THÈSE PRÉSENTÉE
POUR OBTENIR LE GRADE DE

DOCTEUR DE L'UNIVERSITÉ DE BORDEAUX

École Doctorale des Sciences de l'Ingénieur
Spécialité : Électronique

par **Thibaud Tonnellier**

Contribution à l'amélioration des performances de décodage des turbo codes : algorithmes et architecture

Directeur de thèse : Christophe Jégo
Co-encadrants de thèse : Bertrand Le Gal
Camille Leroux

préparée au Laboratoire IMS
en collaboration avec Thales Alenia Space

soutenue le 5 Juillet 2017

Jury :

Jean-Pierre Cances	-	Professeur des Universités	-	Université de Limoges	*Rapporteur*
Michel Jézéquel	-	Professeur de l'IMT	-	IMT Atlantique	*Rapporteur*
Charly Poulliat	-	Professeur des Universités	-	INP Toulouse	*Président*
Nicolas Van Wambeke	-	Docteur Ingénieur	-	Thales Alenia Space	*Examinateur*
Bertrand Le Gal	-	Maître de Conférences	-	Bordeaux INP	*Co-encadrant*
Camille Leroux	-	Maître de Conférences	-	Bordeaux INP	*Co-encadrant*
Christophe Jégo	-	Professeur des Universités	-	Bordeaux INP	*Directeur*

Thèse réalisée au Laboratoire de l'Intégration du Matériau au Système (IMS)
de Bordeaux, au sein de l'équipe CSN du groupe Conception.

Université de Bordeaux, Laboratoire IMS
UMR 5218 CNRS - Bordeaux INP
351 Cours de la Libération
Bâtiment A31
33405 Talence Cedex

Thèse réalisée en collaboration avec Thales Alenia Space

26 Avenue Jean François Champollion
31100 Toulouse

A mes parents, à ma compagne.

Remerciements

Je tiens en premier lieu à remercier les membres du jury pour l'intérêt qu'ils ont porté à ces travaux. Je remercie donc très chaleureusement Monsieur Jean-Pierre Cances, professeur de l'Université de Limoges et Monsieur Michel Jézéquel, professeur de l'Institut Mines-Télecom Atlantique de m'avoir fait l'honneur de rapporter ce travail de thèse. Je remercie vivement Monsieur Charly Poulliat, professeur de l'Institut National Polytechnique de Toulouse d'avoir présidé le jury.

Merci à Messieurs Benjamin Gadat et Nicolas Van Wambeke pour nos échanges et vos encadrements dans le cadre de vos fonctions respectives au sein de Thales Alenia Space.

Je remercie grandement Bertrand, Camille et Christophe qui ont su répondre à mes (trop ?) nombreuses sollicitations durant ces années.
Bertrand, merci pour ton soutien sans faille et ton dynamisme quotidien exemplaire, j'ai particulièrement apprécié travailler à tes côtés.
Camille, merci pour l'ensemble de nos moments passés ensemble ; que ce soit au sein du laboratoire, lors de nos enseignements matinaux ou enfin en dehors de tout ceci.
Christophe, merci d'avoir toujours fait en sorte de mettre en place le cadre de travail le plus agréable possible, merci pour tes relectures en première ligne et ton animation d'équipe de haut-vol.

Merci à tous mes collègues d'avoir contribué à l'excellente ambiance de travail. Tout d'abord, l'ensemble de l'équipe CSN et plus généralement les membres du groupe Conception. Ainsi, merci Jérémie, Dominique, Jean-Baptiste, Adrien, Camilo, Guillaume B., Guillaume D., Imen, Jonathan, Mathieu, Olivier, Vincent, Yann. Merci aussi à François, Nathalie et Yann. Enfin, un grand merci à Raphaël, sans qui ces années au laboratoire auraient été fondamentalement différentes.
Au niveau du groupe Signal et Image, merci à Guillaume F. et à Romain pour nos nombreux échanges.

Je tiens à remercier tout particulièrement Arnaud pour m'avoir fait découvrir le monde de la recherche, sans qui il ne me serait venu l'envie de réaliser une thèse.

Finalement, je remercie amoureusement Fanny qui a su me supporter et tolérer des horaires de retour bien trop tardifs ainsi qu'une disponibilité dans notre vie commune bien trop faible.

Résumé

Les turbo codes sont une classe de codes correcteurs d'erreurs approchant la limite théorique de capacité formulée par Claude Shannon. Conjointement à leurs excellentes performances de décodage, la complexité calculatoire modérée des turbo décodeurs a permis leur inclusion dans de nombreux standards de communications numériques.

Une des métriques permettant la caractérisation de codes correcteurs d'erreurs est l'évolution du taux d'erreurs binaires en fonction du rapport signal sur bruit. Dans le cadre des turbo codes, une courbe de performance de décodage comprend deux zones principales. Dans la première zone, une faible amélioration de la qualité du canal de transmission entraîne de grandes améliorations au niveau des performances de décodage. En revanche dans la seconde, une amélioration de cette qualité ne résulte qu'en une amélioration marginale des performances de décodage. Cette seconde région est nommée zone du plancher d'erreurs. Elle peut empêcher l'utilisation de turbo codes dans des contextes nécessitant de très faibles taux d'erreurs. C'est pourquoi la communauté scientifique a proposé différentes optimisations favorisant la construction de turbo codes atténuant ce plancher d'erreurs. Cependant, ces approches ne peuvent être considérées pour des turbo codes déjà standardisés.

Dans ce contexte, cette thèse adresse le problème de la réduction du plancher d'erreurs en s'interdisant de modifier la chaîne de communications numériques du côté de l'émetteur. Pour ce faire, un état de l'art de méthodes de post-traitement de décodage est dressé pour les turbo codes. Il apparaît que les solutions efficaces sont coûteuses à mettre en œuvre car elles nécessitent une multiplication des ressources calculatoires ou impactent fortement la latence globale de décodage.

Dans un premier temps, deux algorithmes basés sur une supervision de l'évolution de métriques internes aux décodeurs, sont proposés. L'un deux permet d'augmenter la convergence du turbo décodeur. L'autre ne permet qu'une réduction marginale du plancher d'erreurs. Dans un second temps, il est observé que dans la zone du plancher d'erreurs, les trames décodées par le turbo décodeur sont très proches du mot de code originellement transmis. Ceci est démontré par une proposition de prédiction analytique de la distribution du nombre d'erreurs binaires par trame erronée. Cette dernière est réalisée grâce au spectre de distance du turbo code. Puisque ces erreurs binaires responsables du plancher d'erreurs sont peu nombreuses, une métrique permettant de les identifier est mise en œuvre. Ceci mène alors à l'établissement d'un algorithme de décodage permettant

de corriger des erreurs résiduelles. Cet algorithme, appelé algorithme Flip-and-Check se base sur un principe de création de mots candidats et de vérifications successives par un code détecteur d'erreurs. Grâce à cet algorithme de décodage, un abaissement du plancher d'erreurs d'un ordre de grandeur est obtenu pour les turbo codes de différents standards (LTE, CCSDS, DVB-RCS et DVB-RCS2), ce, tout en conservant une complexité calculatoire raisonnable.

Finalement, une architecture matérielle de décodage implémentant l'algorithme Flip-and-Check est présentée. Une étude préalable de l'impact des différents paramètres de l'algorithme est menée. Elle aboutit à la définition de valeurs optimales pour certains de ces paramètres. D'autres sont à adapter en fonction des gains visés en terme de performances de décodage. Cette architecture démontre alors la possible intégration de cet algorithme aux turbo décodeurs existants ; permettant alors d'abaisser le plancher d'erreurs des différents turbo codes présents dans les différents standards de télécommunication.

Mots clefs : Turbo-codes, Erreurs résiduelles, Post-traitement, Implémentation matérielle

Abstract

Since their introduction in the 90's, turbo codes are considered as one of the most powerful error-correcting code. Thanks to their excellent trade-off between computational complexity and decoding performance, they were chosen in many communication standards.

One way to characterize error-correcting codes is the evolution of the bit error rate as a function of signal-to-noise ratio (SNR). The turbo code error rate performance is divided in two different regions : the waterfall region and the error floor region. In the waterfall region, a slight increase in SNR results in a significant drop in error rate. In the error floor region, the error rate performance is only slightly improved as the SNR grows. This error floor can prevent turbo codes from being used in applications with low error rates requirements. Therefore various constructions optimizations that lower the error floor of turbo codes has been proposed in recent years by scientific community. However, these approaches can not be considered for already standardized turbo codes.

This thesis addresses the problem of lowering the error floor of turbo codes without allowing any modification of the digital communication chain at the transmitter side. For this purpose, the state-of-the-art post-processing decoding method for turbo codes is detailed. It appears that efficient solutions are expensive to implement due to the required multiplication of computational resources or can strongly impact the overall decoding latency.

Firstly, two decoding algorithms based on the monitoring of decoder's internal metrics are proposed. The waterfall region is enhanced by the first algorithm. However, the second one marginally lowers the error floor. Then, the study shows that in the error floor region, frames decoded by the turbo decoder are really close to the word originally transmitted. This is demonstrated by a proposition of an analytical prediction of the distribution of the number of bits in errors per erroneous frame. This prediction rests on the distance spectrum of turbo codes. Since the appearance of error floor region is due to only few bits in errors, an identification metric is proposed. This lead to the proposal of an algorithm that can correct residual errors. This algorithm, called Flip-and-Check, rests on the generation of candidate words, followed by verification according to an error-detecting code. Thanks to this decoding algorithm, the error floor of turbo codes encountered in different standards (LTE, CCSDS, DVB-RCS and DVB-RCS2) is lowered by one order of magnitude. This performance improvement is obtained without considering an important

computational complexity overhead.

Finally, a hardware decoding architecture implementing the Flip-and-Check algorithm is presented. A preliminary study of the impact of the different parameters of this algorithm is carried out. It leads to the definition of optimal values for some of these parameters. Others has to be adapted according to the gains targeted in terms of decoding performance. The possible integration of this algorithm along with existing turbo decoders is demonstrated thanks to this hardware architecture. This therefore enables the lowering of the error floors of standardized turbo codes.

Key words : Turbo codes, Residual Errors, Post-processing, Hardware architecture

Table des matières

Table des figures

Liste des tableaux

Liste des notations

α	Métrique d'état (ou de nœud) aller
$\arg\max(\cdot)$	Nombre de mots de code de poids de Hamming d
A_d	Nombre de mots de code de poids de Hamming d
β	Métrique d'état (ou de nœud) retour
B	Degré de parallélisme de sous-bloc
d_{min}	Distance minimale du code
E_b	Énergie moyenne par bit d'information
$\mathbf{c}$	Mot de code
C	Capacité du canal de transmission
$\hat{\mathbf{d}}$	Mot décidé
$\mathrm{erfc}(\cdot)$	Fonction d'erreur complémentaire
γ	Métrique de branche
$H(X)$	Entropie de la variable aléatoire X
$H(X,Y)$	Entropie conjointe des variables aléatoires X et Y
$H(X\|Y)$	Entropie conditionnelle de la variable aléatoire X sachant Y
K	Taille du mot d'information non codé
$L(\cdot)$	Valeur LLR
$\max^*(\cdot,\cdot)$	Opérateur max-étoile
ν	Nombre de mémoires d'un code convolutif
$N \rightarrow N$	Oscillation dans le domaine naturel entre deux itérations
$N \rightarrow I$	Oscillation entre le domaine naturel et le domaine entrelacé
N_0	Densité spectrale mono-latérale du bruit
N	Taille du mot d'information codé
Π	Fonction d'entrelacement
q	Taille du vecteur des positions les moins fiables pour l'algorithme FNC
R	Rendement du code ($R = K/N$)
σ	Variance du bruit
$\mathrm{sgn}(\cdot)$	Fonction signe
W_d	Somme des poids de Hamming des A_d séquences de poids de Hamming d
W	Nombre de fenêtres pour un SISO avec fenêtre glissante

Liste des acronymes

APP	A Posteriory Probability
ARP	Almost Regular Permutation
ARQ	Automatic Repeat-reQuest
AWGN	Additive White Gaussian Noise
BCJR	Algorithme de décodage MAP nommé d'après ses inventeurs : Bahl, Cocke, Jelinek et Raviv
BE	Bit Error, nombre d'erreurs binaires
BER	Bit Error Rate
BF	Backward-Forward
BFLY	Butterfly
BPSK	Binary Phase-Shift Keying
CCSDS	Consultative Committee for Space Data Systems
CIM	Correction Impulse Method
CRC	Cyclic Redundancy Check
DRP	Dithered Relatively Prime
DVB-RCS	Digital Video Broadcasting - Return Channel via Satellite
EML-MAP	Enhanced max log - Maximum A Posteriori
FB	Forward-Backward
FDM	Fonction De Masse
FE	Frame Error
FER	Frame Error Rate
FNC	Flip and Check
FSM	Forced Symbol Method
LDPC	Low Density Parity Check Codes
LIFO	Last-In First-Out
LLR	Log Likelihood Ratio
LL	Log Likelihood
LTE	Long Term Evolution
LVA	List Viterbi Algorithm
MAP	Maximum A Posteriori
ML	Maximum Likelihood : maximum à vraisemblance
OSC	Oscillations
OSD	Ordered Statistic Decoding
QPP	Quadratic Permutation Polynomial
QPSK	Quadrature Phase-Shift Keying
RAM	Random Access Memory

RSC	Recursive Systemactic Convolutionnal
SB	Sub-Block
SC	Self-Corrected
SISO	Soft Input Soft Output
SNR	Signal-to-Noise Ratio
SOVA	Soft Output Viterbi Algorithm
SW	Sliding-Window
WEF	Weight Enumerating Function

Introduction

Les travaux développés dans le cadre de cette thèse portent sur la proposition de méthodes et la définition d'architectures associées permettant l'amélioration des performances de décodage des turbo codes. Tout d'abord, le contexte des codes correcteurs d'erreurs est dressé. Ceci mène à la problématique traitée tout au long de ce manuscrit. L'agencement du manuscrit est ensuite détaillé. Enfin, les contributions réalisées au cours des travaux de thèse sont énoncés.

Contexte et problématique

L'année 2016 marque le centenaire de la naissance de Claude Shannon. Ingénieur en génie électrique et mathématicien, il est le père de la théorie de l'information. L'origine de cette discipline – se plaçant à l'intersection des mathématiques, des statistiques, du traitement du signal et de l'électronique – remonte à la publication de l'article fondateur "A mathematical Theory of Communication" [1], paru en 1948 dans la revue interne des laboratoires Bell. Quelques mois auparavant, au même endroit, le transistor est inventé par John Bardeen, Walter Brattain et William Shockley. Ces deux avancées scientifiques, concomitantes, ont en moins de 70 ans amplement modifié nos sociétés et leurs manières d'interagir entre elles.

Via la formalisation de l'information de manière abstraite et mathématique, le sens d'un message n'importe plus. Il est désormais possible de mesurer et de quantifier l'information. Dans son article, Shannon montre que chaque canal de transmission peut faire transiter une quantité maximale d'information de manière fiable. De fait, en raison du bruit inhérent aux canaux de transmission, des erreurs peuvent apparaître lors de la réception du message. Afin de palier cela, Shannon démontre qu'il existe un système de codage correcteur d'erreurs, basé sur l'envoi d'informations redondantes permettant lors de la réception de corriger complètement les effets des distorsions et ainsi, de reconstituer parfaitement le contenu du message émis. Cependant, Shannon ne fournit aucune indication quant à la façon de concevoir un tel code correcteur d'erreurs.

Face à l'enjeu que représente la transmission fiable de l'information, la communauté scientifique s'est consacrée à la construction de tels codes. Tout d'abord, en 1950, Richard Hamming invente un code éponyme permettant de corriger tous les messages contenant une unique erreur binaire [2]. S'en est suivi quinze années de découvertes successives avec les codes de Reed-Muller [3, 4] en 1954, puis les codes convolutifs par Peter Elias en 1955

[5], les codes de BCH en 1959 [6, 7], les codes LDPC par Robert Gallager en 1962 [8]. Ces différentes familles de codes correcteurs d'erreurs permettent de corriger de 1 à $\mathfrak{n}$ erreurs de transmission dans un message en fonction de la quantité d'information redondante ajoutée lors de l'émission du message. Enfin, en 1966, David Forney introduit le principe de la concaténation de codes correcteurs d'erreurs [9]. Cependant, à partir de 1966, les avancées sur la construction de codes correcteurs d'erreurs se font rares. Cela fait alors dire à Robert McEliece en 1971 que « Le codage de canal est mort » [10].

C'était sans compter sur la découverte deux décennies plus tard des turbo codes par Claude Berrou et Alain Glavieux [11]. Grâce à leurs performances, cette nouvelle famille de codes correcteurs d'erreurs se place en rupture avec les schémas de codage alors existants. Dès leur introduction, les turbo codes sont employés dans différents standards de communications numériques adressant des contextes applicatifs divers et variés. L'une des métriques de performance d'un code correcteur d'erreurs est sa capacité de correction en fonction de la qualité de transmission. Pour un turbo code et d'autres familles de codes correcteurs d'erreurs, une telle courbe de performance est divisée en deux parties. Dans la première, nommée région de convergence, un faible incrément de la qualité de transmission résulte en une amélioration importante des performances de décodage. En revanche, dans la région du plancher d'erreurs, l'augmentation de la qualité de transmission ne résulte qu'en une amélioration marginale des performances de décodage. Cette région est alors particulièrement limitante pour des applications nécessitant de très faibles taux d'erreurs.

Par exemple, pour le stockage de masse, les taux d'erreurs cibles sont de plus en plus critiques et atteignent actuellement 10^{-18}. Dans un contexte de lien optique en espace libre, correspondant à des communications satellitaires par faisceau optique, un plancher d'erreurs situé sous les 10^{-9} est recherché. Le même ordre de grandeur est attendu pour le standard CCSDS-2. Enfin les applications de télémesure ou de contrôle-commande d'aéronef sans humain à bord nécessitent elles aussi des taux d'erreurs particulièrement faibles. Dès lors, une amélioration des performances de décodage, et notamment la réduction du plancher d'erreurs est primordiale.

D'autre part, lorsqu'un turbo code est retenu pour un standard de communications numériques, ses performances sont adaptées aux contraintes applicatives correspondant au cas d'emploi de ce standard. Cependant, les besoins applicatifs évoluent au cours du temps et peuvent diverger de ceux originellement considérés. Ainsi, des exigences de plus faibles taux d'erreurs pour une qualité de transmission constante peuvent apparaître. Deux solutions sont alors envisageables. La première consiste à repenser le code correcteur d'erreurs afin d'atteindre ces nouveaux besoins. Néanmoins, cette approche est particulièrement coûteuse. De fait, l'ensemble de l'infrastructure devient obsolète et un nouveau déploiement d'équipements, que ce soit pour la transmission ou la réception, s'avère nécessaire. Cette solution est par exemple difficilement envisageable pour un contexte de communications satellitaires.

Une autre solution consiste à modifier uniquement la partie réception. Dans ce contexte, les modifications des performances se situent au niveau des fonctions de réception. Ainsi, ces dernières années, la communauté scientifique a proposé différentes approches permettant d'améliorer les performances de décodage des turbo codes. Néanmoins, ces approches

sont coûteuses à mettre en œuvre. C'est pourquoi les implémentations matérielles de telles solutions sont rares. Les travaux conduits durant cette thèse répondent alors à la problématique de proposer de nouveaux algorithmes de décodage des turbo codes. Ceux-ci doivent permettre une amélioration des performances de décodage tout en limitant le surcoût de l'implémentation matérielle des solutions de décodage mises en œuvre. La réponse à cette problématique, qui constitue le travail de thèse, est récapitulée dans ce manuscrit. Ce dernier est organisé en quatre chapitres.

Structure du manuscrit de thèse

Le premier chapitre expose les concepts et les notions associées aux travaux de thèse. Dans un premier temps, les notions essentielles des codes correcteurs d'erreurs sont abordées. Celles-ci sont illustrées par le théorème du codage de canal, une présentation des différentes familles de codes correcteurs d'erreurs et enfin des métriques permettant la caractérisation des performances de décodage d'un code correcteur d'erreurs. Dans un second temps, les turbo codes sont détaillés. La construction des turbo codes est tout d'abord exposée. Dans un second temps, le principe de turbo décodage est expliqué. Ceci amène l'introduction de la problématique du plancher d'erreurs. Les raisons de son apparition ainsi que les propositions de la littérature quant à sa réduction sont alors détaillées.

Le deuxième chapitre est consacré à l'étude des oscillations de métriques impliquées dans le décodage itératif des turbo codes. Suite à une observation fine de celles-ci, un algorithme les exploitant est proposé. Celui-ci, inspiré d'une approche originellement développée pour les codes LDPC, permet d'améliorer les performances de décodage de turbo codes dans la zone de convergence. Cependant, ces améliorations ne sont pas observées pour toutes les familles de turbo codes. Finalement, une tentative d'utilisation des oscillations est considérée dans un contexte de décodage répété d'une même trame.

Le troisième chapitre traite de la contribution majeure de ces travaux de thèse. Elle consiste en la présentation d'une méthode permettant de corriger les erreurs résiduelles rencontrées dans la zone du plancher d'erreurs lors du décodage de turbo codes. À la faveur d'une prédiction analytique, les erreurs résiduelles sont caractérisées. Ceci permet alors la comparaison de différentes métriques permettant de les détecter. Une de ces métriques, grâce à son fort pouvoir d'identification, est alors retenue comme cœur de la proposition d'un algorithme de correction des erreurs résiduelles. Cet algorithme a pour propriété d'abaisser drastiquement le plancher d'erreurs de tous les turbo codes standardisés qui ont pu être considérés dans ce manuscrit.

Le quatrième chapitre est dédié à la description d'une architecture matérielle adaptée à l'algorithme de correction des erreurs résiduelles. Cet algorithme, par son principe, peut être vu comme une extension d'un turbo décodeur. Dès lors les différents ordonnancements de turbo décodage sont successivement présentés. Suite à cela, afin d'assurer la maîtrise de la complexité calculatoire de cet algorithme, une étude des différents paramètres sur les performances de décodages est menée. Celle-ci permet alors la proposition d'une architecture matérielle de référence adaptée à un ordonnancement particulier de turbo

décodage. Le coût matériel de l'architecture est comparé à celui d'une architecture matérielle de turbo décodage équivalente. Finalement, des projections sont esquissées quant aux modifications requises à l'adaptation de cette architecture aux autres ordonnancements de turbo décodage existants.

Contributions des travaux de thèse

Les différentes contributions originales de ces travaux de thèse sont :

1. La proposition d'un algorithme de décodage permettant l'amélioration de la convergence de décodage de turbo codes dans certains contextes. Son principe est d'annuler la contribution des décodeurs élémentaires du turbo décodeur lorsqu'un changement de signe sur cette contribution est détectée d'une itération à l'autre. Dans le cadre du standard CCSDS, des gains de 0,1 dB sont observés en comparaison d'un turbo décodage conventionnel. Ceci est détaillé dans le deuxième chapitre.
2. La proposition d'un algorithme de décodage basé sur le décodage successif d'une même trame permettant une réduction limitée du plancher d'erreurs. Un abaissement d'un ordre de grandeur pour de hautes valeurs du rapport signal à bruit est atteint pour différents turbo codes. Cependant les gains sont obtenus au prix d'un surcoût calculatoire important. Ceci est détaillé dans le deuxième chapitre.
3. La formalisation d'une prédiction de la distribution des erreurs résiduelles dans la zone du plancher d'erreurs des turbo codes. Celle-ci se base directement sur le spectre de distances des turbo codes. Le spectre est plus facile à obtenir que les fonctions recenseuses de poids jusqu'alors utilisées. Ceci est détaillé dans le troisième chapitre.
4. La proposition d'un algorithme de décodage permettant une réduction drastique du plancher d'erreurs des turbo codes standardisés grâce à la correction des erreurs résiduelles. Cet algorithme, nommé *Flip and Check* a un surcoût en terme de complexité calculatoire maîtrisé. Il permet d'abaisser le plancher d'erreurs d'au moins un ordre de grandeur dans tous les contextes applicatifs considérés dans ce manuscrit. Ceci est détaillé dans le troisième chapitre.
5. La proposition d'une architecture matérielle adaptée à l'algorithme Flip and Check de correction des erreurs résiduelles. Cette architecture matérielle démontre la possible intégration de la technique proposée pour des systèmes contraints, grâce à un impact limité au niveau de la complexité calculatoire. De plus, la latence du processus de turbo décodage n'est pas impactée. Ceci est détaillé dans le quatrième chapitre.

Ces différentes contributions ont fait l'objet de publications scientifiques :

— Communications nationales avec actes :
  — T. Tonnellier, C. Leroux, B. Le Gal, C. Jego, B. Gadat, and C. Poulliat, "L'algorithme Self-Corrected Max-Log-MAP pour le décodage des turbo codes", in JNRDM, May 2015.

  - T. Tonnellier, C. Leroux, B. Le Gal, C. Jego, B. Gadat, and C. Poulliat, "Extension du principe self-corrected de l'information extrinsèque au décodage itératif de turbo codes," in GRETSI, September 2015.
  - T. Tonnellier, C. Leroux, B. Le Gal, C. Jego, B. Gadat, and N. Van Wambeke, "Correction des erreurs résiduelles lors du processus de turbo décodage : algorithme et architecture," in GDR SoC-SiP, June 2017.

- Communications internationales avec actes :
  - T. Tonnellier, C. Leroux, B. Le Gal, C. Jego, B. Gadat, and N. Van Wambeke, "Lowering the error floor of double-binary turbo codes : the Flip and Check algorithm" in 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), Sept 2016.
  - A. Cassagne, T. Tonnellier, C. Leroux, B. Le Gal, O. Aumage, and D. Barthou, "Beyond Gbps turbo decoder on multi-core CPUs," in 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), Sept 2016.
  - T. Tonnellier, C. Leroux, B. Le Gal, C. Jego, B. Gadat, and N. Van Wambeke, "Hardware architecture for lowering the error floor of LTE turbo codes," in Conference on Design and Architectures for Signal and Image Processing (DASIP), Oct 2016.

- Communication dans une revue internationale avec comité de lecture :
  - T. Tonnellier, C. Leroux, B. L. Gal, B. Gadat, C. Jego, and N. Van Wambeke, "Lowering the error floor of turbo codes with CRC verification," IEEE Wireless Communications Letters, vol. 5, no. 4, pp. 404–407, Aug 2016.

1 Contexte et état de l'art

Ce premier chapitre expose les concepts de base du codage de canal ainsi que les turbo codes.

Dans la première section, les notions essentielles des codes correcteurs d'erreurs sont présentées. Sont notamment présentés le concept d'information dans le cadre des télécommunications, la théorie de l'information de Claude Shannon ainsi que quelques codes correcteurs élémentaires.

Dans la seconde section, les turbo codes convolutifs sont définis. D'abord leur construction, puis leur décodage est détaillé. Les performances des turbo codes standardisés sont succinctement exposées. Ceci amène la prédiction des performances asymptotiques des turbo codes. Finalement, différentes méthodes permettant d'améliorer les performances des turbo codes standardisés sont décrites.

1.1 Définitions et fondamentaux

1.1.1 Introduction

« Le problème fondamental de la communication est de reproduire à un endroit donné de manière exacte ou approximative un message sélectionné à un autre endroit. »

Voilà ce qu'écrivit Claude Shannon dans son papier "A mathematical Theory of Communication" [1], paru en 1948 alors qu'il exerçait en tant qu'ingénieur au sein des laboratoires Bell. Ses travaux prennent notamment leurs sources dans les publications de Harry Nyquist [12] et Ralph Hartley [13], chercheurs eux aussi au sein des laboratoires de Bell alors qu'ils visent à augmenter la vitesse de transmission des signaux sur les lignes de télégraphe.

Hartley est sans doute le premier à employer le mot information dans un contexte technique ou scientifique. Cette quantité mesurable reflète alors la capacité d'un récepteur à distinguer un message émis par une source plutôt qu'un autre.

En posant deux théorèmes fondamentaux, Claude Shannon ouvre la voie à la théorie de l'information. Tout d'abord, il dresse le modèle d'une communication, connu aussi sous le nom de *paradigme de Shannon* dans lequel une source génère un message pour un destinataire. La source et le destinataire sont séparés par un canal, média de transmission du message. Ce média est caractérisé par un phénomène de propagation et par un phénomène de perturbation. Le premier permet au message d'atteindre le destinataire alors que le second dégrade le message transmis.

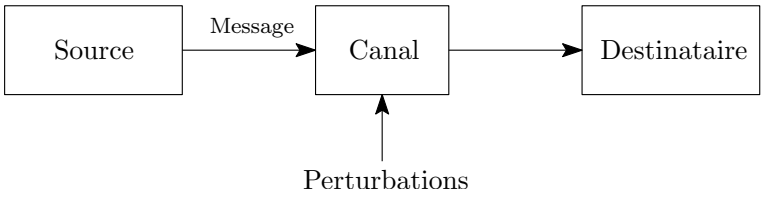


FIGURE 1.1 – Schéma simplifié d'une communication.

Les deux théorèmes fondamentaux de Shannon définissent deux limites théoriques. L'une, relative au codage de source, consiste à obtenir le maximum de concision dans l'expression d'un message, mais n'est pas traitée dans ce manuscrit. L'autre limite concerne le codage de canal qui vise à rendre robuste un message face aux perturbations du canal et demeure au cœur de ces travaux de thèse. Avant de détailler le codage de canal, il est nécessaire de définir la mesure de l'information.

1.1.2 La mesure de l'information

1.1.2.1 Variable aléatoire

Information propre Soit X une variable aléatoire dont les réalisations appartiennent à l'ensemble fini $\mathcal{X}$. L'observation de l'événement $X = x$, avec $x \in \mathcal{X}$, a pour probabilité p.

La quantité d'information associée à la réalisation de cet événement est définie par

$$h(x) = \log(\frac{1}{p}) = -\log(p).$$

La base du logarithme fixe l'unité de la mesure de l'information. Habituellement la base 2 est choisie et l'unité est le *bit*. Néanmoins, ce nom peut prêter à confusion avec le chiffre du système binaire. Ainsi, le système international [14] définit le *shannon* (Sh) comme l'unité dé mesure de l'information. Un shannon représente la quantité d'information associée à la réalisation d'un des deux événements équiprobables qui s'excluent mutuellement.

Qualitativement, fournir une information est équivalent à lever une incertitude. Ainsi, plus un événement est imprévu, plus il apporte de l'information.

Entropie La valeur moyenne de l'information propre calculée sur la totalité de l'ensemble $\mathcal{X}$ est appelée entropie (ou incertitude) de la variable aléatoire X et est notée $H(X)$. Soit n le cardinal de $\mathcal{X}$ et p_i la probabilité que l'événement $X = x_i$ se réalise avec $i \in [\![1; n]\!]$, alors,

$$H(X) = \sum_{i=1}^{n} p_i \times h(x_i) = -\sum_{i=1}^{n} p_i \times \log(p_i).$$

L'entropie correspond à la quantité d'information délivrée par une source. Elle est exprimée en shannon par symbole, les symboles étant les réalisations possibles de cette source. L'entropie d'une source est toujours positive ou nulle. Sa valeur est maximale lorsque les symboles à la sortie de la source sont équiprobables.

1.1.2.2 Couple de variables aléatoires

Considérons maintenant une deuxième variable aléatoire Y dont les réalisations appartiennent à l'ensemble fini $\mathcal{Y}$, avec $m = \text{card}(\mathcal{Y})$.

Entropie conjointe En étendant la définition précédente, l'entropie conjointe de deux variables aléatoires est définie par :

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(X = x_i, Y = y_j) \times \log P(X = x_i, Y = y_j).$$

Elle mesure la quantité moyenne d'information contenue dans un système de deux variables aléatoires. Ainsi, naturellement, $H(X,Y) \geqslant \max[H(X), H(Y)]$.

Entropie conditionnelle $H(X|Y = y_j)$ est l'entropie de la variable X sachant que la variable Y prend la valeur y_j et est définie par :

$$H(X|Y = y_j) = -\sum_{i=1}^{n} P(X = x_i|Y = y_j) \times \log P(X = x_i|Y = y_j)$$

En calculant la moyenne de $H(X|Y = y_j)$ pour tout $y_j \in \mathcal{Y}$ est obtenue l'entropie conditionnelle de X sachant Y :

$$H(X|Y) = \sum_{j=1}^{m} P(Y = y_j) \times H(X|Y = y_j).$$

Elle décrit l'incertitude restante sur la variable aléatoire X lorsque la variable aléatoire Y est connue.

Information mutuelle L'information mutuelle mesure la quantité d'information partagée entre X et Y. En d'autres termes, elle mesure la dépendance entre ces deux variables. Ainsi, l'information mutuelle est nulle si et seulement si les variables sont indépendantes et elle croît lorsque leur degré de dépendance augmente.

$$I(X;Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(X = x_i, Y = y_j) \times \log \left(\frac{P(X = x_i, Y = y_j)}{P(X = x_i)P(Y = y_j)} \right),$$

L'expression de l'information mutuelle peut être formulée en utilisant l'entropie conditionnelle

$$I(X;Y) = H(X) - H(X|Y). \tag{1.1}$$

1.1.3 Le codage de canal

Dans le cadre de ce manuscrit, seuls sont considérés des canaux de transmission causaux sans effet mémoire et stationnaires. En d'autres termes, la sortie du canal à l'instant t ne dépend que de son entrée en l'instant t.
Considérons maintenant que X soit la variable aléatoire à l'entrée du canal et Y la variable aléatoire à sa sortie.

1.1.3.1 Capacité d'un canal

Par définition, la capacité du canal est l'information mutuelle maximale entre X et Y : $C = \sup_{\underline{p}} I(X;Y)$, avec $\underline{p}$ la distribution de probabilité des symboles à l'entrée du canal.
La capacité représente la quantité d'information maximale que le canal peut transporter et son unité est le shannon par symbole.

D'après l'équation 1.1, afin d'obtenir une communication efficace, il est nécessaire que $H(X|Y)$ soit aussi petit que possible. Or, ce terme, dépend du canal. Plus le canal est bruité, plus ce terme est grand. Le message source doit alors être modifié pour que la communication satisfasse une contrainte de qualité. Ceci constitue le codage canal. Son principe est d'ajouter de la redondance en quantité suffisante pour éviter toute ambiguïté lors du décodage.

1.1.3.2 Le théorème du codage de canal

Le second théorème de Shannon énonce : quel que soit $\epsilon > 0$, il existe un code de taille N, avec N suffisamment grand, tel que si $R < C$ – avec R le rendement du code et C la capacité du canal – la probabilité d'erreur après décodage soit inférieure à ϵ.

Ce théorème fournit donc une mesure de la performance d'un système de communication. Un système atteignant la capacité du canal peut être considéré comme optimal. Dès lors, le but du codage de canal est de définir des codes permettant de s'approcher au plus près de cette limite théorique tout en assurant un décodage relativement simple.

Ainsi, le schéma présenté en Figure 1.1 se doit d'être modifié en prenant en compte ce codage. Aussi, afin de rendre possible la propagation de l'information à travers le canal, il est nécessaire de mettre en forme le flux de données. Par exemple, dans le cas d'une communication sans fil, ce flux doit être représenté par un signal haute fréquence afin de pouvoir être émis par une antenne de taille raisonnable. Ceci est le rôle du modulateur. La Figure 1.2 présente une chaîne de communications numériques classique prenant en compte ces deux nouveaux éléments.

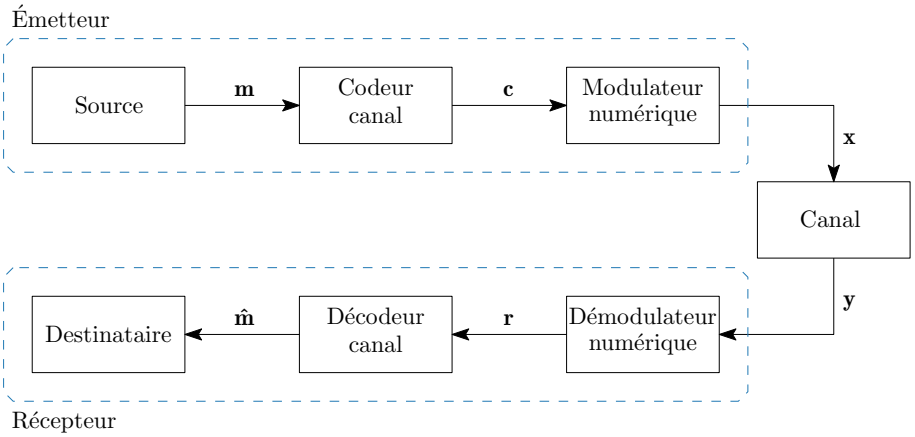


FIGURE 1.2 – Schéma fondamental de communication (ou paradigme de Shannon).

La source génère un flux de symboles $\mathbf{m}$ qui constitue le message à envoyer. Le codeur canal transforme ce message en un mot de code $\mathbf{c}$. Ceci est fait en ajoutant de la redondance au message. Ainsi, $H(m) > H(c)$. Le rapport R entre la taille du mot présent à l'entrée de l'encodeur et la taille du mot codé est appelé le rendement du code. Le mot de code est ensuite mis en forme par le modulateur afin d'être transmis via le canal.
La chaîne de réception est le dual de la chaîne d'émission. Le démodulateur traite la forme d'onde reçue du canal pour transmettre les symboles bruités au décodeur. Si la sortie du démodulateur consiste en une suite binaire, le décodage est dit à *entrées dures*. Si par contre, la sortie du filtre adapté est directement passée au décodeur, le décodage est dit à *entrées pondérées* ou *entrées souples*. Le décodeur, se servant de la séquence reçue et de la connaissance des règles d'encodage fournit une estimation sur le mot transmis le plus probable $\hat{\mathbf{m}}$.

1.1.3.3 Le modèle du canal

Afin de décrire les perturbations subies par le message transitant à travers le canal de transmission, différents modèles peuvent être utilisés. Cependant, le choix du modèle se porte très souvent dans la littérature sur le canal à bruit additif blanc Gaussien (AWGN). Ce canal modélise notamment très bien le bruit thermique qui est une des sources de bruit toujours présente du côté du récepteur.

La loi liant la sortie y_i d'un canal AWGN à son entrée x_i est de la forme $y_i = x_i + n_i$ avec N une variable indépendante et identiquement distribuée suivant une loi normale (ou gaussienne) centrée en zéro et de variance $\sigma^2 = \frac{N_0}{2}$. Nous avons donc $N \sim \mathcal{N}(0, \sigma^2)$ et il vient :

$$P(y_i|x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i)^2}{2\sigma^2}\right) \tag{1.2}$$

Capacité du canal AWGN À partir de l'équation 1.1 et de la densité de probabilité du bruit exprimé par l'équation 1.2, la capacité du canal AWGN est donnée par

$$C = \frac{1}{2}\log_2\left(1 + \frac{P}{\sigma^2}\right), \tag{1.3}$$

exprimé en shannon par symbole émis, avec P la puissance du signal émis.

1.1.4 Les familles de codes correcteurs

Les codes correcteurs d'erreurs utilisés dans les standards de communications numériques sont habituellement classifiés en deux familles distinctes. Il s'agit des codes en blocs et des codes convolutifs. Par définition, les codes convolutifs calculent la redondance de manière continue à mesure que le flot de données arrive alors que les codes en blocs génèrent la redondance par blocs de données.

1.1.4.1 Codes en blocs linéaires

Historiquement, les premiers codes correcteurs d'erreurs découverts furent les codes en blocs linéaires. La théorie mathématique les définissant est basée sur les corps de Galois. Dans la suite, la majorité des opérations se dérouleront dans le corps de Galois à 2 éléments noté GF_2.

Définition Un code en bloc est une application g de GF_2^k dans GF_2^n, avec $k < n$, qui à un vecteur m de taille k associe un autre vecteur c de taille n. L'ensemble des 2^k vecteurs c est appelé code C. Chaque vecteur c du code C est appelé *mot de code*. Le rapport k/n est appelé le rendement du code. Le nombre d'éléments binaires de redondance ajoutés est donné par $n - k$. Si g est linéaire, alors il s'agit d'un code en bloc linéaire. Il existe alors une matrice G à k lignes et n colonnes associée à l'application g. Elle est appelée matrice génératrice du code $C(n,k)$.

Aussi, nous pouvons définir H, la matrice génératrice du code dual de $C(n,k)$. Ainsi, pour tout mot de code c de $C(n,k)$, nous avons :

$$cH^t = 0.$$

Si G peut s'écrire de la forme $G = [I_k,P]$ avec I_k la matrice identité de taille k, alors le code $C(n,k)$ est dit *systématique*.

Détection d'erreur Avant d'évoquer le principe de décodage, il est nécessaire de présenter celui de la détection d'erreur. Supposons que nous codons un message m de taille k par un code en bloc linaire $C(n,k)$, tel que $c = mG$. Si ce message est transmis sur un canal bruité sans effet mémoire, le démodulateur à décision dure fournit au décodeur le vecteur r, de taille n représentant le mot reçu. Celui-ci peut s'écrire de la forme

$$r = c + e$$

avec e un vecteur ligne dont les composantes binaires correspondent aux éventuelles erreurs de transmission. La détection d'erreur se fait alors en calculant le *syndrome* :

$$s = rH^t = (c + e)H^t = eH^t.$$

$s = 0$ si et seulement si r est un mot du code. Si $s \neq 0$, alors il existe des erreurs de transmission. La réciproque n'est pas vraie car la combinaison linéaire de deux mots de code est aussi un mot du code. Il s'agit de la conséquence directe de la linéarité de g.

Principe du décodage Il est possible de définir une distance entre deux mots de code, appelée *distance de Hamming* ($d_H(c,c')$). Définissons tout d'abord le poids de Hamming d'un mot de code ($P_H(c)$) comme étant le nombre de ses éléments non nul. La distance de Hamming entre deux mots de codes est égale au poids de Hamming de leur combinaison linéaire : $d_H(c,c') = P_H(c + c')$.

Le principe du décodage par maximum de vraisemblance consiste en la recherche du mot de code $\hat{c}$ le plus vraisemblable, c'est-à-dire celui qui est à la distance de Hamming minimale du mot de code reçu r. Cela revient à trouver $\hat{c}$, tel que

$$d_H(r,\hat{c}) \leqslant d_H(r,\hat{c}) \qquad \forall c \neq \hat{c} \in C.$$

Il vient donc assez naturellement que les performances de décodage d'un code sont dépendantes des distances de Hamming séparant les différents mots de code. La valeur minimale de cet ensemble de distances est appelée *la distance minimale du code*, notée d_{min}.

$$d_{min} = \min_{c,\ c' \in C,\ c \neq c'} d_H(c,c').$$

Comme le code est linéaire, la distance minimale est aussi égale au poids minimal des mots de code non nuls. Ainsi, le poids des mots de code (excepté le mot de code nul) est compris entre d_{min} et n. Le nombre de mots de code de poids de Hamming d est appelé *multiplicité* et est noté A_d.

Codes cycliques Un code en bloc linéaire est dit cyclique si la permutation circulaire vers la gauche d'un mot de code est aussi un mot de code. C'est-à-dire que si $c = \{c_0,...,c_j,...,c_{n-1}\}$ est un mot de code, $c' = \{c_1,...,c_j,...,c_{n-1},c_0\}$ est aussi un mot de code.

Ces codes sont souvent représentés sous forme polynomiale. Or, en électronique numérique, une division polynomiale dans GF est aisément réalisable à partir de registres à décalage. Ils ont donc été particulièrement étudiés et sont présents aujourd'hui dans de nombreux standards de communications numériques. Nous pouvons par exemple citer les codes BCH du nom de leurs inventeurs Bose, Ray-Chaudhuri et Hocquenghem [7]. Ils ont la particularité de pouvoir être construits en fonction du pouvoir de correction visé.

Une autre catégorie de codes cycliques est largement utilisée dans les standards de communication actuels. Il s'agit des codes à redondance cyclique (CRC) [15] qui possèdent un fort pouvoir de détection d'erreurs, mais aucun pouvoir de correction.

1.1.4.2 Codes convolutifs

Définition Les codes convolutifs ont été proposés par Peter Elias en 1955 comme une alternative aux codes en blocs [5]. Elias cherche alors à définir un code à longueur variable. Le codage est fait de telle sorte que la sortie dépende de l'entrée courante mais aussi des entrées précédentes. Un codeur convolutif peut être vu comme un filtre. La sortie est alors le produit de convolution entre l'entrée et la réponse impulsionnelle du codeur.

En utilisant la notation de Forney de l'opérateur de retard D [16], le message peut être exprimé sous la forme $m = \sum_k m_k D^k$. De la même manière, le mot codé devient $c = \sum_n c_n D^n$. L'exposant de D représente alors l'instant auquel m_k et c_k apparaissent à l'entrée et à la sortie du codeur, respectivement. c_k peut aussi être exprimé comme une combinaison linéaire des ν précédents éléments du message : $c_k = \sum_{j=0}^{\nu} g_j m_{k-j}$. La suite d'éléments g_j est appelée séquence génératrice du code. Elle est souvent exprimée en octal. ν représente le nombre d'éléments de mémorisation à l'intérieur du codeur.

Exemple Soit le codeur convolutif de rendement $R = 1/2$, de mémoire $\nu = 2$ présenté en Figure 1.3. Ses deux séquences génératrices définissant les sorties c_1 et c_2 sont respectivement $G_1 = (5)_8$ et $G_2 = (7)_8$. Puisque $\nu = 2$, ce codeur peut être dans 2^ν états différents. Ainsi, un codeur convolutif peut être vu comme une machine de Mealy. Ce diagramme d'états, présenté en Figure 1.4, montre les $2^{\nu+1}$ différentes transitions entre les 2^ν états internes.

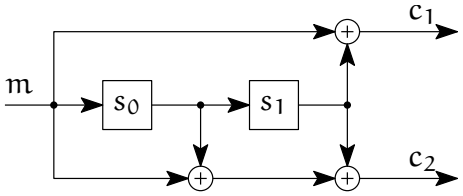


FIGURE 1.3 – Exemple de codeur convolutif.

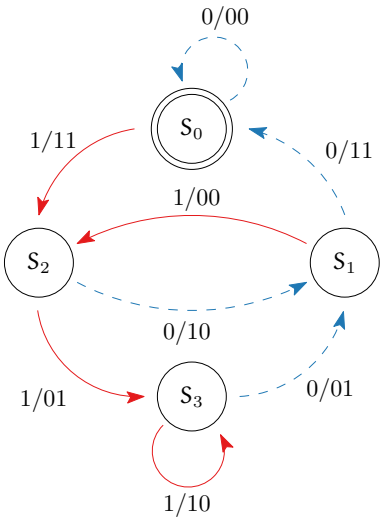


FIGURE 1.4 – Machine d'états associée au codeur convolutif de la Figure 1.3.

Représentation en treillis La représentation sous la forme d'une machine d'états, bien qu'elle permette de se rendre compte aisément du fonctionnement du codeur, est difficile à exploiter pour le décodage. C'est pourquoi, une représentation sous forme de treillis est privilégiée. Elle a été utilisée pour la première fois par Forney en 1973 [17]. Cette représentation permet de rendre compte à la fois de l'état interne du codeur, des transitions et de l'évolution temporelle. La Figure 1.5 présente le diagramme en treillis associé au codeur convolutif de la Figure 1.3, en considérant que l'état interne du codeur est S_0. Comme il n'y a qu'une entrée, de chaque nœud partent 2 branches et vers chaque nœud convergent 2 branches. À partir de $\nu+1$ unités de temps, quel que soit l'état initial, le motif du treillis se répète. Cette durée est appelée longueur de contrainte du code convolutif. Lors du codage d'une séquence, la connexion entre les différents nœuds est nommée chemin dans le treillis. Sur la Figure 1.5 sont présentées sur les branches les 8 associations entrées/sorties possibles.

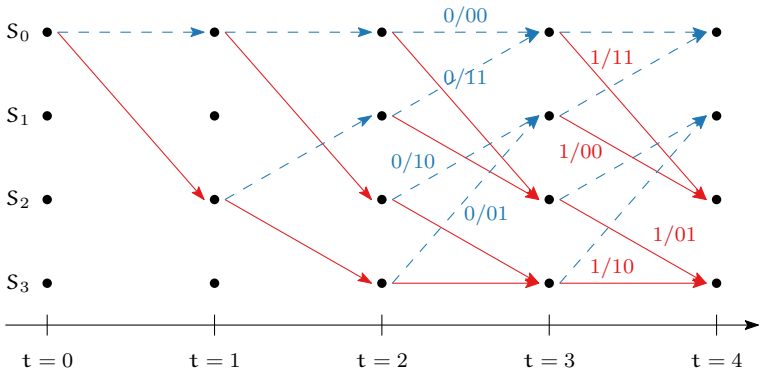


FIGURE 1.5 – Diagramme en treillis du codeur convolutif de la Figure 1.3.

Terminaison Comme évoqué précédemment, l'un des intérêts des codeurs convolutifs est de permettre d'obtenir des mots de codes de taille infinie. Néanmoins, dans le contexte de protocoles de communications numériques, les messages ont une taille prédéfinie. Se pose alors la question de la fermeture du treillis, impactant les performances de décodage. Trois solutions de terminaisons ont été proposées dans la littérature :

1. pas de terminaison. L'état final ne peut donc être connu par le décodeur. Ainsi, les derniers symboles encodés sont moins bien protégés.
2. forcer l'encodeur à finir dans un état connu. Quel que soit l'état interne du codeur après que les K bits d'information aient été codés, un retour dans n'importe quel état est possible en y ajoutant ν bits supplémentaires. Le message transmis est alors composé de la séquence initiale concaténée avec cette séquence supplémentaire. Une légère baisse de rendement est donc induite par cette technique. Ceci est d'autant plus vrai que K est petit. Néanmoins, la connaissance par le décodeur de l'état initial et de l'état final est bénéfique quant aux performances de décodage. Bien souvent, l'état initial et l'état final sont fixés à l'état 0.
3. utiliser un codeur circulaire. Dans ce cas, l'état initial et l'état final sont les mêmes (mais pas forcément l'état 0). Cet état porte le nom d'état de circulation [18]. Le treillis peut alors être vu comme un cercle. Tous les bits ont la même protection et ce sans induire une modification du rendement. Par contre, un pré-codage du message est nécessaire afin d'identifier l'état de circulation.

Code systématique et récursif Les codes convolutifs peuvent être classés selon 2 critères principaux. Ils sont systématiques ou non-systématiques et récursifs ou non-récursifs. Comme pour les codes en blocs linéaires, un code est dit *systématique* lorsque la séquence d'information apparaît dans le mot de code. La matrice génératrice contient donc la matrice identité.

Un code convolutif est dit *récursif* si une rétroaction est présente au sein du codeur, en amont de ses registres à décalages. Cela implique qu'au moins un des termes de la séquence génératrice est fractionnel.

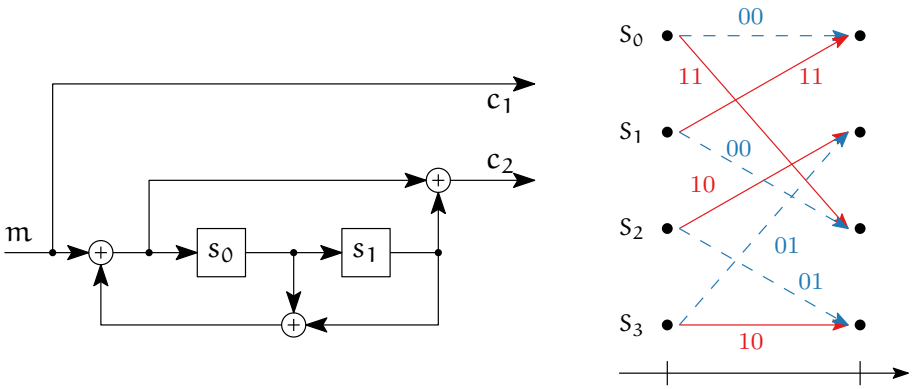


FIGURE 1.6 – Exemple de codeur convolutif récursif systématique et le treillis associé.

Ainsi, la version systématique récursive du codeur présenté en Figure 1.3 est donné en Figure 1.6. Une représentation en treillis de ce code est également fournie. Sa matrice génératrice est $G(D) = \left[1 \quad \frac{1+D^2}{1+D+D^2}\right]$. Nous pouvons remarquer une similitude avec le treillis présenté dans la Figure 1.5 : pour chaque branche, les mêmes sorties sont présentes et elles relient les mêmes nœuds. En revanche, les bits d'informations qui y sont associés diffèrent. Généralement, les codeurs récursifs ont des propriétés de distance plus favorables que les codeurs non-récursifs.

Décodeur Pour le décodage de codes convolutifs, deux principaux algorithmes de décodage sont proposés dans la littérature. Les deux types de décodeurs sont basés sur l'exploration du treillis associé au code. Le premier est le décodeur à maximum de vraisemblance par séquence, implanté par l'algorithme de Viterbi [17], [19] et [20]. Le second est le décodeur à maximum *a posteriori* bit à bit, implémenté par l'algorithme BCJR [21]. Le premier minimise la probabilité d'erreur sur la séquence considérée. Quant au second, il minimise la probabilité d'erreur au niveau des bits. De manière plus formelle, le décodeur à maximum de vraisemblance (ML) prend une décision $\hat{d}$ tel que $\hat{m} = \arg\max_{c} P(r|x)$. Le décodeur à maximum *a posteriori* prend une décision sur les bits d'information $\hat{m}_l$ tel que $\hat{m}_l = \arg\max_{m_l} P(m_l|y)$. Le décodage basé sur l'algorithme MAP est détaillé dans la section suivante.

1.1.5 Les performances d'un code

Dans cette section, les performances d'un code sont discutées. Tout d'abord sont présentées les limites théoriques qui découlent des résultats présentés dans la section 1.1.3.3. Ensuite, ces limites théoriques sont comparées aux performances de décodage en fonction du rapport signal à bruit.

1.1.5.1 La limite de Shannon

En reprenant l'équation 1.3, comme pour un canal AWGN, $\sigma^2 = N_0/2$ avec N_0 la densité spectrale du bruit et comme $P = R \times E_b$, avec E_b l'énergie moyenne par bit d'information, nous pouvons écrire :

$$C = \frac{1}{2}\log_2\left(1 + \frac{2RE_b}{N_0}\right).$$

Selon le théorème de Shannon, des communications fiables sont possibles sur un canal de transmission si et seulement si $R < C$. Ainsi, $2R < \log_2\left(1 + \frac{2RE_b}{N_0}\right)$, ce qui donne

$$\frac{2^{2R} - 1}{2R} < \frac{E_b}{N_0}. \tag{1.4}$$

En faisant tendre R vers 0, nous avons

$$\frac{E_b}{N_0} > \ln 2 = -1{,}59 \text{ dB}.$$

En considérant un canal AWGN, aucun système ne peut transmettre de l'information de manière fiable pour un rapport signal à bruit inférieur à $-1{,}6$ dB.

1.1.5.2 Capacité du canal BI-AWGN

La limite préalablement présentée considère un alphabet infini et réel présent à l'entrée du canal. Or, lorsqu'une modulation numérique est employée, les entrées sont discrètes et font partie d'un alphabet fini. Dans le cadre d'une modulation à changement de phase binaire (BPSK), chaque bit du mot de code est translaté sur $\{-1; 1\}$. Ce canal AWGN contraint sur son entrée est appelé canal AWGN à entrée binaire (BI-AWGN). Sa capacité, en utilisant l'équation 1.1 et selon [22, Chapitre 8], est donnée par :

$$C_{BI-AWGN} = -\int_{-\infty}^{\infty} p(y)\log_2\left(p(y)\right)dy - 0{,}5\log_2\left(2\pi e\sigma^2\right), \tag{1.5}$$

avec

$$\begin{aligned} p(y) &= \frac{1}{2}\left(p(y|x=+1) + p(y|x=-1)\right) \\ &= \frac{1}{2}\frac{1}{\sqrt{2\pi\sigma}}\left(\exp\left(\frac{-(y+1)^2}{2\sigma^2}\right) + \exp\left(\frac{-(y-1)^2}{2\sigma^2}\right)\right). \end{aligned}$$

Afin de trouver les couples de valeurs R et σ qui vérifient cette équation, le calcul numérique de l'intégrale est nécessaire. Dans [23], une version Monte-Carlo de ce calcul est présentée sous forme de pseudo-code.

Finalement, les capacités (en bit par symbole sur le canal) associées aux équations 1.4 et 1.5 en fonction du rapport signal à bruit (E_b/N_0, en dB) sont fournies en Figure 1.7.

Nous pouvons remarquer que pour de faibles rendements, la capacité du canal BI-AWGN est très proche de celle du canal AWGN non contraint. Ces courbes sont les limites du rapport signal à bruit à partir desquelles des communications sans erreur sont possibles, pour un rendement de codage donné et en considérant des paquets de taille infinie. Pour des tailles finies, les codes modernes se situent très près de ces limites, à environ 0,5 dB.

1.1.5.3 Gain de codage

La probabilité d'erreur sur le canal de transmission est fonction du rapport signal à bruit. Comme esquissé précédemment, il peut être exprimé comme le rapport entre l'énergie moyenne par élément binaire transmis (E_b) et la densité spectrale mono-latérale du bruit (N_0).

En l'absence de codage, le rapport signal à bruit à l'entrée du récepteur est égal à $\frac{E_b}{N_0}$. En revanche, dans le cadre d'un codage de canal de rendement R, la probabilité d'erreur sur le canal est fonction de $\frac{RE_b}{N_0}$.

Pour une modulation numérique BPSK dont les symboles sont émis sur un canal BI-

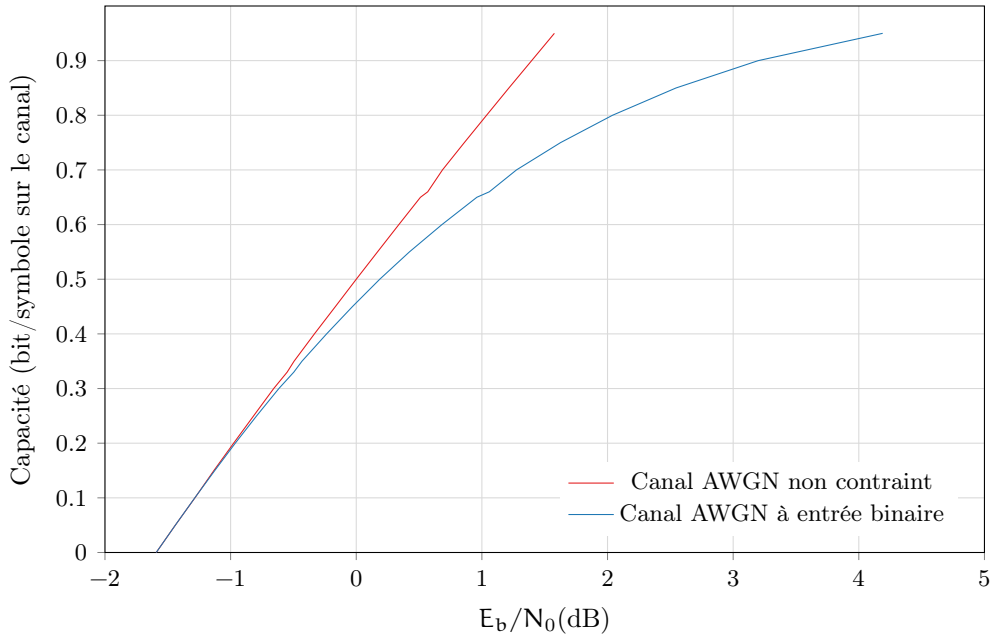


FIGURE 1.7 – Capacité du canal en fonction du rapport signal à bruit pour le canal AWGN non contraint et pour le canal AWGN à entrée binaire.

AWGN, la probabilité d'erreur binaire est donnée par

$$P_e = \frac{1}{2}\text{erfc}\sqrt{\frac{E_b}{N_0}} \tag{1.6}$$

avec erfc la fonction d'erreur complémentaire. Sur la Figure 1.8, cette probabilité est tracée en fonction du rapport signal à bruit $\frac{E_b}{N_0}$. Dans ce cas, une probabilité d'erreur binaire de 10^{-4} est obtenue pour un rapport signal à bruit légèrement supérieur à 8 dB.

D'après la Figure 1.7, il existe un code de rendement $\frac{1}{2}$ qui permet d'obtenir une probabilité d'erreur binaire très faible (par exemple, de l'ordre de 10^{-4}) pour peu que le rapport signal à bruit soit supérieur à 0,19 dB. D'après le théorème de Shannon, la zone située à gauche de cette limite n'est pas atteignable.

Considérons le cas du code convolutif systématique et récursif (RSC) de matrice génératrice $G(D) = \left[1 \quad \frac{1+D^2}{1+D+D^2}\right]$ (présenté en Figure 1.6). En effectuant un décodage utilisant l'algorithme BCJR pour différentes valeurs de variance du bruit du canal AWGN dans le cadre d'une simulation Monte-Carlo, la courbe présentée en Figure 1.8 est obtenue.

La distance, pour un taux d'erreur binaire (BER) cible, entre la courbe bleue et la courbe rouge est appelée *gain de codage*. En l'occurrence, pour un BER de 10^{-4}, le gain de codage en utilisant le code RSC sus-présenté vaut environ 3 dB. Le BER représente donc le rapport entre le nombre de bits erronés et le nombre de bits d'information transmis. Le taux d'erreur peut aussi être exprimé selon le nombre de trames erronées par trames envoyées. Nous parlons alors de taux d'erreur trame (FER).

Toujours pour un BER de 10^{-4}, l'écart avec la limite de Shannon est de 5 dB. Ainsi, plus un code est performant, plus il s'approchera de cette limite. Néanmoins, la complexité du décodage est un paramètre très important à prendre en compte dans la caractérisation de codes correcteurs d'erreurs. Les turbo codes possèdent justement ces deux caractéristiques : ils se situent à environ 1 dB de la limite de Shannon tout en étant basés sur un algorithme de décodage ayant une complexité calculatoire raisonnable. Ces derniers sont détaillés dans la section suivante.

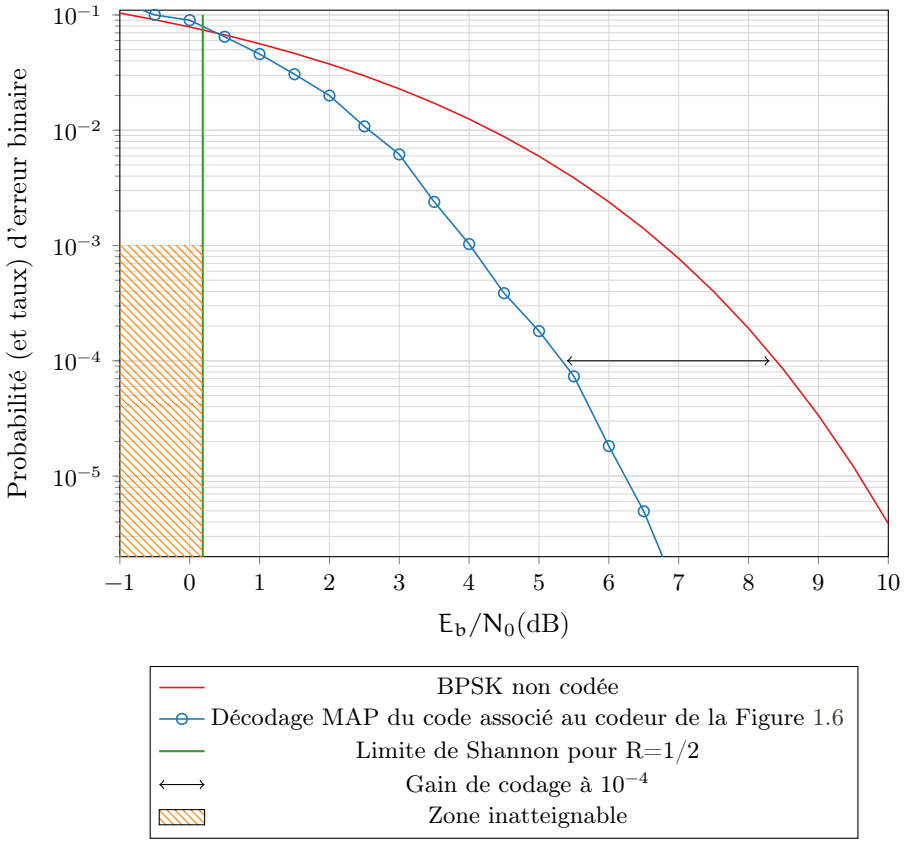


FIGURE 1.8 – Probabilité d'erreur de la modulation BPSK sur le canal BI-AWGN, du code RSC de matrice génératrice $\mathsf{G}(\mathsf{D}) = \left[1 \quad \frac{1+\mathsf{D}^2}{1+\mathsf{D}+\mathsf{D}^2}\right]$ et la limite de Shannon pour la canal AWGN et un rendement de 1/2.

1.2 Les turbo codes

Les turbo codes convolutifs ont été proposés par Claude Berrou, Alain Glavieux et Punya Thitimajshima en 1993 [11]. Dans la littérature, ils portent aussi le nom de *codes convolutifs concaténés en parallèle*. Le décodeur associé est construit autour de deux décodeurs à entrées et sorties pondérées (SISO) qui travaillent de concours en échangeant de l'information, appelée *information extrinsèque* au cours d'un processus itératif.

1.2.1 La construction des turbo codes

1.2.1.1 Historique

Les réflexions, expérimentations et observations qui permirent à Berrou et Glavieux la définition des turbo codes sont présentées par eux-mêmes dans [24], 5 ans après la première publication sur les turbo codes. Ces travaux commencèrent avec la volonté de transcrire l'algorithme de Viterbi à sorties souples (SOVA) – pensé par G. Battail [25] puis par Hagenauer et Hoeher [26] – en transistors CMOS. Ce travail fut valorisé en 1993 [27] mais leur permit surtout d'acquérir une maîtrise quant au décodage probabiliste. Aussi, ils eurent le sentiment qu'un décodeur SISO pouvait être vu comme un amplificateur de SNR. Cette analogie les mena vers le principe de contre-réaction qui, en électronique, permet d'obtenir des amplificateurs avec de meilleures propriétés, notamment en terme de stabilité et de linéarité.

La complexité calculatoire de l'implémentation naïve d'un décodeur à maximum de vraisemblance croit avec l'inverse de la probabilité d'erreur du code associé. Ainsi, afin de permettre d'obtenir de bonnes performances en terme de correction d'erreurs tout en proposant un décodeur possédant une complexité raisonnable, David Forney a proposé durant sa thèse de doctorat en 1965 la concaténation série de deux codes « simples » [9]. En effet, il a montré que les codes concaténés permettent d'obtenir des probabilités d'erreurs qui décroissent exponentiellement pour un décodeur dont la complexité calculatoire croît pronominalement avec la taille du code.

A partir des années 70, la NASA standardise les codes concaténés pour des applications spatiales. Ce schéma de codage est présenté en Figure 1.9. Dans un premier temps le message est traité par un codeur de Reed-Solomon (RS) [28]. Il s'agit d'une généralisation des codes BCH évoqués précédemment. Le mot de code résultant est alors entrelacé, puis fournit à un codeur convolutif. Une fois que ce mot de code est transmis à travers le canal puis démodulé, il est décodé par un décodeur de Viterbi, désentrelacé et décodé par un décodeur de RS. L'étape d'entrelacement et de désentrelacement permet de brasser les salves d'erreurs, présentes en sortie du décodeur de Viterbi. Ce schéma de codage permet d'obtenir des gains de codage d'environ 7 dB dans le cas d'un canal BI-AWGN pour une probabilité d'erreur de 10^{-5}.

À la fin des années 80, Hagenauer et Hoeher proposent la concaténation série de codeurs convolutifs [29]. À partir d'un décodeur SOVA [27], Berrou s'attelle à expérimenter le décodage de cette concaténation série. Il décide alors de réinjecter la sortie du second décodeur (le décodeur extérieur) dans le premier (le décodeur intérieur). Pour ce faire, il est nécessaire de reconstruire $\hat{\mathbf{c}}_1$ après décodage par le décodeur extérieur. C'est alors que Berrou remarque que le BER de ces symboles reconstruits est inférieur à celui du mot décodé $\hat{\mathbf{c}}$. Cette constatation n'avait alors jamais été décrite dans la littérature. Il restait alors à remplacer les codeurs constitutifs par des versions systématiques. Ainsi, la sortie du second décodeur pouvait être réinjectée à l'entrée du premier. Mais les codes convolutifs systématiques ont des propriétés moins intéressantes que les codes convolutifs non-systématiques, à moins qu'il s'agisse d'un code convolutif récursif systématique (RSC). L'étude des codes RSC fut donnée à Punya Thitimajshima dans le cadre de sa thèse [30].

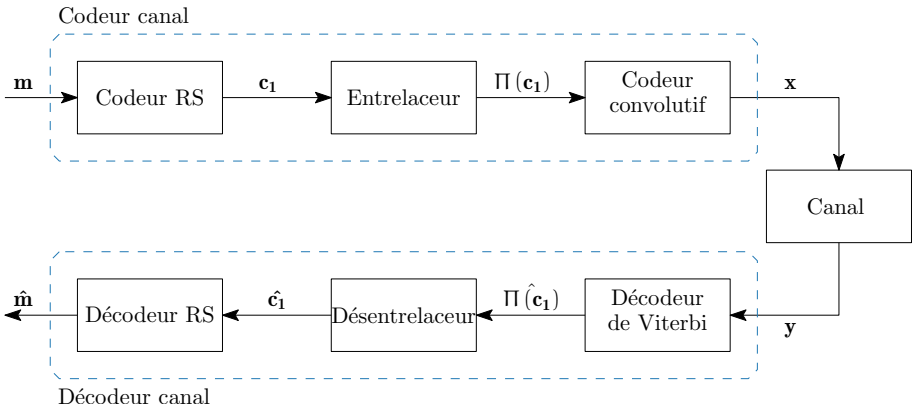


FIGURE 1.9 – Principe de concaténation série standardisé par la NASA.

L'idée de la *concaténation parallèle* est concomitante à la démarche d'implémentation matérielle de la concaténation série des codes RSC. En effet, une architecture matérielle de schéma concaténé en série – de part les différents rendements des codes constituants – nécessite différentes horloges. La concaténation parallèle simplifie ce problème puisque les deux codeurs constitutifs et les deux décodeurs constitutifs fonctionnent de manière synchronisée. De plus, cette nouvelle structure permet aussi, à rendement égal, d'obtenir plus de symboles redondants pour un des deux codeurs élémentaires. Par exemple, pour obtenir un code concaténé de rendement 1/3, dans le cas d'une concaténation parallèle, les deux codeurs auront un rendement 1/2. En revanche, pour une concaténation série, l'un aura un rendement 1/2 et l'autre un rendement 2/3. En effet, si R_p est le rendement global obtenu pour la concaténation parallèle de 2 codeurs de rendement R_1 et R_2 et R_s celui obtenu pour une concaténation série, ils ont respectivement pour expression :

$$R_p = \frac{R_1 \times R_2}{R_1 + R_2 - R_1 \times R_2} \quad \text{et} \quad R_s = R_1 \times R_2$$

Finalement, cette plus grande diversité, provenant d'un plus grand nombre d'information redondante par information systématique, permet d'obtenir un seuil de convergence qui se produit à plus faible SNR. Ce seuil de convergence permet au schéma concaténé en parallèle de s'approcher à seulement 0,7 dB de la limite de Shannon [11]. Néanmoins, les schémas concaténés en série permettent d'obtenir des distances minimales plus importantes impliquant de meilleures performances asymptotiques [31]. Dans la littérature, afin de distinguer ces deux types de turbo codes, les notations PCCC pour codes convolutifs concaténés en parallèle et SCCC pour codes concaténés en série sont utilisés.

Les éléments constitutifs d'un turbo code concaténé en parallèle, comme présenté en Figure 1.10, sont maintenant détaillés. Dans un premier temps, les codeurs RSC, puis l'entrelaceur et enfin le principe de poinçonnage sont traités dans la suite de cette section.

1.2.1.2 Codeur convolutif récursif

Depuis les travaux de Forney, la concaténation de codes repose sur le codage de la séquence d'information par de multiples codeurs constitutifs, ayant une faible complexité

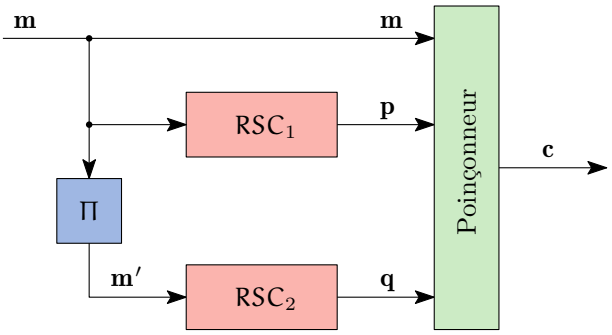


FIGURE 1.10 – Schéma générique d'un codeur de turbo code.

calculatoire au niveau du décodeur. Dans le cadre des turbo codes convolutifs, il s'agit de codes RSC à 4, 8 ou 16 états. Augmenter le nombre d'états du codeur revient à accroître la distance minimale du code mais implique aussi une augmentation de la complexité calculatoire du décodeur. Les meilleurs codes RSC de rendement 1/2 adaptés aux turbo codes ont été identifiés et leurs distances minimales ont été calculées dans [32]. Ces résultats sont reportés dans le tableau 1.1. Les distances minimales sont relativement faibles. Dès lors, les performances asymptotiques des turbo codes résultent de l'étape d'entrelacement.

Nombre d'états	**4**	**8**	**16**
$\mathbf{G(D)}$	$\left[1, \frac{1+D+D^2}{1+D^2}\right]$	$\left[1, \frac{1+D+D^3}{1+D^2+D^3}\right]$	$\left[1, \frac{1+D^3+D^4}{1+D+D^2+D^4}\right]$
$\mathbf{d_{min}/A_{d_{min}}}$	5/2	6/1	7/3

TABLEAU 1.1 – Codeurs RSC usuellement utilisés pour des turbo codes à 4, 8 et 16 états avec leur distance minimale et la multiplicité associée.

1.2.1.3 Entrelacement

Le premier but de l'entrelacement est de lutter contre les paquets d'erreurs. Ainsi lors de l'étape d'entrelacement, la séquence d'information dans *l'ordre naturel* est mélangée par une fonction de permutation. Cette séquence est ensuite présentée dans *l'ordre entrelacé* au second codeur RSC. Les propriétés de dispersion d'une permutation peuvent être mesurées en trouvant le minimum de la somme de la distance spatiale entre deux indices dans l'ordre naturel et dans l'ordre entrelacé. Cet écart (spread en anglais) est défini dans [33] par :

$$S_{min} = \min_{i_1,i_2,i_1\neq i_2} \left(|i_1 - i_2| + |\Pi(i_1) - \Pi(i_2)|\right), \forall (i_1,i_2) \in [\![0,K-1]\!]^2,$$

avec K la taille de l'entrelaceur et Π la fonction d'entrelacement. Il a été démontré [33] que l'écart minimum a pour borne supérieure $\lfloor\sqrt{2K}\rfloor$.

Finalement, une bonne permutation doit permettre d'obtenir un écart suffisant afin de minimiser la corrélation entre les deux dimensions. Aussi, elle doit permettre d'atteindre de grandes distances minimales pour assurer de bonnes performances asymptotiques tout en pouvant être facilement implémentée. Ainsi, de part son rôle clef, la conception d'entrelaceurs efficaces a été particulièrement étudiée dans la littérature. Dans la suite, quatre types d'entrelaceurs sont détaillés :

- L'*entrelaceur uniforme* n'est pas un cas pratique d'entrelaceur. Il ne peut être utilisé pour un standard de communication. Il a été défini par Benedetto en 1995 dans [34] afin d'estimer les performances moyennes des schémas concaténés. Le principe de l'entrelaceur uniforme est de permuter un mot d'information de taille K et de poids w en un autre mot de poids w parmi toutes les combinaisons possibles $\binom{K}{w}$, ce avec une probabilité équirépartie. Pour tout mot d'information, la permutation est choisie aléatoirement. Ainsi, cet entrelaceur permet d'extraire les paramètres des codes constituants d'un schéma concaténé sans avoir à tenir compte de la permutation.
- La permutation *dithered relatively prime (DRP)* a été pensée par Crozier [35]. Elle est basée sur deux opérations de désordres (dither) séparées par une permutation régulière basée sur un entier premier relatif. Les entrelaceurs DRP ont l'avantage de pouvoir être facilement calculés en temps réel tout en permettant d'atteindre de bonnes performances asymptotiques. Néanmoins, ils n'ont jamais été standardisés.
- La *permutation presque régulière (ARP)* a été développée par Berrou et son équipe en 2004 [36]. Elle est de la forme $j = \Pi(i) = Pi + Q(i) \mod K$, avec P premier et K et $Q(i)$ entiers. Le nombre de valeurs différentes pour $Q(i)$ est souvent fixé à 4. Ce type de permutation a été choisi dans le cadre des standards de télédiffusion DVB-RCS [37], DVB-RCS2 [38] et DVB-RCT [39]. Cet entrelaceur possède l'avantage de présenter un parallélisme égal au cardinal de Q. Cette propriété est intéressante pour l'implémentation de décodeurs rapides.
- Enfin les entrelaceurs basés sur des *permutations polynomiales quadratiques (QPP)* [40] sont utilisés dans le standard de téléphonie de quatrième génération [41]. Cet entrelaceur est souvent de la forme $j = \pi(i) = f_1 i^2 + f_2 i \mod K$, avec f_1 et f_2 entiers. La conception de ces entrelaceurs se réduit donc à trouver des coefficients polynomiaux. Ils permettent d'obtenir des distances minimales importantes [42]. Ce type de permutation possède l'avantage de pouvoir être construite en respectant des propriétés de non contention. Ceci confère des possibilités de hauts degrés de parallélisme pour la conception du décodeur associé [43].

Récemment, il a été montré que ces 3 trois types d'entrelaceurs sont équivalents [44]. En effet, les permutations DRP et QPP peuvent s'écrire sous la forme d'une ARP. Ainsi, le formalisme lié à l'entrelaceur ARP devrait être le seul conservé à l'avenir.

Il est à noter que selon le type de fermeture du treillis (*cf.* 1.1.4.2), la terminaison peut être ou non incluse dans la séquence entrelacée. Ceci peut modifier les performances asymptotiques en impactant la distance minimale du code.

1.2.1.4 Poinçonnage

L'intérêt du poinçonnage est de supprimer certains bits de la séquence codée afin d'augmenter le rendement. Par exemple, le codeur présenté en Figure 1.10 possède un rendement nominal de $\frac{1}{3}$. Si nous supprimons à tour de rôle une des deux informations redondantes, le rendement du code devient $\frac{1}{2}$. Usuellement, seuls des symboles de redondance sont supprimés. Le poinçonnage est souvent représenté sous forme matricielle comme dans les équations 1.7. La première ligne correspond au poinçonnage de l'information redondante produite dans l'ordre naturel et la seconde ligne, celle produite dans l'ordre entrelacé. Un 1 signifie que le symbole est gardé, un 0 qu'il est poinçonné. Ainsi, à partir d'un turbo code de rendement natif de $\frac{1}{3}$, les matrices de poinçonnage 1.7a à 1.7c élèvent son rendement à respectivement $\frac{2}{5}$, $\frac{1}{2}$ et $\frac{2}{3}$. Cette construction aisée de codes à rendements plus élevés est un atout indéniable vis-à-vis des codes en blocs. Il est cependant à noter que les matrices de poinçonnage modifient les propriétés du code. Tout comme les entrelaceurs elles doivent être sélectionnées avec soin.

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{(1.7a)} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{(1.7b)} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{(1.7c)}$$

1.2.2 Le décodage itératif

Comme exprimé précédemment, l'innovation majeure des turbo codes est d'obtenir des performances en terme de correction d'erreurs proche de la limite de Shannon tout en proposant un principe de décodage relativement simple. De par la présence de la phase d'entrelacement, un décodage MAP est trop complexe. En effet, cela impliquerait des calculs sur un treillis composé d'un trop grand nombre d'états. Ainsi en même temps que Berrou *et al.* proposent la construction des turbo codes, ils introduisent une méthode de décodage sous-optimale mais à faible complexité [11]. Cette technique est basée sur un processus itératif. D'abord, chaque code RSC est traité par un décodeur associé. Mais, comme le code est systématique et que chaque sous-code provient du même mot d'information (présenté dans un ordre différent), les informations produites par les deux décodeurs peuvent être combinées. Ainsi, l'information délivrée par l'un des décodeurs est utilisée par le second. Le décodage itératif permet donc d'augmenter la fiabilité de la décision aux cours des itérations. L'information échangée par les décodeurs est appelée *information extrinsèque*. Un schéma générique du décodeur itératif correspondant au codeur générique de la Figure 1.10 est présenté en Figure 1.11. Le principe du décodage itératif est désormais introduit.

1.2.2.1 Décodage SISO : l'algorithme BCJR

Considérons un turbo code binaire sans poinçonnage composé de deux codeurs RSC de rendement 1/2. La séquence d'information binaire de taille K $\mathbf{m} = (m_1,m_2,...,m_K)$ est appliquée au premier codeur RSC. Sa sortie est notée par $\mathbf{c^p} = (c_1^p,c_2^p,...,c_K^p)$. La séquence d'information est aussi entrelacée et devient $\mathbf{m'} = (m'_1,m'_2,...,m'_K)$. Cette dernière est ensuite codée par le second RSC afin d'obtenir $\mathbf{c'^p} = (c_1'^p,c_2'^p,...,c_K'^p)$. La sortie du

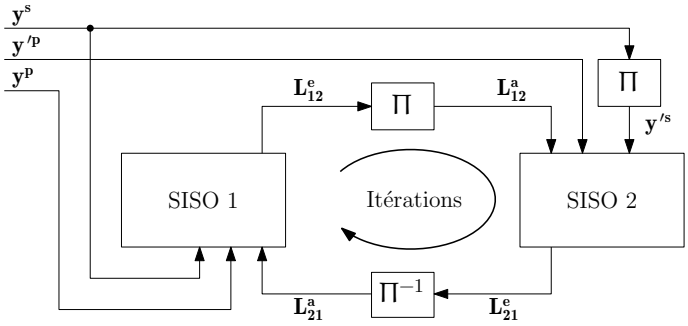


FIGURE 1.11 – Structure d'un décodeur de turbo code.

turbo codeur correspond alors à la concaténation de ces trois séquences $(\mathbf{m},\mathbf{c^p},\mathbf{c'^p})$. Une modulation numérique de type BPSK est employée afin de transmettre cette séquence sur un canal BI-AWGN. La sortie du démodulateur est notée par $(\mathbf{y^s},\mathbf{y^p},\mathbf{y'^p})$. Le premier décodeur SISO va alors travailler sur le couple $(\mathbf{y^s},\mathbf{y^p})$ et le second décodeur SISO sur le couple $(\mathbf{y'^s},\mathbf{y'^p})$ tel que $y'^s = \Pi(y^s)$.

Deux principaux algorithmes peuvent être utilisés dans les décodeurs SISO. La condition nécessaire dans un contexte itératif est qu'ils fournissent des sorties pondérées. Ainsi, il peut s'agir de l'algorithme SOVA ou de l'algorithme BCJR modifié par Berrou [11] afin de fournir des probabilités *a posteriori* pour chaque symbole d'information. Ce dernier, puisqu'il utilise la règle de décision MAP, est aussi appelé algorithme MAP. Néanmoins, comme cette version modifiée fournit des probabilités *a posteriori*, la notation APP rencontrée dans la littérature est plus précise.

Il a été montré que l'algorithme SOVA possède une complexité calculatoire au moins deux fois plus faible que celle de l'algorithme APP [45]. Néanmoins, cette faible complexité calculatoire s'accompagne d'une dégradation des performances de décodage. Celle-ci s'établit d'un facteur deux à un facteur 10 suivant le nombre d'itérations considérées [46]. C'est pourquoi, en pratique, l'algorithme APP est retenu lors de l'implémentation du décodeur SISO. Les calculs menés par l'un des décodeurs doivent être explicités. Cependant, afin de faciliter la lecture de cette section, les calculs intermédiaires sont déportés en Annexe A.1.

Le principe du décodage à maximum *a posteriori* symbole par symbole est de calculer pour tout $k \in [\![1;K]\!]$:

$$\hat{m}_k = \underset{u_k \in \{0,1\}}{\arg\max}\, P(u_k|\mathbf{y}). \tag{1.8}$$

Cependant, si nous prenons en compte la représentation sous forme de treillis du codeur RSC, ce calcul est strictement équivalent à celui des probabilités de transitions formulé par :

$$\hat{m}_k = \underset{i \in \{0,1\}}{\arg\max} \sum_{(s,s') \in \Gamma_{k,i}} P(s_{k-1} = s, s_k = s', \mathbf{y}),$$

où $\Gamma_{k,i}$ représente l'ensemble des chemins de la $k^{\text{ième}}$ section de treillis pour lesquels $u_k = i$.

En exploitant les propriétés du treillis et la relation de Bayes, la probabilité jointe peut s'écrire sous la forme d'un produit de trois termes. Tout d'abord, $\mathbf{y}$ peut être décomposé en trois sous séquences : le passé exprimé par $\mathbf{y_{<k}}$, le présent $\mathbf{y_k}$ et le futur $\mathbf{y_{>k}}$. Ainsi $\mathbf{y} = (\mathbf{y_{<k}},\mathbf{y_k},\mathbf{y_{>k}})$. La probabilité jointe a alors pour expression :

$$\begin{aligned} P(s_{k-1} = s,s_k = s',\mathbf{y}) &= P(s_{k-1} = s,s_k = s',\mathbf{y_{<k}},\mathbf{y_k},\mathbf{y_{>k}}) \\ &= \underbrace{P(\mathbf{y_{>k}}|s_k = s')}_{\beta_k(s')} \times \underbrace{P(s_k = s',\mathbf{y_k}|s_{k-1} = s)}_{\gamma_k(s,s')} \times \underbrace{P(s_{k-1} = s,\mathbf{y_{<k}})}_{\alpha_{k-1}(s)}. \end{aligned}$$

Chaque calcul de probabilité jointe est donc ramené au produit de trois termes :

- la probabilité d'état aller : $\alpha_{k-1}(s)$, représentant la probabilité qu'à l'instant k-1 l'état courant dans le treillis soit s et que la séquence reçue jusqu'alors soit $\mathbf{y_{<k}}$,
- la probabilité d'état retour : $\beta_k(s')$, représentant la probabilité qu'à l'instant k la future séquence soit $\mathbf{y_{>k}}$ sachant que l'état courant est s',
- la probabilité de transition : $\gamma_k(s,s')$ représente la probabilité qu'à l'instant k l'état courant soit s' et que le symbole reçu soit $\mathbf{y_k}$ sachant que l'état précédent est s.

Calcul des probabilités d'état De part leur définition, les probabilités d'état peuvent être exprimées récursivement. Ainsi,

$$\alpha_k(s) = \sum_{s'} \alpha_{k-1}(s')\gamma_k(s,s') \qquad \text{et} \qquad \beta_{k-1}(s) = \sum_{s'} \beta_k(s')\gamma_k(s,s'). \tag{1.9}$$

En supposant que le treillis commence et termine dans l'état s_0, les conditions initiales pour ces deux probabilités sont :

$$\alpha_0(s) = \begin{cases} 1 & \text{si } \ s = s_0 \\ 0 & \text{sinon} \end{cases} \qquad \text{et} \qquad \beta_K(s) = \begin{cases} 1 & \text{si } \ s = s_0 \\ 0 & \text{sinon} \end{cases}$$

Calcul des probabilités de transition Tout d'abord, il est à noter que si la transition dans une tranche de treillis entre l'état s et l'état s' n'existe pas, alors $\gamma_k(s,s') = 0$. Ensuite, en partant de la définition de γ_k et en utilisant la relation de Bayes, nous obtenons :

$$\gamma_k(s,s') = P(m_k)P(\mathbf{y_k}|c_k).$$

Puisque nous considérons que le message produit par la source est équiprobable, nous avons $P(m_k) = 1/2$. Ensuite, en reprenant l'équation 1.2 établissant la probabilité conditionnelle liant la sortie et l'entrée d'un canal AWGN, nous obtenons :

$$\gamma_k(s,s') = P(m_k)\frac{1}{(2\pi\sigma^2)}\exp\left(-\frac{\|\mathbf{y}_k - \mathbf{c}_k\|^2}{2\sigma^2}\right),$$

avec $\mathbf{y}_k = (y_k^s,y_k^p)$ et $\mathbf{c}_k = (c_k^s,c_k^p)$. Finalement, en développant la norme et en supprimant les termes ne dépendant pas de m_k qui se simplifieront dans le calcul suivant, nous obtenons :

$$\gamma_k(s,s') = P(m_k)\exp\left(\frac{y_k^s c_k^s + y_k^p c_k^p}{\sigma^2}\right), \tag{1.10}$$

Calcul des probabilités *a posteriori* L'équation 1.8 permet d'obtenir la décision dure $\hat{m}_k$. Cependant, dans le cadre d'un décodage itératif, il est nécessaire d'avoir une valeur pondérée – représentant la fiabilité de cette décision – exprimée sous forme d'un logarithme de rapport de vraisemblance (LLR) :

$$\begin{aligned} L(m_k) &= \ln \frac{P(m_k = 1|\mathbf{y})}{P(m_k = 0|\mathbf{y})} \\ &= \ln \frac{\sum_{(s,s')\in\Gamma_{k,1}} \alpha_{k-1}(s)\gamma_k(s,s')\beta_k(s)}{\sum_{(s,s')\in\Gamma_{k,0}} \alpha_{k-1}(s)\gamma_k(s,s')\beta_k(s)} \end{aligned} \tag{1.11}$$

Toujours dans le cas d'un codeur RSC de rendement 1/2, d'une modulation BPSK (prenant les 2 valeurs possibles ± 1) et d'un canal BI-AWGN, le LLR peut être décomposé en trois éléments :

$$\begin{aligned} L(m_k) &= \ln \frac{\sum_{(s,s')\in\Gamma_{k,1}} \alpha_{k-1}(s) \cdot P(m_k) \exp \frac{y_k^s c_k^s + y_k^p c_k^p}{\sigma^2} \cdot \beta_k(s)}{\sum_{(s,s')\in\Gamma_{k,0}} \alpha_{k-1}(s) \cdot P(m_k) \exp \frac{y_k^s c_k^s + y_k^p c_k^p}{\sigma^2} \cdot \beta_k(s)} \\ &= \frac{2}{\sigma^2} y_k^s + \ln \frac{P(m_k = 1)}{P(m_k = 0)} + \ln \frac{\sum_{(s,s')\in\Gamma_{k,1}} \alpha_{k-1}(s) \cdot P(m_k) \exp \frac{y_k^p c_k^p}{\sigma^2} \cdot \beta_k(s)}{\sum_{(s,s')\in\Gamma_{k,0}} \alpha_{k-1}(s) \cdot P(m_k) \exp \frac{y_k^p c_k^p}{\sigma^2} \cdot \beta_k(s)} \\ &= L_c y_k^s \quad + L^a(m_k) \quad + L^e(m_k) \end{aligned} \tag{1.12}$$

Le premier terme correspond à l'effet du canal sur les bits systématiques. Le second est l'information *a priori*. Enfin le dernier terme est l'information extrinsèque qui ne dépend ni des probabilités a priori, ni de l'information systématique.

Ainsi, lors du décodage itératif d'un turbo code (cf. Figure 1.11), le décodeur SISO_1 reçoit le couple $(\mathbf{y^s},\mathbf{y^p})$. À partir de l'équation 1.12, les LLR *a posteriori* sont calculés en considérant les LLR *a priori* nuls. En effet, les bits d'information sont équiprobables. Ensuite, afin d'en extraire les informations extrinsèques (notées $\mathbf{L_{12}^e}$), les informations du canal correspondant aux bits systématiques sont soustraites. L'information extrinsèque est alors traitée par l'entrelaceur Π. Elles sont alors utilisées en tant qu'information *a priori* par le décodeur SISO_2 conjointement avec $(\mathbf{y'^s},\mathbf{y'^p},)$ pour fournir les informations extrinsèques (notées $\mathbf{L_{21}^e}$). Ces dernières sont permutées selon Π^{-1} afin que le SISO_1 les utilise lors de l'itération suivante dans le domaine naturel en tant qu'information *a priori*.

Ce processus est répété jusqu'à ce qu'un nombre fixé d'itérations soit atteint ou jusqu'à ce qu'un critère d'arrêt soit vérifié.

Comme présenté précédemment, l'algorithme APP a pour principal inconvénient sa complexité calculatoire. Cette dernière provient du nombre conséquent de multiplications et de calculs d'exponentielles. Afin de réduire cette complexité calculatoire plusieurs simplifications ont été proposées.

1.2.2.2 Simplifications de l'algorithme APP

La simplification de l'algorithme APP passe par la transposition dans le domaine logarithmique des différentes probabilités calculées. Cette version se nomme log-APP. Les détails des calculs sont présentés en Annexe A.2.

Les probabilités précédentes sont alors remplacées par les métriques suivantes :
- La métrique d'état aller : $\tilde{\alpha}_k(s) = \ln \alpha_k(s)$,
- La métrique d'état retour : $\tilde{\beta}_k(s) = \ln \beta_k(s)$,
- La métrique de branche : $\tilde{\gamma}_k(s,s') = \ln \gamma_k(s,s')$.

En introduisant l'opérateur max* défini par $\max^*(x,y) = \ln(e^x + e^y)$, il vient naturellement :

$$\tilde{\alpha}_k(s) = \max_{s'}{}^* \left(\tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s',s)\right) \quad \text{et}$$
$$\tilde{\beta}_{k-1}(s) = \max_{s'}{}^* \left(\tilde{\beta}_k(s') + \tilde{\gamma}_k(s',s)\right)$$

Les conditions initiales sont dorénavant :

$$\tilde{\alpha}_0(s) = \begin{cases} 0 & \text{si } s = s_0 \\ -\infty & \text{sinon} \end{cases} \quad \text{et} \quad \tilde{\beta}_K(s) = \begin{cases} 0 & \text{si } s = s_0 \\ -\infty & \text{sinon} \end{cases}$$

Finalement les valeurs *a posteriori* deviennent :

$$L(\tilde{m_k}) = \max_{(s,s')\in\Gamma_{k,1}}{}^* \left(\tilde{\alpha}_{k-1}(s)\tilde{\gamma}_k(s,s')\tilde{\beta}_k(s)\right) - \max_{(s,s')\in\Gamma_{k,0}}{}^* \left(\tilde{\alpha}_{k-1}(s)\tilde{\gamma}_k(s,s')\tilde{\beta}_k(s)\right). \tag{1.13}$$

Cependant, en l'état, la complexité calculatoire demeure. C'est pourquoi, un travail complémentaire sur l'opérateur max* est nécessaire. Ainsi, il est aisément démontrable (cf A.2.2) que :

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

Les différentes simplifications de l'algorithme log-APP vont alors consister en des approximations plus ou moins fines du terme logarithmique, noté dans la suite $f_c(x,y)$.

La première approximation est nommée constant-log-APP [47]. Si la différence $|x-y|$ est inférieure à un seuil, alors une constante de correction est ajoutée à la valeur du maximum :

$$\max^*(x,y) = \max(x,y) + \begin{cases} 0 & \text{si } |x-y| > T \\ C & \text{si } |x-y| \leqslant T. \end{cases} \tag{1.14}$$

Dans [48], les valeurs présentées pour T et C valent respectivement 1,5 et 0,5.

Dans le cadre du linear-log-APP, le terme logarithmique est approximé par une fonction affine [49] :

$$\max^*(x,y) = \max(x,y) + \begin{cases} 0 & \text{si } |x-y| > T \\ a(|y-x| - T) & \text{si } |x-y| \leqslant T, \end{cases} \tag{1.15}$$

avec $a = 0{,}24904$ et $T = 2{,}5068$.

Finalement, la dernière approximation consiste à ne pas ajouter de terme de correction. Ainsi, l'opérateur max* est remplacé uniquement par l'opérateur max. Cette approximation, appelée max-log-APP (ML-APP) est celle à plus faible complexité calculatoire. Cependant, les performances de décodage se situent alors à environ 0,5 dB de celles de l'algorithme log-APP.

Ces trois différentes approximations sont récapitulées dans la Figure 1.12. Au niveau des performances de décodage, plus l'approximation a une complexité calculatoire élevée, plus elle se rapproche de la fonction originelle et donc des performances optimales. Des comparaisons des impacts de ces approximations sont présentées dans [45] et [50].

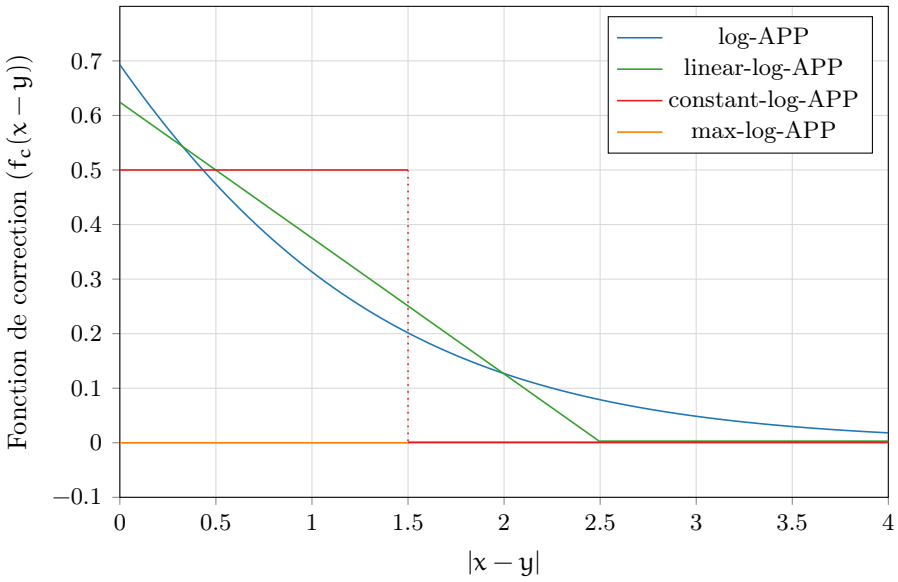


FIGURE 1.12 – Fonctions de corrections des différentes approximations de l'algorithme log-APP.

En 2000, afin de réduire l'impact de l'approximation, il a été proposé d'utiliser un facteur d'échelle sur l'information extrinsèque avant qu'elle soit consommée par le décodeur SISO [51]. Cette méthode, appelée enhanced-max-log-APP (EML-APP), est inspirée de [26] dans le cadre de l'algorithme SOVA. En effet, il avait été remarqué que les valeurs des LLR calculés par les décodeurs SISO étaient sur-évaluées. Ce facteur d'échelle (SF) peut être constant au cours du processus itératif. Dans ce cas, la valeur de 0,75 est souvent choisie. En effet, elle peut être facilement réalisée en sommant un décalage vers la gauche de 2 et un décalage vers la gauche de 1. Autrement, la remise à l'échelle peut être adaptative. Le facteur évolue alors de 0,5 pour la première itération jusqu'à 1,0 pour la dernière. En moyenne, grâce à l'utilisation de ce facteur, les performances de décodage avec l'EML-APP ne se situent plus qu'à 0,1 dB de celles du log-APP. Dans [52], l'influence de la valeur du facteur d'échelle est étudiée afin de minimiser le taux d'erreur binaire (BER) et le taux d'erreur trame (FER).

1.2.3 Les turbo codes double binaires

En 1999, les turbo codes non binaires ont été introduits [53]. À la différence des turbo codes binaires, les turbo codes double binaires traitent 2 bits en parallèle. Il en découle que les codeurs RSC constituants sont eux mêmes double binaires. Pour ce faire, des extensions multi binaires ont été développées. Le codeur constitutif employé est alors de rendement nominal $n/(n+1)$ avec n le nombre de donnés binaires simultanément présentes à l'entrée du codeur [54]. La figure 1.13 présente un tel codeur sans faire apparaître la sortie pour des soucis de lisibilité. Il est toutefois à noter qu'utiliser $n > 2$ n'apporte qu'un intérêt limité en terme de performances de décodage pour des RSC de mémoire 3 ou 4 [55]. En effet, trop peu de diversité dans l'interconnexion ne peut être obtenus dans de tels cas.

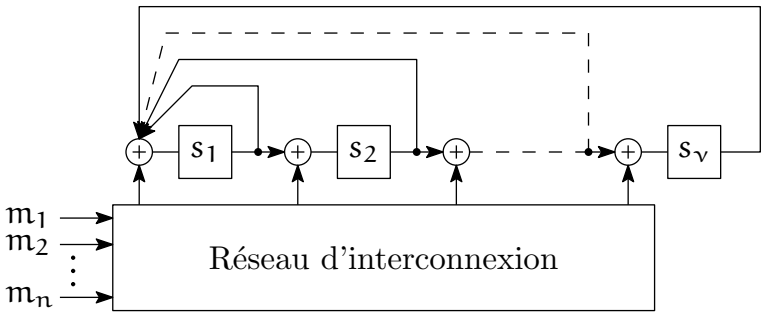


FIGURE 1.13 – Structure générique d'un codeur RSC m-binaire et de taille de contrainte $\nu + 1$.

Pour le cas double binaire, le rendement des codes constituants est donc 2/3. Le rendement nominal du turbo code obtenu est donc 1/2. Cependant, il est possible d'ajouter une seconde sortie d'information redondantes, pour chaque code constituant, afin d'obtenir un turbo code de rendement 1/3.

Selon [56], les avantages des turbo codes double binaires sont les suivants :

- une plus grande distance minimale,
- un moindre poinçonnage : puisque le turbo code est de rendement nominal 1/2, moins de poinçonnage – comparativement aux turbo codes binaires – est nécessaire pour atteindre de forts rendements,
- une latence réduite : puisque les données sont traitées par groupe de deux bien qu'il y ait plus de calculs par tranche de treillis à mener,
- une meilleure robustesse du décodeur : la dégradation au niveau des performances entre l'algorithme MAP et ses versions simplifiées est moins importante que dans le cadre binaire.

Dans la littérature, l'entrée des codeurs constituants est dénoté par le couple (A_k,B_k). Les sorties correspondantes sont notées (Y_k,W_k). Si la trame à coder contient K bits, alors $K/2$ symboles sont codés. Dans ce cas, nous avons $k \in [\![0,K/2-1]\!]$.

L'emploi de codes double binaires permet un entrelacement des données sur deux niveaux. Le premier agit au niveau intra-symboles et le second au niveau inter-symboles. Ainsi, la première permutation consiste à permuter l'ordre du couple tous les deux symboles :

si k est pair, (A_k,B_k) devient (B_k,A_k). Pour le second niveau d'entrelacement, une permutation comme celles présenté en section 1.2.1.3 est appliquée.

Le décodage des turbo codes double binaires est basé sur une extension de l'algorithme log-APP. Cependant, au lieu de calculer un seul LLR par tranche de treillis, il faut, dans ce cas, en calculer trois. En effet, selon 1.10, chaque LLR correspond au rapport entre la probabilité qu'un 1 soit émis et la probabilité qu'un 0 soit émis. Dans le cadre double binaire, il s'agit de calculer le rapport entre la probabilité de chaque symbole possiblement émis et la probabilité qu'un symbole 0 soit émis. Ceci est exprimé par :

$$L_{(a,b)}(A_k,B_k) = \ln \frac{P(A_k = a,B_k = b|y)}{P(A_k = 0,B_k = 0|y)},$$

avec (a,b) valant (0,1), (1,0) ou (1,1).

Il en suit que lors du processus itératif, trois valeurs extrinsèques sont échangées par tranche du treillis. Finalement, les décisions binaires sont prises en testant le signe de :

$$L(A_k) = \max^*\left(L_{(1,0)}(A_k,B_k),L_{(1,1)}(A_k,B_k)\right) - \max^*\left(L_{(0,1)}(A_k,B_k),0\right)$$
$$L(B_k) = \max^*\left(L_{(0,1)}(A_k,B_k),L_{(1,1)}(A_k,B_k)\right) - \max^*\left(L_{(1,0)}(A_k,B_k),0\right)$$

puisque, par définition, $L^e_{(0,0)}(A_k,B_k) = 0$.

Pour conclure, les turbo codes double binaires, tout comme leurs antécédents binaires ont été standardisés. Les principales caractéristiques de ces codes sont présentées en section suivante.

1.2.4 Les turbo codes standardisés

Peu après la publication des premiers résultats confirmant les performances de décodage des turbo codes, plusieurs comités de standardisation se sont intéressés à cette nouvelle famille de codes correcteurs d'erreurs.

Ainsi, en 1999, le Comité Consultatif pour les Systèmes de Données Spatiales (CCSDS, qui regroupe différentes agences spatiales à travers le monde), propose dans leur livre de recommandations d'employer un turbo code à 16 états dans le cadre des communications avec l'espace lointain [57]. Aujourd'hui, nous retrouvons des turbo codes essentiellement dans des contextes de télécommunication mobile et de diffusion vidéo numérique.

Depuis la fin des années 90, le 3GPP a adopté un turbo code à 8 états dans le cadre de la technologie de téléphonie mobile de troisième génération UMTS. En même temps, son concurrent, le 3GPP2, fait de même pour la norme CDMA2000, avec un turbo code très proche de celui de l'UMTS mais proposant plus de rendements. Enfin, dans le cadre de la technologie de téléphonie mobile de quatrième génération, les standards LTE et LTE-Advanced [41] adoptent les mêmes codes constituants mais un nouvel entrelacement basé sur des permutations QPP est retenu.

Le consortium européen DVB, dès 1994, s'est intéressé aux turbo codes pour le standard de diffusion vidéo numérique terrestre DVB-T. Cependant, ce n'est qu'en 2000 que les

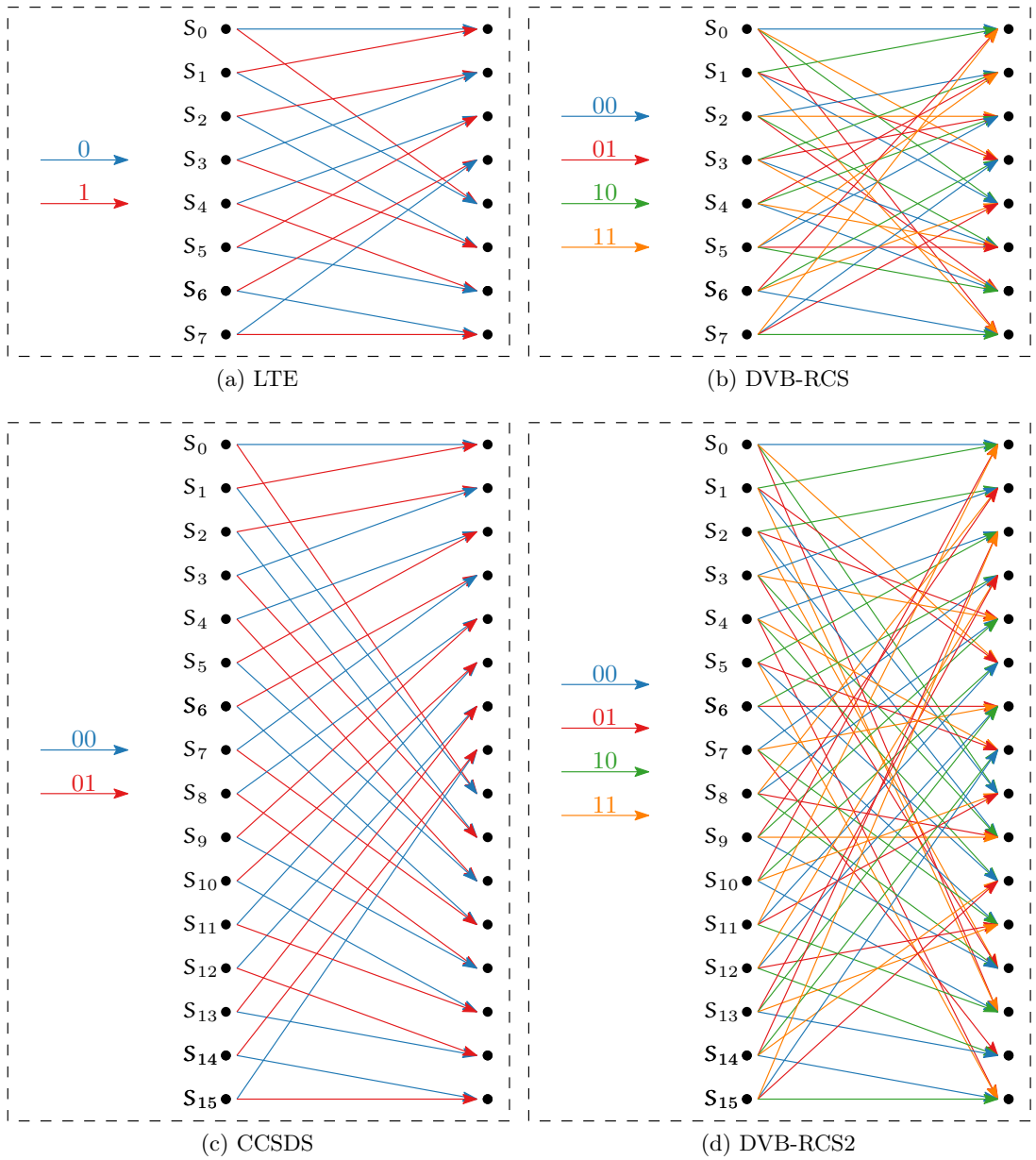
turbo codes sont standardisés dans ce contexte afin d'assurer la voie retour permettant l'ajout de services tels que des programmes TV interactifs ou l'accès à internet. Ainsi deux standards, le DVB-RCS [37] et le DVB-RCT [39] spécifient respectivement la voie retour du standard DVB-S et la voie retour du standard DVB-T. Ces deux turbo codes standardisés sont des codes double binaires 8 états et leur entrelaceur est basé sur une permutation ARP. Depuis 2012, la seconde génération du standard DVB-RCS2 [38] comprend un turbo code similaire mais à 16 états.

D'autres standards utilisent des turbo codes tels que le WiMAX [58] pour des communications numériques sans fil. Les turbo codes sont aussi présents dans les standards HomePlug AV et HomePlug AV2 [59] spécifiant des communications par courant porteur en ligne. Ces trois turbo codes sont similaires à celui du DVB-RCS.

Dans la suite de ce manuscrit, les standards CCSDS, LTE, DVB-RCS et DVB-RCS2 seront utilisés comme cadre applicatif, car à eux 4 ils représentent une diversité conséquente quant aux turbo codes usités. Le tableau 1.2 récapitule les propriétés de ces turbo codes. Les treillis associés aux codeurs RSC élémentaires des turbo codes de chacun de ces standards sont présentés en Figure 1.14. Les performances de décodage de ces turbos codes sont présentées, de manière non exhaustive, en Figures 1.15 à 1.18. Pour toutes ces simulations Monte Carlo, 100 trames erronées ont été obtenues par point de SNR. L'algorithme de décodage est l'EML-APP avec comme facteur d'échelle 0,75 durant 8 itérations. Dans le cas des turbo codes doubles binaires, le facteur de remise à l'échelle vaut 0,5 pour les deux premières itérations et 0,85 pour les 6 suivantes.

Tableau 1.2 – Les turbo codes standardisés considérés au long de ce manuscrit.

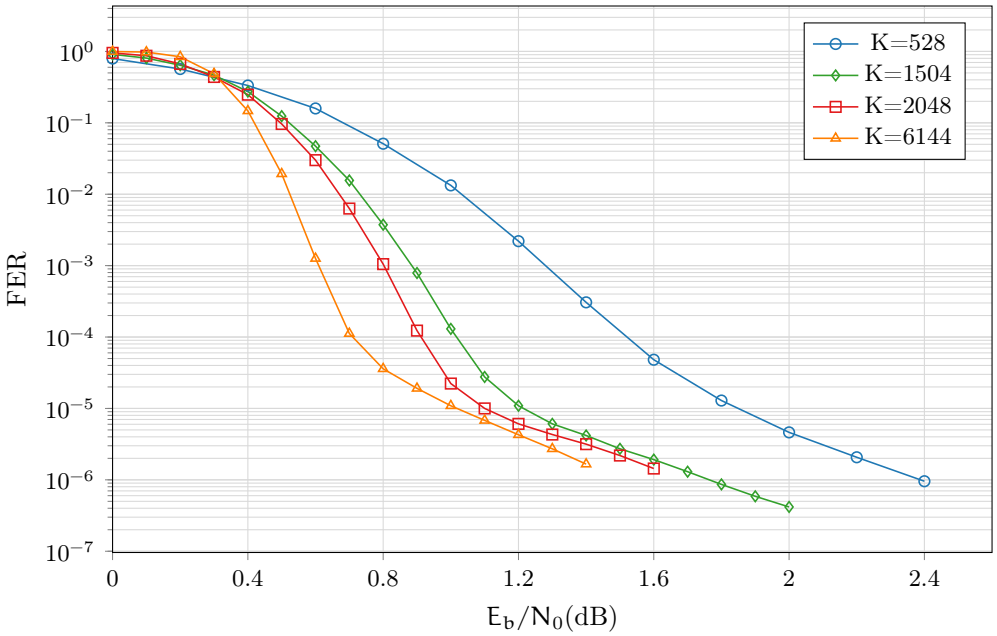
	Binaire		Double binaire	
	LTE	**CCSDS**	**DVB-RCS**	**DVB-RCS2**
Type de communication	Mobiles		Satellitaires	
Nombre d'états	8	16	8	16
Terminaison du treillis	Retour à l'état 0		Circulaire	
Tailles de trame (bits)	40 → 6144	1784 → 8920	96 → 1728	128 → 3008
Rendements	Rate-Matching	1/6 → 1/2	1/3 → 6/7	1/3 → 7/8



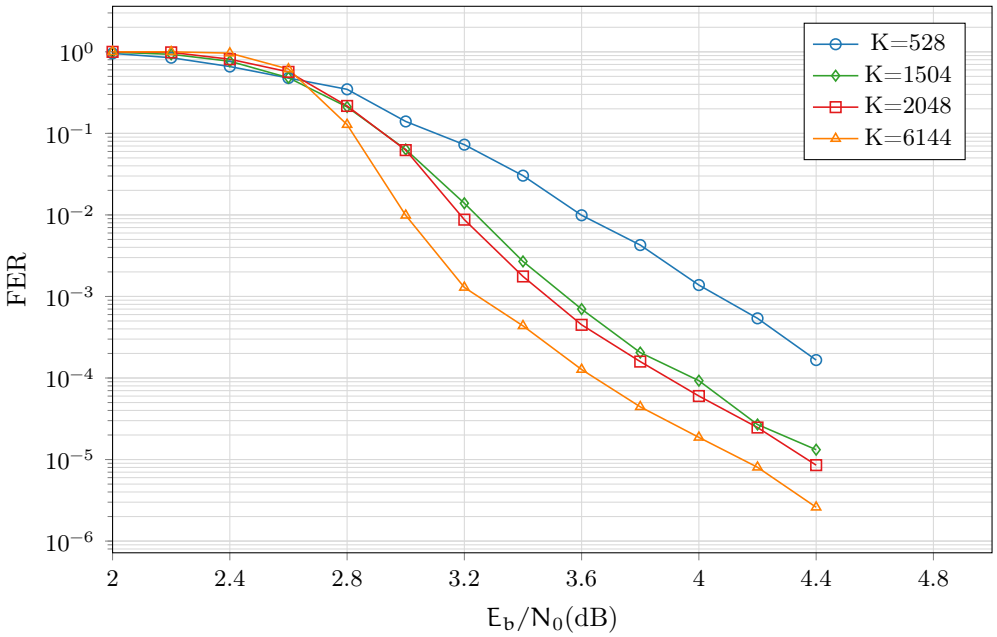
(a) LTE (b) DVB-RCS

(c) CCSDS (d) DVB-RCS2

FIGURE 1.14 – Treillis associés aux codeurs RSC élémentaires des turbo codes du tableau 1.2.

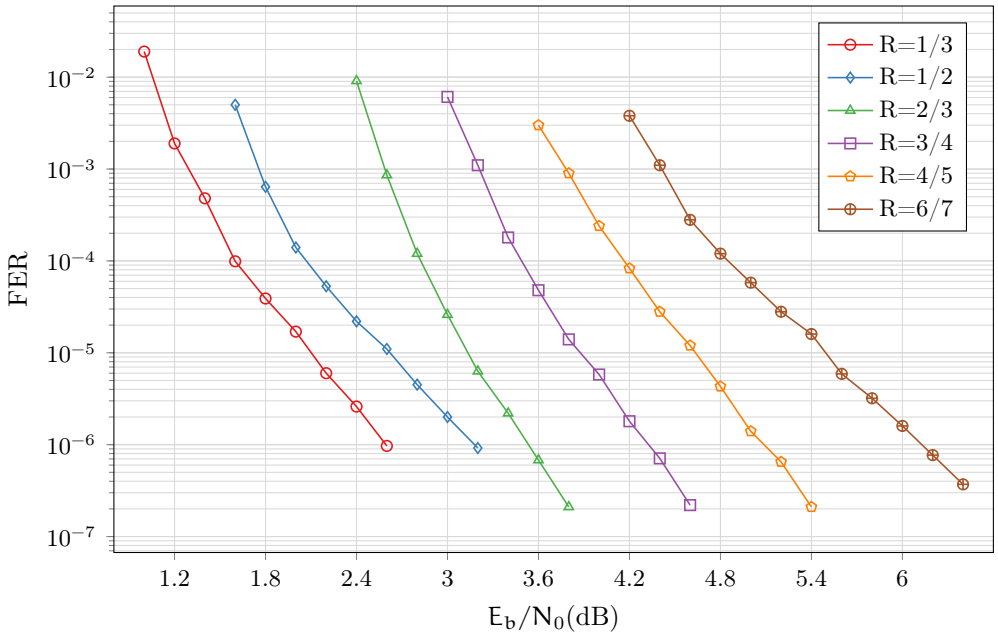


(a) LTE R=1/3

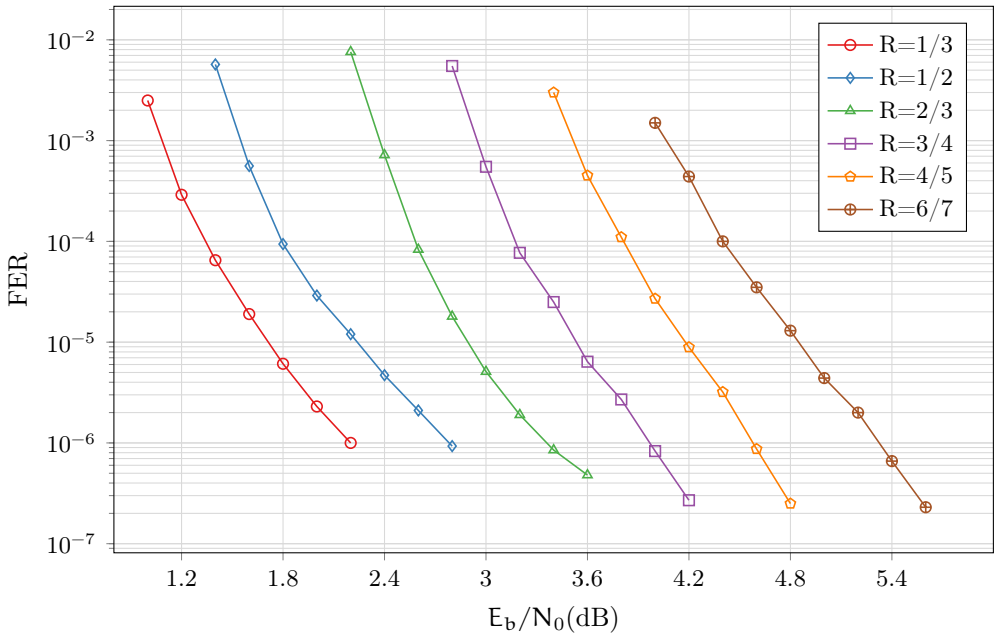


(b) LTE R=4/5

FIGURE 1.15 – Taux d'erreur trame pour le standard LTE pour deux rendements et différentes tailles de trame. Décodage par l'algorithme EML-MAP effectuant 8 itérations.



(a) DVB-RCS - K=440



(b) DVB-RCS - K=752

FIGURE 1.16 – Taux d'erreur trame pour le standard DVB-RCS pour plusieurs rendements et deux tailles de trame. Décodage par l'algorithme EML-MAP effectuant 8 itérations.

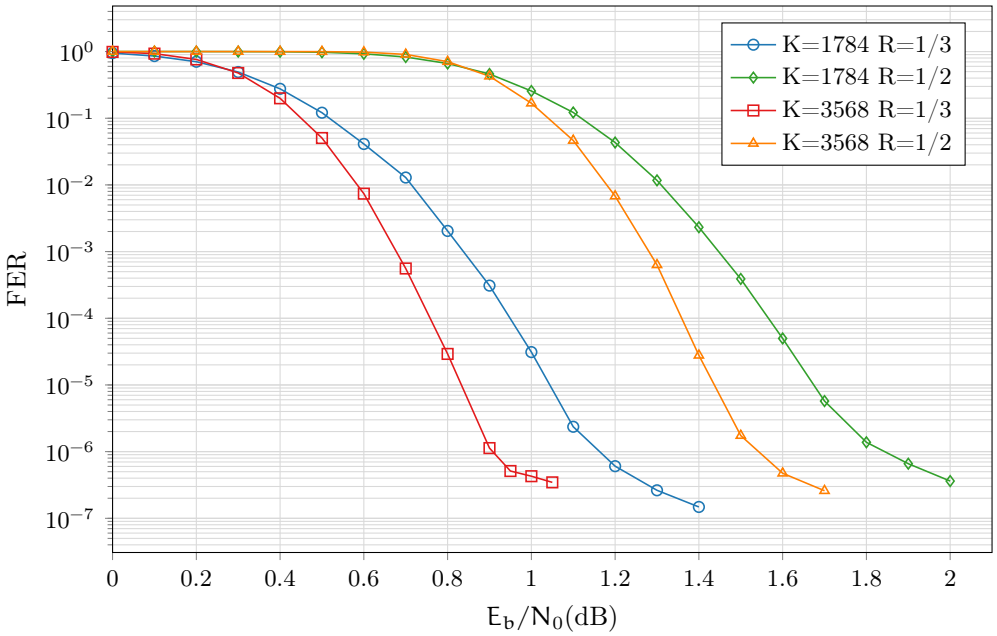


FIGURE 1.17 – Taux d'erreur trame pour le standard CCSDS pour deux rendements et deux tailles de trame. Décodage par l'algorithme EML-MAP effectuant 8 itérations, critère d'arrêt génie.

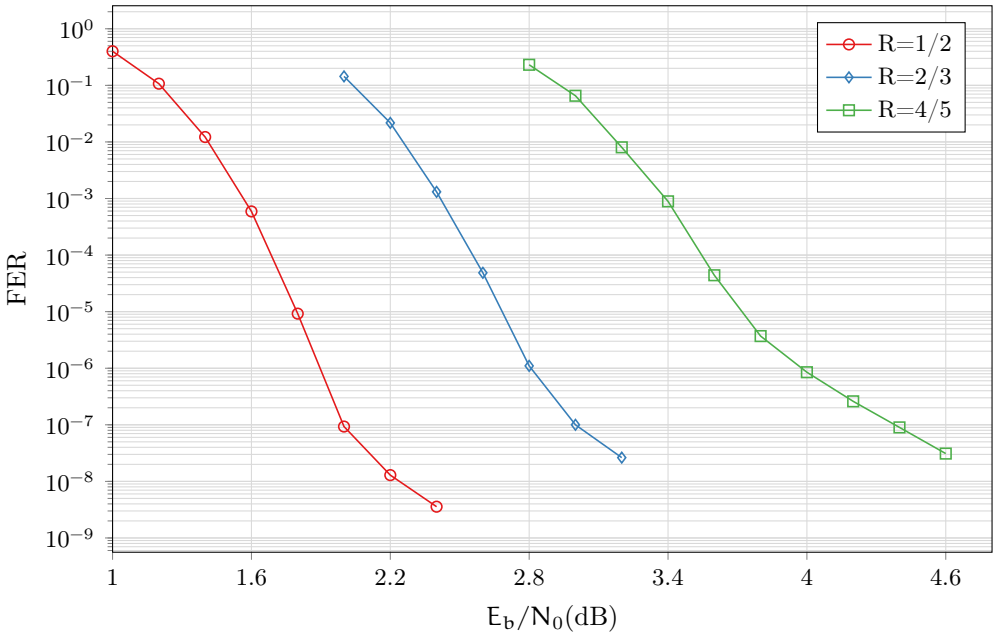


FIGURE 1.18 – Taux d'erreur trame pour le standard DVB-RCS2. K=752, différents rendements. Décodage par l'algorithme EML-MAP effectuant 8 itérations, critère d'arrêt génie.

A partir des Figures 1.15 à 1.18, nous pouvons remarquer que les performances de décodage des turbo codes peuvent être différenciées selon trois régions :

- La première zone, correspondant aux faibles valeurs de SNR est nommé « zone de non convergence ». Pour de telles valeurs de SNR, le système codé peut-être moins performant que le système de transmission non codé.
- Passé le seuil de convergence, une faible augmentation du SNR implique une baisse significative du taux d'erreur. Cette zone est nommée « waterfall region » en anglais.
- Enfin, dans la région du « plancher d'erreurs », la courbe de performance s'aplatit : le taux d'erreur baisse légèrement lorsque le SNR augmente.

Afin d'obtenir des codes approchant la capacité du canal, il est nécessaire que le seuil de convergence soit atteint pour des valeurs de SNR les plus faibles possible. La méthode du diagramme EXIT [60] permet de caractériser et de prévoir ce seuil.

Dans la zone de convergence, l'augmentation du nombre d'itérations améliore les performances. En revanche, dans la zone du plancher d'erreurs, seulement quelques itérations sont nécessaires et les suivantes n'amènent qu'un gain marginal en terme de performances.

Le plancher d'erreurs peut limiter l'utilisation des turbo codes pour des applications nécessitant de très faibles taux d'erreur. Comme cette zone de gain asymptotique est essentiellement influencée par la distance minimale de Hamming du code, le problème du plancher d'erreurs est discuté dans la section suivante.

1.2.5 Le problème du plancher d'erreurs

1.2.5.1 Performances asymptotiques sur canal BI-AWGN

Nous avons préalablement expliqué qu'en raison de l'étape d'entrelacement, un décodage ML de turbo codes n'est pas possible (cf. 1.1.4.2). Ainsi, les méthodes de décodage itératives employées sont sous-optimales. Cependant, elles approchent les performances du décodage ML pour des valeurs de SNR moyennes à élevées. Dans cette section, nous allons estimer les performances d'un décodeur ML pour un turbo code utilisé sur le canal BI-AWGN.

Puisque les codes constituants et l'entrelaceur d'un turbo code sont linéaires, lui-même l'est sous réserve d'omettre la séquence de terminaison. Cette propriété de linéarité combinée à la symétrie du canal considéré implique que la probabilité d'erreur sachant qu'un certain mot de code $\mathbf{c}$ a été émis est la même que si un autre mot de code $\mathbf{c}'$ était transmis. Ceci permet de simplifier l'analyse en considérant que le mot de code tout à zéro ($\mathbf{0}$) est transmis.

Ainsi, la probabilité qu'un mot soit erroné vaut :

$$\begin{aligned} P_{cw} &= P(\text{erreur} \mid \mathbf{0}) \\ &= P(\underset{\hat{\mathbf{c}} \neq \mathbf{0}}{\text{decide}}\, \hat{\mathbf{c}} \mid \mathbf{0}) \\ &\leqslant \sum_{\hat{\mathbf{c}} \neq \mathbf{0}} P(\text{decide}\, \hat{\mathbf{c}} \mid \mathbf{0}). \end{aligned} \tag{1.16}$$

Ceci correspond au fait que le décodeur choisisse le mot de code $\hat{\mathbf{c}}$ alors que $\mathbf{0}$ est transmis.

Soit $m(\mathbf{c})$ la représentation après transmission du mot de code $\mathbf{c}$. Nous pouvons alors écrire $m(\mathbf{0}) = (\sqrt{E_c}, \sqrt{E_c}, ..., \sqrt{E_c})$, avec $E_c = R \times E_b$.
Posons P_w la probabilité que le bruit blanc gaussien implique que le vecteur reçu $\mathbf{r}$ soit plus près de $m(\hat{\mathbf{c}})$ que de $m(\mathbf{0})$, avec $m(\hat{\mathbf{c}}) = (\sqrt{E_c}, -\sqrt{E_c}, ..., \sqrt{E_c})$ et w le nombre de positions différant de $m(\mathbf{0})$.

Ainsi, $m(\hat{\mathbf{c}})$ et $m(\mathbf{0})$ sont séparés par la distance (Euclidienne) $d_E = 2\sqrt{wE_c} = 2\sqrt{wRE_b}$. Introduisons la fonction $Q(x)$ représentant la probabilité qu'une variable aléatoire Gaussienne ait une valeur plus grande que x fois l'écart type autour de la moyenne ($x = \frac{y_\mu}{\sigma}$). Nous pouvons alors exprimer P_w en utilisant cette fonction puisque P_w représente la probabilité que le bruit en direction de $\hat{\mathbf{c}}$ soit plus grand que $\frac{d_E}{2} = \sqrt{wRE_b}$:

$$P_w = Q\left(\frac{d_E}{2\sigma}\right) = Q\left(\frac{\sqrt{wRE_b}}{\sigma}\right) = Q\left(\sqrt{\frac{2wRE_b}{N_0}}\right). \tag{1.17}$$

Or, en posant A_w la multiplicité ou le nombre de mots de code de poids w, nous pouvons réécrire l'équation 1.16 de la sorte :

$$P_{cw} \leqslant \sum_w A_w P_w.$$

En reprenant l'équation 1.17, nous pouvons finalement écrire :

$$P_{cw} \leqslant \sum_w A_w Q\left(\sqrt{\frac{2wRE_b}{N_0}}\right) \tag{1.18}$$

Cette équation donne donc une borne supérieure quant à la probabilité d'erreur trame en utilisant un décodeur ML. En découle son équivalent binaire exprimé par :

$$P_b \leqslant \sum_w \frac{W_w}{K} Q\left(\sqrt{\frac{2wRE_b}{N_0}}\right) \tag{1.19}$$

où W_w est la somme des poids de Hamming des A_w séquences binaires générant des mots de code de poids de Hamming w.

Ainsi, selon les valeurs (w,A_w,W_w) associées à un code, nous pouvons exprimer des bornes quant aux performances obtenues avec un décodeur ML. Les valeurs (w,A_w,W_w) se nomment spectre de distance du code et les bornes sont les *bornes de l'union*. Or, il

s'avère qu'en utilisant un décodeur sous-optimal itératif, les performances approchent les bornes dans le régime asymptotique.

C'est pourquoi, de nombreux travaux de recherche ont porté sur l'extraction des spectres de distance afin de prévoir le comportement asymptotique d'un turbo code.

1.2.5.2 Méthodes d'obtention du spectre de distance de turbo codes

Les premières méthodes visant à obtenir le spectre de distance sont basées sur des approches par force brute [61]. Leur principe est de considérer toutes les séquences possibles pour le premier codeur constitutif, de les entrelacer et de les coder avec le second codeur constitutif afin d'obtenir la distance du code. Cependant, pour des distances minimales conséquentes, cette recherche devient trop complexe à mener.

En 2001, un algorithme basé sur l'utilisation de sous codes contraints a été proposé par Garello *et al.* [62]. Cet algorithme permet de déterminer exactement les premiers termes du spectre. Cependant, lorsque la taille du code et la distance minimale sont conséquentes, le temps de calcul est prohibitif rendant cet algorithme difficile à exploiter.

Ainsi, plusieurs techniques se basant sur la capacité du décodeur et non sur les propriétés du codes ont été proposées. En 2002, Berrou *et al.* ont proposé une méthode rapide basée sur la capacité d'un décodeur à compenser l'insertion d'un événement d'erreur [63]. Son principe est d'insérer une impulsion croissante A_i dans le mot de code tout-à-zéro, ce pour toutes les positions i, $(0 \leqslant i \leqslant K-1)$. Lorsque l'impulsion est suffisamment grande, le décodeur ne converge plus vers le mot de code $\mathbf{0}$ (tout à zéro). Posons A_i^* l'amplitude maximale de l'impulsion à la position i ne mettant pas en défaut le décodeur. Il a alors été montré que dans le cas d'un décodeur ML, la distance impulsionnelle définie par $d_{imp} = \min_i(A_i)$ est aussi la distance minimale du code. Puisque le décodage des turbo codes n'est pas ML, la véritable d_{min} n'est pas toujours obtenue par cette méthode. Il a été constaté que cette méthode est plutôt pessimiste mais néanmoins, des distances plus grandes que la vraie d_{min} ont été trouvées [64].

En 2004, Garello *et al.* ont proposé une méthode similaire à la précédente [65]. Dans celle-ci, l'impulsion d'erreur empêche le décodeur de converger vers le mot de code $\mathbf{0}$. Il converge donc vers une séquence d'entrée non-nulle. En encodant cette séquence d'entrée, un mot de code non-nul est obtenu. Le poids de Hamming de ce mot de code est alors une borne supérieure de la distance minimale du code. Cependant, cette méthode propose des résultats probants uniquement pour des codes possédant de faibles d_{min} [64].

Enfin, en 2005, Crozier *et al.* ont proposé deux améliorations, découlant du travail précédent [66]. Une première impulsion est placée à une position i dans la partie systématique du mot de code. Une seconde est ajoutée à une position j. Ensuite le décodage est mené et les distances et multiplicités associées sont relevées de la même manière que dans [65]. Les auteurs considèrent que placer la seconde impulsion dans la zone $[\![i; i+2D]\!]$, avec D la distance minimale du code estimée est suffisant. Cependant, dans [64], il est précisé que la fiabilité de cette méthode est accrue si $j \in [\![i; K-1]\!]$. Pis encore, selon nos

TABLEAU 1.3 – Spectre de distance pour trois turbo codes du standard LTE.

K	d/A_d/W_d							
528	23/1/1	24/1/2 33/16/54	25/1/3 34/12/38	27/3/9 35/20/80	28/3/6 36/24/92	29/6/16 37/163/527	31/8/28 38/1019/6030	32/5/18 39/163/527
2048	27/1/1	28/2/4 36/1/4	29/1/3 37/3/8	30/1/2 38/5/16	31/3/9 39/12/18	33/2/6 40/6/24	34/1/4 41/16/62	35/2/6 41/16/62
6144	26/1/2	32/1/4	33/1/3 40/3/4	34/1/4 41/7/3	36/1/4 42/9/34	37/1/3 43/10/46	38/1/4 44/6/30	39/2/5 45/8/40

expérimentations, dans le cas de turbo codes non circulaires, afin d'approcher au plus près les résultats fournis par la méthode [62], imposer $\mathsf{i} \in [\![0; \mathsf{K}-1]\!]$ et $\mathsf{j} \in [\![\mathsf{i}; \mathsf{N}]\!]$ semblent offrir le meilleur compromis entre la finesse des résultats et le temps de simulation.

Exemples de prévision du plancher d'erreurs En utilisant la méthode de la double impulsion de Crozier [66] ou la méthode utilisant les sous-codes contraints de Garello [62], nous pouvons calculer les spectres de distance pour les différents turbo codes standardisés.

Le tableau 1.3 présente les premiers termes du spectre de distance pour trois turbo code du standard LTE : $\mathsf{K} \in \{528; 2048; 6144\}$ et $\mathsf{R} = 1/3$. Nous pouvons remarquer que les multiplicités sont faibles et que les distances minimales sont globalement croissantes avec K.

En utilisant ces données et l'équation 1.18, nous pouvons ajouter sur les courbes de performances de la Figure 1.15 la borne de l'union. Il est remarquable que pour les 3 codes considérés, la courbe de performance, en terme de taux d'erreur trame, est parallèle à la borne de l'union lorsque le décodeur est en régime asymptotique. Pour K=528 et K=2048, elle est située à un facteur d'approximativement 2 de la borne de l'union. Pour K=6144, cet écart est un peu plus important et représente un facteur 4. L'écart à la borne de l'union s'explique par la sous-optimalité de l'algorithme de décodage conjointement avec le faible nombre d'itération employé, à savoir 10.

1.2.6 Les méthodes d'abaissement du plancher d'erreurs

Afin d'abaisser le plancher d'erreurs des turbo codes, différentes méthodes peuvent être employées.

1.2.6.1 Modification des paramètres des turbo codes

Dans un premier temps, la longueur de contrainte des codeurs convolutifs peut être augmentée. C'est notamment l'une des stratégies employée lors de la ratification de la seconde version du standard DVB-RCS. En effet, les codeurs convolutifs sont passés de 8 états à 16 états. Il en résulte un plancher d'erreurs abaissé d'environ une décade. En revanche, la complexité calculatoire du décodeur double lorsque le nombre d'états double.

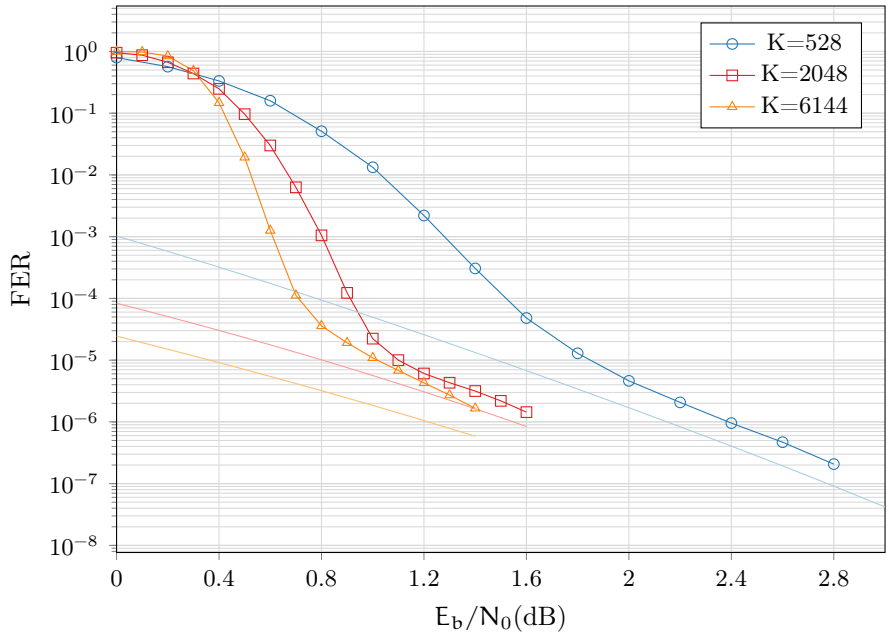


FIGURE 1.19 – Bornes de l'union et performances de décodages par l'algorithme EML-MAP effectuant 8 itérations pour trois turbo codes du standard LTE.

Ainsi, hormis le paramètre de la longueur de contrainte, le spectre de distance est régi par l'entrelaceur. C'est pour cela que de nombreuses recherches se sont concentrées sur la fonction d'entrelacement. De plus, l'entrelaceur n'a, *a priori*, pas d'impact sur la complexité de décodage. Néanmoins, des limites existent quant aux distances et multiplicités minimales atteignables. Aujourd'hui, nous avons à disposition des structures d'entrelaceurs (comme évoqué en section 1.2.1.3) permettant d'obtenir des distances minimales aussi grandes que nécessaire. Cependant, des innovations consistent en la conception d'entrelaceurs conjointement à celle des matrices de poinçonnage. Cette conception conjointe permet d'obtenir des distance minimales très importantes pour des turbo codes à haut rendement [67].

Cependant, un autre vecteur d'optimisation des performances se situe au niveau du décodage itératif. En effet, les méthodes de décodage itératives sont sous-optimales. Ainsi, plusieurs méthodes de décodage effectuant des traitements complémentaires après un turbo décodage usuel ont été considérées. Ces approches peuvent résulter en l'obtention de plusieurs séquences d'information probables. Ainsi, l'utilisation d'un code détecteur d'erreurs concaténé en amont du turbo codeur est une approche envisageable. Plus encore, la concaténation série de différents codes permet d'accroître les performances des turbo codes.

1.2.6.2 Concaténation de code en série et décodages associés

Parmi les divers codes qu'il est possible de concaténer avec un turbo code, trois ont particulièrement été considérés dans la littérature. La première section présente l'emploi

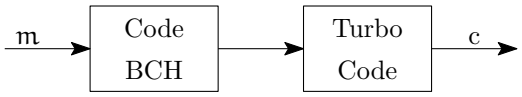


FIGURE 1.20 – Concaténation série d'un code BCH et d'un turbo code.

d'un code BCH qui permet de corriger, par construction, un nombre défini d'erreurs. La deuxième section concerne l'utilisation d'un code RSC avec rendement unitaire permettant d'augmenter la distance minimale du code. Enfin, la dernière section s'intéresse à l'adjonction d'un code CRC. Ce dernier est alors majoritairement employé pour sélectionner un mot de code parmi plusieurs candidats.

1.2.6.2.1 Code BCH Dès l'apparition des turbo codes, il a été remarqué que les motifs d'erreurs des turbo codes en régime asymptotique sont constitués d'un faible nombre de bits erronés. Autrement dit, si pour une valeur de SNR élevée, le turbo décodeur ne converge pas vers la bonne séquence d'information, alors la séquence décodée est très proche du mot de code émis. Partant de ce constant, Andersen propose en 1996 de concaténer un code BCH avec un turbo code [68]. La Figure 1.20 présente cette concaténation série au niveau du codeur. Après un décodage du turbo code non fructueux, un décodage dur à partir du code BCH est effectué permettant d'éliminer les erreurs résiduelles. Cette approche résulte en un abaissement du plancher d'erreurs. Ceci s'explique par une augmentation de la distance minimale du code via la concaténation série. En revanche, la convergence est retardée en raison de la baisse du rendement global provenant de l'ajout de redondance supplémentaire. Par construction, le nombre de bits de redondance nécessaires (R_{BCH}) pour qu'un code BCH corrige t erreurs a pour expression :

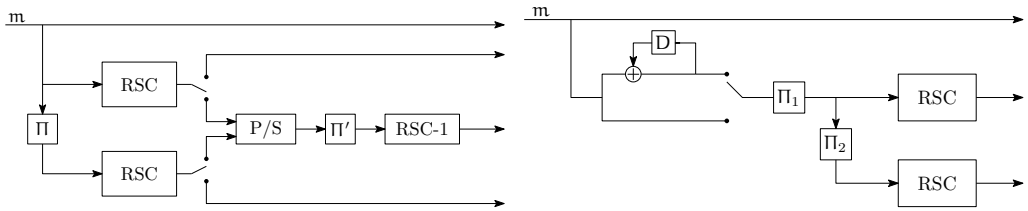
$$R_{BCH} = t \times \lceil \log_2 K \rceil.$$

Ainsi, plus K est grand, moins l'impact sur le rendement est important. De plus, la complexité calculatoire d'un décodeur dur BCH est raisonnable. À titre d'exemple, un schéma de codage équivalent a été choisi dans le cadre du standard DVB-S2 où un code LDPC est concaténé avec un code BCH.

Afin de réduire la perte de rendement, une analyse empirique est réalisée dans [69]. Cette dernière permet d'identifier les positions les plus souvent erronées dans la séquence d'information. Dès lors, seules ces positions sont protégées par un code BCH, impliquant un surcoût moins important en terme de bits de redondance supplémentaires.

Il est à noter que le code BCH peut aussi servir de code détecteur d'erreurs et donc de critère d'arrêt lors d'un processus de décodage itératif. Ceci permet d'augmenter le nombre maximal d'itérations du turbo décodeur car le nombre d'itérations moyen n'est que peu impacté, comme décrit dans [70]. Ainsi, la convergence est améliorée.

1.2.6.2.2 Code RSC de rendement 1 Le principe d'une concaténation basée sur un tel code est d'encoder partiellement soit l'information redondante provenant des



(a) Turbo code 3D (b) Turbo code pré-codé

FIGURE 1.21 – Concaténation d'un turbo code et d'un RSC de rendement 1.

codeurs élémentaires soit l'information systématique avant de la fournir aux codeurs élémentaires.

Le première forme de concaténation est nommée turbo codes 3D. Ces derniers ont été présentés par Berrou *et al.* en 2007 [71], puis analysés successivement dans [72] et [73]. La deuxième forme est nommée quant à elle turbo code pré-codé. Leur proposition remonte à 2011 [74]. Leur structure a été modifiée puis optimisée dans [75]. Les codeurs associés sont présentés en Figure 1.21.

Que ce soit pour les turbo codes 3D ou les turbo codes pré-codés, l'utilisation d'un code RSC de rendement 1 permet de ne pas modifier le rendement global du schéma de codage. Cependant, plus la part d'information traitée par ce codeur additionnel est importante, plus la convergence est retardée et plus la distance minimale est augmentée. Ainsi, un compromis peut être trouvé suivant les performances visées.

Dans les deux cas, trois décodeurs SISO doivent échanger mutuellement de l'information. La complexité calculatoire du décodage augmente de façon notable. Dans [72], une hausse de la complexité calculatoire du décodeur d'environ 10% est estimée. De plus, la latence de décodage augmente puisqu'une itération de décodage correspond dorénavant à l'activation successive des 3 décodeurs SISO.

1.2.6.2.3 Code CRC La troisième concaténation utilise des codes détecteurs d'erreurs CRC. La concaténation série d'un tel code avec un turbo code est présenté en Figure 1.22. Cette famille de code possède un pouvoir de détection important. Ainsi, ce type de détecteur est souvent défini dans les standards employant des turbo codes. Par exemple, dans le standard LTE, ils permettent l'utilisation de demandes de renvoi automatique (ARQ). Dans ce cas, lorsque le récepteur détecte une erreur de transmission, après turbo décodage via l'utilisation du code CRC, il peut demander une retransmission des données. L'ARQ permet une amélioration de la qualité de service. D'autre part, les codes CRC peuvent aussi permettre d'arrêter le processus de décodage itératif dès lors que le turbo décodeur a convergé vers le bon mot de code. Des analyses comme dans [76] ont été menées afin de déterminer les probabilités d'erreurs non détectées et de fausse alerte lorsque qu'une détection CRC est effectuée au cours du processus itératif.

Dans un but d'améliorer les performances de décodage, les codes CRC ont été utilisés de

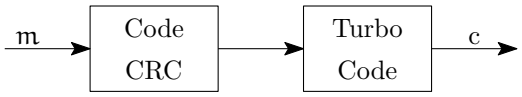


FIGURE 1.22 – Concaténation série d'un code CRC et d'un turbo code.

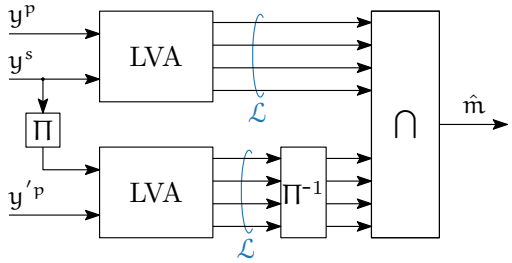
diverses manières. Celles-ci sont maintenant présentées.

Décodage par liste Cette famille de décodage a été pensée des la fin des années 50 indépendamment par Elias et Wozencraft [77, 78]. Dans le cadre des codes convolutifs, son principe est de fournir les $\mathcal{L}$ séquences les plus probables correspondant à des chemins dans le treillis. Ces $\mathcal{L}$ séquences peuvent être obtenues grâce à l'algorithme de Viterbi par liste (LVA) selon une approche parallèle ou une approche séquentielle [79]. Plus récemment, une dérivation de l'algorithme ML-APP permettant d'obtenir $\mathcal{L}$ séquences a été proposée [80]. Dans le cadre du décodage de turbo codes par liste, trois principales approches peuvent être appliquées :

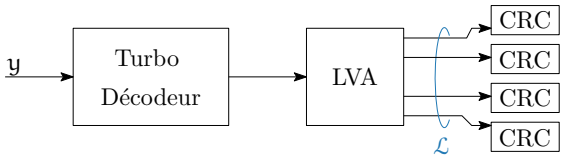
1. **Intersection de liste :** Chacun des codes constituants est décodé selon un algorithme par liste. Dès que l'intersection des deux ensembles de liste consiste en une seule séquence, le décodage s'arrête. Cette séquence est alors la sortie du décodeur [81].
2. **LVA post turbo :** Le LVA peut aussi être réalisé en dehors du processus itératif usuel. Dans ce cas, il est exécuté après chaque itération ou à la fin du processus [82]. Les informations *a posteriori* du turbo décodeur servent alors d'informations *a priori* pour le LVA. Afin de déterminer la bonne séquence parmi les $\mathcal{L}$ séquences obtenues, un code détecteur d'erreurs est utilisé. Grâce à cette technique des gains notables, de l'ordre d'un facteur dix avec une liste de taille trois dans le plancher d'erreurs, sont obtenus.
3. **LVA pré turbo :** Le décodage liste peut aussi être appliqué avant le processus itératif. Il doit alors fournir $\mathcal{L}$ séquences distinctes de sorties souples qui seront utilisées en tant qu'information *a priori* par $\mathcal{L}$ turbo décodeurs distincts. A nouveau, l'identification de la bonne séquence est réalisée par un code détecteur d'erreurs. Cette méthode propose des gains aussi bien dans la zone de convergence que dans le plancher d'erreurs, mais possède un surcoût calculatoire important [83]. Ces gains représentent un facteur cinq pour de petites tailles de trame avec une liste de taille 16.

Les décodeurs associés à ces différentes utilisations du LVA sont récapitulés en Figure 1.23.

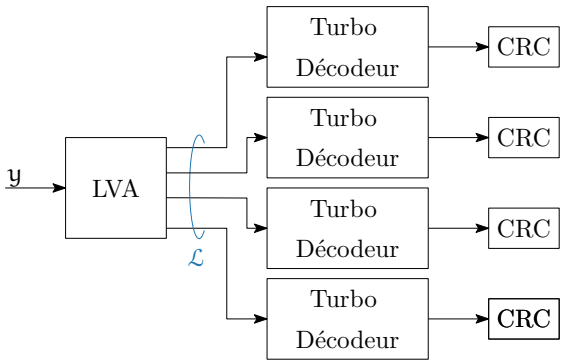
Décodage utilisant le décodage par statistiques ordonnées (OSD) Des approches utilisant un tel décodage ont été proposées pour les turbo codes [84]. Le principe réside dans le fait d'appliquer l'OSD [85] de rang 1 après chaque itération de turbo décodage ou après le processus itératif. Une présentation schématique de ce principe est donné en Figure 1.24. Le décodage OSD se base sur la matrice génératrice du code. Celle-ci



(a) Intersection de listes



(b) LVA post turbo



(c) LVA pré turbo

FIGURE 1.23 – Décodeur utilisant l'algorithme de Viterbi par liste.

est permutée selon la fiabilité des informations. Grâce à une élimination Gaussienne et en inversant les décisions dures du décodeur turbo, K mots sont identifiés. Finalement, le bon mot est sélectionné à l'aide d'un code détecteur d'erreurs. D'importants gains sont obtenus dans la zone de convergence pour de petites tailles de trame et de hauts rendements. Cette méthode a été étendue en considérant la matrice globale de la concaténation série du code CRC et du turbo code dans [86]. Dans ce cas, le code CRC perd son pouvoir de détection mais les performances de décodage sont améliorées car la distance minimale du code concaténé est exploitée. Cependant, la complexité calculatoire de cette méthode est telle que seuls des décodages de trame de petite taille peuvent être envisagés.

Multiples turbo décodages Cette méthode a été présentée par Crozier et Ould-Cheikh-Mouhamedou dans [87] puis reformulée dans [88]. Son principe est le suivant. Si, à la fin du processus itératif, le décodeur n'a pas convergé vers le bon mot de code, les valeurs absolues des informations *a posteriori* sont triées par ordre croissant. La valeur de

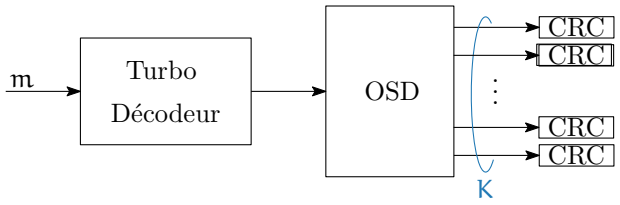


FIGURE 1.24 – Décodage associant un turbo code et un décodeur OSD.

l'information systématique correspondant à la position la moins fiable est alors forcée à une valeur supérieure à celle de la distance minimale du code. Un nouveau turbo décodage itératif est alors effectué. Le résultat est vérifié grâce à un code détecteur d'erreurs. Si le mot ne vérifie toujours pas le code détecteur d'erreurs, alors, la seconde position la moins fiable est forcée à une valeur supérieure à la distance minimale du code, tout en ayant remis la position la moins fiable à sa valeur originelle. Un nouveau turbo décodage itératif est alors effectué. Ce processus est répété jusqu'à ce qu'un nombre prédéfini de positions testées soit atteint ou jusqu'à ce que le code détecteur d'erreurs valide le mot courant.

Dans le cadre des turbo codes double binaires, les quatre possibilités de valeurs pour le couple (A_i,B_i) sont considérées. Les performances de décodage sont améliorées à la fois dans la zone de convergence et au niveau du plancher d'erreurs mais ce au prix, d'une multiplication par $4 \times L$ de la complexité calculatoire avec L le nombre de positions examinées.

Cette technique a été étendue par Pfletschinger dans [89]. Cette fois toutes les informations provenant du canal sont considérées pour être fixées à une valeur supérieure à celle de la distance minimale du code. Ainsi, N turbo décodages successifs peuvent être nécessaires. Afin d'obtenir les meilleures performances de décodage, un algorithme LVA est conjointement employé. Cependant, bien que cette méthode présente les meilleures performances de décodage, elle ne peut être considérée dans un contexte temps réel du fait de sa latence d'exécution et de la complexité calculatoire résultante.

1.2.6.3 Récapitulatif des différentes méthodes d'abaissement du plancher d'erreurs

Dans cette section, différentes méthodes d'abaissement du plancher d'erreurs sont récapitulées et comparées en terme de gains de décodage et de complexité de décodage. Une synthèse de cette comparaison est fournie dans le tableau 1.4.

Bien évidemment, toutes ces méthodes permettent d'abaisser le plancher d'erreurs. De plus, les techniques utilisant l'OSD ou le turbo décodage multiple proposent des gains aussi dans la zone de convergence. Cependant, leurs complexités calculatoires les excluent d'applications sous forte contrainte d'exécution.

La concaténation avec un code BCH propose des gains intéressants dans la zone du plancher d'erreurs tout en introduisant une complexité calculatoire modérée. Néanmoins,

TABLEAU 1.4 – Synthèse des différentes méthodes améliorant les performances de décodage.

	BCH	RSC-1	List Decoder	OSD	Décodages multiples
Redondance ajoutée	Oui	Non	CRC	CRC	CRC
Localisation des gains	Plancher	Plancher	Plancher	Convergence et plancher	Convergence et plancher
Complexité calculatoire	👍	👍	👍	👎	👎

il est nécessaire d'ajouter des informations redondante (provenant du code BCH). Ceci implique une modification du rendement du code et donc un décalage sur le seuil de convergence. En revanche, l'utilisation d'un code RSC de rendement 1 ne possède pas cet inconvénient mais nécessite cependant une modification du schéma de codage. Cette approche ne peut donc être utilisée dans des contextes déjà standardisés. De plus, le troisième SISO impacte négativement la complexité calculatoire et la latence du décodeur.

Finalement, les approches basées sur un décodage LVA successif au turbo décodage possèdent l'avantage de nécessiter uniquement l'adjonction d'un code CRC au turbo code. Des gains de performance sont observés dans le plancher d'erreurs mais la complexité calculatoire ajoutée n'est pas négligeable.

1.3 Conclusion

Dans ce premier chapitre, les notions permettant d'aborder le codage de canal ont été successivement introduites. Puis, la construction, le décodage et les performances des turbo codes ont été détaillés. Des outils permettant d'estimer les performances d'un turbo code sur le canal BI-AWGN ont été présentées. Finalement, plusieurs méthodes permettant d'améliorer les performances de décodage des turbo codes ont été rapportées. Dans le cadre de turbo codes déjà standardisés, seules des méthodes appliquées au processus de décodage peuvent être employées. Or, de part la sous-optimalité du turbo décodage, des améliorations peuvent encore exister. Il est aussi à noter que de nombreux standards de communications numériques incluent un code CRC concaténé avec le turbo code. Ceci permet d'envisager une plus grande diversité de décodage. Pour l'ensemble de ces raisons, dans le chapitre suivant, afin d'améliorer les performances de décodage, des techniques de décodage tirant parti des oscillations au sein du turbo décodeur sont détaillées.

2 Des oscillations du décodage itératif

Ce deuxième chapitre étudie et vise à exploiter les oscillations de métriques impliquées dans le décodage itératif de turbo codes.

Dans un premier temps, les oscillations sont définies et observées statistiquement. De là, des premières conclusions sont énoncées quant à l'exploitation de ces oscillations dans un but d'améliorer les performances de décodage de turbo codes.

Ensuite, une adaptation pour les turbo codes d'un algorithme originellement proposé pour le décodage des codes LDPC est établie. Ses performances selon différents contextes sont déterminés et discutés.

Finalement, une tentative d'utilisation de l'observation des oscillations est considérée en dernière section dans un contexte de décodage répété d'une même trame.

2.1 Introduction

En 1996, Benedetto et Montorsi listent des questions ouvertes liées aux turbo codes [90]. La première, qu'ils qualifient de « question théorique importante » concerne la convergence des algorithmes itératifs sous-optimaux. « Convergent-ils toujours ? Sous quelles conditions ? »

Peu après, il a été montré par des arguments géométriques que la convergence de ces algorithmes vers la solution à maximum de vraisemblance ou vers une solution stable n'est pas garantie [91]. Dans [92], l'évolution de la valeur des LLR associés aux informations *a posteriori* est présentée au cours du processus itératif. Les auteurs classifient alors le comportement des LLR lors du décodage de trames selon 3 modes :

1. Tous les bits convergents rapidement et de la même manière vers une solution stable.
2. La plupart des bits convergent rapidement vers une solution stable alors que les autres convergent de manière différente (croissance plus lente).
3. Les valeurs des LLR oscillent, croisant ou non le seuil de décision (0). Dans ce cas, l'évolution des valeurs des LLR est d'allure sinusoïdale.

De ces constatations, les auteurs proposent un critère d'arrêt pour le processus itératif basé sur la valeur moyenne des LLR. Une comparaison est faite par rapport à un seuil prédéterminé. Cependant, la valeur de ce seuil ne peut être obtenue que de manière empirique.

D'autres critères d'arrêt basés sur l'analyse des changements de signe des informations produites par les décodeurs élémentaires avaient déjà été proposés dans la littérature. Le Sign Change Ratio (SCR) [93] est une approximation du critère basée sur la Cross Entropy (CE) [94]. Le principe consiste à compter le nombre de changement de signes $\mathsf{C}(\mathtt{i})$ des informations extrinsèques produites par le second SISO entre l'itération $\mathtt{i}$ et $\mathtt{i}-1$. Des simulations montrent qu'arrêter les itérations lorsque $\mathsf{C}(\mathtt{i}) \leqslant (0{,}0005 \sim 0{,}03) \times \mathsf{K}$ permet d'obtenir des performances similaires en termes de nombre d'itérations moyen et de décodage que le critère CE.

Le critère d'arrêt nommé Sign Difference Ratio (SDR) [93] est une variante du SCR. Dans ce cas, $\mathsf{C}(\mathtt{i})$ compte le nombre de fois où l'information *a posteriori* et l'information extrinsèque diffèrent à l'itération $\mathtt{i}$. Ce critère aboutit aux mêmes performances que le SCR tout en permettant de réduire la quantité d'information à stocker.

Ainsi, dans le cadre des turbo codes, les oscillations au sein des décodeurs ont été étudiées pour mieux comprendre le fonctionnement du processus itératif mais aussi pour fournir des critères d'arrêt performants. Dans ce contexte, performant signifie permettant de réduire significativement le nombre moyen d'itérations tout en conservant les performances de décodage obtenues avec un nombre d'itération fixe.

Pour la famille des codes LDPC, le processus de décodage est également itératif. Des phénomènes oscillatoires sont donc aussi observables. Néanmoins pour les codes LDPC, les équations de parités permettent la détection d'erreurs. Ainsi, les oscillations ont été étudiées dans un but d'améliorer les performances de décodage. Deux approches majeures

ont été considérées. La première approche consiste en une modification de l'algorithme à propagation de croyance (BP) pour le décodage des codes LDPC [95]. Son principe est le suivant. Si le nouveau LLR extrinsèque (qui correspond au calcul des nœuds de variable) change de signe lors de la nouvelle itération, alors la valeur calculée lors de l'itération précédente est sommée à la valeur courante. Les auteurs constatent une amélioration des performances de décodage par comparaison avec l'algorithme BP usuel.

La seconde approche, nommée Self-Corrected (SC) [96], est une modification de l'algorithme Min-Sum. Ce dernier étant déjà une simplification de l'algorithme BP. Son principe est similaire à l'approche précédente. Si le nouveau LLR extrinsèque change de signe lors de la nouvelle itération, alors ce LLR est mis à zéro avant d'être transmis aux nœuds de parité. Des gains de performances sont observés dans la zone de convergence permettant au SC de gagner 0,4 dB sur le Min-Sum. Il n'est plus qu'à 0,1 dB du BP, bien plus complexe.

Avant de présenter une adaptation de ces approches aux turbo codes, une étude statistique concernant les oscillations de l'information extrinsèque au cours du décodage itératif des turbo codes est maintenant menée. Le but de cette étude est d'observer le comportement d'un turbo décodeur lors de son fonctionnement afin de pouvoir extraire des informations supplémentaires permettant peut-être d'améliorer ses performances de décodage.

2.2 Observations statistiques des oscillations dans le processus turbo

2.2.1 Cadre de l'étude

Les paramètres du couple codeur/décodeur pour cette étude sont les suivants :
- Turbo code binaire,
- Standards LTE (8 états) et CCSDS (16 états),
- Algorithme EML-MAP itérant 32 fois.

Ainsi un instantané du comportement d'un turbo décodeur usuel dans un contexte binaire peut être capturé. Ceci représente un cas d'usage suffisamment large, permettant de tirer des conclusions sur les turbo décodeurs binaires standardisé. Le choix de 32 itérations au maximum permet d'obtenir suffisamment d'amplitude quant aux oscillations afin de les visualiser aisément.

Dans la suite, une oscillation est définie comme un changement de signe de l'information extrinsèque associée à un bit d'information. Dans cette étude, quatre types d'oscillations peuvent être considérés. La Figure 2.1 illustre ces 4 types d'oscillations pour l'information extrinsèque sur un schéma de turbo décodeur dans lequel les étapes d'entrelacement n'apparaissent pas pour plus de simplicité.

Le schéma de la Figure 2.1 présente une symétrie centrale. Ceci nous permet de simplifier sa présentation. Considérons donc uniquement le décodeur élémentaire opérant dans le domaine naturel. Ce dernier fournit lors de l'itération j un avis sur le signe de chacun

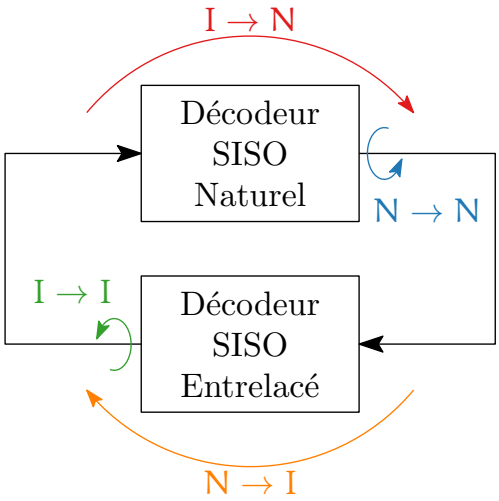


FIGURE 2.1 – Les différents types d'oscillations possibles pour l'information extrinsèque.

des bits de la trame, exprimé par l'information extrinsèque $\mathbf{L_{12}^{e}}^{(j)}$. À l'itération $j+1$, cet avis ($\mathbf{L_{12}^{e}}^{(j+1)}$) peut changer. Cette oscillation est notée N → N, car elle est le résultat de deux itérations de calcul dans le domaine naturel. Après entrelacement, cette information extrinsèque est utilisée par le décodeur du domaine entrelacé pour produire une nouvelle information extrinsèque, $\mathbf{L_{21}^{e}}^{(j+1)}$. Une nouvelle oscillation peut donc subvenir. Elle est notée N → I car la référence est le domaine naturel et est obtenue dans le domaine entrelacé. Cette dernière oscillation peut être interprétée comme un désaccord entre les deux décodeurs élémentaires. Les oscillations I → I et I → N s'obtiennent naturellement par symétrie.

Des statistiques quant à l'occurrence de ces quatre types d'oscillations au cours du processus itératif vont maintenant être présentées. Celles-ci sont obtenues, dans un premier temps, dans le contexte du turbo code du standard LTE. Plusieurs valeurs de SNR sont considérées afin d'analyser les oscillations suivant le contexte de fonctionnement du décodeur itératif (zones de non-convergence, de convergence et de plancher d'erreurs). Pour toutes ces statistiques, les trames considérées sont au nombre de 100. Tout d'abord l'étude sera statique : les moyennes et les distributions des oscillations par bits seront présentées. Dans un second temps, l'étude sera dynamique en considérant l'évolution des oscillations au cours du processus itératif.

2.2.2 Turbo codes du standard LTE

2.2.2.1 Nombre moyen d'oscillations par bit

La première statistique qui peut être obtenue aisément est le nombre moyen d'oscillations par bits. Les bits peuvent être répartis selon deux groupes principaux :

— les bits appartenant à une trame correcte à la dernière itération,
— les bits appartenant à une trame erronée à la dernière itération.

Ensuite, le second groupe peut lui-même être subdivisé en deux sous-groupes :

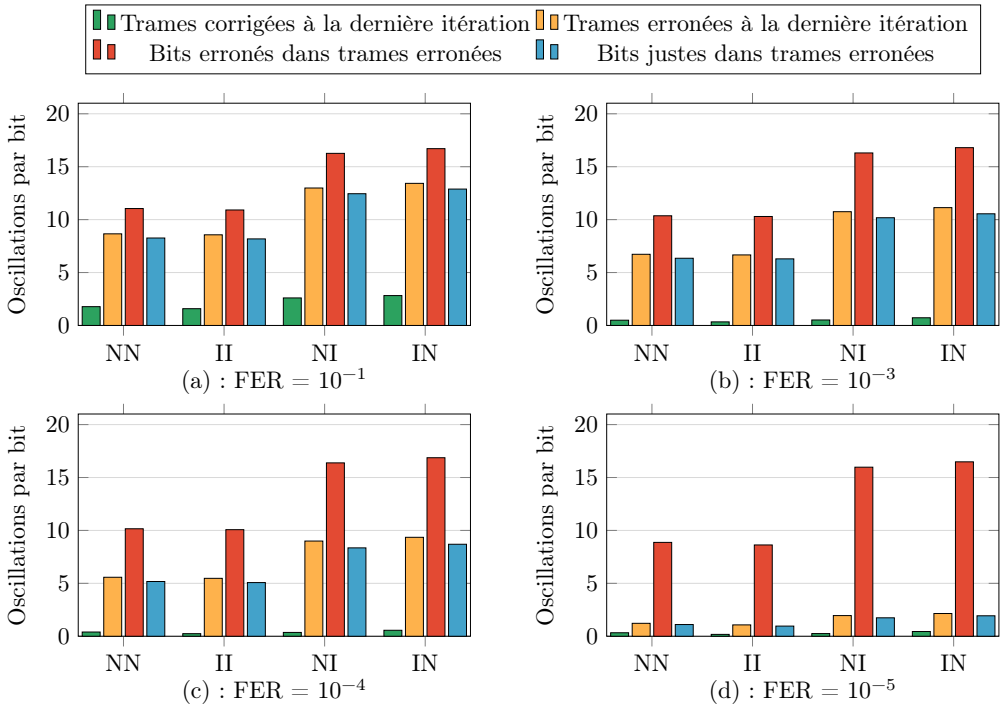


FIGURE 2.2 – Nombre moyen d'oscillations pour différents taux d'erreurs trame cibles, turbo code du standard LTE (K=1024, R=1/3).

— les bits corrects dans une trame erronée à la dernière itération,
— les bits erronés dans une trame erronée à la dernière itération.

La Figure 2.2 présente l'évolution du nombre d'oscillations par bit selon la classification qui vient d'être introduite et ce, en fonction de la diminution du taux d'erreur trame pour le standard LTE. Le turbo code considéré a une taille de trame de 1024 bits d'information pour un rendement 1/3. La sous-figure (a) correspond à la zone de non-convergence, les sous-figures (b) et (c) à la zone de convergence et finalement la (d) à la zone du plancher d'erreurs. Plusieurs constations notables peuvent en être extraites.

Tout d'abord, le nombre moyen d'oscillations par bit est le même que nous considérions comme référence, le décodeur opérant dans le domaine naturel, ou celui opérant dans le domaine entrelacé. Ainsi, dans la suite des analyses, uniquement les oscillations $\mathsf{N} \rightarrow \mathsf{N}$ et $\mathsf{N} \rightarrow \mathsf{I}$ seront détaillées. De plus, quelque soit la valeur du SNR, les informations extrinsèques d'une trame corrigée n'oscillent pas ou très peu. En comparaison, les informations extrinsèques d'une trame erronée oscillent quant à elles environ dix fois plus. Cette constatation est le principe fondamental sur lequel reposent les critères d'arrêts SCR et SDR. En effet, le nombre d'oscillations permet d'estimer si une trame a convergé ou non.

Toujours à partir de la Figure 2.2, en considérant uniquement les trames erronées, il est notable que le nombre moyen d'oscillations par bit (tous bits confondus) est très proche

du nombre moyen d'oscillations par bit corrigé. Ceci provient du fait que le nombre de bits erronés par trame erronée (BE/FE) est relativement faible. En effet, ce dernier est largement inférieur à 100 dans la zone de convergence. Il est inférieur à 10 dans la zone du plancher d'erreurs. Cela signifie que dans une trame erronée, la majorité des bits d'information est corrigée à la fin du processus de décodage.

Il est encore plus remarquable que les LLR des bits erronés oscillent plus que ceux des bits corrigés. Plus la valeur de SNR augmente, plus cette constatation est vérifiée. Nous pouvons constater sur la Figure 2.2 que les LLR des bits erronés oscillent environ dix fois pour l'oscillation $\mathsf{N} \rightarrow \mathsf{N}$ et environ seize fois pour l'oscillation $\mathsf{N} \rightarrow \mathsf{I}$, ce quelque soit la valeur du SNR. En revanche, le nombre moyen d'oscillations des LLR des bits corrigés diminue graduellement lorsque la valeur de SNR augmente. Ainsi, au sein des trames erronées, au niveau du plancher d'erreurs du turbo code du standard LTE, les LLR des bits erronés oscillent quatre fois plus que ceux des bits corrigés.

Pour conclure les observations résultant de la Figure 2.2, pour une raison de symétrie dans l'architecture du décodeur, le nombre d'oscillations par bit est lui aussi symétrique vis-à-vis de l'entrelaceur. Ainsi, seuls deux types d'oscillations peuvent être considérées dans l'étude. Le nombre d'oscillations par bit est un critère discriminant les trames corrigées. Ceci a permis la définition des critères d'arrêts SDR et SCR. Plus encore, si un bit est erroné, alors, en moyenne, il aura plus oscillé que les bits corrigés. Cependant, pour pouvoir améliorer le processus de décodage, il est intéressant d'essayer d'établir si la réciproque est vraie : « Si un bit a oscillé plus que la moyenne, alors il est erroné ». Pour répondre à cette question, les statistiques doivent être présentées selon une granularité différente : celles-ci se doivent d'être présentées au niveau trame.

2.2.2.2 Comportement des oscillations au niveau trame au cours du processus itératif

Dans cette partie, les oscillations sont présentées au cours du processus itératif. Le turbo code du standard LTE avec comme paramètres K=1024 et R=1/3 est toujours employé. Seules deux valeurs de SNR sont considérées afin de faciliter l'analyse. La première correspond à un taux d'erreur trame de 10^{-2}, donc au début de la zone de convergence. La valeur de SNR associée vaut 0,65 dB. La seconde correspond à un taux d'erreur trame de 10^{-5}, situant les performances de décodage dans la zone du plancher d'erreurs. La valeur de SNR associée vaut 1,2 dB.

Les résultats pour un taux d'erreur trame de 10^{-2} sont présentés dans la Figure 2.3. Cette Figure est divisée en six sous-figures. La première colonne traite des trames erronées à l'issue du processus itératif. La seconde colonne concerne les trames corrigées. Deux types d'oscillations sont présentées : le nombre d'oscillations $\mathsf{N} \rightarrow \mathsf{N}$, le nombre d'oscillations $\mathsf{N} \rightarrow \mathsf{I}$. Conjointement à ceci, le nombre de bits d'information erronés est aussi présenté. Pour chacune de ces sous figures, chaque courbe représente l'évolution du nombre d'oscillations pour une trame donnée. Cent trames erronées et cent trames corrigées y sont présentées.

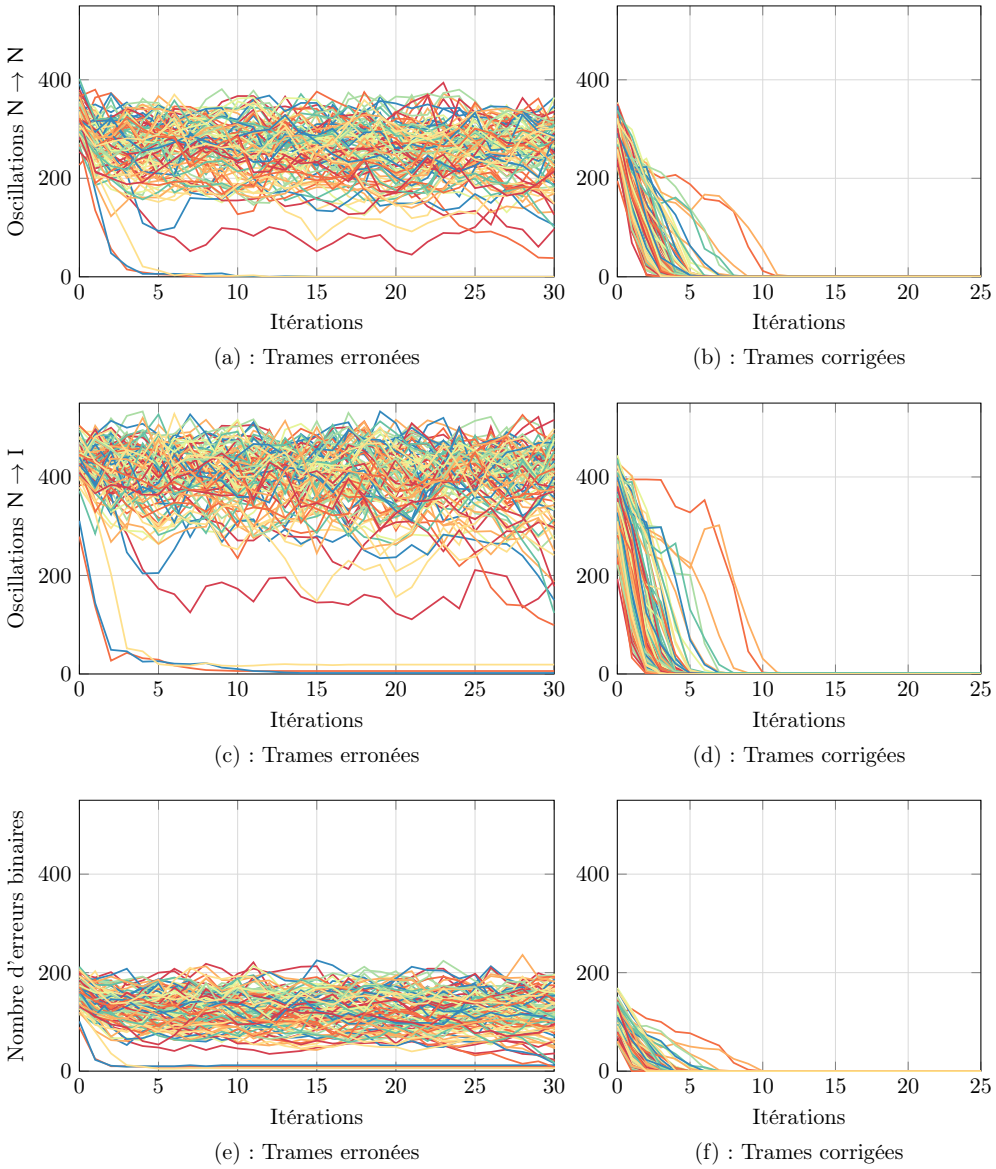


FIGURE 2.3 – Oscillations au cours des itérations dans le cadre du standard LTE (K=1024, R=1/3) pour un taux d'erreur trame de 10^{-2}. Représentations de 100 trames erronées et 100 trames corrigées.

Nous pouvons remarquer que les trames corrigées à la $32^{\text{ème}}$ itération alors qu'elles ne l'étaient pas à l'issue de la $10^{\text{ème}}$ itération sont marginales. Ceci est à corroborer avec le choix réalisé dans la majorité des architectures de turbo décodeur de fixer le nombre maximal d'itérations à moins de 10. Il est aussi remarquable que le nombre d'erreurs semble être fortement corrélé avec le nombre d'oscillations. En effet, les rares trames erronées n'oscillant que peu sont celles qui ne possèdent que peu de bits erronés à la fin du processus de décodage. Plus généralement, la valeur du nombre d'oscillations semble correspondre au double du nombre d'erreurs.

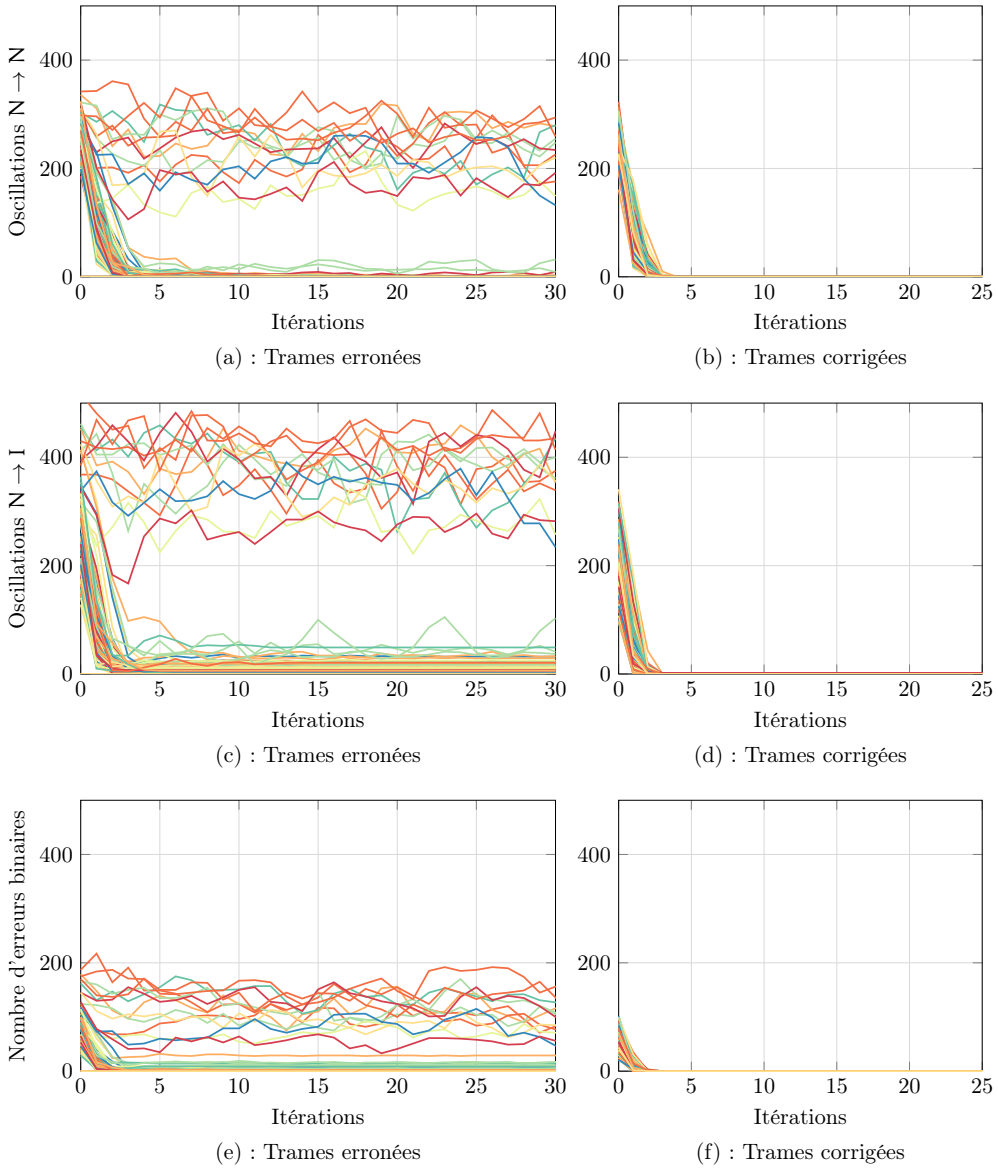


FIGURE 2.4 – Oscillations au cours du processus itératif dans le cadre du standard LTE (K=1024, R=1/3) pour un taux d'erreur trame de 10^{-5}. Représentations de 100 trames erronées et 100 trames corrigées.

La Figure 2.4 fournit les mêmes informations que la Figure 2.3 mais pour un taux d'erreur trame cible de 10^{-5}. Pour le standard LTE, ceci correspond à un fonctionnement du turbo décodeur au début du plancher d'erreurs. Nous pouvons remarquer que pour les trames corrigées, la convergence est plus rapide : seulement quelques itérations sont nécessaires. Pour les trames erronées à l'issue du processus itératif, des trames possédant un nombre important d'erreurs sont rares. Moins de 20% des trames ont plus de 20 bits erronés. Elles correspondent à celles présentant le plus grand nombre d'oscillations. De fait, la majorité des trames erronées possèdent un faible nombre de bits d'information erronés.

L'état interne de ces trames sont majoritairement constant après la cinquième itération.

Finalement, ces nouvelles observations permettent de mieux analyser les oscillations durant le processus itératif. Il est alors notable que d'une part, pour les trames erronées, deux « modes » d'oscillations sont visibles :

— le nombre d'oscillations est très important. Des variations sont observées au cours des itérations mais elles restent dans la plage de valeurs [200,400].
— le nombre d'oscillations est faible et varie peu.

D'autre part, une corrélation forte entre le nombre d'erreurs et le nombre d'oscillations a été constaté. Cependant, cette analyse a été menée au niveau trame. Il est donc maintenant nécessaire de descendre de niveau de granularité pour observer la distribution des oscillations au sein des trames erronées.

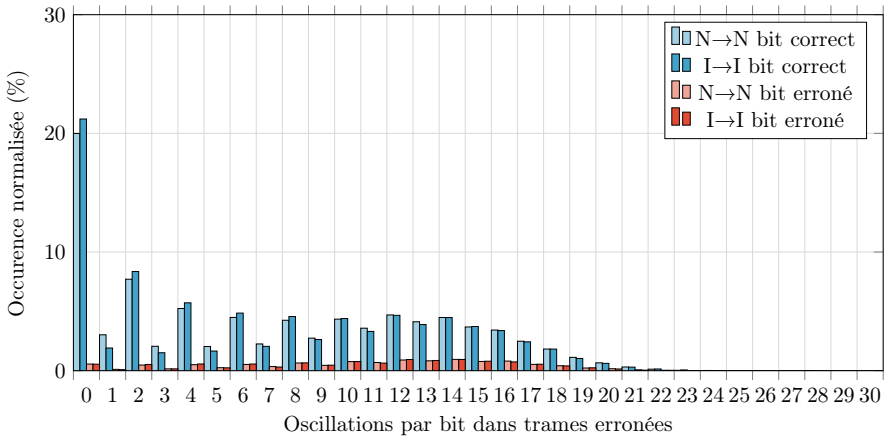
2.2.2.3 Distribution des oscillations au sein des trames erronées

Dans cette section, la distribution des oscillations dans les trames erronées est analysée. Dans les sections précédentes, il a été mis en évidence que les bits erronés d'une trame erronée oscillent plus que les bits corrects. La présente analyse permet d'établir si la réciproque est vraie. Pour ce faire, la Figure 2.5 présente l'occurrence du nombre d'oscillations par bit pour un taux d'erreur trame cible de 10^{-2}. Cette Figure, composée de deux histogrammes, considère d'abord les oscillations $\mathsf{N} \rightarrow \mathsf{N}$ et $\mathrm{I} \rightarrow \mathrm{I}$ (a) puis, les oscillations $\mathsf{N} \rightarrow \mathrm{I}$ et $\mathrm{I} \rightarrow \mathsf{N}$ (b). Les occurrences sont normalisées par rapport au nombre de bits d'information total par trame.

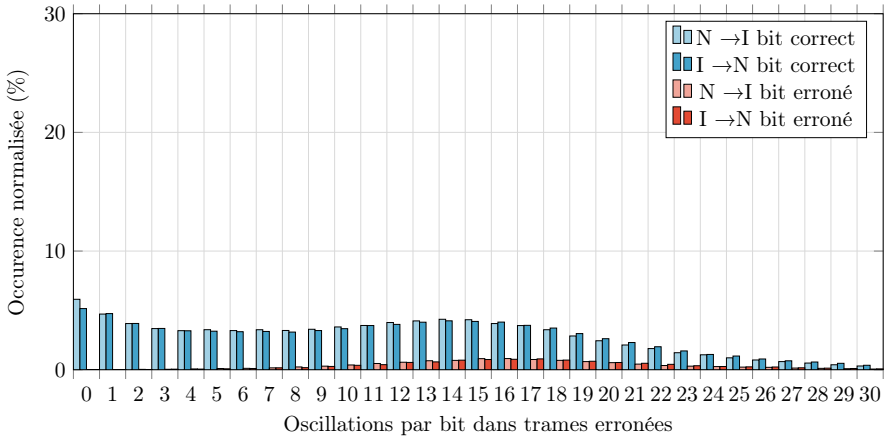
Dans Figure 2.5 (a), 20% des bits sont corrects et n'ont pas oscillé. Il est à noter que les nombres d'oscillations impairs sont sous représentés vis-à-vis des nombres d'oscillations pairs. Il semblerait que, majoritairement, la première décision prise par un décodeur SISO soit la bonne. Néanmoins, nous n'avons pu tirer de conclusion plus pertinente quant cette observation. La distribution des oscillations des bits erronés est, quand à elle, relativement équirépartie.

Les oscillations $\mathsf{N} \rightarrow \mathrm{I}$ et $\mathrm{I} \rightarrow \mathsf{N}$ ne présentent pas d'irrégularité comme constatée dans les cas $\mathsf{N} \rightarrow \mathsf{N}$ et $\mathrm{I} \rightarrow \mathrm{I}$. L'allure de la distribution des oscillations pour les bits corrects à l'issu du processus itératif semble être l'association de deux gaussiennes : l'une centrée autour de 0 oscillation et l'autre autour de 14 oscillations. Pour les bits erronés, l'allure des distributions est également de forme gaussienne et centrée autour de 16 oscillations. Ces allures sont à mettre en corrélation avec le bruit additif lui-même gaussien utilisé sur le canal.

Pour le taux d'erreur trame considéré dans cette analyse, le nombre moyen de bits erronés par trame erronée (BE/FE) vaut 113. En ramenant ceci à la taille de la trame, en moyenne, 11% des bits sont erronés dans une trame erronée. Si le nombre d'oscillations est considéré, la probabilité de bit erroné varie de 0,1% à 19,9%. Il apparaît alors que les bits ayant oscillé moins de 11 fois ont une probabilité d'être erronés inférieure à la probabilité moyenne. Plus encore, en considérant les bits ayant oscillé 15 fois ou plus, la probabilité qu'ils soient erronés vaut 18%. La probabilité d'obtenir un bit erroné a alors quasiment



(a) : Oscillations



(b) : Désaccords

FIGURE 2.5 – Distribution du nombre d'oscillations par bit pour un taux d'erreur trame de 10^{-2}, pour le turbo code du standard LTE (K=1024, R=1/3).

doublé. Ces observations sont formalisées dans les équations suivantes :

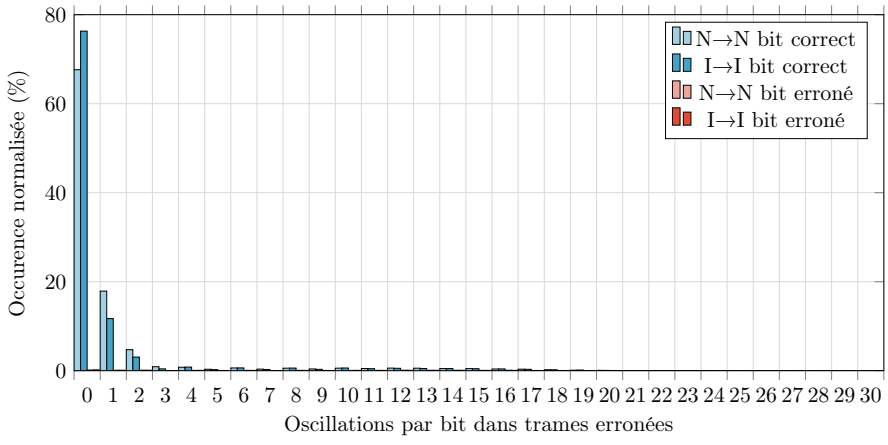
$$\mathsf{P}_{\mathtt{i}\in\mathsf{K}}(\text{bit } \mathtt{i} \text{ est en erreur}) = 11\%$$

$$\mathsf{P}_{\mathtt{i}\in\mathsf{K}}(\text{bit } \mathtt{i} \text{ est en erreur} \mid \mathrm{OSC}(\mathtt{i}) > 15) = 18\%.$$

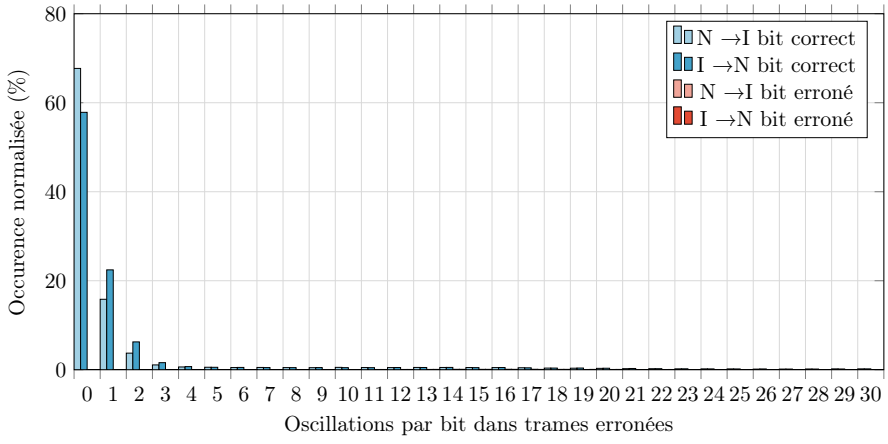
Néanmoins, les bits correctement décodés oscillent eux aussi. De plus, comme ils sont majoritaires, il n'existe pas de nombre d'oscillations permettant d'obtenir une probabilité d'obtenir un bit erroné supérieure à celle d'obtenir un bit correct. C'est-à-dire qu'il n'existe pas $\mathtt{n}$ tel que :

$$\mathsf{P}_{\mathtt{i}\in\mathsf{K}}(\text{bit } \mathtt{i} \text{ est en erreur} \mid \mathrm{OSC}(\mathtt{i}) > \mathtt{n}) > 50\%.$$

La distribution des oscillations est maintenant présentée en Figure 2.6 pour un taux d'erreur trame cible de 10^{-5}. De part la rapidité de convergence, une seule gaussienne est maintenant visible : celle centrée autour de 0 oscillation. La seconde gaussienne disparaît



(a) : Oscillations



(b) : Désaccords

FIGURE 2.6 – Distribution du nombre d'oscillations par bit pour un taux d'erreur trame de 10^{-5}, pour le standard LTE (K=1024, R=1/3).

progressivement au fur et à mesure que la valeur de SNR augmente, jusqu'à ce qu'elle ne soit plus visible lorsque la valeur de SNR correspond à une position dans le plancher d'erreurs.

Dans le cas considéré, ramené en pourcentage, le taux de bits erronés par trame erronée vaut 1,5%. En l'occurrence, puisque K=1024, cela signifie que 15 bits sont en moyenne erronés. Dès qu'un bit oscille plus de quatre fois, le taux de bits erronés par trame erronée est supérieur à ce taux moyen. Plus encore, un bit ayant oscillé plus de 8 fois a une probabilité de 15% d'être erroné. Ce qui formalisé donne :

$$\mathsf{P}_{\mathtt{i}\in\mathsf{K}}(\text{bit } \mathtt{i} \text{ est en erreur}) = 0{,}15\%$$

$$\mathsf{P}_{\mathtt{i}\in\mathsf{K}}(\text{bit } \mathtt{i} \text{ est en erreur} \mid \mathrm{OSC}(\mathtt{i}) > 8) = 15\%.$$

Cette probabilité est donc multipliée par un facteur 10. Néanmoins, même dans le cas d'un positionnement dans le plancher d'erreurs, un bit ayant fortement oscillé a plus de chance d'être corrigé que d'être erroné quelque soit le nombre d'oscillations observées.

2.2.3 Turbo codes du standard CCSDS

La même étude à été menée dans le cadre du standard CCSDS. Un rendement de 1/3 est toujours considéré. En revanche, le nombre de bits d'information est quant à lui fixé à 1784. Pour faciliter la lecture de l'analyse comparée des résultats suivante, les visualisations graphiques de ces statistiques sont déportées en Annexe B.

Nombre moyen d'oscillations par bit Tout d'abord les bits erronés oscillent légèrement plus dans le contexte CCSDS que dans le contexte LTE. En effet, quelque soit la valeur de SNR, 12 oscillations $\mathsf{N} \to \mathsf{N}$ et 16 oscillations $\mathsf{N} \to \mathsf{I}$ sont observées. Elles s'établissaient respectivement à 10 et 15 pour le turbo code du standard LTE. Les nombres d'oscillations des bits corrigés diminuent quant à eux avec l'augmentation de la valeur de SNR. Cependant, plus que de la valeur du taux d'erreur trame ou de la valeur de SNR, c'est la position dans la courbe de performance qui dicte ce phénomène. Par exemple, le ratio entre le nombre d'oscillations pour les bit erronés et les bits corrects au sein des trames erronées vaut 1,7 pour le turbo code du standard CCSDS fonctionnant à un taux d'erreur trame de 10^{-6}. Ce dernier est de 1,9 pour le turbo code du standard LTE fonctionnant à un taux d'erreur trame de 10^{-4}. Ainsi, deux ordres de magnitude séparent ces deux mesures, bien qu'elles correspondent toutes deux à un fonctionnement légèrement en amont de la zone du plancher d'erreurs, assurant des statistiques similaires.

Comportement des oscillations au cours du processus Pour de faibles valeurs de SNR, le turbo code du standard CCSDS requiert quelques itérations supplémentaires pour converger en comparaison de son homologue LTE. Cette constatation peut être imputée à l'augmentation de la longueur de contrainte entre les deux turbo codes. Néanmoins, la grande majorité des trames est encore corrigée en moins de 10 itérations. À nouveau, une forte corrélation apparaît entre le nombre d'oscillations et le nombre d'erreurs. Ainsi, le nombre d'erreurs est proche de la moitié du nombre d'oscillations $\mathsf{N} \to \mathsf{N}$. Le nombre d'erreurs est plus important dans le cas du turbo code du standard CCSDS. Mais cette hausse est directement liée aux tailles de trames considérées, à savoir plus importante pour le turbo code du standard CCSDS.
Lorsque la valeur de SNR augmente, le même comportement que celui décrit précédemment pour le turbo code du standard LTE apparaît. Moins d'itérations sont nécessaires pour corriger les trames et une grande part des trames erronées à l'issu du processus de décodage n'a que peu d'erreurs binaires. Néanmoins, à fort SNR, une plus grande démarcation est visible entre les trames erronées avec peu d'erreurs et celles contenant un nombre importants d'erreurs.

Distribution des oscillations au sein des trames erronées En ce qui concerne la dernière statistique, le constat est similaire. Pour un taux d'erreur trame cible de 6×10^{-7}, le taux de bits erronés par trame erronée, ramené en pourcentage, vaut 3,1%. La probabilité d'obtenir un bit erroné est supérieur à ce taux dès que le bit considéré a oscillé plus de 6 fois. Plus encore, un bit ayant oscillé plus de 11 fois a une probabilité de

16% d'être erroné. Ce qui formalisé donne :

$$\mathsf{P}_{\mathtt{i}\in\mathsf{K}}(\text{bit } \mathtt{i} \text{ est en erreur}) = 3{,}1\%$$

$$\mathsf{P}_{\mathtt{i}\in\mathsf{K}}(\text{bit } \mathtt{i} \text{ est en erreur} \mid \mathrm{OSC}(\mathtt{i}) \geqslant 11) = 16\%.$$

Ainsi, la probabilité de choisir un bit erroné a crû d'un facteur 5.

Finalement, les observations réalisées dans le cadre du turbo code du standard CCSDS sont similaires à celles obtenues avec le turbo code du standard LTE. Le paragraphe suivant récapitule ces résultats et conclut cette section.

2.2.4 Analyse globale et conclusions

De ces différentes observations statistiques, plusieurs constats peuvent être énoncés.

1. Tout d'abord, il a été noté que si des oscillations surviennent lors d'une itération, alors la trame n'est pas encore corrigée.
2. De plus, une corrélation forte entre le nombre d'erreurs et le nombre d'oscillations a été mis en évidence.
3. De fait, plus que du nombre d'états du codeur ou de l'entrelaceur, c'est du régime de fonctionnement du turbo décodeur que dépend le nombre d'oscillations et la distribution de ces oscillations.
4. Si un bit oscille fortement lors du processus de décodage, alors, relativement aux autres, il a plus de chance d'être erroné.
5. Ainsi, c'est parmi les bits oscillant le plus que se trouve la plupart des bits erronés.
6. Cependant, comme les bits corrigés à l'issu du processus itératif oscillent eux aussi et sont majoritaires dans la trame, le nombre d'oscillations ne permet pas de prédire de manière fiable si un bit est erroné.

En conclusion, l'observation des oscillations ne semble pas pouvoir permettre de proposer une méthode directe au sein du turbo décodage permettant l'amélioration des performances des turbo codes. Cependant, de part la forte corrélation entre le nombre d'erreurs et le nombre d'oscillations, une modification du décodage EML-MAP exploitant les oscillations est décrite dans la section suivante.

2.3 L'algorithme Self-Corrected EML-MAP

En 2008, V. Savin a proposé une modification de l'algorithme Min-Sum [97] pour le décodage des codes LDPC permettant d'approcher les performances de l'algorithme Sum-Product [97]. Cet algorithme de décodage est nommé Self-Corrected Min-Sum [96].

Dans un premier temps, le principe de cet algorithme est présenté pour le décodage des codes LDPC. Ensuite, une adaptation pour le décodage des turbo codes est proposée. Les performances de ce nouvel algorithme seront alors détaillées.

2.3.1 La méthode originelle pour les codes LDPC

Le décodage des codes LDPC, comme le décodage des turbo codes repose sur un processus itératif. Pour les codes LDPC, chaque itération est composée de trois étapes :

1. Calcul des nœuds de parité,
2. Calcul des nœuds de variable,
3. Calcul de l'information *a posteriori*.

Le principe du Self-Corrected est de détecter les messages non fiables échangés entre les nœuds de variable et de parité lors du décodage itératif et de les « effacer ». Cela correspond à fixer à 0 les valeurs LLR associées à ces messages. Ainsi, la contribution d'un nœud de variable au nœud de parité est effacée si son signe a changé par rapport à l'itération précédente. Dans le cas contraire, sa valeur est transmise, sans modification, aux nœuds de parité. De la sorte, le décodeur détecte les décisions sur les nœuds de variable qui ne sont pas fiables et les écarte du processus de décodage. Hormis cette étape, les autres étapes de cet algorithme sont les mêmes que celles d'un algorithme de propagation de croyance.

Finalement, l'algorithme Self-Corrected permet d'approcher les performances de l'algorithme Sum-Product dans la zone de convergence alors que ce dernier possède une complexité calculatoire bien plus importante. Pis encore, dans certains contextes, il peut même le surpasser dans la zone du plancher d'erreurs.

2.3.2 Adaptation aux turbo codes binaires

La transposition du Self-Corrected Min-Sum au décodage des turbo codes est relativement aisée. En effet, les messages provenant des nœuds de variable à destination des nœuds de parité correspondent aux informations extrinsèques échangées lors du processus itératif de décodage de turbo code.

Ainsi, l'annulation de la contribution peut être effectuée lorsqu'une oscillation $\mathsf{N} \rightarrow \mathsf{N}$ ou $\mathsf{I} \rightarrow \mathsf{I}$ est détectée. L'Algorithme 1 détaille cette adaptation pour le décodage des turbo codes. Le principe de cette approche est donc le suivant : si l'information extrinsèque calculée par un décodeur SISO pour un bit de la trame change de signe d'une itération à la suivante, alors, celle-ci est annulée avant qu'elle ne soit transmise à l'autre décodeur SISO.

Il est à noter que cette adaptation de l'information extrinsèque ne doit pas être appliquée dès la première itération du processus de décodage. En effet, dans ce cas, l'intérêt du processus itératif du turbo décodage se verrait annihilé par l'approche Self-Corrected. Une dégradation des performances vis-à-vis de celles obtenues avec l'algorithme EML-MAP simple serait alors observée. Il est donc préférable de laisser l'algorithme EML-MAP converger durant les premières itérations et ensuite d'appliquer le principe Self-Corrected (SC). À l'aide de simulations Monte-Carlo, il a été montré que le nombre d'itérations sans SC doit être choisi entre 3 et 5 pour obtenir les meilleures performances de décodage. Ces dernières sont présentées en section suivante. Dans la suite, le principe SC est appliqué à partir de la quatrième itération.

Algorithme 1 : : Self-Corrected EML-MAP.

pour *j : 1 à* I_{max} **faire**
  Décodage $SISO_1$ EML-MAP
  si *j>4* **alors**
    pour chaque $k \in K$ **faire**
      si $sgn\left(\mathbf{L_{12}^{e\,(j)}}(k)\right) \neq sgn\left(\mathbf{L_{12}^{e\,(j-1)}}(k)\right)$ **alors**
        $\mathbf{L_{12}^{e\,(j)}}(k) \leftarrow 0$
  Décodage $SISO_2$ EML-MAP
  si *j>4* **alors**
    pour chaque $k \in K$ **faire**
      si $sgn\left(\mathbf{L_{21}^{e\,(j)}}(k)\right) \neq sgn\left(\mathbf{L_{21}^{e\,(j-1)}}(k)\right)$ **alors**
        $\mathbf{L_{21}^{e\,(j)}}(k) \leftarrow 0$

2.3.3 Performances du principe SC appliqué aux turbo codes binaires

Cette section présente les performances de décodage obtenues avec l'algorithme Self-Corrected appliqué au turbo décodage. Dans un premier temps, l'étude est menée en considérant un entrelaceur uniforme. Dans un second temps, un cas concret de turbo codes standardisé est considéré.

2.3.3.1 Entrelaceur uniforme

Comme présenté en section 1.2.1.3, l'entrelaceur uniforme permet d'extraire les paramètres des codes constituant un turbo code. Il permet aussi de décorréler l'impact de l'étape d'entrelacement des performances de décodage.

L'étude est menée sur des turbo codes binaires à 8, 16 et 32 états. Pour les deux premiers, le codeur du standard LTE et celui du standard CCSDS sont respectivement considérés. Leurs polynômes générateurs en octal sont {13,15} et {23,33}. Le turbo code à 32 états présentant les meilleures propriétés de distance a quant à lui pour générateur {43,63}. C'est celui-ci que nous avons retenu.

La Figure 2.7 présente à la fois le taux d'erreur binaire et le taux d'erreur trame pour les trois turbo codes sus-mentionnés. La trame traitée par le turbo codeur se compose de 1504 bits. Le décodage est réalisé par l'algorithme EML-MAP ou par l'algorithme Self-Corrected EML-MAP (SC EML-MAP), suivant l'algorithme ci-dessus. Dans les deux cas, le facteur de remise à l'échelle des informations extrinsèques vaut 0,75 tout au long du processus itératif. Le nombre maximal d'itérations est fixé à 8. Un critère d'arrêt basé sur un CRC de 24 bits [1] est utilisé afin de réduire le temps de simulation.

Au niveau du taux d'erreur trame, nous pouvons remarquer que l'algorithme SC EML-

1. provenant du standard LTE

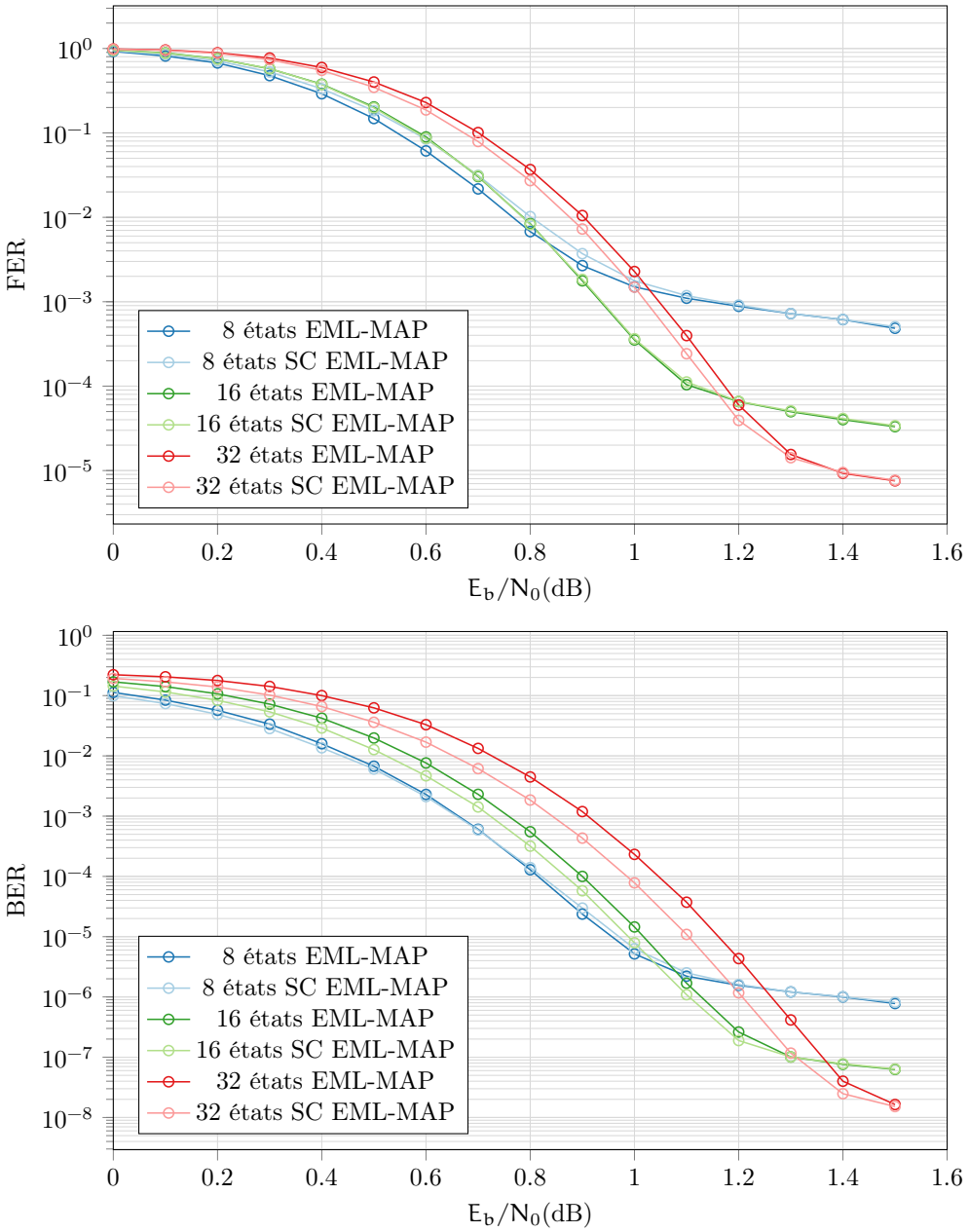


FIGURE 2.7 – Comparaison des performances de décodage entre l'algorithme EML-MAP et l'algorithme SC EML-MAP, 8 itérations au maximum, $\mathsf{K} = 1504$, $\mathsf{R} = \frac{1}{3}$.

MAP a des performances moins intéressantes que celles de l'algorithme EML-MAP pour un turbo code à 8 états. Pour le turbo code à 16 états, les performances sont équivalentes. Enfin, pour le turbo code à 32 états, un léger gain apparaît en faveur de l'algorithme SC. En revanche, en ce qui concerne le taux d'erreur binaire, des gains apparaissent pour les turbo codes à 16 et à 32 états. Les performances sont les mêmes pour le turbo code à 8 états.

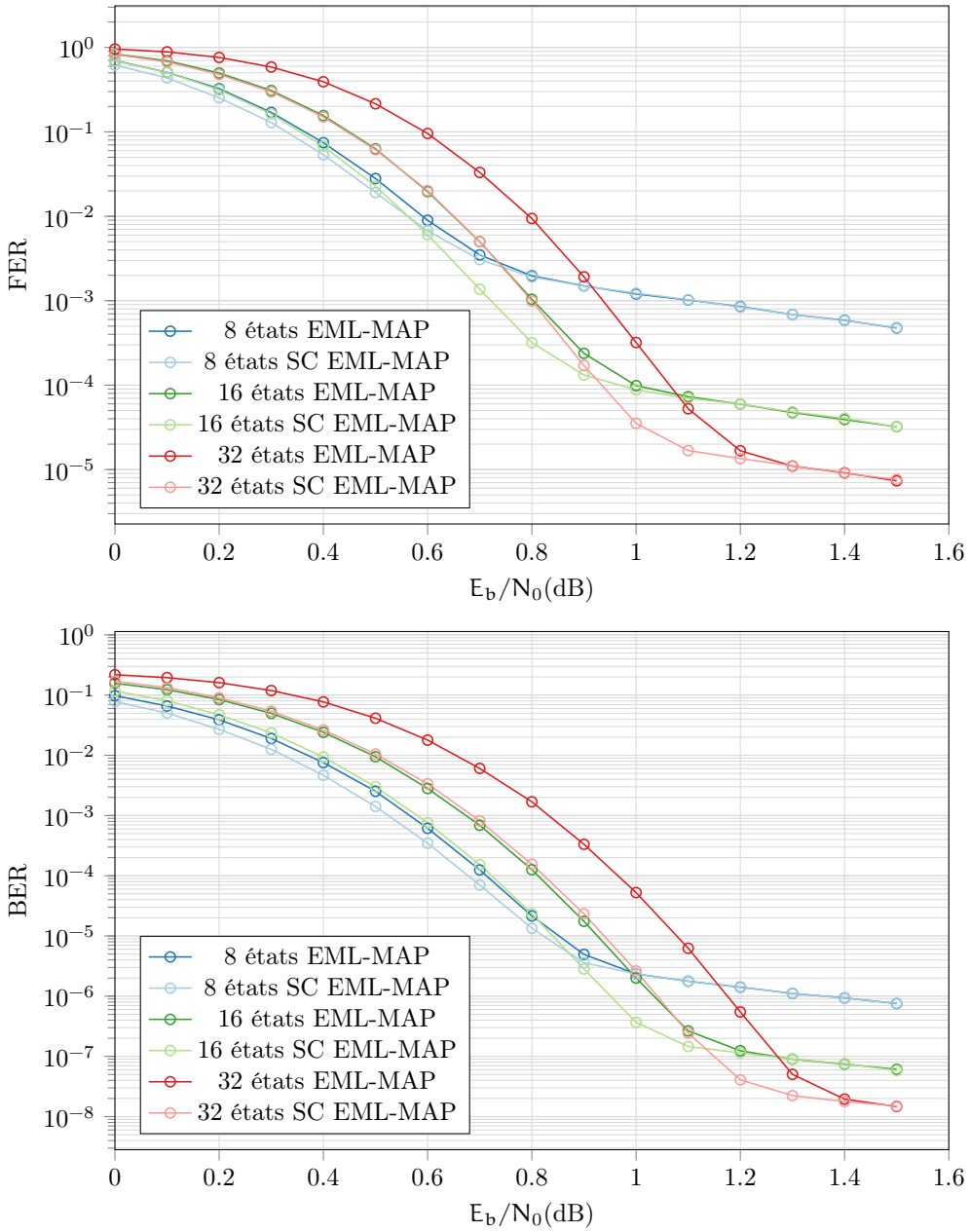


FIGURE 2.8 – Comparaison des performances de décodage entre l'algorithme EML-MAP et l'algorithme SC EML-MAP, 32 itérations au maximum, $K = 1504$, $R = \frac{1}{3}$.

Ainsi, dans tous les cas, le nombre de bits erronés par trame erronée est réduit via l'adjonction « d'effacements » de l'information extrinsèque. En réduisant les oscillations au cours du processus itératif, mais sans néanmoins les inhiber, le nombre d'erreurs est diminué. Un impact important quant à la longueur de contrainte du codeur élémentaire apparaît : la correction apporte des gains lorsque la longueur de contrainte augmente. La sous-optimalité du décodage est donc légèrement réduite.

La Figure 2.8 reprend la même étude mais en fixant cette fois le nombre maximal d'itérations à 32. Maintenant, quelque soit le nombre d'états, des gains sont obtenus à la fois en terme de taux d'erreur trame et en taux d'erreur binaire.
Dans les deux cas, nous pouvons montrer que les taux d'erreurs dans la zone de convergence sont équivalents entre un turbo code de longueur de contrainte L décodé via l'algorithme EML-MAP itérant 32 fois et un turbo code de longueur de contrainte $2 \times \mathsf{L}$ décodé via l'algorithme SC EML-MAP itérant 32 fois. En résumé, en utilisant un entrelaceur uniforme et en fixant le nombre maximal d'itérations à 32, les gains obtenus dans la zone de convergence en utilisant l'algorithme SC EML-MAP à la place de l'algorithme EML-MAP correspondent à :

— 0,02 dB pour un turbo code à 8 états,
— 0,1 dB pour un turbo code à 16 états,
— 0,15 dB pour un turbo code à 32 états.

Comme présenté en Chapitre 1, l'augmentation de la longueur de contrainte permet une hausse de la distance minimale du code. Cela permet donc d'abaisser le plancher d'erreurs d'un turbo code. Cependant, le décalage du seuil de convergence inhérent à cette augmentation peut empêcher son choix dans un contexte nécessitant de bonnes performances de correction pour de faibles valeurs de SNR. Grâce à l'algorithme SC EML-MAP, ce désagrément disparaît. Il est possible de profiter d'une augmentation de la $\mathtt{d}_{\mathtt{min}}$ du code tout en ne subissant pas l'un des désavantages inhérent.

Dans la section suivante, le turbo code du standard CCSDS est considéré. Ce turbo code possède une longueur de contrainte de 5. Les résultats qui viennent d'être exposés montrent que cette longueur de contrainte est la longueur minimale requise afin d'observer des gains notables au niveau performances.

2.3.3.2 Turbo code standardisé

La Figure 2.9 présente la comparaison des performances de décodage d'un turbo code du standard CCSDS. La taille de la trame considérée est de 1784 bits et le rendement du code vaut 1/3. Dans ce standard, un code CRC est défini en amont du turbo code [57]. Ce CRC, nommé CRC-CCITT-16, a pour polynôme générateur $(1021)_{16}$. Il a donc une traille de 16 bits. Il est employé dans la suite comme critère d'arrêt. De part sa faible taille, sa distance minimale est elle aussi relativement faible. Afin de ne pas dégrader les performances de décodage provenant d'erreurs non détectées par le code CRC, la vérification n'est réalisée qu'à partir de l'itération 4 du processus de turbo décodage. C'est aussi à partir de cette itération que le principe SC commence à être utilisé.

Dans ce contexte, il apparaît que les gains obtenus grâce au principe Self-Corrected, exprimés en terme de dB, sont similaires à ceux de la section précédente, dans le cas d'un turbo code à 16 états. Ainsi, en considérant 32 itérations, les gains correspondent à un ordre de grandeur en terme de taux d'erreur trame et de taux d'erreur binaire dans la zone de convergence vis-à-vis de l'algorithme EML-MAP. Ces gains permettent d'obtenir les mêmes performances en terme de FER que celles obtenues en considérant l'algorithme MAP itérant 10 fois. La complexité calculatoire de l'algorithme MAP est largement

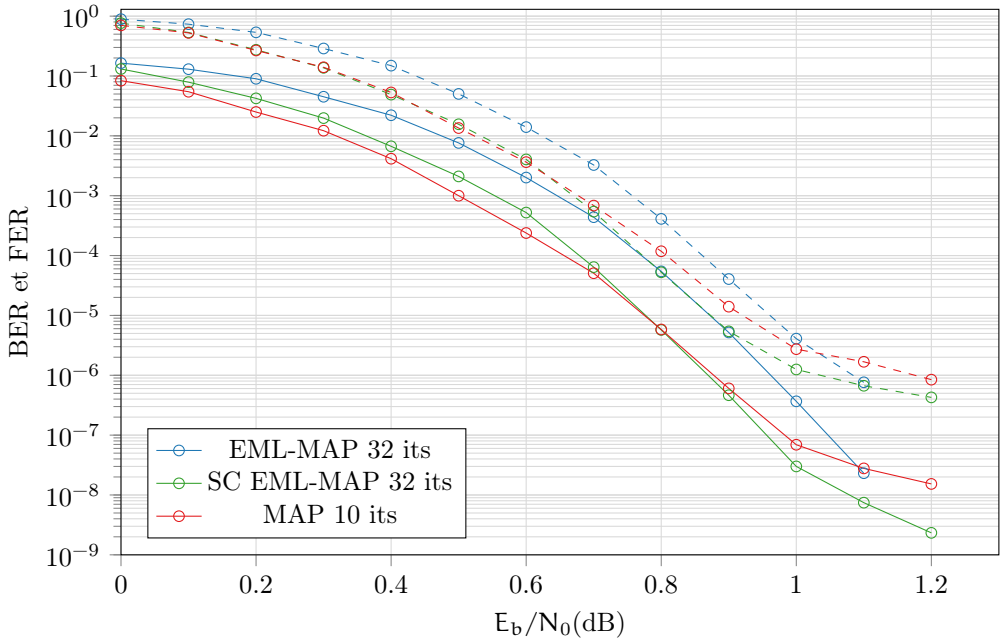


FIGURE 2.9 – Comparaison des performances décodage entre les algorithme EML-MAP, SC EML-MAP et MAP. Turbo code du standard CCSDS (K=1784, R=1/3).

supérieure à celle de l'algorithme EML-MAP. En effet, ce premier requiert l'emploi de fonctions complexes tels que des logarithmes, des exponentielles et des opérations de multiplication. Ainsi, le coût en latence qui est dû au triplement du nombre d'itérations nécessaires au bon fonctionnement du SC EML-MAP est à mettre en regard de la complexité calculatoire de l'algorithme MAP. De plus, grâce à l'utilisation du code CRC comme critère d'arrêt, le nombre moyen d'itérations est largement inférieur à 10. Par exemple, pour une valeur de SNR de 1 dB, plus de 99% des trames sont corrigées à la quatrième itération. Celle-ci étant la première itération à partir de laquelle le CRC est vérifié.

Le tableau 2.1 donne le nombre moyen d'itérations nécessaire au turbo décodeur selon qu'il soit basé sur l'algorithme SC EML-MAP ou sur l'algorithme EML-MAP. Dans les deux cas, ces valeurs sont approximativement les mêmes. Puisque la contribution de certaines valeurs extrinsèques est annulée et puisqu'il faut un nombre conséquent d'itérations pour observer des gains lors de l'application du SC EML-MAP, il pourrait être attendu que le principe SC ralentisse la vitesse de convergence du processus itératif. Cependant, cela n'est pas constaté dans le nombre d'itérations moyen.

En fait, les nombres moyens d'itérations sont proches du nombre moyen d'itérations utilisées lorsqu'un critère d'arrêt génie – c'est-à-dire en connaissant la trame transmise – est considérée [98]. L'écart observé s'explique par le fait que le critère d'arrêt génie peut être déclenché dès la première itération, à la différence de la vérification du code CRC.

	EML-MAP	SC EML-MAP	Génie ([98])
0,6 dB	5,9	5,9	4,0
0,8 dB	4,3	4,4	3,2
1,0 dB	4,0	4,0	–

TABLEAU 2.1 – Comparaison du nombre moyen d'itérations nécessaires selon l'algorithme de turbo décodage.

2.3.3.3 Turbo codes double binaires

Des tentatives d'extension du principe Self-Corrected de l'information extrinsèque au cas des turbo codes double binaires ont été menées. Cependant, lors du décodage d'un tel code, trois valeurs de LLR sont échangées entre les décodeurs élémentaires pour un symbole. Chacun de ces LLR représente alors respectivement la probabilité que le symbole 1, le symbole 2 ou le symbole 3 ait été émis rapporté à la probabilité que le symbole 0 ait été émis.

Dans le cas binaire, le principe Self-Corrected peut être interprété de la sorte : « lorsqu'un des décodeurs élémentaire change d'avis sur la valeur d'un bit d'une itération à l'autre, sa contribution au processus itératif est annulée pour le bit considéré ». Dans un contexte double binaire, la décision correspond à 0 si les trois LLR sont négatifs et à l'index du maximum sinon. Ainsi, l'élément déclencheur de la correction SC peut être le changement de l'index du maximum parmi ces trois LLR.

« L'annulation de la contribution » dans le cadre binaire ne peut avoir différentes interprétations. En effet, cela correspond forcement à passer la valeur du LLR considéré à 0. Ainsi le poids de la probabilité que le bit soit 0 et le poids de la probabilité que le bit soit 1 sont les mêmes. En revanche, dans le cas double binaire l'interprétation n'est pas aussi triviale, car 4 probabilités sont simultanément considérées. Ainsi, plusieurs interprétations sont possibles. Pour plus de clarté, un exemple illustrant ces propositions est présenté dans le tableau 2.2. L'approche Self-Corrected peut donc affecter les valeurs des informations extrinsèques :

- en les assignant toutes trois à la même valeur (a) :
  - si la valeur 0 est choisie, alors seul le canal contribue au calcul des métriques de branches,
  - si une autre valeur est choisie, elle n'aura toute fois pas d'impact en raison de l'opération ACS dans le calcul des métriques de nœuds.
- en assignant deux données extrinsèques à une même valeur et la troisième à une autre. Trois cas sont alors à considérer :
  1. répétition de la valeur maximale (b),
  2. répétition de la valeur médiane (c),
  3. répétition de la valeur minimale (d) et (e).
- en usant d'une remise à l'échelle supplémentaire (f).
- en inhibant le calcul nouvellement effectué par le décodeur (g).

	Valeur initiale		Propositions						
	It n	It n+1	a	b	c	d	e	f	g
$L_1^e(k)$	7	6	0	8	6	2	6	6	7
$L_2^e(k)$	5	8	0	8	6	6	2	4,5	5
$L_3^e(k)$	2	2	0	2	2	2	2	1,5	2

TABLEAU 2.2 – Exemples des différentes interprétations considérées pour la correction de l'information extrinsèque pour les turbo codes double binaires.

L'ensemble de ces propositions a été mise en œuvre et évalué par des simulations Monte-Carlo. Les turbo codes des standards DVB-RCS et DVB-RCS2 ont été considérés. Aucun gain significatif n'a été relevé, quels que soient les rendements de code ou les tailles de trames retenues. Dans le meilleur des cas, un gain d'un facteur 2 apparaît en terme de taux d'erreur trame.

Des espoirs résidaient dans le standard DVB-RCS2 puisque ce turbo code double binaire se compose de 16 états, nombre d'états permettant l'observation de gains notables dans le cas de turbo codes binaires. Cependant, l'absence de gains est sans doute à mettre en corrélation avec la constatation énoncée dans [56]. Cette dernière indique que l'un des avantages des turbo codes double binaires vis-à-vis de leurs homologues binaires est le plus faible écart au niveau des performances de décodages entre l'algorithme EML-MAP et l'algorithme MAP. Ainsi, dans le cas double binaire, la sous optimalité de l'algorithme ML-MAP vis-à-vis de l'algorithme MAP est moins importante. Les gains pouvant être obtenus par un algorithme visant à corriger la sous optimalité pour de tels codes ne peuvent être donc que plus faibles.

À la vue des faibles gains obtenus via cette tentative de transposition, l'étude a été interrompue pour ce contexte. En section suivante, une étude architecturale portant sur le décodage de turbo codes binaire selon l'algorithme SC EML-MAP est menée.

2.3.4 Etude architecturale

Dans cette section, la description d'une architecture matérielle permettant le décodage de turbo codes exploitant le principe Self-Corrected de l'information extrinsèque est menée.

La Figure 2.10 présente l'ensemble de l'architecture proposée. Elle a pour base un décodeur ML-MAP séquentiel, comme décrit dans [99]. Son principe est le suivant. Les informations du canal et l'information extrinsèque calculée à la demi-itération précédente sont traités séquentiellement. De la première paire de données, les métriques de branches sont calculées. Elles permettent alors de calculer les métriques de nœuds aller. Parallèlement, les métriques de branches sont mémorisées dans un registre tampon LIFO (Last In First Out ou dernier arrivé, premier servi). Une fois les métriques de nœud aller calculées, le calcul des métriques de nœuds retour débute. Pour la première section de treillis, les métriques de branches, de nœuds aller et de nœuds retour sont alors calculées. Elles sont combinées selon qu'elles correspondent à une transition associée à un 0 émis ou un 1 émis. Le maximum

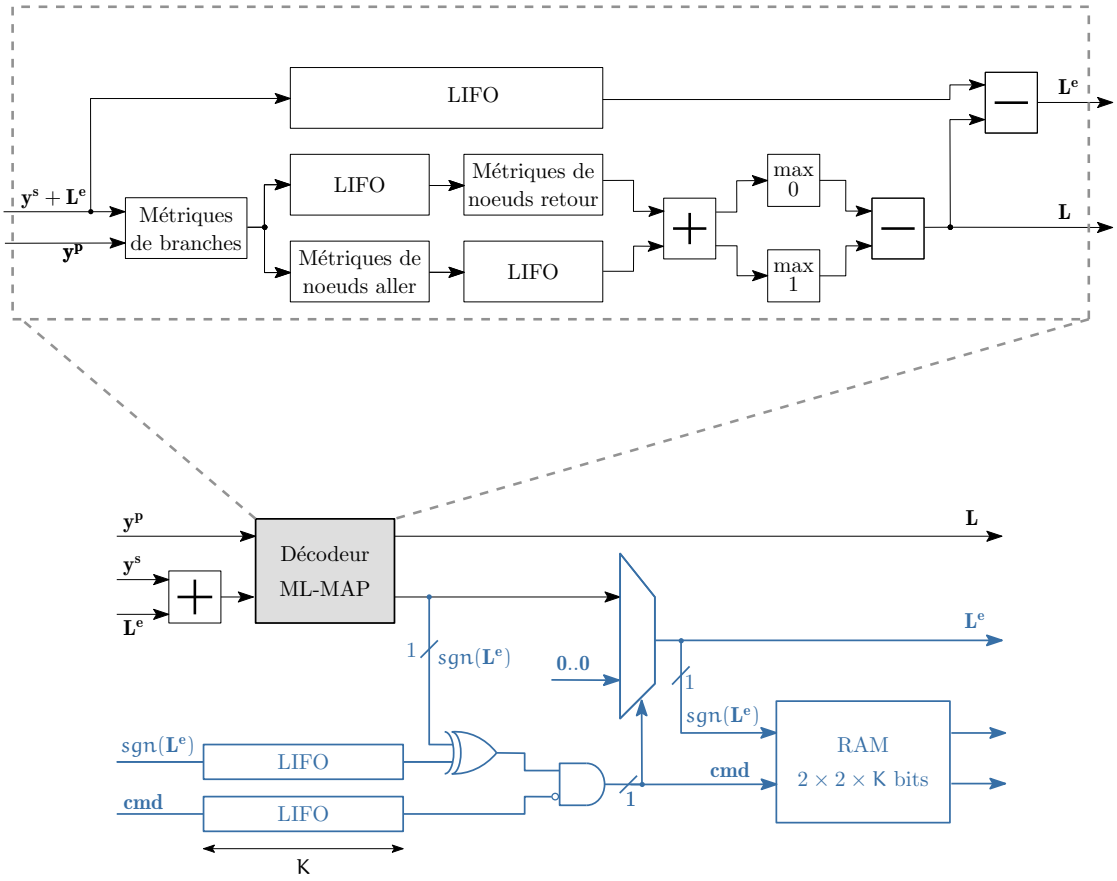


FIGURE 2.10 – Architecture matérielle d'un turbo décodeur séquentiel intégrant le principe Self-Corrected de l'information extrinsèque.

correspondant à chacune des transitions est extrait. Ils sont alors soustrait entre eux afin d'obtenir l'information *a posteriori*. N'y reste plus qu'à soustraire l'information provenant du canal et celle calculée à la demi-itération précédente pour obtenir l'information extrinsèque. Celle-ci est alors stockée en mémoire afin de pouvoir être utilisée lors de la demi-itération suivante.

Le principe Self-Corrected de l'information extrinsèque doit être réalisé juste avant que l'information extrinsèque ne soit stockée. Pour sa mise en œuvre, il est nécessaire d'avoir à disposition deux informations. D'une part, le signe de l'information extrinsèque calculée à l'itération précédente et d'autre part la sauvegarde de l'utilisation ou non du principe Self-Corrected à l'itération précédente. Ainsi, il est nécessaire de stocker ces deux informations sur deux bits dans une mémoire. Dès lors, il suffit de combiner ces deux informations avec le signe de l'information extrinsèque qui vient d'être calculée par le décodeur SISO. Le résultat permet de commander un multiplexeur assignant en sortie l'information extrinsèque inchangée ou 0.

En résumé, l'essentiel de la complexité matérielle introduite par le principe SC consiste en

un effort supplémentaire de mémorisation. Par bit d'information, deux bits additionnels sont nécessaires. De plus, comme ces deux bits sont requis à l'itération suivante et non à la demi-itération suivante, la taille totale de la mémoire additionnelle représente $2 \times 2 \times \mathsf{K}$ bits. En comparaison, une architecture séquentielle de turbo décodeur requiert le stockage de l'information du canal et de l'information extrinsèque. D'après [99], un choix de quantification usuel considère l'utilisation de 6 bits pour l'information provenant du canal et de 7 bits pour l'information extrinsèque. Il est donc nécessaire de stocker au total $(3 \times 6)\mathsf{K} + 7 \times \mathsf{K}$ bits. L'effort de mémorisation relatif au bon fonctionnement de l'algorithme SC entraîne donc une hausse de 16% du coût global de mémorisation du décodeur. La logique combinatoire nécessaire est quant à elle négligeable puisqu'elle représente un multiplexeur 7 bits, une porte ou-exclusif, une porte inverseuse et une porte et.

Cette architecture peut être adaptée à un degré de parallélisme plus élevé. Cela correspond à utiliser plusieurs décodeurs SISO chacun travaillant sur une portion de la trame. La taille de la mémoire nécessaire à la mise en œuvre de l'algorithme SC EML-MAP n'est donc pas modifiée. En revanche, la logique combinatoire se doit d'être multipliée par le nombre de décodeurs SISO utilisés.

2.3.5 Conclusion

Une adaptation de l'algorithme Self-Corrected originellement proposé pour les codes LDPC a été présentée pour les turbo codes. Cet algorithme, peu coûteux à mettre en œuvre permet d'obtenir, sans modifier le turbo codeur, des gains dans la zone de convergence. Toutefois, cette approche suppose quelques contraintes afin que les gains énoncés soient conséquents. D'une part, le nombre maximal d'itérations se doit d'être plus important que dans les schémas de décodage courants. Cependant, grâce à l'utilisation d'un critère d'arrêt, le nombre moyen d'itération employé reste similaire. Ce travail a fait l'objet de publications en conférences nationales [113, 114].

D'autre part, afin d'observer des gains aussi en terme de taux d'erreur trame, il est nécessaire que le nombre d'états du codeur soit supérieur à 8. Aucune hypothèse n'a pu être vérifiée, quant à la cause de ce phénomène. D'autant plus que les oscillations internes au turbo décodeur semblent avoir le même comportement entre 8 et 16 états à la vue des statistiques précédentes.

Les transpositions au cas des turbo codes double binaires n'ont pas abouti à des gains significatifs. Le faible écart de performance entre les algorithmes MAP et ML-MAP dans ce contexte peut expliquer ce résultat.

Finalement, une architecture matérielle compatible avec un décodeur EML-MAP séquentiel a été proposée. Il a été montré que l'essentiel du surcoût de complexité matérielle de cet algorithme réside dans la mémorisation. Ce surcoût reste modéré puisque seuls $4 \times \mathsf{K}$ bits supplémentaires sont nécessaires.

2.4 Turbo décodages successifs

Comme évoqué dans le chapitre premier, une approche permettant l'observation de gains de performance à la fois dans la zone de convergence et celle du plancher d'erreurs est celle reposant sur des décodages multiples. Dans la littérature, trois articles majeurs dérivent cette méthode [87, 88, 89]. Dans la suite, le principe de ce type de décodage est rappelé. Des performances sont présentées. Une adaptation considérant le nombre d'oscillation comme métrique est enfin présentée.

2.4.1 État de l'art : Correction Impulse Method

L'approche Correction Impulse Method (CIM) est basée sur l'application de turbo décodages successifs. Tout d'abord, un turbo décodage classique est réalisé. Si la trame n'est pas corrigée, alors une identification des positions les moins fiables est effectuée. L'extraction est réalisée en prenant en considération les informations *a posteriori* obtenues lors de la dernière itération. Plus la valeur absolue de l'information *a posteriori* est faible, moins la décision est fiable. Une liste de positions non fiables est alors extraite. Tour à tour, la valeur de l'information systématique provenant du canal associée aux positions concernées est saturée à une valeur supérieure à la distance minimale du code. Le signe de cette valeur de saturation prend le signe opposé au mot obtenu lors du premier turbo décodage. Un nouveau turbo décodage est alors appliqué. Si le décodage est correct, le processus est stoppé. Sinon, une nouvelle valeur d'information systématique provenant du canal est saturée et le turbo décodage associé est appliqué. Le nombre maximal de turbo décodages pour une même trame est fixé à L. L'algorithme 2 récapitule l'ensemble de ces opérations.

Algorithme 2 : : Correction Impulse Method (CIM).

1 $(\mathbf{L},\hat{\mathbf{d}}) \leftarrow$ `Turbo Décodage EML-MAP(`$\mathbf{y^s}$, $\mathbf{y^p}$, $\mathbf{y'^p}$`)`
2 **idx** $\leftarrow$ `Trie selon valeurs croissantes(`**L**`)`
3 `Sauvegarde(`$\mathbf{y^s}$`)`
4 $j \leftarrow 0$
5 **tant que** $j < L$ *et* `CRCheck(`$\hat{\mathbf{d}}'$`)` $==$*false* **faire**
6 `Restauration(`$\mathbf{y^s}$`)`
7 $\mathbf{y^s}[\mathbf{idx}[j]] \leftarrow (-1)^{\hat{d}} \times d_{min}$
8 $\hat{\mathbf{d}}' \leftarrow$ `Turbo Décodage EML-MAP(`$\mathbf{y^s}$, $\mathbf{y^p}$, $\mathbf{y'^p}$`)`
9 $j \leftarrow j + 1$

L'amplitude de la saturation est choisie de telle sorte qu'elle soit supérieure à d_{min} pour s'assurer que le décodeur fournisse un résultat différent du premier. Ceci est à mettre en corrélation avec la méthode d'obtention de la distance minimale d'un turbo code par la méthode d'impulsion d'erreur [63] présentée dans le premier chapitre.

Dans [87], les turbo codes double binaires sont aussi considérés. Pour chaque symbole, 4 valeurs LLR *a posteriori* sont générées. La métrique permettant d'identifier les positions

les moins fiables consiste alors à prendre la différence entre la plus grande valeur de LLR et la deuxième plus grande valeur de LLR. Trois turbo décodages sont réalisés pour chaque position identifiée. Ainsi, les trois sens de saturation associés aux trois autres symboles possibles en excluant le symbole décodé sont considérés. Le nombre maximal de turbo décodages vaut alors $3 \times \mathsf{L} + 1$.

Dans cette publication, le cadre des simulations consiste en des turbo codes binaires et double binaires à 8 états, de rendement 1/3 et 1/2 et avec une taille d'information de 1504 bits. L'algorithme EML-MAP itérant au maximum 16 fois est employé. Plusieurs nombres de décodages supplémentaires sont considérés, allant de 1 à la taille de la trame. Dans tous les cas, des gains dans les zone de convergence apparaissent, augmentant avec L. Cependant, les gains les plus significatifs se trouvent dans la zone du plancher d'erreurs. Dans le cas binaire, avec $\mathsf{L} = 1$, les gains atteignent un ordre de grandeur. La multiplication par 4 de L permet à chaque fois d'augmenter les gains d'environ un ordre de magnitude.
Dans le cas double binaire, les gains globaux sont moins importants mais dépassent tout de même un ordre de magnitude avec $\mathsf{L} = 4$.

Dans [88], il est montré qu'une amélioration des performances est observée lorsque tous les sens de saturation sont testés. Ainsi, dans le cas binaire, deux turbo décodages sont effectués par position identifiée. Pour un turbo code double binaire, 4 turbo décodages peuvent être réalisés successivement. Dans cette publication, des simulations sont présentées pour un turbo code double binaire à 8 états et de rendement 1/2. La taille de trame est de 1504. Dans performances similaires à celles de [87] sont observées. Cette approche est nommée Forced Symbol Method (FSM).

Finalement, [89] étend le principe de turbo décodages successifs en considérant cette fois l'ensemble des positions de la trame ($1/\mathsf{R} \times \mathsf{K}$). Ainsi, au maximum, $2 \times 1/\mathsf{R} \times \mathsf{K}$ turbo décodages peuvent être réalisés pour une trame. Les informations de redondance provenant du canal peuvent être elles aussi saturées. Cette approche permet donc d'aboutir aux meilleures performances de décodage grâce au principe de turbo décodages successifs. De part son contexte ne nécessitant pas toujours une forte contrainte en latence, le standard CCSDS est choisi comme cadre d'étude. À l'inflexion entre la zone de convergence et celle du plancher d'erreurs, cet algorithme permet d'obtenir un gain de deux ordres de magnitude faisant passer le taux d'erreur trame d'environ 10^{-6} à 10^{-8}. En adjonction du FSM, un décodage de Viterbi par liste est aussi considéré. Cependant, les auteurs indiquent que le gain de performance amené par ce second algorithme est marginal. Finalement, les auteurs statuent sur le fait qu'un code CRC de taille 16 ou moins est restrictif dans cette approche puisque la probabilité d'erreurs non détectées devient trop importante. Dans [88], les auteurs considèrent qu'un code CRC de taille 24 bits est suffisant pour que l'approche de turbo décodages successifs ne soit pas impactée par les faux positifs issus du code CRC.

La Figure 2.11 (a) présente les taux d'erreur trame relevés dans [87] dans le cadre d'un turbo code binaire, de rendement 1/3 et de taille de trame 1504. Un entrelaceur DRP (cf. 1.2.1.3) est utilisé. Chaque processus de turbo décodage itère jusqu'à 16 fois. La notation CIM(L) signifie qu'au maximum L différentes positions dans la trame peuvent

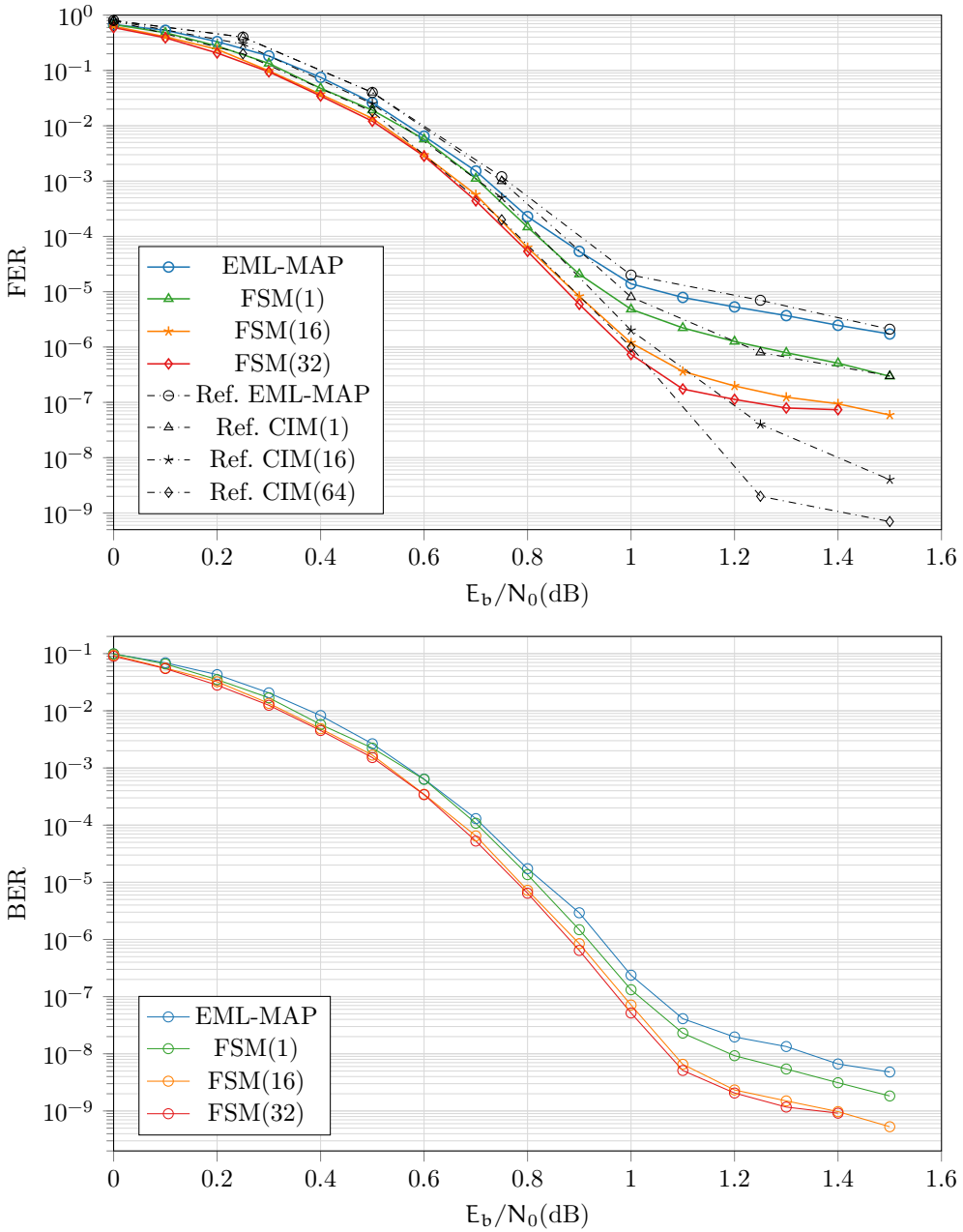


FIGURE 2.11 – Comparaison des performances de décodage entre les algorithme EML-MAP et FSM (recopiées de [87] et rejouées) avec plusieurs profondeurs de recherche. Turbo code du standard LTE (K=1504, R=1/3).

être saturées. Ces différentes courbes sont présentées en pointillé.

Une reproduction des résultats a été tentée. Cependant, dans [87] les paramètres de l'entrelaceur DRP ne sont pas fournis. L'entrelaceur LTE a alors été utilisé. Ce dernier est basé sur une permutation QPP. Les propriétés de distance du turbo code résultant

sont donc normalement aussi bonnes que celles obtenues avec l'entrelaceur DRP. Dans [87], un critère d'arrêt génie est considéré. Afin de statuer sur un système réaliste, le code CRC du standard LTE est utilisé comme critère d'arrêt. Finalement, dans la tentative pour rejouer les résultats, la méthode FSM [88] est employé, ceci étant censé améliorer les performances puisque considérant les deux sens de saturation.

Sans turbo décodages successifs, les performances de décodage sont semblables. Un léger gain en convergence apparaît sur la courbe pleine provenant de l'augmentation du nombre d'itérations. En considérant $\mathrm{L} = 1$, les performances dans la zone du plancher d'erreurs sont à nouveau similaires. Cependant, la référence considère un seul sens de saturation par position retenue alors que l'approche FSM(1) en considère deux.

Lorsque l'identification des 16 positions les moins fiables est appliquée, pour une valeur de SNR inférieure à 1,1 dB, les performances de décodage sont similaires. En revanche dans la tentative de reproduction des résultats, la zone du plancher d'erreurs arrive dès 1.2 dB alors que dans [87], l'inflexion de la courbe est bien moins marquée permettant à ce schéma de décodage d'atteindre un FER de 4.10^{-9} pour une valeur de SNR de 1,5 dB alors qu'elle n'est que de 8.10^{-8} dans nos expérimentations Si la valeur de L augmente, des résultats similaires sont observés. Pour la référence, l'inflexion diminue, le plancher d'erreurs se voit drastiquement abaissé fournissant des gains importants en utilisant cette méthode. Cependant, dans notre tentative de reproduction des résultats ce comportement n'apparaît pas, la zone du plancher d'erreurs se trouvant être dans la plage de taux d'erreur $[10^{-8}\ ;\ 10^{-7}]$. Plusieurs valeurs de saturation ont aussi été considérées, sans succès sur l'amélioration des performances :

- 0,0, correspondant à une annulation de la contribution,
- 4,0, correspondant à une faible saturation du décodeur. Cette valeur est similaire à celles obtenues en sortie du canal,
- 10,0, correspondant environ à la moitié de la distance minimale du code,
- 25,0, correspondant environ à la distance minimale du code, comme préconisé dans [87, 88],
- 35,0, correspondant à une valeur supérieure à la $\mathtt{d}_{\mathtt{min}}$, assurant, normalement un décodage différent.

Aucune de ces valeurs n'a permis d'aboutir à un comportement réellement différent au niveau des simulations Monte-Carlo.

Les résultats en terme de taux d'erreur binaire sont quant à eux difficiles à analyser. En effet, les simulation Monte-Carlo ont été réalisées de telle sorte que si le code CRC n'est pas vérifié, la trame produite par le décodeur est la dernière calculée. Or, celle-ci correspond à la trame avec l'impulsion localisée sur la position la plus fiable parmi celles considérées. Il en résulte que si la trame n'est pas corrigée une augmentation du nombre d'erreurs dans la trame erronée apparaît. Ainsi, si le code CRC n'est pas vérifié, afin d'obtenir les meilleurs performances en taux d'erreurs trame et binaire, il faudrait transmettre le mot obtenu à l'issue du premier décodage.

Plusieurs hypothèses peuvent expliquer la différence des résultats en terme de taux d'erreur trame. Tout d'abord, dans [87], l'impact de la méthode est évalué pour un sous ensemble de trames erronées à l'issue du premier décodage. Ainsi, les taux d'erreurs

calculés risquent de ne pas être statistiquement fiables.
D'autre part, l'entrelaceur peut aussi avoir un impact important. En effet, il est conjecturé dans [88] que cette méthode permet, grâce à l'utilisation du CRC, d'augmenter la distance minimale du code. Ainsi, au niveau du spectre de distance du code, cela revient à supprimer au moins son premier terme. Les distance minimales obtenues via les différents formalismes d'entrelaceurs courants (DRP, QPP et ARP) sont équivalentes. Ceci est vérifié par le fait que les courbes de la Figure 2.11 sont semblables avec un décodage EML-MAP. En revanche, entre deux entrelaceurs permettant d'atteindre la même distance minimale (ayant le plus grand impact sur la zone du plancher d'erreurs), les termes suivants du spectre peuvent différer. Si la distance entre les deux premières valeurs du spectre de distance est plus importante pour l'entrelaceur DRP considéré que pour l'entrelaceur LTE, alors la suppression du premier terme augmente d'autant plus l'abaissement du plancher d'erreurs. Malheureusement, cette explication ne peut être vérifiée qu'avec l'obtention des paramètres de l'entrelaceur DRP utilisé dans [87].
Finalement, la dernière explication pouvant être conjecturé est la limitation induite par les capacités de détection du code CRC considéré. Cependant, il est indiqué dans [88] qu'un code CRC de taille 24 bits devrait être suffisante. Ainsi, des tests incluant l'utilisation d'un critère d'arrêt génie ont été réalisés. Des gains ont été observés sans toutefois permettre de reproduire les performances présentées dans [87].

Pour conclure sur ces techniques utilisant des décodages successifs, malgré les différentes configurations testées, les gains énoncés dans la zone du plancher d'erreurs n'ont pu être reproduits. Des gains sont observés mais ils dépassent difficilement un ordre de magnitude dans le plancher d'erreurs. En revanche, dans la zone de convergence, les gains observés sont similaires.

Néanmoins, le principe de décodage successif est attrayant car relativement facile à mettre en œuvre pour une implémentation logicielle, car la mise en mémoire tampon des données n'est pas un problème. Pour des communications ayant une faible contrainte de latence, ce principe permet d'apporter une solution relativement simple pour abaisser le plancher d'erreurs. Dans la section suivante, une métrique basée sur les oscillations au sein du turbo décodeur est alors proposée. Les performances résultantes sont comparées avec les performances présentées dans cette section.

2.4.2 Utilisation de métriques basées sur les oscillations

La section 2.2 a mis en exergue une corrélation entre le nombre d'erreurs à l'issu du décodage et le nombre d'oscillations sur la valeur des informations extrinsèques durant le processus itératif. Ainsi, la proposition faite dans cette section en découle. Elle consiste à identifier les positions dans la trame qui ont le plus oscillé avant de réaliser, à partir de cette liste, des turbo décodages successifs.

En effet, il parait intuitif de statuer que saturer l'information du canal d'un bit oscillant fortement lors du premier décodage va fondamentalement modifier le résultat produit par le turbo décodeur. Ainsi, l'algorithme 2 n'est que peu modifié. Durant le premier turbo décodage, en plus de fournir la trame décodée, le décodeur doit produire le nombre

d'oscillations recensées pour chacun des bits durant le processus itératif. Cette information est alors triée selon un ordre décroissant. Les index associés aux bits ayant les plus oscillé sont alors pris en compte pour modifier l'information du canal avant d'effectuer des turbo décodage successifs. L'algorithme 3 récapitule ces différentes étapes.

Algorithme 3 : : Forced Symbol Method, identification sur oscillation (OFSM).

```
1  (L,d̂,osc) ← Turbo Décodage EML-MAP(y^s, y^p, y'^p)
2  idx ← Trie selon valeurs decroissantes(osc)
3  Sauvegarde(y^s)
4  j ← 0
5  tant que j < 2 × L et CRCheck(d̂) ==false faire
6      Restauration(y^s)
7      y^s[idx[j]] ← (−1)^(j mod 2) × d_min
8      d̂' ← Turbo Décodage EML-MAP(y^s, y^p, y'^p)
9      j ← j + 1
10     return(d̂)
```

Le comportement des oscillations diffère selon un décodage basé sur l'algorithme EML-MAP ou sur l'algorithme ML-MAP. En effet, l'introduction d'un facteur d'échelle réduit l'amplitude des valeurs de l'information extrinsèque. Ceci permet d'atténuer l'avis du décodeur et donc de réduire son impact sur le décodage à la demi-itération suivante. Ceci a pour effet de réduire les oscillations puisque le poids des information calculées à la demi-itération précédente est lui même réduit. Ainsi, l'algorithme EML-MAP possède de meilleures performances de base que l'algorithme ML-MAP. Cependant, la réduction des oscillations au sein du processus itératif a pour effet de rendre moins discriminante la métrique introduite précédemment.

La Figure 2.12 présente les performances de décodage obtenues en exploitant le principe de décodages successifs selon une métrique basée sur le nombre d'oscillations. Un turbo code du standard LTE est considéré avec des trames de 1504 bits et un rendement 1/3. Chaque turbo décodage peut effectuer jusqu'à 32 itérations. La notation OFSM(L) signifie qu'au plus L positions différentes seront saturées dans les deux directions. Différents algorithmes sont utilisés au sein du processus itératif. Sont considérés les algorithmes :

- EML-MAP,
- SC EML-MAP,
- ML-MAP et
- SC ML-MAP.

Ainsi, il s'agit de l'algorithme max-log-MAP avec ou sans remise à l'échelle et avec ou sans principe Self-Corrected de l'information extrinsèque.

Lorsque l'algorithme EML-MAP est employé, les décodages successifs permettent un léger gain dans la zone de convergence et dans la zone du plancher d'erreurs. Ce gain correspond à un facteur 3 en terme de taux d'erreur. Si maintenant l'algorithme SC EML-MAP est choisi, une nouvelle amélioration dans la zone de convergence apparaît. Celle-ci correspond au gain apporté par le principe Self-Corrected de l'information extrinsèque au premier décodage. Si maintenant la remise à l'échelle n'est plus considérée (algorithme SC

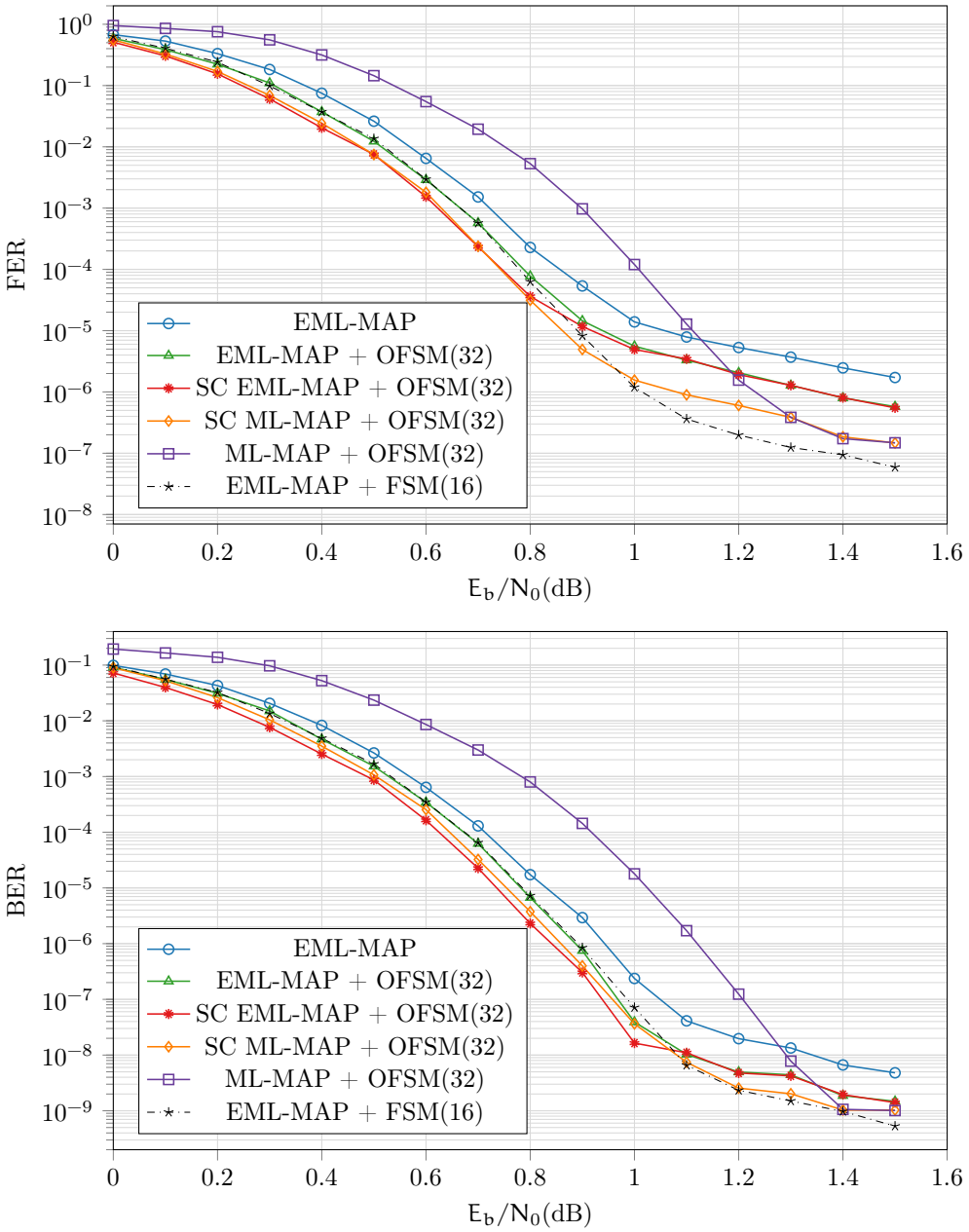


FIGURE 2.12 – Comparaison des performances de décodage entre les algorithme EML-MAP, FSM(16) et différents OFSM(32). Turbo code du standard LTE (K=1504, R=1/3).

ML-MAP), alors les gains sont constants dans la zone de convergence. En revanche, ils augmentent dans la zone du plancher d'erreurs. Ainsi, des gains à la fois dans la zone de convergence et dans le plancher d'erreurs approchent l'ordre de grandeur. Finalement, sans modification de l'information extrinsèque (algorithme ML-MAP), un décalage important sur le seuil de converge apparaît. Celui-ci représente presque 0,2 dB en terme de valeur de SNR. En revanche, cette courbe de performance rejoint celle obtenue avec le principe SC dans la zone du plancher d'erreurs.

Toutefois, les performances obtenues avec l'algorithme FSM(16) sont proches voire meilleures. C'est pourquoi l'approche d'identification par les oscillations seules n'apporte pas de gains suffisants par rapport à l'état de l'art.

2.5 Conclusion

Dans ce deuxième chapitre, une étude des oscillations internes au processus de décodage itératif des turbo codes a été menée. Dans un premier temps, des observations statistiques ont permis d'observer de fortes corrélations entre des oscillations et des erreurs. Cependant, il n'y a pas pour autant d'implication entre ces oscillations et les erreurs. Cette dernière constatation rend délicate une possibilité d'identification de la position des erreurs grâce à l'observation des oscillations.

Cependant, une utilisation de cette observation est proposée afin de corriger l'information extrinsèque échangée au cours du processus itératif. L'algorithme résultant a été proposé pour les turbo codes binaires, aucune adaptation intéressante pour les turbo codes double binaires n'a pu être trouvé. Dans le cas binaire, sous couvert d'un nombre maximal d'itérations important, le seuil de convergence apparaît plus vite qu'avec l'algorithme EML-MAP fonctionnant avec le même nombre d'itérations. Des gains de décodage ont été obtenus dans le cadre du standard CCSDS, représentant un ordre de grandeur en terme de taux d'erreur trame ou 0,1 dB de gain pour la valeur de SNR. Une architecture matérielle a été proposée afin de démontrer le faible surcoût au niveau des complexités calculatoire et mémoire.

Finalement, les oscillations ont été étudiées en tant que métrique permettant l'activation de décodages successifs. Cependant, aucun gain conséquent n'a pu être observé. Dans le chapitre suivant, une proposition d'algorithme se servant de principes connexes à ceux utilisés dans le cadre des décodage successifs est faite. Ce dernière aboutit à un abaissement du plancher d'erreurs sans utiliser de décodage souple supplémentaire.

3 Caractérisation, identification et correction des erreurs résiduelles

Ce chapitre troisième vise à présenter une méthode permettant de corriger les erreurs résiduelles rencontrées dans la zone du plancher d'erreurs lors du décodage des turbo codes.

Suite à une observation de ces erreurs grace à des simulations Monte-Carlo, une prédiction de leurs distributions basée sur la borne de l'union est proposée. Ceci permet alors de les caractériser.

Dans un second temps, plusieurs critères d'identification de la position de ces erreurs sont proposés et comparés. Ceux-ci sont aussi bien adaptés aux turbo codes binaires qu'aux double binaires.

Ceci mène alors à la proposition d'un algorithme permettant de corriger les erreurs résiduelles des turbo codes. Il a pour propriété d'abaisser drastiquement le plancher d'erreurs lors du décodage des turbo codes. Une comparaison avec l'état de l'art met en exergue l'intérêt de cette approche.

3.1 Analyse des erreurs résiduelles

3.1.1 Caractérisation théorique

Comme présenté dans le premier chapitre, l'apparition du plancher d'erreurs dépend de la distribution des mots de codes possédant un faible poids [100]. Plus encore, il a été montré que la zone du plancher d'erreurs émanait de la présence *d'erreurs résiduelles* [70]. La Figure 3.1 présente l'évolution du nombre moyen de bits erronés par trame erronée en fonction de la valeur de SNR pour six turbo codes binaires différents. Cinq de ces codes appartiennent au standard LTE, le cinquième au standard CCSDS. À chaque fois, un rendement de 1/3 et un turbo décodage basé sur l'algorithme EML-MAP itérant 8 fois sont considérés. Dans la zone de convergence, de nombreuses erreurs sont présentes à l'issu du décodage. En revanche, dès lors que le décodeur fonctionne dans la zone du plancher d'erreurs, le nombre moyen d'erreurs binaires par trame erronée est inférieur à 10. De là vient leur nom d'erreurs résiduelles.

Une caractérisation de la distribution de ces erreurs résiduelles a été proposée dans [101]. Cette dernière est basée sur les fonctions recenseuses de poids (Weight Enumerator Functions ou WEF en anglais, [22]). En posant $P^{\langle ML \rangle}(m)$ la probabilité que m bits soient erronés dans une trame et sachant que la trame est erronée après un décodage à maximum de vraisemblance, l'expression suivante est obtenue :

$$P^{\langle ML \rangle}(m) \simeq \frac{W^m \, A_m^{\langle CP \rangle}(Z)|_{W=Z=e^{-RE_b/N_0}}}{\sum_{w=1}^{N} W^w \, A_w^{\langle CP \rangle}(Z)|_{W=Z=e^{-RE_b/N_0}}} \tag{3.1}$$

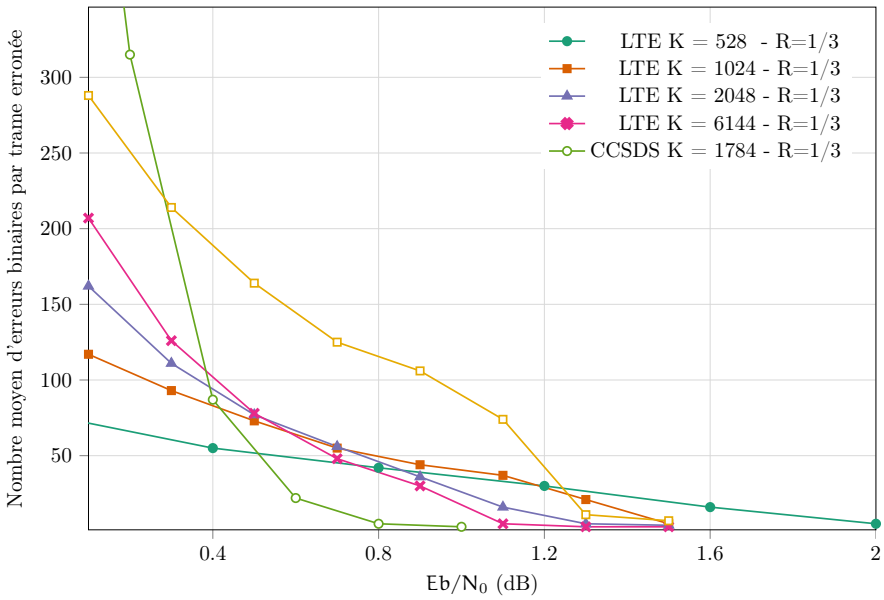


FIGURE 3.1 – Évolution du nombre moyen d'erreurs binaires par trame erronée pour différentes valeurs de SNR et différents turbo codes. Décodage effectué par l'algorithme EML-MAP itérant 8 fois.

où $A_w^{\langle CP \rangle}(Z)$ est le WEF conditionnel du turbo code considéré [90].

En supposant que les séquences d'information générant un mot de code de poids w ont majoritairement le même poids, l'équation précédente peut alors être transformée en :

$$P^{\langle ML \rangle}(m) \approx \frac{\sum\limits_{w, \frac{W_w}{A_w}=m} A_w \cdot \exp\left(-wR\frac{E_b}{N_0}\right)}{\sum\limits_{w} A_w \cdot \exp\left(-wR\frac{E_b}{N_0}\right)} \tag{3.2}$$

avec, comme déjà introduit dans le premier chapitre, A_w le nombre de mots de code de poids de Hamming w. La multiplicité des bits d'information W_w est la somme des poids de Hamming des A_w séquences binaires générant des mots de code de poids de Hamming w. Au prix d'une faible imprécision, cette forme de l'équation permet d'utiliser directement le spectre de distances d'un turbo code pour estimer le nombre de bits erronés dans le plancher d'erreurs. Les résultats obtenus avec cette équation diffèrent alors légèrement de ceux obtenus avec l'Équation 3.1. En effet, certaines valeurs sont masquées par l'opération de moyenne : $\frac{W_w}{A_w} = m$.

Ainsi, le spectre de distances d'un turbo code permet de calculer la probabilité d'obtenir m erreurs binaires dans une trame erronée en considérant un décodage ML pour un taux d'erreur suffisamment faible et un canal AWGN. De part la présence du terme en exponentielle, plus le poids du mot de code w est distant de d_{min}, moins la multiplicité associée a un impact sur la distribution des erreurs résiduelles. Ceci peut néanmoins être fortement atténué par une multiplicité très importante (de l'ordre du millier, rencontré fréquemment lors des expérimentations).

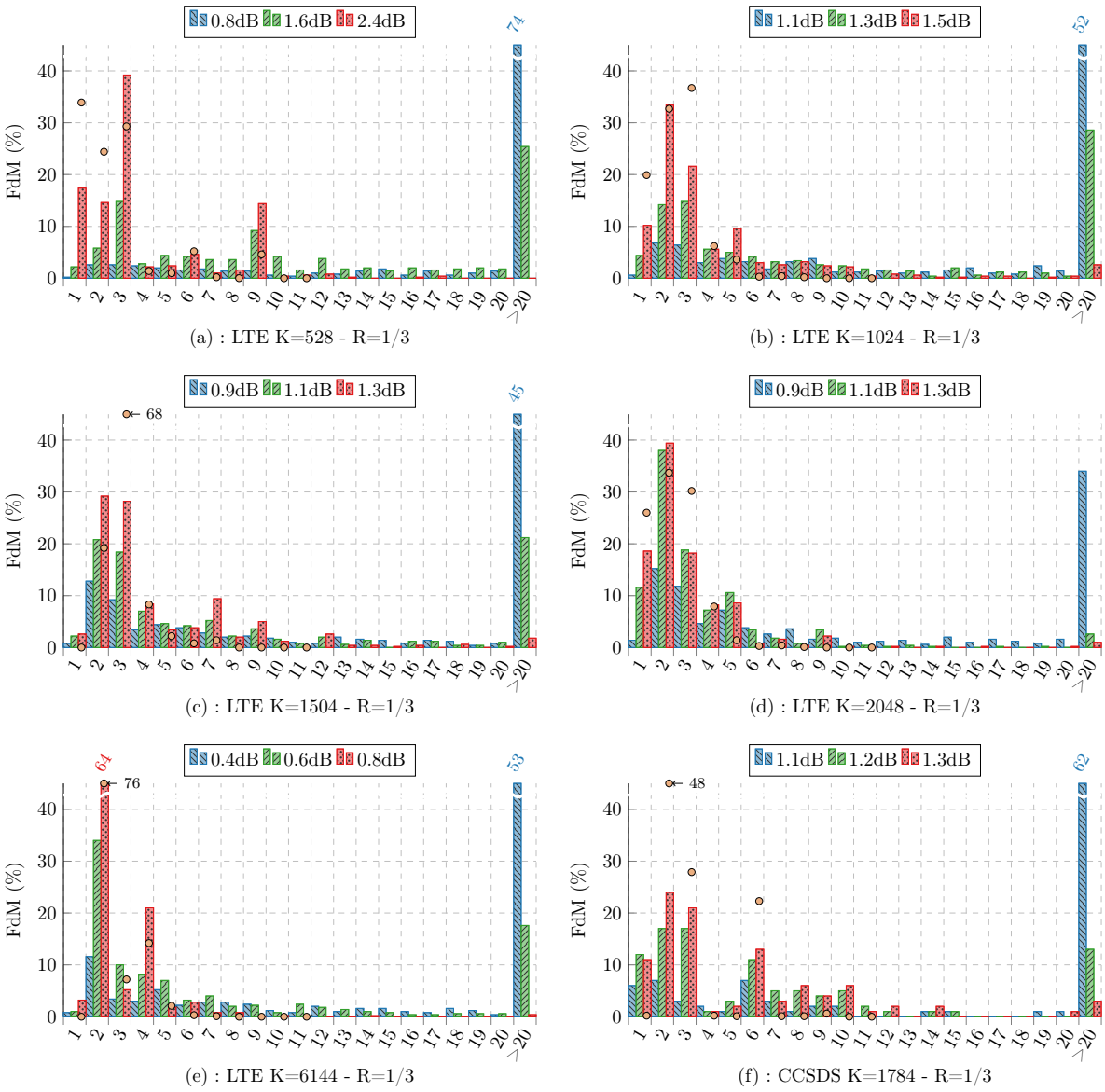
Dans la section suivante, ces résultats théoriques sont comparés à des distributions observées du nombre d'erreurs binaires dans les trames erronées de la zone du plancher d'erreurs avec un décodage basé sur l'algorithme ELM-MAP.

3.1.2 Observations considérant des turbo codes standardisés

Une méthode proposant de prédire théoriquement la distribution des erreurs résiduelles dans la zone du plancher d'erreurs a été établie. L'obtention de la distribution réelle de ces erreurs est maintenant réalisée et elle est comparée avec cette méthode. Dans un premier temps, des turbo codes binaires de tailles de trames différentes et de rendement 1/3 sont considérés. Dans un second temps, des turbo codes double binaires de deux tailles de trame mais de différents rendements sont considérés. Ainsi, un large spectre de paramètres est exploité sans présenter un nombre trop conséquent de données.

3.1.2.1 Dans le cas de turbo codes binaires

La Figure 3.2 présente la distribution des erreurs résiduelles pour 5 turbo codes du standard LTE et un du standard CCSDS. Le rendement du code est fixé à 1/3. Pour



(a) : LTE K=528 - R=1/3

(b) : LTE K=1024 - R=1/3

(c) : LTE K=1504 - R=1/3

(d) : LTE K=2048 - R=1/3

(e) : LTE K=6144 - R=1/3

(f) : CCSDS K=1784 - R=1/3

FIGURE 3.2 – Distribution du nombre d'erreurs binaires pour différentes valeurs de SNR et différents turbo codes des standards LTE et CCSDS. Décodage EML-MAP itérant 8 fois.

chacun de ces histogrammes, 500 trames erronées après décodage via l'algorithme EML-MAP itérant 8 fois sont considérées. Pour chacun des turbo codes, trois valeurs de SNR sont présentées. La première correspond à un fonctionnement du turbo code dans la zone de convergence. La seconde valeur correspond au point d'inflexion de la courbe de performance et la troisième à la zone du plancher d'erreurs. Conformément aux résultats de la Figure 3.1, lorsque la valeur de SNR augmente, la distribution des erreurs se concentre sur de faibles nombres d'erreurs binaires. Il apparaît alors que pour la troisième valeur de SNR, la très grande majorité des trames erronées contiennent moins de 10 erreurs binaires. Plus encore, hormis pour le turbo code du standard CCSDS, 70% des

TABLEAU 3.1 – Distribution théorique des erreurs dans le plancher d'erreurs selon l'équation 3.1 pour différents turbo codes standardisés

	1	2	3	4	5	6	7	8	9	10	>10
LTE R=1/3, K=528 @ 2,4 dB	33,9%	24,4%	29,3%	1,4%	1%	5,2%	0,2%	0,02%	4,6%	0%	0%
LTE R=1/3, K=1024 @ 1,5 dB	19,9%	32,7%	36,7%	6,2%	3,6%	0,3%	0,4%	0,2%	0%	0%	0%
LTE R=1/3, K=1504 @ 1,3 dB	0%	19,2%	68%	8,3%	2,2%	0,8%	1,4%	0%	0%	0%	0%
LTE R=1/3, K=2048 @ 1,3 dB	26%	33,7%	30,2%	7,9%	1,4%	0,3%	0,4%	0,1%	0%	0%	0%
LTE R=1/3, K=6144 @ 0,8 dB	0%	76%	7,2%	14,2%	2,1%	0,3%	0,1%	0%	0%	0%	0%
CCSDS R=1/3, K=1784 @ 1,3 dB	0,2%	48,1%	27,9%	0,2%	0,1%	22,3%	0,1%	0,1%	0,6%	0%	0%

trames erronées ont strictement moins de 5 erreurs.

Sont aussi présentés sur la Figure 3.2 les résultats théoriques obtenus pour la valeur de SNR la plus grande grâce à l'équation 3.1 et au spectre de distances calculé via la Double Impulse Method (cf. 1.2.5.2). Les spectres de distances de ces différents turbo codes sont présentées en Annexe C. Les distributions théoriques calculées sont reportées pour plus de lisibilité dans le Tableau 3.1. Suivant le turbo code considéré, des différences entre la valeur théorique et la valeur mesurée apparaissent. Néanmoins, la tendance pour les valeurs calculées est conforme aux mesures obtenues. Ainsi, la seule occurrence non prédite correspond à un seul bit erroné pour le standard CCSDS, où 0,2% était prédit alors que 11% sont mesurés. Cette observation trouve un explication dans la sous-optimalité du décodage itératif des turbo codes, discuté plus loin.

Afin de présenter la déviation entre la théorie développée dans la première section de ce Chapitre et les mesures effectuées, un calcul d'erreur quadratique moyenne (EQM, Équation 3.3) est effectué pour chacun des turbo codes. Les valeurs obtenues sont récapitulées dans le tableau 3.2.

$$\mathsf{EQM} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(T_i - M_i\right)^2} \quad \text{avec} \quad \begin{cases} T_i & \text{la valeur théorique} \\ M_i & \text{l'observation} \end{cases} \tag{3.3}$$

Les valeurs d'EQM sont alors comprises entre 3,8 et 9,5, impliquant que les valeurs théoriques et mesurées sont relativement proches. Ainsi même si des écarts apparaissent, ces équations permettent de prédire l'allure de la distribution des erreurs binaires dans la zone du plancher d'erreurs. Les écarts s'expliquent par le fait que les Équations 3.1 et 3.2 ne sont vraies que dans le cas d'un décodage ML. Or ce dernier ne peut pas être effectué dans le contexte des turbo codes en raison de la complexité calculatoire. Il semble alors apparaître que le décodage EML-MAP entraîne un étalement des valeurs vis-à-vis de celles théoriques n'étant vraies que pour un décodage ML.

Plus le ratio $\frac{W_d}{A_d}$ des premiers termes du spectre de distances du code est faible, plus le nombre moyen de bits erronés par trame erronée dans le plancher d'erreurs sera faible lui aussi. Pour les turbo codes standardisés, les ratios $\frac{W_d}{A_d}$ sont habituellement faibles. Ceci implique alors une distribution des erreurs binaires dans le plancher d'erreurs centrée sur les faibles nombres d'erreurs.

TABLEAU 3.2 – Erreur quadratique moyenne entre les valeurs théoriques et les simulations Monte-Carlo

	EQM
LTE R=1/3, K=528 @ 2,4 dB	6,2
LTE R=1/3, K=1024 @ 1,5 dB	9,4
LTE R=1/3, K=1504 @ 1,3 dB	9,5
LTE R=1/3, K=2048 @ 1,3 dB	3,8
LTE R=1/3, K=6144 @ 0,8 dB	6,7
CCSDS R=1/3, K=1784 @ 1,3 dB	6,9

Maintenant, une analyse similaire vient confirmer et étendre ces résultats dans le contexte de turbo codes double binaires.

3.1.2.2 Dans le cas de turbo codes double binaires

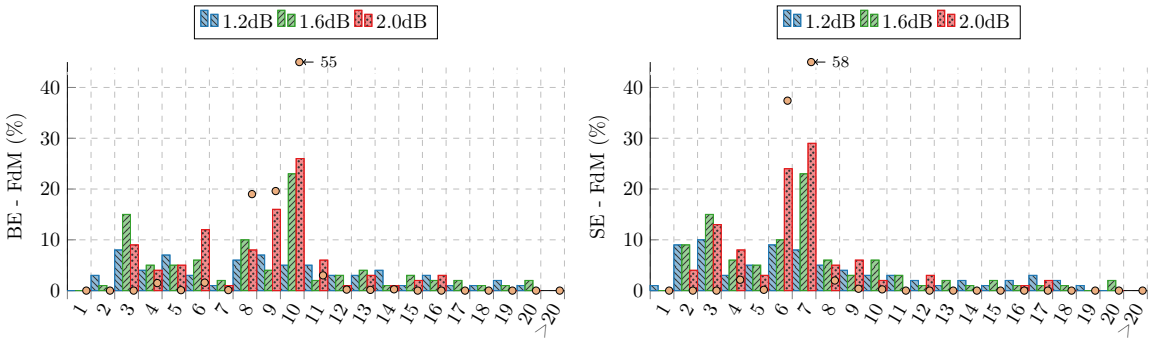
Dans le contexte de turbo codes double binaires, deux analyses complémentaires sont nécessaires. En effet, en plus du nombre d'erreurs binaires (noté BE), le nombre d'erreurs symboles (noté SE) peut être aussi comptabilisé. Par définition, le nombre d'erreurs symboles est compris entre $\frac{BE}{2}$ et BE. En effet, si un bit d'un symbole est erroné, alors le symbole est lui même erroné. Mais, un symbole erroné se compose d'un ou deux bits erronés.

Les méthodes d'obtention du spectre de distances telles que la Double Impulse Method permettent de calculer la multiplicité des bits d'information (W). Par extension, il est aussi possible de calculer la multiplicité des symboles d'information (W^S). En considérant l'équation 3.2, la probabilité que m symboles soient erronés dans une trame sachant que la trame est erronée après un décodage ML a pour expression :

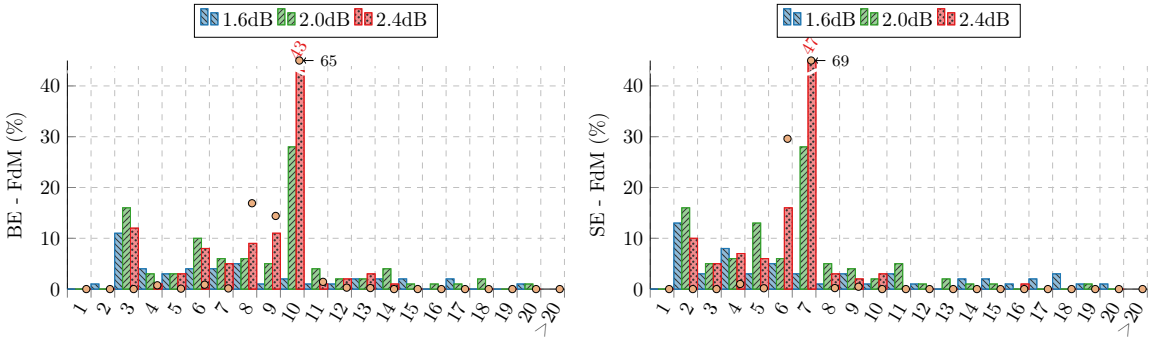
$$P_S^{\langle ML \rangle}(m) \approx \frac{\sum\limits_{w, \frac{W_w^S}{A_w}=m} A_w \cdot \exp\left(-wR\frac{E_b}{N_0}\right)}{\sum\limits_{w} A_w \cdot \exp\left(-wR\frac{E_b}{N_0}\right)} \tag{3.4}$$

La Figure 3.3 présente la distribution des erreur binaires et symboles pour différents turbo codes du standard DVB-RCS avec K=440 symboles et différents rendements allant de R=1/3 à R=6/7. Le processus de décodage est réalisé par l'algorithme EML-MAP itérant 8 fois où le facteur de remise à l'échelle vaut 0,5 pour les deux premières itérations et 0,85 au cours des itérations suivantes. 100 trames erronées après décodage sont considérées pour les expérimentations. Comme dans le cas de turbo codes binaires, il apparaît que la très grande majorité des trames contiennent au plus 10 erreurs binaires. En ce qui concerne le nombre d'erreurs symboles, la distribution est semblable mais décalée vers la gauche de 0 à 3 erreurs suivant le turbo code considéré. Ceci est conforme avec le fait qu'une erreur symbole corresponde à 1 ou 2 erreurs binaires.

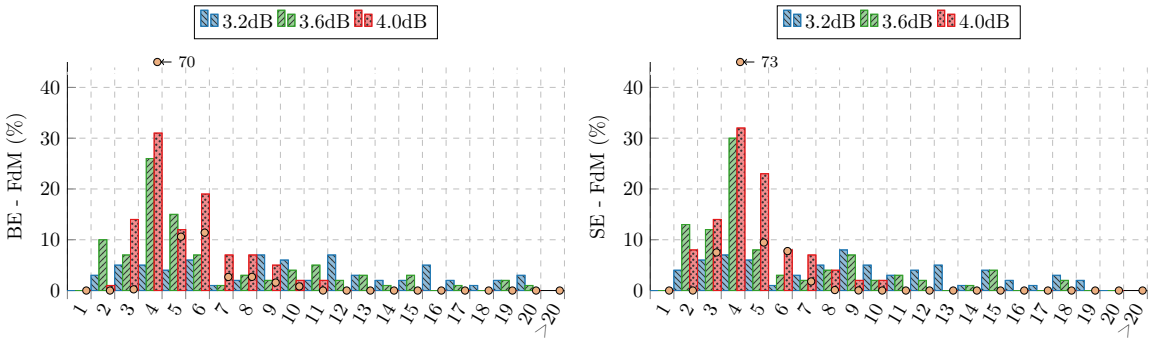
Afin de comparer ces données expérimentales avec des valeurs théoriques, l'équation 3.4 est utilisée avec les spectres de distances obtenus grâce à la méthode DIM pour les



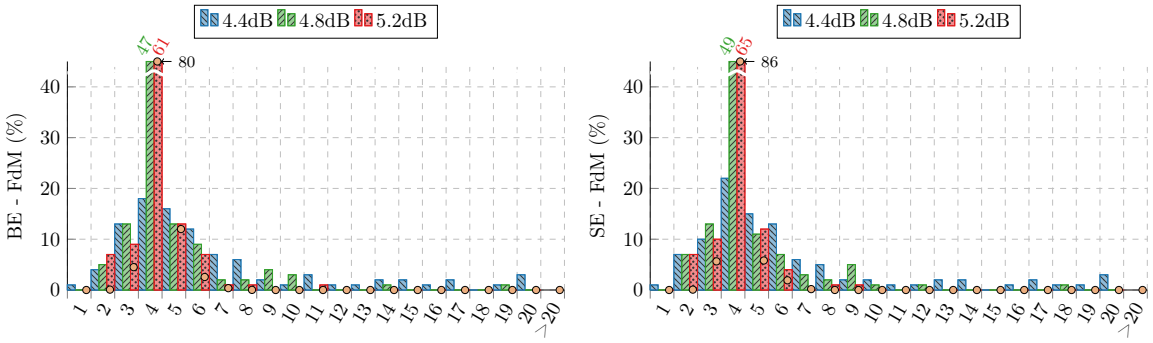
(a) : DVB-RCS - K=440 R=1/3



(b) : DVB-RCS - K=440 R=1/2



(c) : DVB-RCS - K=440 R=3/4



(d) : DVB-RCS - K=440 R=6/7

FIGURE 3.3 – Distribution du nombre d'erreurs binaires et symboles pour différentes valeurs de SNR et pour différents turbo codes du standard DVB-RCS pour K=440. Décodage EML-MAP itérant 8 fois.

différents turbo codes. Ces spectres de distances sont présentés en Annexe C. Les résultats théoriques sont calculés en considérant la valeur de SNR la plus élevée considérée pour les simulations et sont reportés sur la Figure 3.3. Il apparaît alors, qu'à nouveau, les équations 3.2 et 3.4 permettent de prédire la distribution des erreurs dans la zone du plancher d'erreurs de manière fiable. Nous pouvons constater que toutes les valeurs d'occurrence maximale sont correctement prédites que ce soit pour les erreurs binaires ou les erreurs symboles. Des écarts apparaissent pour les faibles nombres d'erreurs. Ceci s'explique là encore par la validité des différents équations dans le cas d'un décodage ML, non réaliste pour des turbo codes de tailles conséquentes. Ainsi, dans le cas d'un décodage EML-MAP, la distribution observée des erreurs est plus étalée que celle prédite.

La même étude a été menée dans le cas des turbo codes du standard DVB-RCS avec K=752. Les résultats numériques ainsi que leurs spectres de distances sont fournis en Annexe C pour des rendements valant 1/3, 1/2, 3/4 et 6/7. Ces résultats complémentaires confirment ceux détaillés dans cette sous-section. Ainsi, la prédiction théorique de la distribution des erreurs binaires ou symboles est valide ce quelque soit le turbo code considéré. Il peut être binaire ou double binaire, de petite ou grande taille de trame et à faible ou fort rendement.

Finalement, lorsqu'un processus de décodage itératif échoue au niveau du plancher d'erreurs, la trame résultante est très proche du mot de code transmis. Cette constatation semble se réaliser pour tout turbo code standardisé, incluant les turbo codes double binaires. Dans la section suivante, différents critères d'identification de ces erreurs résiduelles sont proposés, évalués et comparés.

3.2 Comparaison de critères d'identification

Dans cette section, une comparaison de plusieurs métriques permettant l'identification des positions des erreurs résiduelles est proposée. Ces différentes métriques sont basées sur des quantités internes du turbo décodeur qui seront détaillées ci-après.
Suite aux résultats obtenus dans le deuxième chapitre des métriques reposant sur des oscillations sont écartées. Ainsi, seules des métriques prenant en compte l'amplitude d'informations internes au turbo décodeurs sont privilégiées.

3.2.1 Les différentes métriques considérées pour les turbo codes binaires

Comme nous l'avons vu dans le chapitre précédent, le module de l'information *a posteriori* calculée par le turbo décodeur représente le niveau de confiance associé à chaque bit du mot décodé. Ainsi, une première métrique pouvant être établie est la suivante :

$$\Delta_1(k) = |L^a(k)| \text{ , avec } k \in [\![0;\ K]\!] \tag{3.5}$$

Par ailleurs, décorréler les informations du canal de la métrique peut s'avérer intéressant. En effet, cela permet d'exprimer seulement l'avis propre au décodeur. Cette métrique est alors uniquement basée sur les informations extrinsèques et a pour expression :

$$\Delta_2(k) = |L^e_{12}(k) + L^e_{21}(k)| \tag{3.6}$$

D'autre part, il peut être judicieux de considérer une norme au sens mathématique. L'évaluation de la norme 1 (distance de Manhattan) est suffisante puisque son emploi vise à ordonner des positions. En effet, l'ordre est conservé quelque soit la norme p ($p \in \mathbb{N}$) employée. Les expressions des deux équations précédentes deviennent alors respectivement :

$$\Delta_3(k) = |y^s(k)| + |L^e_{12}(k)| + |L^e_{21}(k)| \tag{3.7}$$

et

$$\Delta_4(k) = |L^e_{12}(k)| + |L^e_{21}(k)| \tag{3.8}$$

Pour l'ensemble de ces métriques, les positions les moins fiables sont extraites en identifiant les valeurs k minimisant ces métriques.

3.2.1.1 Résultats d'identification

Afin de comparer ces différentes métriques, des statistiques d'identification vont être exploitées. Le protocole expérimental est le suivant. Après chaque itération du processus de turbo décodage, les q positions les moins fiables dans la trame sont identifiées grâce à chacune de ces métriques. Si l'ensemble des erreurs est présent dans ces q éléments, alors l'identification est considérée comme réussie. Les turbo codes considérés sont des

turbo codes binaires du standard LTE. Plusieurs valeurs de SNR sont successivement étudiées. Afin d'obtenir des données valides, 500 trames erronées après un décodage par l'algorithme EML-MAP itérant 8 fois sont observées.

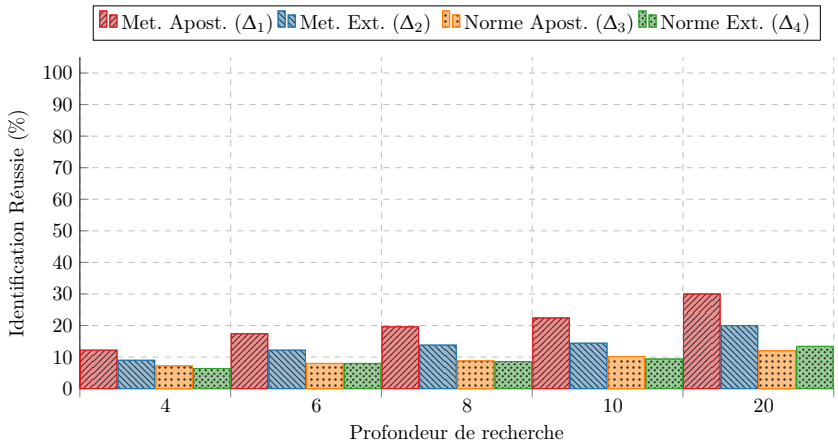
La Figure 3.4 présente les résultats pour le turbo code LTE avec K=1024 et R=1/3. Le nombre de positions les moins fiables considérées $\mathfrak{q}$ s'étalent de 4 à 20. Chaque sous-figure correspond à une valeur de SNR différente. Il apparaît tout d'abord que les deux critères d'identification basés sur une norme (Δ_3 et Δ_4) présentent les moins bonnes performances. Ceci peut être interprété en remarquant que les normes ne permettent pas d'identifier les cas où chaque décodeur possède un avis ferme (forte amplitude) mais différent l'un de l'autre (signes opposés). Les deux autres métriques identifient en revanche cette configuration. La même constatation est faite pour d'autres turbo codes. Ainsi, l'utilisation d'une norme (au sens mathématique) est écartée dans la suite, ce au profit des métriques Δ_1 et Δ_2.

Ces deux dernières possèdent des performances proches avec un avantage pour la métrique Δ_1. Il est à noter que le pouvoir d'identification de ces deux métriques augmente lorsque la valeur de SNR augmente. Ceci est corrélé avec la diminution du nombre moyen d'erreurs binaires par trame constatée dans la section précédente.

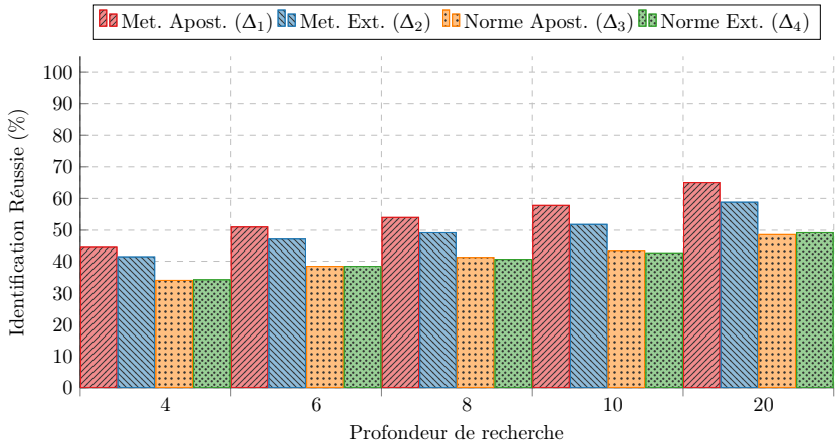
Toujours en considérant la Figure 3.4, selon la profondeur de recherche employée et pour une valeur de SNR de 1,1 dB, entre 10 et 30% des trames ont l'ensemble de leurs erreurs binaires identifiées. Ces taux ne permettent pas de proposer une correction intéressante. En effet, en supposant que ces 30% de trames dont les erreurs sont identifiées soient effectivement corrigées par un mécanisme basé sur cette métrique, le taux d'erreur trame passerait alors de $2{,}3 \times 10^{-4}$ à $1{,}6 \times 10^{-4}$. L'amélioration des performances serait alors toute relative.

En considérant maintenant une valeur de SNR de 1,3 dB, le taux d'identification maximal atteint 65% pour $\mathfrak{n} = 20$ avec la métrique Δ_1. Le taux d'erreur trame résultant passerait alors de $1{,}5 \times 10^{-5}$ à $5{,}2 \times 10^{-6}$. Cette fois, des gains substantiels sont possibles. D'après les données de la Figure 3.2, dans ce même cas expérimental, 71% des trames ont 20 erreurs ou moins. L'emploi de la métrique Δ_1 semble alors permettre d'identifier l'ensemble des erreurs de 91% des trames ayant moins de 20 erreurs binaires. Ce taux s'établit à 83% pour la métrique Δ_2. Il apparaît donc que seules de rares trames à faible nombre d'erreurs binaires n'ont pas l'ensemble de leurs erreurs identifiées. Il est attendu que plus la valeur de SNR augmente, plus le nombre d'erreurs binaires par trame erronée baisse. Par conséquent, les métriques identifient encore plus d'erreurs.

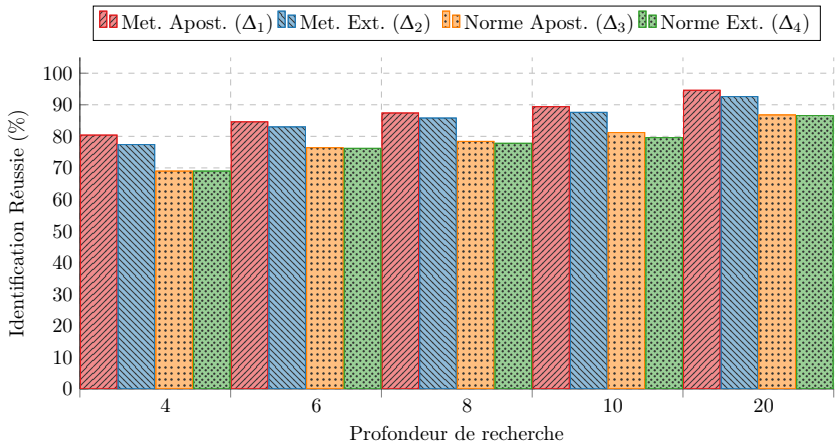
Ceci est vérifié, pour une valeur de SNR de 1,5 dB. Avec une profondeur de recherche fixée à 10, 87,6% des trames erronées ont l'intégralité de leurs erreurs identifiées grâce à la métrique Δ_2. Ce taux atteint 89,4% avec la métrique Δ_1. Pour cette valeur de SNR, 93,8% des trames possède 10 erreurs ou moins. Ainsi, les métriques permettent une identification quasiment parfaite de la position des erreurs résiduelles. Finalement, si chaque trame erronée dont la totalité des erreurs sont identifiées est corrigée, le taux d'erreur trame résultant passe de $3{,}6 \times 10^{-6}$ à une valeur comprise entre $3{,}8 \times 10^{-7}$ et $4{,}4 \times 10^{-7}$ suivant la métrique Δ_1 ou Δ_2 employée.



(a) : LTE K=1024, R=1/3, SNR=1.1dB (FER=2.3×10^{-4})



(b) : LTE K=1024, R=1/3, SNR=1.3dB (FER=1.5×10^{-5})



(c) : LTE K=1024, R=1/3, SNR=1.5dB (FER=3.6×10^{-6})

FIGURE 3.4 – Pourcentage d'identification réussie des erreurs résiduelles pour le turbo code du standard LTE K=1024, R=1/3. Décodage EML-MAP itérant 8 fois.

La Figure 3.5 présente les statistiques d'identification réussies pour trois autres turbo codes du standard LTE. Dans les trois cas considérés, les observations faites lors de l'analyse de la Figure 3.4 sont vérifiées. Ces histogrammes valident donc la pertinence des métriques Δ_1 et Δ_2 pour l'identification des bits erronés lors du processus itératif de décodage. Dans tous les cas, pour des valeurs de SNR correspondant à un fonctionnement des turbo décodeurs dans la zone du plancher d'erreurs, au moins 90% des trames erronées ont l'ensemble de leurs erreurs identifiées par la métrique Δ_1 pour un profondeur de recherche de taille 6.

Dans la section suivante, les métriques proposées sont étendues et caractérisées pour des turbo codes double binaires.

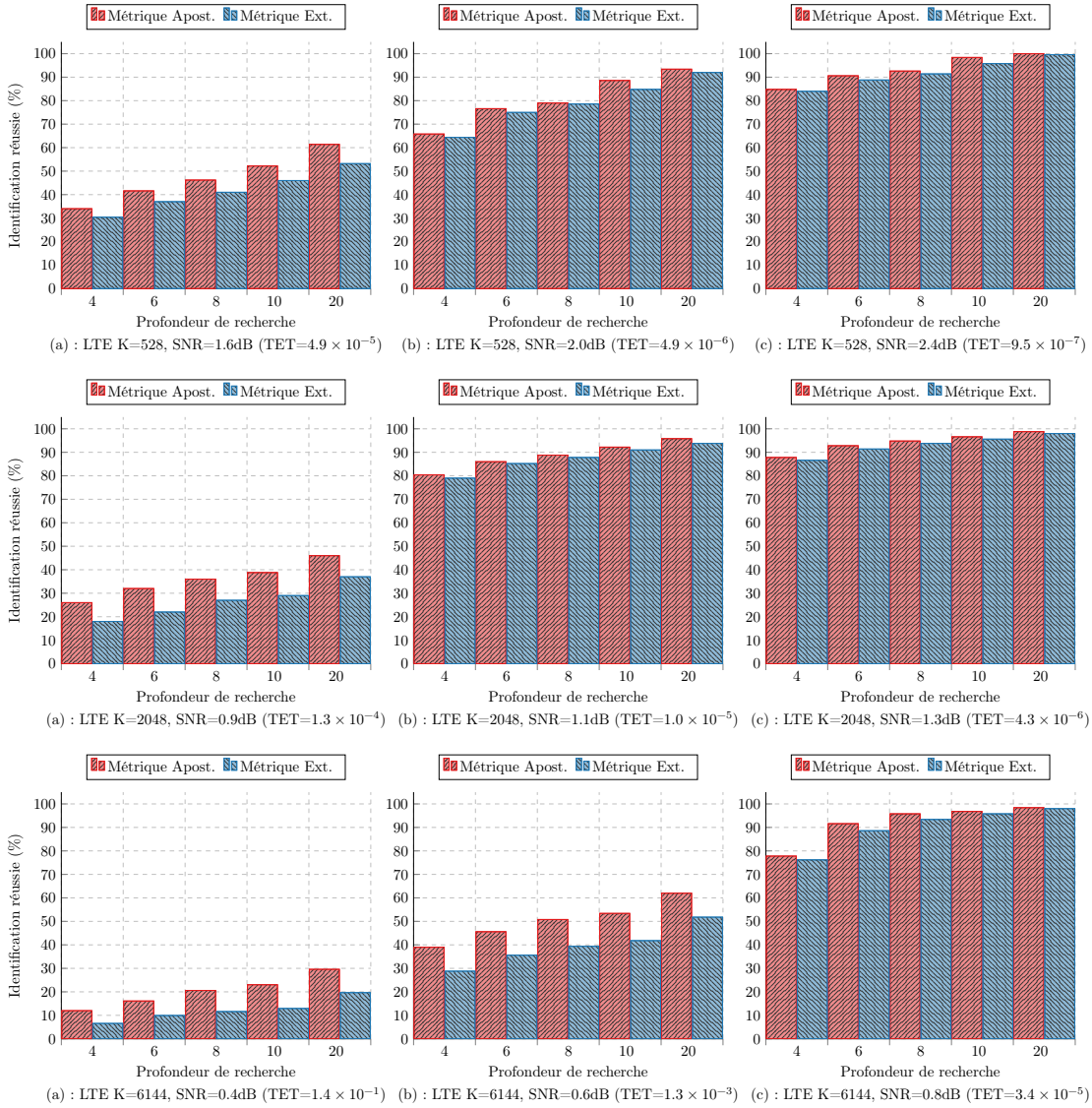


FIGURE 3.5 – Pourcentage d'identification réussie des erreurs résiduelles pour différents turbo codes du standard LTE K=528, K=2048 et K=6144 avec R=1/3. Décodage EML-MAP itérant 8 fois.

3.2.2 Extension des critères d'identification aux turbo codes double binaires

Le décodage des turbo codes double binaires a été détaillé en section 1.2.3. Pour rappel, ce décodage est basé sur l'échange de 4 valeurs extrinsèques par symbole. En réalité, dans un but de simplification de l'architecture matérielle associée, seules trois valeurs sont échangées puisqu'elles sont normalisées vis-à-vis de la probabilité d'obtenir le symbole **0**.

La généralisation des métriques Δ_1 et Δ_2 au cas double binaire n'est pas triviale. Dans la suite, une extension permettant d'atteindre une identification au moins aussi bonne que celle présentée précédemment est décrite. L'équation de la métrique Δ_1 dans le cas binaire peut être exprimée de la sorte :

$$\begin{aligned}\Delta_1(k) &= |L^a(k)| \\ &= |l_0^a(k) - l_1^a(k)|,\end{aligned}$$

où $l_b^a(k)$ représente la log-vraisemblance (LL) *a posteriori* que le bit considéré prenne la valeur b, c'est-à-dire, $l_b^a(k) = \log\left(P(\hat{b_k} = b)\right)$. Ceci peut alors être interprété comme la distance entre les deux LLs de chaque symbole binaire possible (0 et 1). L'extension directe au cas double binaire devrait alors considérer le calcul de 6 distances analogues puisque le nombre de symboles est porté à 4 (**0**,**1**,**2**, et **3**). Une combinaison de ces 6 distances serait alors nécessaire pour caractériser chaque symbole avec une valeur scalaire. Cette valeur refléterait la fiabilité du symbole considérée. Cependant, de tels calculs s'avèrent complexes à entreprendre. Nous proposons donc de considérer uniquement les deux symboles les plus probables S_{M_x} et S_{M_y} qui maximisent l_s^a :

$$S_{M_x}(k) = \underset{s\in[\![0;3]\!]}{\arg\max}\,(l_s^a(k)) \qquad \text{et} \qquad S_{M_y}(k) = \underset{s\in[\![0;3]\!]\setminus S_{M_x}(k)}{\arg\max}\,(l_s^a(k))$$

La métrique a alors pour expression :

$$\Delta_1'(k) = l^a_{S_{M_x}}(k) - l^a_{S_{M_y}}(k) \tag{3.9}$$

À l'aide d'un raisonnement similaire, la métrique analogue à Δ_2 a pour expression :

$$\Delta_2'(k) = l^{e,\mathrm{sum}}_{S_{M_x}}(k) - l^{e,\mathrm{sum}}_{S_{M_y}}(k), \tag{3.10}$$

avec

$$S_{M_x}(k) = \underset{s\in[\![0;3]\!]}{\arg\max}\,(l_s^{e,\mathrm{sum}}(k)) \qquad \text{et} \qquad S_{M_y}(k) = \underset{s\in[\![0;3]\!]\setminus S_{M_x}(k)}{\arg\max}\,(l_s^{e,\mathrm{sum}}(k))$$

et $l_s^{e,\mathrm{sum}}(k) = \sum_{p=1}^{2} l_{s,p}^e(k)$.

Ces métriques peuvent s'interpréter comme suit. Pour une position donnée k, si les LLs des deux symboles les plus probables sont proches l'un de l'autre, la position k est considérée comme étant moins fiable qu'une position pour laquelle cette distance serait plus grande.

3.2.2.1 Résultats d'identification

Le même protocole expérimental que celui décrit dans le cas de turbo codes binaires est exploité. Cependant, des turbo codes du standard DVB-RCS sont maintenant considérés. Le décodage utilise l'algorithme EML-MAP itérant 8 fois. Le facteur de remise à l'échelle vaut 0,5 pour les deux premières itérations et 0,85 pour les itérations suivantes. Profitant de la circularité du treillis, un processus de décodage par passage de message est employé. Les tailles de trame considérées sont K=440 et K=752 symboles d'informations. Les turbo codes choisis ont des rendements valant 1/3, 3/4 et 6/7. Ceci permet de couvrir un spectre relativement exhaustif des turbo codes double binaires à 8 états standardisés.

La Figure 3.6 présente les statistiques d'identification pour les cas K=752. Les résultats pour K=440 sont déportés en Annexe C pour une meilleure lisibilité du document. Cette Figure est divisée en six sous-figures. La première ligne correspond aux cas R=1/3. Les mêmes propriétés d'identification que celles présentées dans le cas binaire apparaissent. Au fur et à mesure que la valeur du SNR croît, le taux identification augmente. La métrique basée sur l'information *a posteriori* Δ_1' présente de meilleures statistiques d'identification que celles basées uniquement sur les informations extrinsèques Δ_2'. Cet écart diminue avec l'augmentation du SNR. Finalement, pour une profondeur de recherche de 20 et pour une valeur de SNR de 2,2 dB, quelque soit la métrique considérée, plus de 95% des trames erronées ont l'ensemble de leurs erreurs résiduelles identifiées.

La deuxième ligne d'histogrammes présente quant à elle ces statistiques pour un rendement de 3/4. Dans ce cas, la métrique basée uniquement sur les informations extrinsèques Δ_2' possède de moins bonnes performances que celle basée sur les informations *a posteriori* Δ_1'. La même constatation est réalisée pour un rendement de 6/7. Ainsi, plus le rendement augmente, moins la métrique basée sur les informations extrinsèques s'avère pertinente. Cette constatation est amplifiée pour de faibles valeurs de SNR. Ainsi, dans tous les cas, la métrique basée sur les information *a posteriori* Δ_1' possède de bonnes performances d'identification des erreurs résiduelles. Il est à noter qu'un taux d'identification réussie de plus de 90% est atteint dans la zone du plancher d'erreurs en considérant une profondeur de recherche de 20.

En reprenant le calcul permettant d'obtenir l'information extrinsèque, il apparaît que cette dernière dépend essentiellement de l'information de parité. Or, les forts rendements sont obtenus par poinçonnage de l'information de parité. Ceci est la cause probable de de la pertinence la métrique Δ_1' vis-à-vis de la métrique Δ_2' lorsque le rendement augmente. En effet, comme le nombre de symboles de parité diminue avec l'augmentation du rendement, la métrique Δ_2' est porteuse de moins d'information que la métrique Δ_1'. Cette constatation doit normalement se transposer aux cas de turbo codes binaires.

3.2.3 Conclusions sur les métriques d'identification

Faisant suite à la mise en exergue de l'existence d'erreurs résiduelles dans la région du plancher d'erreurs des turbo codes, différentes métriques permettant l'identification de ces erreurs ont été étudiées. Une comparaison des performances de ces différentes métriques a

FIGURE 3.6 – Pourcentage d'identification réussie des erreurs résiduelles pour différents turbo codes du standard DVB-RCS K=752, R=1/3, 3/4 et 6/7. Décodage EML-MAP itérant 8 fois.

permis de converger vers une métrique, nommée Δ_1 (et Δ_1' pour son homologue adaptée aux turbo codes double binaires). En considérant la valeur absolue de l'information *a posteriori*, cette métrique permet d'identifier la totalité des erreurs résiduelles d'une trame erronée dans 90% des cas dans la zone du plancher d'erreurs pour une profondeur de recherche raisonnable, ce quelque soit le turbo code standardisé considéré. Ce taux devrait permettre d'observer des gains de l'ordre d'une décade en considérant un algorithme capable d'exploiter cette métrique. La section suivante a justement pour objet la proposition d'un tel algorithme de décodage.

3.3 L'algorithme de décodage Flip and Check

Cette section présente un algorithme permettant de tirer parti de la métrique Δ_1 sélectionnée. Dans un premier temps, le principe de cet algorithme est détaillé. Dans un second temps, ses performances sont évaluées pour différents turbo codes standardisés.

3.3.1 Principe général de l'algorithme Flip and Check

Afin de faciliter la compréhension de l'algorithme présenté ci après, ses concepts clefs sont maintenant introduits. L'algorithme décrit repose sur trois actions successives et complémentaires. Tout d'abord, les positions les moins fiables dans la trame décodée sont identifiées grâce à la métrique Δ_1 présentée dans la section précédente. Ensuite, différents mots candidats sont générés à partir de ces positions. Finalement, grâce à un code détecteur d'erreurs le bon mot de code est identifié.

Dans un grand nombre de standards de communications numériques, le turbo code est concaténé en série avec un code CRC. En effet, à la différence d'un code LDPC, un turbo code ne peut détecter si la trame décidée correspond à un mot de code. C'est pourquoi l'algorithme décrit ci-après va exploiter conjointement les capacités de détection d'un code CRC et la métrique d'identification proposée.

L'utilisation d'un code CRC implique une baisse de rendement du schéma de codage due à l'ajout d'information de redondance. Ceci résulte alors en un décalage de la convergence de décodage. Ce décalage peut se calculer et vaut $10 \times \log_{10}\left(\frac{\mathsf{K}}{\mathsf{K}-\mathrm{Taille}_{\mathrm{CRC}}}\right)$ dB. Ainsi, bien évidemment, plus la taille du CRC est importante, plus ce décalage de convergence est important. De même, pour une taille fixe du code CRC, plus la taille de la trame est faible, plus l'impact du code CRC sur le rendement est important. Dans les standards de communications numériques utilisant des turbo codes, la taille du code CRC varie de 16 bits à 32 bits. Ainsi, il est d'une taille de 16 bits pour les standards CCSDS et DVB-RCS, d'une taille de 24 bits pour le standard LTE et d'une taille de 32 bits pour le standard DVB-RCS2. Aussi, suivant la taille de la trame considérée, le décalage de convergence induit par le code CRC sera plus important lorsque K est petit. Le tableau 3.3 récapitule l'impact en dB du décalage de convergence pour différentes tailles de trames du standard LTE.

En revanche, l'impact en terme de taux de trames erronées, pour une valeur de SNR constante, ne peut être prédéterminé car il dépend des performances de décodage. Dans la suite, pour chaque standard utilisé, le code CRC associé à ce standard est exploité. Dans le cas d'un décodage par l'algorithme EML-MAP, il est également utilisé en tant que

Tableau 3.3 – Calculs du décalage de convergence pour différents turbo codes du standard LTE

	K=528	K=1024	K=2048	K=6144
δ_{dB}	0,202	0,103	0,051	0,017

critère d'arrêt. Pour l'algorithme détaillé ci-après, son emploi est nécessaire. Le rendement exact le comptabilisant est alors considéré dans tous les cas. Par exemple, pour un turbo code du standard LTE de rendement 1/3, le rendement exact vaut $\frac{\mathsf{K}-24}{3\times\mathsf{K}+3\times4}$.

3.3.2 Application aux turbo codes binaires

3.3.2.1 Détail de l'algorithme

Durant les $\mathrm{I}_{\min}$ premières itérations, le turbo décodeur itère sans que le code CRC ne soit vérifié. Ceci permet d'éviter les faux positifs qui peuvent survenir lorsque le mot décodé est situé à une distance importante du mot de code transmis. A l'itération $\mathrm{I}_{\min}$, si le code CRC est vérifié, le processus s'arrête. Sinon, la fiabilité de chacun des bits systématiques est caractérisée en utilisant la métrique Δ_1. Les q positions minimisant cette métrique sont alors extraites et stockées dans le vecteur Ω. À partir de ces q positions, il est possible de générer $2^{\mathsf{q}}-1$ mots candidats en inversant la décision prise par le décodeur sur un sous-ensemble de ces positions les moins fiables. Pour chacun de ces candidats, le code CRC est testé. Si le CRC d'un des candidats est vérifié, alors le décodage s'arrête. Sinon, l'itération suivante du processus est réalisée. Ceci est répété jusqu'à ce qu'un mot vérifie le CRC où jusqu'à ce que le nombre maximal d'itération soit atteint (I_{TC}). L'algorithme 4 récapitule l'ensemble de ces opérations.

Dans cet algorithme de décodage le paramètre q est particulièrement important. En effet, il détermine le compromis entre les performances de décodage et la complexité calculatoire. Ce paramètre définit la taille de l'espace de recherche des positions les moins fiables. Une valeur élevée de q permet assurément d'identifier plus d'erreurs, conformément à ce qui a été présenté en Figures 3.5 et 3.6. Néanmoins, lorsque q est incrémenté de 1, le nombre de vérifications de CRC est doublé.

Algorithme 4 : L'algorithme Flip and Check pour les turbo codes binaires

```
pour i : 1 à I_min faire
    Itération de Turbo Décodage
pour i : I_min + 1 to I_TC faire
    (d̂,L) = Itération de Turbo Décodage
    si CRCheck(d̂) ==true alors
        return(d̂)
    sinon
        pour k : 1 à K faire
            Δ_{1,k} = |L_k|
        Ω = Extraction des positions les moins fiables(Δ, q)
        pour j : 1 à 2^q − 1 faire
            D =Génération du mot candidat(Ω,j,d̂)
            si CRCheck(D) ==true alors
                return(D)
return(d̂)
```

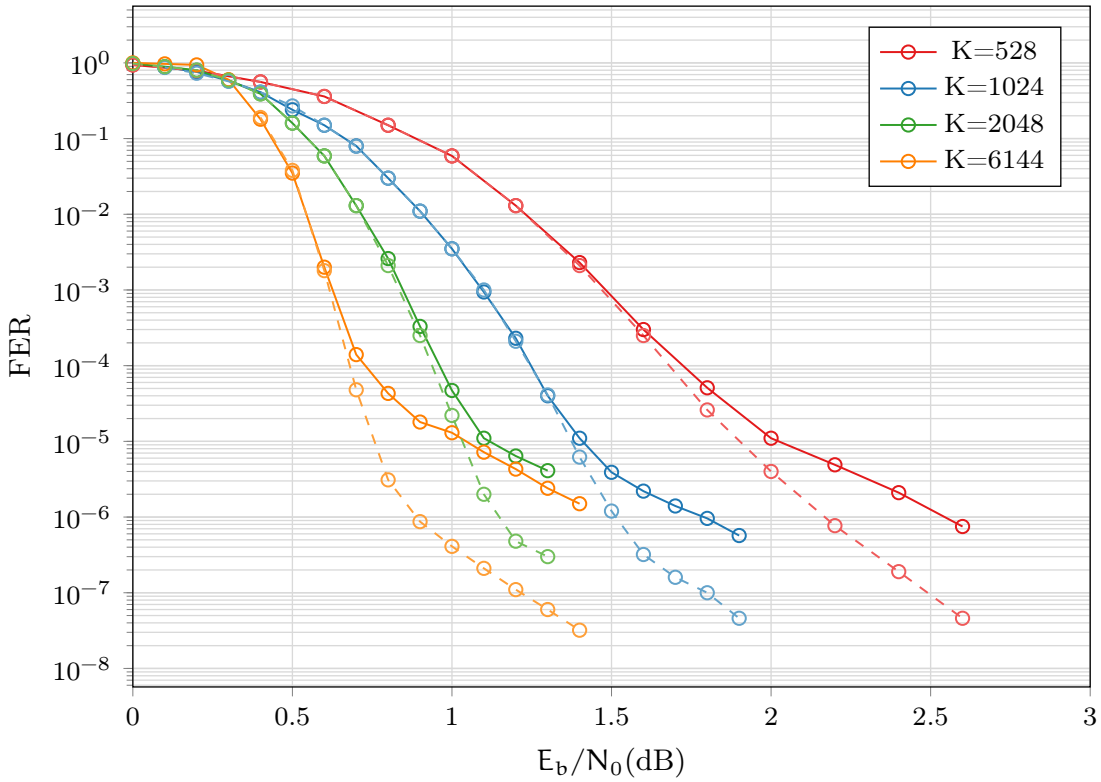


FIGURE 3.7 – Comparaison de performances de décodages entre EML-MAP et FNC. Standard LTE, K=528, 1024, 2048 et 6144. R=1/3. Décodeurs itérant jusqu'à 8 fois.

Dans la suite, le choix est alors fait de fixer la valeur de q à 10. De la sorte, selon les statistiques d'identification précédentes, un ordre de grandeur devrait être atteint sur les performances de décodages, ce sans trop impacter la complexité calculatoire globale du décodage. Cet impact relatif sur la complexité calculatoire sera confirmé dans une section prochaine, comparant cet algorithme de décodage avec les propositions de l'état de l'art traitant de la réduction du plancher d'erreurs des turbo codes. Plus encore, le chapitre prochain étudiera l'impact sur les performances de décodage d'une réduction de la valeur de ce paramètre q.

Dans la section suivante, une présentation des performances de décodage dans le cadre de turbo codes des standards LTE et CCSDS est réalisée. Cet algorithme est nommé Flip and Check et est abrégé en FNC.

3.3.2.2 Performances de décodage

Cette section présente des résultats de simulations Monte Carlo réalisées avec une représentation des données en virgule flottante. Les tailles des trames considérées pour le standard LTE sont 528, 1024, 2048 et 6144. Dans tous les cas, le rendement est fixé à 1/3. Afin d'éviter les problèmes de faux positifs liés à l'emploi du code CRC de ce standard, la valeur de I_{min} est fixée selon la taille de la trame de 2 à 5. Le processus de turbo décodage

TABLEAU 3.4 – Comparaison entre les valeurs de taux d'erreur trames attendues avec une correction parfaite et celui réellement obtenu

	K=528 @ 2,6 dB	**K=1024** @ 1,6 dB	**K=2048** @ 1,35 dB	**K=6144** @ 0,82 dB
FER EML-MAP (Fig. 3.7)	7×10^{-7}	$2{,}3 \times 10^{-6}$	3×10^{-6}	4×10^{-5}
BE $\leqslant 10$ (Fig. 3.2)	98%	94%	97%	97%
FER attendu	$1{,}4 \times 10^{-8}$	$1{,}4 \times 10^{-7}$	9×10^{-8}	$1{,}2 \times 10^{-6}$
FER FNC (Fig. 3.7)	5×10^{-8}	3×10^{-7}	2×10^{-7}	2×10^{-6}

peut itérer jusqu'à 8 fois. Enfin, la valeur de q pour l'algorithme FNC est fixée à 10. La Figure 3.7 présente une comparaison des performances entre un décodage utilisant l'algorithme EML-MAP avec critère d'arrêt sur code CRC (courbes en traits pleins) à un décodage basé sur l'algorithme FNC (courbes en pointillés).

Pour les quatre cas considérés, l'amélioration du processus de turbo décodage obtenu par l'algorithme FNC représente un gain en terme de FER au moins égal à un ordre de grandeur dans la zone du plancher d'erreurs. Cela signifie qu'au moins 90% des trames erronées sont corrigées par l'approche FNC. Puisque q vaut 10, l'algorithme FNC ne peut corriger au maximum que 10 erreurs binaires. Au début de ce chapitre, la Figure 3.2 présentait le nombre d'erreurs binaires par trame erronée pour ces turbo codes. De là, il est possible d'extrapoler le taux d'erreur trame obtenu si l'ensemble des trames contenant 10 erreurs ou moins est corrigé. Ainsi, le tableau 3.4 présente ce taux d'erreur trame attendu et celui réellement observé avec l'algorithme de décodage FNC et $\mathsf{q} = 10$. Il est à noter que lors de l'obtention des histogrammes de la Figure 3.2, le code CRC n'était pas utilisé. Il n'était alors pas comptabilisé dans le rendement. En revanche, pour les données de la Figure 3.7, le code CRC est soit utilisé en tant que critère d'arrêt, soit dans le processus FNC. Un décalage de la valeur du SNR par $10 \times \log_{10}\left(\frac{\mathsf{K}}{\mathsf{K}-24}\right)$ doit alors être considéré afin de pouvoir exploiter l'ensemble de ces données.

À la vue des performances obtenues vis-à-vis de celles attendues, présentées dans le Tableau 3.4 nous pouvons statuer sur l'absence de dégradations due à d'éventuels faux positifs émanant de la détection via le code CRC. Ceci provient conjointement de sa distance minimale suffisamment importante (provenant directement de sa taille) et de l'adaptation de $\mathrm{I}_{\min}$ en fonction de la taille de la trame.

Concernant le taux d'erreur binaire, les gains sont légèrement moins importants que ceux présentés pour le taux d'erreur trame. En effet, de part son principe, l'algorithme FNC corrige les trames à faible nombre d'erreurs binaires. Ainsi, les trames erronées après processus FNC possèdent un nombre important d'erreurs binaires. Néanmoins, des gains d'environ un ordre de grandeur sont observés dans certains cas considérés. Ces résultats sont présentés en Annexe C.

La Figure 3.8 présente les performances de l'algorithme FNC cette fois dans le contexte d'un turbo code du standard CCSDS. Les conditions sont les suivantes. La taille de la trame vaut 1784 et le rendement nominal R=1/3 est considéré. Puisque les turbo codes du

standard CCSDS possèdent une longueur de contrainte de 5, le nombre moyen d'itérations pour décoder une trame dans la zone du plancher d'erreurs est plus élevé que dans le contexte LTE. Aussi, le code CRC définit dans le standard CCSDS possède une longueur de 16. Ainsi son pouvoir de détection d'erreurs est plus faible que celui du standard LTE. Suite à ces deux constatations, la nombre maximal d'itérations est fixé à 10 et la valeur de I_{min} est fixée à 5 afin que les performances de décodage ne soit pas limitées par le code CRC.

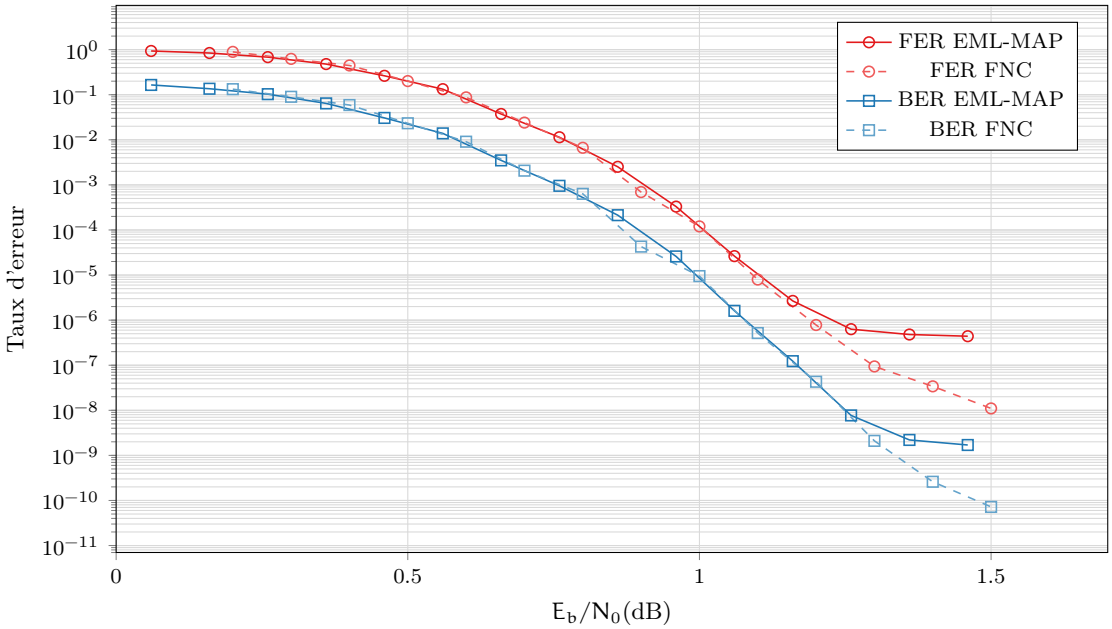


FIGURE 3.8 – Comparaison de performances de décodages entre EML-MAP et FNC. Standard CCSDS, K=1784. Décodeurs itérant jusqu'à 10 fois.

Il est visible sur la Figure 3.8 que les gains sont légèrement moins importants que pour le standard LTE. Néanmoins, ces gains sont proches de l'ordre de grandeur. Par cet exemple, nous entrevoyons une limite du principe FNC. En effet, si le code CRC définit par le standard considéré possède un risque important d'erreur non détectée, alors ce dernier est limitant dans les gains de décodage obtenu avec l'algorithme FNC.
Une comparaison avec l'état de l'art portant sur la réduction de la zone du plancher d'erreurs par des approches basées sur un post-traitement est maintenant dressée.

3.3.2.3 Positionnement par rapport à l'état de l'art

Performances de décodage Dans le chapitre premier, diverses méthodes permettant d'abaisser le plancher d'erreurs des turbo codes ont été présentées. Il s'agit des méthodes FSM (ou CIM), de la concaténation en série d'un code BCH et enfin du décodage par liste. Les méthodes de décodages par listes et FSM sont dépendantes d'un code détecteur d'erreurs, comme le principe FNC. Ainsi, pour obtenir une comparaison cohérente, la courbe de référence correspond à un turbo code sans concaténation avec un code CRC, bien que celui-ci soit déjà présent dans la majorité de standards utilisant un turbo code.

Ceci permet de ne pas désavantager l'approche considérant une concaténation avec un code BCH pouvant aussi agir comme code détecteur d'erreurs.

Dans [87], les performances de décodage de la méthode CIM sont présentées en prenant pour cadre un turbo code ayant pour polynôme générateur $(13{,}15)_8$. La taille de la trame vaut 1504. Le turbo décodeur implémente l'algorithme EML-MAP et itère 16 fois. Cependant dans cette publication, la détection d'erreurs est réalisée par un génie. Cela signifie que la décision du décodeur est comparée avec la trame émise. Afin de se positionner dans un contexte réaliste, le génie doit être remplacé par un code CRC. Dans la Figure 3.9, nous considérons pour ce cas le code CRC du standard LTE, de taille 24 bits. C'est pourquoi la courbe de performances associée à la méthode CIM est translatée vers la droite de $10 \times \log_{10}\left(\frac{1504}{1504-24}\right) = 0{,}07$ dB. Pour des raisons de complexité calculatoire qui seront détaillées plus loin, seul le cas correspondant à un turbo décodage supplémentaire au maximum (CIM(1)) est considéré. Dans ce cas, les performances de cette méthode et celles de l'algorithme FNC sont semblables.

Dans [68], un code BCH est concaténé avec le turbo code dans le but de corriger les erreurs résiduelles après le processus de turbo décodage. Dans la Figure 3.9, les performances associées à cette technique sont reportées. La capacité de correction d'un code BCH est directement donnée par la quantité d'information redondante (*cf.* premier chapitre). Or, cet ajout abaisse le rendement du schéma de codage. En résulte alors un décalage de la courbe de performance dans la zone de convergence vers la droite. Deux codes BCH différents sont testés : l'un corrigeant toutes les trames contenant 4 erreurs ou moins (t=4), l'autre corrigeant celles contenant 10 erreurs ou moins (t=10). Dans le premier cas, les performances dans la zone du plancher d'erreurs sont moins intéressantes que celles obtenues par l'algorithme FNC. Il faut en effet que le code BCH ait un pouvoir de correction de 10 pour obtenir les mêmes performances que l'algorithme FNC dans la zone du plancher d'erreurs. Cependant, dans ce cas, la pénalité de convergence vaut 0,33 dB. Son utilisation dans un contexte de communications numériques est alors compromise. L'emploi d'un tel code concaténé peut néanmoins avoir un intérêt pour des tailles de trames conséquentes. En effet la taille de l'information redondante est liée au logarithme en base 2 de la taille de trame. C'est la raison pour laquelle un tel schéma a été choisi avec le code LDPC du standard DVB-S2.

Complexité calculatoire Selon [102], les calculs nécessaires pour qu'un turbo décodeur effectue une itération de l'algorithme EML-MAP pour un turbo code à 8 états s'établissent à $130 \times \mathsf{K}$ additions et $62 \times \mathsf{K}$ opérations de comparaison et sélection.

Pour chaque itération, l'algorithme FNC est basé sur deux opérations principales. Tout d'abord, il s'agit de l'extraction des positions les moins fiables. Ceci peut être réalisé par un tri par insertion. Il nécessite alors $(\mathsf{K} - \mathsf{q}) \times \mathsf{q}$ opérations de comparaison. Ensuite les 2^{q} vérifications du code CRC doivent avoir lieu. Néanmoins, chacune de ces vérifications peut correspondre à une opération élémentaire. De la sorte, même si le nombre de calculs de CRC croit exponentiellement avec q, il reste relativement faible en comparaison du nombre d'opérations nécessaires à chaque itération du turbo décodeur. Aussi, il est à noter que l'approche FNC n'est pas à proprement parler un algorithme de post-traitement.

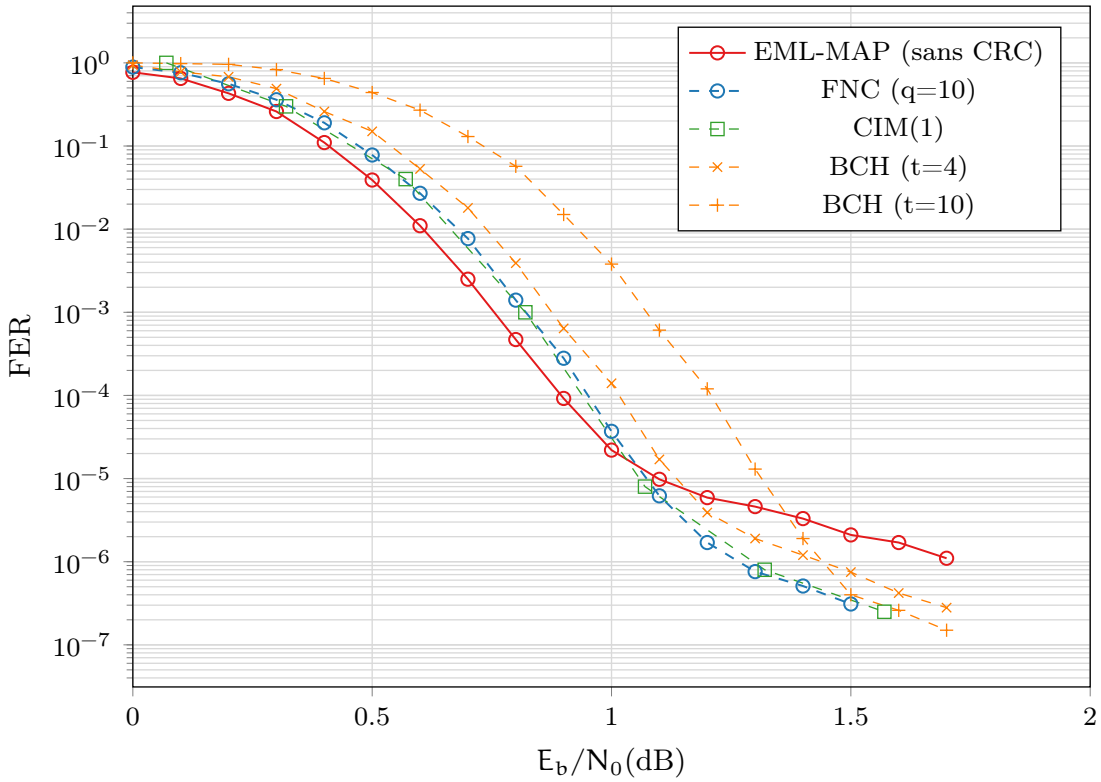


FIGURE 3.9 – Comparaison de performances de décodages entre EML-MAP, FNC, CIM et concaténation avec BCH. K=1504 et générateur polynomial $(13,15)_8$. Décodeurs itérant jusqu'à 16 fois.

En effet, son application sur les données produites par le turbo décodeur à l'itération i peut être envisagé lors de l'itération $i+1$. De la sorte, il n'impacte que peu la latence du système. Cette notion sera développée en détail dans le prochain chapitre.

L'approche CIM(1) repose sur l'emploi de deux processus de turbo décodage successifs. Sa complexité calculatoire est donc égale à deux fois celle d'un turbo décodage classique. A cela, il faut rajouter un tri des valeurs, qui est le même que pour l'algorithme FNC. En revanche, ce tri n'est effectué qu'une seul fois par trame alors que dans le cas de l'algorithme FNC il est effectué après chaque itération. En ce qui concerne la latence, la méthode CIM ne peut effectuer le post-traitement qu'après la fin du premier processus de décodage. Elle impacte alors fortement la latence maximale du système. Ce qui n'est pas le cas de l'approche FNC.

Enfin, en ce qui concerne la concaténation avec un code BCH, le décodage de ce dernier est basé sur un calcul de syndrome et l'emploi de l'algorithme de Berlekamp-Massey suivi pas celui de Chien. Ces algorithmes reposent sur des additions et multiplications dans un corps de Galois $GF(2^{\log_2 K})$, qui possèdent une complexité calculatoire relativement importante. Aussi, quelques additions et opérations de comparaison et sélection sont nécessaires (respectivement $t \times K$ et $6t \times K$).

TABLEAU 3.5 – Comparaison de la complexité calculatoire pour les différentes approches de post-traitement des turbo codes, valeurs théoriques

	TDec	TDec + FNC	TDec + CIM(1)	TDec + BCH
Add	$130.K.I_{TC}$	$K(130.I_{TC} + I_{TC} - I_{min})$	$2.130.K.I_{TC} + K$	$K(130.I_{TC} + t)$
CS	$62.K.I_{TC}$	$K(62.I_{TC} + q\,(I_{TC} - I_{min}))$	$2.62.K.I_{TC} + K$	$K(62.I_{TC} + 6.t)$
CRC	$I_{TC} - I_{min}$	$2^q.(I_{TC} - I_{min})$	$2(I_{TC} - I_{min})$	0
Add $GF(2^{11})$	0	0	0	$K(t-1) + t^2$
Mul $GF(2^{11})$	0	0	0	$2.t.K + \frac{3.t(t+1)-4}{2}$

TABLEAU 3.6 – Comparaison de la complexité calculatoire pour $K = 1504$, $I_{TC} = 16$, $I_{min} = 3$, $q = 10$ et $t = 10$

	TDec	TDec + FNC	TDec + CIM(1)	TDec + BCH
Add	3 128 320	3 143 360	6 258 144	3 143 360
CS	1 491 968	1 642 368	2 985 440	1 582 208
CRC	13	13 312	26	0
Add $GF(2^{11})$	0	0	0	13 636
Mul $GF(2^{11})$	0	0	0	30 243

Le Tableau 3.5 récapitule les valeurs formelles des opérations à réaliser pour ces différentes approches. Le Tableau 3.6 correspond aux applications numériques pour un turbo code 8 états avec K=1504 et R=1/3 et un processus de décodage itérant 16 fois (contexte de la Figure 3.9). A la vue des performances de décodage obtenues et de sa complexité calculatoire, l'algorithme FNC se compare favorablement aux approches de réduction du plancher d'erreurs des turbo codes.

3.3.2.4 Conclusions

L'algorithme Flip and Check pour les turbo codes binaires a été présenté. Celui-ci permet d'abaisser le plancher d'erreurs des turbo codes binaires d'au moins une décade. Ses performances sont proches de celles atteignables via une identification parfaite. La complexité calculatoire de cet algorithme réside dans les vérifications du code CRC. Cette complexité croit exponentiellement avec le nombre de positions les moins fiables considérées. Toutefois, une comparaison avec l'état de l'art a montré que son surcoût calculatoire est raisonnable tout en fournissant de très bonnes performances de décodage. Finalement, cette méthode peut s'adapter à des turbo codes standardisés dès lors qu'un code détecteur d'erreurs, comme un code CRC, y est concaténé en série. Ce travail à fait l'objet d'une publication dans revue internationale avec comité de lecture [115]. Dans la suite, une adaptation aux turbo codes double binaires est détaillée.

3.3.3 Application aux turbo codes double binaires

Cette section propose une transposition de l'algorithme FNC au cas des turbo codes double binaires. Après avoir décrit les transformations nécessaires, l'algorithme adapté aux turbo codes double binaires est exprimé. Enfin, une présentation des performances obtenues est réalisée.

3.3.3.1 Identification des symboles les moins fiables

Lors de la comparaison des métriques d'identification effectuée au début de ce chapitre, plusieurs propositions ont été comparées. L'une d'entre elles, la métrique Δ_1', possède les meilleures performances d'identification. C'est donc celle qui est retenue pour la transposition au cas double binaire. Cependant, dans le cas double binaire, les positions identifiées correspondent à des symboles double binaires. Chaque position représente alors 4 symboles différents. Suivant cette constatation, une adaptation directe requerrait la génération de 4^q mots candidats. Le nombre de vérifications du code CRC ayant la même valeur, seules de faibles valeurs de q pourraient être considérées. Or, d'après les statistiques d'identification présentées en Figure 3.6, afin de pouvoir observer une amélioration notable des performances de décodage, le choix de q implique de se porter *a minima* à 8 afin de corriger 80% des trames erronées dans la zone du plancher d'erreurs. Dans ce cas, 65536 vérifications de CRC seraient nécessaires, ce qui est irréaliste dans un contexte temps réel visant une faible complexité.

C'est pourquoi, il est nécessaire de réduire le nombre de symboles testés par position identifiée. Pour ce faire, une analyse a été menée quant à l'identification du bon symbole à l'aide d'un critère génie. Cette identification est semblable à celle permettant l'extraction des positions les moins fiables. Soient $S_{M_x}(k)$, $S_{M_y}(k)$, $S_{M_z}(k)$, et $S_{M_t}(k)$ respectivement le symbole le plus probable au sens des informations *a posteriori*, le second symbole le plus probable, le troisième et enfin le symbole le moins probable. Ils ont alors pour expression :

$$
\begin{aligned}
S_{M_x}(k) &= \arg\max_{s\in[\![0;3]\!]} \left(l_s^a(k)\right) \\
S_{M_y}(k) &= \arg\max_{s\in[\![0;3]\!]\setminus S_{M_x}(k)} \left(l_s^a(k)\right) \\
S_{M_z}(k) &= \arg\max_{s\in[\![0;3]\!]\setminus\{S_{M_x}(k),S_{M_y}(k)\}} \left(l_s^a(k)\right) \\
S_{M_t}(k) &= \{0,1,2,3\} \setminus \{S_{M_x}(k),S_{M_y}(k),S_{M_z}(k)\}
\end{aligned}
$$

Le tableau 3.7 présente des statistiques quant au symbole réellement transmis en fonction des valeurs *a posteriori* associées à ce symbole lorsqu'il est erroné à l'issue de l'itération courante. Ces statistiques ont été obtenues pour différents turbo codes du standard DVB-RCS. Cela permet de déduire les symboles à considérer lors de la génération des mots candidats. Nous pouvons constater que la première colonne vaut toujours 0% puisque le symbole le plus probable correspond au symbole décodé. Cependant, si pour obtenir S_{M_ξ} l'information *a posteriori* doit être reconstruite (cas où le décodeur SISO fournit en sortie l'information extrinsèque et le vecteur décidé), il est important de ne pas ommetre le facteur de remise à l'échelle. En effet, l'identification via la métrique Δ' est correct en

TABLEAU 3.7 – Statistiques sur le symbole transmis vis-à-vis des différents symboles possibles

	S_{M_x}	S_{M_y}	S_{M_z}	S_{M_t}
i=3	0%	96%	3%	1%
i=4	0%	96%	3%	1%
i=5	0%	97%	3%	0%
i=6	0%	97%	3%	0%
i=7	0%	97%	3%	0%
i=8	0%	97%	2%	1%

raison du faible écart entre les valeurs *a posteriori*. Ainsi, une légère modification dans leurs calculs modifie grandement la trame décodée. Il serait alors possible d'aboutir à ce que le symbole le plus probable estimé ne soit pas celui décodé.

À la lecture du tableau, il apparaît que dans la très grande majorité des cas, le symbole transmis correspond au symbole le deuxième plus probable. Cela implique que considérer plus que deux symboles ne permet alors pas d'augmenter de manière significative la probabilité de fournir le bon mot de code alors que la complexité calculatoire résultante serait tout à fait notable.

Finalement, de part cette analyse, considérer uniquement les deux symboles les plus probables est suffisant pour pouvoir identifier dans la plupart des cas le bon mot de code. Le nombre de vérifications de code CRC reste alors équivalent au cas binaire et vaut 2^q. Une étape supplémentaire consistant à l'extraction des deux symboles les plus probables est néanmoins nécessaire. Dès lors, il est possible de proposer l'algorithme complet, sujet de la prochaine sous-section.

3.3.3.2 Détail de l'algorithme pour les turbo codes double binaires

L'algorithme FNC pour les turbo codes double binaires est très similaire à celui du cas binaire. Seules quelques opérations supplémentaires s'avèrent nécessaires. Il peut être détaillé comme suit. Tout d'abord, I_{min} itérations de turbo décodages seuls sont effectuées. Ceci permet de réduire le risque de faux positifs lié à l'utilisation d'un code CRC. À partir de l'itération suivante, le code CRC commence à être vérifié. S'il ne l'est pas, le principe FNC est alors appliqué. D'abord, pour chaque position symbole dans la trame, les deux symboles les plus probables sont extraits au sens de l'information *a posteriori*. Ceci permet de calculer la métrique Δ_1'. Une fois que tous ses indices ont été calculés, les q positions des symboles les moins fiables sont extraites en triant par ordre croissant la métrique Δ_1'. Ces positions sont stockées dans le vecteur Ω. Ensuite, à partir des positions contenues dans Ω et des indices de S_{M_x} et S_{M_y}, les valeurs des deux symboles les plus probables sont extraites pour chacune des positions présupposées erronées. Il reste ensuite à générer les 2^q mots candidats en remplaçant la valeur des symboles aux positions correspondantes dans le mot produit à cette itération. Finalement, ces mots candidats sont vérifiés en utilisant le code CRC. Si un mot de code est trouvé, alors le processus itératif est stoppé.

Sinon, une nouvelle itération du processus de turbo décodage est effectuée. L'algorithme 5 récapitule l'ensemble de ces opérations.

Dans le contexte de turbo codes double binaires, l'algorithme FNC nécessite donc plus d'opérations que dans le cas binaire. Ceci provient directement de la manipulation de symboles. Cependant, l'opération d'extraction nécessaire des deux symboles les plus probables correspond à une manipulation pour l'hypothétique correction de deux bits. Dès lors, la complexité globale de l'algorithme par bit décodé n'est que peu modifiée. Ces deux extractions de valeurs maximales correspondant aux lignes 9, 10 et 13 de l'algorithme 5) correspondent à un tri entre 4 valeurs. Comme ces opérations ne sont réalisées qu'une fois par itération, leur impact sur la complexité calculatoire globale reste faible. Dans la suite de cette sous-section, une présentation des performances de décodage dans le contexte des standards DVB-RCS et DVB-RCS2 est réalisée.

Algorithme 5 : L'algorithme Flip and Check pour les turbo codes double binaires

1 **pour** i : *1 à* I_{min} **faire**
2 `Itération de Turbo Décodage`
3 **pour** i : $I_{min}+1$ *to* I_{TC} **faire**
4 $(\hat{\mathbf{d}},\mathbf{L})$ = `Itération de Turbo Décodage`
5 **si** `CRCheck`$(\hat{\mathbf{d}})$ ==*true* **alors**
6 `return`$(\hat{\mathbf{d}})$
7 **sinon**
8 **pour** k : *1 à* K **faire**
9 $S_{M_x}(k) = \underset{s\in[\![0;3]\!]}{\arg\max}\,(l^a_s(k))$
10 $S_{M_y}(k) = \underset{s\in[\![0;3]\!]\setminus S_{M_x}(k)}{\arg\max}\,(l^a_s(k))$
11 $\Delta'_k = l^a_{S_{M_x}}(k) - l^a_{S_{M_y}}(k)$
12 Ω = `Extraction des positions des symboles les moins fiables`(Δ, q)
13 ξ = `Extraction des 2 symboles les plus fiables`$(\Omega, S_{M_x}, S_{M_y})$
14 **pour** j : *1 à* 2^q-1 **faire**
15 $\mathbf{D}$ =`Génération du mot candidat`$(\Omega,\xi,j,\hat{\mathbf{d}})$
16 **si** `CRCheck`$(\mathbf{D})$ ==*true* **alors**
17 `return`$(\mathbf{D})$
18 `return`$(\hat{\mathbf{d}})$

3.3.3.3 Performances de décodage

Dans cette section les résultats de simulations Monte Carlo réalisées avec une représentation des données en virgule flottante sont détaillés. Un canal AWGN est toujours considéré. En revanche, la modulation numérique est maintenant une QPSK. Le nombre maximum d'itérations réalisées par le turbo décodeur est fixé à 8 et I_{min} vaut 3. Comme dans le cas binaire, la valeur de q est fixée à 10. Dans le standard DVB-RCS, les tailles de trame sont comprises entre quelques dizaines d'octets et 200 octets. Sept rendements différents sont définis allant de 1/3 à 6/7. Le code CRC considéré dans ce standard est de longueur 16. La Figure 3.10 compare les performances du turbo décodage conventionnel avec celles

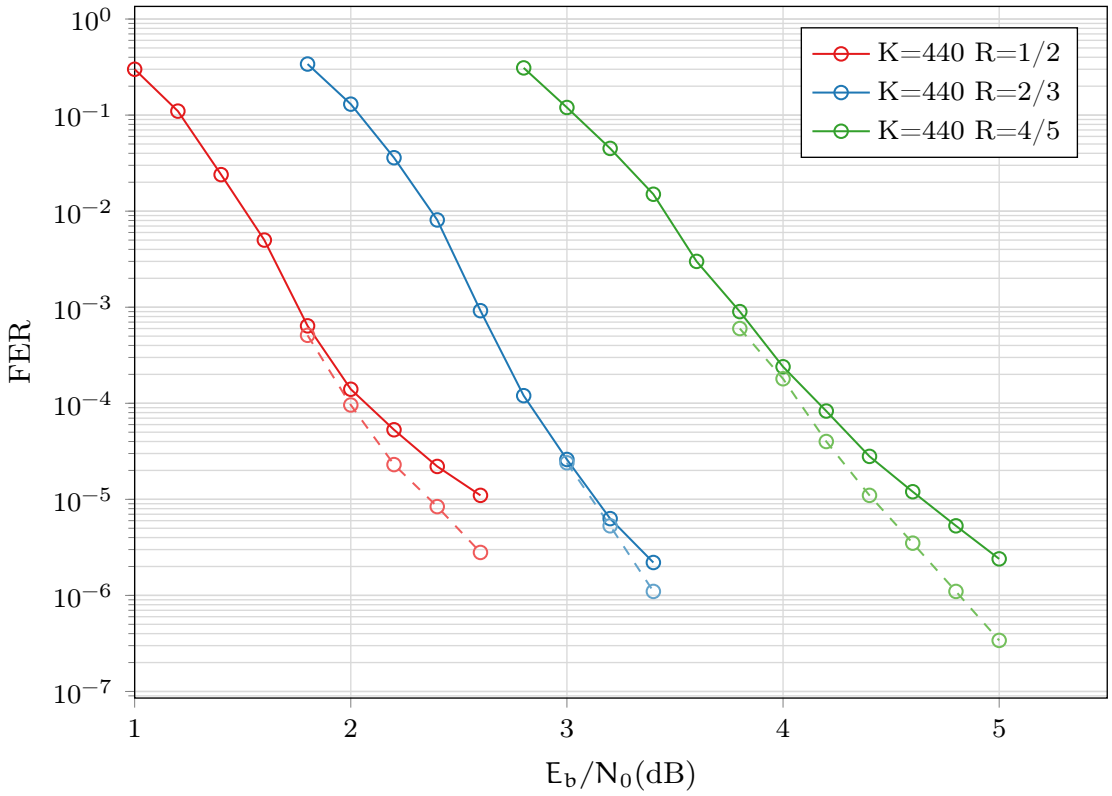


FIGURE 3.10 – Comparaison des performances de décodage entre EML-MAP et FNC. Standard DVB-RCS, K=440, rendements de 1/2 à 6/7. Décodeurs itérant jusqu'à 8 fois.

de l'algorithme FNC pour K=440 symboles, pour différentes valeurs de rendement et en considérant le code CRC 16-CCITT. Il apparaît que grâce à l'algorithme FNC, dans la zone du plancher d'erreurs, pour différents rendements de codes, des gains approchant un ordre de grandeur sont atteints. En terme de SNR, cela correspond à des gains allant de 0,2 dB à plus de 0,4 dB.

Pour les autres tailles de trame du standard, les gains sont équivalents. En résumé, il apparaît que la pente de la courbe dans la zone du plancher d'erreurs est modifiée, l'inflexion est moins importante grâce à l'algorithme FNC.

En 2012, la deuxième version du standard DVB-RCS a été publiée. Le code correcteur considéré est toujours un turbo code double binaire. En revanche, afin d'améliorer les performances de correction, la longueur de contrainte des codes constituants est augmentée. Des turbo codes à 16 états sont alors proposés. Un nouvel entrelaceur ARP ainsi que des tailles de trame plus conséquentes sont également retenus. La taille du code CRC concaténé en série double, pour passer à une taille de 32 bits. Enfin, de nouveaux schémas de poinçonnage sont ajoutés. De la sorte, ces turbo codes font partis des codes correcteurs les plus performants et flexibles standardisés. La complexité calculatoire de ce code a toutefois doublé en comparaison avec celle du turbo code de la génération précédente.

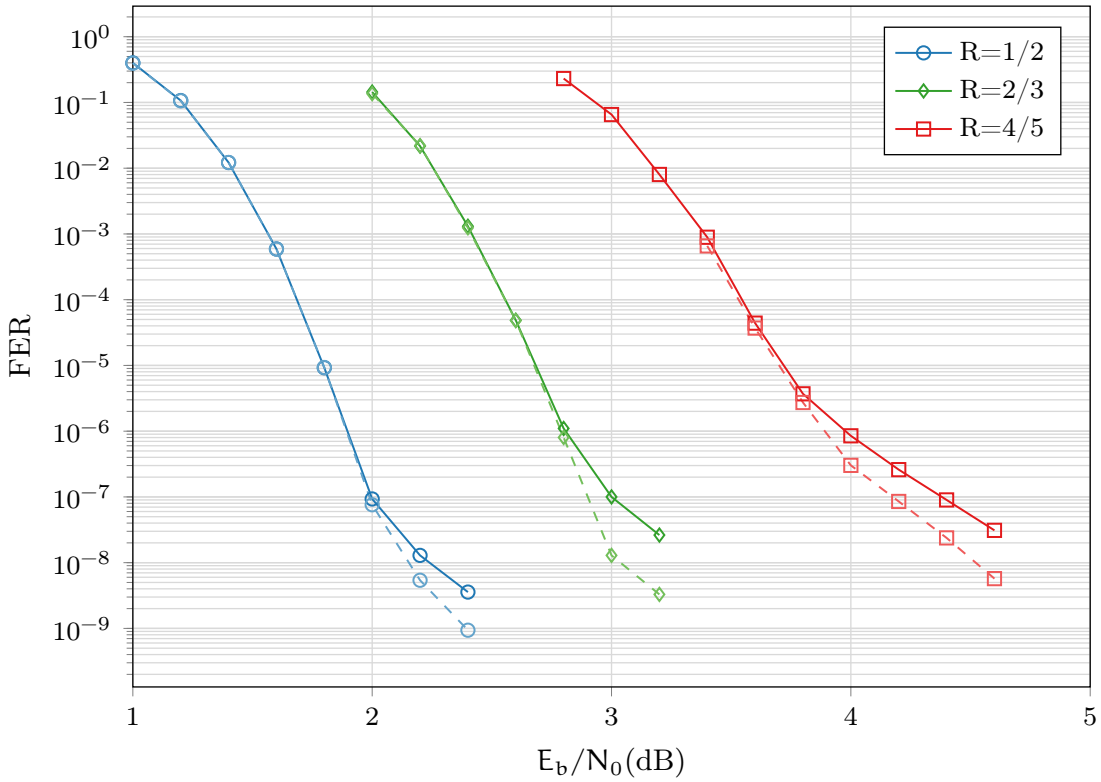


FIGURE 3.11 – Comparaison de performances de décodages entre EML-MAP et FNC. Standard DVB-RCS 2, K=752, rendements de 1/2 à 4/5. Décodeurs itérant jusqu'à 8 fois.

La Figure 3.11 présente une comparaison des performances de décodage pour le standard DVB-RCS 2 avec K=752. Différents rendements sont considérés : 1/2, 2/3 et 4/5. Les autres paramètres de la simulation restent les mêmes que pour le cas 8 états. Tout d'abord, en considérant un décodage conventionnel, il apparaît que les performances de ce code surpassent très largement celles du DVB-RCS. En effet, les points d'inflexions des différentes courbes de performance apparaissent 2 à 3 décades plus bas. La zone du plancher d'erreurs correspond alors à des taux d'erreur trame de l'ordre de 10^{-7}. Ceci entraîne un temps conséquent pour les simulations Monte-Carlo. De plus, comme l'algorithme FNC vise à abaisser le plancher d'erreurs, apercevoir une trame erronée devient alors extrêmement rare. Ainsi les courbes obtenues ne comptent que 50 trames erronées avec l'algorithme de décodage FNC.

Il apparaît que même dans un contexte de turbo code très performants, des gains conséquents dans la zone du plancher d'erreurs sont obtenus grâce à l'emploi de l'algorithme FNC. À nouveau, un changement de la pente de la courbe dans cette zone est visible. Ce travail a fait l'objet d'une publication en conférence internationale avec acte [116].

3.4 Conclusion

Dans ce troisième chapitre, une étude sur les erreurs résiduelles des turbo codes a été menée. Dans un premier temps, leur existence a été caractérisée grâce aux spectres de distances des turbo codes. Ces erreurs étant responsables de la zone du plancher d'erreurs, des propositions de critères d'identifications ont alors été comparées.

Cette comparaison a permis de mettre en exergue une métrique, permettant d'identifier de façon quasi-systématique les erreurs résiduelles à l'issue du processus de turbo décodage. À partir de celle-ci, un algorithme, nommé Flip and Check a été proposé. Cet algorithme tire parti de l'augmentation de la distance minimale obtenue via la concaténation en série d'un code détecteur d'erreurs et d'un turbo code. Un tel schéma de codage est couramment rencontré dans les standards de communications numériques. Le principe de cet algorithme repose sur l'identification des positions les moins fiables dans la séquence en train d'être décodée. De là, différents mots de code candidats sont générés puis vérifiés en utilisant le code détecteur d'erreurs. Des gains d'au moins un ordre de grandeur ont été observés, ce, quelque soit le turbo code standardisé considéré. Ainsi, des gains de performances conséquents sont observables pour des turbo codes binaires à 8 ou 16 états ainsi que pour des turbo codes double binaires eux aussi à 8 ou 16 états. La complexité calculatoire additionnelle nécessaire à cet algorithme est modérée, comme le montre l'analyse de complexité calculatoire. Cela rend cette approche compétitive par rapport à l'état de l'art sur la réduction du plancher d'erreurs des turbo codes.

Dans le chapitre suivant, après une étude des architectures matérielles de turbo décodeurs, des propositions d'implémentation de l'algorithme Flip and Check sont détaillées.

4 Architecture matérielle de correction des erreurs résiduelles

Dans le chapitre précédent, un algorithme permettant la correction des erreurs résiduelles a été présenté. Le quatrième chapitre étudie l'implémentation matérielle de cet algorithme. Puisque ce dernier doit être interfacé avec un turbo décodeur, il est nécessaire dans un premier temps d'étudier les architectures matérielles des turbo décodeurs existantes dans la littérature.

Suite à cela, dans un but d'adéquation entre l'algorithme et l'architecture, l'influence des différents paramètres de l'algorithme FNC sur les performances de décodage sera étudiée. Ceci afin d'identifier le meilleur compromis entre la complexité calculatoire et le gain en correction d'erreurs résiduelles.

Enfin, un cas d'étude sera considéré afin de proposer une architecture matérielle de correction des erreurs résiduelles. Des résultats d'implémentation sur cible FPGA illustreront la complexité matérielle de cette architecture. Ce chapitre se terminera sur des propositions d'adaptations à réaliser pour pouvoir adresser un très large spectre de turbo décodeurs.

4.1 Les architectures matérielles de turbo décodeurs

Dans cette section, un état de l'art des architectures de turbo décodeurs est mené. Dans un premier temps, des détails sont donnés pour des architectures basées sur un processus itératif de turbo décodage séquentiel. Dans ce cas, une étude du parallélisme au sein d'un décodeur SISO (parallélisme intra-SISO) est menée. Dans un second temps, des architectures matérielles faisant intervenir plusieurs décodeurs SISO en même temps sont détaillées. Ceci correspond au parallélisme inter-SISO.

4.1.1 Traitement itératif séquentiel et parallélisme intra-SISO

Le décodage des turbo codes est basé sur un échange d'informations extrinsèques entre différentes instances de décodage SISO. À partir des informations du canal et des informations *a priori*, chaque instance de décodage SISO évalue les informations *a posteriori*. De là, les informations extrinsèques sont déduites. Ces dernières sont alors utilisées en tant qu'informations *a priori* lors de la demi-itération suivante conjointement avec les informations de l'autre domaine. La Figure 4.1 présente la séquentialité de ces opérations lors du processus de décodage itératif. Dans ce cas, un seul décodeur SISO effectue toutes les instances de décodage.

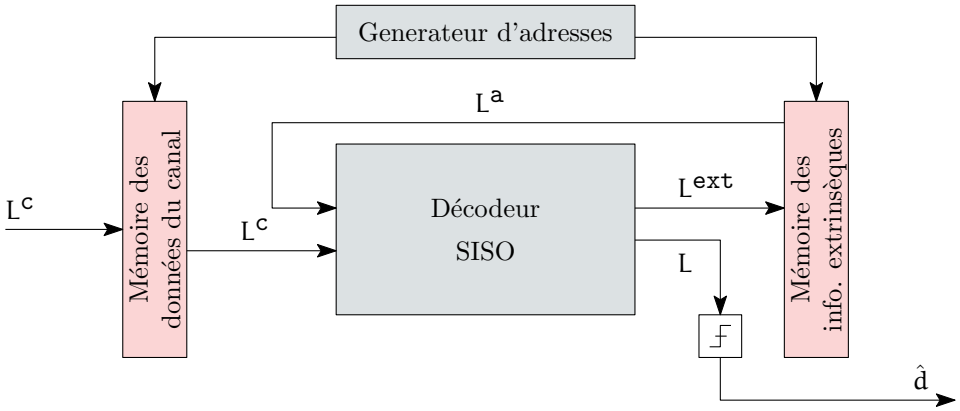


FIGURE 4.1 – Principe du turbo décodage séquentiel.

Dans les architectures matérielles de turbo décodeurs, l'algorithme de décodage SISO choisi pour l'obtention des informations *a posteriori* est l'algorithme MAP, ou plutôt une des ses simplifications, à savoir l'algorithme EML-MAP. Cet algorithme repose sur trois calculs successifs et séquentiels. Suivant l'ordonnancement choisi et le degré de parallélisme retenu pour ces calculs, différentes valeurs au niveau des ressources calculatoires, de mémorisation et des temps d'exécutions sont obtenues. Une description incrémentale des différentes approches de l'implémentation du décodeur SISO est maintenant menée, suivant la classification proposée dans [103].

4.1.1.1 Traitement séquentiel de l'algorithme BCJR

BCJR aller-retour Les étapes de l'algorithme BCJR sont détaillées ci-après. Tout d'abord, le calcul des métriques de branches, notées γ, est effectué. Ceci permet d'initier

le calcul récursif des métriques de nœuds aller, notées α. Chaque métrique de nœud aller est calculée en combinant la transition courante du treillis avec les métriques α de la section de treillis précédente. Le calcul récursif des métriques de nœuds retour, notées β, est similaire mais considère un parcours du treillis dans l'autre sens, en partant de la fin du treillis. Finalement, les informations *a posteriori* sont calculées en combinant les trois métriques α, β et γ.

Comme dans le cas de l'algorithme de Viterbi, chaque calcul de métrique de nœud est réalisé par une opération ACS (Addition-Comparaison-Sélection). En allouant autant d'unités ACS qu'il existe de nœuds dans une section du treillis, il est possible de calculer parallèlement toutes les métriques de nœuds (aller ou retour) d'une même section. Ce parallélisme ne nécessite que peu de surcoût matériel puisque seules les unités ACS sont dupliquées sans nécessiter de mémoire additionnelle. En effet chaque opération ACS est exécutée sur le même ensemble de données. Ainsi, toutes les métriques de nœuds d'une section de treillis peuvent être calculées en un cycle d'exécution. Le calcul récursif des métriques de nœuds nécessite alors au total K cycles d'exécutions.

Dans ce contexte, un cycle d'exécution après le début du calcul des métriques de nœuds retour, il est possible d'entamer le calcul de l'information *a posteriori*. Cet ordonnancement des opérations correspond à celui originellement proposé par Bahl, Cocke, Jelinek et Raviv lors de la présentation de leur algorithme de décodage [21]. Dans le suite du manuscrit, cet ordonnancement des opérations est nommé Forward-Backward (FB). Il est schématisé dans la Figure 4.2. Il apparaît que le temps requis pour exécuter une demi-itération du processus de décodage itératif équivaut à $2 \times K$ cycles d'horloge. Dans le cadre du standard LTE, à cause des bits de terminaison de treillis, cette durée est légèrement plus importante en réalité. Cependant, afin d'exprimer plus simplement le nombre de cycles nécessaires, le temps de traitement des bits de terminaison n'est jamais pris en compte dans ce chapitre. Pour pouvoir disposer des informations *a posteriori*, la mémorisation des K métriques de nœuds aller est requise.

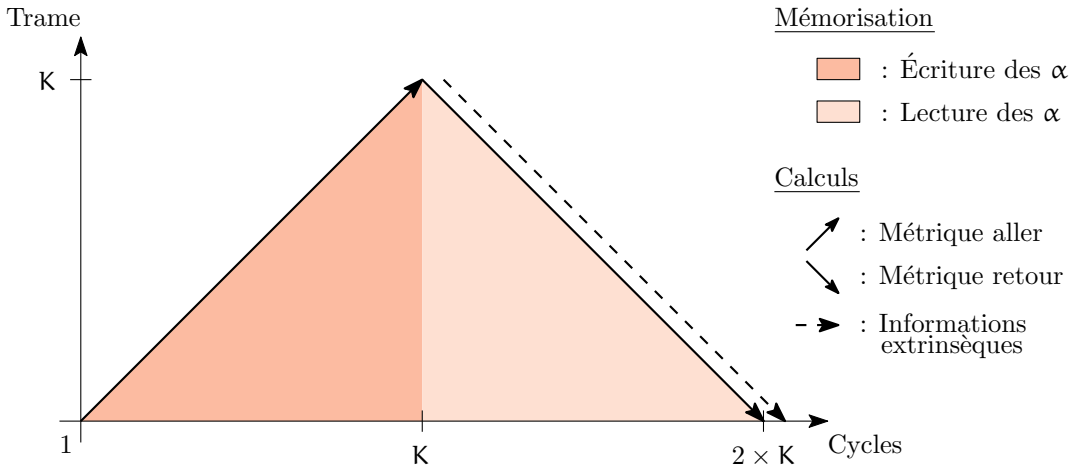


FIGURE 4.2 – Ordonnancement BCJR Aller-Retour (FB).

BCJR aller-retour à l'aide d'une fenêtre glissante La technique de fenêtre glissante (sliding-window) a été proposée pour faciliter le traitement associé à un ordonnancement FB. L'objectif réside en la réduction de la quantité de données à mémoriser. Son principe est le suivant. La trame à décoder est découpée en W fenêtres de tailles $\frac{K}{W}$. Après avoir calculé K/W métriques de nœuds aller, le calcul des métriques de nœuds retour débute à l'index K/W. Lors du cycle d'exécution suivant, le calcul des informations *a posteriori* est initié. À la fin de la récursion retour, le traitement est appliqué à une autre fenêtre de la trame. Ce procédé est répété W fois, jusqu'à ce que l'ensemble de la trame soit traitée. Les besoins de mémorisation se limitent alors à K/W métriques de nœuds aller. Cependant, la suppression de la récursion aux limites des fenêtres introduit une dégradation au niveau des performances de décodage. Pour palier ce problème, deux techniques ont été proposées. Il s'agit de l'utilisation :

— d'une fenêtre d'acquisition. Dans ce cas, le calcul des métriques de nœuds retour est appliqué sur quelques sections de treillis autour de la fenêtre. Une étude empirique a montré que la taille de prologue ayant le meilleur compromis entre calculs supplémentaires et gain de décodage se situe entre 3 et 5 fois la longueur de contrainte du code constituant [104].

— du passage de messages. Son principe consiste à utiliser les métriques de nœuds calculées lors de l'itération précédente comme valeurs initiales pour les limites de la fenêtre. Cette technique propose de meilleures performances de décodage que la technique de la fenêtre d'acquisition pour un surcoût de mémorisation limité [105].

Si l'ordre des calculs est inversé, c'est-à-dire si en premier lieu le calcul des métriques de nœud retour est effectué, il est possible de produire les informations extrinsèques dans l'ordre croissant en n'utilisant que $K/W + K$ cycles. Dans ce cas, les unités de calcul de nœuds aller et retour doivent fonctionner en même temps. Ceci ne nécessite pas de ressources matérielles supplémentaires puisqu'elles sont déjà présentes dans le cas de l'ordonnancement FB. Les besoins en mémorisation sont en revanche réduits à K/W données. Cet ordonnancement est nommé dans la suite Backward-Forward-Sliding-Window (BF-SW). La Figure 4.3 schématise un tel ordonnancement pour $W = 2$.

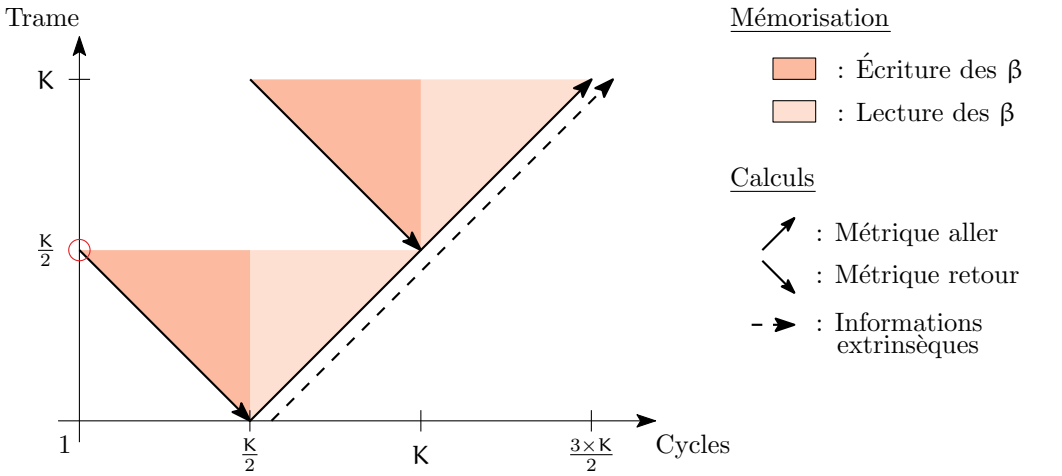


FIGURE 4.3 – Ordonnancement BCJR Retour-Aller avec fenêtre glissante, W=2 (BF-SW).

4.1.1.2 Traitement parallèle de l'algorithme BCJR

BCJR en ailes de papillon À partir de l'ordonnancement BF-SW, en ne considérant qu'une seule fenêtre mais en doublant le nombre d'unités de calcul d'information *a posteriori*, il est possible de débuter les calculs récursifs des métriques de nœuds aller et des métriques de nœuds retour au même cycle. Dans ce cas, à partir de la moitié du parcours du treillis, deux informations extrinsèques sont produites durant chaque cycle d'exécution. Cet ordonnancement est nommé BCJR en ailes de papillon (butterfly, abrégé en BFLY dans la suite) [106]. Il est illustré par la Figure 4.4. De la sorte, seuls K cycles d'exécution sont nécessaires pour effectuer une demi-itération de turbo décodage. Le nombre de données à mémoriser est similaire à celui de l'ordonnancement BCJR aller-retour. En revanche, les informations extrinsèques sont produites par paires à partir du $K/2^{\text{ième}}$ cycle d'exécution. Cet ordonnancement permet donc de réduire le temps de traitement du turbo décodage d'un facteur deux par rapport à l'ordonnancement FB.

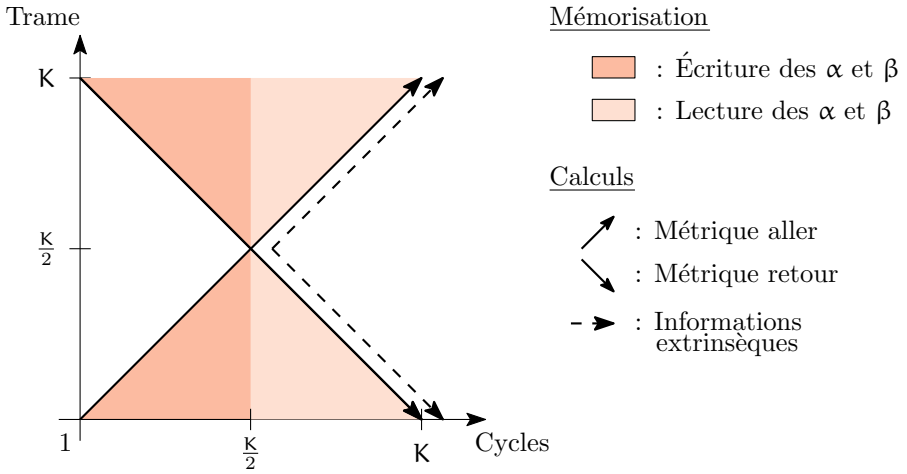


FIGURE 4.4 – Ordonnancement BCJR Butterfly (BFLY).

BCJR en ailes de papillon à l'aide d'une fenêtre glissante Toujours dans un but de réduire le nombre de données à mémoriser, il est possible d'utiliser le concept de fenêtre glissante conjointement à l'ordonnancement BFLY. Comme pour l'ordonnancement FB, le découpage de la trame en W fenêtres permet de réduire les ressources de mémorisation d'un facteur W. Cependant, dans ce cas, aucun gain en latence n'est obtenu. La Figure 4.5 détaille cet ordonnancement (abrégé en BFLY-SW) pour $W = 2$. Dans ce cas, les informations extrinsèques sont produites par paires durant deux intervalles de temps ayant pour durée K/4 cycles d'exécution.

4.1.2 Parallélisme inter-SISO

Il est aussi possible d'agir sur d'autres niveaux de parallélisme en plus de celui concernant l'ordonnancement des traitements de l'algorithme BCJR. Cette section présente alors l'interaction de décodeurs SISO lorsqu'ils fonctionnent en parallèle. Tout d'abord un parallélisme opérant au niveau du calcul d'une demi-itération est présenté.

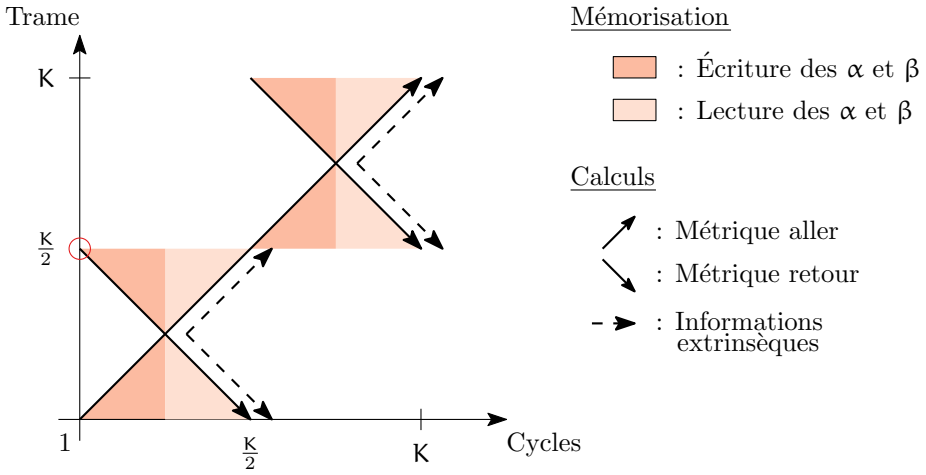


FIGURE 4.5 – Ordonnancement BCJR Butterfly avec fenêtre glissante (BFLY-SW).

4.1.2.1 Parallélisme de sous-bloc

À partir de l'algorithme basé sur une fenêtre glissante, afin de gagner un niveau de parallélisme supplémentaire, il est possible d'augmenter le nombre de décodeurs SISO. Chaque décodeur opère alors sur une fenêtre différente de la trame. La Figure 4.6 illustre un tel degré de parallélisme pour un nombre de sous blocs de deux (B = 2). Dans ce cas, deux décodeurs SISO se répartissent le traitement de la trame. Un décodeur SISO s'occupe des informations allant de l'index 1 à K/2 et le second de l'index K/2+1 à K. Là encore, une initialisation aux limites des sous-trames s'avère nécessaire. Les mêmes techniques que présentées précédemment peuvent être envisagées. Cependant, cette fois, en plus des métriques de nœuds retour, les métriques de nœuds aller doivent aussi être considérées. Aussi ce degré de parallélisme implique que l'entrelaceur puisse lui aussi être parallélisé afin qu'aucun conflit d'accès ne puisse apparaître [107]. La temps nécessaire pour effectuer une demi-itération est alors réduit d'un facteur B. Cependant, les ressources calculatoires associées aux décodeurs SISO sont multipliées par ce même facteur.

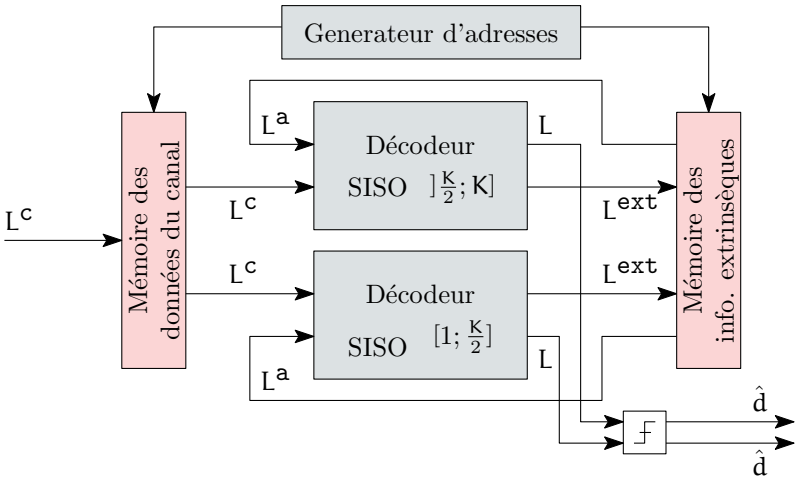


FIGURE 4.6 – Principe de turbo décodage basé sur plusieurs décodeurs SISO.

Le choix de l'ordonnancement au sein de chaque décodeur SISO n'est pas contraint. La Figure 4.7 présente un niveau de parallélisme de 2 associé à l'ordonnancement BFLY pour un nombre de sous-blocs valant 2. Cet ordonnancement est nommé BFLY-SB dans la suite. Il apparaît que dans ce cas, K/2 cycles d'exécution sont nécessaires pour effectuer une demi-itération. Durant les K/4 derniers cycles d'exécution, 4 informations extrinsèques sont produites en parallèle. Finalement, le nombre de données à mémoriser est équivalent à celui d'un ordonnancement BFLY.

Afin de réduire les besoins en terme de mémorisation, il est possible de combiner le parallélisme de sous-blocs avec le principe de fenêtre glissante. De la sorte, en considérant W fenêtres différentes, la taille de la mémoire est réduite d'un facteur W au niveau des métriques de nœuds. La Figure 4.8 présente un ordonnancement Butterfly avec 2 fenêtres glissantes et un parallélisme de décodeurs SISO de 2. Cet ordonnancement est nommé BFLY-SB-SW dans la suite du manuscrit.

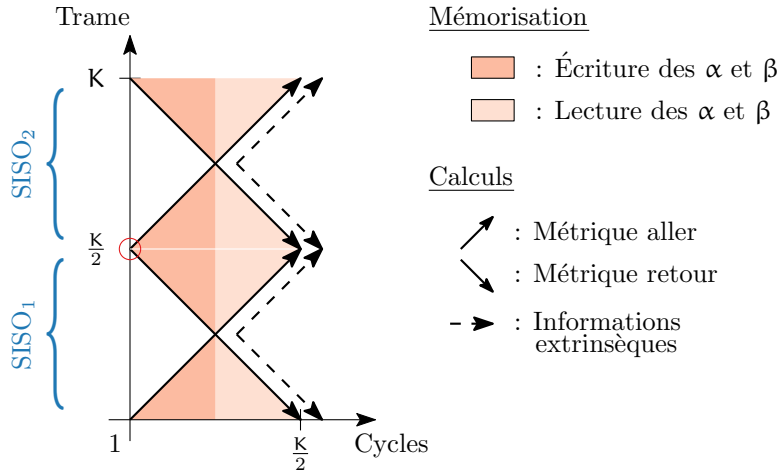


FIGURE 4.7 – Ordonnancement BCJR Butterfly en parallèle (BFLY-SB).

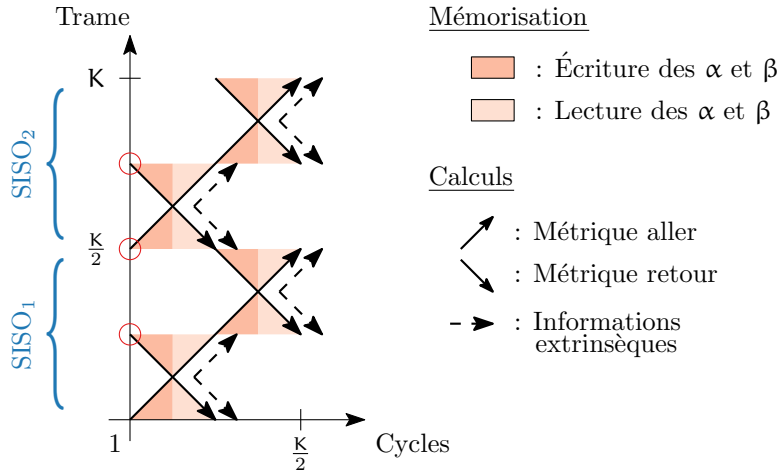


FIGURE 4.8 – Ordonnancement BCJR Butterfly avec fenêtre glissante en parallèle (BFLY-SW-SB).

4.1.2.2 Parallélisme de décodeur composant

Plus encore, il est possible de considérer un autre niveau de parallélisme. Il est en effet possible de faire opérer le décodage des deux codes convolutifs élémentaires en même temps [108]. Dans ce cas, après chaque itération, chaque décodeur élémentaire travaillant dans un certain domaine fournit ses informations extrinsèques au décodeur élémentaire utilisant les données du canal de l'autre dimension. Cependant, cette technique nécessite à performances de décodage égales deux fois plus d'itérations que le turbo décodage conventionnel. Ceci vient du fait que le principe itératif du turbo décodage conventionnel n'est plus respecté dans cette organisation.

Pour palier cela, en 2005, le décodage dit shuffled a été proposé [109]. Dans ce cas, les informations extrinsèques sont échangées entre les deux domaines dès qu'elles ont été calculées. Chaque décodeur composant dispose donc plus vite des informations *a priori* mises à jour avant la fin des traitements associés à la demi-itération courante. Ceci a alors pour effet d'augmenter la vitesse de convergence du turbo décodeur. Il est cependant à noter que ceci est vrai avec un ordonnancement Butterfly-Replica [110]. Ce dernier permet, au prix d'un doublement de la mémoire associée aux métriques de nœuds, de produire des informations extrinsèques durant toute la durée de la demi-itération. Cela participe à l'augmentation de la vitesse de convergence. La Figure 4.9 présente un échange d'informations extrinsèques au cours du processus de décodage shuffled pour deux décodeurs composants travaillant en simultané.

Pour de telles implémentations de turbo décodeurs, une approche utilisant de multiples processeurs à jeu d'instructions spécifiques (ASIP) est à privilégier [111].

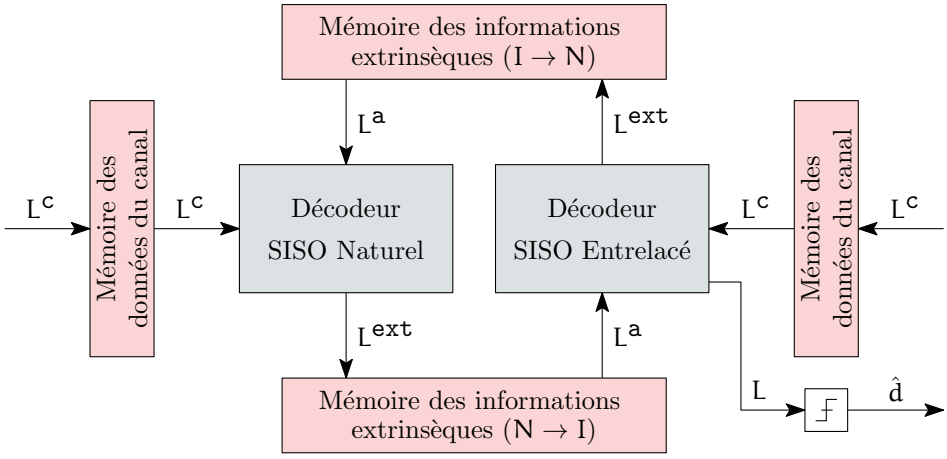


FIGURE 4.9 – Principe de turbo décodage Shuffled.

4.1.2.3 Parallélisme au niveau du processus turbo

La dernière technique permettant d'augmenter l'efficacité du processus de décodage revient à dupliquer les décodeurs SISO au cours du traitement itératif. Ceci permet alors de décoder simultanément plusieurs trames. Cette méthode revient à utiliser une architecture

pipeline pour le turbo décodeur. Ainsi des décodeurs SISO sont assignés à une ou plusieurs itérations. Cela implique une organisation pipeline pour mémoriser les trames entre les décodeurs SISO. Dans ce cas, la profondeur maximale du pipeline équivaut à deux fois le nombre maximal d'itérations. Bien que permettant une augmentation conséquente du débit, cette approche ne réduit pas la latence du processus de décodage. En revanche, elle s'avère coûteuse sur le plan matériel puisque toutes les ressources calculatoires et de mémorisation sont dupliquées P fois, avec P la profondeur du pipeline.

4.1.3 Conclusion

Dans cette première section, différentes organisations d'architectures matérielles dédiées au turbo décodage ont été présentées. Différentes techniques visant à augmenter le parallélisme de calcul dans le processus de décodage ont été décrites. Elles favorisent une diminution de la latence globale du décodeur et/ou une réduction de la quantité de données à mémoriser. Néanmoins les gains s'obtiennent aux dépens de la complexité calculatoire. De plus, elles impactent l'ordonnancement de la génération des informations *a posteriori*. Une synthèse de ces informations est donnée dans le tableau 4.1. Les résultats pour le décodeur SISO sont donnés en fonction de l'ordonnancement et du degré de parallélisme.

Puisque l'algorithme FNC peut être vu comme une extension de l'algorithme de turbo décodage, il est nécessaire qu'une architecture matérielle de l'algorithme FNC puisse être adaptée à celle d'un turbo décodeur. L'interconnexion entre ces deux architectures matérielles indépendantes nécessite un transfert du turbo décodeur vers le bloc FNC des métriques Δ, basées sur les informations *a posteriori*. Dans la suite de ce chapitre, une architecture matérielle adaptée à un turbo décodeur BF-SW est proposée. En effet, ce contexte considère une transmission des métriques de façon séquentielle, permettant une interconnexion plus aisée. En même temps, cet ordonnancent permet de réduire la latence du décodeur et les quantités de données mémorisées vis-à-vis d'un ordonnancement FB. Mais avant cela, une étude est menée sur l'impact des différents paramètres de l'algorithme FNC sur les performances de décodage et sur la complexité calculatoire. De plus, la robustesse à la quantification de l'algorithme FNC est évaluée en étudiant ses performances pour une représentation des informations en virgule fixe.

Tableau 4.1 – Caractérisation des différentes architectures matérielles de turbo décodeur existantes dans la littérature

Ordonnancement	Durée d'une demi-itération	Durée d'une itération	Durée de la transmission d'informations *a posteriori*	Nombre d'informations *a posteriori* simultanées
FB	$2 \times \mathsf{K}$	$4 \times \mathsf{K}$	K	1
BF-SW (W)	$\mathsf{K} + \frac{\mathsf{K}}{W}$	$2 \times (\mathsf{K} + \frac{\mathsf{K}}{W})$	K	1
BFLY	K	$2 \times \mathsf{K}$	$\frac{\mathsf{K}}{2}$	2
BFLY-SW (W)	K	$2 \times \mathsf{K}$	$\frac{\mathsf{K}}{2} = W \times \frac{\mathsf{K}}{2 \times W}$	2
BFLY-SB (B)	$\frac{\mathsf{K}}{\mathsf{B}}$	$2 \times \frac{\mathsf{K}}{\mathsf{B}}$	$\frac{\mathsf{K}}{2 \times \mathsf{B}}$	$2 \times \mathsf{B}$
BFLY-SW-SB (W,B)	$\frac{\mathsf{K}}{\mathsf{B}}$	$2 \times \frac{\mathsf{K}}{\mathsf{B}}$	$\frac{\mathsf{K}}{2 \times \mathsf{B}} = W \times \frac{\mathsf{K}}{2 \times W \times \mathsf{B}}$	$2 \times \mathsf{B}$

4.2 Étude de l'impact des paramètres de l'algorithme FNC

L'algorithme FNC est ajustable en fonction de plusieurs paramètres. Ces derniers influencent à la fois les performances de décodage mais également la complexité calculatoire. Cette section vise à caractériser l'impact de chacun de ces paramètres sur les performances de décodage dans le cadre de turbo codes du standard LTE. L'objectif est de réduire la complexité calculatoire de l'architecture FNC et de présenter la dégradation des performances de décodage associée, si elle existe. Pour ce faire, trois axes sont considérés. Tout d'abord, le nombre de fois où l'approche FNC est appliquée par trame. Ensuite, le nombre des mots candidats considérés est étudié. Finalement, la quantification des informations à l'intérieur du processus FNC est considérée.

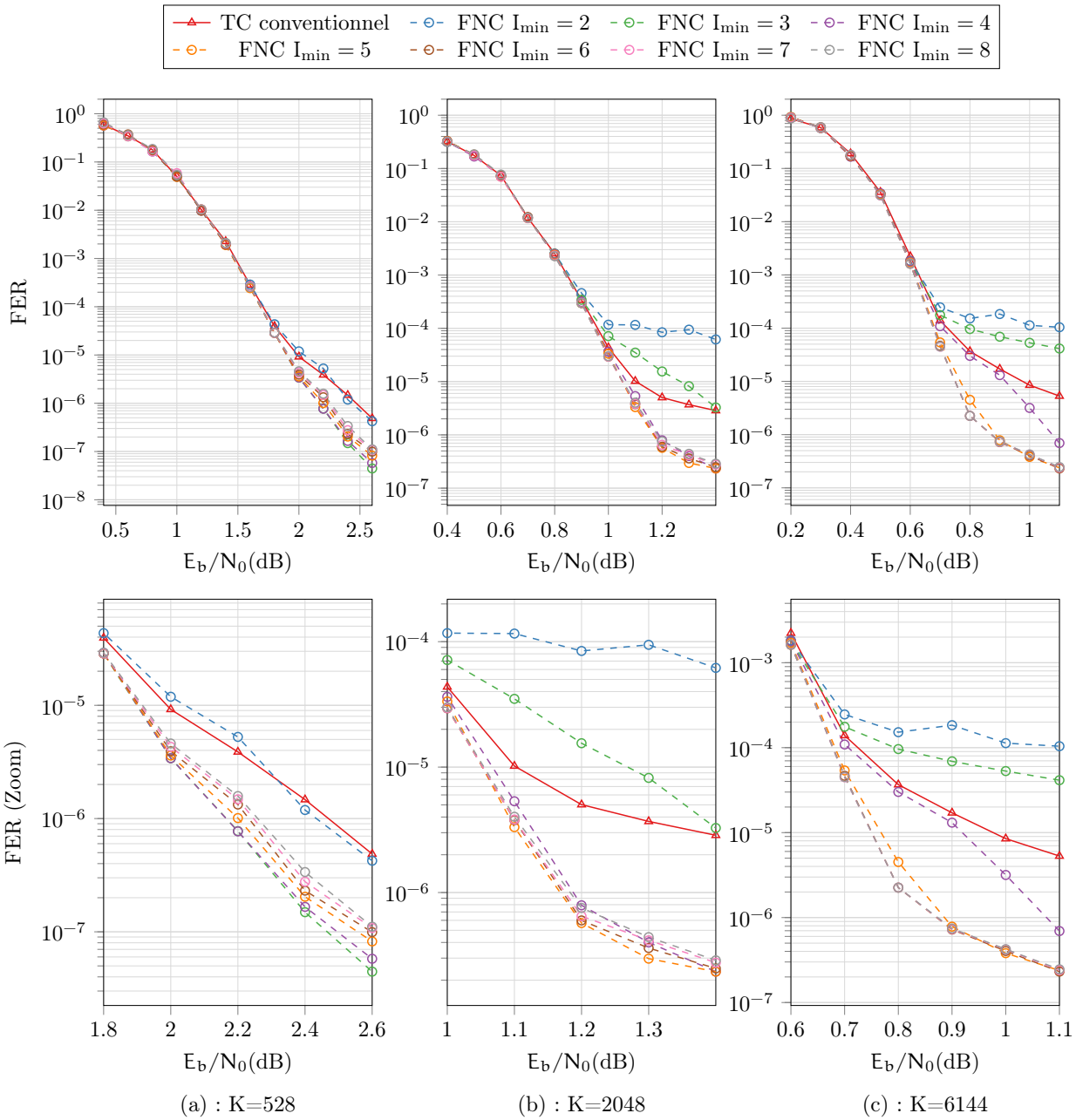
Vis-à-vis de l'algorithme FNC dans le cas binaire présenté dans le chapitre précédent (*cf.* Algorithme 4), une première modification est effectuée. Dorénavant, l'application de l'algorithme FNC n'est plus réalisée après la demi-itération dans le domaine entrelacé mais après la demi-itération dans le domaine naturel. De la sorte, il n'est plus nécessaire d'effectuer une opération de désentrelacement avant de pouvoir effectuer l'algorithme FNC. L'impact sur les performances de décodage est imperceptible mais cette modification permet une mise en œuvre plus naturelle.

4.2.1 Paramètres liés au processus itératif

Le premier ensemble de paramètres correspond au choix des itérations du processus de turbo décodage sur lesquelles le principe FNC peut s'appliquer. Il a été indiqué dans le chapitre précédent que si l'itération minimale à partir de laquelle l'algorithme est appliqué a une valeur trop faible, alors les performances de décodage peuvent être dégradées. Ceci dépend des erreurs non détectées par le code détecteur d'erreurs. Cette probabilité est fonction de la taille de trame et de la taille du code CRC. En effet, pour une taille du code CRC constante, la probabilité d'erreurs non détectées croît avec la taille de la trame.

La Figure 4.10 présente les performances de décodage de l'algorithme FNC avec $\mathsf{q} = 10$ en fonction de la valeur de l'itération à partir de laquelle le principe FNC est appliqué ($\mathrm{I_{min}}$), ce pour différents turbo codes du standard LTE de rendement 1/3 (K=528, K=2048 et K=6144). Dans ces trois cas, $\mathrm{I_{min}}$ évolue de 2 à 8. L'algorithme est donc appliqué respectivement de 7 à 1 seule fois. Il apparaît alors qu'il existe une valeur optimale de $\mathrm{I_{min}}$, offrant de meilleures performances de décodage. Elle s'établit à 3 pour K=528 et à 5 pour K=2048 et K=6144. Cette valeur est notée $\mathrm{I_{m_o}}$. En deçà de cette valeur, la trop forte occurrence d'erreurs non détectées lors de la vérification du code CRC dégrade les performances de décodage. Au delà de cette valeur $\mathrm{I_{m_o}}$, les performances de décodage s'amenuisent légèrement à chaque fois qu'elle est incrémentée.

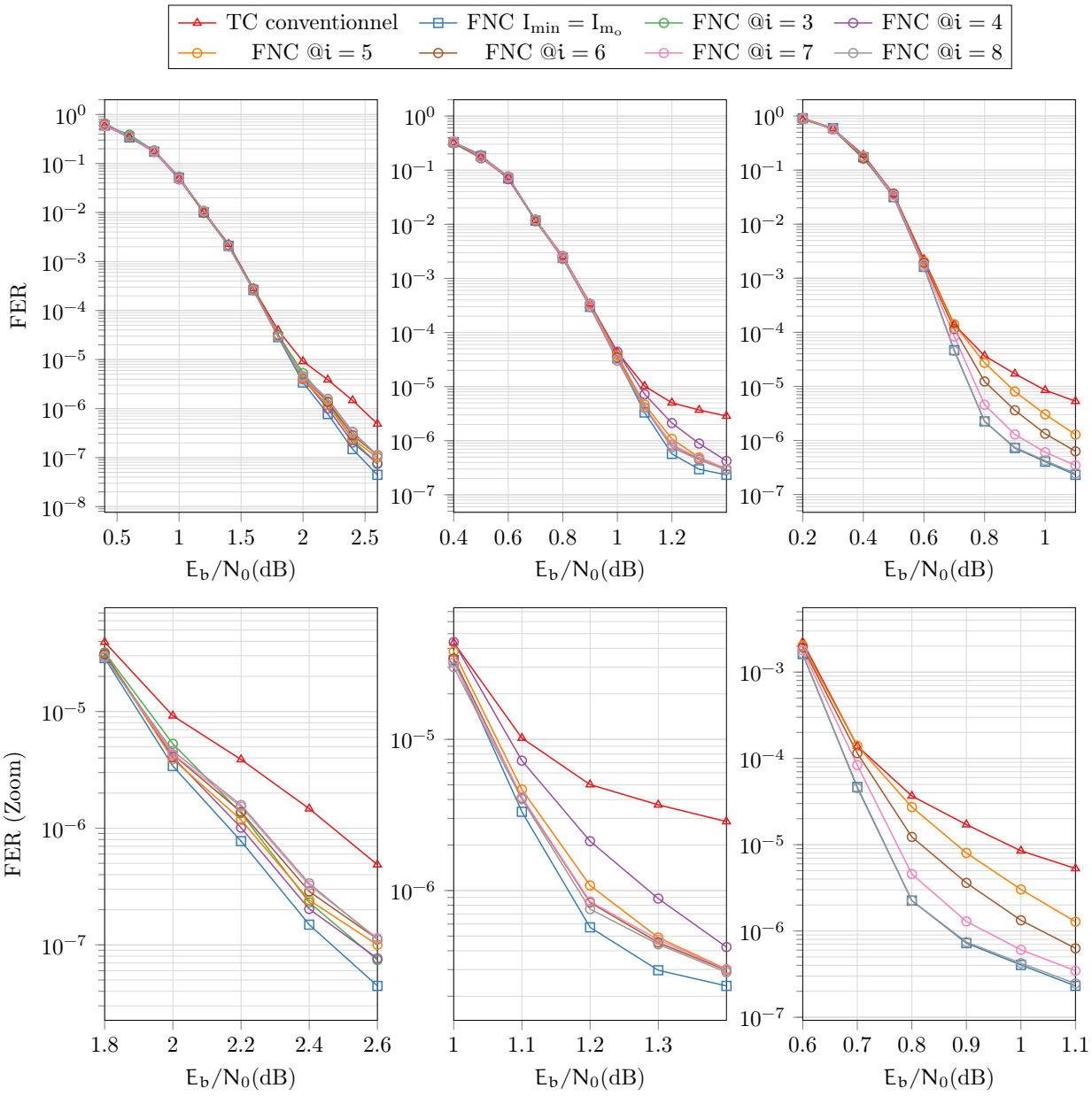
La Figure 4.11 présente les performances de décodage obtenues avec l'algorithme FNC lorsque celui-ci n'est appliqué qu'à une seule itération (noté $@\mathtt{i} = \mathtt{j}$) pour les trois turbo codes étudiés dans cette section. Les valeurs de $\mathtt{j}$ pour lesquelles les performances de décodage moins bonnes que celles obtenues avec l'algorithme de décodage conventionnel sont exclues. Dans les cas favorables, les performances de décodage sont également



(a) : K=528 (b) : K=2048 (c) : K=6144

FIGURE 4.10 – Comparaison des performances de décodage de l'algorithme FNC pour $q = 10$ et différentes valeurs de I_{min}. Turbo codes du standard LTE de rendement 1/3. Turbo décodeur basé sur l'algorithme EML-MAP itérant au plus 8 fois.

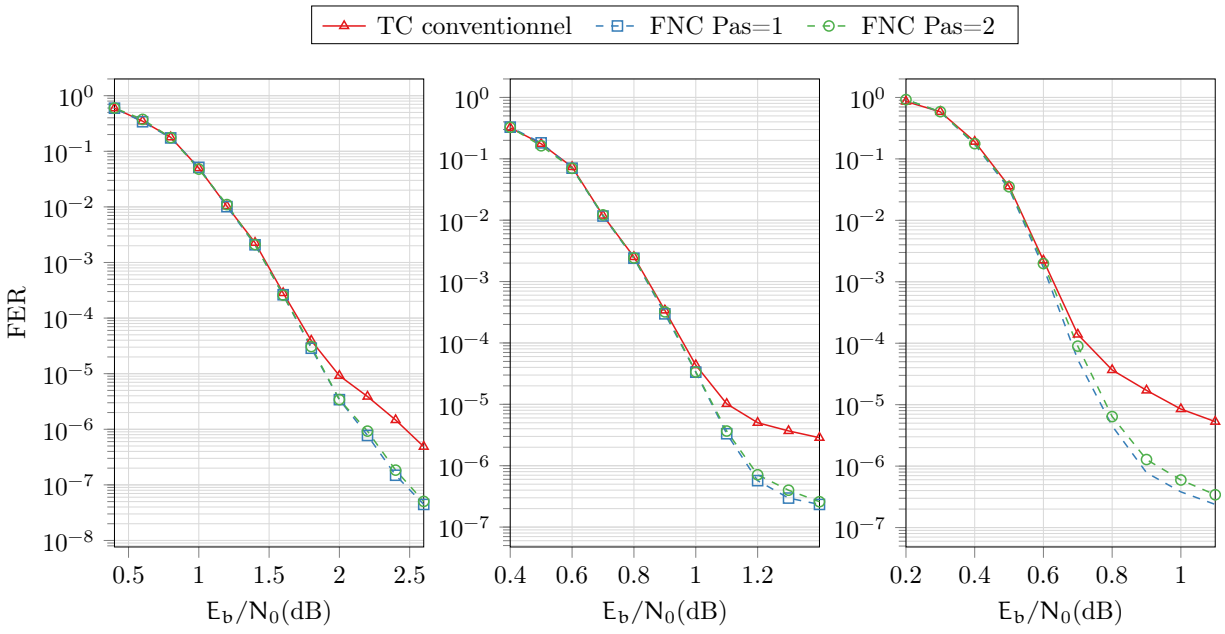
comparées à celles obtenues avec $I_{min} = I_{m_o}$. Il apparaît alors qu'appliquer une seule fois l'approche FNC lors du turbo décodage dégrade les performances en comparaison à une application à chaque itération à partir de $I_{min} = I_{m_o}$. Cependant, cette dégradation est minime puisqu'elle ne représente qu'un facteur deux en terme de FER dans le meilleur des cas.



(a) : K=528 (b) : K=2048 (c) : K=6144

FIGURE 4.11 – Performances de décodage de l'algorithme FNC pour $q = 10$ appliqué à une seule itération, en comparaison d'une application à toutes les itérations à partir de l'itération I_{m_o}. Turbo codes du standard LTE de rendement 1/3. Turbo décodeur basé sur l'algorithme EML-MAP itérant au plus 8 fois.

Finalement, il semble intéressant d'appliquer l'algorithme FNC qu'une itération sur deux. Ceci permet de disposer d'un budget temporel plus conséquent pour exécuter l'algorithme FNC. Le Figure 4.12 permet une comparaison des performances pour les turbo codes du standard LTE en considérant une application du principe FNC soit toutes les itérations soit toutes les deux itérations (noté Pas=2). Dans les deux cas, l'itération minimale est



(a) : K=528 (b) : K=2048 (c) : K=6144

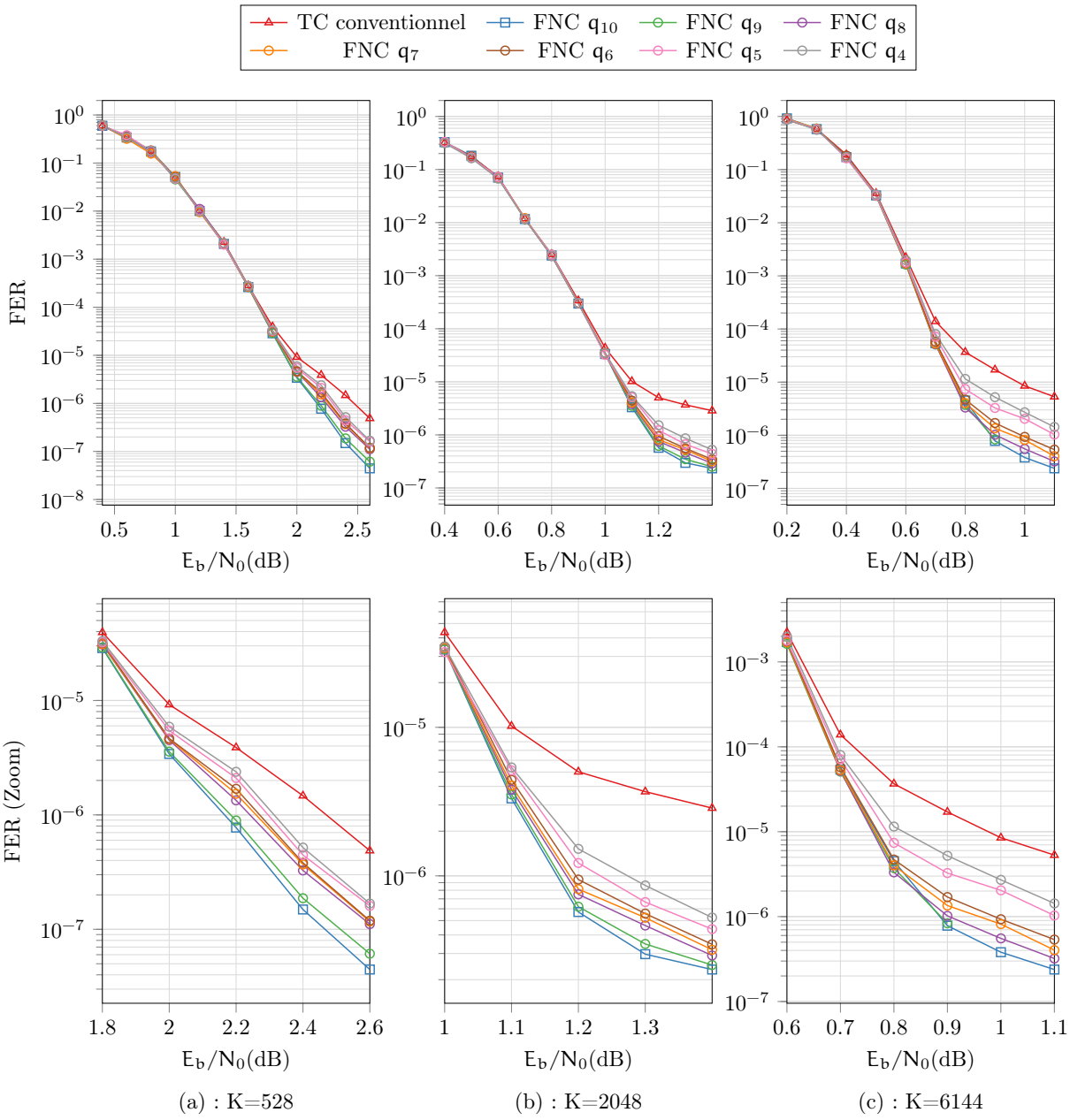
FIGURE 4.12 – Comparaison des performances de l'algorithme FNC appliqué toutes les deux itérations (Pas=2) ou toutes les itérations (Pas=1) à partir de l'itération I_{m_o} pour $q = 10$ et K=528, K=2048 ou K=6144. Turbo décodeur basé sur l'algorithme EML-MAP itérant au plus 8 fois.

fixée à I_{m_o}. Le nombre d'applications de l'algorithme est donc divisé par 2 par rapport aux références. Il est intéressant de constater que la dégradation de performances est minime.

En conclusion, il apparaît qu'une valeur trop faible de I_{min} dégrade les performances de l'algorithme. La valeur de l'itération I_{m_o} dépend des capacités de détection du code détecteur d'erreur. L'application de l'algorithme FNC après chaque itération de turbo décodage permet d'obtenir les meilleures performances de décodage. Chaque suppression dans l'application du principe FNC dégrade les performances de décodage. Néanmoins, ne pas appliquer à chaque itération l'algorithme mais seulement une itération sur deux permet d'augmenter significativement le budget temps pour l'exécution de l'algorithme tout en ayant un impact peu significatif au niveau des performances de décodage.

4.2.2 Variation du nombre de mots candidats considérés

Il a été introduit dans le chapitre précédent que les performances de l'algorithme FNC sont fortement conditionnées par la taille du vecteur de recherche des positions probablement erronées q. En effet, incrémenter q de 1 revient à considérer deux fois plus de mots candidats. Ceci augmente de fait la probabilité d'identifier le bon mot de code. Cependant, la complexité calculatoire double également pour chaque incrément de q. En effet, il est alors nécessaire de générer deux fois plus de mots candidats. Ceci implique un doublement du nombre de vérifications de CRC. Enfin, l'étape de tri est elle aussi impactée. Pour ces



(a) : K=528 (b) : K=2048 (c) : K=6144

FIGURE 4.13 – Performances de l'algorithme FNC appliqué toutes les itérations à partir de l'itération I_{m_o} pour différentes valeurs de q et K=428, K=2048 et K=6144. Turbo décodeur basé sur l'algorithme EML-MAP itérant au plus 8 fois.

raisons, il est nécessaire de minimiser la valeur de q

La Figure 4.13 présente l'impact de l'évolution de la valeur de q pour les turbo codes du standard LTE d'une taille de trame allant de K=528 à K=6144. Dans tous les cas, l'algorithme FNC est utilisé toutes les itérations à partir de l'itération $I_{min} = I_{m_o}$ de telle sorte que les performances de décodage soient les meilleures pour $q = 10$. Dans

l'ensemble, comme attendu, il apparaît que la réduction de q impacte négativement les performances de décodage. Cependant, cet impact dépend du turbo code considéré. Par exemple, pour $K = 528$, le passage de $q = 10$ à $q = 8$ a un impact plus défavorable que le passage de $q = 8$ à $q = 6$. En effet, la réduction de $q = 10$ à $q = 8$ dégrade d'un facteur 2 les performances de décodage en terme de FER. Cette dégradation n'est que d'un facteur 1,2 pour le passage de $q = 8$ à $q = 6$. Ceci s'explique par la distribution des erreurs binaires de ce turbo code. D'après la Figure 3.2 du chapitre précédent (représentant la distribution du nombre d'erreur dans la zone du plancher d'erreurs), 15% des trames erronées possèdent exactement 9 erreurs binaires. Elles ne peuvent donc être corrigées que pour $q \geqslant 9$. En revanche, pour $K = 2048$, la majorité des trames erronées contiennent strictement moins de 6 erreurs binaires. Il apparaît alors que le passage de $q = 10$ à $q = 6$ ne réduit que d'un facteur 2 les performances de décodage alors que 16 fois moins de mots candidats sont considérés. Enfin, pour K=6144, la situation est relativement similaire.

Finalement, il apparaît que la valeur de q proposant le meilleur ratio entre amélioration des performances de décodage dépend fortement du turbo code considéré. Plus la valeur de q est grande, meilleures sont les performances. Cependant, avoir des valeurs de $q > 9$ n'apportent que des gains marginaux pour les trois codes considérés. Ceci serait à nuancer dans le contexte d'un autre standard considérant un code CRC possédant de meilleures propriétés en terme de distance.

Le dernier vecteur de modifications de l'algorithme FNC en vue de son implémentation matérielle concerne la représentation de l'information en virgule fixe. Ceci est l'objet de la prochaine section.

4.2.3 L'algorithme FNC et la quantification de l'information

Jusqu'alors, les informations manipulées par le turbo décodeur utilisent une représentation en virgule flottante. Dans cette section, il est vérifié que l'algorithme FNC fournit des performances semblables lorsque la dynamique et la précision des données manipulées par le turbo décodeur est contrainte. Pour ce faire, une étude de la dynamique des métriques Δ est menée car elle impacte la complexité des opérations utilisées lors du tri.

Dans un premier temps, il est nécessaire de vérifier que l'algorithme FNC supporte le passage en virgule fixe des informations manipulées par le turbo décodeur. Pour ce faire, les performances de décodage sont comparées suivant le format des données internes aux turbo décodeurs. Afin de mener cette étude, l'outil de simulation de chaîne de communications numériques AFF3CT a été utilisé. [1] Cet outil open-source supporte de nombreuses configurations permettant notamment de répondre aux contextes du standard LTE. Le turbo décodeur logiciel décrit permet d'obtenir parmi les meilleurs débits de la littérature [117]. Cet outil de simulation de chaîne de communications numériques permet une liberté totale sur la quantification des données du canal. Dans la suite, la notation pour la quantification de l'information est $Q_{n.d}$, avec n le nombre de bits au total et d le nombre de bits utilisés pour coder la partie fractionnaire. Les formats de

1. AFF3CT : A Fast Forward Error Correction Tool est sous licence MIT et est téléchargeable depuis http://aff3ct.github.io. L'ensemble des propositions décrites dans ce manuscrit y sont intégrées.

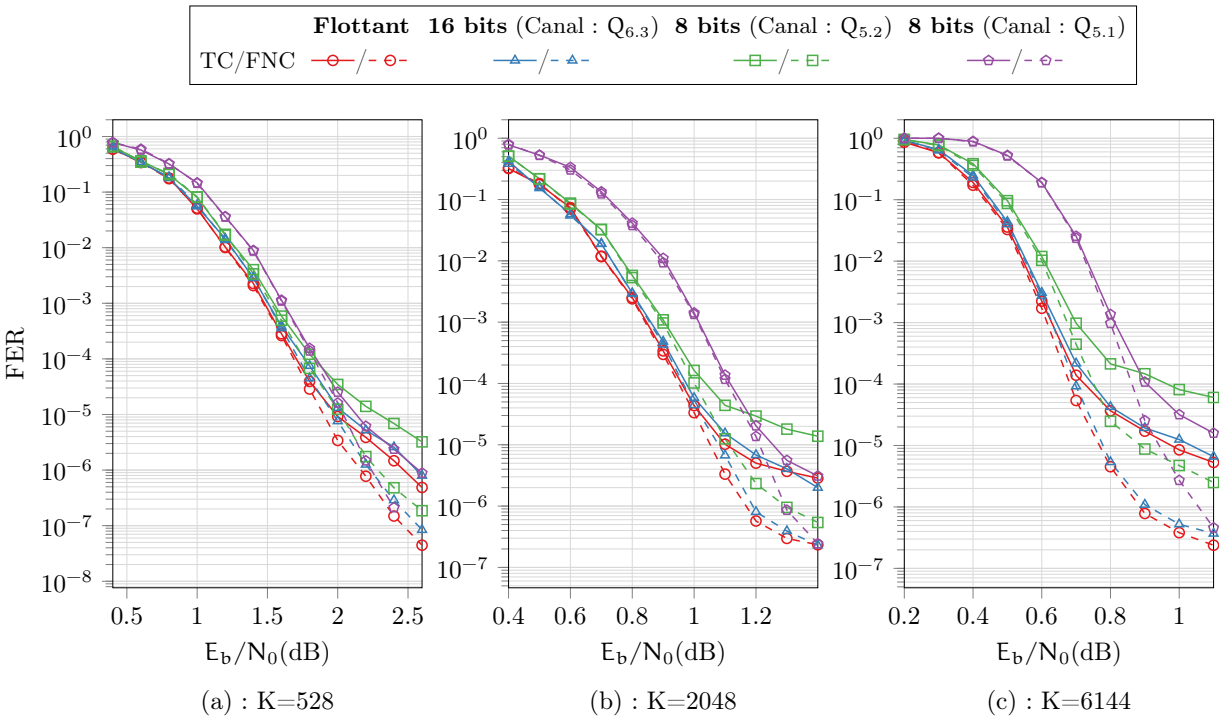
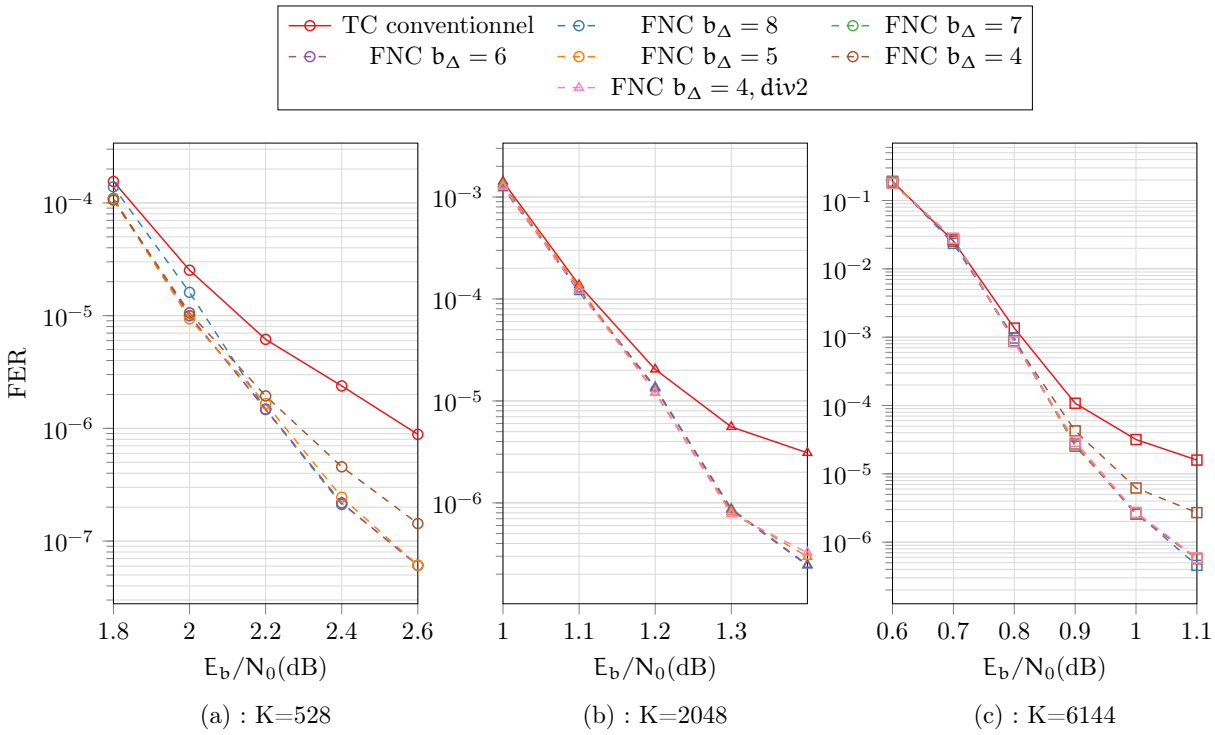


FIGURE 4.14 – Comparaison des performances de l'algorithme FNC $\mathsf{q} = 10$ suivant la représentation de l'information choisie, pour différents turbo codes du standard LTE. Turbo décodeur basé sur l'algorithme EML-MAP itérant au plus 8 fois.

données pouvant être considérés pour le turbo décodeur sont quant à eux limités à trois. Il s'agit soit d'une représentation flottante de l'information, soit d'une quantification des données internes sur 16 bits, soit enfin, d'une quantification sur 8 bits. Les performances de décodage sur 16 bits sont très proches de celles obtenues avec une représentation flottante de l'information. En revanche, en limitant la quantification des données à 8 bits, la quantification des données du canal à un impact fondamental sur les performances de décodage. En effet, si les données du canal utilisent 5 bits au total dont 2 pour la partie fractionnaire ($Q_{5.2}$), alors les performances de décodage dans la zone convergence sont proches de la représentation flottante. Cependant, le plancher d'erreurs apparaît plus tôt. Il est situé un ordre de grandeur au dessus. Dans le cas où les données du canal utilisent la représentation $Q_{5.1}$, la dégradation des performance de décodage se situe dans la zone de convergence et représente environ 0.2 dB. En revanche, pour les plus hautes valeurs de SNR considérées, les performances de décodage rejoignent celles obtenues dans le cas d'une quantification sur 16 bits. Ainsi avec cette dernière configuration, les performances dans la zone du plancher d'erreurs sont proches. Cette différence de performances s'explique par la nature itérative du processus de turbo décodage. Plus encore, il apparaît que la précision a un impact sur la convergence alors que la dynamique a un impact sur le plancher d'erreurs.

Les performances de décodage en activant ou non le principe FNC sont récapitulées en Figure 4.14 pour ces différents contextes. En ce qui concerne les performances de décodage obtenues avec l'algorithme FNC, il est remarquable que quelque soit le choix



(a) : K=528 (b) : K=2048 (c) : K=6144

FIGURE 4.15 – Performances de l'algorithme FNC $q = 10$ pour différents turbo codes du standard LTE, selon la quantification b_Δ de la métrique Δ. Turbo décodeur 8 bits, informations du canal sur 5 bits ($Q_{5.1}$), basé sur l'algorithme EML-MAP itérant au plus 8 fois.

de la représentation de l'information, les gains dans la zone de convergence atteignent ou dépassent un ordre de grandeur. Ainsi, il apparaît que l'algorithme FNC est robuste vis-à-vis d'une représentation en virgule fixe. Plus encore, les gains qu'il apporte ne sont pas conditionnés par les choix de quantification réalisés au sein du turbo décodeur.

Au sein de l'algorithme FNC, un tri des bits décodés est réalisé afin d'identifier les positions les moins fiables. La complexité calculatoire de ce tri est fonction de q. Cependant, la dynamique de Δ influe elle aussi sur la complexité calculatoire. En conséquence, réduire la dynamique de Δ doit permettre d'observer certains avantages.

Comme la métrique Δ de l'algorithme FNC est basée sur la valeur absolue d'une somme, sa dynamique et sa précision restent identique à celles des opérandes. Or, comme ce sont les valeurs minimales de la métrique Δ qui sont exploitées, il est possible de saturer la dynamique des données sans impacter les performances d'identification. Ceci est démontré en Figure 4.15. Dans tous les cas, le turbo décodeur manipule des données sur 8 bits et les données du canal sont sur 5 bits ($Q_{5.1}$). L'algorithme FNC est appliqué à chaque itération à partir de l'itération I_{m_o}. Le nombre de bits utilisés pour la métrique Δ, noté b_Δ varie de 8 à 4 bits. Il apparaît alors que lorsque b_Δ vaut 8, 7, 6 ou 5 bits les performances de décodage sont équivalentes pour les trois tailles de trame considérées. À partir de $b_\Delta = 4$ bits une dégradation des performances de décodage apparaît. Cette dernière peut être

réduite en effectuant une division par deux de la métrique avant d'effectuer la saturation.

En revanche dans le cas où 16 bits sont utilisées pour la dynamique des données internes au turbo décodeur, 6 bits sont nécessaires à la représentation de la métrique Δ afin d'obtenir des performances de décodage identiques à celles obtenues sans saturation.

4.2.4 Conclusion

Dans cette section, une étude de l'impact des différents paramètres de l'algorithme FNC a été menée. En effet, ces derniers ont pour la plupart un impact notable sur les performances de décodage et la complexité calculatoire de l'algorithme FNC.

La valeur de la profondeur de recherche $\mathtt{q}$ doit être adaptée par le concepteur d'un turbo décodeur utilisant l'algorithme FNC en fonction des performances de décodage visées conjointement à l'ajout de complexité calculatoire toléré. Il en est de même quant au choix des itérations de turbo décodage après lesquelles appliquer l'algorithme. Le compromis peut s'exprimer comme suit : « Quel est le surcoût calculatoire que je suis prêt à accepter pour abaisser le plancher d'erreurs. »

Toutefois, il apparaît qu'il existe une valeur optimale quant à l'itération minimale. Cette valeur dépend des capacités de détection du code détecteur d'erreurs et donc de la taille de la trame considérée. De même, il existe une valeur de $\Delta_{\mathtt{b}}$ permettant d'obtenir une réduction de la complexité de la fonction de tri tout en garantissant les performances de décodage. Cette valeur dépend essentiellement de la dynamique des informations *a posteriori*.

La section suivante présente une architecture matérielle pour l'algorithme FNC dans le contexte d'un turbo décodeur séquentiel basé sur l'ordonnancement Backward-Forward-Sliding-Window. L'impact de la variation du paramètre clef $\mathtt{q}$ sur la complexité matérielle est étudié.

4.3 Implémentation matérielle de l'algorithme FNC

Cette section présente une architecture matérielle pour l'algorithme FNC. Cette architecture se doit de s'adapter aux architectures matérielles de turbo décodeurs existants, afin d'abaisser le plancher d'erreurs des différents turbo codes présents dans les standards de communications numériques actuels. Cependant, il a été vu dans la première section de ce chapitre qu'une grande diversité dans les choix architecturaux pour les turbo décodeurs est possible. Cette section se focalise sur une architecture adaptée à l'ordonnancement BF-SW, correspondant à un cas de référence. Dans un second temps, des projections sont faites quant à l'adaptation de cette architecture matérielle à des turbo décodeurs plus complexes et plus rapides. Afin de mener cette étude architecturale, le choix est fait de considérer une application de l'algorithme FNC toutes les itérations. Il a été vu dans la section précédente que ceci permet d'obtenir les meilleures performances de décodage avec l'algorithme FNC.

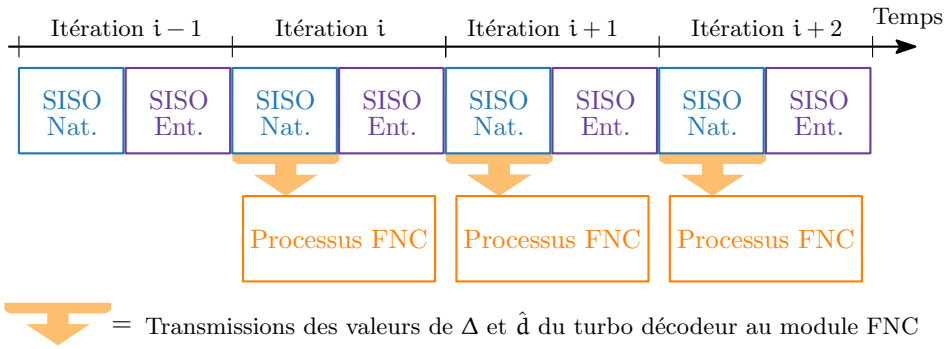


FIGURE 4.16 – Ordonnancement du système représentant les interactions entre le processus FNC et le turbo décodeur conventionnel.

4.3.1 Principe et ordonnancement de l'architecture matérielle développée

L'algorithme FNC est un algorithme de post-traitement en regard des calculs des décodeurs SISO. En effet, l'information *a posteriori* doit être obtenue pour pouvoir calculer la métrique Δ, qui est le point d'entrée de l'algorithme FNC. Cependant, durant les opérations de l'algorithme FNC, le processus de turbo décodage se poursuit en parallèle. Dans le cas d'un turbo décodeur exploitant un ordonnancement BF-SW, la durée d'une itération équivaut à $2 \times (K + K/W)$ cycles d'exécution (pour rappel, W est le nombre de fenêtres). Ainsi, si une application de l'algorithme FNC est considérée après chaque itération de turbo décodage, la durée de traitement FNC ne doit pas excéder $2 \times (K + K/W)$ cycles d'exécution. Afin d'être compatible avec n'importe quelle taille de fenêtre de traitement, la taille de fenêtre la plus petite possible doit être supportée. Cela revient à considérer le nombre maximal de fenêtres glissantes ($W \rightarrow K$). Ainsi, un temps d'exécution de $2 \times K$ cycles est visé. Ceci est représenté par la Figure 4.16. Dans ce scénario, l'architecture matérielle de l'algorithme FNC est sur-contrainte car un turbo décodeur utilisant l'ordonnancement BF-SW nécessite légèrement plus de $2 \times K$ cycles pour effectuer une itération de turbo décodage. Cependant, cette architecture matérielle de référence pourra être améliorée ultérieurement afin de tirer parti de ces quelques cycles d'exécution supplémentaires.

La Figure 4.17 présente l'ordonnancement des opérations réalisées par l'architecture matérielle associée à l'algorithme FNC. En accord avec la Figure 4.16, l'algorithme FNC est appliqué à partir de l'itération i. Durant les K premiers cycles d'exécution, le décodeur SISO fournit au bloc FNC la métrique Δ conjointement avec le mot décidé $\hat{d}$. Afin de conserver les q positions des décisions $\hat{d}$ les moins fiables, le tri des métriques Δ est réalisé à la volée lors de la réception de ces informations via un module nommé *trieur*. En effet, dans le cas de l'ordonnancement BF-SW, les K métriques Δ sont fournies séquentiellement durant K cycles d'exécution. Parallèlement, le mot décidé par le turbo décodeur $\hat{d}$ est mémorisé dans la *mémoire de bits décidés*. Ces données permettent de générer lors de l'étape suivante les 2^q mots candidats. La génération est réalisée lors des K cycles suivants par le *générateur de mots*. Ce dernier se base pour cela sur les bits mémorisés $\hat{d}$ ainsi que sur les q positions des minimums fournis par le module trieur. Les 2^q mots candidats sont produits à la volée et transmis aux *unités de vérification de CRC*. Finalement, au bout

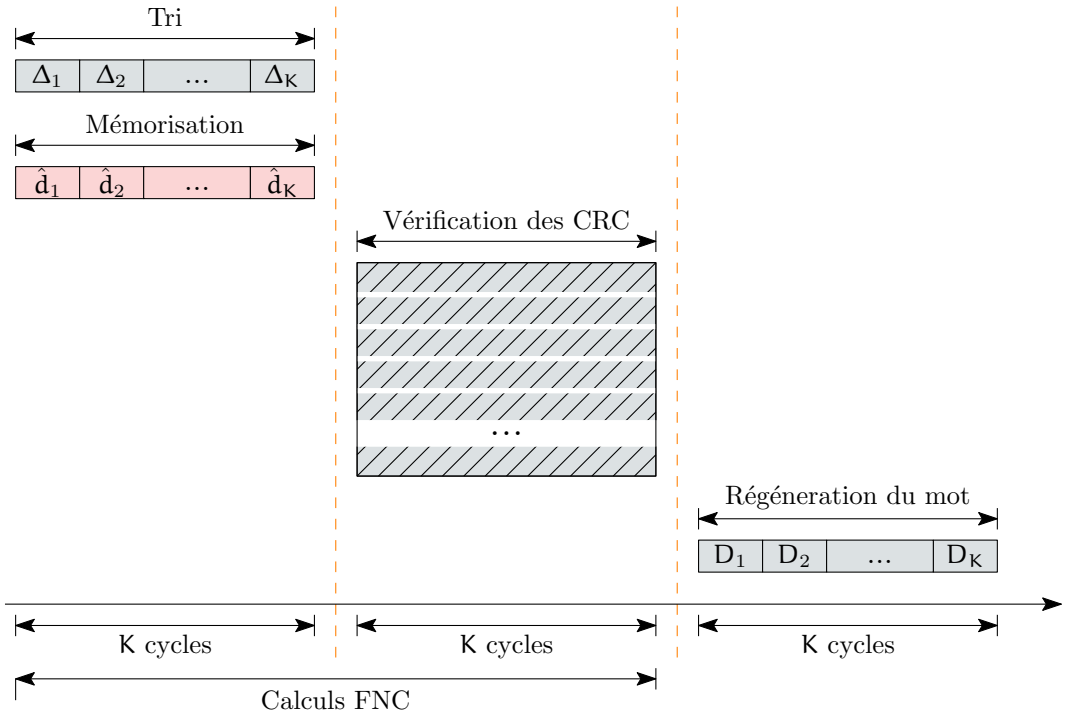


FIGURE 4.17 – Ordonnancement des opérations successives au sein de l'architecture FNC.

de K cycles, si un mot vérifiant le CRC est trouvé, ce mot est régénéré et constitue la sortie du bloc FNC. Cette régénération permet d'éviter le stockage des 2^q mots candidats. Ainsi, si un mot de code validant le code CRC est trouvé, le processus FNC requiert au total $3 \times K$ cycles d'exécution, après lesquels le processus de turbo décodage est stoppé. Le schéma bloc présentant l'interconnexion entre les différentes unités de cette solution architecturale est fourni en Figure 4.18.

4.3.2 Détail des unités composant l'architecture matérielle FNC

Dans cette sous-section, les différentes unités composant l'architecture matérielle pour l'algorithme FNC sont détaillées.

Mémoire des bits décidés Le principe de l'algorithme FNC est de générer $2^q 1$ mots candidats à partir du mot décidé à l'itération courante par le turbo décodeur $\hat{d}$. Cependant, cette génération n'est possible qu'après avoir effectué l'identification des q positions les moins fiables dans le mot décidé $\hat{d}$. C'est pourquoi ce dernier doit être mémorisé. Cette unité peut être implémentée au sein d'une mémoire simple port de taille K. Puisque les données en mémoire sont écrites et lues de manière ordonnée, il est aussi possible d'implémenter cette unité au sein d'un registre à décalage de profondeur K.

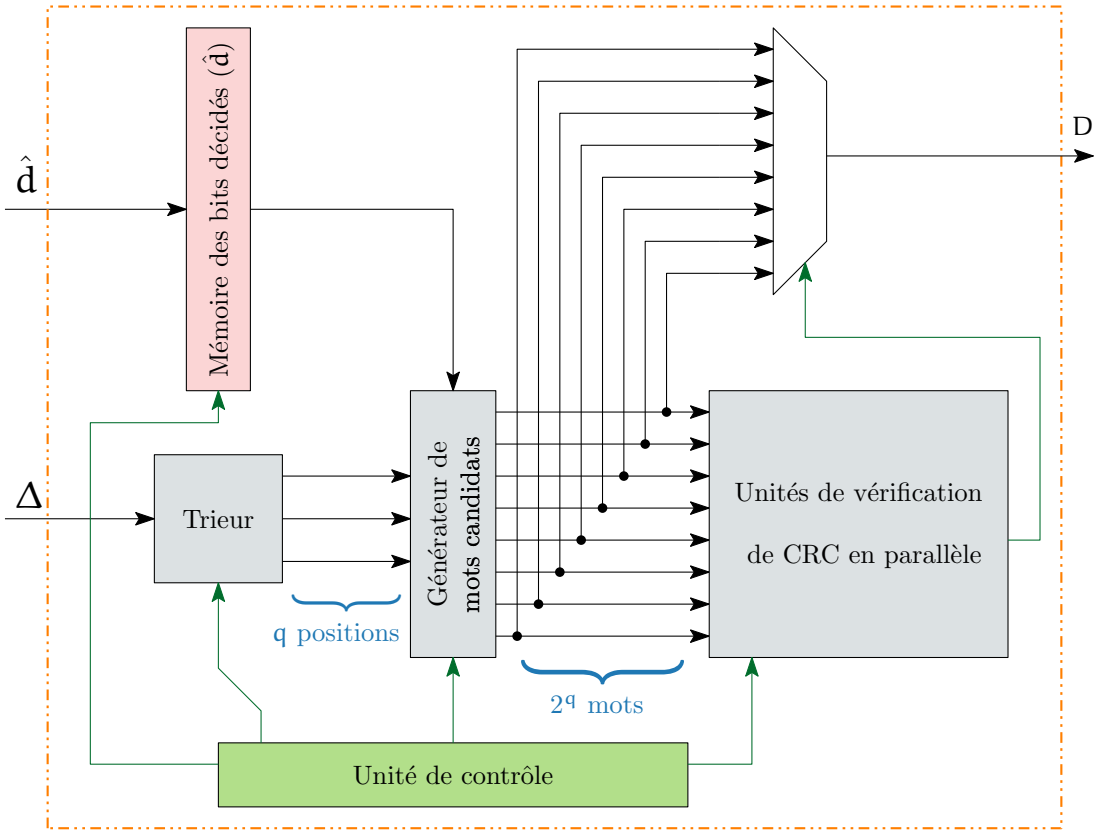


FIGURE 4.18 – Interconnexion des différentes unités de l'architecture FNC.

Trieur de métrique Δ De même que les décisions $\hat{d}_i$ du turbo décodeur, une nouvelle métrique Δ_i est fournie à chaque cycle d'exécution. Afin de trier ces métriques au fil de l'eau, un tri par insertion est utilisé. La nouvelle valeur de la métrique Δ est alors successivement comparée avec les valeurs correspondant aux q positions les moins fiables. Suivant le résultat de ces comparaisons, les valeurs de Δ et leur positions associées sont potentiellement mises à jour. La complexité matérielle de cette unité dépend donc à la fois de q et de la quantification de Δ (pour rappel, noté Δ_b). Il a été vu que 4 bits pour Δ_b permet d'assurer une très légère dégradation des performances de décodage. Finalement, pour l'implémentation de ce bloc, un effort de $q \times (\Delta_b + \lceil \log_2 K \rceil)$ bits de mémorisation est nécessaire. Les éléments de calcul consistent essentiellement en q comparateurs d'une taille de Δ_b bits interconnectés à un réseau de multiplexeurs.

Générateur de mots candidats Cette unité génère séquentiellement (bit à bit) tous les mots candidats en même temps, en combinant le mot décodé et les positions des bits les moins fiables. Pour ce faire, lorsque la position du bit courant coïncide avec une position minimisant la métrique Δ, alors la valeur de ce bit est inversée avant d'être transmise à l'unité de vérification des codes CRC (c'est-à-dire qu'un 1 devient un 0 et inversement). Ceci peut être implémenté en utilisant 2^q portes OU-EXCLUSIF dont une des entrées est commandée par un réseau de portes OU comparant la position courante

et les positions les moins fiables. Ainsi, la complexité matérielle est conditionnée par le choix de la valeur de q et $\log_2(K)$.

Unités de vérification de CRC en parallèle Puisque le choix a été fait d'assurer les vérifications des 2^q mots candidats en K cycles d'exécution, 2^q décodeurs de CRC séquentiels sont implémentés en parallèle. Conformément au polynôme générateur du code CRC du standard LTE (g_{CRC24A}), chaque détecteur CRC nécessite 24 D Flip-Flop et 13 portes ou-exclusif. En effet, un code CRC est basé sur un registre à décalage à rétroaction linéaire (LFSR) dont la boucle de retour consiste en des opérations ou-exclusif entre le bit d'entrée et les valeurs de certaines D Flip-Flop. La complexité calculatoire de cette unité dépend de q, suivant une loi exponentielle.

Unité de contrôle La dernière unité consiste en une machine à 3 états permettant de déterminer la phase courante du processus FNC. Pour rappel, celles-ci sont :

- la réception des données et le tri des métriques Δ,
- la vérification des différents mots candidats par les décodeurs de CRC et
- la régénération du mot de code valide.

Aussi, cette unité a pour rôle majeur de mettre à jour un compteur permettant à chaque unité d'avoir connaissance de la position courante dans le mot du bit en train d'être exploité.

La section suivante présente des résultats d'implémentations de cette architecture matérielle sur une cible FPGA (Field-Programmable Gate Array, ou réseau de portes programmables)

4.3.3 Résultats d'implémentation sur circuit FPGA

Des résultats d'implémentation après placement et routage sur circuit FPGA sont détaillés dans cette section. L'architecture matérielle présentée en Figure 4.18 a été décrite en VHDL. Cette description est générique pour les paramètres q, K et Δ. Cela signifie que l'architecture FNC peut être générée pour n'importe quelle taille de trame de turbo code binaire et n'importe quel nombre de positions les moins fiables identifiées. Dans tous les cas, l'architecture générée est conforme au bit près avec le modèle logiciel ayant permis d'obtenir les courbes de performances présentées au cours de ce chapitre.

Pour la plus grande valeur de K du standard LTE (K=6144) un seul bloc RAM est nécessaire pour stocker le mot décidé $\hat{d}$ au sein du circuit FPGA. Ainsi, l'architecture matérielle proposée ne nécessite qu'un seul bloc RAM, ce quelle que soit la taille de trame considérée. Dans la suite, seul le cas K=6144 est considéré car il correspond au cas le plus complexe.

La valeur de Δ_b est fixée à 4 bits dans un premier temps. Cette valeur permet de ne pas avoir de dégradation de performances de décodage pour un turbo décodeur manipulant en interne des données quantifiées sur 8 bits. Ce format de quantification permet de

TABLEAU 4.2 – Résultats après Placement-Routage pour l'architecture associée à l'algorithme FNC K=6144, $\Delta_b = 4$ et différentes valeurs de q pour le circuit FPGA XILINX VIRTEX-6 XC6VLX75T-1.

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
5	849	940	1038	1	230
6	1438	1725	1844	1	216
7	2302	3278	3127	1	207
8	4283	6267	5731	1	172
9	7970	12528	10850	1	132
10	15193	24833	23720	1	103

minimiser le chemin critique de la fonction de trie de la métrique Δ. Ceci sera conforté par des données concernant la fréquence maximale d'exécution.

Les performances de décodage de l'algorithme FNC dépendent de la valeur du paramètre q. Il en est de même pour la complexité matérielle de l'architecture décrite. Le tableau 4.2 présente les résultats d'implémentation pour une cible matérielle XILINX VIRTEX-6 XC6VLX75T-1 pour plusieurs valeurs de q. Ces résultats ont été obtenus grâce à l'outil ISE 14.7. Les paramètres de l'outil pour les options de synthèse logique et de placement-routage sont celles par défaut afin de faciliter la reproductibilité des résultats. La Figure 4.19 présente une partie de ces résultats de façon graphique.

Comme attendu, il apparaît que le nombre de ressources matérielles nécessaires croît exponentiellement avec q. A chaque incrément de q, les ressources sont quasiment multipliées par 2. En ce qui concerne la fréquence maximale atteignable par cette architecture, elle décroit lorsque le nombre de mots candidats augmente. Ainsi, elle varie entre 103 MHz pour $q = 10$ et 293 MHz pour $q = 4$. Cette réduction de la fréquence s'explique par l'augmentation de la complexité du tri par insertion et donc du chemin critique attenant. En effet, cette opération de tri doit être réalisée en 1 cycle d'horloge. Aussi, en sortie du CRC, une RÉDUCTION-EN-OU entre les sorties de toutes les unité de CRC a lieu. De part l'augmentation de la profondeur de cette réduction avec l'augmentation de q, le chemin critique se voit augmenté puisque cette opération est elle aussi exécutée en un cycle d'horloge.

Nous allons considérer le cas $\Delta_b = 6$ pour pouvoir observer les meilleures performances de décodage avec un turbo décodeur utilisant une dynamique interne des données allant jusqu'à 16 bits. Les résultats de synthèse logique sont récapitulés dans le tableau 4.3. Dans ce cas, il apparaît que le nombre de ressources requises augmente légèrement. La fréquence maximale de fonctionnement baisse de 3 à 26 MHz en fonction de la valeur de q. Une fois de plus, cela s'explique par l'augmentation de la complexité dans le tri. Comparer des données sur 6 bits implique d'augmenter la largeur des comparateurs et multiplexeurs. En revanche, dans le cas $q = 10$, la fréquence augmente de 8 MHz. L'explication de cette

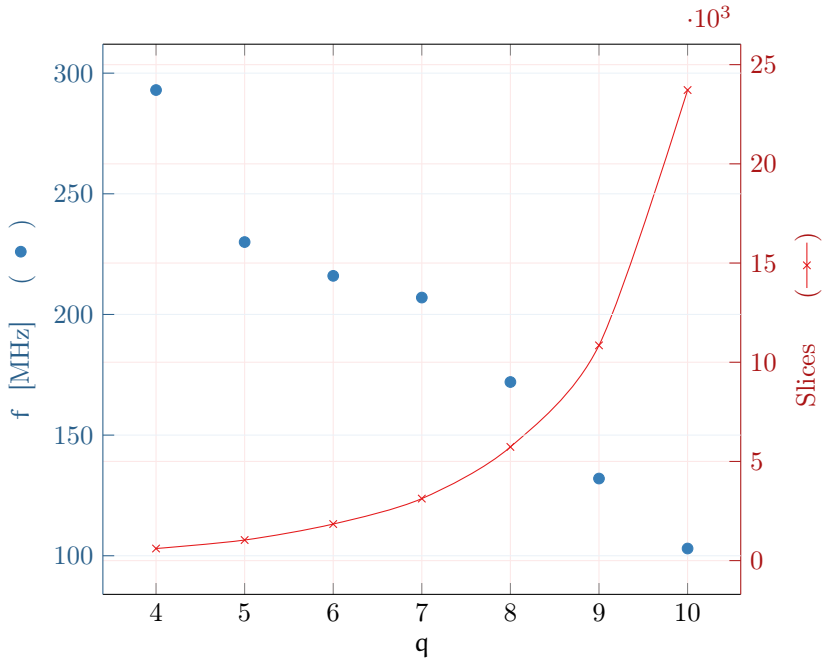


FIGURE 4.19 – Évolution de la fréquence maximale de fonctionnement et du nombre de Slices utilisées en fonction de q. Courbes réalisées en fonction des données du tableau 4.2.

particularité réside sans doute dans les heuristiques internes à l'outil Xilinx ISE.

La répartition des ressources assignées selon les différentes unités (trieur, mémorisation et vérifications de CRC) est aussi fonction de la valeur de q. En effet, la seule unité dont la complexité croît de manière exponentielle avec q est l'unité de calcul des CRC. De fait, plus q prend une valeur élevée, plus cette unité devient prépondérante dans l'architecture. Ceci est récapitulé dans le tableau 4.4 dont les données ont été obtenues pour une cible technologique de type ASIC tout en gardant la hiérarchie de l'architecture. Ainsi, pour le cas $q = 10$ la quasi totalité des ressources est utilisée pour l'unité de calcul des CRC.

TABLEAU 4.3 – Résultats après Placement-Routage pour l'architecture associée à l'algorithme FNC (K=6144, $\Delta_b = 6$ et différentes valeurs de q pour le circuit FPGA XILINX VIRTEX-6 XC6VLX75T-1).

q	LUT	FF	Slices	BRAM	f (MHz)
4	543	547	641	1	267
5	809	950	1008	1	227
6	1465	1737	1834	1	248
7	2325	3292	3155	1	190
8	4248	6383	5973	1	163
9	8099	12546	10930	1	125
10	15432	24853	23135	1	111

Tableau 4.4 – Répartition des ressources de l'architecture matérielle selon les différentes unités suivant la valeur de q. Technologie ST 130nm

q	Trieur	Mémorisation	Calculs de CRC	Contrôle
4	21%	6%	67%	16%
6	9%	2%	74%	8%
10	1%	0.2%	93%	5%

Puisque l'algorithme FNC peut être vu comme une extension d'un turbo décodeur, les résultats d'implémentation sont comparés avec ceux d'un turbo décodeur de la littérature. Ce dernier est disponible auprès de Xilinx sous forme d'un IP Core. De la sorte, il est possible d'estimer le surcoût matériel induit par l'algorithme FNC. Le tableau 4.5 reporte les ressources requises pour l'implémentation de ce turbo décodeur [112] sur la même cible FPGA que celle utilisée dans notre étude. Il apparaît que pour un contexte $\mathsf{q} = 6$, l'architecture matérielle associée à l'algorithme FNC présente un surcoût matériel d'environ 50%. Au niveau des performances de décodage, avec $\mathsf{q} = 6$ un ordre de grandeur est approché dans la majorité des cas pour le standard LTE. Si toutefois l'obtention de meilleures performances de décodage est visée, l'implémentation pour $\mathsf{q} = 9$ va jusqu'à tripler le nombre de Slices. En terme de mémorisation, l'architecture matérielle n'a qu'un impact minime dans tous les cas puisqu'il représente moins de 10% de surcoût. Il est cependant à noter qu'en l'état, cette architecture est limitante au niveau de la fréquence maximale de fonctionnement. Cependant, elle est sur-contrainte vis-à-vis de l'architecture de référence [112]. En effet, dans le cas le plus favorable, cette dernière effectue une itération du processus de décodage en $2{,}2 \times \mathsf{K}$ cycles. Or, l'architecture FNC a été contrainte de telle sorte à fournir le résultat d'un CRC en $2 \times \mathsf{K}$ cycles. Il est donc possible de revoir l'architecture afin de ne plus être limité en fréquence par le bloc FNC.

Une architecture matérielle pour l'algorithme FNC pouvant s'adapter à tout turbo décodeur basé sur un ordonnancement BF-SW avec une taille de fenêtre aussi petite que possible a été présentée. Cette architecture permet d'obtenir les meilleures performances de décodage atteignables avec l'algorithme FNC dans le contexte du standard LTE. Ce travail a fait l'objet d'une publication en conférence internationale avec acte [118] ainsi qu'en conférence nationale [119]. Il est possible de réduire le nombre d'itérations sur lesquelles est appliqué l'algorithme FNC tout en garantissant des performances de décodage similaires (Figure 4.12). Ainsi, en appliquant l'algorithme FNC toutes les deux itérations et en modifiant la machine d'états de l'architecture, le calcul des différents CRC peut s'effectuer durant 1,5 itérations. Ceci permettrait une réduction par trois de la surface requise pour l'implémentation de l'unité de calcul des CRC, prépondérante dans

Tableau 4.5 – Résultats de synthèse de l'architecture matérielle de turbo décodeur sur cible FPGA Xilinx Virtex-6 xc6vlx75t-1

LUT	FF	Slices	BRAM	f (MHz)	T/P (Mbps)
3712	4062	3765	13	285	~ 17 ($\mathrm{I_{TC}} = 8$)

le coût matériel de cette architecture. Dans la suite de cette section, des projections quant à la transposition de cette architecture à d'autres ordonnancements de turbo décodage sont présentées.

4.3.4 Projections sur d'autres ordonnancements de turbo décodeurs

L'architecture matérielle détaillée dans la sous-section précédente est adaptée à un turbo décodage nécessitant 2K cycles par itération et produisant les informations *a posteriori* durant K cycles à raison d'une donnée par cycle. Ceci correspond à un ordonnancement BF-SW avec une taille de fenêtre la plus petite possible. Cette section présente différentes adaptations selon l'ordonnancement de turbo décodage retenu. Pour rappel, le tableau 4.1 récapitule les différents ordonnancements de turbo décodage.

Adaptation pour un ordonnancement de type Butterfly (BFLY) Dans ce cas, la durée d'une itération vaut toujours $2 \times \mathsf{K}$. Cependant, deux données *a posteriori* sont fournies en même temps durant K/2 cycles. Il est alors nécessaire de scinder la mémoire de bits décidés (Figure 4.18) en deux bancs complémentaires. Ceci permet de mémoriser deux données en parallèle : l'une s'effectuant dans l'ordre ascendant et l'autre selon l'ordre descendant des positions. Pour la même raison, l'unité de tri se doit d'être dupliquée. L'une d'elle a pour rôle d'obtenir les q positions les moins fiables sur l'intervalle $[\![1; \mathsf{K}/2]\!]$. L'autre effectue parallèlement la même opération sur l'intervalle $]\!]\mathsf{K}/2; \mathsf{K}]\!]$. De ces deux sous-ensembles de positions les moins fiables de taille q, il est nécessaire de les combiner en un seul ensemble de taille q. Il faut alors effectuer un troisième tri de q positions parmi $2\mathsf{q}$. Cependant, comme les 2 listes de q positions sont triées, ce tri supplémentaire peut être réalisé en seulement ϵ cycles d'horloge, avec $1 \leqslant \epsilon \leqslant \mathsf{q}$ en fonction des ressources matérielles mises en œuvre. Dans le cas $\epsilon = \mathsf{q}$, ce tri ne nécessite qu'un seul comparateur de taille Δ_b bits commandant un multiplexeur à deux entrées de $\lceil \log_2(\mathsf{K}) \rceil$ bits ainsi que la mémorisation des q positions (chacune de $\lceil \log_2(\mathsf{K}) \rceil$ bits). En ce qui concerne l'unité de vérification des CRC, aucune modification n'est nécessaire pour son utilisation dans le contexte de l'ordonnancement BFLY. En effet, après les étapes de tri, $\frac{3\mathsf{K}}{2}$ cycles d'exécution sont disponibles avant que de nouveaux couples de données Δ et $\hat{\mathsf{d}}$ ne soient produits par le turbo décodeur. Finalement, uniquement une modification des unité de tri (impliquant aussi une légère modification de la machine d'états) de l'architecture de référence est nécessaire pour une adaptation à un turbo décodage basé sur un ordonnancement de type Butterfly. Dans ce cas, la Figure 4.20 récapitule les ressources nécessaires ainsi que l'ordonnancement des opérations pour une telle architecture matérielle associée à l'algorithme FNC.

Adaptation pour un ordonnancement de type Butterfly avec fenêtre glissante (BFLY-SW) Dans le cas de l'utilisation d'une fenêtre glissante, un tri supplémentaire de q parmi $2 \times \mathsf{q}$ est nécessaire afin de réunir les q positions identifiées sur chacune des fenêtres. La machine d'état doit à nouveau être modifiée pour ce contexte. La modification consiste en l'ajout d'un état pour cette fusion de listes ordonnées. De la sorte, une alternance a lieu entre l'état où les deux tris en parallèle sont exécutés et l'état permettant

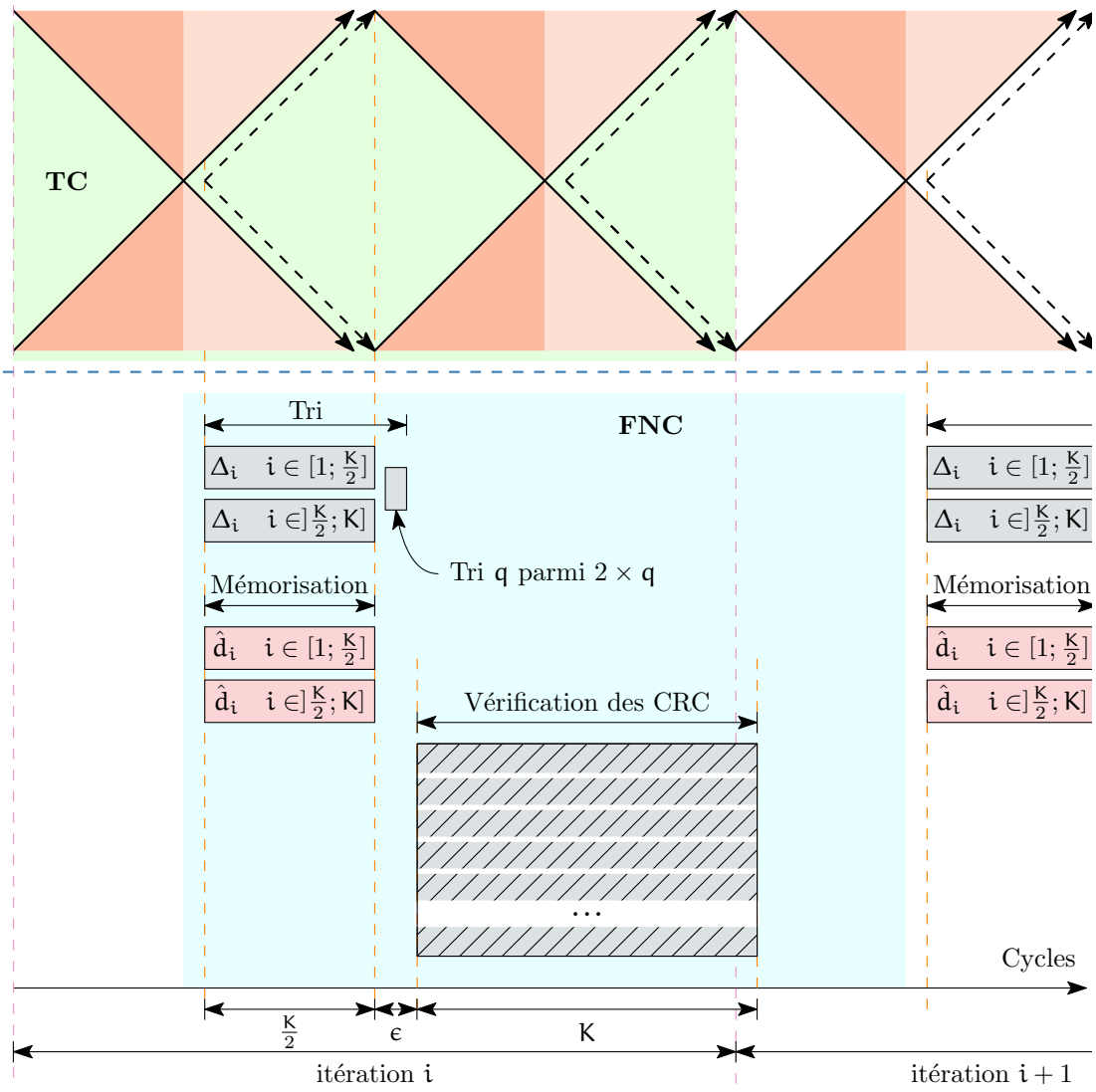


FIGURE 4.20 – Adaptation de l'ordonnancement de l'architecture FNC à un turbo décodage basé sur un ordonnancement de type Butterfly.

la réunion des deux listes ordonnées en une seule. L'alternance entre les deux états doit être opérée W fois, W étant le nombre de fenêtres glissantes considérées. En ce qui concerne, la vérification des CRC, K cycles d'exécution sont disponibles. Ainsi, aucune modification n'est nécessaire dans le cas d'un turbo décodage basé sur l'ordonnancement BFLY-SW. La durée durant laquelle sont produites les informations *a posteriori* est, comme pour l'ordonnancement BFLY, de $\frac{K}{2}$ cycles d'exécution. Cependant, la production s'étale sur $K - \frac{K}{2W}$ cycles. Ceci a pour effet de réduire à $\frac{K}{W}$ le nombre de cycles supplémentaires disponibles. De fait, la montée en fréquence de l'architecture pourrait être très contrainte dans les cas considérant des fenêtres de traitement de petite taille.

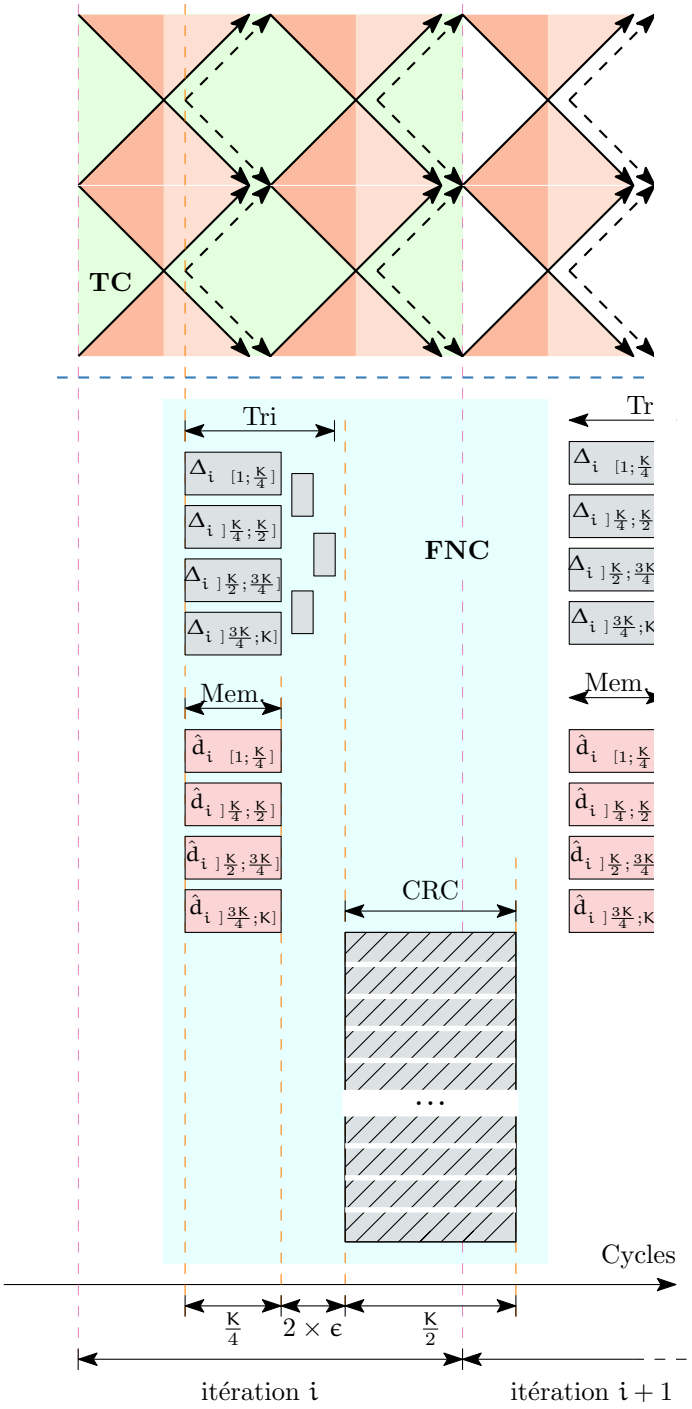


FIGURE 4.21 – Adaptation de l'ordonnancement de l'architecture FNC à un turbo décodage basé sur un ordonnancement de type Butterfly avec un degré de parallélisme de 2 (BFLY-SB).

Adaptation pour un ordonnancement de type Butterfly avec parallélisme de sous-bloc (BFLY-SB) Un parallélisme de degré B est obtenu grâce à une multipli-

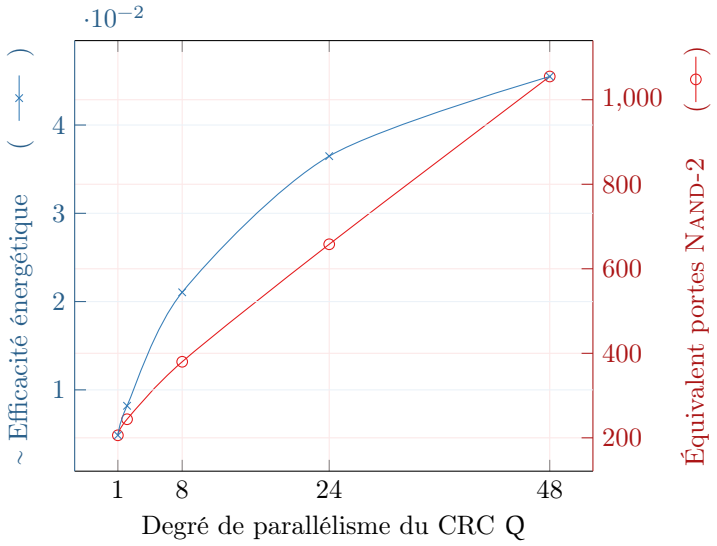


FIGURE 4.22 – Évolution du coût matériel de l'implémentation d'un calcul de CRC suivant le degré de parallélisme. Résultats de synthèses logique en équivalent portes NAND-2. Technologie ST 130nm. Fréquence de 500MHz.

cation par B des ressources matérielles. Cela a alors pour effet de réduire la latence de traitement du même facteur. Le nombre des unités de tri et de mémorisation doit être lui aussi multiplié par B puisque $2 \times B$ données *a posteriori* sont fournies simultanément. Pour assurer les vérifications de CRC, il est alors nécessaire de paralléliser les unités de traitement de CRC. En effet, celles-ci doivent être capables de traiter les B bits de la trame durant un cycle d'exécution. La valeur maximale usuelle de B trouvée dans la littérature est 16. La Figure 4.22 présente les résultats de synthèse logique en équivalent porte (cible ASIC) pour une unité de vérification CRC du standard LTE. Cinq degrés de parallélisme (de 1 à 48) sont étudiés. Il apparaît alors que le coût de l'unité de calcul de CRC croît mais l'incrément du degré de parallélisme ne correspond pas à un doublement du coût matériel. Ceci est corroboré par une métrique qui correspond au calcul du degré de parallélisme divisé par le nombre équivalent de portes NAND-2, comme présenté dans la Figure 4.22. Cette métrique est une indication de l'efficacité énergétique. Ce résultat ouvre l'opportunité d'une modification intéressante de l'architecture FNC de référence. En effet, par exemple, une unité séquentielle de CRC nécessite 203 portes NAND-2. Pour le cas $q = 10$, l'unité de vérification de CRC requiert alors 207669 portes NAND-2. Une unité de calcul de CRC d'un degré de parallélisme de 48 nécessite 1055 portes NAND-2. Une telle unité nécessite $K/48$ cycles d'exécution afin de vérifier un CRC. Afin d'effectuer les 1023 calculs de CRC, en exactement K cycles d'exécution, il est alors nécessaire d'implémenter seulement 22 unités de calcul de CRC d'un degré de parallélisme de 48. Ceci permet alors de n'utiliser plus que 23210 portes NAND-2 pour implémenter l'unité de vérification des CRC. De la sorte, l'empreinte de cette unité peut être réduite d'un facteur 9. La Figure 4.21 présente les ressources nécessaires ainsi que l'ordonnancement des opérations de l'architecture adaptée à un turbo décodeur de type BFLY-SB pour un parallélisme de 2 ($B = 2$).

TABLEAU 4.6 – Récapitulatif des modifications requises par rapport à l'architecture de référence afin de l'adapter à d'autres ordonnancements de turbo décodeurs.

	Unité de Tri	Mémorisation	Unité CRC	Machine d'états
BFLY	$\times 2$ + 1 fusion	2 bancs	–	Tri et mémorisation
BFLY-SW (W)	$\times 2$ + 2 fusions	2 bancs	Idem BFLY	Idem BFLY + tri
BFLY-SB (B)	$\times 2\mathsf{B}$ + (B-1) fusions	B$\times$ BFLY	Parallélisme (B)	tri et mémorisation

Adaptation pour un ordonnancement de type Butterfly à fenêtre glissante et avec parallélisme de sous-bloc (BFLY-SW-SB) Cet ordonnancement de turbo décodage, le plus complexe étudié dans ce chapitre consiste en la combinaison des deux ordonnancements précédents. Une architecture matérielle adaptée à un tel turbo décodage doit alors adresser toutes les modifications précédemment introduites. Ainsi, le nombre de tri à effectuer en parallèle est de $2 \times \mathsf{B}$. De ces $2 \times \mathsf{B}$ listes de q positions, $2 \times \mathsf{B} - 1$ opérations de fusion doivent être réalisées afin de ne garder que les q positions les moins fiables de la trame. Chaque unité de CRC doit être adaptée à un degré de parallélisme de B afin d'effectuer l'ensemble des vérifications de CRC en $\frac{\mathsf{K}}{\mathsf{B}}$ cycles. Enfin, le nombre de cycles entre la fin des opérations de CRC et le début du prochain tri sur le nouveau jeu de données doit valoir $\frac{\mathsf{K}}{2 \times \mathsf{B} \times W}$.

Conclusion Dans cette sous-section les modifications à apporter à l'architecture FNC de référence afin de l'adapter à différents ordonnancements de turbo décodage ont été introduites. À la vue de ces modifications envisageables, il apparaît possible d'adapter l'architecture FNC aux différents ordonnancements de turbo décodage. Ceci montre la polyvalence de l'algorithme FNC quant à son implémentation. Cependant, ces synthèses ainsi que les résultats restent à mener. Le tableau 4.6 récapitule les différentes projections sur l'architecture matérielle suivant le turbo décodage considéré.

4.4 Conclusion

Dans ce quatrième chapitre, une architecture matérielle de correction des erreurs résiduelles a été détaillée. Afin d'aboutir à cette proposition, différents ordonnancements de turbo décodage ont été présentés dans un premier temps. Ces ordonnancements, séquentiels ou parallèles, peuvent adresser un large panel applicatif répondant à différents contextes.

Dans un second temps, une étude quant à l'influence des paramètres de l'algorithme FNC a été menée. Il apparaît que ceux-ci interagissent à la fois sur la complexité calculatoire et sur les performances de turbo décodage. Certains de ces paramètres possèdent des valeurs optimales alors que d'autres doivent être adaptés selon un compromis entre performances de décodage et complexité matérielle résultante.

Finalement, la troisième section présente le cas d'étude d'une architecture matérielle pour l'algorithme FNC, adaptée à un ordonnancement de turbo décodage particulier. Le principe, l'ordonnancement et la composition de cette architecture ont été détaillés.

S'en est suivi des résultats d'implémentation pour une cible FPGA. Ces résultats ont été comparés à un turbo décodeur vendu sous forme d'IP Core par Xilinx. Finalement, des projections quant à la transposition de ce cas d'étude sur d'autres types de turbo décodeur ont été menées. Ceci conforte le fait qu'une implémentation matérielle de l'algorithme FNC est envisageable ce quelque soit le turbo décodeur considéré ; afin de permettre une correction des erreurs résiduelles et donc un abaissement du plancher d'erreurs des turbo codes.

Conclusions et perspectives

Conclusions

Dans un contexte de communications numériques, le besoin d'obtenir des taux d'erreurs aussi faibles que possible devient primordial pour de nombreux domaines applicatifs. Ces travaux de thèse se sont alors concentrés sur l'amélioration des performances de décodage des turbo codes et notamment la réduction de leurs planchers d'erreurs. Pour ce faire, différentes techniques associées au turbo décodage et des solutions architecturales adaptées ont été proposées.

Dans le deuxième chapitre, des observations statistiques ont permis de mettre en exergue une corrélation forte entre l'occurrence d'oscillations de métriques internes au processus de turbo décodage et la présence d'erreurs à l'issue du décodage. Cependant, aucune relation d'implication directe entre les occurrences d'oscillations et les erreurs n'a pu être trouvée. Ceci rend alors délicat l'identification de la position des erreurs à partir de l'observation des oscillations. Néanmoins, un algorithme, nommé SC EML-MAP, a été proposé afin d'améliorer la pertinence de l'information extrinsèque échangée au cours du processus itératif. Cet algorithme est inspiré d'une approche originellement proposée dans le cadre du décodage des codes LDPC. Dans le cas de turbo codes binaires, sous couvert d'un nombre maximal d'itérations important, le seuil de convergence est amélioré par rapport à celui obtenu avec l'algorithme EML-MAP fonctionnant avec le même nombre d'itérations. Dans le cadre du standard CCSDS, des gains de décodage représentant un ordre de grandeur au niveau du taux d'erreur trame ont été observés. Ainsi, pour une valeur de SNR constante les performances de décodage des turbo codes peuvent être améliorées. Une architecture matérielle a été proposée afin d'estimer le surcoût au niveau de la complexité calculatoire de cette approche. Ce surcoût s'est avéré modeste.

Dans le troisième chapitre, un algorithme de correction des erreurs résiduelles, nommé Flip and Check (FNC), a été défini. Partant du principe que de nombreux standards de communications numériques incluent un code CRC concaténé avec le turbo code, il est possible d'envisager une plus grande diversité dans le décodage en tirant parti des nouvelles propriétés de distance de cette concaténation. L'algorithme FNC comprend trois étapes. Après l'identification des positions les moins fiables dans la trame en cours de traitement par le turbo décodeur, différents mots candidats sont générés puis vérifiés en utilisant un code détecteur d'erreurs. L'utilisation de l'algorithme FNC permet d'obtenir des gains de performance de décodage dans la région du plancher d'erreurs d'au moins un

ordre de grandeur. Ces gains ont été observés pour des turbo codes binaires ou double binaires à 8 ou 16 états. Les performances de l'algorithme reposent principalement sur le pouvoir d'identification de la métrique permettant la sélection des positions les moins fiables dans la séquence d'information reçue. Afin de définir cette métrique, différentes approches d'identification ont été proposées et comparées. Il apparaît que la métrique mise en exergue permet d'identifier de façon quasi-systématique les erreurs résiduelles à l'issue du processus de turbo décodage. Les performances de décodage de l'algorithme FNC reposent aussi sur le pouvoir de détection du code détecteur d'erreurs associé. En effet, la génération de multiples mots candidats augmente de fait le risque d'erreur non détectée. Néanmoins la présence dans les standards de communications numériques de codes CRC dont les tailles atteignent 32 bits permet d'offrir des propriétés de distance conséquentes. Ainsi, le risque de non-détection n'impacte pas de manière significative la mise en œuvre de l'algorithme FNC.

Enfin, le quatrième chapitre détaille une proposition d'architecture matérielle pour l'implémentation de l'algorithme FNC. L'algorithme FNC pouvant être vu comme une extension du processus de turbo décodage conventionnel, son implantation peut être réalisée sans impacter le processus itératif de décodage. Cependant, l'interconnexion entre ces deux architectures matérielles indépendantes nécessite un transfert de différentes informations du turbo décodeur vers le bloc FNC. Afin de pouvoir déterminer la manière dont cette interconnexion peut être réalisée, différents ordonnancements de turbo décodage ont été présentés dans un premier temps. Puis, une caractérisation de l'impact des différents paramètres de l'algorithme FNC a été menée. En effet ces différents paramètres ont un influence sur les performances de décodage de l'algorithme et sur la complexité calculatoire résultante. L'objectif a alors été de réduire la complexité calculatoire de l'algorithme FNC et de caractériser l'éventuelle dégradation des performances de décodage associée. Cette étude a été faite dans le cadre du standard LTE. Ces analyses ont permis d'aboutir au cas d'étude d'une architecture matérielle associée à l'algorithme FNC pour un ordonnancement de turbo décodage particulier : l'ordonnancement Aller-Retour avec fenêtre glissante. Les résultats d'implémentation pour une cible FPGA ont été comparés à un turbo décodeur existant sous forme d'IP Core chez Xilinx. Il apparaît alors que le surcoût calculatoire induit par l'algorithme FNC est raisonnable et peut être adapté en fonction des gains de performance de décodage recherchés. Finalement, des propositions quant à la transposition de cette première architecture matérielle à d'autres types de turbo décodeur ont été introduites. Ces propositions confortent le fait qu'une implémentation matérielle de l'algorithme FNC est envisageable, ce quelque soit le turbo décodeur considéré.

Pour conclure sur les travaux de thèse présentés dans ce manuscrit, bien que la présentation originelle des turbo codes remonte maintenant à près de 25 ans, la proposition d'algorithmes permettant l'amélioration de leurs performances de décodage est encore possible. Des algorithmes présentant des gains dans la zone de convergence ou bien dans la zone du plancher d'erreurs ont été au cœur de ces travaux de thèse. Ces algorithmes ont une complexité calculatoire maîtrisée. Ceci permet d'envisager dès à présent leur utilisation dans divers contextes applicatifs nécessitant une amélioration des performances de décodage de turbo codes standardisés. Pour ce faire, elles requièrent la concaténation d'un turbo code avec un code détecteur d'erreurs. Il est d'ailleurs intéressant de remarquer qu'en ce qui concerne les codes polaires, l'algorithme de décodage concentrant une

forte émulation auprès de la communauté scientifique est le décodage par liste. Celui-ci considère l'obtention de N mots candidats. La sélection parmi ceux-ci du mot décidé est effectuée par une vérification d'un code CRC. Ainsi, la considération conjointe des différents éléments constituants une chaîne de communications numériques lors de la réception de l'information permet d'observer de meilleures performances de décodage. Afin d'illustrer plus encore ce propos, il est possible de citer les chaînes de communications numériques associant une modulation codée et un code correcteur d'erreurs. De la sorte, le décodage peut être réalisé par un système doublement itératif. Dans ce contexte, un échange d'informations extrinsèques est effectué entre une turbo démodulation et le décodeur de canal, permettant de continuer de s'approcher de la limite théorique de Shannon.

Perspectives

Plusieurs perspectives d'études et de recherches restent à explorer à la suite de ce travail de thèse. Ci-après, une liste non exhaustive de ces perspectives est donnée.

Tout d'abord la première perspective d'étude concerne une limite actuelle des travaux de thèse. Les résultats de simulation démontrant la pertinence des algorithmes de décodage présentés ont été réalisés pour des canaux de communication basés exclusivement sur un bruit blanc gaussien. Certes, ce modèle de canal est le plus utilisé par la communauté scientifique pour comparer les performances de codes correcteurs d'erreurs. Cependant, lors de communications numériques réelles, des interférences entre symboles, voire des effacements peuvent apparaître à la réception. Ainsi, considérer des modèles de canaux de transmission prenant en compte ces contraintes s'avère nécessaire.

La contribution majeure de ces travaux de thèse repose sur la proposition d'un algorithme de corrections des erreurs résiduelles des turbo codes. Cependant, l'apparition d'un plancher d'erreurs semble être rencontré dès lors qu'un décodage itératif est considéré. Ainsi, il peut être intéressant d'étendre ces résultats à d'autres codes correcteurs d'erreurs. Une étude préliminaire semble montrer que dans le contexte de certains codes LDPC, une identification des erreurs résiduelles par une métrique semblable à celle utilisée au cœur de l'algorithme FNC est possible. Cependant, afin de proposer des gains conséquents en terme de performance de décodage, il est nécessaire que les capacités d'identification d'une telle métrique soient particulièrement efficaces. En ce qui concerne le décodage des codes polaires, grâce à un décodage par liste, avec un nombre de mots candidats conséquent (32), il semblerait qu'aucun plancher d'erreurs ne soit rencontré pour des taux d'erreurs trame de 10^{-10}. Néanmoins, il est possible que cette constatation diffère pour les codes polaires pressentis pour la 5G. Il est en effet difficile d'adresser à la fois un seuil de convergence faible et des performances asymptotiques. De plus, la quantification de l'information peut amener l'apparition d'un plancher d'erreurs. Peut-être alors que dans ce contexte, une transposition de l'algorithme FNC serait envisageable.

Enfin, de nombreuses perspectives d'études concernent les aspects architecturaux associés à l'algorithme FNC. Ainsi, il serait nécessaire d'évaluer au mieux la complexité calculatoire de telles architectures adaptées pour un turbo décodeur à fort degré de parallélisme. Pour

ce faire, des implantations pour une cible technologique de type ASIC sont indispensables. Mais avant cela, des travaux quant à la réduction de la complexité calculatoire peuvent encore être menés. Le principal défi quant à la réduction de la complexité calculatoire de l'algorithme FNC repose sur l'unité de calculs de CRC. Une première solution a été envisagée à la fin du chapitre 4. En effet, il a été observé que l'efficacité d'une unité de calcul de CRC augmente avec son degré de parallélisme. Or, cette augmentation de parallélisme est concomitante avec une réduction du temps nécessaire pour la vérification d'un CRC. Il apparaît alors opportun de maximiser le degré de parallélisme de chaque unité de vérification de CRC afin d'en réduire le nombre. Ainsi, une seule unité de vérification pourrait alors tester la validité de plusieurs mots candidats durant un même intervalle de temps.

A Compléments au Chapitre 1

Afin d'alléger la lecture du premier chapitre, certains calculs de ce chapitre sont déroulés dans cette annexe.

A.1 Détails des calculs de l'algorithme APP

Cette partie vise à démontrer l'ensemble des calculs présentés dans la section 1.2.2.1.

A.1.1 Décomposition de la probabilité jointe

Cette décomposition est basée sur le partitionnement de la séquence reçue $\mathbf{y}$ en trois sous-séquences. La première représentant le passé $\mathbf{y}_{<\mathbf{k}}$, la seconde le présent $\mathbf{y}_{\mathbf{k}}$ et enfin le futur $\mathbf{y}_{>\mathbf{k}}$.

Le calcul suivant est basé sur la relation de Bayes : $P(A|B) = \frac{P(A,B)}{P(B)}$ et sa version ternaire $P(A|B,C) = \frac{P(A,B|C)}{P(B|C)}$.

Aussi, sont nécessaires au déroulement de ce calcul les propriétés de Markov du treillis de l'encodeur RSC considéré. En effet :

$$
\begin{aligned}
P(s_{k-1}=s,s_k=s',\mathbf{y}) &= P(s_{k-1}=s,s_k=s',\mathbf{y}_{<\mathbf{k}},\mathbf{y}_{\mathbf{k}},\mathbf{y}_{>\mathbf{k}}) \\
&= P(\mathbf{y}_{>\mathbf{k}}|s_{k-1}=s,s_k=s',\mathbf{y}_{<\mathbf{k}},\mathbf{y}_{\mathbf{k}}) \times P(s_{k-1}=s,s_k=s',\mathbf{y}_{<\mathbf{k}},\mathbf{y}_{\mathbf{k}}) \\
&= P(\mathbf{y}_{>\mathbf{k}}|s_k=s') \times P(s_{k-1}=s,s_k=s',\mathbf{y}_{<\mathbf{k}},\mathbf{y}_{\mathbf{k}}) \\
&= P(\mathbf{y}_{>\mathbf{k}}|s_k=s') \times P(s_k=s',\mathbf{y}_{\mathbf{k}}|s_{k-1}=s,\mathbf{y}_{<\mathbf{k}}) \times P(s_{k-1}=s,\mathbf{y}_{<\mathbf{k}}) \\
&= \underbrace{P(\mathbf{y}_{>\mathbf{k}}|s_k=s')}_{\beta_k(s')} \times \underbrace{P(s_k=s',\mathbf{y}_{\mathbf{k}}|s_{k-1}=s)}_{\gamma_k(s,s')} \times \underbrace{P(s_{k-1}=s,\mathbf{y}_{<\mathbf{k}})}_{\alpha_{k-1}(s)} \\
&= \alpha_{k-1}(s) \times \gamma_k(s,s') \times \beta_k(s')
\end{aligned}
$$

A.1.2 Calcul récursif de α

Par définition, $\alpha_{k-1}(s) = P(s_{k-1} = s,\mathbf{y_{<k}})$. Or, sachant que $\sum_B P(A,B) = P(A)$ et en appliquant le théorème de Bayes,

$$\begin{aligned}
\alpha_k(s) &= \sum_{s'} P(s_k = s,s_{k-1} = s',\mathbf{y_{<k},y_k}) \\
&= \sum_{s'} P(s_k = s,\mathbf{y_k}|s_{k-1} = s',\mathbf{y_{<k}}) \times P(s_{k-1} = s',\mathbf{y_{<k}}) \\
&= \sum_{s'} P(s_k = s,\mathbf{y_k}|s_{k-1} = s') \times P(s_{k-1} = s',\mathbf{y_{<k}}) \\
&= \sum_{s'} \gamma_k(s',s) \times \alpha_{k-1}(s')
\end{aligned}$$

Ainsi, les valeurs de $\alpha_k(s)$ peuvent être calculées récursivement en parcourant le treillis dans l'ordre chronologique, à partir des valeurs initiales $\alpha_0(s)$ et des probabilités de transition.

A.1.3 Calcul récursif de β

En utilisant les mêmes propriétés calculatoires que pour le calcul de α :

$$\begin{aligned}
\beta_{k-1}(s) &= P(\mathbf{y_{>k-1}}|s) \\
&= \sum_{s'} P(s',\mathbf{y_k},\mathbf{y_{>k}}|s) \\
&= \sum_{s'} P(\mathbf{y_{>k}}|s',s,\mathbf{y_k})P(s',\mathbf{y_k}|s) \\
&= \sum_{s'} P(\mathbf{y_{>k}}|s')P(s',\mathbf{y_k}|s) \\
&= \sum_{s'} \gamma_k(s,s') \times \beta_k(s')
\end{aligned}$$

A.1.4 Calcul de la probabilité *a posteriori*

La probabilité *a posteriori* a pour expression :

$$\begin{aligned}
\|\mathbf{y_k} - \mathbf{c_k}\|^2 &= (y_k^s - c_k^s)^2 + (y_k^p - c_k^p)^2 \\
&= (y_k^s)^2 - 2y_k^s c_k^s + (c_k^s)^2 + (y_k^p)^2 - 2y_k^p c_k^p + (c_k^p)^2 \\
&= (y_k^s)^2 + (y_k^p)^2 + 2 - 2 * (y_k^s c_k^s + y_k^p c_k^p)
\end{aligned}$$

Ainsi,

$$\gamma_k(s,s') = \frac{P(m_k)}{2\pi\sigma^2} \exp\left(-\frac{(y_k^s)^2 + (y_k^p)^2 + 2}{2\sigma^2}\right) \exp\left(\frac{(y_k^s c_k^s + y_k^p c_k^p)}{\sigma^2}\right)$$

Or, la première exponentielle n'est pas dépendante de m_k (ou du chemin $s \mapsto s'$) et peut donc être supprimée du numérateur et du dénominateur dans l'expression de la probabilité *a posteriori*.

A.2 Détails des calculs pour les algorithmes sub-APP

A.2.1 Calcul des métriques

Dans le cadre des algorithmes sub-APP, les probabilités manipulées sont transformées en métriques en prenant le logarithme népérien de celles-ci. Ainsi, nous avons :

$$\begin{aligned}\tilde{\alpha}_k(s) &= \ln \sum_{s'} \alpha_{k-1}(s') \times \gamma_k(s',s) \\ &= \ln \sum_{s'} \exp\left(\tilde{\alpha}_{k-1}(s')\right) \times \exp\left(\tilde{\gamma}_k(s',s)\right) \\ &= \ln \sum_{s'} \exp\left(\tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s',s)\right)\end{aligned}$$

Des calculs très similaires permettent d'obtenir $\tilde{\beta}_k(s)$ et $\tilde{\gamma}_k(s,s')$.

A.2.2 Opérateur max*

En partant de la définition de l'opérateur max* et en utilisant une disjonction de cas, on obtient :

$$\begin{aligned}\max^*(x,y) &= \ln\left(e^x + e^y\right) \\ &= \begin{cases} \ln\left(e^x\left(1 + e^{y-x}\right)\right) \text{si} x > y \\ \ln\left(e^y\left(1 + e^{x-y}\right)\right) \text{si} x < y \end{cases} \\ &= \begin{cases} x + \ln\left(1 + e^{y-x}\right) & \text{si} \quad x > y \\ y + \ln\left(1 + e^{x-y}\right) & \text{si} \quad x < y \end{cases} \\ &= \max(x,y) + \ln\left(1 + e^{|x-y|}\right)\end{aligned}$$

B Compléments au Chapitre 2

Dans un soucis de lisibilité, l'ensemble des résultats statistiques obtenus dans le cadre du standard CCSDS sont situés dans cette annexe. Ils confortent les propos tenus en section 2.2.3.

Cette section présente les observations statistiques obtenues avec le standard CCSDS. La Figure B.1 présente les valeurs moyennes d'oscillations, ce pour différentes valeurs de SNR. Les Figures B.2 et B.3 présentent l'évolution des oscillations au cours des itérations pour, respectivement, un taux d'erreur trame correspondant au seuil de convergence et un correspondant au plancher d'erreurs. Finalement, les Figures B.4 et B.5 présentent la distribution normalisée des oscillations pour ces deux valeurs de SNR considérées.

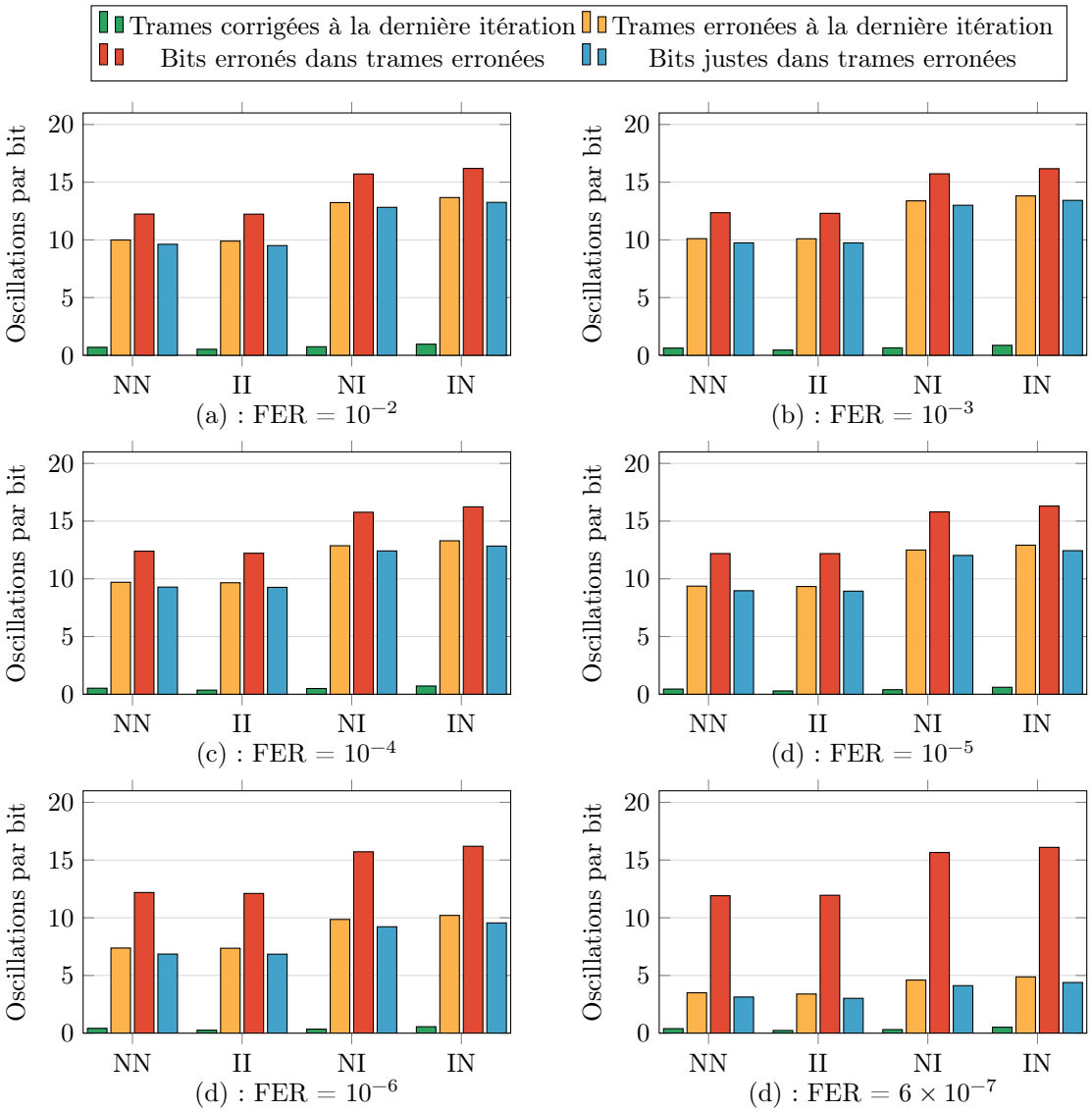
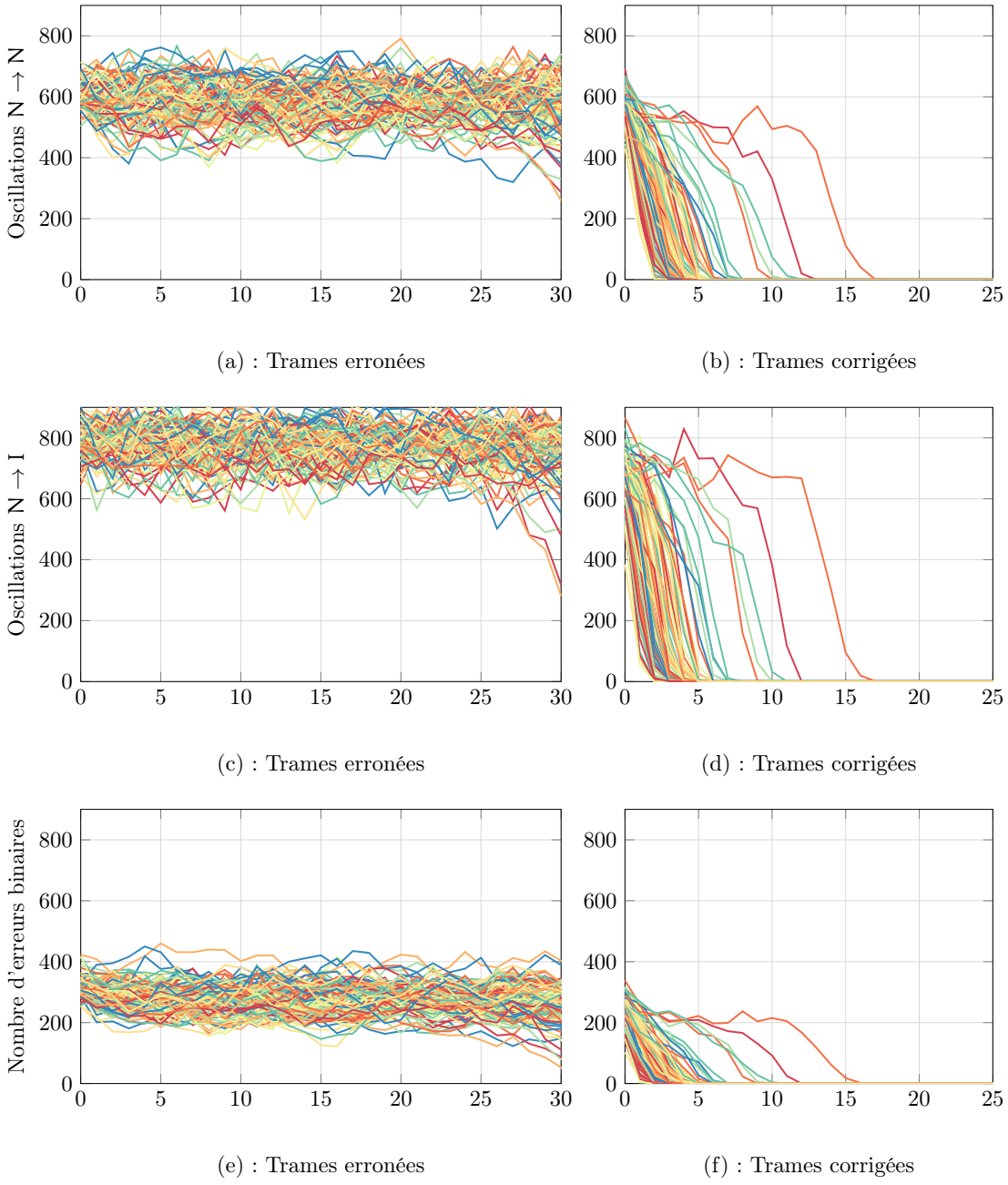


FIGURE B.1 – Nombre moyen d'oscillations pour différents taux d'erreurs trames cibles, turbo code du standard CCSDS (K=1784, R=1/3).

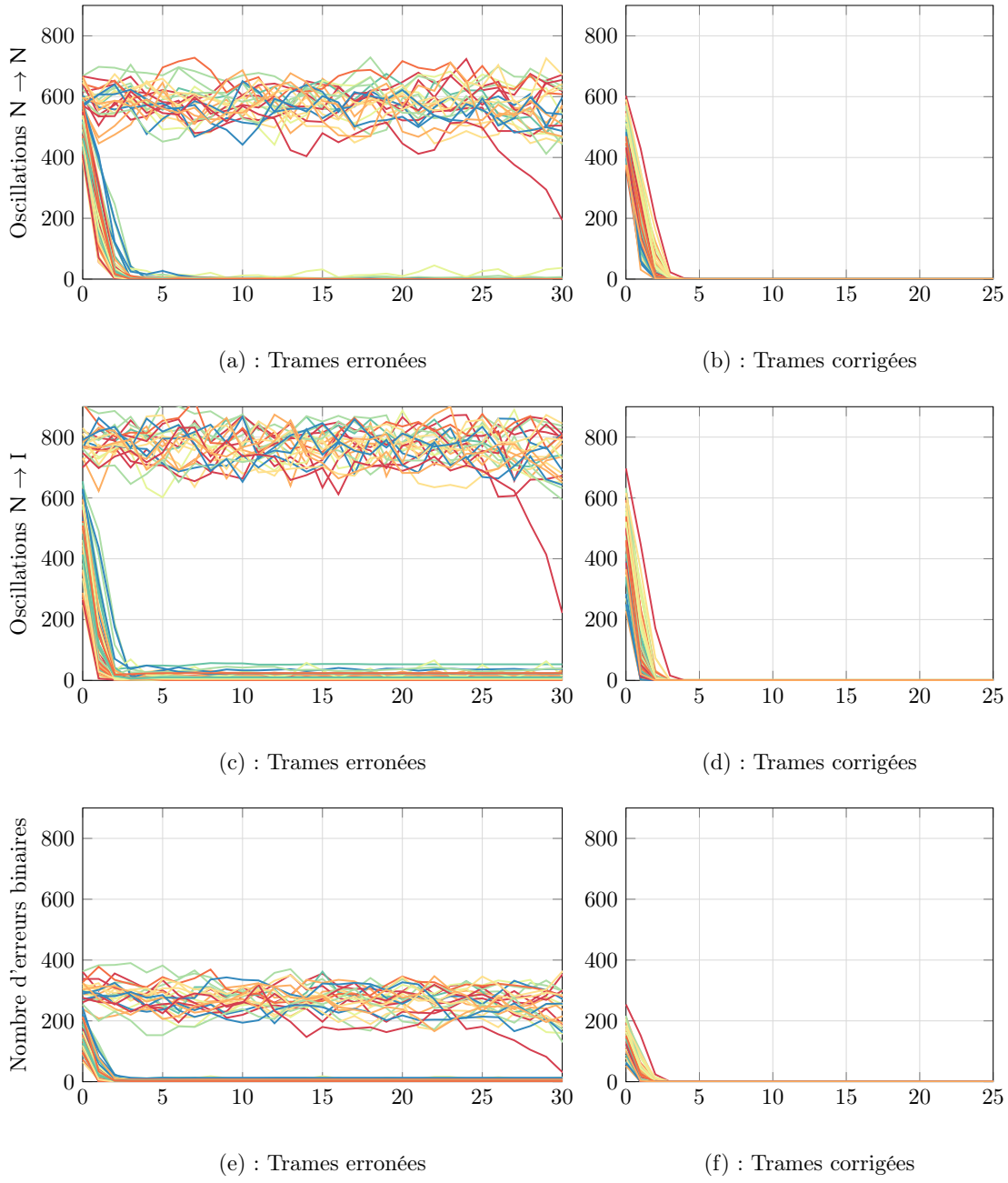


(a) : Trames erronées (b) : Trames corrigées

(c) : Trames erronées (d) : Trames corrigées

(e) : Trames erronées (f) : Trames corrigées

FIGURE B.2 – Oscillations au cours des itérations dans le cadre du standard CCSDS (K=1784, R=1/3) pour un taux d'erreur trame de 10^{-2}.

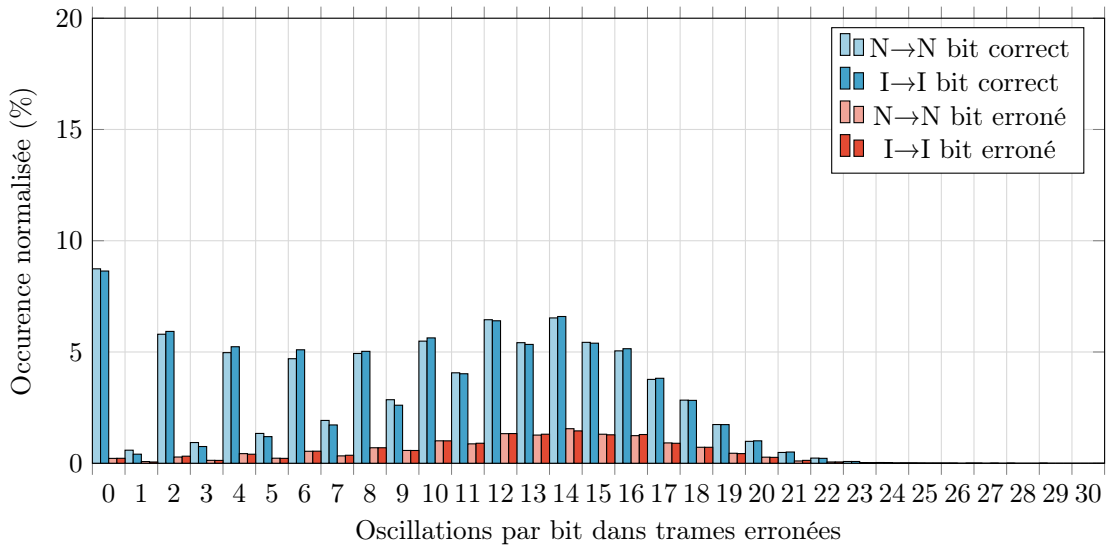


(a) : Trames erronées (b) : Trames corrigées

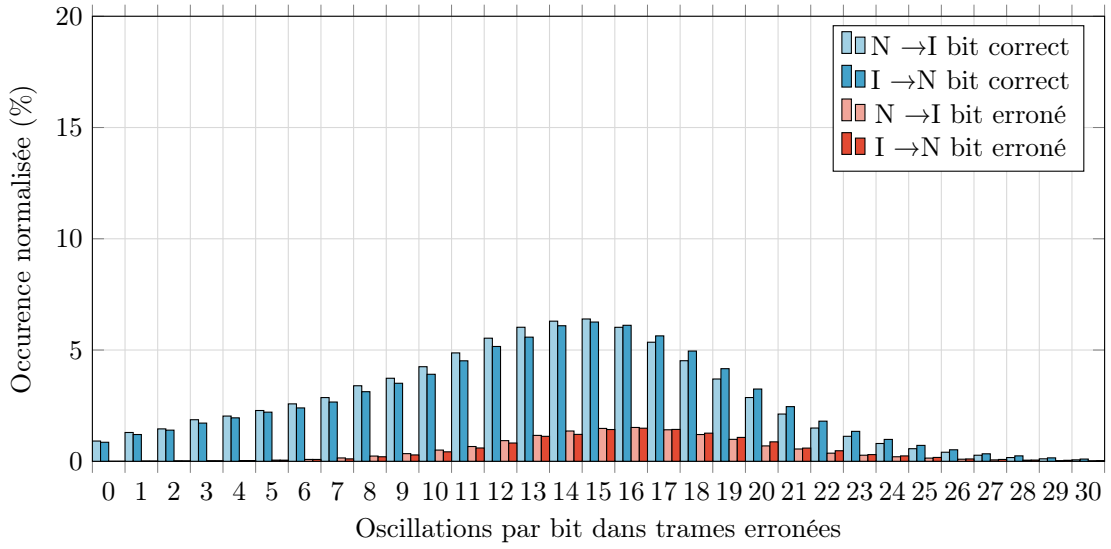
(c) : Trames erronées (d) : Trames corrigées

(e) : Trames erronées (f) : Trames corrigées

FIGURE B.3 – Oscillations au cours des itérations dans le cadre du standard CCSDS (K=1784, R=1/3) pour un taux d'erreur trame de 6×10^{-7}.

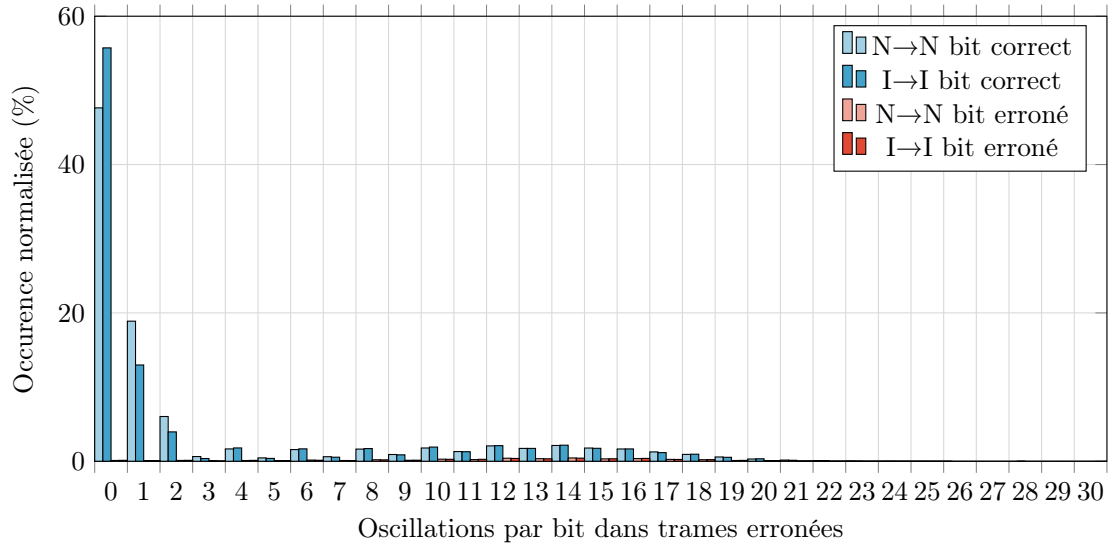


(a) : Oscillations

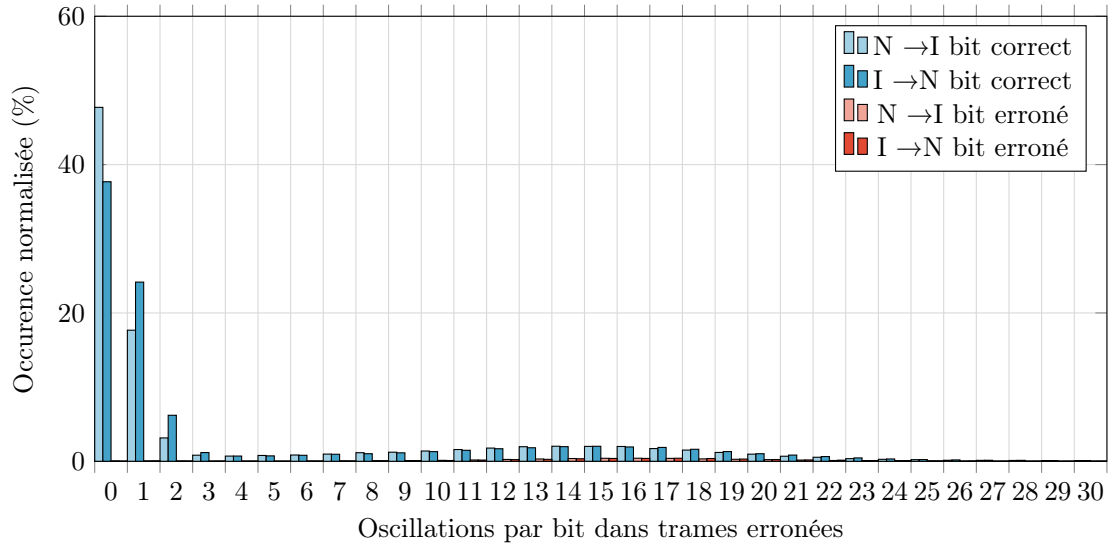


(b) : Désaccords

FIGURE B.4 – Distribution du nombre d'oscillations par bit pour un taux d'erreur trame de 10^{-2}, pour le turbo code du standard CCSDS (K=1784, R=1/3).



(a) : Oscillations



(b) : Désaccords

FIGURE B.5 – Distribution du nombre d'oscillations par bit pour un taux d'erreur trame de 6×10^{-7}, pour le turbo code du standard CCSDS (K=1784, R=1/3).

C Compléments au Chapitre 3

Toujours dans un objectif de lisibilité plus aisé, cette dernière annexe présente des résultats du troisième chapitre.

Tout d'abord, les spectres de distances complets de turbo codes standardisés sont donnés. Ensuite, les distributions d'erreurs du turbo code du standard DVB-RRCS pour K=752 et R=1/3 sont présentés. Puis les pourcentages d'identification pour les métriques Δ_1' et Δ_2' pour différents codes du standard DVB-RCS avec K=440 sont donnés.

Finalement, des performances de décodage obtenues pour l'algorithme FNC sont données ; d'abord le BER pour 4 turbo codes du standard LTE puis le FER pour des turbo codes du standard DVB-RCS avec K=752.

TABLEAU C.1 – Spectres de distances de turbo codes binaires standardisés

LTE	K=528	d	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
		a	1	1	1	0	3	3	6	0	6	4	11	10	14	12	96	886	1421	45	123	156	192	227	140	55	285	52	62
		w	1	2	3	0	9	6	16	0	20	16	39	32	58	50	318	5268	12535	210	779	864	984	982	850	286	1415	260	292
	K=1024	d	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49				
		a	1	2	3	1	2	1	2	2	7	3	7	9	11	11	20	30	147	437	49	64	583	179	511				
		w	1	4	9	2	6	2	6	6	21	12	27	32	43	44	78	142	719	1770	241	326	2949	1148	3407				
	K=1504	d	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49		
		a	1	0	0	1	1	1	2	1	2	5	1	6	7	3	14	13	17	22	32	28	47	49	185	91	1414		
		w	3	0	0	2	3	2	6	4	6	18	3	22	23	12	54	54	75	90	154	126	223	246	909	486	9696		
	K=2048	d	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49				
		a	1	2	1	0	3	0	2	1	2	1	1	3	8	5	12	15	16	425	23	37	240	45	290				
		w	1	4	3	0	9	0	6	4	6	4	3	8	26	20	44	62	72	1702	117	178	1180	230	1872				
	K=6144	d	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49			
		a	1	0	0	0	0	0	1	1	1	0	1	1	1	2	3	7	9	10	6	8	10	21	20	25			
		w	2	0	0	0	0	0	4	3	4	0	4	3	4	8	12	25	34	46	30	40	42	99	102	135			
CCSDS	K=1784	d	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49											
		a	1	0	0	7	4	2	5	2	5	4	3	6	7	4	713	8											
		w	2	0	0	17	9	6	13	5	14	18	6	19	39	21	4248	39											

TABLEAU C.2 – Spectres de distance de turbo codes du standard DVB-RCS

K=440	R=1/2	d	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
		a	110	0	220	0	440	220	770	1962	998	2006	1279	942	475	379	135	32	307	31	195	12	388	207	138	54	11
		w	1100	0	1870	0	3740	1980	5610	19616	8881	17439	11145	7182	3530	2679	1458	313	1944	278	1348	89	2964	1677	858	485	107
		ws	770	0	1320	0	2750	1540	4510	13533	6216	12215	7718	5120	2721	1935	1015	213	1392	201	954	63	2205	1254	648	294	78
	R=1/3	d	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
		a	110	0	220	0	0	0	220	110	1209	868	329	532	1307	1069	976	1025	642	653	454	231	267	34	115	14	20
		w	1100	0	1870	0	0	0	2200	1210	9889	7690	3181	5926	10702	9020	8509	8231	6470	5746	4098	1747	2367	284	1349	87	170
		ws	770	0	1320	0	0	0	1540	880	7472	5636	2523	4099	7855	6548	5884	6276	4263	4154	2512	1170	1661	202	905	73	125
	R=3/4	d	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
		a	27	193	600	2319	5490	3348	918	756	668	627	582	673	612	771	785	813	849	851	882	791	804	665	752	624	632
		w	108	810	3439	13675	34710	20506	5601	4748	4264	4046	3672	4232	3840	4855	5077	5175	5508	5561	5783	4966	4928	4433	4625	3993	4182
		ws	108	741	2939	11424	28458	16658	4581	3873	3519	3293	2990	3443	3138	3933	4122	4135	4475	4484	4636	4053	4087	3672	3730	3299	3416
	R=6/7	d	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
		a	10	51	615	3376	8394	1915	1382	1429	1266	1406	1416	1419	1448	1317	1361	1504	1474	1513	1666	1632	1722	1601	1632	1652	1888
		w	40	206	2810	16954	47379	10556	7800	8037	7274	8193	7964	7960	8046	7520	7864	8425	8541	8517	9422	9322	9698	8937	9149	9231	10589
		ws	40	193	2681	15833	43467	9577	7140	7315	6672	7450	7297	7242	7375	6855	7244	7801	7852	7873	8725	8666	8974	8210	8424	8538	9800
K=752	R=1/2	d	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
		a	376	376	0	752	1128	905	1697	3942	3557	2518	2007	2221	192	218	245	0	4	150	9	0	0	151	66	343	182
		w	3384	3008	0	6768	11280	8228	14714	33248	30349	19832	15641	21038	1726	1756	1748	0	37	1200	65	0	0	909	397	2388	1095
		ws	2256	2256	0	4512	7520	5548	10192	23266	20822	14700	10889	13942	1298	903	1026	0	25	900	49	0	0	755	265	1702	730
	R=1/3	d	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57
		a	376	0	376	752	0	0	752	188	381	2557	1465	1676	2052	2268	2677	800	176	1819	449	417	706	7	14	2	1
		w	3384	0	3760	6392	0	0	7520	1316	3434	24508	13154	16770	17718	21200	24119	6237	1746	16356	3497	3661	3096	56	84	15	9
		ws	2256	0	2632	4512	0	0	4888	752	2479	16999	9315	12105	12505	14941	16656	4345	1224	11285	2416	2158	2561	40	56	13	6
	R=3/4	d	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
		a	27	148	1462	5088	11114	11434	2411	455	496	601	621	507	571	536	711	750	1006	1163	1202	1035	939	1080	1062	941	989
		w	171	1025	9674	34032	75545	76442	15659	2901	3028	3773	3971	3376	3681	3384	4306	4576	6131	7230	7299	6740	6165	6679	6756	5963	6346
		ws	126	716	7449	26713	60008	60986	12306	2223	2361	2977	3125	2666	2846	2593	3287	3507	4673	5515	5570	5157	4761	5142	5298	4684	4916
	R=3/4	d	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
		a	199	1542	8737	23872	4320	1948	1722	1642	1910	1858	1653	1640	1604	1546	1654	1940	2081	2238	2319	2104	2039	2246	2270	2318	2392
		w	826	7197	48082	144563	24812	11673	10662	9979	10846	10701	9698	9415	8975	8675	9277	10901	11419	12389	13257	12236	12020	13085	13246	13382	13585
		ws	735	6467	43018	127920	21611	10289	9474	8760	9513	9152	8311	8153	7826	7697	8207	9838	10363	11218	11929	11030	10838	11743	11786	12047	12149

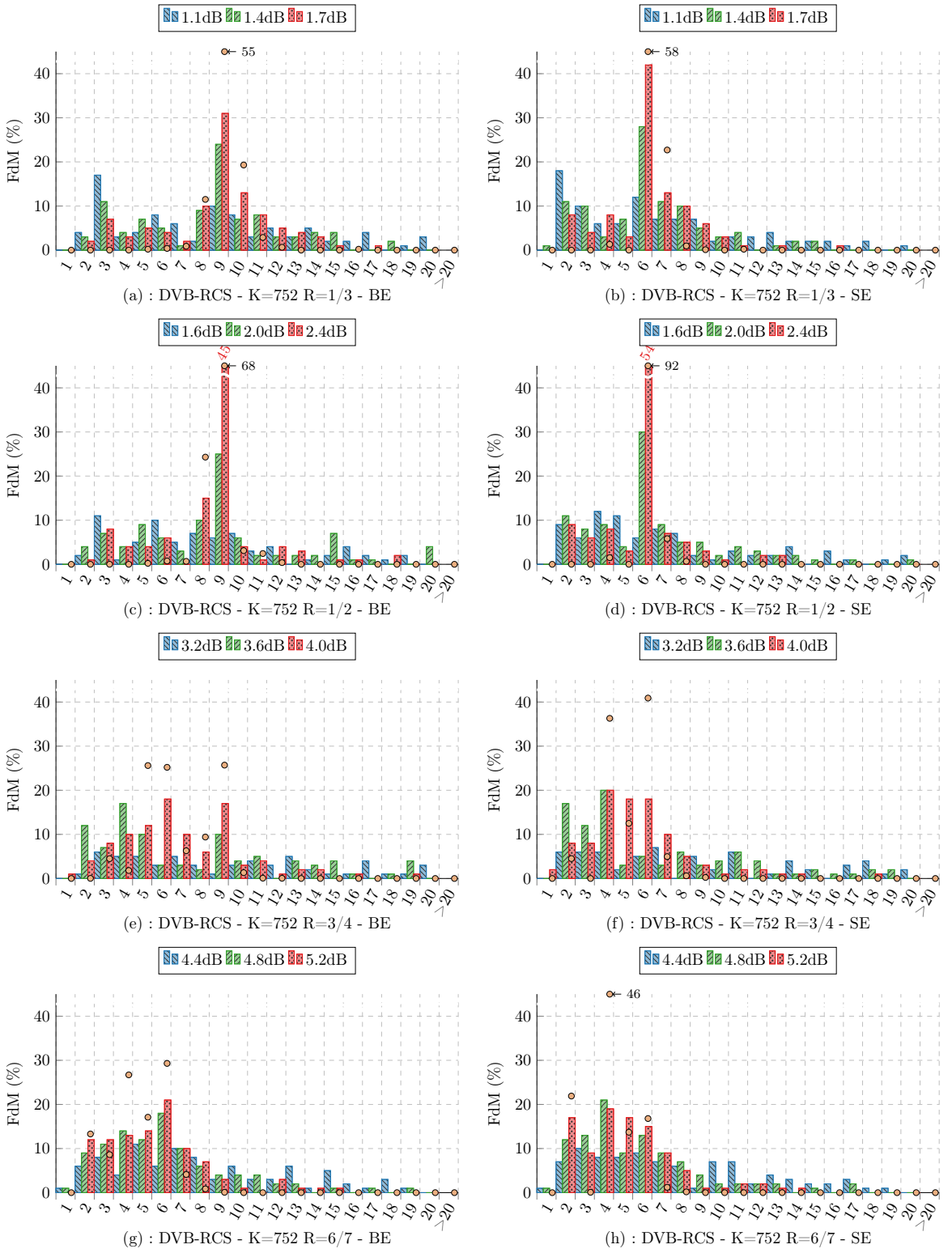


FIGURE C.1 – Distribution du nombre d'erreurs binaires et symboles pour différentes valeurs de SNR et pour différents turbo codes des standards DVB-RCS pour K=752. Décodage EML-MAP itérant 8 fois.

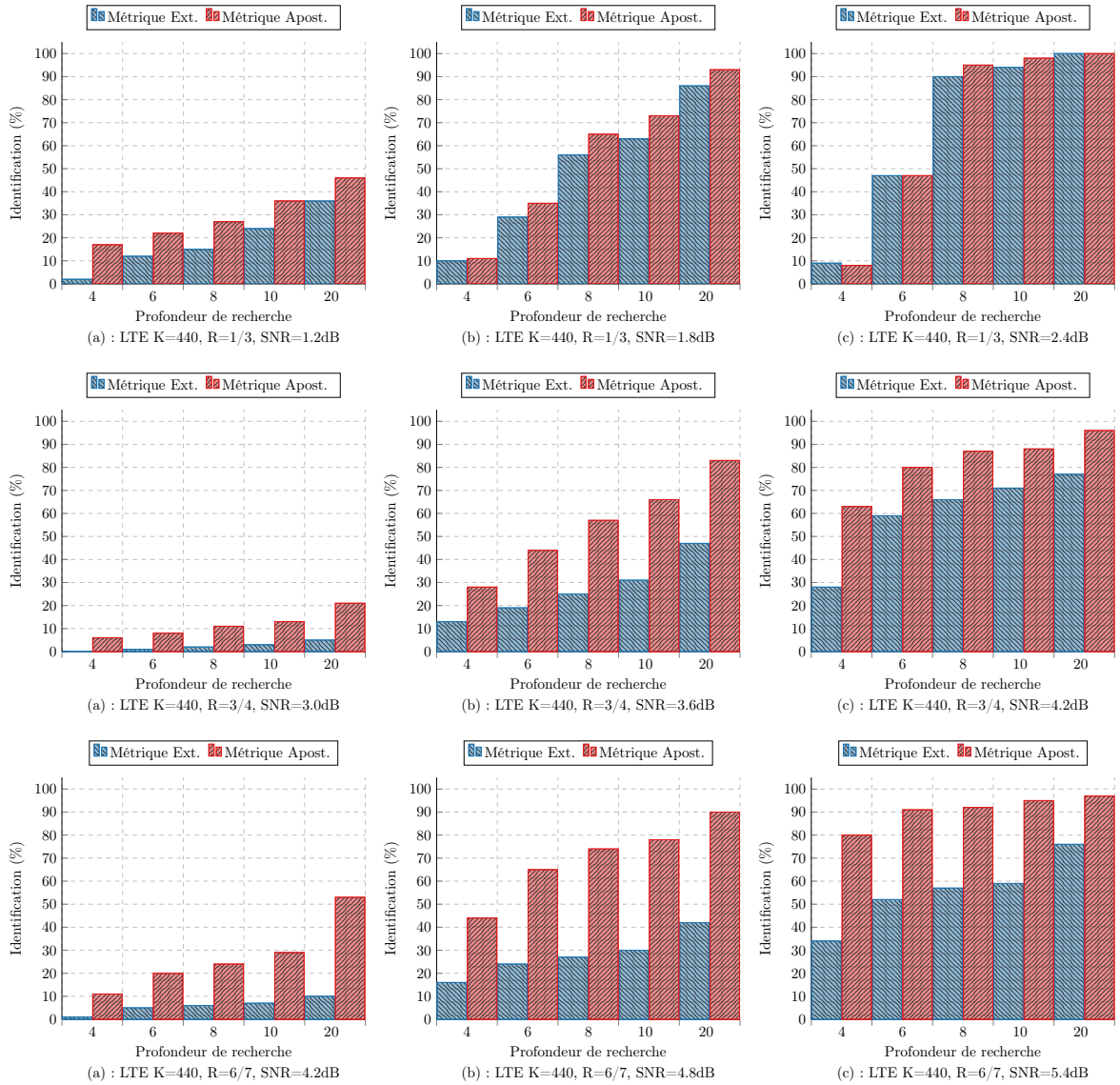


FIGURE C.2 – Pourcentage d'identification pour différents turbo codes du standard DVB-RCS K=440, R=1/3, 3/4 et 6/7. Décodage EML-MAP itérant 8 fois.

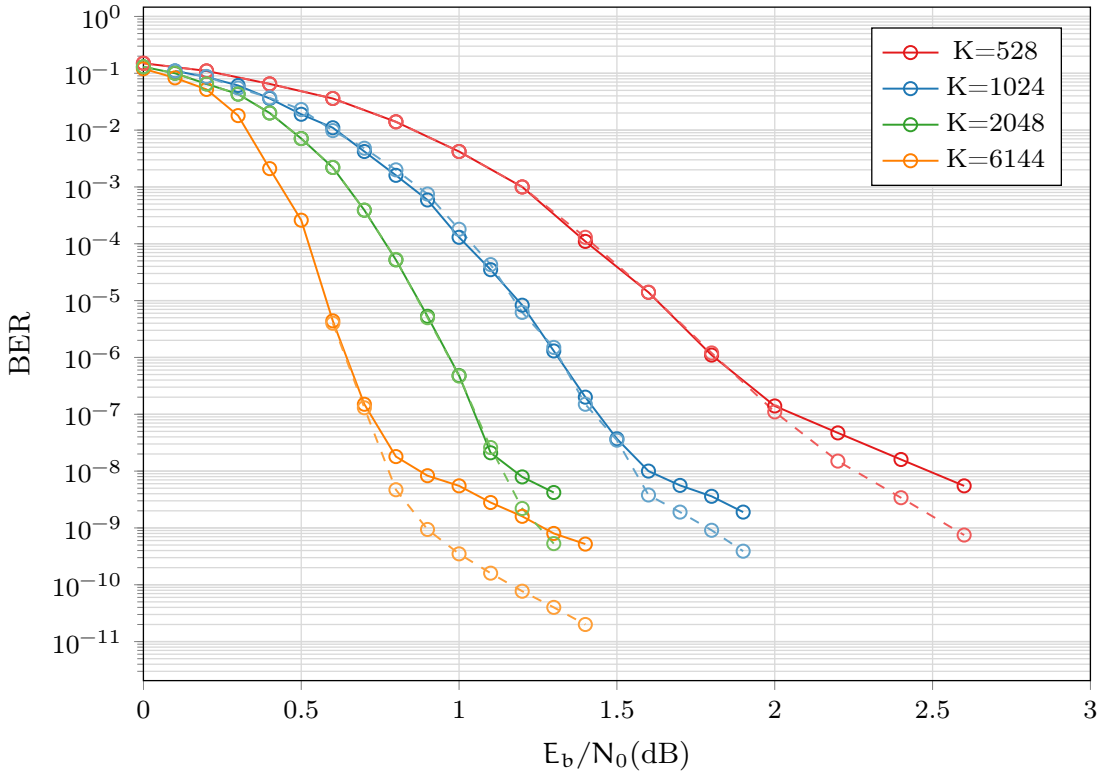


FIGURE C.3 – Comparaison de performances de décodages en terme de taux d'erreur binaire entre EML-MAP et FNC. Standard LTE, K=528, 1024, 2048 et 6144. R=1/3. Décodeurs itérant jusqu'à 8 fois.

Bibliographie

[1] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, 1948.

[2] R. W. Hamming, “Error detecting and error correcting codes,” *Bell System Technical Journal*, vol. 29, no. 2.

[3] I. Reed, “A class of multiple-error-correcting codes and the decoding scheme,” *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 38–49, September 1954.

[4] D. E. Muller, “Application of Boolean algebra to switching circuit design and to error detection,” *Transactions of the IRE. Professional Group on Electronic Computers*, vol. EC-3, no. 3, pp. 6–12, Sept 1954.

[5] P. Elias, “Coding for two noisy chanels,” *IRE Convention Record*, 1955.

[6] A. Hocquenghem, “Codes correcteurs d’erreurs,” pp. 147–156, 1959.

[7] R. C. Bose and D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes,” *Information and control*, vol. 3, no. 1, pp. 68–79, 1960.

[8] R. Gallager, “Low-density parity-check codes,” *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[9] G. D. Forney, *Concatenated codes*. MIT Press, 1966, vol. 11.

[10] R. McEliece, *Workshop on "Future Directions"*, 1993.

[11] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near shannon limit error-correcting coding and decoding : Turbo-codes. 1,” in *IEEE International Conference on Communications (ICC)*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[12] H. Nyquist, “Certain factors affecting telegraph speed,” *Bell System Technical Journal*, 1924.

[13] R. Hartley, “Transmission of information,” *Bell System Technical Journal*, 1928.

[14] International Organization for Standardization (ISO), “IEC 80000 - Part13 :Information science and technology,” International Organization for Standardization, Geneva, Switzerland., Tech. Rep., 2008.

[15] W. W. Peterson and D. T. Brown, “Cyclic codes for error detection,” *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, Jan 1961.

[16] G. D. Forney, “Convolutional codes i : Algebraic structure,” *IEEE Transactions on Information Theory*, vol. 16, no. 6, pp. 720–738, 1970.

[17] ——, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.

[18] H. Ma and J. Wolf, "On tail biting convolutional codes," *IEEE Transactions on Communications*, vol. 34, no. 2, pp. 104–111, Feb 1986.

[19] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, April 1967.

[20] ——, "Convolutional codes and their performance in communication systems," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 751–772, October 1971.

[21] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, Mar 1974.

[22] W. Ryan and S. Lin, *Channel codes : classical and modern.* Cambridge University Press, 2009.

[23] S. J. Johnson, *Iterative error correction : turbo, low-density parity-check and repeat-accumulate codes.* Cambridge University Press, 2009.

[24] C. Berrou and A. Giavieux, "Reflections on the prize paper : "near optimum error correcting coding and decoding : turbo-codes," IEEE Inform," *Theory Society Newsletter*, vol. 48, no. 2, 1998.

[25] G. Battail, "Pondération des symboles décodés par l'algorithme de Viterbi," *Annales des Télécommunications*, vol. 42, no. 1, pp. 31–38, 1987.

[26] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *IEEE Global Telecommunications Conference (GLOBECOM)*, Nov 1989, pp. 1680–1686 vol.3.

[27] C. Berrou, P. Adde, E. Angui, and S. Faudeil, "A low complexity soft-output Viterbi decoder architecture," in *IEEE International Conference on Communications (ICC)*, vol. 2, May 1993, pp. 737–740 vol.2.

[28] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[29] J. Hagenauer and P. Hoeher, "Concatenated viterbi-decoding," in *International Workshop on Information Theory*, September 1989.

[30] P. Thitimajshima, "Les codes convolutifs récursifs systématiques et leur application à la concaténation parallèle," Ph.D. dissertation, Dépt. Signal et Communications, 1992, Institut Mines-Télécom-Télécom Bretagne-UEB.

[31] D. Declercq, M. Fossorier, and E. Biglieri, *Channel Coding : Theory, Algorithms, and Applications : Academic Press Library in Mobile and Wireless Communications*, ser. Academic Press library in mobile and wireless communications.

[32] M. S. C. Ho, S. S. Pietrobon, and T. Giles, "Improving the constituent codes of turbo encoders," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 6, 1998, pp. 3525–3529 vol.6.

[33] S. Crozier, "New high-spread high-distance interleavers for turbo-codes," in *20th Biennial Symposium on Communications, Kingston, Ontario, Canada*, 2000, pp. 3–7.

[34] S. Benedetto and G. Montorsi, "Average performance of parallel concatenated block codes," *Electronics Letters*, vol. 31, no. 3, pp. 156–158, Feb 1995.

[35] S. Crozier and P. Guinand, "High-performance low-memory interleaver banks for turbo-codes," in *IEEE 54th Vehicular Technology Conference (VTC)*, vol. 4, 2001, pp. 2394–2398 vol.4.

[36] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan, and M. Jezequel, "Designing good permutations for turbo codes : towards a single model," in *IEEE International Conference on Communications (ICC)*, vol. 1, June 2004, pp. 341–345.

[37] Digital Video Broadcasting (DVB), "Interaction Channel for Satellite Distribution Systems," ETSI EN 301 790, v1.5.1, May 2009.

[38] ——, "Second Generation DVB Interactive Satellite System (DVB-RCS2) ; Part 2 : Lower Layers for Satellite Standard," ETSI EN 301 545–2, v1.1.1, January 2012.

[39] ——, "Interaction Channel for Digital Terrestrial Television (RCT) Incorporating Multiple Access OFDM," ETSI EN 301 958, v1.1.1, March 2002.

[40] J. Sun and O. Y. Takeshita, "Interleavers for turbo codes using permutation polynomials over integer rings," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 101–119, Jan 2005.

[41] 3rd Generation Partnership Project (3GPP), "Evolved Universal Terrestrial Radio Access (E-UTRA)," Multiplexing and Channel Coding, 3GPP TS 36.212, R11, 2012.

[42] E. Rosnes and O. Y. Takeshita, "Optimum distance quadratic permutation polynomial-based interleavers for turbo codes," in *IEEE International Symposium on Information Theory (ISIT)*, July 2006, pp. 1988–1992.

[43] O. Y. Takeshita, "On maximum contention-free interleavers and permutation polynomials over integer rings," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1249–1253, March 2006.

[44] R. G. Bohórquez, C. A. Nour, and C. Douillard, "On the equivalence of interleavers for turbo codes," *IEEE Wireless Communications Letters*, vol. 4, no. 1, pp. 58–61, Feb 2015.

[45] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain," in *IEEE International Conference on Communications (ICC)*, vol. 2. IEEE, 1995, pp. 1009–1013.

[46] O. D. S. González., "La montée en débit dans les architectures de turbo décodage de codes convolutifs," Ph.D. dissertation, École doctorale STIC - UBS, 2013, Institut Mines-Télécom-Télécom Bretagne-UEB.

[47] W. J. Gross and P. G. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *Electronics Letters*, vol. 34, no. 16, pp. 1577–1578, Aug 1998.

[48] B. Classon, K. Blankenship, and V. Desai, "Turbo decoding with the constant-log-MAP algorithm," in *2nd International Symposium on Turbo Codes and Related Topics (ISTC)*, Sept 2000, pp. 467–470.

[49] J.-F. Cheng and T. Ottosson, "Linearly approximated log-MAP algorithms for turbo decoding," in *51st IEEE Vehicular Technology Conference (ICC)*, vol. 3, 2000, pp. 2252–2256 vol.3.

[50] M. C. Valenti and J. Sun, "The UMTS turbo code and an efficient decoder implementation suitable for software-defined radios," *International Journal of Wireless Information Networks*, vol. 8, no. 4, pp. 203–215, 2001.

[51] J. Vogt and A. Finger, "Improving the max-log-MAP turbo decoder," *Electronics Letters*, vol. 36, no. 23, pp. 1937–1939, Nov 2000.

[52] H. Balta and C. Douillard, "On the influence of the extrinsic information scaling coefficient on the performance of single and double binary turbo codes," *Advances in Electrical and Computer Engineering*, vol. 2013, no. 2, pp. 77– 84, May 2013.

[53] C. Berrou and M. Jezequel, "Non-binary convolutional codes for turbo coding," *Electronics Letters*, vol. 35, no. 1, pp. 39–40, Jan 1999.

[54] C. Douillard and C. Berrou, "Turbo codes with rate-m/(m+1) constituent convolutional codes," *IEEE Transactions on Communications*, vol. 53, no. 10, pp. 1630–1638, Oct 2005.

[55] H. Balta, C. Douillard, and R. Lucaciu, "Multi-non-binary turbo codes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, pp. 1–15, 2013.

[56] C. Berrou, M. Jezequel, C. Douillard, and S. Kerouedan, "The advantages of non-binary turbo codes," in *IEEE Information Theory Workshop*, 2001, pp. 61–63.

[57] CCSDS, "TM Synchronization and Channel Coding," CCSDS - 131.0-B-2 - Blue Book, v2, August 2011.

[58] IEEE, "Standard for Local and Metropolitan Area Networks, Part 16," Medium Access Con- trol and Physical Layer Specifications, IEEE Std 1901-2010, Tech. Rep., 2010.

[59] ——, "Standard for Broadband Over Power Line Networks," CCSDS - 131.0-B-2 - Blue Book, v2, August 2011.

[60] S. ten Brink, "Convergence of iterative decoding," *Electronics Letters*, vol. 35, no. 10, pp. 806–808, May 1999.

[61] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 3, Nov 1994, pp. 1298–1303 vol.3.

[62] R. Garello, P. Pierleoni, and S. Benedetto, "Computing the free distance of turbo codes and serially concatenated codes with interleavers : algorithms and applications," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 800–812, May 2001.

[63] C. Berrou, S. Vaton, M. Jezequel, and C. Douillard, "Computing the minimum distance of linear codes by the error impulse method," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, Nov 2002, pp. 1017–1020 vol.2.

[64] Y. Ould-Cheikh-Mouhamedou, S. Crozier, and P. Kabal, "Comparison of distance measurement methods for turbo codes," in *The 9th Canadian Workshop of Information Theory*, vol. 1, June 2005, pp. 36–39.

[65] R. Garello and A. Vila-Casado, "The all-zero iterative decoding algorithm for turbo code minimum distance computation," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 1, June 2004, pp. 361–364.

[66] S. Crozier, P. Guinand, and A. Hunt, "Estimating the minimum distance of turbo-codes using double and triple impulse methods," *IEEE Communications Letters*, vol. 9, no. 7, pp. 631–633, July 2005.

[67] R. Garzon Bohorquez, C. Abdel Nour, and C. Douillard, "Improving Turbo Codes for 5G with Parity Puncture-Constrained Interleavers," in *9th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, 2016.

[68] J. D. Andersen, "Turbo codes extended with outer BCH code," *Electronics Letters*, vol. 32, no. 22, pp. 2059–2060, Oct 1996.

[69] K. R. Narayanan and G. L. Stuber, "Selective serial concatenation of turbo codes," *IEEE Communications Letters*, vol. 1, no. 5, pp. 136–139, Sept 1997.

[70] O. Y. Takeshita, O. M. Collins, P. C. Massey, and D. J. Costello, "On the frame-error rate of concatenated turbo codes," *IEEE Transactions on Communications*, vol. 49, no. 4, pp. 602–608, April 2001.

[71] C. Berrou, A. G. i. Amat, Y. O. C. Mouhamedou, C. Douillard, and Y. Saouter, "Adding a rate-1 third dimension to turbo codes," in *IEEE Information Theory Workshop*, Sept 2007, pp. 156–161.

[72] C. Berrou, A. G. I. Amat, Y. Ould-Cheikh-Mouhamedou, and Y. Saouter, "Improving the distance properties of turbo codes using a third component code : 3D turbo codes," *IEEE Transactions on Communications*, vol. 57, no. 9, pp. 2505–2509, September 2009.

[73] E. Rosnes and A. G. i Amat, "Performance analysis of 3-D turbo codes," *IEEE Transactions on Information Theory*, vol. 57, no. 6, pp. 3707–3720, June 2011.

[74] S. Tong, H. Zheng, and B. Bai, "Precoded turbo code within 0.1 dB of shannon limit," *Electronics Letters*, vol. 47, no. 8, pp. 521–522, April 2011.

[75] R. G. Bohorquez, C. A. Nour, and C. Douillard, "Precoding techniques for turbo codes," in *21st European Wireless Conference*, May 2015, pp. 1–6.

[76] M. El-Khamy, J. Lee, and I. Kang, "Detection analysis of CRC-assisted decoding," *IEEE Communications Letters*, vol. 19, no. 3, pp. 483–486, March 2015.

[77] P. Elias, *List decoding for noisy channels.* Massachusetts Institute of Technology, 1957.

[78] J. M. Wozencraft, "List decoding," *Quarterly Progress Report*, vol. 48, pp. 90–95, 1958.

[79] N. Seshadri and C. E. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 313–323, Feb 1994.

[80] C. F. Leanderson and C. E. W. Sundberg, "The max-log list algorithm (MLLA) a list-sequence decoding algorithm that provides soft-symbol output," *IEEE Transactions on Communications*, vol. 53, no. 3, pp. 433–444, March 2005.

[81] J. S. Sadowsky, "A maximum likelihood decoding algorithm for turbo codes," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, Nov 1997, pp. 929–933 vol.2.

[82] K. R. Narayanan and G. L. Stuber, "List decoding of turbo codes," in *IEEE International Conference on Communications (ICC)*, vol. 1, Jun 1998, pp. 141–145 vol.1.

[83] A. Akmalkhodzhaev and A. Kozlov, "New iterative turbo code list decoder," in *XIV International Symposium on Problems of Redundancy in Information and Control Systems (REDUNDANCY)*, June 2014, pp. 15–18.

[84] Y. Xu, M. Jiang, M. Ding, and Y. Yang, "An efficient OSD-aided iterative decoding algorithm for lte turbo codes," in *International Conference on Wireless Communications Signal Processing (WCSP)*, Oct 2012, pp. 1–4.

[85] M. P. C. Fossorier and S. Lin, "Soft-input soft-output decoding of linear block codes based on ordered statistics," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 5, 1998, pp. 2828–2833 vol.5.

[86] Y. Wei, M. Jiang, B. Xia, W. Chen, and Y. Yang, "A CRC-aided hybrid decoding algorithm for turbo codes," *IEEE Wireless Communications Letters*, vol. 2, no. 5, pp. 471–474, October 2013.

[87] Y. Ould-Cheikh-Mouhamedou, S. Crozier, K. Gracie, P. Guinand, and P. Kabal, "A method for lowering turbo code error flare using correction impulses and repeated decoding," in *4th International Symposium on Turbo Codes and Related Topics (ISTC)*, April 2006, pp. 1–6.

[88] Y. Ould-Cheikh-Mouhamedou and S. Crozier, "Improving the error rate performance of turbo codes using the forced symbol method," *IEEE Communications Letters*, vol. 11, no. 7, pp. 616–618, July 2007.

[89] S. Pfletschinger and M. Navarro, "Enhanced turbo decoding for error floor reduction," in *9th International Conference on Systems, Communication and Coding (SCC)*, Jan 2013, pp. 1–6.

[90] S. Benedetto and G. Montorsi, "Unveiling turbo codes : some results on parallel concatenated codes," in *IEEE International Symposium on Information Theory*, Sep 1995, pp. 32–.

[91] T. Richardson, "The geometry of turbo-decoding dynamics," *IEEE Transactions on Information Theory*, vol. 46, no. 1, pp. 9–23, Jan 2000.

[92] A. C. Reid, T. A. Gulliver, and D. P. Taylor, "Convergence and errors in turbo-decoding," *IEEE Transactions on Communications*, vol. 49, no. 12, pp. 2045–2051, Dec 2001.

[93] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1117–1120, Aug 1999.

[94] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, Mar 1996.

[95] S. Gounai, T. Ohtsuki, and T. Kaneko, "Modified belief propagation decoding algorithm for low-density parity check code based on oscillation," in *63rd IEEE Vehicular Technology Conference (VTC)*, vol. 3, May 2006, pp. 1467–1471.

[96] V. Savin, "Self-corrected Min-Sum decoding of LDPC codes," in *2008 IEEE International Symposium on Information Theory*, July 2008, pp. 146–150.

[97] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Department of Electrical Engineering, 1996, linköping University.

[98] A. Matache, S. Dolinar, and F. Pollara, "Stopping rules for turbo decoders," *TMO Progress Report 42-142*, 2000.

[99] D. Declercq, M. Fossorier, and E. Biglieri, *Channel Coding : Theory, Algorithms, and Applications Academic Press Library in Mobile and Wireless Communications*, 1st ed. Academic Press, 2014.

[100] L. C. Perez, J. Seghers, and D. J. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1698–1709, Nov 1996.

[101] K. Li and M. Motani, "On the distribution of residual errors of turbo codes and its application to concatenated codes," in *IEEE 56th Vehicular Technology Conference (VTC)*, vol. 2, 2002, pp. 985–989 vol.2.

[102] David Gnaedig, "High-Speed decoding of convolutional Turbo Codes," Ph.D. dissertation, Université de Bretagne Sud, 2005.

[103] O. Muller, A. Baghdadi, and M. Jézéquel, "Parallelism efficiency in convolutional turbo decoding," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, no. 1, p. 927920, 2010.

[104] T. Wolf, "Initialization of sliding windows in turbo decoders," in *3rd International Symposium on Turbo Codes and Related Topics (ISTC)*, Sept 2003, pp. 219–222.

[105] O. Muller, A. Baghdadi, and M. Jezequel, "Exploring parallel processing levels for convolutional turbo decoding," in *2nd International Conference on Information Communication Technologies*, vol. 2, 2006, pp. 2353–2358.

[106] M. M. Mansour and N. R. Shanbhag, "VLSI architectures for SISO-APP decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 627–650, Aug 2003.

[107] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaving laws to parallel turbo and LDPC decoder architectures," *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2002–2009, Sept 2004.

[108] D. Divsalar and F. Pollara, "Multiple turbo codes for deep-space communications," 1995.

[109] J. Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209–213, Feb 2005.

[110] J. Zhang, Y. Wang, M. Fossorier, and J. S. Yedidia, "Replica shuffled iterative decoding," in *IEEE International Symposium on Information Theory (ISIT)*, Sept 2005, pp. 454–458.

[111] O. Muller, A. Baghdadi, and M. Jezequel, "From parallelism levels to a multi-ASIP architecture for turbo decoding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 1, pp. 92–102, Jan 2009.

[112] Xilinx. (2009) 3GPP turbo decoder specifications. [Online]. Available : http://www.xilinx.com/support/documentation/ip_documentation/tcc_decoder_3gpp_ds318.pdf

Publications

[113] T. Tonnellier, C. Leroux, B. Le Gal, C. Jego, B. Gadat, and C. Poulliat, "L'algorithme Self-Corrected Max-Log-MAP pour le décodage des turbo codes," in *18ème Journées Nationales du Réseau Doctoral en Micro-nanoélectronique (JNRDM)*, 2015.

[114] ——, "Extension du principe Self-Corrected de l'information extrinsèque au décodage itératif de turbo codes," in *GRETSI*, Sept 2015.

[115] T. Tonnellier, C. Leroux, B. Le Gal, B. Gadat, C. Jego, and N. Van Wambeke, "Lowering the error floor of turbo codes with CRC verification," *IEEE Wireless Communications Letters*, vol. 5, no. 4, pp. 404–407, Aug 2016.

[116] T. Tonnellier, C. Leroux, B. Le Gal, C. Jego, B. Gadat, and N. Van Wambeke, "Lowering the error floor of double-binary turbo codes : The flip and check algorithm," in *9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sept 2016, pp. 156–160.

[117] A. Cassagne, T. Tonnellier, C. Leroux, B. Le Gal, O. Aumage, and D. Barthou, "Beyond Gbps turbo decoder on multi-core CPUs," in *9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sept 2016, pp. 136–140.

[118] T. Tonnellier, C. Leroux, B. Le Gal, C. Jego, B. Gadat, and N. Van Wambeke, "Hardware architecture for lowering the error floor of LTE turbo codes," in *Conference on Design and Architectures for Signal and Image Processing (DASIP)*, Oct 2016, pp. 107–112.

[119] ——, "Correction des erreurs résiduelles lors du processus de turbo décodage : algorithme et architecture," in *GDR SoC-SiP*, June 2017.