépendantes les unes des autres. L'algorithme de décodage proposé par Massey est très simple à réaliser à partir d'un simple décodeur à seuil. Ce type de décodeur se distingue des autres algorithmes utilisés pour décoder les codes convolutionnels, car il calcule ses décisions en fonction des considérations algébriques et non pas probabilistes comme l'algorithme de décodage de Viterbi [9].

2.4.2 Codes convolutionnels Doublement orthogonaux

Dans cette section, nous introduisons les codes doublement orthogonaux (CDO) qui ont été proposés en 1998 par C. Cardinal, D. Haccoun, F. Gagnon et N. Batani dans [7] et [41] afin de contourner les inconvénients associés aux codes Turbo, comme la latence causée par les entrelaceurs et la complexité de décodage [3]. Le décodeur itératif des codes Turbo est remplacé par un décodeur à seuil itératif plus simple. Cependant, un nombre de conditions de double orthogonalité sur les connexions au niveau des registres à décalage doit être satisfait dans le but de garder l'indépendance des observables qu'assurait la présence des entrelaceurs. Les codes CDO sont définis comme suit [19].

> **Définition 2.2 :** *Un code convolutionnel systématique de taux de codage* $r = 1/2$ *est dit être doublement orthogonal (CDO) si l'ensemble des connexions* $A = \{\alpha_1, \dots, \alpha_{|A|}\}$*, satisfait aux conditions suivantes :*
>
> 1. *Les différences* $(\alpha_j - \alpha_k)$ *sont distinctes (CSO),*
> 2. *Les différences des différences* $(\alpha_j - \alpha_k) - (\alpha_m - \alpha_l)$ *sont distinctes,*
> 3. *L'ensemble des différences de différences et l'ensemble des différences simples sont disjoints,*
>
> *avec les indices* $(j, k, l, m) \epsilon \{1,2, \dots, |A|\}$ *tels que* $j \neq k, l \neq m, l \neq k, m \neq j$.

Nous notons, dans la définition 2.2, la présence de certaines différences de différences qui peuvent être identiques. En effet, certaines permutations des indices (k, m) et (j, l) produisent des différences de différences égales. Dans ce même contexte, les auteurs de [21] et [22] ont démontré que la deuxième condition de la définition 2.2 peut être simplifiée sans trop dégrader les performances de correction d'erreur. Pour cela, ils ont défini les codes convolutionnels doublement orthogonaux simplifiés (S-CDO) permettant à un certain nombre de différences de différences d'être identiques. C'est la raison pour laquelle ces codes sont dits *simplifiés*, puisque l'indépendance des observables n'est pas complète.

Pour remédier à ce problème, de nouveaux codes CDO multi-registres à décalage ont été introduits pour garantir une indépendance totale des observables jusqu'à la deuxième itération de l'algorithme de décodage. Un code CDO de taux de codage $r = b/c$, composé par $(c - b)$ registres à décalage en parallèle, peut être définit par la matrice des connexions $\boldsymbol{A} = [\alpha_{i,j}]$ de

dimension b x $(c-b)$, où l'élément $\alpha_{i,j}$ représente la connexion entre le i—ième symbole d'entrée du codeur et le $\alpha_{i,j}$—ième élément de délai du j—ième registre à décalage et $\max_{i,j}(\alpha_{i,j}) = m_s$. Nous pouvons ainsi définir les codes CDO de taux de codage $r = b/c$ comme suit :

> **Définition 2.3** : *Un code convolutionnel systématique de taux de codage $r = b/c$ et représenté par une matrice de connexion de dimension b x $(c-b)$ dont les éléments sont notés $\{\alpha_{i,j}\}$ est doublement orthogonal s'il vérifie les conditions suivantes :*
>
> 1. *Les différences $(\alpha_{j,n} - \alpha_{k,n})$ sont distinctes,*
> 2. *Les différences de différences $(\alpha_{j,n} - \alpha_{k,n}) - (\alpha_{l,s} - \alpha_{k,s})$ sont distinctes,*
> 3. *L'ensemble des différences de différences et l'ensemble des différences simples sont disjoints,*
>
> *avec les indices $(j,k,l) \epsilon \{1,2,\dots,b\}$, $(n,s) \epsilon \{1,2,\dots,(c-b)\}$ tels que $j \neq k, l \neq k, s \neq n$.*

Notons que chaque symbole d'information à l'entrée du codeur n'est pas injecté plus d'une fois dans le même registre, permettant ainsi d'obtenir une indépendance totale des observables sur deux itérations. Cependant, les codes CDO multi-registres offrent des performances de corrections d'erreurs comparables à celles obtenues avec les codes S-CDO [23]. Pour remédier à ces faibles performances, l'auteur de [19] a ajouté une boucle de retour au sein du codeur CDO multi-registres. Les résultats de simulation de cette classe de code sont extrêmement intéressants puisqu'ils montrent que les codes convolutionnels récursifs multi-registres doublement orthogonaux (RCDO) sont les codes CDO les plus performants à hauts et faibles rapports signal sur bruit. Cette classe prometteuse des codes RCDO sera étudiée en détail au chapitre 3. Une présentation d'un codeur convolutionnel récursif de taux de codage $r = b/c$ est illustrée à la Figure 2.8. En effet, les bits d'information sont injectés dans le codeur par blocs de b bits, et les $(c-b)$ symboles de parité sont calculés, à l'instant $i = 0,1,2,\dots$, en utilisant non seulement les bits d'information précédant les bits d'informations courants $u_i^{(k)}$avec $k \in \{1,\dots,b\}$ mais aussi les symboles de parité précédant les symboles de parité courants $v_i^{(k)}$ avec $k \in \{b+1,\dots,c\}$.

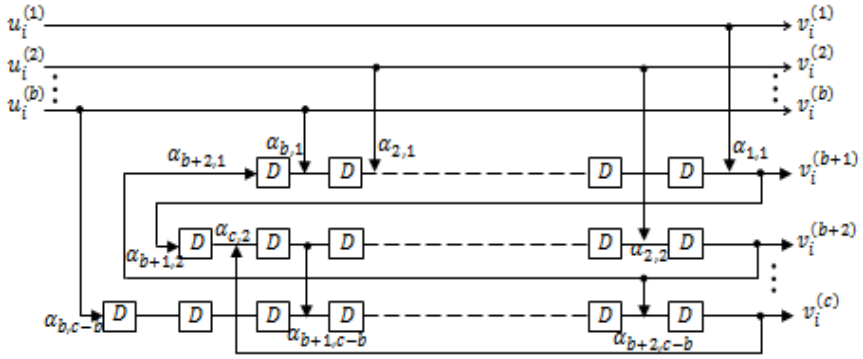


FIGURE 2-8 : Structure d'un codeur convolutionnel récursif systématique composé de $(c-b)$ registres en parallèle et d'un taux de codage $r = b/c$.

Nous pouvons mentionner que, seuls les codes RCDO non terminés ont été étudiés dans un contexte de transmission continu. Or, de nos jours, plusieurs systèmes de communication basés sur la transmission par paquets comme dans les normes Ethernet, WiMax et LTE sont de plus en plus utilisés et intégrés dans presque toutes les machines mobiles. D'où la nécessité d'examiner le problème lié à la détermination de la séquence de terminaison associée à un code convolutionnel RCDO. Pour étudier ce problème ouvert pour tous les types de codes convolutionnels récursifs, nous présentons dans la section suivante quelques caractéristiques liées aux codes LDPC en blocs et convolutionnels. Ceci nous permettra par la suite de déterminer la séquence de terminaison des codes RCDO.

2.5 Codes LDPC

2.5.1 Codes LDPC en blocs

Les codes LDPC ont été introduits par Robert Gallager en 1962 [5]. Ces codes linéaires en blocs LDPC-B (n, k) sont définis par l'ensemble de $(n-k)$ équations de parité :

$$\mathbf{v} \cdot \boldsymbol{H}^{\boldsymbol{T}} = \mathbf{0} \tag{2.11}$$

où $\mathbf{v} = (v_1, v_2, \dots, v_n)$ est un mot de code binaire de longueur n et $\boldsymbol{H}$ est une matrice de parité creuse de dimension $(n-k) \times n$. Les premiers codes LDPC-B développés par Gallager sont des codes réguliers [5], pour lesquels la matrice $\boldsymbol{H}^{\boldsymbol{T}}$ a la particularité d'avoir un nombre fixe d_λ (d_ρ) de uns sur chaque ligne (colonne). Cela signifie que chaque symbole v_i, $i = 1, \dots, n$, est utilisé

par d_λ équations de parité et chaque symbole de parité est calculé par une équation modulo-2 de d_ρ bits d'entrée.

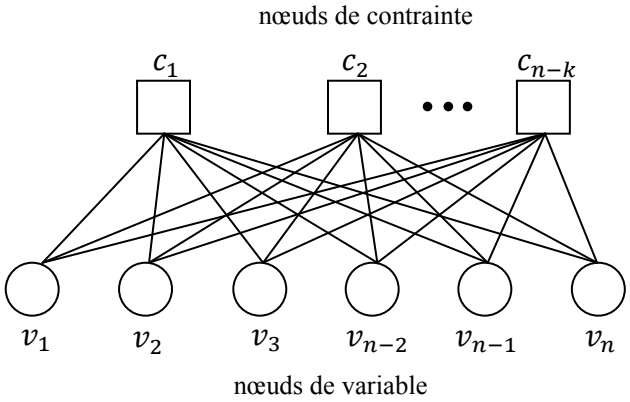


FIGURE 2-9 : Graphe biparti de Tanner d'un code LDPC en bloc régulier $\left(n,\ d_\lambda = 3, d_\rho = 6\right)$ de taux de codage $r = k/n$.

Les codes LDPC peuvent être représentés sous forme de graphes bipartis de Tanner [10]. Un graphe est biparti lorsque l'ensemble de nœuds V se divise en deux sous-ensembles disjoints V_v et V_c, tels que chaque arête joint un nœud dans V_v avec un nœud dans V_c sans que deux nœuds dans V_v ou dans V_c soient connectés. Dans le graphe de Tanner de la Figure 2.9, les deux ensembles de nœuds disjoints V_v et V_c représentent les n nœuds variables $(v_1, v_2, \dots, v_n)$ formant le mot de code et les $(n-k)$ nœuds de contraintes $(c_1, c_2, \dots, c_{n-k})$ qui désignent les $(n-k)$ équations de parité. Si un nœud $v_i \in V_v$ est relié avec un nœud $c_j \in V_c$, cela indique que ce symbole v_i fait partie de l'équation de parité représentée par la contrainte c_j, $c_j = \sum_{i=1}^{n} v_i h_{i,j}$, où $h_{i,j}$ est l'élément de la matrice de parité $\boldsymbol{H^T}$de coordonnées (i,j) et $j = 1, 2, \dots, (n-k)$. Un code LDPC-B est dit régulier si les n nœuds variables et les $(n-k)$ nœuds de contraintes du graphe de Tanner sont de degrés $\left(d_\lambda, d_\rho\right)$ constants avec $d_\lambda \neq d_\rho$. Le nombre d'arêtes est donc égal à :

$$\Gamma = n d_\lambda = (n-k) d_\rho \tag{2.12}$$

À partir de la relation (2.12), nous pouvons exprimer le taux de codage $r = k/n$ du code LDPC-B simplement en fonction des degrés d_λ et d_ρ des nœuds du graphe biparti :

$$r = 1 - \frac{d_\lambda}{d_\rho} \tag{2.13}$$

Contrairement aux codes réguliers, les codes irréguliers [11] n'imposent aucune contrainte sur les degrés des nœuds du graphe. Le degré des nœuds variables est alors donné par la distribution $\lambda(x)$ et le degré des nœuds de contraintes par la distribution $\rho(x)$ tel que :

$$\lambda(x) = \sum_{i=2}^{d_\lambda} \lambda_i x^{i-1} \tag{2.14}$$

$$\text{et} \quad \rho(x) = \sum_{i=2}^{d_\rho} \rho_i x^{i-1} \tag{2.15}$$

où d_λ (d_ρ) représente le degré maximal des nœuds variables (contraintes) et λ_i (ρ_i) représente le pourcentage des arêtes incidentes aux nœuds de degré i dans V_v (V_c). Notons que le nombre de nœuds de degré i dans V_v (V_c) est donné par n_i^v (n_i^c), d'où les proportions λ_i et ρ_i peuvent être calculés comme suit :

$$\lambda_i = \frac{in_i^v}{\Gamma} \tag{2.16}$$

$$\text{et} \quad \rho_i = \frac{in_i^c}{\Gamma} \tag{2.17}$$

Par définition, $\lambda(1) = \rho(1) = \sum_i \lambda_i = \sum_i \rho_i = 1$. Donc le degré moyen des nœuds variables $\overline{d_\lambda}$ et des nœuds de contraintes $\overline{d_\rho}$ est le suivant :

$$\overline{d_\lambda} = \frac{\Gamma}{n} = \frac{1}{\int_0^1 \lambda(x)dx} \tag{2.18}$$

$$\overline{d_\rho} = \frac{\Gamma}{(n-k)} = \frac{1}{\int_0^1 \rho(x)dx} \tag{2.19}$$

De même, nous pouvons définir le taux de codage des codes LDPC irréguliers en fonction des degrés moyens :

$$r = 1 - \frac{\overline{d_\lambda}}{\overline{d_\rho}} \tag{2.20}$$

Les codes LDPC-B peuvent être ainsi dénotés par $\left(n, \overline{d_\lambda}, \overline{d_\rho}\right)$ représentant, respectivement, la longueur du mot de code, degrés moyens des nœuds variables et des nœuds de contraintes (le code régulier étant juste un cas particulier). Il a été démontré dans la littérature [12], que si les distributions de l'irrégularité sont optimisées, alors les codes LDPC irréguliers fournissent de meilleures performance que les réguliers, et ce, pour une complexité de décodage équivalente.

2.5.2 Codes LDPC convolutionnels

Dans cette sous-section, nous passons brièvement en revue les codes convolutionnels LDPC (LDPC-C), initialement proposés par Festrom et Zigangirov [24]. D'une part, les codes LDPC-C sont considérés comme une sous-catégorie des codes convolutionnels puisqu'ils partagent les mêmes caractéristiques. D'autre part, les codes LDPC-C représentent une variante de la forme en bloc originale des codes LDPC. Nous commençons par définir la matrice de parité d'un code LDPC-C.

Comme pour un code LDPC-B, un code LDPC-C est défini à partir de sa matrice de parité $\mathbf{H}^{\mathrm{T}}$**,** qui est creuse. Un code C de taux de codage $r = b/c$ est LDPC-C si l'ensemble des mots de codes binaires $\mathbf{v} = (\dots, \mathbf{v_0}, \mathbf{v_1}, \dots, \mathbf{v_t}, \dots)$ représente l'espace nul de $\mathbf{H}^{\mathrm{T}}$, c'est-à-dire $C = \{\mathbf{v} \mid \mathbf{vH}^{\mathrm{T}} = \mathbf{0},\ \mathbf{v_t} \in \mathbb{F}_2^c\}$**.** Cette définition est exactement la même que celle d'un code LDPC-B. Cependant, la matrice de parité $\mathbf{H}^{\mathrm{T}}$ définissant le code LDPC-C ayant un taux de codage $r = b/c$ est une matrice de dimension infinie et peut être exprimée comme suit :

$$\mathbf{H}^{\mathrm{T}} = \begin{bmatrix} \ddots & & \ddots & & & \\ & H_0^T(t-m_s) & \dots & H_{m_s}^T(t) & & 0 \\ & & \ddots & \vdots & \ddots & \\ & 0 & & H_0^T(t) & \dots & H_{m_s}^T(t+m_s) \\ & & & & \ddots & \vdots & \ddots \end{bmatrix}, \tag{2.21}$$

où les sous-matrices $H_i^T(t), i = 0, 1, \dots, m_s$, sont des matrices binaires de dimension c x $(c - b)$. La valeur maximale de i, telle que pour un t donné la matrice $H_i^T(t)$ est non nulle, est appelée la mémoire du code m_s. Théoriquement, le nombre de combinaisons de sous-matrices $H_i^T(t)$ dans $\mathbf{H}^{\mathrm{T}}$ peut être infini, ce qui entraîne un espace infini de complexité pour les algorithmes de codage et de décodage. Par conséquent, dans la pratique, les sous-matrices sont périodiques dans le temps. Notons la période de la matrice de parité $\mathbf{H}^{\mathrm{T}}$ par T_s. Par définition, $H_i^T(t) = H_i^T(t + T_s)$ pour tous t et i possibles. Il s'ensuit que les codes LDPC-C représentent une catégorie des codes convolutionnels périodique à temps-variant. Dans le cas où $T_s = 1$, les équations de parité ne varient pas en fonction du temps et par conséquent le code est appelée code LDPC-C à temps-invariant. Lorsque $T_s > 1$, le code LDPC-C est à temps-variant. Les auteurs de [24] ont proposé une méthode de construction d'un code convolutionnel LDPC-C à partir d'un code LDPC-B. Un exemple de cette construction est illustré à la Figure 2.10.

Exemple 2.1 — Un code convolutionnel LDPC-C $(m_s = 4, d_\lambda = 3, d_\rho = 6)$ de taux de codage $r = b/c = 1/2$.

Nous commençons avec une matrice de parité d'un code LDPC-B (8, 2, 4) de dimension 8 x 4. Cette matrice sera coupée en dessous de la diagonale (en terme de sous-matrices de dimensions c x $(c - b)$, i.e. 2 x 1) comme illustré dans la Figure 2.10(a). Ensuite, la partie inférieure sera déplacée vers la droite comme montré à la Figure 2.10(b). Puis, des sous-matrices constantes de dimension c x $(c - b)$ seront rajoutées à gauche (Figure 2.10(c)) afin de garantir que les matrices $H_0^T(t)$ soient de rang complet. Ainsi, la matrice obtenue représente une période $T_s = 3$ et sera répétée indéfiniment (Figure 2.10(d)) pour former une matrice de parité $\mathbf{H}^{\mathrm{T}}$ ayant une mémoire $m_s = 4$, $d_\lambda = 3$ uns sur chaque ligne et $d_\rho = 6$ uns sur chaque colonne. Les codes LDPC-C réguliers peuvent être ainsi notés par (m_s, d_λ, d_ρ).

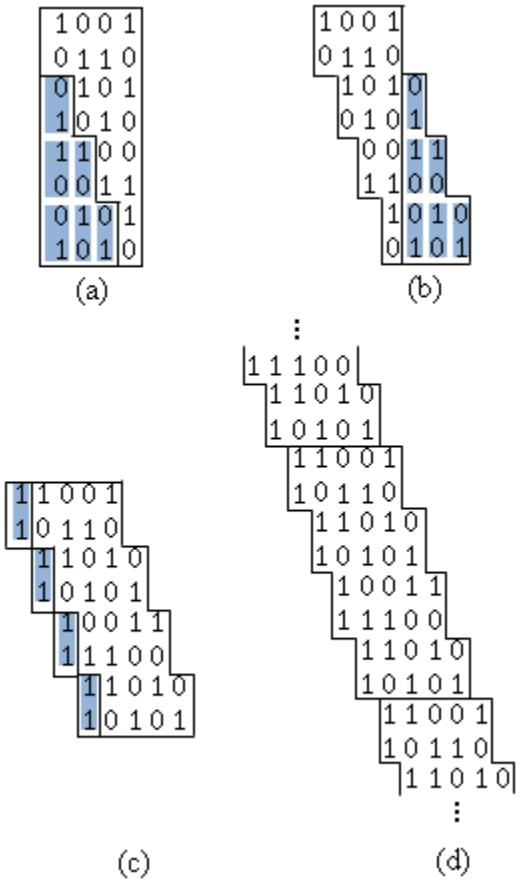


FIGURE 2-10 : Illustration de la méthode de Feltström-Zigangirov pour construire la matrice de parité du code LDPC-C ($m_s = 4, d_\lambda = 3, d_\rho = 6$) de taux de codage $r = 1/2$.

Nous présentons le graphe de Tanner associé aux codes LDPC-C de la façon suivante : chaque ligne de $\mathbf{H}^{\mathrm{T}}$ représente un nœud variable de V_v et chaque nœud contrainte de l'ensemble V_c représente une colonne de $\mathbf{H}^{\mathrm{T}}$. Un nœud de l'ensemble V_v est relié à un nœud de l'ensemble V_c si et seulement si l'élément correspondant dans la matrice $\mathbf{H}^{\mathrm{T}}$ est une valeur non nulle. Comme mentionné précédemment, il y a une étroite correspondance entre la matrice de parité $\mathbf{H}^{\mathrm{T}}$ et sa représentation par graphe de Tanner. Cependant, les propriétés associées aux codes convolutionnels permettent au graphe de Tanner d'un code LDPC-C de se différencier de plusieurs façons à celui d'un code LDPC-B. Nous illustrons ces différences avec l'exemple suivant.

Exemple 2.2 — Illustration du graphe de Tanner du code LDPC-C $(3, 2, 3)$ de taux de codage $r = 1/3$.

Nous considérons le code convolutionnel LDPC-C à temps-invariant (i.e., $T_s = 1$) défini par la matrice de parité $\mathbf{H}^{\mathrm{T}}$ suivante :

$$\mathbf{H}^{\mathrm{T}} = \begin{array}{c} t = \begin{array}{cccccccccccccccc} 0 & & 1 & & 2 & & 3 & & 4 & & 5 & & 6 & & 7 & \end{array} \\ \left[\begin{array}{cccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \vdots & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 1 & 0 & 0 \end{array}\right] \end{array} \begin{array}{l} \vdots \\ v_0^{(1)} \\ v_0^{(2)} \\ v_0^{(3)} \\ v_1^{(1)} \\ v_1^{(2)} \\ v_1^{(3)} \\ v_2^{(1)} \\ v_2^{(2)} \\ v_2^{(3)} \\ v_3^{(1)} \\ v_3^{(2)} \\ v_3^{(3)} \\ v_4^{(1)} \\ v_4^{(2)} \\ v_4^{(3)} \\ v_5^{(1)} \\ v_5^{(2)} \\ v_5^{(3)} \\ \vdots \\ \vdots \\ \vdots \end{array} \tag{2.22}$$

Comme nous pouvons le constater, ce code convolutionnel peut être caractérisé par les paramètres suivants : la mémoire du code m_s qui égale à 3, le nombre d_λ d'éléments non nuls sur

chaque ligne de la matrice $\mathbf{H}^{\mathrm{T}}$ qui égal à 2 et le nombre d_ρ d'éléments non nuls sur chaque colonne de la matrice $\mathbf{H}^{\mathrm{T}}$ qui égal à 3. La Figure 2.11 représente le graphe de Tanner associé à ce code convolutionnel LDPC-C.

La toute première différence entre le graphe de Tanner d'un code convolutionnel et celui d'un code en bloc repose sur le fait que le code convolutionnel est de longueur indéfinie de sorte que le graphe de Tanner correspondant est également de dimension indéfini. De ce fait, chaque symbole du mot de code généré par le codeur convolutionnel sera associé à un indice temporel. Ainsi, les nœuds variables correspondants aux $c = 3$ symboles générés à chaque instant t seront alignés verticalement, comme indiqué à la Figure 2.11. De même, les nœuds de contraintes, représentants les $(c - b) = 2$ équations de contrôle de parité existantes à chaque instant t, seront disposés verticalement (Figure 2.11). Par conséquent, l'indexation temporelle du graphe de Tanner d'un code convolutionnel s'avère nécessaire, contrairement à celui d'un code en bloc.

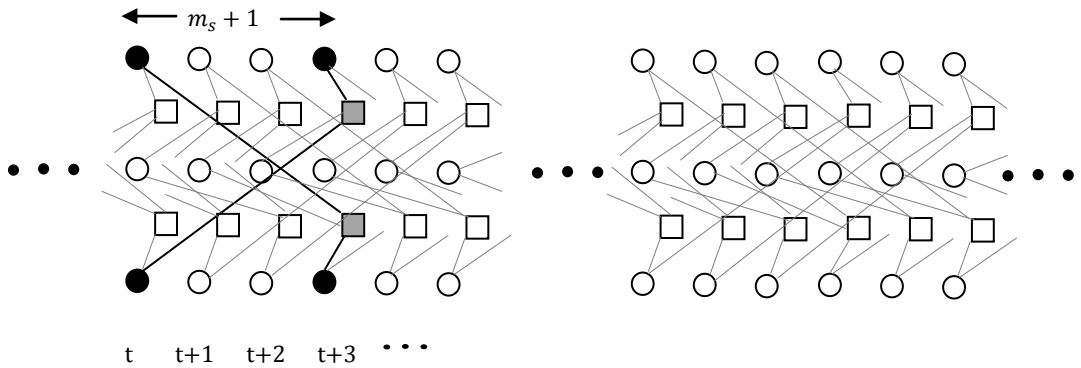


FIGURE 2-11 : Graphe de Tanner du code convolutionnel de l'Exemple 2.2.

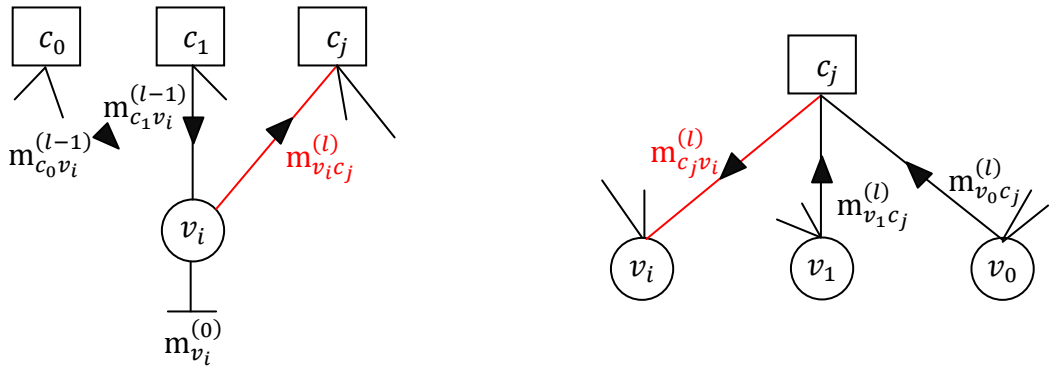
Nous remarquons que la mémoire d'un code convolutionnel permet de définir la structure locale dans le graphe de Tanner. En effet, la mémoire m_s représente la distance maximale entre deux nœuds variables connectés à un nœud de contrainte donné. Comme nous pouvons le constater dans la Figure 2.11, la paire de nœuds variables, situés aux instants t et $t + 3$, sont séparés de $m_s = 3$ unités de temps. Nous notons également que dans un graphe de Tanner associé à un code convolutionnel, les nœuds variables sont connectés seulement aux nœuds de contraintes situés au même instant ou à des instants ultérieurs. Enfin, les connexions des nœuds dans le graphe de Tanner du code LDPC-C à temps-invariant sont les mêmes pour tous les instants t possibles.

Typiquement, les codes LDPC possèdent une mémoire m_s (ou $(c-b)$) assez grande ce qui ne rend pas la représentation en treillis pratique. En effet, puisque les matrices de parité définissant les codes LDPC sont creuses, les graphes de Tanner qui en découlent sont à leur tour de faible densité. Comme nous le verrons dans la section suivante, le graphe de Tanner offre un moyen pratique pour décrire les algorithmes de décodage itératifs.

2.6 Décodage itératif

Dans cette section, nous nous intéressons au décodage itératif des codes correcteurs d'erreur. Dès que la taille du mot de code devient importante, le décodage effectué à l'aide d'un décodeur à maximum de vraisemblance présente une complexité trop élevée pour une réalisation pratique. Pour remédier à ce problème, Gallager a introduit un algorithme de décodage itératif sous-optimal avec un nombre constant d'opérations par bit [5]. Cet algorithme à faible complexité a ensuite été revu par Mackay et Neal [6], Kschischang *et al* [25] et Richardson et Urbanke [26] dans le cadre de la théorie des graphes. Ainsi, un des algorithmes les plus utilisés dans le décodage itératif des codes LDPC est connu sous le nom de "propagation de croyance" (en anglais *Belief Propagation*, BP). Cet algorithme consiste à échanger de l'information entre les nœuds du graphe biparti à travers les branches. Ces messages transitant de nœuds en nœuds peuvent être soit des probabilités, soit des logarithmiques des rapports de vraisemblance. Le principe de BP est l'application de la règle de Bayes afin de calculer une estimation de chaque bit. Ces estimations, représentant des messages se propageant sur les branches du graphe biparti, sont alors échangées à chaque itération pour calculer les probabilités *a posteriori* sur chaque bit. Dans le cas d'un décodage sur un graphe sans cycle, les messages échangés sont statistiquement indépendants, la factorisation de Bayes conduit au calcul exact des probabilités *a posteriori* des nœuds variables. Dans le cas des codes couramment utilisés, le graphe de Tanner associé comporte des cycles. Ainsi, l'hypothèse sur l'indépendance des messages n'est plus valide. L'objectif devient donc de construire un graphe de Tanner dont le plus petit cycle est le plus grand possible. Pour ce faire, il existe plusieurs méthodes de recherche des codes comme par exemple, l'algorithme *Progressive Edge-Growth* (PEG) [43] [44]. Le principe de l'algorithme PEG consiste à placer progressivement les arêtes entre les nœuds variables et les nœuds de contraintes tout en essayant à chaque étape de maximiser le cycle du sous-graphe local. Bien que

l'optimisation locale soit appliquée, la dépendance entre les différents messages échangés est devenue faible et de très bonnes performances de décodage ont été obtenues.



(a) (b)

FIGURE 2-12 : Propagation des messages dans un graphe biparti.

Chaque itération l de l'algorithme BP est composée essentiellement de deux phases. La première étape, illustrée à la Figure 2.12 (a), consiste à la mise à jour des messages transmis d'un nœud variable vers un nœud de contrainte. La seconde étape, illustrée à la Figure 2.12 (b), calcule les messages propagés des nœuds de contraintes vers les nœuds variables. Enfin, après avoir réalisé un nombre maximum d'itérations, l'information *a posteriori* pour chacun des nœuds variables est calculée afin de prendre une décision. Nous précisons qu'en réalité les messages échangés représentent l'information probabiliste *a posteriori* des symboles provenant du canal, ou de façon équivalente sur le logarithme des rapports de vraisemblance (LRV) de ces symboles. Nous notons $m_{v_i c_j}^{(l)}$, le message transmis à la l-ème itération d'un nœud variable v_i vers un nœud de contrainte c_j alors que le message échangé à la l-ème itération d'un nœud de contrainte c_j vers un nœud variable v_i est noté $m_{c_j v_i}^{(l)}$. Les équations de mise à jour des messages s'écrivent comme suit :

$$m_{v_i c_j}^{(l)} = \begin{cases} m_{v_i}^{(0)}, & \text{pour } l = 0 \\ m_{v_i}^{(0)} + \sum_{\substack{c' \in \mathcal{N}(v_i) \\ c' \neq c_j}} m_{c' v_i}^{(l-1)}, & \text{pour } l \geq 1 \end{cases} \tag{2.23}$$

$$\mathrm{m}_{c_j v_i}^{(l)} = \ln \frac{1+\prod_{\substack{v' \in \mathcal{N}(c_j) \\ v' \neq v_i}} \tanh\left(\frac{\mathrm{m}_{v' c_j}^{(l)}}{2}\right)}{1-\prod_{\substack{v' \in \mathcal{N}(c_j) \\ v' \neq v_i}} \tanh\left(\frac{\mathrm{m}_{v' c_j}^{(l)}}{2}\right)} \tag{2.24}$$

où $\mathrm{m}_{v_i}^{(0)}$ représente le LRV observé en sortie du canal en v_i, $\mathcal{N}(v_i)$ l'ensemble des nœuds de contraintes connectés au nœud variable v_i, et $\mathcal{N}(c_j)$, l'ensemble des nœuds variables connectés au nœud de contrainte c_j. Après chaque itération l, une décision peut être prise sur l'information a posteriori $\mathrm{m}_{v_i}^{(l)}$ associé au nœud variable v_i. Cette information a posteriori est égale à la somme de tous les messages rentrant dans le nœud variable concerné :

$$\mathrm{m}_{v_i}^{(l)} = \mathrm{m}_{v_i}^{(0)} + \sum_{\substack{c' \in \mathcal{N}(v_i) \\ c' \neq c_j}} \mathrm{m}_{c' v_i}^{(l)} \tag{2.25}$$

Finalement, une décision dure sur la valeur binaire de chaque nœud variable est prise, à chaque itération l, en fonction du signe de l'information *a posteriori* :

$$\hat{v}_i = \begin{cases} 0, & \text{si } \mathrm{m}_{v_i}^{(l)} \geq 0, \\ 1, & \text{autrement.} \end{cases} \tag{2.26}$$

Ainsi, le processus itératif est arrêté au bout d'un nombre maximum d'itérations. Nous pouvons également arrêter le processus itératif avant le nombre maximum d'itérations en calculant à chaque itération le syndrôme. Si celui-ci est nul alors le décodage itératif a convergé vers un mot de code et le processus peut être arrêté.

Dans la littérature, une méthode permettant de prédire les performances du décodage itératif par propagation de croyance, nommée évolution de densité, a été développé par Richardson *et al.* [12]. Cette méthode consiste à suivre les densités de probabilités des différents messages se propageant dans le graphe lors de l'échange des informations sous l'hypothèse que ces messages soient indépendants (i.e. le graphe de Tanner est localement sans cycle) et identiquement distribués [26]. Cette technique permet alors de savoir quel est le rapport signal sur bruit E_b/N_0 au dessus de laquelle l'algorithme itératif de décodage peut converger, et ainsi d'en trouver la paire de distributions $\rho(x)$ et $\lambda(x)$ adéquate. Dans [45], les auteurs ont réussi à trouver la paire de distributions d'un code LDPC-B irrégulier ayant une longueur de bloc égale à $n = 10^7$ et un

taux de codage $r = 1/2$ capable de s'approcher à 0.0045 dB de la limite de shannon pour un TEB égal à 10^{-6}.

2.7 Conclusion

Dans ce chapitre, nous avons brièvement présenté les codes en blocs et les codes convolutionnels dans le cadre des codes correcteurs d'erreurs. Les codes Turbo, les codes convolutionnels doublement orthogonaux et les codes LDPC en blocs et convolutionnels ont été présentés ainsi que leurs propriétés. De même, nous avons présenté un bref état de l'art sur le décodage itératif et tout ce que nous considérons comme essentiels pour la bonne compréhension de la suite du mémoire.

Dans le chapitre suivant, nous nous intéresserons tout particulièrement aux codes RCDO, qui offrent à nos yeux un excellent compromis entre performance d'erreur et simplicité de décodage. Ceci nous permettra par la suite de construire des codes RCDO terminables qui peuvent être appliqués à des systèmes de communications basés sur une transmission par paquets.

CHAPITRE 3

ANALYSE ET CONSTRUCTION DES CODES CONVOLUTIONNELS RÉCURSIFS DOUBLEMENT ORTHOGONAUX

Le concept des codes convolutionnels récursifs multi-registres doublement orthogonaux (RCDO) a été présenté initialement dans [19]. Une étude récente [28] a montré que les codes RCDO représentent une version convolutionnelle des codes LDPC-B et une alternative fascinante pour construire les codes convolutionnels LDPC-C à temps-invariant. Les codes RCDO sont spécifiés entièrement par leur matrice de contrôle de parité, creuse et semi-infinie, et peuvent être représentés à l'aide d'un graphe de Tanner infini, pour lequel les longueurs des cycles sont contrôlés par les conditions de double orthogonalité imposées sur l'ensemble de connexions définissant la structure du codeur. Par conséquent, les codes RCDO peuvent être décodés par un décodeur itératif composé d'un nombre de décodeurs à seuil identiques en série. Il a été montré dans [19] et [28] que la convergence du décodeur itératif RCDO vers la capacité du canal dépend essentiellement de la distribution des arrêtes $(\lambda(x), \rho(x))$ dans le graphe de Tanner associé.

Dans ce chapitre, nous présentons un aperçu détaillé et une analyse complète des codes RCDO. La définition des codes RCDO ainsi que la terminologie utilisée tout au long de ce mémoire sont données dans la Section 3.1. À la Section 3.2, nous présentons les conditions de double orthogonalité et l'algorithme permettant de construire les matrices de contrôle de parité RCDO à partir des distributions (λ, ρ). La technique d'encodage continu des codes convolutionnels récursifs multi-registres est envisagée à la Section 3.3. Puis à la Section 3.4, nous illustrons le processus d'encodage à l'aide d'un simple code RCDO. Enfin, nous étudions le décodage itératif des codes RCDO à la Sections 3.5.

3.1 Définitions et terminologies des codes RCDO

Comme nous l'avons vu pour les codes LDPC-C, les codes convolutionnels récursifs doublement orthogonaux sont obtenus à partir des matrices de contrôle de parité à temps-invariant. Un encodeur RCDO est spécifié par la matrice des connexions $\boldsymbol{A} = [\alpha_{i,j}]$ de dimension $c \times (c-b)$ où, c représente le nombre de symboles générés à la sortie de l'encodeur, dont les b premiers symboles correspondent aux symboles d'information à l'entrée du codeur convolutionnel, $(c-b)$ est donc le nombre de symboles de parité générés par l'encodeur et qui correspond aussi au nombre de ses registres à décalage et finalement, $\alpha_{i,j}$ représente la position de la connexion entre le $i-$ième symbole à la sortie du codeur et le $\alpha_{i,j}-$ième élément de délai du $j-$ième registre à décalage. Le taux de codage du codeur RCDO est alors égal à $r = b/c$. Par ailleurs, un code convolutionnel RCDO étant de longueur indéfinie par définition, sa matrice de parité est aussi de dimension indéfinie et peut être représentée sous deux formes. La représentation polynômiale, intitulée *transformée en* D [9], est plus commode pour définir un encodeur RCDO ainsi que ses conditions de double orthogonalité [29]. Cependant, la représentation temporelle ou binaire [24], est plus adéquate pour étudier la problématique de terminaison des codes récursifs convolutionnels.

3.1.1 Représentation polynômiale des matrices de contrôle de parité des codes RCDO

Dans cette section, nous représentons la matrice de contrôle de parité binaire $\mathbf{H}^{\mathrm{T}}$ sous sa forme la plus compacte en utilisant la *transformée en* D. Avec cette représentation, la matrice de contrôle $\mathbf{H}^{\mathrm{T}}(D)$ d'un code RCDO de taux de codage $r = b/c$ peut être exprimée sous cette forme générale :

$$\mathbf{H}^{T}(D) = \begin{pmatrix} h_{1,1}D^{\alpha_{1,1}} & \cdots & h_{1,(c-b)}D^{\alpha_{1,(c-b)}} \\ h_{2,1}D^{\alpha_{2,1}} & \cdots & h_{2,(c-b)}D^{\alpha_{2,(c-b)}} \\ \vdots & & \vdots \\ h_{b,1}D^{\alpha_{b,1}} & \cdots & h_{b,(c-b)}D^{\alpha_{b,(c-b)}} \\ h_{b+1,1}D^{\alpha_{b+1,1}} & \cdots & h_{b+1,(c-b)}D^{\alpha_{b+1,(c-b)}} \\ \vdots & & \vdots \\ h_{c,1}D^{\alpha_{c,1}} & \cdots & h_{c,(c-b)}D^{\alpha_{c,(c-b)}} \end{pmatrix} \tag{3.1}$$

de dimension $c \times (c-b)$. Les coefficients $h_{i,j}$ dans (3.1), $h_{i,j} \in \{0,1\}$, indiquent l'existence d'une connexion, i.e. la valeur $h_{i,j}$ est égale à 1, entre le $i-$ième symbole à la sortie du codeur RCDO et le $j-$ième registre à décalage. Sinon, $h_{i,j}$ prend la valeur zéro. La variable D représente un opérateur de délai et l'exposant de D, qui correspond à la valeur $\alpha_{i,j}$, constitue la matrice de connexions $\boldsymbol{A}$ d'un encodeur RCDO. Nous notons que la mémoire du code m_s est égale à la valeur la plus grande de $\alpha_{i,j}$, c'est-à-dire $m_s = \max_{i\in\{1,\dots,c\},j\in\{1,\dots,(c-b)\}}(\alpha_{i,j})$. Étant donné que les codes RCDO considérés sont systématiques, les éléments formant la diagonale inférieure de la matrice $\mathbf{H}^T(D)$ sont égaux à 1, c'est-à-dire que les coefficients $h_{b+j,j}$ prennent la valeur 1, et les éléments $\alpha_{b+j,j}$ égaux à 0 pour tout $j \in \{1,\dots,(c-b)\}$. L'auteur de [27] a montré, que le contrôle des éléments $h_{i,j}$ non nuls dans la matrice de parité $\mathbf{H}^T(D)$ et l'établissement d'une configuration bien précise de ces éléments à l'aide de l'algorithme *Évolution de la Densité de Probabilité* permet d'approcher les performances d'erreur des codes RCDO de la capacité de Shannon. De plus, comme nous le verrons dans la Section 3.3, un ensemble de conditions de double orthogonalité doit être imposé aux positions des connexions $\alpha_{i,j}$ des codeurs RCDO [28].

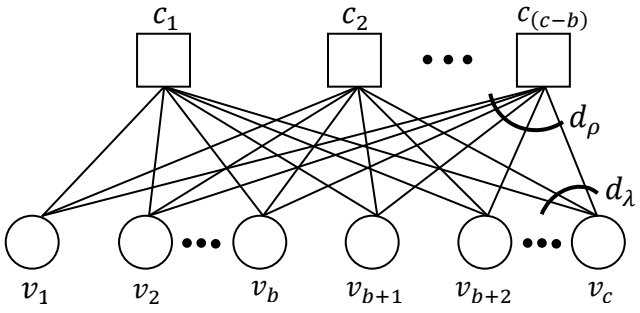


FIGURE 3-1 : Protographe représentatif d'un ensemble de codes convolutionnels récursifs multi-registres de taux de codage $r = b/c$. Les nœuds représentés par des cercles sont des nœuds variables et les nœuds représentés par des carrés sont des nœuds de contraintes.

À partir de la matrice de contrôle de parité $\mathbf{H}^T$ d'un code RCDO, nous considérons une représentation graphique du code à l'aide d'un graphe biparti de Tanner. En effet, ce graphe de Tanner peut être obtenu à partir de plus petit graphe biparti fondamental, appelé *protographe* [30]. La Figure 3.1 présente un protographe général associé à une matrice de contrôle de parité

$\mathbf{H}^T(D)$ d'un code RCDO de taux de codage $r = b/c$. Sur cette figure, chaque nœud variable représente une ligne de $\mathbf{H}^T(D)$ tandis que chaque nœud de contrainte représente une colonne de $\mathbf{H}^T(D)$. Ainsi, le nombre de nœuds variables est égal à c, puisque la matrice $\mathbf{H}^T(D)$ est composée de c lignes, alors que le nombre de nœuds de contraintes est égal à $(c-b)$ qui correspond au nombre de colonne de $\mathbf{H}^T(D)$. Chaque élément $h_{i,j}$ égal à 1, représente une arête dans le protographe reliant le $i-$ième nœud variable (v_i) au $j-$ième nœud de contrainte (c_j). Le degré d_λ (d_ρ) d'un nœud variable v_i (de contrainte c_j) correspond au nombre d'arêtes incidentes à ce dernier et représente le nombre d'éléments non nuls à la $i-$ième ligne ($j-$ième colonne) de la matrice $\mathbf{H}^T(D)$. Un protographe est dit régulier, si le degré d_λ de tous les nœuds variables ainsi que le degré d_ρ de tous les nœuds de contraintes sont constants. Dans le cas contraire, le protographe est dit irrégulier.

3.1.2 Représentation binaire des matrices de contrôle de parité des codes RCDO

Par définition, un code convolutionnel multi-registres RCDO de longueur infinie est généré à partir d'une matrice de contrôle de parité de dimension infinie. La matrice de contrôle $\mathbf{H}^{\mathrm{T}}$ d'un code RCDO de taux de codage $r = b/c$ peut être exprimée sous sa forme binaire comme suit :

$$\mathbf{H}^{\mathrm{T}} = \begin{bmatrix} \ddots & & \ddots & & & \\ & H_0^T(t-m_s) & \dots & H_{m_s}^T(t) & 0 & \\ & & \ddots & \vdots & \ddots & \\ & 0 & & H_0^T(t) & \dots & H_{m_s}^T(t+m_s) \\ & & & & \ddots & \vdots & \ddots \end{bmatrix}, \tag{3.2}$$

où $H_i^T(t), i = 0, 1, \dots, m_s$, sont des sous-matrices binaires de dimension $c \times (c-b)$. Pour un code RCDO systématique, ces sous-matrices peuvent être subdivisées davantage en une partie d'information $H_i^{(u)}(t)$ de dimension $b \times (c-b)$ et en une partie de parité $H_i^{(v)}(t)$ de dimension $(c-b) \times (c-b)$.

$$H_i^T(t) = \begin{bmatrix} h_i^{(1,1)}(t) & \dots & h_i^{(1,c-b)}(t) \\ \vdots & & \vdots \\ h_i^{(c,1)}(t) & \dots & h_i^{(c,c-b)}(t) \end{bmatrix} \xrightarrow{\text{Code systématique}} \left[H_i^{(u)}(t)\ H_i^{(v)}(t)\right]^T. \tag{3.3}$$

Pour ce type de code multi-registres, chaque unité de temps t correspond à une sous-matrice colonne de $\mathbf{H}^{\mathrm{T}}$, i.e. $(c-b)$ équations de contraintes. Pour chaque sous-matrice $H_i^T(t)$, l'indice i localise sa position relative dans les $(c-b)$ équations de contraintes. Dans la Figure 3.2, la *mémoire du code* m_s peut être définie comme le nombre maximal des sous-matrices $H_i^T(t)$ en verticale ou en horizontale dans $\mathbf{H}^{\mathrm{T}}$ moins un. La *longueur de contrainte* d'un code convolutionnel RCDO est égale à $\vartheta = c \cdot (m_s + 1)$. Typiquement, la longueur de contrainte et la mémoire du code sont des paramètres essentiels à la définition d'un code convolutionnel, puisqu'ils déterminent le nombre maximal des bits passés impliqués dans le calcul du symbole de parité actuel. Pour les codes en blocs, le terme "taille" est parfois utilisé pour représenter la longueur du bloc de ces codes. De même pour les codes convolutionnels RCDO, la "taille" d'un code peut être définie comme étant la mémoire du code ou la longueur de contrainte. Évidement, les paramètres m_s et ϑ d'un code RCDO sont de tailles finies, ce qui lui permet d'être codé/décodé avec une complexité finie.

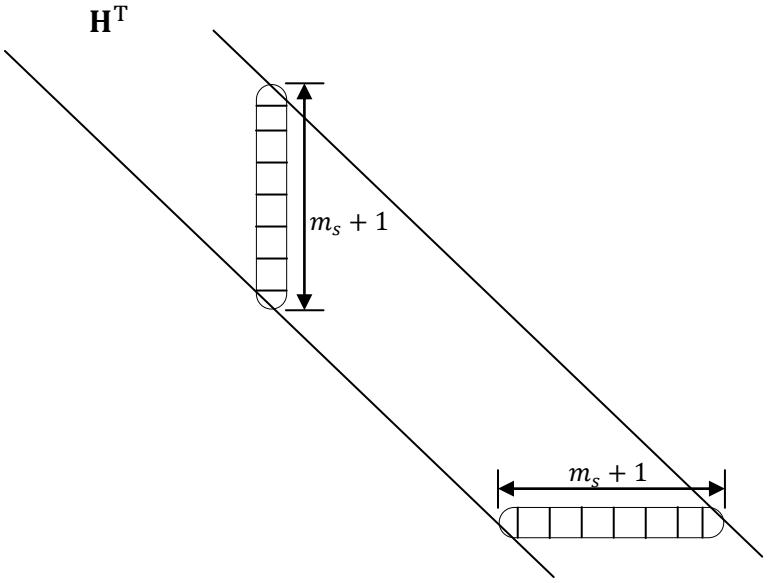


FIGURE 3-2 : Représentation de la matrice de contrôle de parité infinie d'un code RCDO, ayant un taux de codage $r = b/c$ et une mémoire m_s. Chaque cellule à droite représente une sous-matrice de dimension $c \times (c-b)$.

Même avec une mémoire de code m_s finie, l'implémentation du codeur/décodeur RCDO ne serait pas possible sans l'ajout des contraintes supplémentaires sur la valeur des sous-matrices. Théoriquement, la matrice de contrôle de parité $\mathbf{H}^{\mathrm{T}}$ peut être composée d'un nombre infini de

combinaisons de sous-matrices $H_i^T(t)$. Toutefois, ces combinaisons infinies des sous-matrices rendent la complexité des algorithmes de codage et de décodage infinie. Pour cela, en pratique, nous construisons des sous-matrices périodiques dans le temps. Par définition, les codes RCDO sont répertoriés sous la catégorie des codes convolutionnels à temps-invariant de période $T_s = 1$. Ceci se traduit par le fait que les équations de contraintes ne varient pas avec le temps et que les sous-matrices sont de période 1 tel que $H_i^T(t) = H_i^T(t+1)$ pour toutes valeurs de i et de t possibles.

Par analogie avec la partition de la matrice de contrôle en sous-matrices, un mot de code RCDO est, à son tour, composé de sous-vecteurs. Nous définissons le mot de code de longueur infinie $\mathbf{v}$ généré à partir d'un code RCDO de taux de codage $r = b/c$, comme suit :

$$\mathbf{v} = [\dots, \mathbf{v_0}, \dots, \mathbf{v_t}, \dots], \tag{3.4}$$

où $\mathbf{v_t}$ est un sous-vecteur de longueur c. Dans le cas d'un code systématique, chaque sous-vecteur $\mathbf{v_t} = \left[v_t^{(1)}, v_t^{(2)}, \dots, v_t^{(c)}\right]$ est composé de deux parties : les b premiers bits du vecteur $\mathbf{v_t}$ représentent les b bits d'information $\mathbf{u_t} = \left[u_t^{(1)}, u_t^{(2)}, \dots, u_t^{(b)}\right]$, par ailleurs, les $(c-b)$ dernières composantes binaires du vecteur $\mathbf{v_t}$ correspondent aux $(c-b)$ symboles de parité.

Le graphe de Tanner associé à un code RCDO est également de longueur infinie, périodique et de temps-invariant. Toutes les connexions entre les nœuds se répètent de la même façon à chaque instant t. À la Figure 3.3, nous présentons un graphe de Tanner infini associé à un code RCDO à temps-invariant de taux de codage $r = 2/4$ et de mémoire m_s égale à 2. Comme pour les codes en blocs, chaque colonne de $\mathbf{H}^{\mathrm{T}}$ correspond à un nœud de contrainte dans le graphe de Tanner et chaque nœud variable représente une ligne de $\mathbf{H}^{\mathrm{T}}$. Un nœud variable est relié à un nœud de contrainte si et seulement si l'élément correspondant dans $\mathbf{H}^{\mathrm{T}}$ est égal à 1. Les codes RCDO réguliers ou irréguliers sont définis de la même manière que les codes en blocs. Pour un code RCDO régulier, le nombre d_λ d'éléments non nuls sur chaque ligne de $\mathbf{H}^{\mathrm{T}}$, ainsi que le nombre d_ρ d'éléments non nuls sur chaque colonne de $\mathbf{H}^{\mathrm{T}}$ sont uniformes. Dans la suite de ce mémoire, nous notons ces codes RCDO par (m_s, d_λ, d_ρ). Dans le cas contraire, le code est irrégulier et sera caractérisé par une paire de distributions $(\lambda(x), \rho(x))$.

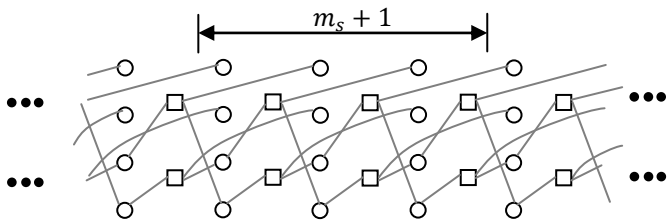


FIGURE 3-3 : Graphe de Tanner associé à un code RCDO ayant un taux de codage $r = 2/4$ et une mémoire $m_s = 2$. Les nœuds ronds sont des nœuds variables et les nœuds carrés sont les nœuds de contraintes.

3.2 Construction des encodeurs RCDO

Pour une catégorie de codes correcteurs d'erreurs, le défi majeur auquel sont confrontés les chercheurs est de déterminer comment construire des codes ayant les meilleures performances et les paramètres souhaités, tels que le taux de codage r, la mémoire m_s, etc. Bien que les codes algébriques traditionnels sont généralement prédéfinis et peuvent être générés à l'aide des approches algébriques, la construction des codes qui permettent de s'approcher de la capacité du canal demeure une tâche complexe. Pour les codes Turbo, la construction de ce type de code repose sur la sélection des encodeurs constituants et la conception de l'entrelaceur [33]. Pour les codes en blocs LDPC-B, des recherches approfondies ont été menées afin de construire des codes. Plusieurs méthodes ont été proposées, parmi lesquelles on retrouve des techniques de construction basées sur la géométrie finie [9], les protographes [34], etc. De même pour les codes convolutionnels LDPC-C, les méthodes de construction existantes dans la littérature peuvent être grossièrement classées en construction aléatoire [24], [35] et en construction algébrique [36], [37] et [38].

Ce n'est que récemment que les codes convolutionnels récursifs multi-registres doublement orthogonaux RCDO ont fait leur apparition parmi l'ensemble des codes correcteurs d'erreurs. Dans cette section, nous présentons les conditions de double orthogonalité qui doivent être imposées sur les positions des connexions $\{\alpha_{i,j}\}$ afin d'éliminer les petits cycles dans le graphe de Tanner des codes RCDO [29]. De plus, nous introduisons un algorithme basé sur un ensemble d'heuristiques pour la recherche des codes RCDO offrant des performances d'erreur aussi près que possible de la capacité de Shannon [27].

3.2.1 Conditions d'orthogonalité

Les résultats préliminaires montrent que les conditions de double orthogonalité permettent de contrôler la longueur des cycles qui existent dans les graphes de Tanner associés aux codes RCDO [29]. Lorsque les conditions de double orthogonalité sont respectées, le graphe de Tanner des codes RCDO ne contient aucun cycle de longueur 4, 6 et 8 [28]. Cette propriété permet aux messages échangés entre les nœuds variables et les nœuds de contraintes dans le graphe de Tanner des codes RCDO d'être indépendants les uns des autres sur deux itérations successives. Dans ce qui suit, nous définissons les conditions de la simple orthogonalité et de double orthogonalité que doivent satisfaire les positions des connexions $\alpha_{i,j}$ du codeur récursif convolutionnel multi-registres.

Définition 3.1 : *Un code convolutionnel récursif multi-registres de taux de codage $r = b/c$ est dit simplement orthogonal si l'ensemble des connexions satisfait la condition suivante :*

Les différences $(\alpha_{i,j} - \alpha_{i,k})$ sont distinctes des différences $(\alpha_{l,j} - \alpha_{l,k})$,

où les indices $(i, l) \epsilon \{1,2, \dots, c\}$ et $(j, k) \epsilon \{1,2, \dots, (c - b)\}$ tels que $i \neq l, j \neq k$.

Tous les codes convolutionnels récursifs simplement orthogonaux possèdent un graphe de Tanner dont la longueur du plus petit cycle est supérieure ou égale à 6. Cependant, lors du décodage des mots de code à l'aide d'un décodeur itératif, ces conditions de simple orthogonalité ne garantissent pas l'indépendance entre les messages échangés d'une itération à l'autre. Il s'ensuit que l'ajout de deux conditions supplémentaires s'avère nécessaire pour assurer un échange de messages indépendants sur deux itérations successives.

Définition 3.2 : *Un code convolutionnel récursif multi-registres de taux de codage $r = b/c$ est dit doublement orthogonal (RCDO) si l'ensemble des connexions satisfait les conditions suivantes :*

1. Les différences $(\alpha_{i,j} - \alpha_{i,k})$ sont distinctes des différences $(\alpha_{l,j} - \alpha_{l,k})$, $i \neq l, j \neq k$.

2. Les différences $(\alpha_{i,j} - \alpha_{i,k})$ sont distinctes des différences de différences $(\alpha_{l,j} - \alpha_{l,s}) - (\alpha_{m,s} - \alpha_{m,k})$, $i \neq m, i \neq l, j \neq k, s \neq k, l \neq m, s \neq j$.

3. Les différences de différences $(\alpha_{l,j} - \alpha_{l,s}) - (\alpha_{m,j} - \alpha_{m,k})$ *sont distinctes des différences de différences* $(\alpha_{e,k} - \alpha_{e,b}) - (\alpha_{r,s} - \alpha_{r,b})$, $l \neq m, l \neq r, e \neq m, e \neq r, b \neq k, b \neq s, k \neq j, s \neq j$.

où les indices $(i, l, m, e, r) \epsilon \{1,2, \dots, c\}$ *et* $(j, k, s, b) \epsilon \{1,2, \dots, (c - b)\}$.

Les conditions de double orthogonalité représentent une alternative aux algorithmes, tels que *Progressive Edge Growth* (PEG) [31], afin de construire le graphe de Tanner associé à un code. Ces conditions doivent être appliquées seulement sur les éléments $\alpha_{i,j}$ pour les quelles les coefficients $h_{i,j}$ de la matrice $\mathbf{H}^T(D)$ sont égaux à un. Nous rappelons que les conditions de double orthogonalité ne peuvent être vérifiées que s'il existe au plus une connexion entre la $i-$ième sortie de l'encodeur et le $j-$ième registre à décalage. Dans le cas où il existe plus qu'une connexion entre la $i-$ième sortie de l'encodeur et le $j-$ième registre à décalage, les différences de différences ne seront plus complètement distinctes, d'où l'apparition des cycles de longueurs 6 et 8 dans le graphe de Tanner de l'encodeur RCDO. Il a été montré dans [32] que dans le choix des valeurs $\alpha_{i,j}$ de la matrice de contrôle de parité $\mathbf{H}^T(D)$ même en appliquant les conditions de double orthogonalité les performances d'erreur des codes ne s'améliorent pas suffisamment. Par ailleurs, la recherche de la distribution adéquate des arrêtes $(\lambda(x), \rho(x))$ dans le graphe de Tanner du code RCDO, et par conséquent la distribution des coefficients $h_{i,j}$ non nuls dans $\mathbf{H}^T(D)$, joue un rôle important dans la construction des codes RCDO capables d'approcher la capacité de Shannon.

3.2.2 Algorithme de recherche de codes RCDO

L'algorithme de construction des codes RCDO utilisé est basé sur un ensemble d'heuristiques de recherche aléatoire des positions de connexion $\{\alpha_{i,j}\}$ satisfaisant les conditions de double orthogonalité. Cet algorithme à été introduit dans [27] afin de générer des graphes de Tanner comportant des cycles de longueurs supérieures ou égales à 10 tout en minimisant la latence introduite par une itération de décodage, c'est-à-dire que nous cherchons des codes dont la mémoire du code m_s est la plus petite possible. Le principal avantage de cet algorithme est sa flexibilité. En effet, nous pouvons générer des codes RCDO à temps-invariant réguliers et irréguliers ayant n'importe quelle longueur de contrainte ϑ et n'importe quel taux de codage r.

Dans le but de décrire l'algorithme de recherche, nous considérons un code RCDO de taux de codage $r = b/c$ défini par sa matrice de contrôle de parité $\mathbf{H}^T(D)$ de dimension $c \times (c-b)$ composée par les connexions $\alpha_{i,j}$ et les éléments $h_{i,j}$, $i \in \{1, \dots, c\}$, $j \in \{1, \dots, (c-b)\}$. La recherche des valeurs $\alpha_{i,j}$ s'effectue après avoir trouvé la paire de distributions $(\lambda(x), \rho(x))$ associée au protographe du code RCDO et produit la configuration des coefficients non nul $h_{i,j}$ dans $\mathbf{H}^T(D)$. Nous définissons, S, Γ, m_s^{max}, $position$ comme étant respectivement, le sous-ensemble des connexions qui vérifie les conditions de double orthogonalité, le nombre d'éléments non nuls dans $\mathbf{H}^T$, la valeur maximale permise des connexions, et le vecteur qui regroupe les emplacements des connexions dans la matrice. Nous notons $\boldsymbol{E}$ la matrice des connexions, qui sont numérotées dans le vecteur $position$ en lisant le long des colonnes comme suit :

$$\boldsymbol{E}_{c\times(c-b)} = \begin{pmatrix} 1 & \vdots & \cdots & \vdots & c(c-b-1)+1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ b & \vdots & \cdots & \vdots & c(c-b-1)+b \\ b+1 & \vdots & \cdots & \vdots & c(c-b-1)+b+1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ c & \vdots & \cdots & \vdots & c\times(c-b) \end{pmatrix}. \tag{3.5}$$

Par exemple, si $b = 2$ et $c = 4$, pour trouver les connexions $\alpha_{i,j}$ dont leurs coefficients $h_{i,j}$ sont non nuls dans $\mathbf{H}^T(D)$ telle que

$$\mathbf{H}^T(D) = \begin{pmatrix} D^{\alpha_{1,1}} & 0 \\ 0 & D^{\alpha_{2,2}} \\ D^{\alpha_{3,1}} & D^{\alpha_{3,2}} \\ D^{\alpha_{4,1}} & D^{\alpha_{4,2}} \end{pmatrix},$$

le vecteur $position$ sera $\{1, 3, 4, 6, 7, 8\}$. La longueur du vecteur $position$ qui correspond au nombre total de connexions, est égale à Γ. Nous nous servons aussi d'un compteur, cpt, pour indiquer le numéro de la connexion en cours de recherche située dans la matrice $\mathbf{H}^T(D)$ à l'emplacement $position(cpt)$. Par définition, la matrice de contrôle $\mathbf{H}^T(D)$ d'un code RCDO ayant un taux de codage $r = b/c$, est construite de sorte que chaque élément formant la diagonale inférieure soit égal à 1, c'est-à-dire que les éléments $\alpha_{(b+j),j}$ prennent la valeur zéro et que les coefficients $h_{(b+j),j}$ soient égaux à 1, $1 \leq j \leq (c-b)$. Le principe de l'algorithme est fournit ci-dessous.

Algorithme de construction de la matrice de contrôle $\mathbf{H}^{T}(D)$ d'un code RCDO

$m_s^{max} \in \mathbb{N}$ fixée, $cpt \leftarrow 0$, $S = \{\emptyset\}$.

tant que $cpt < \Gamma$ **faire**

sélectionner la connexion $\alpha_{i,j}$ qui se trouve à l'emplacement $position(cpt)$.

générer aléatoirement une valeur entière $\alpha_{i,j}$ qui est inférieure à m_s^{max}, $S \leftarrow S \cup \{\alpha_{i,j}\}$.

si S vérifie les conditions de double orthogonalité **alors**

$cpt \leftarrow cpt + 1$

Sinon

retirer la nouvelle valeur ajoutée $\alpha_{i,j}$ de S: $S \leftarrow S \setminus \{\alpha_{i,j}\}$.

fin si

fin tant que

Précisons que si après un certain temps nous ne trouvons pas de solution, nous recommençons une nouvelle recherche.

3.3 Encodage des codes RCDO

Dans cette section, nous présentons et discutons la procédure d'encodage des codes récursifs convolutionnels doublement orthogonaux. L'opération de codage RCDO peut être basée sur la matrice de contrôle du code si les éléments formant la diagonale inférieure de la matrice $\mathbf{H}^{T}(D)$ sont égaux à 1 et par conséquent que lorsque les sous-matrices $H_0^T(t)$ de la matrice binaire $\mathbf{H}^{\mathrm{T}}$ sont de rang complet pour tout instant t. Pour un code systématique RCDO de taux de codage $r = b/c$, la partie $(c-b) \times (c-b)$ inférieure de chaque sous-matrice $H_0^T(t)$ est une matrice d'identité. Ceci peut se réaliser facilement en utilisant l'algorithme de construction des codes RCDO présenté à la Section 3.3.2.

Selon la définition d'un mot de code RCDO de taux codage $r = b/c$, $\mathbf{v} \cdot \mathbf{H}^{\mathrm{T}} = \mathbf{0}$, le système d'équations de contraintes peut être écrit comme suit :

$$\mathbf{v_t} H_0^T(t) + \mathbf{v_{t-1}} H_1^T(t) + \cdots + \mathbf{v_{t-m_s}} H_{m_s}^T(t) = \mathbf{0},\ t = \cdots,0,1,\cdots . \tag{3.6}$$

Nous remarquons d'après l'équation (3.6) que le présent symbole $\mathbf{v_t}$ dépend des m_s symboles précédemment calculés $\mathbf{v_{t-1}}, \ldots, \mathbf{v_{t-m_s}}$, puisque la sous-matrice $H_0^T(t)$ est de rang complet. Nous

rappelons que la sous-matrice de parité $H_i^{(v)}(t)$ définie dans (3.3) est une matrice d'identité et que le symbole systématique $\mathbf{v_t}$ est composé de b bits d'information $\mathbf{u_t}$ et $(c-b)$ bits de parité conformément à l'équation (3.4). À chaque unité de temps t, les $(c-b)$ bits de parité $v_t^{(i)}$, $b+1 \leq i \leq c$, sont générés en fonction des b bits d'information présents $\mathbf{u_t} = \left[u_t^{(1)}, u_t^{(2)}, \dots, u_t^{(b)}\right]$, les $b \cdot m_s$ anciens bits d'information $\left[\mathbf{u_{t-1}}, \dots, \mathbf{u_{t-m_s}}\right]$ et les $(c-b) \cdot m_s$ anciens bits de parité $\left[v_{t-1}^{(b+1)}, \dots, v_{t-1}^{(c)}, \dots, v_{t-m_s}^{(b+1)}, \dots, v_{t-m_s}^{(c)}\right]$. En remplaçant (3.3) et (3.4) dans (3.6), les équations d'encodage s'écrivent :

$$v_t^{(i)} = \begin{cases} u_t^{(i)}, & 1 \leq i \leq b \\ \sum_{j=1}^{b} \sum_{k=0}^{m_s} h_k^{(j,i-b)}(t) u_{t-k}^{(j)} + \sum_{j=b+1}^{c} \sum_{k=1}^{m_s} h_k^{(j,i-b)}(t) v_{t-k}^{(j)} & , b+1 \leq i \leq c \end{cases}. \quad (3.7)$$

où les opérations d'additions sont modulo-2 et définis dans $GF(2)$. Cependant, l'architecture de l'encodeur RCDO peut être construite en se basant sur les équations de codage présentées en fonction des positions de connexions $\alpha_{i,j}$. Pour cela, nous présentons le mot de code RCDO sous sa forme polynômiale $\mathbf{v}(D)$, $\mathbf{v}(D) = \left[v^{(1)}(D), \dots, v^{(c)}(D)\right]$, où la séquence binaire associée à la $i-$ième sortie du codeur convolutionnel s'écrit sous la forme suivante :

$$v^{(i)}(D) = \sum_t v_t^{(i)} D^t \ , \ v_t^{(i)} \in \{0,1\} \quad (3.8)$$

avec $v_t^{(i)}$ représentant le symbole binaire à la $i-$ième sortie du codeur convolutionnel à l'instant t. En utilisant la forme compacte de la matrice de contrôle de parité $\mathbf{H}^{\mathrm{T}}(D)$, présentée dans (3.1), la définition du mot de code $\mathbf{v}(D)$devient :

$$\mathbf{v}(D)\mathbf{H}^{\mathrm{T}}(D) = \mathbf{0}(D) \quad (39)$$

où $\mathbf{0}(D) = 0 + 0D + 0D^2 + \cdots$ représente le vecteur polynômes nuls. En effectuant la multiplication matricielle, chaque colonne j de l'équation (3.9) s'écrit :

$$\begin{aligned} c^j(D) &= \sum_t \sum_{i=1}^{c} v_t^{(i)} h_{i,j} D^{t+\alpha_{i,j}} = \mathbf{0}(D) \\ &= \sum_l \sum_{i=1}^{c} v_{l-\alpha_{i,j}}^{(i)} h_{i,j} D^l \quad = \sum_l 0 \cdot D^l \end{aligned} \quad (3.10)$$

avec $j \in \{1, \dots, (c-b)\}$. Il s'ensuit qu'à partir de l'équation (3.10), nous remarquons :

$$\begin{aligned} \textstyle\sum_{i=1}^{c} h_{i,j} v_{l-\alpha_{i,j}}^{(i)} &= 0 \\ h_{b+j,j} v_{l-\alpha_{b+j,j}}^{(b+j)} + \textstyle\sum_{\substack{i=1 \\ i\neq b+j}}^{c} h_{i,j} v_{l-\alpha_{i,j}}^{(i)} &= 0. \end{aligned} \tag{3.11}$$

Comme les éléments $h_{b+j,j}$ et $\alpha_{b+j,j}$ dans la matrice de contrôle $\mathbf{H}^{\mathrm{T}}(D)$ d'un code RCDO de taux de codage $r = b/c$ doivent prendre respectivement les valeurs 1 et 0, les équations d'encodage peuvent être déduites de (3.11) et présentées en fonction des positions de connexion $\{\alpha_{i,j}\}$ de la façon suivante :

$$v_t^{(i)} = \begin{cases} u_t^{(i)}, & 1 \le i \le b \\ \sum_{\substack{k=1 \\ k\neq i}}^{c} h_{k,i-b} v_{t-\alpha_{k,i-b}}^{(k)}, & b+1 \le i \le c. \end{cases} \tag{3.12}$$

La construction de l'encodeur RCDO repose totalement sur l'équation (3.12). Un exemple général d'un codeur RCDO de taux de codage $r = b/c$ est présenté à la Figure 3.4. Sur cette figure, l'encodeur peut être considéré comme une machine à états finis (en anglais *Finite State Machine*, FSM). À chaque instant t, la FSM génère à sa sortie c symboles binaires $v_t^{(1)}, v_t^{(2)}, \dots, v_t^{(c)}$, dont les b premiers symboles correspondent aux b bits d'information présents à l'entrée de l'encodeur et les $(c-b)$ derniers symboles correspondent aux $(c-b)$ bits de parité $v_t^{(i)}$, $b+1 \le i \le c$, calculés en fonction des entrées du FSM $u_t^{(1)}, u_t^{(2)}, \dots, u_t^{(b)}$et l'état de l'encodeur. D'après l'équation (3.12), l'état de l'encodeur est composé de l'ensemble des derniers $b \cdot m_s$ bits d'information ainsi que des derniers $(c-b) \cdot m_s$ bits de parité. Lorsque la mémoire du code m_s devient grande, le nombre d'états possible devient extrêmement élevé. Par conséquent, le décodage des codes RCDO ne peut pas être effectué à l'aide d'algorithmes de décodage optimaux comme le décodage Viterbi.

D'autre part, l'encodeur RCDO présenté à la Figure 3.4 partage beaucoup de similarités avec les encodeurs convolutionnels classiques. En effet, les encodeurs convolutionnels à temps-invariant comportent aussi un certain nombre d'éléments de délais déterminé par la mémoire du code. De plus, dans certaines structures d'encodeurs convolutionnels, citons l'encodeur récursif systématique convolutionnel, où les boucles de rétroactions (*feedback*) sont également présentes.

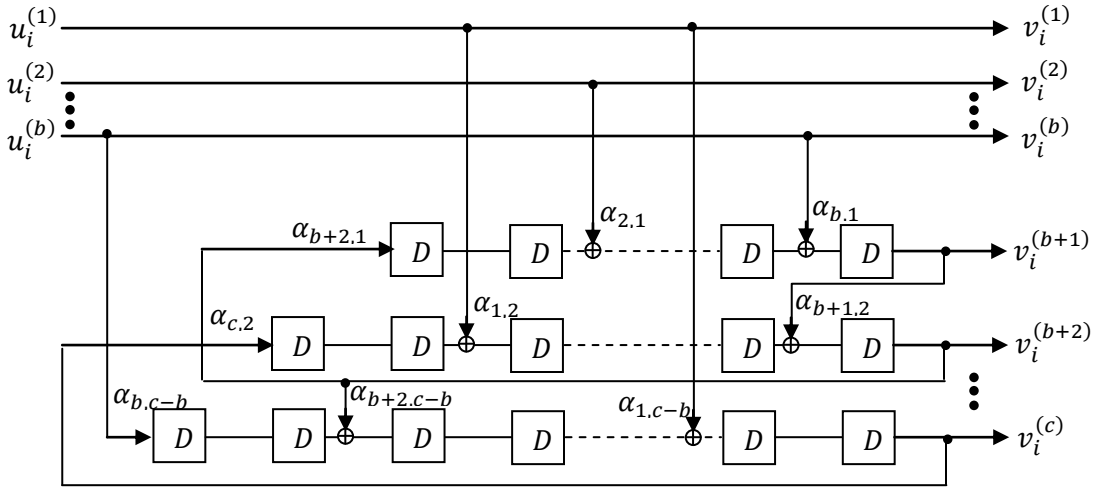


FIGURE 3-4 : Exemple d'un codeur RCDO composé de $(c-b)$ registres à décalage en parallèle et d'un taux de codage $r = b/c$.

Cependant, une différence fondamentale entre l'encodeur RCDO étudié dans ce mémoire et l'encodeur convolutionnel classique, est que les bits de parité formants le mot de code sont directement retournés pour former, avec les bits d'information, l'état de l'encodeur. Notons que la définition de la mémoire du code RCDO est aussi différente de celle des codes convolutionnels classiques, où la mémoire du code spécifie seulement le nombre des bits d'information précédents qui vont intervenir dans le calcul du présent bit de parité. La mémoire du code et la longueur de contrainte des codes RCDO sont définies à partir des positions de connexion $\{\alpha_{i,j}\}$ de l'encodeur, ceci en considérant les bits d'information précédents ainsi que les bits de parité.

3.4 Exemple d'un simple code RCDO

Dans cette section, nous considérons un exemple d'un simple code RCDO afin d'illustrer le processus d'encodage étudié dans ce mémoire. Le code utilisé est un code RCDO irrégulier à temps-invariant de taux de codage $r = 2/4$ et possède les caractéristiques suivantes : la mémoire du code $m_s = 2$, le degré des nœuds variables $d_\lambda = 1$ ou 2 et le degré des nœuds de contraintes $d_\rho = 3$. Ce code peut être généré à l'aide du codeur convolutionnel présenté à la Figure 3.5.

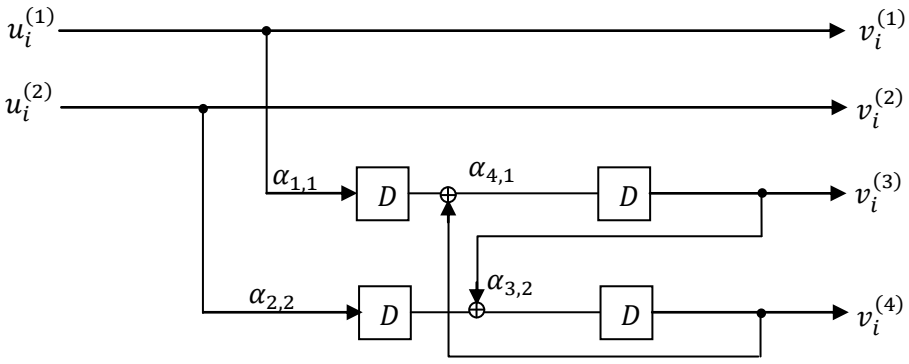


FIGURE 3-5 : Codeur RCDO composé de 2 registres à décalage ayant un taux de codage $r = 2/4$ et une mémoire $m_s = 2$.

Nous représentons la matrice de vérification $\mathbf{H}^{\mathrm{T}}(D)$ de dimension 4×2 associée à ce code par l'équation (3.13).

$$\mathbf{H}^T(D) = \begin{pmatrix} D^{\alpha_{1,1}} & 0 \\ 0 & D^{\alpha_{2,2}} \\ D^{\alpha_{3,1}} & D^{\alpha_{3,2}} \\ D^{\alpha_{4,1}} & D^{\alpha_{4,2}} \end{pmatrix} = \begin{pmatrix} D^2 & 0 \\ 0 & D^2 \\ 1 & D \\ D & 1 \end{pmatrix} \tag{3.13}$$

Selon l'équation (3.12), à chaque instant t, la partie systématique du code qui correspond aux deux bits binaires $v_t^{(i)}$, $1 \leq i \leq b$, s'écrit de la manière suivante :

$$v_t^{(1)} = u_t^{(1)}$$
$$v_t^{(2)} = u_t^{(2)}$$

Au même instant, le codeur génère aussi deux symboles de parité $v_t^{(i)}$, $3 \leq i \leq 4$, suivant les équations de parité suivantes :

$$v_t^{(3)} = v_{t-\alpha_{1,1}}^{(1)} + v_{t-\alpha_{4,1}}^{(4)} = v_{t-2}^{(1)} + v_{t-1}^{(4)} \tag{3.14}$$

$$v_t^{(4)} = v_{t-\alpha_{2,2}}^{(2)} + v_{t-\alpha_{3,2}}^{(3)} = v_{t-2}^{(2)} + v_{t-1}^{(3)} \tag{3.15}$$

$\mathbf{H}^{\mathrm{T}} =$

$v_0^{(3)}$	$v_0^{(4)}$	$v_1^{(3)}$	$v_1^{(4)}$	$v_2^{(3)}$	$v_2^{(4)}$	$v_3^{(3)}$	$v_3^{(4)}$	$v_4^{(3)}$	$v_4^{(4)}$	…
0	0	0	0	1	0					$v_0^{(1)}$
0	0	0	0	0	1					$v_0^{(2)}$
1	0	0	1	0	0					$v_0^{(3)}$
0	1	1	0	0	0					$v_0^{(4)}$
		0	0	0	0	1	0			$v_1^{(1)}$
		0	0	0	0	0	1			$v_1^{(2)}$
		1	0	0	1	0	0			$v_1^{(3)}$
		0	1	1	0	0	0			$v_1^{(4)}$
				0	0	0	0	1	0	$v_2^{(1)}$
				0	0	0	0	0	1	$v_2^{(2)}$
				1	0	0	1	0	0	$v_2^{(3)}$
				0	1	1	0	0	0	$v_2^{(4)}$
						0	0	0	0	$v_3^{(1)}$
						0	0	0		$v_3^{(2)}$
						1	0			$v_3^{(3)}$
						0	1			$v_3^{(4)}$
										⋮

FIGURE 3-6 : Matrice semi-infinie du code RCDO de taux de codage $r = 2/4$ et de mémoire $m_s = 2$.

À partir des équations de parité (3.14) et (3.15), nous présentons à la Figure 3.6 la forme binaire de la matrice de contrôle $\mathbf{H}^{\mathrm{T}}$. Cette matrice semi-infinie est équivalente à celle définie dans (3.1) puisque nous supposons que l'encodeur RCDO débute l'opération d'encodage à l'instant $t = 0$, c'est-à-dire que tous les registres à décalage constituant l'encodeur sont initialisés à l'état zéro et par conséquent tous les bits ayant un indice de temps $t < 0$ sont supposés être zéro. La mise en place d'un mécanisme de contrôle pour segmenter la trame d'information en un nombre de blocs de longueur fixe s'avère non nécessaire, puisque les codes RCDO peuvent être codés et décodés de manière continue.

3.5 Décodage itératif des codes RCDO

L'algorithme de décodage des codes RCDO est mathématiquement similaire à celui de l'algorithme de propagation de croyance utilisé dans le décodage des codes LDPC-B (voir Section 2.5). Cependant, l'architecture du décodeur RCDO est remarquablement différente de celle du décodeur LDPC-B, cela est due au fait que la dimension de la matrice de contrôle ainsi que la taille du graphe de Tanner sont indéfinies. L'objectif de cette section est de présenter et discuter de la structure du décodeur des codes RCDO.

3.5.1 Structure des décodeurs RCDO

Bien que la matrice de contrôle $\mathbf{H}^{\mathrm{T}}$ d'un code RCDO puisse être de longueur indéfinie, la mémoire du code et la longueur de contrainte sont finies, d'où le nombre fini d'éléments non nuls sur chaque ligne et chaque colonne. De plus, les bits de chaque symbole $\mathbf{v_t}$ dans un mot de code seront impliqués dans des équations de parité avec au maximum m_s anciens symboles : de $\mathbf{v_{t-m_s}}$ à $\mathbf{v_t}$. Ceci se traduit sur la représentation du graphe de Tanner, par le fait que chaque nœud de contrainte est connecté à un ensemble de nœuds variables dont le nombre maximal est $\vartheta = c \cdot (m_s + 1)$. En effet, cette propriété rend possible la construction d'un décodeur RCDO pratique. Pour cela, nous appliquons au graphe de Tanner une fenêtre coulissante de taille ϑ (i.e. couvrant ϑ bits du mot de code ou ϑ nœuds variables) dans laquelle l'algorithme à passage de messages est appliqué. Par ailleurs, plusieurs fenêtres peuvent être appliquées sur les différentes parties du graphe de Tanner afin d'échanger simultanément les messages. De cette façon, le décodage itératif des codes RCDO peut être effectué par une chaîne de fenêtres coulissantes (appelées *processeurs*). Il est important de noter que le nombre de processeurs correspond au nombre d'itérations effectuées. À la Figure 3.7, nous présentons le schéma d'opération du décodeur itératif envisagé pour réaliser le décodage des codes RCDO. Le terme "fenêtre coulissante" signifie que les fenêtres se déplacent tout au long du graphe de Tanner afin de mettre à jour les messages à travers les arêtes. D'autre part, le terme "processeur" implique que les messages se décalent dans l'ensemble des processeurs tout en étant mis à jour. Manifestement, les deux termes sont équivalents pour interpréter l'opération de décodage.

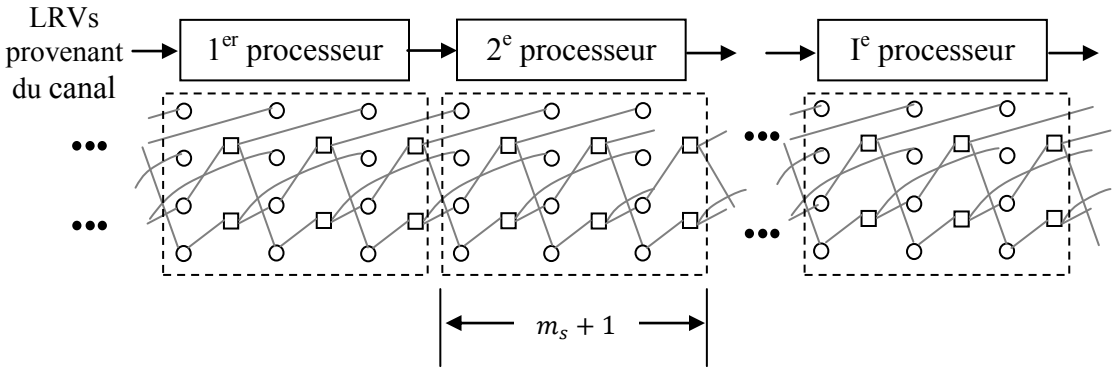


FIGURE 3-7 : Décodeur RCDO composé d'une chaîne de processeurs concaténés et agissant en fenêtres coulissantes tout au long du graphe de Tanner. Le graphe de Tanner est celui associé au code RCDO de taux de codage $r = 2/4$ présenté dans la Section 3.4. Les nœuds ronds sont des nœuds variables et les nœuds carrés sont les nœuds de contraintes.

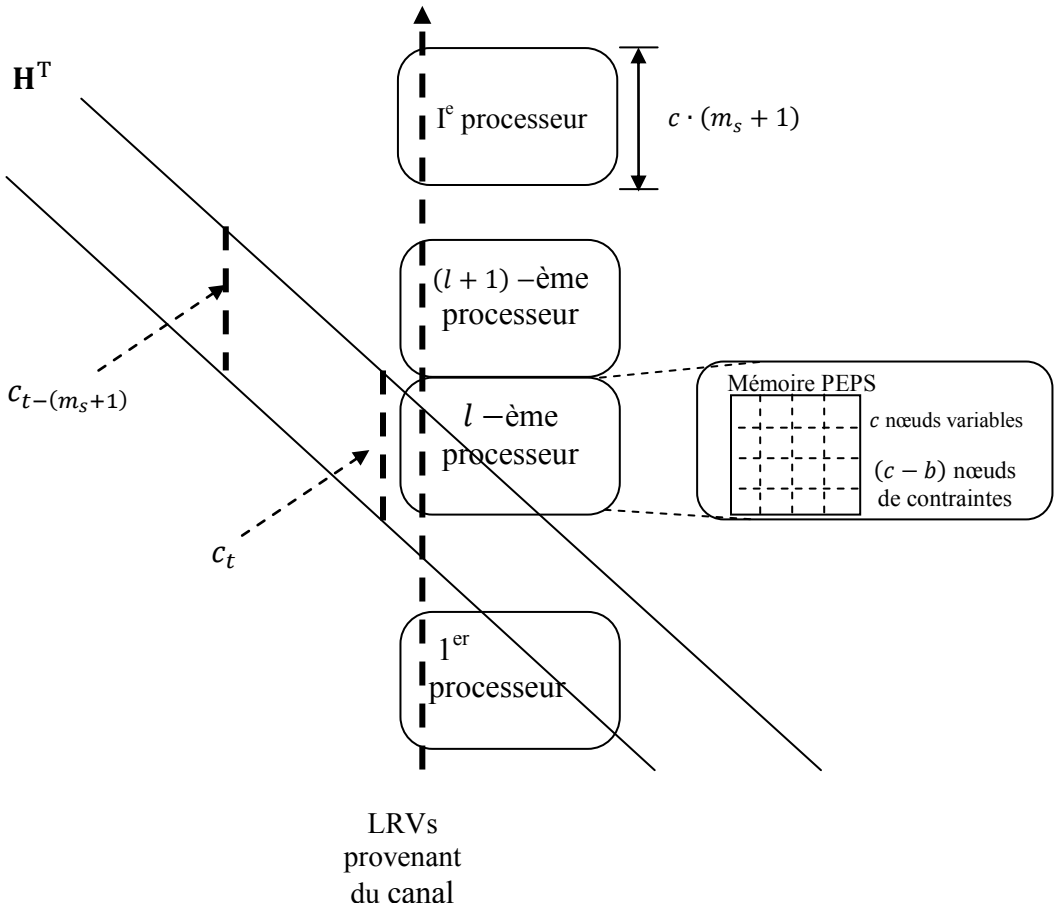


FIGURE 3-8 : Composition de la structure du décodeur itératif d'un code récursif convolutionnel doublement orthogonal ayant un taux de codage $r = b/c$ et une mémoire m_s.

La Figure 3.8 présente une autre illustration de la composition du décodeur avec sa matrice de contrôle de parité correspondante. Les ***I*** processeurs sont identiques et indépendants les uns des autres. Chaque processeur est composé des unités de traitement pour chaque nœud de contrainte et variable, et des unités de mémoire premier-entrée/premier-sortie (FIFO) pour stocker les messages intermédiaires. Lorsqu'un nœud de contrainte c_t est en cours de traitement à la $l-$ième itération par le processeur l, un autre nœud de contrainte $c_{t-(m_s+1)}$ peut être traité au même temps à la $(l+1)-$ième itération par le processeur $l+1$. Nous observons à la Figure 3.8 que lorsque la matrice de contrôle de dimension infinie se déplace le long des processeurs, les messages/LRVs correspondants traversent les processeurs. Il s'ensuit que grâce à la structure spéciale de la matrice de contrôle d'un code RCDO, i.e. une structure en diagonale, les itérations de décodage définies dans le temps seront assignées à des processeurs définis dans l'espace.

3.5.2 Opérations de décodage itératif des codes RCDO

Dans ce mémoire, nous considérons que les bits d'information codés par les codes RCDO de taux de codage $r = b/c$ sont transmis à travers un canal binaire symétrique à bruit additif blanc et gaussien (AWGN), où la sortie du canal est non quantifiée. Nous rappelons que l'énergie d'un bit est notée par E_b et que la densité spectrale bilatérale du bruit blanc est égale à $\frac{N_0}{2}$. À la sortie du canal, nous implémentons un récepteur optimum constitué de filtres adaptés aux signaux transmis. Par conséquent, le signal à la sortie du filtre adapté $y_t^{(i)}$, $1 \leq i \leq c$, est représenté par l'équation suivante [9] :

$$y_t^{(i)} = x_t^{(i)} + n_t^{(i)} \tag{3.16}$$

où $x_t^{(i)}$ indique le signal à la sortie du modulateur BPSK utilisé, $x_t^{(i)} \in \{-\sqrt{rE_b}, +\sqrt{rE_b}\}$ et $n_t^{(i)}$ représente une variable aléatoire de densité de probabilité gaussienne de moyenne nulle et de variance σ^2. Il s'ensuit que la valeur $y_t^{(i)}$ représente une variable aléatoire qui a une densité de probabilité P_y gaussienne de moyenne $x_t^{(i)}$ et de variance σ^2. Dans le cas où les signaux $x_t^{(i)}$sont transmis d'une façon équiprobable, les LRVs utilisés à l'entrée du décodeur itératif RCDO sont calculés comme suit [9] :

$$L\left(y_t^{(i)}\right) = \ln\left(\frac{P_y\left(x_t^{(i)}=+\sqrt{rE_b}|y_t^{(i)}\right)}{P_y\left(x_t^{(i)}=-\sqrt{rE_b}|y_t^{(i)}\right)}\right) = \frac{4rE_b}{N_0} y_t^{(i)} \tag{3.17}$$

Le code RCDO présenté à la Section 3.5 sera utilisé comme un exemple à des fins d'illustration. Nous présentons à la Figure 3.9 les opérations de décodage appliquées sur le graphe de Tanner du code à temps-invariant RCDO de taux de codage $r = 2/4$ et de mémoire $m_s = 2$. Le décodage itératif présenté se compose des quatre étapes suivantes.

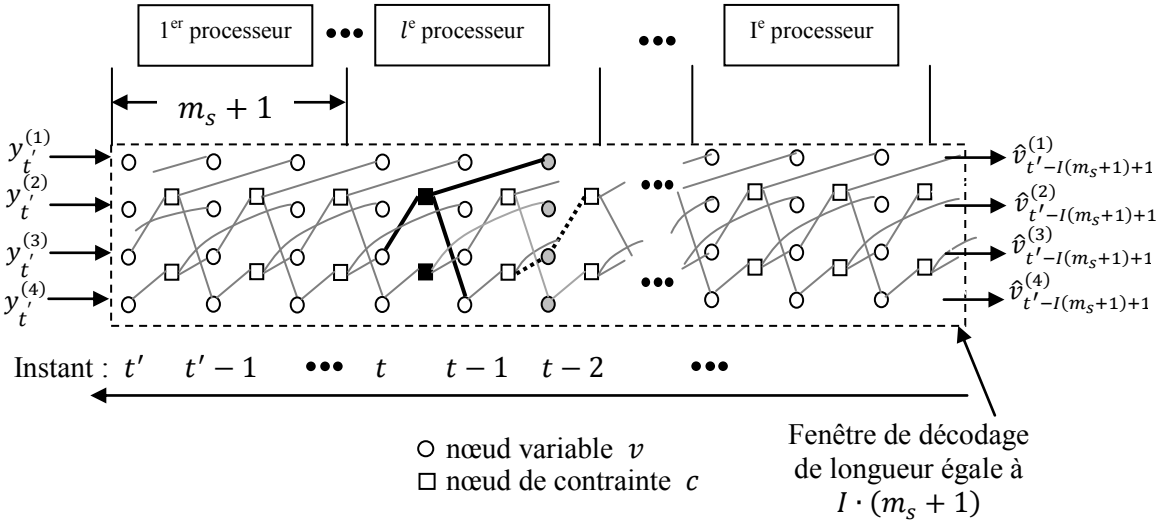


FIGURE 3-9 : Décodeur d'un code RCDO irrégulier. Chaque processeur traite une partie du graphe de Tanner.

1. Initialisation et calcul des entrées du décodeur

Au départ, nous commençons par initialiser les mémoires de stockage FIFO de chaque processeur l à des valeurs LRVs infinies. Ceci s'explique par le fait que l'encodeur RCDO débute à partir de l'état zéro, c'est-à-dire que les éléments de délais qui constituent les registres à décalage du codeur sont initialisés à zéro. En effet, l'assignation des valeurs infinies aux LRVs implique que les symboles qui leur correspondent sont des zéros déterministes. Il s'ensuit que tous les messages issus des nœuds variables v_t avant l'instant $t = 0$ sont des LRVs infinies.

$$\mathrm{m}_{v_t^{(i)}, c_{t+\alpha_{i,j}}^{(j)}}^{(l)} = \infty, \forall\, t < 0 \text{ et } 1 \leq l \leq I \tag{3.18}$$

Par la suite, le premier processeur $(l = 1)$ accepte à son entrée les valeurs LRVs associées aux nœuds variables provenant du canal. Au niveau du graphe de Tanner associé au code RCDO de taux de codage $r = 2/4$, nous avons à chaque instant $c = 4$ nœuds variables et $(c - b) = 2$

nœuds de contrainte qui sont injectés à l'entrée du processeur. Par conséquent, les messages issus des nœuds variables v_t du premier processeur sont initialisés aux valeurs LRVs suivantes :

$$\mathrm{m}^{(0)}_{v_t^{(i)},c_{t+\alpha_{i,j}}^{(j)}} = \mathrm{m}^{(0)}_{v_t^{(i)}} = L\left(y_t^{(i)}\right) \tag{3.19}$$

2. Mise à jour d'un nœud de contrainte :

Nous effectuons la mise à jour des nœuds de contraintes pour chaque processeur l, $1 \leq l \leq I$. À la Figure 3.9, nous montrons les nœuds de contraintes $c_t^{(j)}$, $1 \leq j \leq 2$, du $l-$ème processeur en cours de traitement. La mise à jour des nœuds de contraintes est effectuée à l'entrée de chaque processeur l. Ceci est possible puisque les messages provenant de tous les nœuds variables voisins sont déjà calculés à l'itération $(l-1)$. Plus précisément, pour $l > 1$, ces messages ont été obtenus par le processeur précédent $(l-1)$ puis transmis au présent processeur. Pour le premier processeur $(l = 1)$, ces messages ne sont que les LRVs provenant du canal et qui ont été également injectés dans le présent processeur. Comme nous l'avons vu à la Section 2.6, à chaque processeur l, le message transmis d'un nœud de contrainte $c_t^{(j)}$vers les nœuds variables qui lui sont connectés, peut être calculé selon l'équation (2.24) en utilisant la règle somme-produit. Cependant, la complexité associée à l'implémentation matérielle de cette règle est importante [39]. Dans ce mémoire, nous utilisons une approximation de la règle somme-produit, à savoir l'approximation min-somme [25] pour mettre à jour les nœuds de contraintes :

$$\mathrm{m}^{(l)}_{c_t^{(j)},v_{t-\alpha_{i,j}}^{(i)}} = sign\left(\prod_{k=\{1,\dots,c\}\setminus i} \mathrm{m}^{(l-1)}_{v_{t-\alpha_{k,j}}^{(k)},c_t^{(j)}}\right) \cdot \min_{k=\{1,\dots,c\}\setminus i}\left|\mathrm{m}^{(l-1)}_{v_{t-\alpha_{k,j}}^{(k)},c_t^{(j)}}\right| \tag{3.20}$$

où $\mathrm{m}^{(l)}_{c_t^{(j)},v_{t-\alpha_{i,j}}^{(i)}}$ représente le message échangé dans le processeur l à partir du nœud de contrainte actuel $c_t^{(j)}$ vers le nœud de variable $v_{t-\alpha_{i,j}}^{(i)}$ émis à l'instant $t-\alpha_{i,j}$. De plus, l'information extrinsèque $\mathrm{m}^{(l-1)}_{v_{t-\alpha_{k,j}}^{(k)},c_t^{(j)}}$ calculée par le processeur $(l-1)$, représente le message transmis du nœud variable émis à l'instant $t-\alpha_{k,j}$ vers le nœud de contrainte présent puis décalé vers le processeur l. Précisons que l'indice k peut avoir les valeurs $\{1,\dots,c\}$ à l'exclusion de l'indice i, d'où la notation $k = \{1,\dots,c\}\setminus i$. À la Figure 3.9, nous indiquons l'ensemble des nœuds variables dans le voisinage du nœud de contrainte $c_t^{(1)}$, $\mathcal{N}\left(c_t^{(1)}\right) = \left\{v_{t-\alpha_{k,1}}^{(k)} | k = \{1,\dots,c\}\right\} =$

$\left\{v_{t-2}^{(1)}, v_{t-1}^{(4)}, v_t^{(3)}\right\}$. Pour envoyer un message de $c_t^{(1)}$ vers $v_t^{(3)}$, les messages issus des nœuds variables $\mathcal{N}\left(c_t^{(1)}\right) \setminus v_t^{(3)} = \left\{v_{t-\alpha_{k,1}}^{(k)} | k = \{1, \dots, c\} \setminus 3\right\}$ calculés par le processeur précédent, $\mathrm{m}_{v_{t-2}^{(1)}, c_t^{(1)}}^{(l-1)}$ et $\mathrm{m}_{v_{t-1}^{(4)}, c_t^{(1)}}^{(l-1)}$, seront utilisés. Ainsi, la même opération est effectuée pour chaque nœud dans $\mathcal{N}\left(c_t^{(1)}\right)$.

Pour chaque processeur dans cette étape (à l'exception du processeur 1), nous notons que les messages utilisés proviennent du processeur qui le précède. En effet, les messages et les LRVs provenant du canal vont être décalés du processeur $(l-1)$ vers le processeur l, $l = 2, 3, \dots, I$. Prenons l'exemple du code RCDO de taux de codage $r = 2/4$ présenté à la Section 3.5, à chaque instant le processeur accepte 4 LRVs provenant du canal et 4 messages comme entrées et génère le même nombre de LRVs et de messages comme sorties.

3. Mise à jour d'un nœud variable :

Nous effectuons la mise à jour des nœuds variables pour chaque processeur l, $1 \leq l \leq I$. À la Figure 3.9, nous montrons les nœuds variables $v_{t-2}^{(i)}$, $1 \leq i \leq c$, du l −ème processeur en cours de traitement. La mise à jour des nœuds variables est effectuée à la sortie de chaque processeur puisque les messages provenant de tous les nœuds de contraintes voisins ont été obtenus à l'étape de la mise à jour d'un nœud de contrainte dans le même processeur l. La mise à jour d'un nœud variable s'effectue selon l'équation (3.21) [40] afin de calculer le message envoyé à partir de ce nœud variable vers tous ses voisins des nœuds de contraintes :

$$\mathrm{m}_{v_{t-m_s}^{(i)}, c_{t-m_s+\alpha_{i,j}}^{(j)}}^{(l)} = \mathrm{m}_{v_{t-m_s}^{(i)}}^{(0)} + \sum_{\substack{s=1 \\ s \neq j}}^{(\mathrm{c-b})} \mathrm{m}_{c_{t-m_s+\alpha_{i,s}}^{(s)}, v_{t-m_s}^{(i)}}^{(l)} \tag{3.21}$$

où $\mathrm{m}_{v_{t-m_s}^{(i)}, c_{t-m_s+\alpha_{i,j}}^{(j)}}^{(l)}$ exprime l'information extrinsèque échangée entre le nœud variable $v_{t-m_s}^{(i)}$ et le nœud de contrainte $c_{t-m_s+\alpha_{i,j}}^{(j)}$, $\mathrm{m}_{c_{t-m_s+\alpha_{i,s}}^{(s)}, v_{t-m_s}^{(i)}}^{(l)}$ représente le message envoyé vers le nœud variable $v_{t-m_s}^{(i)}$ depuis le nœud de contrainte $c_{t-m_s+\alpha_{i,s}}^{(s)}$ dans le même processeur l, et $\mathrm{m}_{v_{t-m_s}^{(i)}}^{(0)}$ est la valeur LRV du nœud variable $v_{t-m_s}^{(i)}$ provenant du canal. À la Figure 3.9, nous indiquons l'ensemble des nœuds de contraintes dans le voisinage du nœud variable $v_{t-2}^{(3)}$, $\mathcal{N}(v_{t-2}^{(3)}) =$

$\left\{c_{t-2+\alpha_{3,s}}^{(s)} | s = \{1, \ldots, (c-b)\}\right\} = \left\{c_{t-2}^{(1)}, c_{t-1}^{(2)}\right\}$. Par exemple, pour calculer $\mathrm{m}_{v_{t-2}^{(3)}, c_{t-1}^{(2)}}^{(l)}$ nous effectuons tout simplement la somme de $\mathrm{m}_{v_{t-m_s}^{(i)}}^{(0)}$ et $\mathrm{m}_{c_{t-2}^{(1)}, v_{t-2}^{(3)}}^{(l)}$. Nous remarquons que les équations (3.20) et (3.21) sont équivalentes à celles des équations de mises à jour des nœuds de contraintes et des nœuds variables pour un code LDPC-B introduit au chapitre 2. Ceci permet d'observer clairement l'équivalence mathématique entre le décodage des codes LDPC-B et celui des codes RCDO.

4. Décision finale

La décision finale est effectuée à la sortie du dernier processeur *I* sur la dernière information extrinsèque associée à chaque nœud variable :

$$\mathrm{m}_{v_{\mathrm{t}}^{(i)}}^{(I)} = \mathrm{m}_{v_{\mathrm{t}}^{(i)}}^{(0)} + \sum_{s=1}^{(\mathrm{c-b})} \mathrm{m}_{c_{t+\alpha_{i,s}}^{(s)}, v_t^{(i)}}^{(I)} \tag{3.22}$$

Si la valeur $\mathrm{m}_{v_{\mathrm{t}}^{(i)}}^{(I)}$ était supérieure ou égale à 0, le bit d'information correspondant est décodé par 0, sinon il est décodé par 1.

D'après la Figure 3.9 et les opérations de décodage présentées au dessus, nous pouvons facilement observer que la taille de la mémoire FIFO d'un seul processeur de décodage est déterminée par la longueur de contrainte $\vartheta = c \cdot (m_s + 1)$. Cette dernière correspond également au nombre de nœuds variables au sein d'un processeur. Il est important de rappeler que la définition de la longueur de contrainte des codes RCDO est différente de celle des codes convolutionnels classiques. En effet, la longueur de contrainte d'un code convolutionnel classique désigne la somme des longueurs de tous les registres à décalage constituant le codeur [9]. Cependant, pour les codes RCDO, la longueur de contrainte désigne la portée maximale non nulle en verticale ou en horizontale de la matrice de contrôle de parité $\mathbf{H}^{\mathrm{T}}$. Cette définition couvre les bits d'information et les bits de parité, tandis que la définition de la longueur de contrainte des codeurs non récursifs convolutionnels classiques couvre seulement les bits d'information. Cette différence peut être remarquée à partir des algorithmes de décodage. Pour le décodage des codes convolutionnels classiques, notamment l'algorithme de Viterbi, la complexité de décodage est liée au nombre des états de l'encodeur qui augmente de façon exponentielle avec la longueur de contrainte. Alors que pour le décodage des codes RCDO, la taille d'un processeur de décodage est déterminée par la longueur de la contrainte du code RCDO qui couvre l'ensemble des bits

d'information et des bits de parité. Pareillement, pour les codes LDPC-B, le même algorithme de décodage est utilisé et la taille du décodeur est déterminée par la longueur du bloc qui, par définition, contient l'ensemble des bits d'information et de parité. Il s'ensuit que la longueur de contrainte ainsi que la longueur du bloc doivent être prises en compte lors de la comparaison des performances des codes RCDO par rapport à celles des codes LDPC-B.

3.6 Conclusion

Dans ce chapitre, nous avons défini la matrice de contrôle de parité associée aux codes RCDO sous ces deux formes : la forme polynômiale $\mathbf{H}^T(D)$ et la forme binaire $\mathbf{H}^T$. À partir de cette matrice de contrôle, nous avons décrit la construction de l'encodeur convolutionnel récursif à temps-invariant RCDO composé de plusieurs registres à décalage, dans lequel la mémoire du code m_s correspond à la valeur du plus grand registre à décalage et les connexions reliant les symboles à la sortie de l'encodeur aux différents éléments de délai qui composent les registres à décalage ne varient pas dans le temps. Les codes RCDO sont construits selon une approche algébrique, où des conditions de double orthogonalité sont imposées sur les positions de connexions de l'encodeur afin d'éliminer les petits cycles, c'est-à-dire les cycles de longueurs 4, 6 et 8, dans le graphe de Tanner associé. Ceci permet, comme nous l'avons vu, d'effectuer l'encodage avec les codeurs RCDO en se basant sur la matrice de contrôle de parité. Nous avons également introduit la structure du décodeur itératif pour les codes RCDO qui est composée d'une chaîne de processeurs. Pour chaque processeur, nous avons discuté des opérations de décodage utilisées pour effectuer l'algorithme itératif à passage de messages.

CHAPITRE 4

TERMINAISON DES CODES CONVOLUTIONNELS RÉCURSIFS RCDO

Dans les chapitres précédents, la matrice de contrôle binaire $\mathbf{H}^T$ des codes RCDO est présentée sous une forme infinie. Dans les systèmes de communication basés sur une transmission par paquets, les bits de données sont organisés dans des trames de longueurs finies. Comme nous l'avons mentionné précédemment, les codes RCDO peuvent traiter des trames de longueurs arbitraires et variables, ce qui nécessite l'adaptation de la matrice de parité infinie sur des trames de longueurs finies. Cela nous amène à la problématique de l'initialisation et de la terminaison des codeurs RCDO. Bien que l'initialisation du codeur RCDO soit simple, la terminaison d'un tel codeur est beaucoup plus complexe et est le sujet principal de ce chapitre. Nous commençons par introduire la problématique de terminaison pour les codes convolutionnels en général. Ensuite, le problème de la terminaison des codes RCDO sera défini, discuté et résolu mathématiquement. De plus, nous définissons les conditions de terminaison qui doivent être imposées aux connexions du codeur afin de rendre les codes RCDO terminables. Par ailleurs, nous proposons un nouvel algorithme de recherche capable de construire une architecture efficace des encodeurs RCDO multi-registres terminables.

4.1 Terminaison des codes convolutionnels

La terminaison est un problème associé aux codes convolutionnels, mais non aux codes en blocs. En effet, la terminaison d'un code convolutionnel revient à reconduire son codeur à un état connu, généralement à l'état zéro. Les Figures 4.1 et 4.2 montrent deux types de codeurs convolutionnels. Le premier montré à la Figure 4.1 est défini par une matrice génératrice présentée sous sa forme polynômiale $[1 + D^2 + D^3, 1 + D + D^2 + D^3]$. Comme nous pouvons le constater, il s'agit d'un codeur en boucle ouverte (*feed-forward*) qui peut être ramené à l'état zéro simplement par l'injection d'une séquence de zéros à l'entrée. Notons que la longueur de cette séquence correspond à celle de la mémoire du code qui, dans cet exemple, est égale à 3. D'autre

part, le deuxième codeur présenté à la Figure 4.2 est défini par une matrice génératrice $\left[1\ ,\ \frac{1+D+D^2+D^3}{1+D^2+D^3}\right]$. Ce codeur peut être terminé par l'insertion d'une séquence particulière de bits de valeurs 1 et 0 à l'entrée. Les symboles qui composent cette séquence dépendent de la somme des bits continus dans le registre à décalage de la boucle de rétroaction (*feedback*). Que ce soit pour un codeur à boucle ouverte ou fermé, les bits de terminaison sont ajoutés au mot de code et la séquence de terminaison sera transmise à travers le canal, ce qui se traduit en une réduction du taux de codage effectif [9]. Il est important de mentionner que, plus la longueur de la séquence de la terminaison est grande plus la perte du taux de codage est élevée.

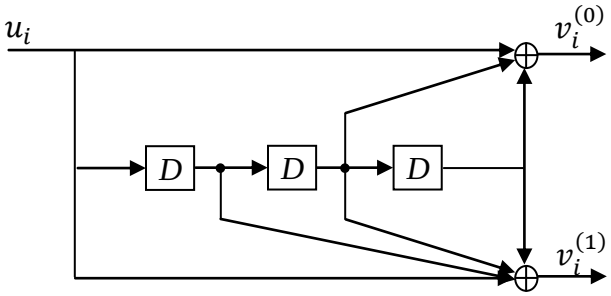


FIGURE 4-1 : Codeur non récursif d'un code convolutionnel $(13, 17)$.

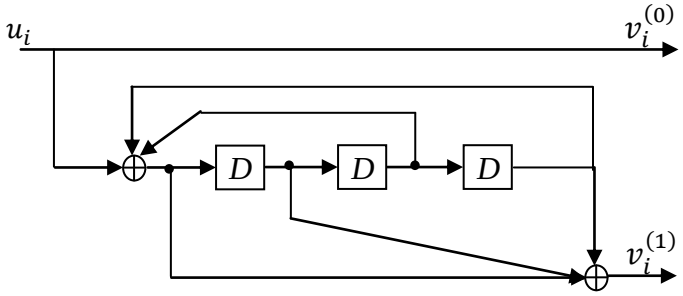


FIGURE 4-2 : Codeur récursif d'un code convolutionnel $(13, 17)$.

Dans la littérature, les codes convolutionnels circulaires (*tail-biting Codes*) peuvent être considérés comme une alternative à la terminaison [46]. En effet, le treillis du code convolutionnel circulaire est forcé de débuter à partir d'un état connu et de converger au même état. L'avantage de ces codes est qu'ils ne nécessitent pas de séquences de terminaison, d'où l'absence de réduction du taux de codage. Néanmoins, l'algorithme de décodage doit être modifié afin de tenir compte des spécificités des codes convolutionnels circulaires. Par exemple, dans

[47] les auteurs montrent que la complexité de décodage BCJR augmente avec le rapport w/n, où la valeur n correspond à la longueur de la trame et où le coefficient w, appelé *wrap depth*, représente une fraction de n. Pour les codes Turbo, où deux codeurs convolutionnels sont utilisés pour la génération des mots de code, il existe plusieurs manières de terminer le code, telles que l'application d'un intervalle de garde à la fin de la trame en cours de codage et la terminaison d'un seul codeur. De plus, notons que le concept des codes convolutionnels circulaires peut être utilisé dans les codes Turbo afin d'éviter la technique de terminaison. Précisons que la performance des différentes méthodes de terminaison des codes Turbo est étroitement liée à la conception et l'architecture de l'entrelaceur [48].

4.2 Adaptation des codes convolutionnels RCDO à des systèmes de communication basés sur une transmission par paquets

4.2.1 Application des codes RCDO à des trames de longueurs finies

L'un des avantages majeurs des codes convolutionnels RCDO est qu'ils peuvent coder et décoder des trames de longueurs arbitraires. Pour ce faire, la matrice de vérification de parité binaire $\mathbf{H}^T$ présentée au chapitre précédent doit être correctement tronquée afin de traiter des trames de données de longueurs finies. Supposons que la trame codée est de longueur $c \cdot n$, et que $\mathbf{v}_{[0,n-1]} = [\mathbf{v_0}, \dots, \mathbf{v_{n-1}}]$, où $\mathbf{v_i}$ est un sous-vecteur de longueur c et que la matrice de parité tronquée d'un code RCDO ayant une mémoire du code m_s soit $\mathbf{H}^T_{[0,n-1]}$. Nous pouvons alors écrire :

$$\mathbf{v}_{[0,n-1]} \cdot \mathbf{H}^T_{[0,n-1]} = \mathbf{0}. \tag{4.1}$$

En nous référant à l'équation (3.2) du chapitre précédent, la matrice de contrôle finie peut être présentée de la façon suivante :

$$\mathbf{H}^T_{[0,n-1]} = \begin{bmatrix} H_0^T(0) & \dots & H_{m_s}^T(m_s) & & & \\ & \ddots & \vdots & \ddots & & \\ 0 & & H_0^T(m_s) & \dots & \ddots & \\ & & & \ddots & & H_{m_s}^T(n-1) \\ & & & & \ddots & \vdots \\ & & & & & H_0^T(n-1) \end{bmatrix}. \tag{4.2}$$

Toutefois, avant d'aller plus loin dans ce chapitre, faisons l'hypothèse que le récepteur est capable de détecter correctement le début et la fin d'une trame codée ayant une taille variable. Ceci est vrai lorsque la synchronisation de trame est réussie. En effet, plusieurs systèmes de communication basés sur une transmission par paquets, tel que le standard Gigabit Éthernet, utilisent les marqueurs délimiteur-début-trame (DDT) et délimiteur-fin-trame (DFT) afin d'aider le récepteur à déterminer le début et la fin d'un paquet [49]. Dans ces systèmes, la détection correcte de ces marqueurs est simple. En effet, ces délimiteurs sont choisis parmi les symboles de la constellation de sorte qu'ils soient uniques et distincts de tous les autres mots de codes. Par conséquent, il est tout à fait légitime de supposer que les marqueurs DDT et DFT peuvent être correctement détectés, à la couche physique, sans décodage et que le début et la fin de la trame sont connus par le décodeur.

Comme nous l'avons mentionné au chapitre précédent, le codeur commence à coder la trame $\mathbf{v}_{[0,n-1]}$ à partir de l'état zéro, ce qui signifie que tous les bits d'information et de parité précédents sont supposés égaux à zéro : $\mathbf{v_t} = \mathbf{0}$ si $t < 0$. Nous appelons cette étape *préréglage* du codeur RCDO. Dans $\mathbf{v}_{[0,n-1]}$, les bits sont codés en fonction des bits précédents, ce qui implique que les m_s premiers symboles de la trame sont associés à des équations de contraintes qui comportent des bits zéro ayant des indices de temps négatifs. Du point de vue du décodeur, comme nous l'avons expliqué à la Section 3.5 du Chapitre 3, les mémoires FIFO des différents processeurs sont initialisées à l'infini, représentant ainsi les bits zéros déterministes avant l'instant $t = 0$. De cette façon, le décodeur est capable de décoder correctement les premiers bits d'une trame.

La question de recherche à laquelle nous tentons de répondre dans cette section est la suivante. Comment pouvons-nous clôturer l'encodage et le décodage d'une trame? En se référant à l'équation (4.1), l'encodage se termine simplement lorsque le dernier symbole $\mathbf{v_{n-1}}$ est généré. Par contre, du côté du décodeur, cette troncature pose des problèmes. En effet, après avoir injecté le dernier symbole codé dans le décodeur RCDO, certains LRVs provenant du canal doivent encore être introduits dans le premier processeur du décodeur afin de permettre les échanges de messages dans la dernière partie du graphe de Tanner défini par $\mathbf{H}^T_{[0,n-1]}$. Les LRVs infinis ne peuvent pas être utilisés comme des LRVs provenant du canal correspondant aux bits dans $\mathbf{v_t}$ $(t \geq n)$, car les bits de valeur zéro ne formulent pas des équations de contraintes valides avec les

bits de fin de la trame $\mathbf{v}_{[0,n-1]}$. Ceci peut s'expliquer à partir de l'encodeur présenté à la Figure 3.4 du Chapitre 3 par le raisonnement suivant : après la fin de la trame, même si tous les bits d'information envoyés à l'entrée du codeur sont égaux à zéros, les bits de parité correspondants peuvent ne pas avoir des valeurs zéros puisque l'encodage de la trame peut se terminer à n'importe quel état. Ainsi, le décodeur ignore les valeurs des LRVs provenant du canal qui correspondent aux bits après la fin de la trame afin de poursuivre le processus de décodage. Dans le cas où la terminaison n'était pas employée, nous attribuons à ces bits de parité des LRVs de valeur zéro, ce qui implique qu'ils sont totalement inconnus. Par conséquent, les bits à la fin de la trame subiront un taux d'erreur plus élevé.

Cette perte de performance peut être vue sous une autre perspective. Selon l'équation (4.1), la trame peut être considérée comme un mot de code LDPC-B dont la matrice de parité est définie par $\mathbf{H}_{[0,n-1]}^{T}$. À partir de l'équation (4.2), nous pouvons facilement constater que les dernières lignes de $\mathbf{H}_{[0,n-1]}^{T}$ possèdent des poids de Hamming faibles, c'est-à-dire que les bits de fin de la trame souffrent d'une faible protection contre les erreurs due à un faible contrôle de parité et que le degré du nœud variable correspondant peut être aussi faible que 1. Or, ces nœuds de faibles degrés peuvent entrainer une dégradation des performances d'erreur.

Pour remédier à ce problème, nous utilisons la technique de terminaison des codes. En fait, c'est en ajoutant une séquence de bits à la fin de la trame, appelée séquence de terminaison, que le codeur peut reconverger vers l'état zéro, ce qui implique que tous les bits d'information et de parité suivants sont égaux à zéro. Soit $\mathbf{v}_{[n,n+L-1]}$ la séquence de terminaison composée de L symboles, alors l'équation de contrainte suivante est satisfaite :

$$\left[\cdots, \mathbf{0}_{1\times c}, \mathbf{0}_{1\times c}, \mathbf{v}_{[0,n-1]}, \mathbf{v}_{[n,n+L-1]}, \mathbf{0}_{1\times c}, \mathbf{0}_{1\times c}, \cdots\right]\mathbf{H}^{T} = \mathbf{0}. \tag{4.3}$$

En comparant les deux équations (4.1) et (4.3), nous pouvons remarquer que la trame codée de longueur finie n'est qu'une partie d'un mot de code infini avec des bits zéros aux deux extrémités. Lorsque l'équation (4.3) est satisfaite, le décodeur peut attribuer à tous les LRVs suivants $\mathbf{v}_{[n,n+L-1]}$ la valeur infinie. Ces LRVs infinis vont pousser les derniers LRVs provenant du canal ainsi que les messages précédents à travers les différents processeurs du décodeur afin de décoder avec succès les bits de fin de la trame. Comme dans le cas des codes convolutionnels, la séquence de bits supplémentaires provoque une certaine perte de taux de codage. Cependant, la

recherche d'une telle séquence de terminaison pour un code RCDO s'avère être un problème non trivial à cause de la récursivité et de la structure multi-registres du codeur.

4.2.2 Définition de la problématique de la terminaison et solution proposée

Dans cette sous-section, nous considérons les codes systématiques convolutionnels RCDO de taux de codage $r = b/c$ pour définir la problématique de la terminaison. Supposons que la longueur d'une trame de données $\mathbf{u}_{[0,n-1]}$ soit de $n \cdot b$ bits. L'architecture du codeur présentée à la Figure 3.4 est utilisée pour générer la trame codée $\mathbf{v}_{[0,n-1]} = [\mathbf{v_0}, \dots, \mathbf{v_{n-1}}]$, où $\mathbf{v_i} = \left[v_i^{(1)}, v_i^{(2)}, \dots, v_i^{(c)}\right]$. La trame $\mathbf{v}_{[0,n-1]}$ est codée à partir de l'état zéro, c'est-à-dire que tous les éléments de délais des registres à décalage du codeur sont initialisés à zéro. À l'instant $t = n$, les b derniers bits de $\mathbf{u}_{[0,n-1]}$ ont été codés et les bits de la séquence de terminaison commencent à entrer dans le codeur. Bien que les bits de la terminaison ne soient pas nécessairement tous égaux à zéro, le codeur devrait atteindre l'état zéro en une durée finie, de préférence le plus rapidement possible afin de réduire la longueur de cette séquence de bits. À une certaine étape du processus de la terminaison, le codeur devrait atteindre un état qu'on appellera *état zéro-partiel*. Un état zéro-partiel est défini comme étant un état à partir duquel l'insertion des bits d'information zéro à l'entrée du codeur vont toujours générer à la sortie des bits de parité eux aussi égaux à zéro [53]. À partir de n'importe quel état zéro-partiel, le codeur a besoin au maximum de $m_s \cdot b$ bits d'entrée égaux à zéro pour être ramené à l'état zéro. Ces $m_s \cdot b$ bits égaux à zéro ainsi que les $m_s \cdot (c - b)$ bits de parité correspondants n'ont pas besoin d'être transmis puisque ce sont des zéros déterministes et que le décodeur peut tout simplement générer en interne des LRVs infinis pour les utiliser. De même pour le codeur, ces bits d'information zéros permettant la reconvergence vers l'état zéro n'ont, en fait, pas besoin d'être codés. Le codeur peut simplement réinitialiser tous ces registres à décalage à zéro une fois l'état zéro-partiel atteint. En nous basant sur ces faits et considérations, les $m_s \cdot b$ bits zéros ne seront pas pris en compte dans le calcul de la perte du taux de codage causé par la terminaison.

Supposons que la *séquence de terminaison* non codée $\mathbf{s}_{[0,L-1]} = [\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_{L-1}]$ soit de longueur $b \cdot L$, où $\mathbf{s}_i = \left(s_i^{(1)}, s_i^{(2)}, \dots, s_i^{(b)}\right)$, $0 \leq i < L$, et où $b \cdot L$ représente le nombre de bits non codés capables de reconduire le codeur vers l'état zéro-partiel. Ces bits sont complètement différents des bits d'information et seront appelés les *bits de terminaison*. Pendant ce temps, le codeur

génère à sa sortie les $(c-b)\cdot L$ bits de parité correspondant $\mathbf{p}_{[0,L-1]} = [\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{L-1}]$, où $\mathbf{p}_i = \left(p_i^{(1)}, p_i^{(2)}, \ldots, p_i^{(c-b)}\right)$ et $0 \leq i < L$. Ces bits seront appelés par la suite les *bits de parité de terminaison*. Ainsi, la séquence de terminaison codée peut être présentée par le vecteur $\mathbf{r}_{[0,L-1]} = [\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{L-1}]$, où $\mathbf{r}_i = \left[\mathbf{r}_i^{(0)}, \mathbf{r}_i^{(1)}\right], \mathbf{r}_i^{(0)} = \mathbf{s}_i, \mathbf{r}_i^{(1)} = \mathbf{p}_i$, et $0 \leq i < L$. À partir de la définition du code, la séquence de terminaison est trouvée correctement si et seulement si l'équation suivante est valide :

$$\left[\mathbf{v}_{1\times cn}, \mathbf{r}_{1\times cL}, \mathbf{0}_{1\times cm_s}\right]\mathbf{H}^{\boldsymbol{T}}_{c(n+L+m_s)\times(c-b)(n+L+m_s)} = \mathbf{0}_{1\times(c-b)(n+L+m_s)}, \tag{4.4}$$

où $\mathbf{H}^{\boldsymbol{T}}_{c(n+L+m_s)\times(c-b)(n+L+m_s)}$ représente la matrice de parité définie entre les instants $t=0$ et $t = n+L+m_s-1$. Dans l'équation (4.4), $\mathbf{r}_{1\times cL}$ désigne la séquence de terminaison codée qui permet de ramener le codeur à un état zéro-partiel et $\mathbf{0}_{1\times cm_s}$ représente la séquence de bits zéros qui reconduit le codeur d'un état zéro-partiel vers l'état zéro. Grâce à la forme diagonale de la matrice de contrôle de parité $\mathbf{H}^T$, l'équation (4.4) peut être réécrite en subdivisant la matrice de parité en des sous-matrices comme suit :

$$\left[\mathbf{v}_{1\times cn}, \mathbf{r}_{1\times cL}, \mathbf{0}_{1\times cm_s}\right]\begin{bmatrix}\mathbf{A}^{\boldsymbol{T}}_{cn\times(c-b)n} & \mathbf{B}^{\boldsymbol{T}}_{cn\times(c-b)(L+m_s)}\\ \mathbf{0}_{cL\times(c-b)n} & \mathbf{C}^{\boldsymbol{T}}_{cL\times(c-b)(L+m_s)}\\ \mathbf{0}_{cm_s\times(c-b)n} & \mathbf{D}^{\boldsymbol{T}}_{cm_s\times(c-b)(L+m_s)}\end{bmatrix} = \mathbf{0}_{1\times(c-b)(n+L+m_s)}. \tag{4.5}$$

En utilisant la définition du code, un mot de code valide implique que $\mathbf{v}_{1\times cn}\cdot\mathbf{A}^{\boldsymbol{T}}_{cn\times(c-b)n} = \mathbf{0}_{1\times(c-b)n}$. Ainsi, l'équation (4.5) devient :

$$\mathbf{r}_{1\times cL}\mathbf{C}^{\boldsymbol{T}}_{cL\times(c-b)(L+m_s)} = \mathbf{v}_{1\times cn}\mathbf{B}^{\boldsymbol{T}}_{cn\times(c-b)(L+m_s)}, \tag{4.6}$$

où les sous-matrices $\mathbf{C}^{\boldsymbol{T}}$ et $\mathbf{B}^{\boldsymbol{T}}$ ont été choisies de manière appropriée à partir de la matrice $\mathbf{H}^{\boldsymbol{T}}_{c(n+L+m_s)\times(c-b)(n+L+m_s)}$ et sont respectivement de dimensions $cL\times(c-b)(L+m_s)$ et $cn\times(c-b)(L+m_s)$. Il s'ensuit que la construction des deux sous-matrices $\mathbf{C}^{\boldsymbol{T}}$ et $\mathbf{B}^{\boldsymbol{T}}$ dépend de la longueur de la terminaison L. Par exemple, la sous-matrice $\mathbf{C}^{\boldsymbol{T}}$ commence à partir de la $\left((c-b)n+1\right)$—ème colonne et de la $(cn+1)$—ème ligne de $\mathbf{H}^{\boldsymbol{T}}_{c(n+L+m_s)\times(c-b)(n+L+m_s)}$. Le côté droite de l'équation (4.6) semble dépendre de toute la trame $\mathbf{v}$. Mais en réalité seuls les $c\cdot m_s$ derniers bits de $\mathbf{v}$ définissant l'état final du codeur sont significatifs puisque seules les

$c \cdot m_s$ dernières lignes de $\mathbf{B}^{\boldsymbol{T}}$ sont non nulles. Soit $\mathbf{w}_{[0,m_s-1]} = \left[\mathbf{v}_{n-m_s}, \ldots, \mathbf{v}_{n-1}\right]$ le vecteur définissant l'état final du codeur, alors

$$\mathbf{v}_{1 \times cn}\mathbf{B}^{\boldsymbol{T}}_{cn \times (c-b)(L+m_s)} = \mathbf{w}_{1 \times cm_s}\mathbf{K}^{\boldsymbol{T}}_{cm_s \times (c-b)(L+m_s)}, \tag{4.7}$$

où $\mathbf{K}^{\boldsymbol{T}}$ est la sous-matrice qui comporte les $c \cdot m_s$ dernières lignes de $\mathbf{B}^{\boldsymbol{T}}$.

Nous pouvons facilement observer que la séquence de terminaison $\mathbf{r}$ peut être obtenue en résolvant l'équation (4.6). Ainsi, nous pouvons présenter le système d'équations linéaires sous une forme d'un vecteur colonne en introduisant (4.7) dans la transposée de (4.6) :

$$\mathbf{C}_{(c-b)(L+m_s) \times cL}\mathbf{r}^{T}_{cL \times 1} = \mathbf{K}_{(c-b)(L+m_s) \times cm_s}\mathbf{w}^{\boldsymbol{T}}_{cm_s \times 1}, \tag{4.8}$$

où $\mathbf{C} = (\mathbf{C}^{\boldsymbol{T}})^{\boldsymbol{T}}$ et $\mathbf{K} = (\mathbf{K}^{\boldsymbol{T}})^{\boldsymbol{T}}$ sont deux sous-matrices qui dépendent de la longueur de la terminaison L. La matrice du système d'équations linéaires présentée par $\mathbf{C}$ est de dimension $(c-b)(L+m_s) \times cL$. En nous référant à [50], l'équation (4.8) peut avoir une ou plusieurs solutions, si le nombre d'inconnues est plus grand que celui des équations du système, c'est-à-dire que $cL \geq (c-b)(L+m_s)$, et la matrice $\mathbf{C}$ possède un rang complet $rang(\mathbf{C}) = (c-b)(L+m_s)$. Supposons qu'il existe un L pour lequel l'équation (4.8) possède au moins une solution, alors

$$\mathbf{r}^{T}_{cL \times 1} = \mathbf{C}^{-1}_{cL \times (c-b)(L+m_s)}\,\mathbf{K}_{(c-b)(L+m_s) \times cm_s}\mathbf{w}^{\boldsymbol{T}}_{cm_s \times 1} = \mathbf{T}_{cL \times cm_s}\mathbf{w}^{\boldsymbol{T}}_{cm_s \times 1}. \tag{4.9}$$

Il est clair à présent que la séquence de la terminaison est déterminée en fonction du vecteur de l'état final du codeur $\mathbf{w}^{\boldsymbol{T}}$ et de la matrice $\mathbf{T} = \mathbf{C}^{-1} \cdot \mathbf{K}$ appelée *matrice de terminaison* où sa dimension dépend également de L.

4.2.3 Terminaison d'un simple code RCDO

Dans cette sous-section, nous reprenons l'exemple du code RCDO présenté à la Section 3.4 du Chapitre 3 afin d'illustrer la procédure de la terminaison des codeurs RCDO. Nous rappelons les paramètres de ce code RCDO irrégulier : taux de codage $r = 2/4$, mémoire du code $m_s = 2$, nombre d'éléments non nuls sur chaque ligne de $\mathbf{H}^T$, $d_\lambda = 1$ ou 2 et nombre d'éléments égaux à 1 sur chaque colonne de $\mathbf{H}^T$, $d_\rho = 3$.

Considérons maintenant la terminaison du code décrit ci-dessus. Nous initialisons d'abord la longueur de la terminaison L à $\left\lceil \frac{c-b}{b} m_s \right\rceil = 2$, c'est-à-dire à la valeur minimale satisfaisant la

condition $cL \geq (c-b)(L+m_s)$. Ensuite, nous sélectionnons les sous-matrices $\mathbf{C}_{8\times8}^{\boldsymbol{T}}$ et $\mathbf{K}_{8\times8}^{\boldsymbol{T}}$ afin de calculer la matrice de terminaison $\mathbf{T}_{8\times8}$. À partir de l'équation (4.5) et de la Figure 3.6 du chapitre précédent, nous pouvons trouver ces deux sous-matrices comme illustré à la Figure 4.3.

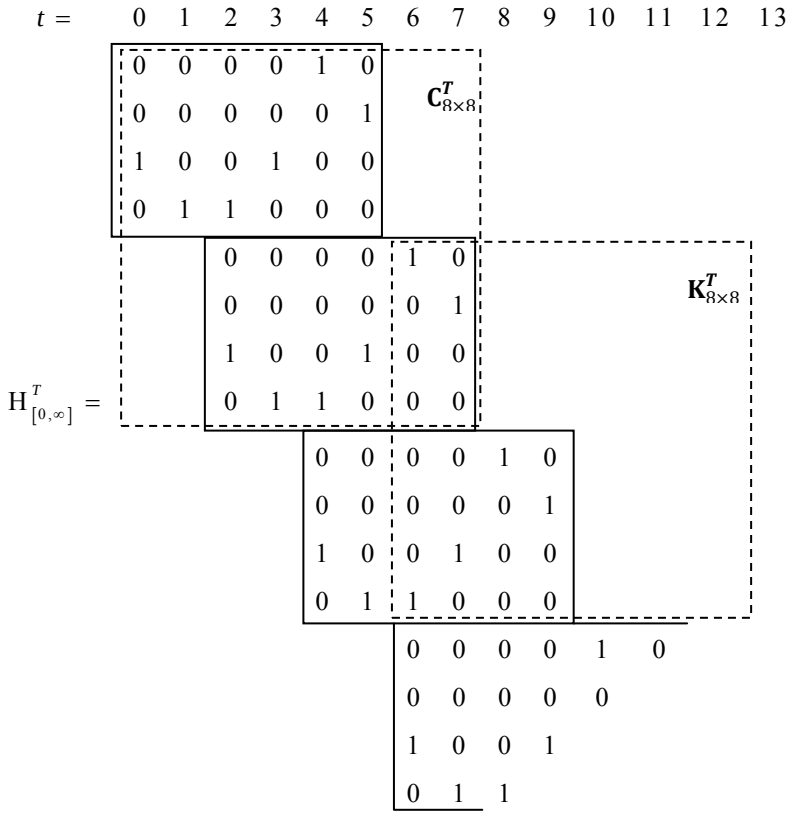


FIGURE 4-3 : Exemple montrant comment sélectionner les deux sous-matrices $\mathbf{C}^{\boldsymbol{T}}$ et $\mathbf{K}^{\boldsymbol{T}}$ à partir de la matrice de parité du code RCDO de taux de codage $r = 2/4$ présenté à la Section 3.4. Dans ce cas, $L = 2$. Notons que dans l'équation (4.5), $\mathbf{C}_{8\times8}^{\boldsymbol{T}}$ est située au dessous de $\mathbf{K}_{8\times8}^{\boldsymbol{T}}$ (une partie de $\mathbf{B}^{\boldsymbol{T}}$). Ici, nous localisons tout simplement $\mathbf{C}_{8\times8}^{\boldsymbol{T}}$ à partir de l'instant $t = 0$ grâce à la périodicité de la matrice.

Dans ce cas, la sous-matrice $\mathbf{C}_{8\times8}$ est une matrice carrée et elle possède un rang complet. Par conséquent, elle a une unique inverse $\mathbf{C}_{8\times8}^{-\mathbf{1}}$. Selon l'équation (4.9), nous aurons la matrice de terminaison suivante pour $L = 2$:

$$\mathbf{T}_{8\times 8} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{4.10}$$

Après avoir trouvé la matrice $\mathbf{T}_{8\times 8}$, la séquence de terminaison peut être obtenue en multipliant la matrice de terminaison $\mathbf{T}_{8\times 8}$ par le vecteur colonne $\mathbf{w}^{\boldsymbol{T}}_{cm_s \times 1}$ contenant les $c \cdot m_s$ derniers bits de la trame codée.

La procédure de recherche d'une séquence de terminaison est mathématiquement simple, mais cependant, peut générer quelques problèmes indésirables. Une difficulté provient de la nature récursive du codeur. Pour cet exemple, la terminaison est possible pour $L = \left\lceil \frac{c-b}{b} m_s \right\rceil = 2$. Toutefois, pour la plupart des codes RCDO proposés dans [27] et [19], l'équation (4.8) peut n'avoir aucune solution puisque la sous-matrice $\mathbf{C}_{(c-b)(L+m_s)\times cL}$ n'est pas de rang complet pour un L initié à la valeur $\left\lceil \frac{c-b}{b} m_s \right\rceil$. De ce fait, la longueur de la terminaison doit être augmentée afin de rendre possible la procédure de terminaison de l'encodeur. En incrémentant L par 1, un degré de liberté supplémentaire est introduit et les contraintes définies dans (4.8) sont relaxées. Nous pouvons continuer à incrémenter la valeur de L jusqu'à ce que la sous-matrice $\mathbf{C}_{(c-b)(L+m_s)\times cL}$ soit de rang complet et qu'une ou plusieurs solutions existent.

4.3 Terminaison des codes RCDO

4.3.1 Processus de la terminaison d'une trame de longueur finie

Un des objectifs de ce mémoire est de construire des codeurs RCDO capables d'être terminés avec des séquences de bits ayant des longueurs les plus petites possibles car ceci permettrait de réduire la perte de taux de codage causée par la séquence de terminaison. Dans cette section, nous présentons la technique de terminaison des codeurs RCDO ayant un taux de codage $r = b/c$ et une mémoire du code m_s. Nous nous concentrons plus particulièrement sur une technique qui permet de générer des séquences de terminaison de longueurs qui ne dépassent pas les $b\left(\left\lceil \frac{c-b}{b} m_s \right\rceil + 1\right)$ bits.

Un aperçu du processus de la terminaison est présenté à la Figure 4.4. Une fois la trame de donnée codée, les $b \cdot \left(\left\lceil \frac{c-b}{b} m_s \right\rceil + 1\right)$ bits de la séquence de terminaison sont ajoutés dans le but de forcer le retour du codeur vers l'état zéro-partiel. Rappelons qu'un état zéro-partiel est défini comme étant un état à partir duquel au maximum $b \cdot m_s$ bits zéros sont requis pour faire reconverger le codeur à l'état zéro. Ensuite, nous avons deux choix à effectuer afin de réinitialiser le codeur à l'état initial, soit nous injectons $b \cdot m_s$ bits zéros supplémentaires dans le codeur, soit nous transmettons un marqueur délimiteur-fin-trame DFT [49] [51].

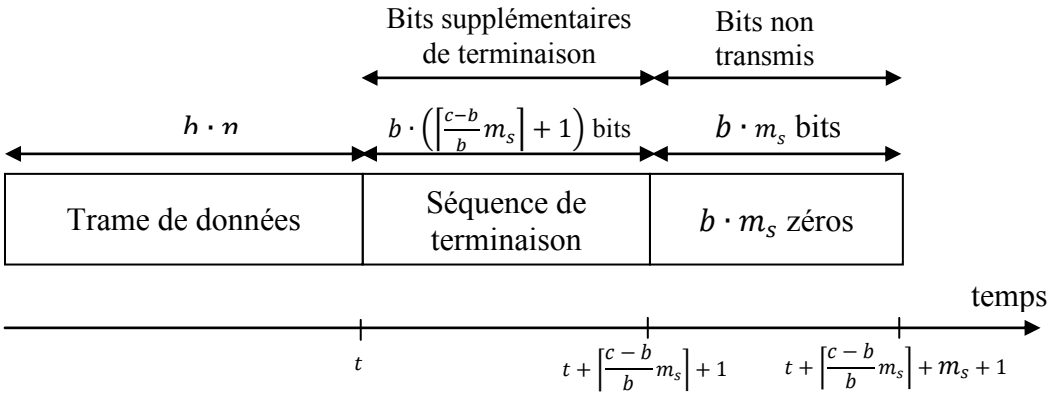


FIGURE 4-4 : Processus de terminaison. Les $b \cdot \left(\left\lceil \frac{c-b}{b} m_s \right\rceil + 1\right)$ bits de la séquence de terminaison sont ajoutés à la fin de la trame de données. Les $b \cdot m_s$ bits zéros supplémentaires sont nécessaires pour réinitialiser le codeur à l'état zéro.

Dans ce mémoire, nous optons pour le deuxième choix, où la technique de terminaison proposée considère que le délimiteur DFT a été décodé correctement et que la réinitialisation du codeur se fait d'une manière automatique sans avoir besoin de transmettre des bits zéro supplémentaires.

4.3.2 Génération de la séquence de la terminaison

La génération de la séquence de terminaison adéquate pour terminer un code RCDO de taux de codage $r = b/c$ se fait en trois étapes. En nous référant au système d'équations linéaires (4.9), la première étape consiste à trouver la longueur correcte de la séquence de terminaison L. Comme nous l'avons vu à la Section 4.2.2, la longueur de la terminaison L doit être suffisamment grande

pour rendre l'équation (4.8) solvable. La valeur minimale possible de L est égale à $\left\lceil \frac{c-b}{b} m_s \right\rceil$. À cette fin, un algorithme de recherche pour déterminer la longueur de terminaison L est proposé :

Algorithme de recherche de la longueur de terminaison L pour un code RCDO ayant un taux de codage $r = b/c$ et une mémoire m_s.
1. Initialiser la longueur de terminaison L : $L = \left\lceil \frac{c-b}{b} m_s \right\rceil$. 2. Sélectionner la sous-matrice $\mathbf{C}^{\boldsymbol{T}}_{cL \times (c-b)(L+m_s)}$. 3. Vérifier le rang de la sous-matrice $\mathbf{C}$. Si $\mathbf{C}$ possède un rang égal à $(c-b)(L+m_s)$, c'est-à-dire que $\mathbf{C}$ est de rang complet, arrêter la recherche et choisir la valeur courante de L, sinon aller à l'Étape 4. 4. Incrémenter L : $L = L + 1$ et aller à l'Étape 2.

La seconde étape nécessaire à la génération de la séquence de terminaison consiste à calculer la matrice de terminaison $\mathbf{T}_{cL \times cm_s}$. Nous rappelons que la matrice $\mathbf{C}$ du système d'équations linéaires binaires (4.8) est de dimension $(c-b)(L+m_s) \times cL$, ainsi le nombre d'équations de contraintes $(c-b)(L+m_s)$ est inférieur au nombre des inconnues cL. Lorsque le système possède au moins une solution, la sous-matrice $\mathbf{C}$ doit être de rang complet, c'est-à-dire que son rang doit être égal à $f = rang(\mathbf{C}) = (c-b)(L+m_s)$. Nous utilisons la technique de l'élimination de Gauss-Jordan pour résoudre (4.8). Comme nous connaissons le rang de la sous-matrice $\mathbf{C}$, l'application de l'élimination de Gauss-Jordan nous permettra d'obtenir une forme échelonnée réduite de $\mathbf{C}$ composée de f colonnes représentant les inconnues de base et $cL - f$ colonnes correspondant aux inconnues libres. Par conséquent, le système (4.8) admet au total $2^{cL-f} = 2^{bL-(c-b)m_s}$ solutions [50].

Généralement, la sous-matrice $\mathbf{C}_{(c-b)(L+m_s) \times cL}$ est une matrice non carrée, sa matrice inverse dans (4.9) est en fait obtenue en appliquant l'élimination de Gauss-Jordan. Ainsi, pour une matrice $\mathbf{C}_{(c-b)(L+m_s) \times cL}$ donnée, nous appliquons la transformation de Gauss-Jordan sur la matrice bordée suivante :

$$[\mathbf{C} \mid \mathbf{I}] \xrightarrow{\text{Transformation de Gauss-Jordan}} [\mathbf{Tr_C} \mid \mathbf{Tr_I}], \tag{4.11}$$

où $\mathbf{I}$ est une matrice d'identité de dimension $(c-b)(L+m_s) \times (c-b)(L+m_s)$, $\mathbf{Tr_C}$ est la forme diagonale de $\mathbf{C}$ après la transformée de Gauss-Jordan, et $\mathbf{Tr_I}$ est l'élément à droite de la matrice bordée après la transformation de Gauss-Jordan. Lorsque $\mathbf{C}$ est une matrice carrée, c'est-à-dire que $L = \frac{c-b}{b} m_s$, alors la matrice $\mathbf{Tr_C}$ est une identité et la matrice $\mathbf{Tr_I}$ représente l'unique inverse de $\mathbf{C}$. Lorsque $L > \frac{c-b}{b} m_s$, la matrice $\mathbf{Tr_C}$ comporte $bL - (c-b)m_s$ colonnes correspondantes aux inconnues libres. Ainsi, la matrice échelonnée réduite $\mathbf{Tr_C}$ peut être représentée sous la forme suivante :

$$\mathbf{Tr_C} = \begin{bmatrix} 1 & 0 & \cdots & 0 & * & 0 & * & 0 \\ 0 & 1 & \cdots & 0 & * & 0 & * & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & * & 0 & * & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & * & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.12}$$

où les éléments égaux à 1 représentent les pivots de Gauss et correspondent aux inconnues de base, tandis que les colonnes comportant des étoiles '*' correspondent aux inconnues libres [50]. Notons que les inconnues libres peuvent être assignées soit à la valeur binaire 0, soit à la valeur binaire 1. Afin de faciliter la transformation de Gauss-Jordan, nous attribuons la valeur binaire 0 à toutes les inconnues libres, et ce, en insérant $bL - (c-b)m_s$ lignes de zéros aux emplacements correspondants dans la matrice $\mathbf{Tr_I}$. Par conséquent, nous obtiendrons la matrice inverse de $\mathbf{C}$, $\mathbf{C^{-1}} = \mathbf{Tr_I}$. Ensuite, nous sélectionnons également la sous-matrice $\mathbf{K}_{(c-b)(L+m_s) \times cm_s}$ à partir de la matrice de contrôle de parité $\mathbf{H}^T$. En se référant à l'équation (4.9), nous obtiendrons la matrice de terminaison de dimension $cL \times cm_s$ en multipliant la matrice inverse $\mathbf{C}^{-1}$ par la matrice $\mathbf{K}$, $\mathbf{T}_{cL \times cm_s} = \mathbf{C}^{-1}_{cL \times (c-b)(L+m_s)} \mathbf{K}_{(c-b)(L+m_s) \times cm_s}$.

Après avoir obtenu la longueur correcte de la séquence de terminaison L, trouvé la matrice inverse de $\mathbf{C}$ et calculé la matrice de terminaison $\mathbf{T}$, la dernière étape consiste à générer les bits de la séquence de terminaison $\mathbf{r}$. Cette étape va s'effectuer de façon progressive. Nous commençons tout d'abord par calculer les b premiers bits de terminaison $\mathbf{r}_0^{(0)} = \mathbf{s}_0$ à partir de l'équation (4.9)

en multipliant le vecteur d'état final du codeur défini à l'instant n, $\mathbf{w}_{cm_s \text{ x } 1}^{(n)} = \left[\mathbf{v_{n-m_s}}, \dots, \mathbf{v_{n-1}}\right] = \left[v_{n-m_s}^{(1)}, \dots, v_{n-m_s}^{(c)}, \dots, v_{n-1}^{(1)}, \dots, v_{n-1}^{(c)}\right]$ par les b premières lignes de la matrice $\mathbf{T}_{cL \times cm_s}$. Ces bits de terminaison non codés sont ensuite envoyés à l'entrée du codeur. Une fois les $(c-b)$ bits de parité de terminaison correspondants $\mathbf{r}_0^{(1)} = \mathbf{p}_0$ générés à la sortie de l'encodeur et l'état du codeur mis à jour, les b seconds bits de terminaison $\mathbf{r}_1^{(0)} = \mathbf{s}_1$ sont également obtenus à partir de l'équation (4.9) en multipliant le vecteur d'état final mis à jour $\mathbf{w}_{cm_s \text{ x } 1}^{(n+1)} = \left[\mathbf{v_{n-m_s+1}}, \dots, \mathbf{v_{n-1}}, \mathbf{r}_0\right] = \left[v_{n-m_s+1}^{(1)}, \dots, v_{n-m_s+1}^{(c)}, \dots, v_{n-1}^{(1)}, \dots, v_{n-1}^{(c)}, \mathbf{s}_0, \mathbf{p}_0\right]$ par les mêmes b premières lignes de la matrice $\mathbf{T}_{cL \times cm_s}$. Remarquons que seules les b premières lignes de la matrice de terminaison $\mathbf{T}_{cL \times cm_s}$ doivent être sauvegardées afin d'implémenter les codes RCDO terminés. Cette approche génère les bits de terminaison graduellement et en continue jusqu'à ce que la longueur de terminaison non codée $b \cdot L$ soit atteinte. Par conséquent, aucun stockage supplémentaire n'est nécessaire pour sauvegarder le vecteur d'état du codeur à la fin de la trame.

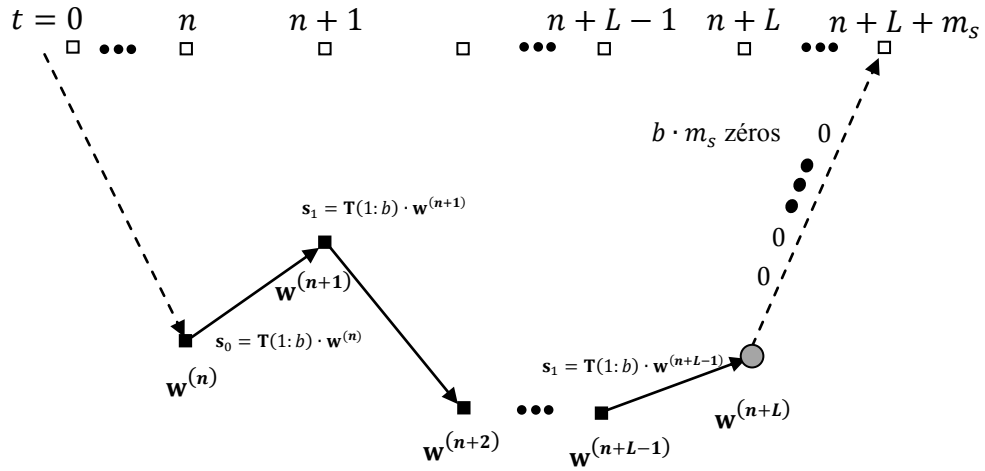


FIGURE 4-5 : Chemin de la terminaison à travers le treillis du code. Les carrés blancs représentent les états zéro, les cercles gris représentent les états zéro-partiel et les carrés noirs représentent des états arbitraires.

Du point de vue d'encodage, un code RCDO peut être considéré comme étant un code en treillis. En effet, le treillis commence à partir de l'état zéro et le processus de la terminaison est supposé le ramener à un état zéro-partiel. Nous pouvons voir que chaque b bits de terminaison envoyés au

codeur produisent une transition d'état. Après chaque changement d'état, les b prochains bits de terminaison sont recalculés en fonction de l'état mis à jour. Ainsi, le processus de la terminaison $\mathbf{r}^T = \mathbf{T} \cdot \mathbf{w}^T$ constitue un chemin à travers le treillis en commençant à partir l'état $\mathbf{w}$ et finissant à un état zéro-partiel, comme illustré à la Figure 4.5.

Notons que le processus de recherche d'une séquence de terminaison d'un codeur RCDO ayant un taux de codage $r = b/c$ est relativement simple, par ailleurs, avant d'exécuter les trois étapes décrites ci-dessus, nous devons nous assurer que la sous-matrice $\mathbf{C}$ de dimension $(c-b)(L+m_s) \times cL$ définie dans (4.8) est inversible, c'est-à-dire que $\mathbf{C}$ est de rang complet, et ce, pour un L comparable à $\frac{c-b}{b} m_s$. Une difficulté de cette approche provient de la construction des matrices de contrôle de parité dont les sous-matrices $\mathbf{C}$ sont inversibles. Malheureusement, les codes RCDO précédemment construits dans [27] et [19] n'ont pas les propriétés nécessaires pour être terminés par une séquence de bits de longueur finie. Dans la section suivante, nous définissons des conditions supplémentaires permettant de construire des codes convolutionnels récursifs multi-registres RCDO terminables.

4.3.3 Complexité associée à la génération de la séquence de terminaison

Grâce à la faible densité de la matrice de contrôle de parité $\mathbf{H}^T$, l'encodage des codes RCDO est effectué avec une complexité linéaire. Pour une trame de longueur N_f, la latence d'encodage est de l'ordre $\mathcal{O}(N_f)$ éléments de délai. Pour un code en blocs LDPC de taille N, la complexité associée à l'encodage d'un mot de code peut atteindre $\mathcal{O}(N^2)$ opérations [58]. Le problème de la complexité d'encodage des codes en blocs LDPC a largement été étudié par plusieurs chercheurs. Dans [58], la complexité pour l'encodage d'un bloc est réduite à environ $\mathcal{O}\left(N^{\frac{3}{2}}\right)$ opérations en utilisant des techniques spécifiques de construction des matrices de contrôle de parité.

Pour un codeur RCDO employant la procédure de la terminaison, la complexité associée à génération de la séquence de terminaison doit être considérée. En général, les matrices de terminaison ne sont pas creuses. En effet, la densité des 1's dans ces matrices pour les codes RCDO terminés simulés est près de 50%. Par conséquent, le délai de la génération de tous les bits de terminaison est de l'ordre $\mathcal{O}(L \cdot m_s)$ éléments de délai. Rappelons que $L \approx m_s$, alors la latence associée à la terminaison nécessite $\mathcal{O}({m_s}^2)$ éléments de délai. En termes de mémoires

requises à la génération des bits de terminaison, la matrice de terminaison nécessite $\mathcal{O}(L \cdot m_s)$ bits. Or, comme nous l'avons décrit à la section précédente, seules les b premières lignes de la matrice de terminaison doivent être sauvegardées afin d'implémenter les codes RCDO terminés de taux de codage $r = b/c$. Par conséquent, la mémoire nécessaire à la génération de la séquence de terminaison est de l'ordre $\mathcal{O}(m_s)$ éléments de mémoires.

Comme nous viendrons de montrer, la complexité associée à la génération de la séquence de terminaison dépend directement de la mémoire du code m_s. Par conséquent, dans le but de minimiser cette complexité, nous présentons à la section suivante un nouvel algorithme capable de générer des codes RCDO terminés tout en réduisant la mémoire m_s.

4.4 Construction des codes convolutionnels récursifs à mutli-registres RCDO terminables

Dans cette section, nous proposons un algorithme de construction des codes RCDO terminables basé sur l'algorithme original présenté à la sous-section 3.2.2 du Chapitre 3. Pour ce faire, nous apportons les modifications nécessaires à l'algorithme de recherche aléatoire [27] de sorte qu'il soit capable de générer des matrices de contrôle de parité dont leurs sous-matrices $\mathbf{C}$ sont inversibles, toute en satisfaisant les conditions de double orthogonalité associées aux codes RCDO. Notons que la méthode proposée permet de construire des codes réguliers ainsi qu'irréguliers. En effet, nous pouvons générer des codes RCDO terminables à temps-invariant avec plusieurs paramètres, y compris la mémoire du code m_s et le taux de codage $r = b/c$, en utilisant le nouvel algorithme de construction proposé.

4.4.1 Conditions de la terminaison

Un codeur RCDO capable de reconverger vers son état initial à l'aide d'une séquence de bits ayant la plus petite longueur possible est éminemment souhaitable. Il a été démontré dans [52] que presque tous les codes (m_s, d_λ, d_ρ) LDPC convolutionnels de taux de codage $r = b/c$ définis par des matrices de contrôle de parité composées par des sous-matrices de permutations et des sous-matrices nulles, peuvent être terminés par des séquences de terminaison dont les longueurs ne dépassent pas les $c \cdot \left(\left\lceil \frac{c-b}{b} m_s \right\rceil + 1\right)$ bits. Il a été aussi prouvé dans [52] qu'en utilisant une telle forme de la matrice de contrôle de parité $\mathbf{H}^T$, la sous-matrice $\mathbf{C}^{\boldsymbol{T}}_{cL \times (c-b)(L+m_s)}$

pourrait être de rang complet. De la même façon, nous construisons une nouvelle classe de codes RCDO définis par une matrice de contrôle de parité $\mathbf{H}^T$ composée par des blocs de matrices de permutations et de matrices nulles.

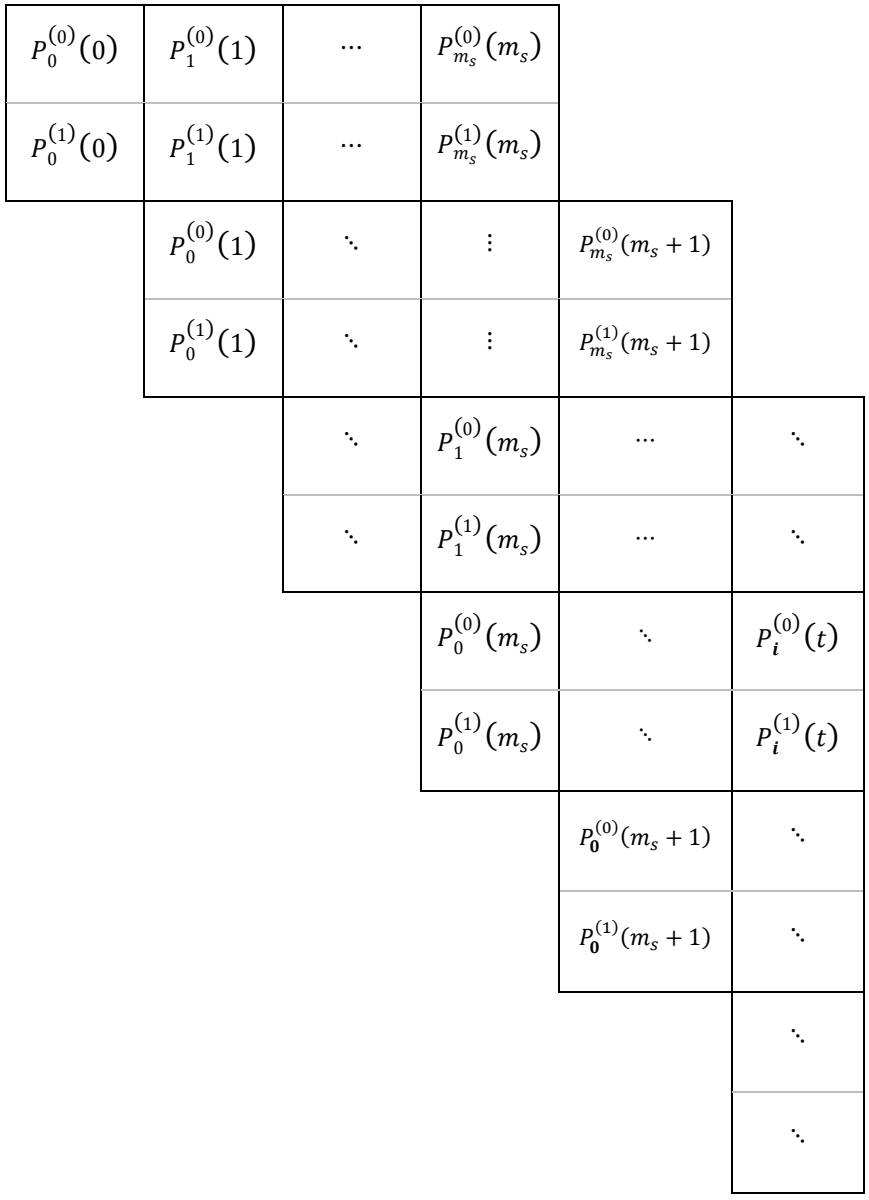


FIGURE 4-6 : Matrice de vérification de parité $\mathbf{H}^T$ composée par des sous-matrices de permutations.

À la Figure 4.6, nous montrons les matrices de permutations constituant la matrice $\mathbf{H}^T$ d'un code RCDO ayant un taux de codage $r = b/c$ et une mémoire du code m_s. Ainsi, les sous-matrices $H_i^T(t)$, $i = 0, 1, \dots, m_s$, peuvent être redéfinies comme suit :

$$H_i^{\boldsymbol{T}}(t) = \begin{bmatrix} P_i^{(0)}(t) \\ \hline P_i^{(1)}(t) \end{bmatrix}, \tag{4.13}$$

où les deux sous-matrices $P_i^{(0)}(t)$ et $P_i^{(1)}(t)$ peuvent êtres soit des matrices nulles, soit des matrices de permutations de dimensions $b \times (c-b)$ et $(c-b) \times (c-b)$, respectivement. Par définition d'un code systématique RCDO de taux de codage $r = b/c$, la sous-matrice $P_0^{(1)}(t)$ est toujours une matrice identité, et ce, quelque soit t.

En nous conformant à la matrice de contrôle de parité binaire $\mathbf{H}^T$ d'un code RCDO terminable obtenue à l'aide de la méthode de construction décrite ci-dessus, la matrice de vérification exprimée sous la forme polynômiale $\mathbf{H}^T(D)$ doit être construite de manière équivalente. Nous présentons un exemple de code convolutionnel récursif ayant un taux de codage $r = b/c = 4/8$ et une mémoire m_s afin d'illustrer la procédure de construction d'un code RCDO terminable.

Précisons que ce code est défini par une matrice de contrôle de parité $\mathbf{H}^T$, dont les sous-matrices $H_i^{\boldsymbol{T}}(t)$ sont de dimensions 8×4. Supposons par exemple que pour un $i \in \{0, \dots, m_s\}$ donné, la sous-matrice $H_i^{\boldsymbol{T}}(t)$ est composée d'une matrice de permutation $P_i^{(0)}(t)$ et d'une matrice nulle $P_i^{(1)}(t)$:

$$H_i^{\boldsymbol{T}}(t) = \begin{bmatrix} P_i^{(0)}(t) \\ \hline P_i^{(1)}(t) \end{bmatrix} = \left[\begin{array}{cccc} 1 & & & \\ & & 1 & \\ & 1 & & \\ \hline 0 & \cdots & & 0 \\ \vdots & \ddots & & \vdots \\ & & \ddots & \\ 0 & & \cdots & 0 \end{array}\right]. \tag{4.14}$$

Ceci se traduit dans la matrice de contrôle $\mathbf{H}^T(D)$ par :

$$\mathbf{H}^T(D) = \left(\begin{array}{cccc} D^i & * & * & * \\ * & * & D^i & * \\ * & D^i & * & * \\ \hline * & * & * & D^i \\ * & \cdots & & * \\ \vdots & \ddots & & \vdots \\ & & \ddots & \\ * & & \cdots & * \end{array}\right), \tag{4.15}$$

où l'exposant de D, $\alpha_{m,n}$, prend la valeur i lorsque l'élément correspondant $h_i^{(m,n)}(t)$ de $H_i^{\boldsymbol{T}}(t)$ est non nul. En généralisant ce processus pour tout $i \in \{0, \dots, m_s\}$, la matrice de vérification d'un code terminable est construite de la façon suivante : les valeurs $\alpha_{m,n}$ de la partie supérieure et de la partie inférieure de la matrice $\mathbf{H}^T(D)$ sont placées de sorte qu'il y a une et une seule valeur égale à $\alpha_{m,n}$ par ligne et par colonne de la sous-matrice concernée. Ainsi, nous identifions les conditions sur les valeurs de $\alpha_{m,n}$ que nous nommons les conditions de terminaison. Rappelons que par définition, les éléments de la diagonale inférieure d'une matrice de contrôle $\mathbf{H}^T(D)$ d'un code RCDO sont égaux à 1, c'est-à-dire que les valeurs $\alpha_{b+n,n}$ sont égaux à zéro, $1 \leq n \leq (c-b)$, ce qui concorde avec les conditions de la terminaison.

Cette nouvelle technique de construction représente le cœur même de ce projet de recherche. En effet, outre les conditions de double orthogonalité imposées aux valeurs $\alpha_{m,n}$, la génération des codes RCDO terminables apporte des exigences supplémentaires sur les connexions des codeurs, en particulier, les valeurs $\alpha_{m,n}$ doivent être assignées selon les conditions de la terminaison proposées ci-dessus.

4.4.2 Algorithme de recherche de codes RCDO terminables

Suite à la section précédente, nous considérons à présent les conditions supplémentaires qui permettent de construire des codes RCDO terminables. Pour appliquer l'algorithme de recherche aléatoire présenté à la Section 3.2.2 du chapitre précédent sur les codes RCDO terminables, nous apportons des modifications afin de tenir compte des nouvelles conditions de la terminaison. L'algorithme ainsi modifié, a pour but de minimiser la valeur de m_s pour une matrice de contrôle $\mathbf{H}^T(D)$ d'un code RCDO terminable de taux de codage $r = b/c$ et qui répond à une certaine paire de distributions $(\lambda(x), \rho(x))$.

Lors de la construction de la matrice $\mathbf{H}^T(D)$, trois contraintes doivent être prises en considération. Dans le cas contraire, le code généré ne correspondra pas à un code RCDO terminable valide. La première contrainte consiste à attribuer la valeur 1 aux éléments formant la diagonale inférieure de la matrice $\mathbf{H}^T(D)$. Comme nous l'avons mentionné précédemment, cette contrainte peut être satisfaite en initialisant les connexions $\alpha_{(b+j),j}$ à la valeur 0 et les coefficients $h_{(b+j),j}$ à la valeur 1, et ce, pour tout $j \in \{1, \dots, (c-b)\}$. La deuxième contrainte consiste à

choisir les positions des connexions des codeurs $\alpha_{i,j}$ de manière à satisfaire les conditions de double orthogonalité. La dernière contrainte est que la même valeur de $\alpha_{i,j}$ ne peut exister qu'une seule fois par ligne et une seule fois par colonne de la sous-matrice supérieure ou inférieure de $\mathbf{H}^T(D)$.

Nous considérons maintenant un cas général d'un code RCDO terminable de taux de codage $r = b/c$ afin de décrire l'algorithme de recherche modifié. Précisons que la mémoire du code est égale à m_s et que la matrice de contrôle de parité $\mathbf{H}^T(D)$ est de dimension $c \times (c-b)$. Nous rappelons les définitions de S et m_s^{max} comme étant respectivement, le sous-ensemble des connexions qui vérifie les conditions de double orthogonalité, et la valeur maximale autorisée des connexions. De plus, nous utilisons une matrice *Position* afin d'indiquer les places des connexions dans la matrice $\mathbf{H}^T(D)$. Notons que chaque ligne de la matrice *Position* regroupe les places des connexions qui vont avoir la même valeur. Prenons par exemple un code convolutionnel récursif de taux de codage $r = b/c = 3/6$ défini par la matrice de contrôle de parité suivante :

$$\mathbf{H}^T(D) = \left(\begin{array}{ccc} D^{\alpha_{1,1}} & D^{\alpha_{1,2}} & 0 \\ D^{\alpha_{2,1}} & 0 & D^{\alpha_{2,3}} \\ 0 & D^{\alpha_{3,2}} & D^{\alpha_{3,3}} \\ \hline D^{\alpha_{4,1}} & D^{\alpha_{4,2}} & 0 \\ 0 & D^{\alpha_{5,2}} & D^{\alpha_{5,3}} \\ D^{\alpha_{6,1}} & 0 & D^{\alpha_{6,3}} \end{array}\right). \tag{4.16}$$

Pour construire une matrice de vérification en respectant les conditions de terminaison, nous représentons les connexions $\alpha_{i,j}$ ayant la même valeur par le même symbole. Ainsi, une des distributions possibles des connexions de la matrice $\mathbf{H}^T(D)$ est exprimée de la manière suivante :

$$\mathbf{H}^T(D) = \left(\begin{array}{ccc} D^{\times} & D^{\blacksquare} & 0 \\ D^{\blacksquare} & 0 & D^{\times} \\ 0 & D^{\times} & D^{\blacksquare} \\ \hline D^{0} & D^{*} & 0 \\ 0 & D^{0} & D^{*} \\ D^{*} & 0 & D^{0} \end{array}\right). \tag{4.17}$$

Par conséquent, en nous basant sur la numérotation des connexions présentée à l'équation (3.5), la matrice *position* est définie de la manière suivante :

$$position = \begin{pmatrix} 1 & 9 & 14 \\ 2 & 7 & 15 \\ 4 & 11 & 18 \\ 6 & 10 & 17 \end{pmatrix}. \tag{4.18}$$

Nous nous servons également d'un compteur, *cpt*, indiquant les connexions en cours de recherche situées dans la matrice $\mathbf{H}^T(D)$ aux emplacements de *cpt*—ème ligne de *position*. Le principe de l'algorithme est décrit comme suit :

Algorithme de construction de la matrice de contrôle $\mathbf{H}^T(D)$ d'un code RCDO terminable

$m_s^{max} \in \mathbb{N}$ fixée, $cpt \leftarrow 0$, $S = \{\emptyset\}$.

tant que ($cpt <$ nombre de ligne de $position$) **faire**

sélectionner les connexions $\alpha_{i,j}$ qui se trouvent aux emplacements de *cpt*—ème ligne de *position*

générer aléatoirement une seule valeur entière pour toutes les $\alpha_{i,j}$ sélectionnées qui est inférieure à m_s^{max}, $S \leftarrow S \cup \{\alpha_{i,j}\}$

si S vérifie les conditions de double orthogonalité **alors**

$cpt \leftarrow cpt + 1$

Sinon

retirer les nouvelles valeurs ajoutées $\alpha_{i,j}$ de S: $S \leftarrow S \setminus \{\alpha_{i,j}\}$

fin si

fin tant que

Précisons que si après un certain temps aucune solution n'a pu être trouvée, la matrice *position* est redéfinie en choisissant une autre distribution des connexions de la matrice $\mathbf{H}^T(D)$, puis une nouvelle recherche est relancée.

4.4.3 Réduction de la mémoire du code RCDO terminable

Dans cette sous-section, nous présentons les résultats de recherches des codes convolutionnels récursifs doublement orthogonaux terminables de taux de codage $r = b/c$ où $c = 2b$ et $b \in \{4, 5, 6, 8, 10\}$. Suite à la méthode de construction décrite à la section précédente, nous avons proposé un nouvel algorithme de recherche capable de générer des codes RCDO terminables tout en réduisant la mémoire m_s.

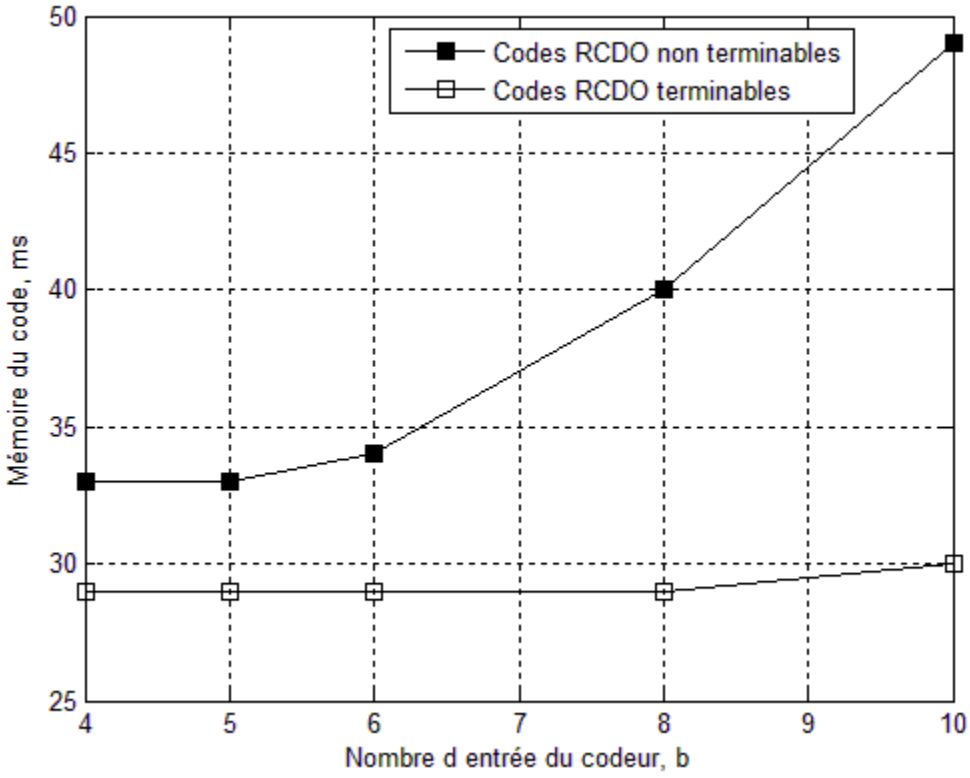


FIGURE 4-7 : Comparaison des mémoires des codes m_s pour différentes tailles des encodeurs RCDO (3,6) réguliers terminables et non terminable.

La Figure 4.7 montre l'évolution de la mémoire du code m_s en fonction du nombre de bits d'information b entrant dans les codeurs RCDO réguliers $(3, 6)$ terminables et non terminables. Il est intéressant de noter que malgré l'imposition d'un nombre supplémentaire de conditions aux connexions $\alpha_{i,j}$ afin de construire des codes RCDO terminables, la mémoire du code m_s de ces derniers a été réduite par rapport à celle des codes RCDO non terminables proposés dans [22]. À

titre de comparaison, Roy dans sa thèse [22] a réussi à construire un code RCDO régulier $(3, 6)$ non terminable de taux de codage $r = 10/20$ ayant la mémoire la plus petite possible $m_s = 49$, cependant, à l'aide de la nouvelle technique de construction proposée nous avons pu construire un code RCDO terminable possédant exactement les mêmes caractéristiques, i.e. le taux de codage $r = b/c = 10/20$ et la paire de distributions $\big(\lambda(x), \rho(x)\big)$, mais ayant une mémoire plus petite, $m_s = 30$. Les résultats obtenus sont très prometteurs puisque la diminution de la valeur m_s se traduit par une réduction de la complexité lors du décodage, ainsi que par une réduction de la latence lors du décodage, ainsi que par une réduction de la perte du taux de codage causé par la séquence de la terminaison. Nous définissons le taux de codage effectif tel que :

$$r_{eff} = \frac{bn}{c(n+L)}, \tag{4.19}$$

avec n représentant le nombre de symboles d'une trame et L le nombre de symboles de la séquence de la terminaison qui est comparable à la mémoire m_s.

4.5 Vérification de la validité de la séquence de la terminaison

Dans le but de vérifier l'exactitude du processus de la terminaison des codes RCDO, nous proposons un test de vérification pratique exécuté sur ordinateur. En effet, la technique de terminaison proposée dans ce mémoire a été vérifiée sur 12 codes RCDO réguliers et irréguliers terminables ayant des mémoires m_s allant de 29 à 205 et des taux de codage $r = b/c = 4/8$, $5/10$, $6/12$, $8/16$ et $10/20$. Les codes ont été construits en utilisant l'algorithme décrit à la Section 4.3.2.2. Le nombre des symboles de la séquence de terminaison L a été choisi suffisamment grand pour rendre l'équation (4.8) solvable. Notons que pour tous les codes simulés, la valeur de L ne dépasse pas $\left(\left\lceil \frac{c-b}{b} m_s \right\rceil + 1\right)$. Les matrices de terminaison sont générées en utilisant la technique d'élimination de Gauss-Jordan, où toutes les inconnues libres ont été assignées à des 0's, et où seules les b premières lignes de chaque matrice de terminaison ont été enregistrées. En nous référant à l'équation (4.4), nous vérifions l'exactitude de la terminaison des codes de taux de codage $r = b/c$ à l'aide de la procédure suivante :

1. Générer une trame de données aléatoires ayant une longueur finie N_f.
2. Coder cette trame de données.
3. Générer les bits de terminaison puis les injecter à l'entrée du codeur, en utilisant la méthode de terminaison décrite à la Section 4.3.1.
4. Envoyer $b \cdot m_s$ bits zéro à l'entrée du codeur et vérifier les bits de parité correspondants à la sortie. Cette étape consiste à réinitialiser le codeur. Si tous les $(c - b) \cdot m_s$ bits sont égaux à zéro, alors la séquence de la terminaison est générée correctement. Sinon une erreur au niveau de la terminaison est déclarée.

Pour tous les codes testés, plus de dix million de trames ont été simulées. Nous n'avons constaté aucune défaillance dans le processus de la terminaison d'un code, c'est-à-dire que le codeur a correctement terminé toutes les trames.

4.6 Conclusion

L'un des points forts des codes RCDO est qu'ils peuvent traiter des trames de longueurs finies et variables. Pour appliquer la matrice de contrôle de parité infinie à une trame, la terminaison des codes est nécessaire afin d'améliorer les performances d'erreur à la fin de la trame. La terminaison des codes RCDO consiste à ramener le codeur vers son état initial. Une trame codée par un codeur RCDO terminé peut être vue comme étant un mot de code RCDO ayant un nombre infini de zéros aux deux extrémités. Il est clair que la terminaison d'un codeur à partir d'un état final est une tâche simple pour les codes convolutionnels non récursifs, mais ne l'est pas pour les codes RCDO.

Le problème mathématique de la terminaison a été défini dans ce chapitre pour un codeur RCDO ayant un taux de codage $r = b/c$ et une mémoire du code m_s. Une séquence de terminaison de longueur $b \cdot L$ doit être déterminée pour reconduire le codeur à un état zéro-partiel. En nous référant à l'équation de vérification de parité d'un code RCDO terminé, la séquence de la terminaison peut être calculée en résolvant un système d'équations linéaires défini par une matrice $\mathbf{C}$ et le vecteur d'état final $\mathbf{w}$ du codeur. Nous avons proposé une nouvelle technique de

terminaison capable de générer graduellement et en continu les bits de terminaison tout en les envoyant à l'entrée du codeur. De plus, nous avons défini les conditions de la terminaison permettant de construire des codes RCDO qui peuvent être terminés par des séquences de terminaison ayant des longueurs les plus petites possibles (i.e. $L \approx m_s$). En nous basant sur les conditions de terminaison, un algorithme de recherche modifié et adapté aux codes RCDO terminables a été proposé. L'algorithme de recherche proposé a permis de réduire la mémoire m_s des codes RCDO terminables. Ainsi, nous avons minimisé le plus possible la perte de taux de codage causée par la terminaison ainsi que réduit la latence au décodage.

CHAPITRE 5

RÉSULTATS DE SIMULATIONS DES CODES CONVOLUTIONNELS RÉCURSIFS DOUBLEMENT ORTHOGONAUX

Après avoir présenté dans le chapitre précédent l'architecture du codeur convolutionnel récursif multi-registres RCDO terminable ainsi que la méthode de génération de la séquence de terminaison qui permet de terminer les codes RCDO, nous rapportons dans ce chapitre les résultats expérimentaux obtenus par simulation à l'ordinateur de l'implémentation de cette architecture. Les performances d'erreur (BER : taux d'erreur binaire et FER : taux d'erreur de trame) des codes RCDO terminables transmis en continu et des codes RCDO terminés transmis en blocs sont exposées. Les données sont générées en considérant une modulation BPSK sur un canal binaire symétrique à bruit additif blanc et gaussien (AWGN), où la sortie du canal est non quantifiée. L'algorithme itératif BP basé sur l'approximation min-somme a été utilisé pour le décodage des codes RCDO [29]. Les performances d'erreur des codes RCDO terminables obtenues par simulations logicielles sur ordinateur sont alors comparées aux limites théoriques [27] et aux performances des codes LDPC en blocs et convolutionnels, effectuées dans plusieurs travaux de recherche [36] [53] [54].

L'objectif de la première partie du chapitre est donc d'étudier les performances des codes RCDO terminés lors d'une transmission par paquets. Ainsi, nous évaluons l'impact de la séquence de terminaison sur les performances d'erreur. L'objectif de la seconde partie du chapitre consiste à étudier les performances des codes RCDO terminables lors d'une transmission en continu, et ce, en les comparant avec les codes RCDO non terminables, les codes LDPC en blocs et les codes LDPC convolutionnels.

5.1 Performances des codes RCDO terminés lors d'une transmission en blocs

Dans le but de préserver certaines compatibilités avec certains systèmes de communication, il est essentiel de coder les données segmentées dans des trames de longueurs prédéterminées. Dans ce contexte, nous évaluons les performances des codes RCDO terminés afin d'étudier les impacts du processus de la terminaison, tels que la perte du taux de codage et l'amélioration de la correction d'erreurs. En nous basant sur la technique de terminaison ainsi que la méthode de construction des codeurs RCDO terminables présentées au chapitre précédent, les résultats de simulation montrent que pour les codes RCDO terminés de longueurs moyennes, ces derniers peuvent offrir des performances proches de la limite de Shannon.

5.1.1 Impact de la terminaison sur les performances

L'application de la terminaison génère deux impacts sur les performances d'un code RCDO : perte du taux de codage et protection inégale contre les erreurs.

5.1.1.1 Perte du taux de codage

En garantissant de bonnes performances à la fin de la trame, la séquence de terminaison codée est transmise et entraîne par conséquent une perte du taux de codage. Pour un code RCDO de taux de codage $r = b/c$, si la longueur d'une trame avant le codage est $b \cdot n$ et la longueur de la séquence de terminaison codée est $c \cdot L$, alors le taux de codage effectif est défini comme suit :

$$r_{eff} = \frac{b \cdot n}{c \cdot n + c \cdot L} = r \cdot \frac{n}{n+L}. \tag{5.1}$$

Pour une séquence de terminaison de longueur fixe, la perte du taux de codage dépend de la longueur de la trame. En effet, plus la longueur de la trame de données est grande plus la perte du taux de codage est faible. Un avantage de la méthode de terminaison proposée est que la perte du taux de codage est réduite, car la longueur de la terminaison L est très proche de la mémoire du code m_s. La Figure 5.1 montre l'évolution du taux de codage effectif du code $(3, 6)$ RCDO terminé ayant un taux de codage $r = 4/8$ et une mémoire du code $m_s = 29$ appliqué aux trames de données de différentes longueurs. À titre de comparaison, nous avons ajouté sur cette figure l'évolution du taux de codage effectif du code $(1024, 3, 6)$ LDPC-B. Ce choix est justifié

puisque les performances d'erreur du code $(1024, 3, 6)$ LDPC-B sont comparables à celles du code $(29, 3, 6)$ RCDO terminé, comme nous allons le voir à la Section 5.1.3 (Figure 5.7). Notons que le code RCDO terminé possède un taux de codage effectif plus élevé que celui du code LDPC-B pour la plupart des longueurs d'une trame de données. Le contraire n'est vrai que lorsque la longueur de la trame codée est égale ou légèrement inférieure à la longueur de bloc du code LDPC-B.

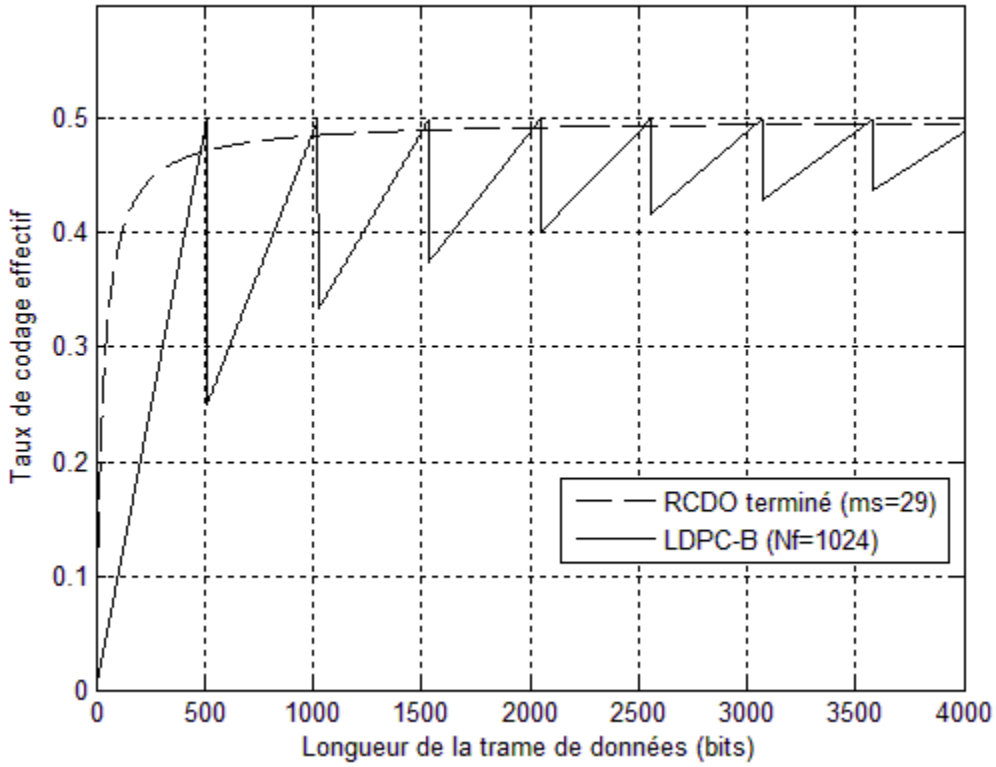


FIGURE 5-1 : Taux de codage effectif obtenu par les codes $(29, 3, 6)$ RCDO terminé et $(1024, 3, 6)$ LDPC-B lorsqu'ils sont appliqués aux différentes longueurs de trames de données.

Lors de l'application d'un code LDPC-B ou d'un code RCDO à un système de communication basé sur une transmission par blocs, la perte du taux de codage causée par le bourrage ou la terminaison entraîne une demande de puissance supplémentaire (i.e. un surplus d'énergie) afin de traiter et de transmettre ces bits additionnels. Précisons que plus la perte du taux de codage est faible plus la puissance supplémentaire requise est réduite. En effet, dans un système de communication basé sur la transmission des trames de longueurs variables, un seul code RCDO

est capable de traiter toutes les longueurs des trames. De plus, la puissance supplémentaire a été réduite par rapport à celle causée par un code LDPC-B comparable, et ce, pour la plupart des longueurs des trames. Un exemple d'un surplus de puissance peut être vu dans la norme 10GBASE-T [55], où un seul code $(2048, 1723)$ LDPC-B est utilisé pour traiter des trames de longueurs variables. À chaque instant, les signaux sont transmis et le codeur LDPC-B accepte à son entrée les blocs de bits pour générer à sa sortie les mots de code qui peuvent contenir des bits de bourrage. Ainsi, l'efficacité de la puissance devient très faible lorsque la trame de données envoyée est de petite longueur.

5.1.1.2 Protection inégale contre les erreurs

Les bits de données codés par un code RCDO terminé subissent une protection inégale contre les erreurs au début et à la fin de la trame. Considérons une trame codée isolée de toutes les autres. Cette trame est codée à partir de l'état zéro en utilisant l'étape de préréglage présentée à la Section 4.2 du chapitre précédent. Parallèlement, le récepteur détecte le marqueur DDT et initialise tous les messages dans le décodeur RCDO à l'infini. De cette manière, le code peut être effectivement fiable au début de la trame. Comme nous l'avons vu à l'équation (4.2) du Chapitre 4, ceci s'explique par le fait que les premiers bits de la trame partagent des équations de contrôle de parité avec les bits zéros déterministes. En effet, les bits zéros déterministes participant aux échanges de messages lors du décodage permettent aux bits connectés aux mêmes nœuds de contraintes d'avoir une probabilité élevée d'être correctement décodés. De plus, grâce aux itérations, cet avantage va se propager aux autres bits qui ne sont pas directement reliés aux nœuds de contraintes connectés aux zéros déterministes.

Dans la plupart des systèmes de communication basés sur une transmission par paquets, le champ de l'entête d'une trame comporte des informations d'adresses importantes. Prenons par exemple la structure d'une trame Ethernet, les adresses MAC (en anglais *Media Access Control*) de source et de destination sont situées au début de la trame. L'amélioration de la protection contre les erreurs dans ces champs d'adresses est très avantageuse afin de réduire les délais de commutation (en anglais *cut-through switching*) [56], où les trames sont transmises dès que leurs adresses de destinations sont évaluées et avant la vérification d'erreurs. Lorsque l'entête d'une trame est correctement reçu, cette trame peut être transmise à la bonne adresse de destinataire et une

retransmission à partir de l'adresse de source peut être demandée lorsque la trame comporte des erreurs. Par contre, un entête mal reçu peut causer des conséquences indésirables.

Comme nous l'avons mentionné à la Section 4.2.1 du Chapitre 4, à la fin du codage de chaque trame, le codeur RCDO terminé converge forcément vers l'état zéro. De façon symétrique, les bits à la fin de la trame partagent des équations de contrôle de parité avec les bits zéros déterministes. Lorsque le récepteur détecte correctement le marqueur DFT et alimente le décodeur par des LRVs infinis provenant du canal, la même amélioration de protection contre les erreurs peut être constatée à la fin de la trame.

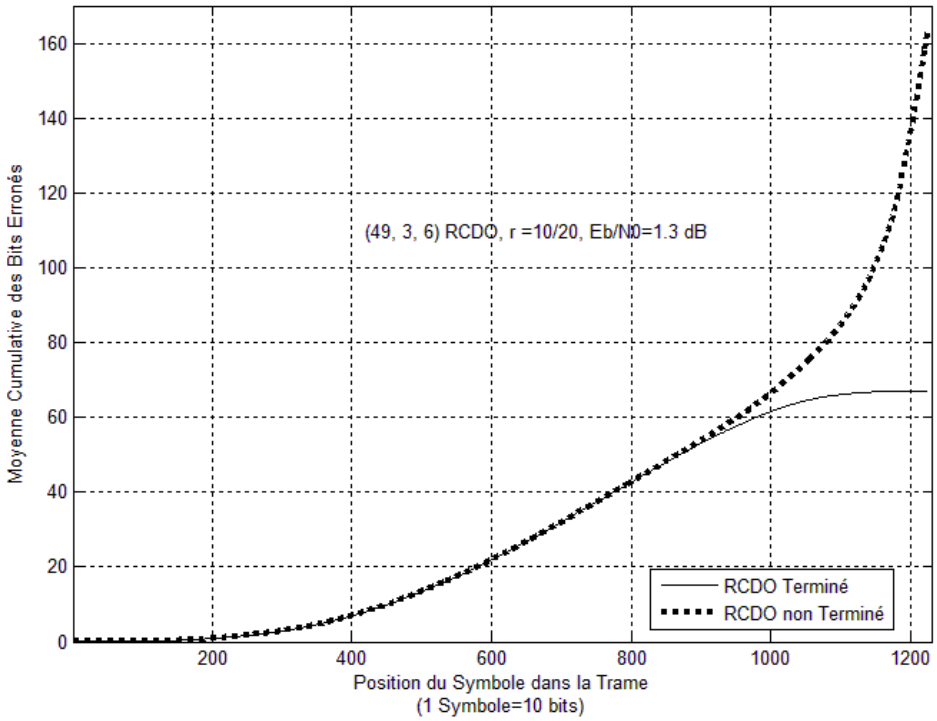


FIGURE 5-2 : La moyenne cumulative des bits erronés dans une trame codée par un codeur RCDO. Le nombre cumulatif des bits erronés est compté à partir du bit de position 0 jusqu'au bit en cours. Les simulations ont été effectuées sans tenir compte de la perte du taux de codage.

La Figure 5.2 montre la protection inégale contre les erreurs obtenue grâce au préréglage et grâce à la terminaison du codeur RCDO. Nous avons tracé la moyenne cumulative des bits erronés au fur et à mesure que nous progressons dans une trame codée à partir de l'état zéro par un codeur RCDO régulier (3, 6) ayant un taux de codage $r = 10/20$ et une mémoire du code $m_s = 49$. La

simulation a été réalisée à un rapport signal sur bruit $E_b/N_0 = 1.3$ dB en effectuant 20 itérations et en utilisant l'approximation min-somme lors du décodage. Les résultats ont été obtenus en faisant la moyenne de plus de 2000 trames de longueur $N_f = 25 \cdot b \cdot m_s = 12250$ bits. Nous remarquons que les performances d'erreur à la fin de la trame dépendent de l'application de la terminaison ou non. Comme nous pouvons le voir à la Figure 5.2, lorsque la terminaison n'est pas employée, le nombre des bits erronés à la fin de la trame augmente de façon rapide. Cependant, l'utilisation de la procédure de la terminaison générant les $(m_s + 1) \cdot b$ bits de terminaison, permet de réduire le nombre de bits erronés à la fin de la trame. Rappelons que la mémoire du code est seulement égale à 49 alors que plus de 1500 bits bénéficient d'un aplatissement de la courbe du nombre d'erreurs cumulées. Il s'ensuit que la terminaison permet non seulement la protection de la séquence de terminaison, mais également celles des $3bm_s$ à $4bm_s$ bits d'information qui la précèdent.

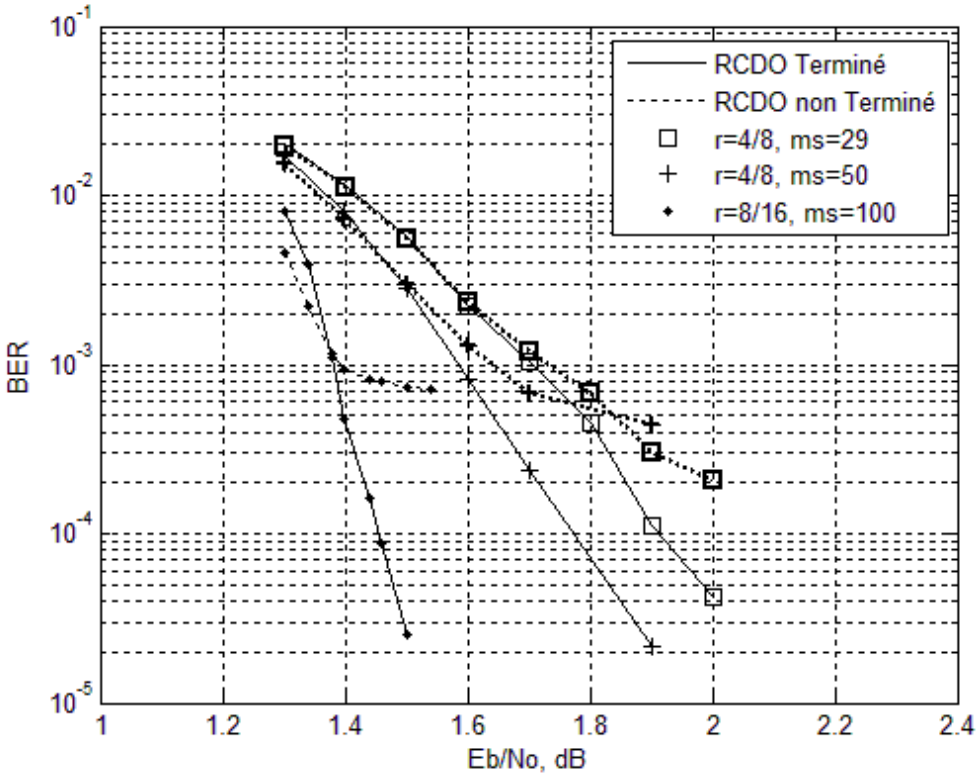


FIGURE 5-3 : Performances d'erreur pour les codes réguliers $(3, 6)$RCDO terminés et non terminés ayant des différents taux de codage r et des mémoires du code m_s.

À la Figure 5.3, nous présentons également les performances d'erreur BER obtenues par les codes RCDO réguliers (3, 6) terminés ayant respectivement les taux de codage $r = 4/8$ (i.e., $b = 4$) et $r = 8/16$ (i.e., $b = 8$). Les mémoires du code $m_s = 29$ et $m_s = 50$ sont utilisées pour le taux de codage $r = 4/8$, alors que la mémoire du code $m_s = 100$ est utilisée pour le taux de codage $r = 8/16$. Ces résultats ont été obtenus en faisant la moyenne de 100 trames de longueur $N_f = b \cdot 10^4$, ce qui résulte en des taux de codage effectifs $r_{eff} = 0.48$, $r_{eff} = 0.47$ et $r_{eff} = 0.45$ pour les mémoires du code 29, 50 et 100 respectivement. À titre de comparaison, nous avons ajouté sur cette figure les résultats obtenus par les codes RCDO non terminés. Nous présentons à l'Annexe A plus de détails sur les codes RCDO non terminés. Comme nous pouvons le constater sur la Figure 5.3, après avoir effectué 25 itérations, les capacités de correction offertes par les codes RCDO sont améliorées lors de l'utilisation de la technique de terminaison. De plus, nous pouvons observer qu'aux faibles valeurs du rapport signal sur bruit, les codes RCDO terminés et non terminés fournissent des performances d'erreur très similaires, alors qu'à haut rapport signal sur bruit, les codes RCDO terminés offrent des performances de correction d'erreur bien meilleurs que celles obtenues par les codes RCDO non terminés. Nous pouvons également remarquer que les codes RCDO non terminés ne convergent jamais vers leurs seuils théoriques et souffrent d'un plancher d'erreur (en anglais, *error floor*) très élevé. Ceci peut s'expliquer, comme indiqué à l'Annexe A, par le fait que ces codes subissent une faible fiabilité de décodage à la fin de la trame.

5.1.2 Performance de la terminaison des codes RCDO

À la section précédente, nous avons étudié deux impacts de la terminaison des codes RCDO. D'une part, la perte du taux de codage dégrade la performance du code à cause de l'augmentation de la consommation d'énergie. D'autre part, la protection inégale le long de la trame contre les erreurs améliore les performances d'erreur. Bien que la perte du taux de codage puisse être simplement calculée, la protection inégale contre les erreurs ne peut pas être facilement analysée. Dans cette section, nous étudions les résultats de simulations des effets combinés de ces deux impacts.

La Figure 5.4 présente les performances d'erreur BER et FER obtenues par le code RCDO régulier (3, 6) terminé ayant un taux de codage $r = 4/8$ et une mémoire $m_s = 29$. Ce code a été appliqué à des trames de longueurs allant de $3b(m_s + 1) = 360$ à $50b(m_s + 1) = 6000$ bits.

Les longueurs de la trame de données sont choisies de sorte qu'elles soient des multiples de $b(m_s + 1)$. De même, à la Figure 5.5, nous présentons les résultats de simulation du code (30, 3, 6) RCDO terminé de taux de codage $r = 10/20$, pour des trames de longueurs allant de $2b(m_s + 1) = 620$ à $100b(m_s + 1) = 31000$ bits. Enfin, la Figure 5.6 montre les effets combinés de la terminaison sur les performances BER en termes de perte du rapport signal sur bruit (SNR), et ce, pour différentes longueurs de trames. Les longueurs des trames codées sont normalisées par rapport à la longueur de contrainte du code $\vartheta = c(m_s + 1)$. Nous avons ajouté sur cette figure l'évolution de la perte du taux de codage exprimée en dB :

$$r_{perdu} = 10 \log_{10} \left(\frac{r}{r_{eff}} \right) \text{ dB.} \tag{5.1}$$

Nous pouvons remarquer sur cette dernière figure que la perte du SNR est un peu plus petite que celle du taux de codage. Pour les deux codes présentés à la Figure 5.6, lorsque la longueur de la trame est égale à $3b(m_s + 1)$, la perte du SNR à un BER de 10^{-5} est comprise entre 0.75 et 0.85 dB. Cependant, lorsque la trame est de longueur $10b(m_s + 1)$, la perte du SNR est seulement d'environ 0.3 dB.

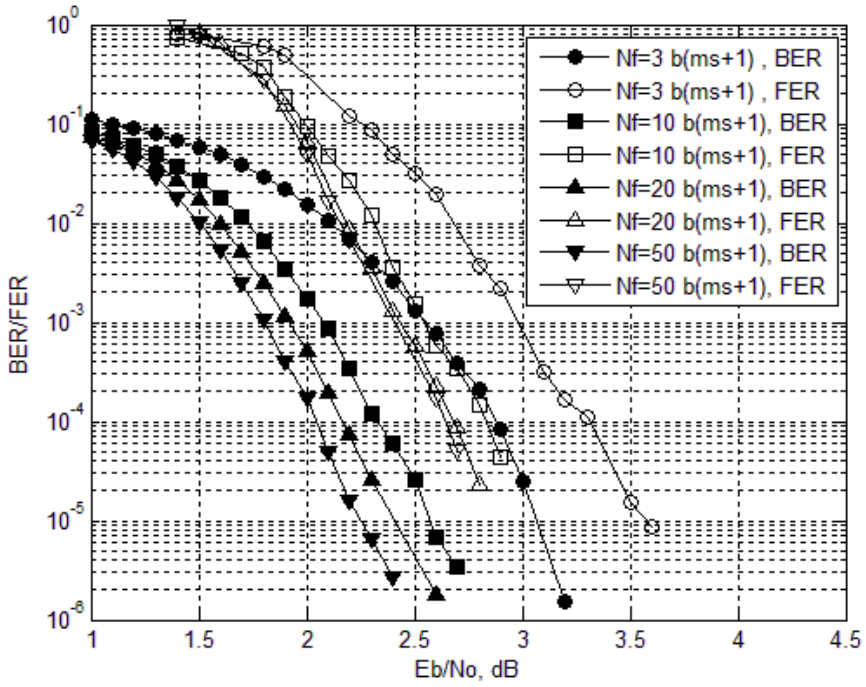


FIGURE 5-4 : Performances d'erreur BER et FER d'un code RCDO terminé ayant un taux de codage $r = 4/8$ et une mémoire du code $m_s = 29$. Les résultats ont été obtenus après avoir effectué 20 itérations de décodage sur des trames de différentes longueurs.

Il est intéressant d'observer à partir des courbes de performances, en particulier celles présentées dans la Figure 5.6, que plus la longueur de la trame est petite plus la perte du taux de codage est élevée et plus la protection contre les erreurs est améliorée. En effet, ceci peut être compris intuitivement car les bits zéros virtuels situés aux deux extrémités de la trame jouent un rôle plus important dans la protection contre les erreurs pour les trames de longueurs plus courtes. Dans le cas extrême, lorsque la trame est de longueur infinie, l'amélioration de la protection contre les erreurs obtenue grâce à la terminaison n'a aucun impact sur les performances d'erreur.

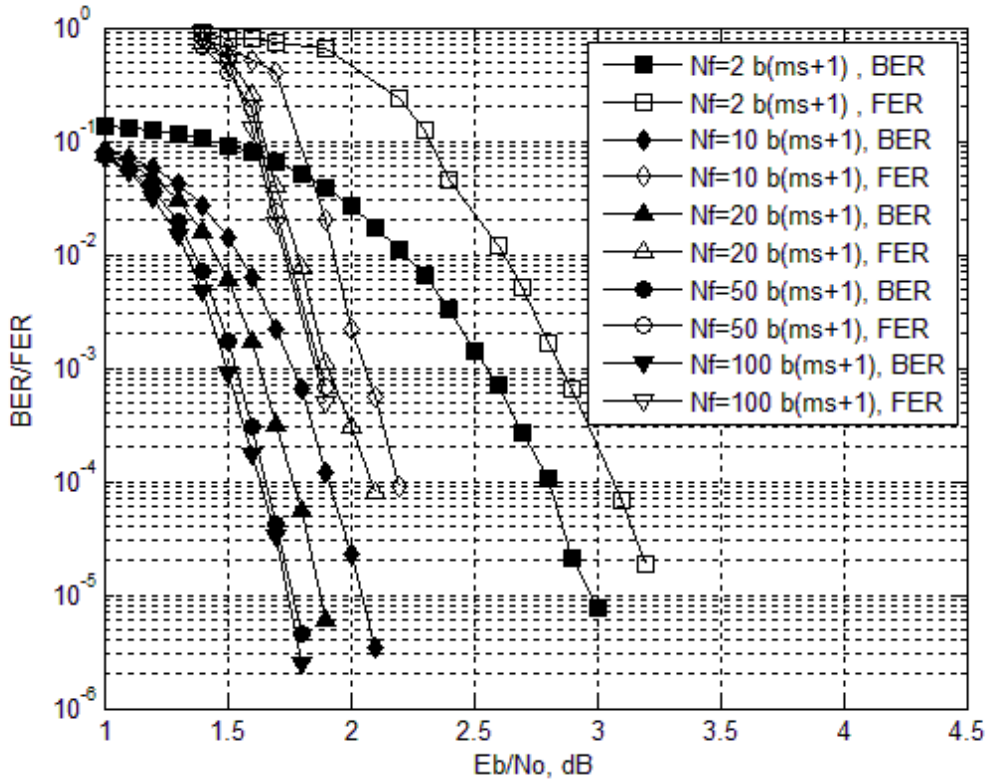


FIGURE 5-5 : Performances d'erreur BER et FER d'un code RCDO terminé ayant un taux de codage $r = 10/20$ et une mémoire du code $m_s = 30$. Les résultats ont été obtenus après avoir effectué 25 itérations de décodage sur des trames de différentes longueurs.

Les codes RCDO terminés permettent de traiter des trames de longueurs variables, mais cet avantage est obtenu au prix d'une perte du SNR causée par la puissance supplémentaire utilisée pour transmettre la séquence de terminaison. Que ce compromis soit rentable ou non, ceci dépend certainement de la conception des systèmes de communication. Dans les normes basées sur la transmission des trames de longueurs variables, tel que Ethernet, la plupart des trames sont de

longueurs élevées ce qui réduit la perte du SNR. De plus, dans certains systèmes de communication à haute vitesse, des trames géantes Ethernet Jumbo [57] sont utilisées afin d'assurer un meilleur usage de la bande passante du réseau tout en réduisant le nombre de trames qui doivent être traitées. En effet, ces trames peuvent avoir une longueur pouvant aller jusqu'à 9000 octets environ [57] ce qui rend la perte du taux de codage r_{perdu} causée par la terminaison négligeable.

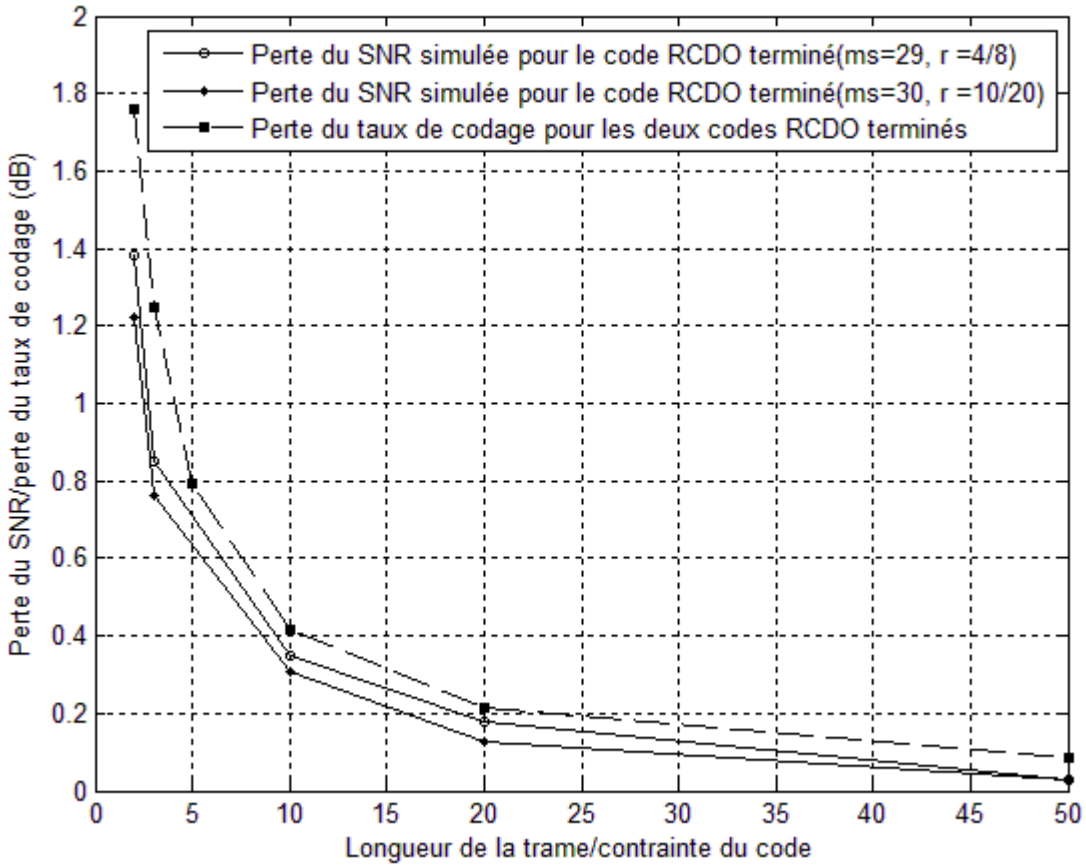


FIGURE 5-6 : Perte du SNR causée par la terminaison des deux codes RCDO terminés appliqués aux différentes longueurs de trames. La perte du SNR est calculée à un BER égal à 10^{-5}. Les courbes en lignes continues sont obtenues à partir des résultats de simulation présentés aux Figures 5.4 et 5.5, alors que la courbe en ligne discontinue est obtenue par le calcul de (5.1).

5.1.3 Comparaison des performances d'erreur obtenues par les codes RCDO terminés et les codes en blocs LDPC

Dans cette section, nous présentons les résultats de simulations par ordinateur de la performance d'erreur des codes convolutionnels récursifs doublement orthogonaux terminés comparée à celle des codes LDPC en blocs.

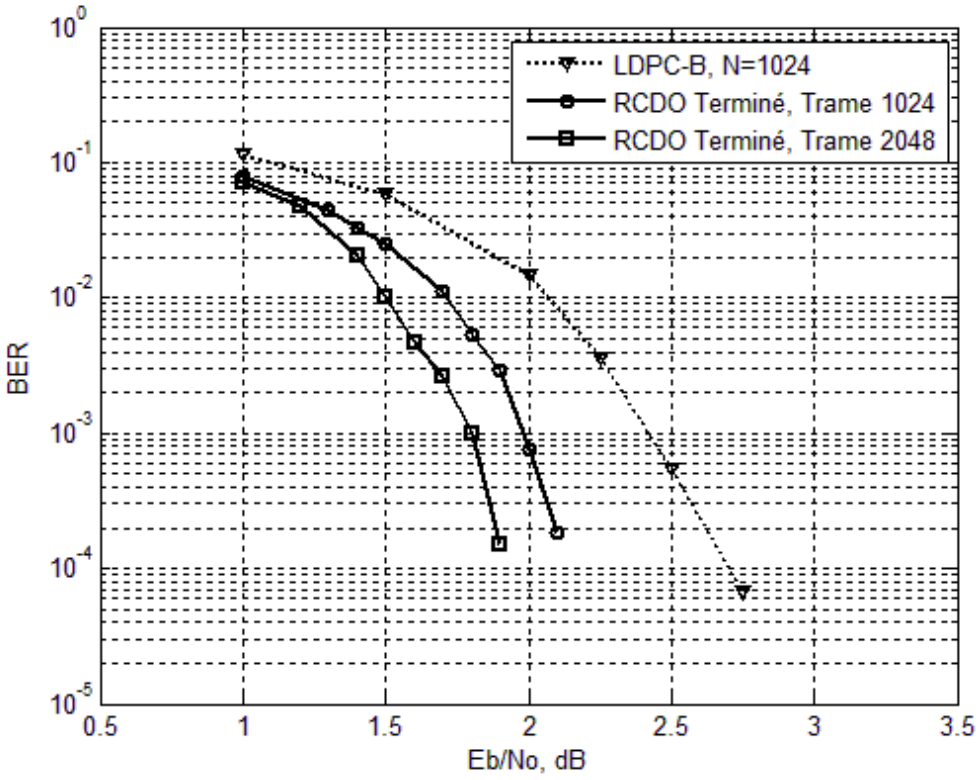


FIGURE 5-7 : Performances d'erreur obtenues par le code (29, 3, 6) RCDO terminé et le code (1024, 3, 6) LDPC-B lorsqu'ils sont appliqués aux trames de longueurs 1024 et 2048 bits. L'approximation min-somme est utilisée avec 50 itérations.

La Figure 5.7 compare les performances BER du code RCDO terminé utilisé à la Figure 5.4 (i.e. $r = 4/8$ et $m_s = 29$) aux résultats obtenus par [53] d'un code LDPC en blocs de taux de codage $r = 1/2$ possédant une longueur de bloc égale à 1024. Les deux codes possèdent la même structure régulière $(3, 6)$ et sont appliqués aux trames de données ayant des longueurs 1024 et 2048 respectivement. Le nombre d'itérations effectuées par chacun des décodeurs est égal à 50. Comme nous pouvons le voir à la Figure 5.7, même en présence d'une perte du taux de codage

causée par la terminaison, le code (29, 3, 6) RCDO terminé de taux de codage $r = 4/8$ offre de meilleures performances BER que celles du code (1024, 3, 6) LDPC-B, et ce, pour les deux longueurs de trames de données utilisées. Rappelons que plus la longueur de la trame de données est élevée (i.e., $N_f = 2048$) plus la perte du taux de codage r_{perdu} est faible et plus les performances d'erreur obtenues par les codes RCDO terminés sont meilleures. De plus, dans les simulations présentées, nous avons illustré le cas idéal pour les codes LDPC-B, puisque les longueurs de trames 1024 et 2048 sont choisies comme des multiples de la longueur du bloc du code LDPC-B. Dans les systèmes de communication transmettant des trames de longueurs variables, tel que la norme Ethernet, l'adaptation des codes LDPC-B aux différentes longueurs de trames peut poser des problèmes.

5.2 Performances des codes RCDO terminables lors d'une transmission en flux continu

Dans cette section, nous comparons un nombre de codes RCDO réguliers terminables aux codes RCDO non terminables proposés dans [27], ainsi qu'aux codes LDPC convolutionnels [53] [54] [36] et enfin aux codes LDPC en blocs [53] [54]. Tous les codes simulés dans cette section possèdent la même structure régulière (3, 6) et sont décodés d'une manière continue sans employer de terminaison. Par ailleurs, toutes les comparaisons des performances obtenues par les codes RCDO terminables, les codes RCDO non terminables, les codes convolutionnels LDPC et les codes en blocs LDPC ont été effectuées à une complexité de processeur égale et à un délai de décodage ainsi qu'au nombre d'unités de mémoires de stockage égaux. Nous comparons à l'Annexe B plusieurs aspects de décodage itératifs des codes RCDO aux codes LDPC-C et aux codes LDPC-B.

5.2.1 Comparaison des performances d'erreur obtenues par les codes RCDO terminables et les codes RCDO non terminables

Suite au chapitre précédent, nous avons étudié la recherche de codes convolutionnels RCDO terminables à l'aide d'une heuristique permettant la minimisation de la mémoire du code m_s, c'est-à-dire que nous avons réduit la complexité de processeur (i.e. $c \cdot (m_s + 1)$), le délai de décodage (i.e. $I \cdot c \cdot (m_s + 1)$, où I désigne le nombre d'itérations de décodage effectuées) et la

mémoire de stockage nécessaire à la réalisation du décodeur itératif RCDO (i.e. $I \cdot (\Gamma + c) \cdot (m_s + 1)$, où Γ désigne le nombre d'arêtes du graphe de Tanner associé au code). Rappelons que l'algorithme de recherche des codes RCDO terminables proposée à la Section 4.4.2 du Chapitre 4, a pour but de minimiser la valeur m_s pour une matrice de contrôle $\mathbf{H}^T(D)$ de dimension fixes et qui répond à une certaine paire de distributions $(\lambda(x), \rho(x))$ [27].

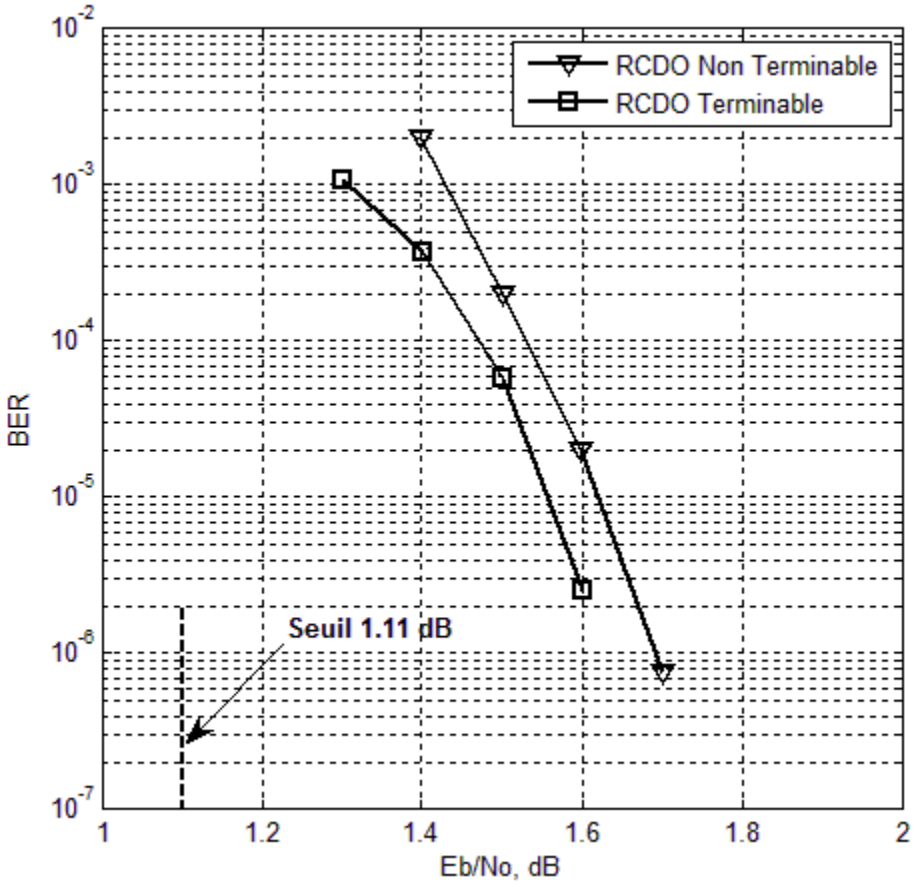


FIGURE 5-8 : Performance d'erreur BER du code RCDO terminable et non terminable. Les deux codes possèdent la même structure régulière $(3, 6)$ et sont de taux de codage $r = 10/20$ et de mémoire du code $m_s = 49$.

À la Figure 5.8, nous présentons les performances d'erreur d'un code RCDO terminable de taux de codage $r = 10/20$ et de mémoire $m_s = 49$ pour lequel les positions des connexions du codeur convolutionnels doivent satisfaire les conditions de la double orthogonalité [27] ainsi que les conditions de la terminaison définies à la Section 4.4.1 du chapitre précédent. Ce code est

défini par une matrice de contrôle de parité $\mathbf{H}^T(D)$ de dimension 20×10 et un graphe biparti de Tanner dont les nœuds variables et les nœuds de contraintes sont tous de degré constant et égal à 3 et 6 respectivement. Précisons que lorsque les codes convolutionnels RCDO réguliers sont définis par la paire de distribution (x^2, x^5), l'algorithme itératif de décodage converge théoriquement vers 1.11 dB. Ainsi, ces codes peuvent au mieux s'approcher à 0.92 dB de la limite de Shannon égale à 0.188 dB. Comme nous pouvons le voir sur la Figure 5.8, après avoir effectué $I = 25$ itérations, ce code terminable offre des performances d'erreur qui se situent à environ 0.5 dB de la valeur du seuil correspondant au graphe biparti.

Nous présentons aussi sur cette même figure les résultats obtenus par [27] d'un code convolutionnel RCDO non terminable pour lequel les positions des connexions du codeur convolutionnels satisfont uniquement aux conditions de la double orthogonalité. Pour assurer une comparaison équitable avec le code RCDO terminable défini ci-dessus, nous nous sommes assurés que le taux de codage, la valeur de la mémoire du code, le nombre d'itérations effectuées et la paire de distribution $\big(\lambda(x), \rho(x)\big)$ soient identiques et égaux à $r = 10/20$, $m_s = 49$, $I = 25$ et (x^2, x^5) respectivement pour les deux codes convolutionnels récursifs. Nous pouvons remarquer sur cette figure que le code RCDO terminable offre un petit gain de codage supplémentaire d'environ 0.07 dB par rapport au code RCDO non terminable. Ceci peut s'expliquer par le fait que la valeur maximale de $\alpha_{i,j}$ (i.e. $m_s = max(\alpha_{i,j})$) est placée sur chaque colonne de la matrice de contrôle de parité $\mathbf{H}^T(D)$ d'un code RCDO terminable. Or, ceci conduit à l'utilisation des LRVs les plus anciens possibles, c'est-à-dire que les LRVs qui correspondent à cette valeur maximale de $\alpha_{i,j}$, dans toutes les équations de contrôle de parité (3.20) afin d'effectuer la mise à jour des nœuds de contraintes. Rappelons que le codeur RCDO commence à coder les bits d'information à partir de l'état zéro et que le récepteur initialise tous les LRVs du décodeur RCDO à la valeur infinie. Il en résulte que les bits au début de la trame sont bien protégés contre les erreurs et que le décodeur génère les premiers LRVs de façon fiable et efficace. Ainsi, les résultats de simulation présentés à la Figure 5.8 montrent que l'utilisation des LRVs les plus anciens possibles dans toutes les équations de mises à jour des nœuds de contraintes mène à améliorer les performances BER du code.

5.2.2 Comparaison des codes RCDO terminables aux codes LDPC

À la section 5.1.3, les performances d'erreur obtenues par le code (29, 3, 6) RCDO terminé ont été comparées à celles obtenues par le code (1024, 3, 6) LDPC en blocs. Compte tenu de la méthode de construction des codes RCDO terminables proposée au Chapitre 4, nous réalisons des comparaisons plus globales sur les performances d'erreur obtenues par les codes RCDO terminables, les codes LDPC convolutionnels ainsi que les codes LDPC en blocs.

Nous présentons aux Figures 5.9 et 5.10 les performances d'erreur d'un nombre de codes RCDO terminables comparées à celles des codes LDPC convolutionnels [53] [54] [36] et à celles des codes LDPC en blocs [53] [54] respectivement. Précisons que tous les codes simulés dans cette section sont réguliers, c'est-à-dire qu'ils sont définis par la paire de distribution (x^2, x^5), et sont décodés en utilisant l'approximation min-somme discutée à la Section 3.5. De plus, tous les codes LDPC-B utilisés dans cette comparaison possèdent une longueur de bloc N_{LDPC-B} égale à la longueur de contrainte des codes RCDO $\vartheta = (m_{s_{RCDO}} + 1) \cdot c$, ce qui permet aux différents décodeurs d'avoir la même complexité de processeur. Alors que, les codes LDPC-C ainsi que les codes RCDO terminables avaient été comparés à un délai de décodage ainsi qu'au nombre d'unités de mémoires de stockage égaux, d'où $\left(m_{s_{RCDO}} + 1\right) \cdot c \cdot I = \left(m_{s_{LDPC-C}} + 1\right) \cdot c \cdot I$. Rappelons que $m_{s_{LDPC-C}}$ et I sont définis comme étant respectivement, la mémoire du code LDPC-C et le nombre d'itérations de décodage. Au Tableau 5.1 nous résumons les différents paramètres des codes simulés dans cette section. Nous pouvons voir sur la Figure 5.9 que pour des mémoires du code relativement petites, le code ($m_{s_{RCDO}} = 31$) RCDO terminable de taux de codage $r = 4/8$ offre, après avoir effectué 50 itérations, un gain de codage asymptotique additionnel d'environ 0.5 dB par rapport au code $\left(m_{s_{LDPC-C}} = 128\right)$ LDPC-C de taux de codage $r = 1/2$. Notons que les simulations ont été effectuées de sorte d'avoir le même délai de décodage et le même nombre d'unités de mémoire nécessaire à la réalisation des deux décodeurs utilisés. De plus, nous avons aussi indiqué sur cette figure les performances d'erreur des codes ayant de grandes mémoires. Par exemple, le code $\left(m_{s_{RCDO}} = 205\right)$ RCDO terminable de taux de codage $r = 10/20$ offre des performances d'erreur identiques à celles du code $\left(m_{s_{LDPC-C}} = 2048\right)$ LDPC-C de taux de codage $r = 1/2$. Finalement, mentionnons qu'après 100 itérations, les performances d'erreur du code (205, 3, 6) RCDO terminable de taux de codage $r = 10/20$ se

situent à moins de 0.1 dB de la valeur du seuil théorique de convergence. À la Figure 5.10, nous avons reproduit les performances d'erreur, après 50 et 100 itérations de décodage, d'un code régulier (1024, 3, 6) LDPC-B de taux de codage $r = 1/2$ présentées dans [53] et [54] respectivement. Comme nous pouvons le constater, tous les codes RCDO terminables offrent des performances d'erreur supérieures à celles de tous les codes LDPC-B, et ce, pour la même complexité de processeur.

TABLEAU 5.1 : Paramètres des codes présentés aux Figures 5.9 et 5.10. Pour les codes RCDO et LDPC-C, leurs mémoires du code sont définies par $m_{s_{RCDO}}$ et $m_{s_{LDPC-C}}$ respectivement. Alors que, pour le code LDPC-B, N_{LDPC-B} désigne la longueur de bloc du code.

Codes	Taux de codage	Itérations	Références
$(m_{s_{RCDO}} = 31)$ RCDO	4/8	50	
$(m_{s_{LDPC-C}} = 128)$ LDPC-C	1/2	50	[53]
$(m_{s_{RCDO}} = 50)$ RCDO	10/20	100	
$(m_{s_{LDPC-C}} = 512)$ LDPC-C	1/2	100	[54]
$(m_{s_{RCDO}} = 205)$ RCDO	10/20	100	
$(m_{s_{LDPC-C}} = 2048)$ LDPC-C	1/2	100	[36]
$(m_{s_{RCDO}} = 127)$ RCDO	4/8	50	
$(N_{LDPC-B} = 1024)$ LDPC-B	1/2	50	[53]
$(m_{s_{RCDO}} = 49)$ RCDO	10/20	100	
$(N_{LDPC-B} = 1024)$ LDPC-B	1/2	100	[54]

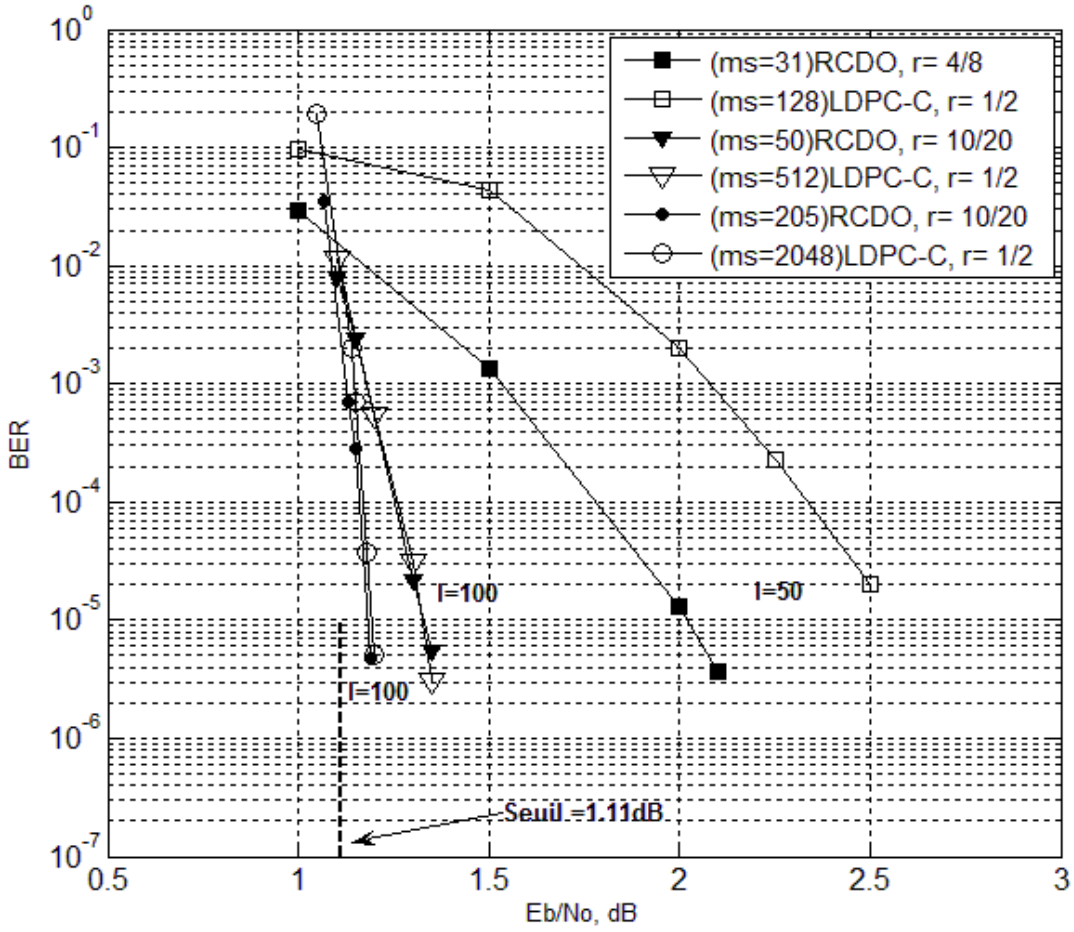


FIGURE 5-9 : Comparaison des performances obtenues par un nombre de codes RCDO terminables et codes LDPC convolutionnels. Le seuil théorique de convergence des codes convolutionnels réguliers $(3, 6)$ est également présenté.

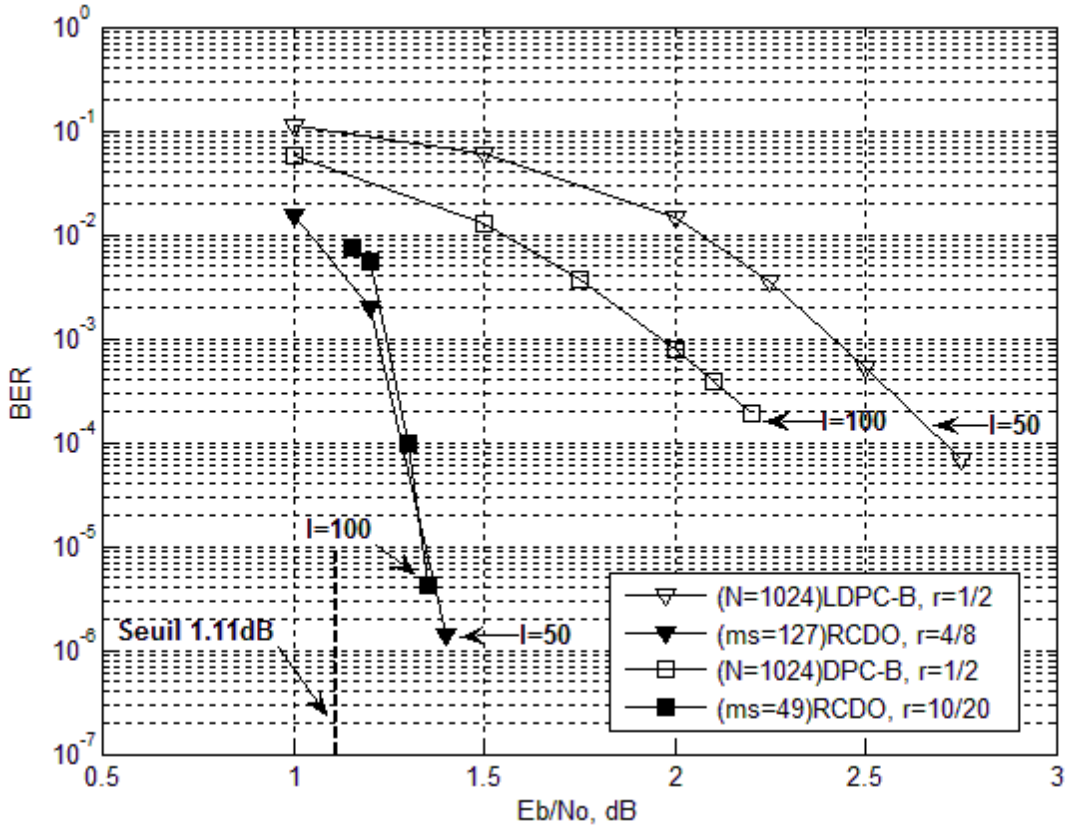


FIGURE 5-10 : Comparaison des performances obtenues par un nombre de codes RCDO terminables et codes LDPC en blocs. Le seuil théorique de convergence des codes réguliers $(3, 6)$ est également présenté.

5.3 Conclusion

Dans cette section, nous avons présenté les résultats de simulations de l'implémentation des codes convolutionnels récursifs multi-registres RCDO terminables conçus dans le cadre de ce projet. À la première partie de ce chapitre, nous avons étudié les performances d'erreur des codes RCDO terminés lors d'une transmission basée sur des trames de longueurs variables. D'une part, nous avons vu que les codes RCDO terminés souffrent d'une perte du taux de codage à cause de la séquence de terminaison. D'autre part, nous avons montré que ces codes RCDO terminés bénéficient d'une amélioration de la correction d'erreurs grâce à l'initialisation et la terminaison de leurs décodeurs et leurs codeurs terminables. En effet, les résultats de simulations montrent que la perte du SNR causée par la terminaison est à peine de quelques dixièmes de décibel

lorsque la longueur de la trame est supérieure à dix fois la longueur de la contrainte du code. Par conséquent, cette faible perte du SNR peut être considérée comme étant le coût du traitement des trames de longueurs variables.

En plus d'avoir étudié les performances d'erreur obtenues par les codes RCDO terminés lors d'une transmission par paquets, nous avons aussi présenté les performances des codes RCDO terminables utilisés dans le contexte d'une transmission en continu. Des simulations ont été effectuées afin de comparer les codes RCDO terminables aux codes RCDO non terminables, aux codes LDPC convolutionnels et aux codes LDPC en blocs. Nos résultats de simulation nous montrent que les conditions de terminaison imposées aux connexions des codeurs RCDO permettent l'amélioration des performances d'erreur lorsque nous les comparons avec les codes RCDO pour lesquels les connexions ne respectent que les conditions de la double orthogonalité. De plus, nous avons effectué des simulations de comparaison entre les codes RCDO terminables proposés dans ce mémoire et les codes LDPC. Deux constats peuvent donc être effectués à partir des résultats obtenus. Premièrement, pour un délai de décodage et une mémoire de stockage égaux, les codes RCDO terminables offrent pratiquement les mêmes performances d'erreur obtenues par les codes LDPC convolutionnels. Deuxièmement, les codes RCDO terminables permettent d'atteindre, pour une même complexité de processeur (i.e., $N_{LDPC-B} = c \cdot (m_{s_{RCDO}} + 1)$), de meilleures performances d'erreur que tous les codes LDPC en blocs.

CHAPITRE 6

CONCLUSION

6.1 Bilan des travaux réalisés

Bien que les codes RCDO puissent offrir des performances d'erreur qui approchent la capacité de Shannon, ces codes font face à plusieurs problèmes qui n'ont pas encore été résolus. Le travail de recherche effectué dans ce projet a été ciblé de façon à exploiter les avantages et à résoudre les problèmes associés aux codes RCDO lorsqu'ils sont appliqués à des trames de longueurs finies et variables. Des contributions ont été apportées dans divers aspects de la conception des codes RCDO terminés, tels que la construction du code, le codeur et la comparaison des performances d'erreur des codes RCDO terminables à celles des codes LDPC-B et LDPC-C.

Codeur RCDO et procédure de la terminaison

Le premier aspect des contributions comprend une compréhension plus approfondie de la conception du codeur RCDO ainsi que des nouveaux algorithmes et procédures afin de gérer la terminaison des codes convolutionnels RCDO. Tout d'abord, les codeurs non terminés sont étudiés. Ces codeurs sont spécifiés entièrement par leur matrice de contrôle de parité présentée sous ses deux formes : la forme polynômiale $\mathbf{H}^T(D)$ et la forme binaire $\mathbf{H}^T$. Nous avons décrit la méthode de construction des codeurs RCDO selon une approche algébrique, où des conditions de la double orthogonalité ont été imposées sur les positions de connexions définissant la structure du codeur dans le but d'éliminer les cycles courts dans le graphe de Tanner des codes RCDO. Plus précisément, les conditions de la double orthogonalité permettent d'éliminer les cycles de longueurs 4, 6 et 8 dans le graphe de Tanner associé au code.

Nous nous sommes intéressés à la terminaison des codes RCDO lorsqu'ils sont appliqués à des trames de longueurs finies, ce qui constitue le levier essentiel pour atteindre leur avantage relatif à la flexibilité de gérer des trames de longueurs variables. Nous avons donc obtenu la bonne séquence de terminaison en résolvant un système d'équations linéaires binaires. Nous avons montré que la séquence de terminaison peut causer une perte du taux de codage. Cette perte est

importante pour les trames de petites longueurs. Les performances d'erreur des codes RCDO terminés ont été également étudiées par des simulations sur ordinateur. Les résultats obtenus, nous montrent que grâce à l'amélioration de la protection contre les erreurs apportée aux deux extrémités de la trame, la perte du SNR par rapport à un code RCDO non terminé est bien inférieure à celle causée par la perte du taux de codage. Dans le but d'intégrer la terminaison dans la conception du codeur, nous avons proposé une nouvelle technique capable de générer graduellement et en continu les bits de terminaison, et ce, avec une latence qui nécessite $\mathcal{O}({m_s}^2)$ unités de temps et une mémoire d'ordre $\mathcal{O}(m_s)$ éléments de stockage. La réduction de la mémoire du code m_s peut donc s'avérer indispensable pour minimiser la complexité associée à la terminaison.

Construction des codes RCDO terminables

Le deuxième élément de contributions repose essentiellement sur l'algorithme de construction des codes RCDO terminables à temps-invariant. Considérant la solution proposée de la terminaison des codes RCDO, nous avons pu définir des conditions supplémentaires, appelées les *conditions de la terminaison*, permettant ainsi de terminer le code par une séquence de bits de longueur la plus petite possible (i.e. $L \approx m_s$). Ces conditions ont été introduites dans la matrice de contrôle de parité binaire $\mathbf{H}^T$ par l'utilisation des sous-matrices de permutations et dans la matrice de vérification polynômiale $\mathbf{H}^T(D)$ en plaçant les connexions $\alpha_{m,n}$ sous une forme diagonale. À partir des conditions de la terminaison, nous avons apporté les modifications nécessaires à l'algorithme original de construction des codes RCDO non terminables, tout en satisfaisant les conditions de la double orthogonalité. L'algorithme proposé permet de construire des codes RCDO terminables réguliers et irréguliers avec plusieurs paramètres comme la mémoire du code m_s, le taux de codage $r = b/c$ et la paire de distributions $\big(\lambda(x), \rho(x)\big)$. Les résultats des recherches effectuées par cet algorithme montrent que la mémoire m_s des codes RCDO terminables a été davantage réduite par rapport aux codes non terminables.

Comparaison des codes RCDO terminables aux codes LDPC en blocs et aux codes LDPC convolutionnels

Dans ce travail de recherche, nous avons, pour la première fois, comparé en détails les performances d'erreur des codes RCDO aux codes LDPC-B et aux codes LDPC-C. La

comparaison tient compte des résultats de recherche obtenus dans ce mémoire, notamment les connexions du codeur RCDO terminable qui satisfont les conditions de terminaison. Par rapport aux codes LDPC-B, les codes RCDO terminés ont la capacité de traiter des trames de longueurs variables, mais cet avantage est obtenu au prix d'une perte du SNR. Malgré cette perte, les codes RCDO terminés offrent des performances d'erreur bien meilleures que celles obtenues par un code LDPC-B ayant une complexité d'encodage plus élevée.

En termes de complexité liée au décodage itératif, nous avons comparé les performances d'erreur des codes RCDO terminables à celles obtenues par les codes LDPC-B et LDPC-C, et ce, dans le contexte d'une transmission en continu. Nous avons constaté que pour le même nombre d'unités de mémoires nécessaire à la réalisation du décodage itératif ainsi que la même latence, les codes RCDO terminables offrent pratiquement les mêmes performances d'erreur obtenues par les codes LDPC convolutionnels. Cependant, pour une même complexité de processeur, les codes RCDO terminables permettent d'atteindre de meilleures performances d'erreur que tous les codes LDPC-B comparables. Par conséquent, les codes RCDO terminables proposés dans ce projet de recherche méritent d'être davantage étudiés, investigués et analysés.

6.2 Améliorations envisageables et futurs axes de recherche

À partir des résultats présentés dans ce mémoire, nous attirons l'attention du lecteur vers des futurs travaux de recherche dans les axes suivants.

Codes convolutionnels RCDO à temps-variant

Dans notre étude, nous avons considéré seulement les codes convolutionnels récursifs terminables à temps-invariant, où les positions des connexions reliant les symboles à la sortie du codeur RCDO aux différents registres à décalage ne varient pas en fonction du temps. Dans [24], les auteurs proposent une classe beaucoup plus générale de codes convolutionnels, soit ceux à temps variant. Ainsi, nous envisageons une extension des codes convolutionnels récursifs doublement orthogonaux à temps variant périodiques, et ce, en faisant varier dans le temps les connexions du codeur convolutionnels.

Codes convolutionnels circulaires RCDO

Au Chapitre 4, nous avons mentionné que les codes convolutionnels circulaires (en anglais, *Tail-biting codes*) ont la capacité de coder des trames de longueurs finies et sont considérés dans la littérature comme étant une alternative à la terminaison des codes convolutionnels. Effectivement, il serait intéressant d'envisager la version circulaire des codes convolutionnels récursifs RCDO, puisque les codes convolutionnels circulaires ne nécessitent aucune séquence de terminaison pour terminer les trames et, par conséquent, ils permettent d'éviter la perte du taux de codage causée par la terminaison. Il serait également intéressant d'explorer les liens possibles entre les codes convolutionnels terminés et circulaires dans le but de mieux gérer les trames de petites longueurs.

Performances d'erreur des codes RCDO pour un canal à évanouissements

Dans notre recherche, nous nous sommes intéressés uniquement au canal AWGN. L'utilisation de codes RCDO terminés pourrait aussi s'avérer avantageuse pour des canaux à évanouissements. Ce type de canal a tendance à introduire des blocs d'erreur de sorte que les erreurs ne peuvent plus être considérées indépendantes les unes des autres. Pour remédier à ce problème, nous pouvons utiliser un entrelaceur adéquat à la sortie du codeur de canal et un délaceur approprié à l'entrée du décodeur. Par conséquent, l'évaluation des performances d'erreur des codes convolutionnels récursifs doublement orthogonaux terminés pour des canaux à évanouissements est d'un grand intérêt pour la plupart des applications dans des canaux de communication terrestres et pourrait faire l'objet de recherches futures.

BIBLIOGRAPHIE

[1] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal,* vol. 27, pp. 379-423, Juillet, 1948.

[2] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal,* vol. 27, pp. 623-656, Octobre, 1948.

[3] C. Berrou, A. Glavieux, et P. Thitimasjshima, “Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-codes (1),” *Proceedings of the IEEE International Conference on Communications*, pp.1064 -1070, 1993.

[4] R. B. Wells, *Applied coding and information theory for engineers*. New Jersey: Prentice Hall, 1999.

[5] R. G. Gallager, “Low-density parity-check codes,” *IRE Trans. Inform. Theory*, pp.21–28, Janvier 1962.

[6] D. J. C. MacKay and R. M. Neal, “Near shannon-limit performance of low-density parity-check codes,” *Electronics Letters*, vol. 32, pp.1645–1646, Août 1996.

[7] C. Cardinal, D. Haccoun, F. Gagnon et N. Batani, “Convolutional self-doubly orthogonal codes for iterative decoding without interleaving,” *Proceedings 1998 IEEE International Symposium on Information Theory*, MIT, pp.280, Août 1998.

[8] F. Gagnon, D. Haccoun, N. Batani, C. Cardinal, “Apparatus for Convolutional Self-Doubly Orthogonal Encoding and Decoding”, *US Patent No 61677552*, Octobre 1997.

[9] S. Lin et D. Costello, *Error Control Coding : Fundamentals and Applications.* Prentice Hall, Inc. Englewood Cliffs, New Jersey, 2004.

[10] R. M. Tanner, “A recursive approach to low complexity codes,” *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547, Septembre 1981.

[11] M. Luby, M. Mitzenmacher, M. A. Shokrollahi et D. Spielman, “Improved low-density parity-check codes using irregular graphs,” *IEEE Transactions on Information Theory*, vol.47, pp.585–598, Février 2001.

[12] T. Richardson, M. A. Shokrollahi et R. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Transactions on Information Theory*, vol.47, pp.619–637, Février 2001.

[13] P. Elias, “Error-free Coding,” *IRE Transactions on Information Theory*, vol. PGIT-4, pp.29–37, 1955.

[14] V. Branka et Y. Jinhong, *Turbo Codes: Principles and Applications*. Kluwer Academic Publishers, Mai 2000.

[15] S. Benedetto et G. Montorsi, “Unveiling Turbo Codes: Some Results On Parallel Concatenated Coding Shemes,” *IEEE Transactions on Information Theory*, vol.42, no.2, pp.409–428, Mars 1996.

[16] J. Hagenauer et P. Hoeher, “A Viterbi Algorithm With Soft-decision Outputs and Its Applications,” *IEEE Global Telecommunication Conference and Exhibition*, vol.3, pp.1680–1686, Novembre 1989.

[17] L. Bahl, J. Cocke, F. Jelinek et J. Raviv, “Optimal Decoding of Linear Codes For Minimizing Symbol Error Rate,” *IEEE Transactions on Information Theory*, vol.20, no.2, pp.284–287, Mars 1974.

[18] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. Spielman et V. Stemann, “Practical Loss-resilient Codes,” *Proc. 29th Annu.ACM Symp. Theory of Computing*, pp.150–159, 1997.

[19] C. Cardinal, *Décodage à Seuil Itératif des Codes Convolutionnels Doublement Orthogonaux.* Thèse de doctorat, École Polytechnique de Montréal, Montréal, 2001.

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

[21] C. Cardinal, D. Haccoun et Y. C. He, “Reduced-complexity Convolutional Self-doubly Orthogonal Codes for Efficient Iterative Decoding,” *IEEE 63rd Vehicular Technology Conference, VTC 2006-Spring.*, vol.3, pp.1372–1376, Mai 2006.

[22] E. Roy, *Recherche et Analyse de Codes Convolutionnels Doublement Orthogonaux Simplifiés au Sense Large.* Mémoire de maîtrise, École Polytechnique de Montréal, Août 2006.

[23] Y. C. He, D. Haccoun et C. Cardinal, “Error Performances of Multi Shift-register Convolutional Self-doubly-orthogonal Codes,” *IEEE Communications Letters,* vol.13, pp.685–687, 2009.

[24] A. Jiménez Feltström et K. S. Zigangirov, “Time-varying Periodic Convolutional Codes With Low-density Parity-check Matrix,” *IEEE transactions on Information Theory,* vol.45, no.6, pp.2181–2191, Septembre 1999.

[25] F. R. Kschischang, B. J. Frey et H. A. Loeliger, “Factor Graphs and The Sum-product Algorithm,” *IEEE Transactions on Information Theory,* vol.47, pp.498–519, Février 2001.

[26] T. J. Richardson et R. L. Urbanke, “The Capacity of Low-density Parity-check Codes Under Message-passing Decoding,” *IEEE Transactions on Information Theory,* vol.47, pp.599–618, Février 2001.

[27] E. Roy, *Étude Des Propriétés Des Codes Convolutionnels Récursifs Doublement-orthogonaux.* Thèse de doctorat, École Polytechnique de Montréal, Décembre 2012.

[28] E. Roy, C. Cardinal et D. Haccoun, “Recursive Convolutional Codes For Time-invariant LDPC Convolutional Codes,” *Proceedings (ISIT), 2010 IEEE International Symposium on Information Theory*, pp.834–838, 2010.

[29] Y.-C. He et D. Haccoun, “An Analysis of The Orthogonality Structures of Convolutional Codes For Iterative Decoding,” *IEEE Transactions on Information Theory*, vol. IT-51, pp. 3247–3261, Septembre 2005.

[30] J. Thorphe, “Low Density Parity Check (LDPC) Codes Constructed From Protographs ,” *JPL INP Progress Report*, pp. 42–154, 2003.

[31] X. -Y. Hu, E. Eleftheriou et D. Arnold, “Regular and Irregular Progressive Edge-growth Tanner Graphs,” *IEEE Transactions on Information Theory*, vol. 51, no. 6 pp. 386–398, Janvier 2005.

[32] C. Cardinal, Y. –C. He et D. Haccoun, “A New Approach For The Construction of Powerful LDPC Convolutional Codes,” *Vehicular Technology Conference, Spring 2008. IEEE*, pp. 1176–1180, Mai 2008.

[33] O. Y. Takeshita, “On Maximum Contention-free Interleavers And Permutation Polynomials Over Integar Rings,” *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1249–1253, Mars 2006.

[34] D. Divsalar, S. Dolinar et C. Jones, “Construction of Protograph LDPC Codes With Linear Minimum Distance,” *IEEE International Symposium on Information Theory*, pp. 664–668, Seattle, WA, USA, Juillet 2006.

[35] A. Sridharan, D. Truhachev, M. Lentmaier, D. J. Castello et K. S. Zigangirov, “On The Free distance of LDPC Convolutional Codes,” *IEEE International Symposium on Information Theory (ISIT)*, Chicago, IL, USA, Juin 2004.

[36] A. Sridharan et D. J. Castello, “A New Construction Method For Low Density Parity Check,” *IEEE Information Theory Workshop*, pp. 212, Bangalore, India, Octobre 2002.

[37] M. P. Fossorier, “Quasi-cyclic Low-density Parity-check Codes From Circulant Permutation Matrices,” *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1788–1793, Août 2004.

[38] Y. Kou, S. Lin et M. P. Fossorier, “Low-density Parity-check Codes Based On Finite Geometries: A Rediscovery And New Results,” *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711–2736, Novembre 2001.

[39] A. Nemr, C. Cardinal, M. Sawan et D. Haccoun, “Very High Throughput Iterative Threshold Decoder For Convolutional Self-doubly Orthogonal Codes,” *Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008. 2008 Joint 6th International IEEE Northeast Workshop*, pp. 257–260, 2008.

[40] T. Richardson et R. Urbanke, *Modern Coding Theory*. University Press, Cambridge, 2008.

[41] C. Cardinal, D. Haccoun et F. Gagnon, “Iterative Threshold Decoding Without Interleaving For Convolutional Self-doubly Orthogonal Codes,” *IEEE Transactions on Communications*, vol. 51, pp. 1274–1282, Août 2003.

[42] K. Gracie et M. Hamon, “Turbo and Turbo-Like Codes: Principles and Applications in Telecommunications,” *Proceedings of the IEEE*, vol. 95, pp. 1228–1254, Juin 2007.

[43] X. Hu, E. Eleftheriou et D. M. Arnold, “Irregular Progressive Edge-growth Tanner graphs,” *Proceedings of the IEEE International Symposium on Information Theory*, Lausanne, Switzerland, 2002.

[44] X. Hu, E. Eleftheriou et D. M. Arnold, “Regular and Irregular Progressive Edge-growth Tanner graphs,” *IEEE Transactions on Information Theory*, vol. 51, pp.386–398, Janvier, 2005.

[45] S. Chung, G. Forney, T. Richardson et R. Urbanke, “On the Design of Low-density Parity-check Codes Within 0.0045 dB of the Shannon Limit,” *IEEE Communications Letters*, vol. 5, no. 2, pp.58–60, Février, 2001.

[46] H. Ma et J. Wolf, “On Tail Biting Convolutional Codes,” *IEEE Transactions on Communications*, vol. COM-34, no. 2, pp.104–111, Février, 1986.

[47] J. Anderson et S. Hladik, “Tail-biting MAP Decoders,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp.279–302, Février, 1998.

[48] J. Hokfelt, O. Edfors et T. Maseng, “On the Theory and Performance of Trellis Termination Methods For Turbo Codes,” *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp.838–846, Mai 2001.

[49] The Institute of Electrical and Electronics Engineers, *IEEE std 802.3-2002*. The IEEE, 2002.

[50] C. Meyer, *Matrix Analysis and Applied Linear Algebra.* Society for Industrial and Applied Math, 2001.

[51] S. Bates, Z. Chen et X. Dong, “Low-density Parity Check Convolutionnal Codes for Ethernet Networks,” *Proceedings IEEE Pacific Rim Conf. Commun., Comput., Signal Process.*, pp.86–91, Août, 2005.

[52] A. Sridharan, *Design and Analysis of LDPC Convolutional Codes.* Thèse de doctorat, Université de Notre Dame, Mai 2005.

[53] Z. Chen, S. Bates et X. Dong, “Low-density Parity-check Convolutional Codes Applied to Packet Based Communication Systems,” *Proceedings IEEE GlobeCom'05,* Novembre. 2005.

[54] S. Bates , Z. Chen , L. Gunthorpe , A. E. Pusane , K. Sh. Zigangirov et D. J. Costello Jr., “A Low-cost Serial Decoder Architecture For Low-density Parity-check Convolutional Codes,” *IEEE Trans. Circuits Syst. I*, vol. 55, no. 7, pp.1967–1976, Août 2008.

[55] IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, *IEEE std. 802.3an*, Septembre 2006.

[56] R. Seifert, *The Switch Book*, John Wiley & Sons, Inc., 2000.

[57] W. Rutherford, L. Jorgenson, M. Siegert, P. V. Epp et L. Liu, “16000-64000 B pMTU Experiments with Simulation: The Case for Super Jumbo Frames at Supercomputing ’05,” *Optical Switching and Networking*, vol. 4, no. 2, pp. 121–130, Juin 2007.

ANNEXE A

CODES CONVOLUTIONNELS RÉCURSIFS DOUBLEMENT ORTHOGONAUX NON TERMINÉS

A.1 Codage et décodage des codes RCDO non terminés appliqués aux trames de longueurs prédéfinies

Une alternative possible pour terminer la fin de d'une trame est de simplement arrêter la transmission des bits d'information et des bits de parité, et d'envoyer le marqueur DFT approprié. Tous les bits d'information suivants peuvent être considérés comme nuls. Cependant, les bits de parité suivants sont inconnus puisque le codeur n'a pas été ramené vers un état connu (i.e. la terminaison n'a pas été employée). Ceci se traduit au niveau du décodeur par l'attribution aux LRVs correspondant à ces bits d'information la valeur infinie, par contre les LRVs associés aux bits de parité doivent être affectés à la valeur zéro. Il en résulte que le décodeur RCDO non terminé souffre d'une faible fiabilité lors du décodage des derniers G bits d'information. Toutefois, lorsque la valeur G est très petite par rapport à la longueur de la trame, il devient possible d'ignorer ces bits d'information. Notons que ces G bits d'information non fiables sont appelés *intervalle de garde*.

Par exemple, pour le code $(49, 3, 6)$ RCDO non terminé de taux de codage $r = 10/20$ présenté à la Figure 5.2, plus de 1500 bits à la fin d'une trame de longueur 12250 bits souffrent d'un taux de bits erronés plus élevé que la normale, et ce, après avoir effectué 20 itérations de décodage à un rapport signal sur bruit $E_b/N_0 = 1.3$ dB. Dans le but de compenser cette faiblesse, plus de 1500 bits de garde sont nécessaires pour les injecter à la fin de la trame de données, c'est-à-dire que la longueur de l'intervalle de garde non codé est d'environ $4bm_s$ bits, ce qui est toujours plus grande que la longueur de la séquence de terminaison (i.e. $(m_s + 1) \cdot b$). Comme nous l'avons montré à la Section 5.1.1, les trames codées à l'aide d'un codeur RCDO terminé profitent d'une amélioration de protection contre les erreurs à la fin de la trame. En comparaison avec la terminaison, l'utilisation d'un intervalle de garde entraîne des gaspillages en termes de

consommation d'énergie et n'offre pas une meilleure performance. Par conséquent, les intervalles de garde ne peuvent pas être considérés comme une solution pratique.

ANNEXE B

COMPARAISON DES COMPLEXITÉS ASSOCIÉES AU DÉCODAGE ITÉRATIF DES CODES RCDO, LDPC CONVOLUTIONNELS ET LDPC EN BLOCS

Dans cette annexe, nous comparons plusieurs aspects de décodage des codes RCDO aux codes LDPC convolutionnels et aux codes LDPC en blocs. Nous analysons la complexité des processeurs, le délai de décodage et le nombre d'unités de mémoires nécessaires à la réalisation des différents décodeurs itératifs simulés dans ce mémoire.

B.1 Complexité des processeurs pour le décodage des codes LDPC en blocs et des codes convolutionnels RCDO

À la Section 3.5, nous avons montré que la fenêtre de décodage d'un code convolutionnel RCDO de taux de codage $r = b/c$ et d'une mémoire m_s est capable de traite $I \cdot \vartheta = I \cdot c(m_s + 1)$ symboles. Cependant, le décodage itératif des codes RCDO est composé par une chaîne de processeurs identiques et indépendantes où le nombre de processeurs correspond au nombre d'itérations effectuées. En effet, le décodeur itératif RCDO est simple à réaliser puisque seulement un seul processeur capable de traiter $\vartheta = c(m_s + 1)$ symboles doit être conçu pour construire l'ensemble du décodeur RCDO. Pour un code en blocs LDPC ayant une longueur de bloc égale à N_{LDPC-B}, le décodeur est composé d'un seul processeur capable d'effectuer le décodage itératif de tous les N_{LDPC-B} symboles. Par conséquent, pour avoir la même complexité de processeur, la longueur de bloc d'un code LDPC-B doit être égale à la longueur de contrainte du code RCDO, d'où $N_{LDPC-B} = c(m_s + 1)$.

B.2 Mémoire nécessaire à la réalisation du décodeur itératif RCDO et LDPC-C

Afin de calculer l'information extrinsèque des symboles à la sortie de tous les processeurs constituant le décodeur itératif, nous avons besoin d'enregistrer un certain nombre de valeur. En nous référant à l'équation (3.21), nous pouvons constater que nous devons enregistrer en mémoire les messages provenant du canal $\mathrm{m}^{(0)}_{v^{(i)}_{t-m_s}}$, où $i \in \{1, \dots, c\}$ et m_s désigne la mémoire du code convolutionnel. C'est-à-dire que pour les I itérations effectuées lors du décodage nous devons donc avoir $I \cdot c(m_s + 1)$ unités de mémoire pour stocker ces valeurs réelles. De plus, nous avons besoin de stocker en mémoire tous les messages traversant chaque arête du graphe de Tanner associé au code convolutionnel. Le nombre total de mémoires nécessaires pour enregistrer les valeurs extrinsèques échangées entre les nœuds variables et les nœuds de contraintes est égal à $I \cdot \Gamma \cdot (m_s + 1)$, où $\Gamma = d_\lambda \cdot c$ désigne le nombre d'arêtes du graphe de Tanner. Par conséquent, lorsque nous effectuons I itérations lors du décodage des codes RCDO et LDPC-C, l'ensemble des processeurs utilisés nécessite un total de $I \cdot (d_\lambda + 1) \cdot c(m_s + 1)$ unités de mémoires.

B.3 Délai de décodage des codes convolutionnels RCDO et LDPC-C

Soit T_s représente le temps qui sépare deux symboles successifs, c'est-à-dire que le débit symbole est égal à $1/T_s$. Alors, pour les décodeurs des codes convolutionnels RCDO et LDPC-C simulés dans ce mémoire, le délai de décodage dépend du nombre de processeurs constituant le décodeur itératif ainsi que du nombre de symboles que nous devons enregistrer afin de calculer les valeurs extrinsèques des symboles décodés. Il en résulte que la durée maximale entre l'arrivée d'un symbole et la décision finale prise par le I −ème processeur est égale à $(I \cdot c(m_s + 1)) \cdot T_s$. Par conséquent, pour assurer une comparaison équitable entre les codes RCDO et les codes LDPC en termes de délai de décodage, nous devons satisfaire l'équation suivante $I \cdot c\left(m_{s_{RCDO}} + 1\right) = I \cdot c\left(m_{s_{LDPC-C}} + 1\right)$, tout en supposant que les temps symbole $T_{s_{LDPC-C}}$ et $T_{s_{RCDO}}$ sont égaux.