N° d'ordre : 436/22

UNIVERSITÉ SULTAN MOULAY SLIMANE
Faculté des Sciences et Techniques
Béni Mellal

Centre d'Etudes Doctorales : Sciences et Techniques

Formation doctorale : Mathématiques et Physiques Appliquées

Structure d'accueil : : Laboratoire de l'innovation mathématiques et technologie de l'information (LIMATI)

THÈSE

Présentée par

Es-said AZOUGAGHE

Pour l'obtention du grade de

Docteur

Spécialité : Informatique

Contribution au décodage des codes en bloc simples, produits et concaténés généralisés

---

Pr. Belaid BOUIKHALENE	FP, Université Sultan Moulay Slimane, Béni Mellal,	Président
Pr. Khalid AUHMANI	ENSA, Université Cadi Ayyad, Marrakech,	Rapporteur
Pr. Fouad AYOUB	CRMEF, Kénitra,	Rapporteur
Pr. Mohamed BASLAM	FST, Université Sultan Moulay Slimane, Béni Mellal	Rapporteur
Pr. Yassine SADQI	FP, Université Sultan Moulay Slimane, Béni Mellal,	Examinateur
Pr. Said SAFI	FP, Université Sultan Moulay Slimane Béni Mellal,	Directeur de thèse
Pr. Abderrazak FARCHANE	FP, Université Sultan Moulay Slimane, Béni Mellal,	Co-Directeur de thèse

À ***la mémoire de ma mère***
À ***mon père***
À ***mon épouse***
À ***mes enfants***
À ***mes sœurs***
À ***mes frères***
À ***ma grande famille***
À ***mes amis.***

Remerciements

Je n'aurais jamais pu réaliser ce travail doctoral sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur et l'intérêt manifestés à l'égard de ma recherche m'ont permis de progresser dans cette phase délicate.

En premier lieu, je remercie mes directeurs de thèse, les professeurs **Abderrazak FARCHANE** et **Said SAFI**, pour la confiance qu'ils m'ont accordé en acceptant d'encadrer ce travail doctoral, pour leurs multiples conseils et pour toutes les heures qu'ils ont consacré à diriger cette recherche. J'aimerais également leur dire à quel point j'ai apprécié leur grande disponibilité et respect sans faille des délais serrés de relecture des documents que je leur ai adressé. Enfin, j'ai été extrêmement sensible à leurs qualités humaines d'écoute et de compréhension tout au long de ce travail doctoral.

Je présente mes sincères remerciements à tous mes enseignants, qui, durant toutes les années de mes études, ont participé à ma formation et pour leurs conseils et encouragements.

J'adresse mes remerciements à tous les membres des deux laboratoires : LIMATI et TIAD et à mes collègues, pour leurs contributions, discussions et les échanges fructueux que nous avons entretenu pendant ces années de recherche.

Enfin, pour leur soutien moral, je tiens à remercier mes parents, mon épouse, mes enfants (**Rayane** et **Rania**), mes frères et mes sœurs pour leurs soutien inconditionnel, sans qui ce travail n'aurait jamais pu voir le jour. Je leur dédie ce travail.

Que tous ceux qui m'ont apporté leurs aides, de près ou de loin dans l'élaboration de ce travail, trouvent ici l'expression de mes sincères gratitudes.

Table des matières

Liste des Publications

Publications dans des Revues et Conférences à Comité de Lecture Internationales

1. **Es-said Azougaghe**, Abderrazak Farchane, Said Safi,Mostafa Belkasmi, *Turbo decoding of concatenated codes based on RS codes using adapted scaling factors*, Infocommunications Journal, Vol. XIV, No 1, March 2022, pp. 11-16.,
   https ://doi.org/10.36244/ICJ.2022.1.

2. **Abderrazak Farchane**, Es-said Azougaghe, et Mostafa Belkasmi*Turbo decoding of product codes based on Dual-R-m decoder* encours de soumission.

3. **Es-said Azougaghe**, Abderrazak Farchane,Idriss Tazigh & Ali azougaghe, *Comparative Study on the McEliece Public-Key Cryptosystem Based on Goppa and QC-MDPC Codes*, The 5th International Conference on Business Intelligence (CBI'21), May 27-29, 2021, Beni Mellal, Morocco.

4. **Es-said Azougaghe**, Abderrazak Farchane, Mostapfa Belkasmi, & Said Safi, *Iterative Decoding of GSCB Codes Based on RS Codes Using Adapted Scaling Factors*,Springer 2^{th} International Conference on Advanced Communication Systems and Information Security (ACOSIS 2019), 20-22 novembre 2019, Marrakech, Morocco.

5. **Es-said Azougaghe**, Abderrazak Farchane, Mostapfa Belkasmi & Safi Said, *Generalized serially concatenated codes : construction and iterative decoding*, The 5th International Conference on Business Intelligence (CBI'19), April 17-19, 2018, Beni Mellal, Morocco.

6. **Es-said Azougaghe**,Abderrazak Farchane, Mostapfa Belkasmi, & Safi Said, *Iterative Decoding of GPCB Codes Based on RS Codes Using Adapted Scaling Factors*, First International Conference on signals, Automation and Telecommunications (ICSAT'2018), 2-4 May 2018 Beni Mellal (Morocco).

Résumé

Cette thèse se focalise sur les codes correcteurs d'erreurs, présente trois contributions principales, dans la première contribution nous avons proposé deux schémas de décodage itératif, l'un pour les codes en blocs concaténés en séries généralisés GSCB à base des codes RS et l'autre pour les codes en blocs concaténés en parallèle généralisés GPCB. Puis, nous avons aussi adaptés les paramètres de pondération α et β, qui sont utilisés dans le schéma de Pyndiah de manière empirique. Cette adaptation est effectuée à l'aide d'une formule qui donne la valeur de α en fonction de la variance de l'information extrinsèque normalisée du mot de code délivrée par le décodeur élémentaire. Nous avons ensuite étudié l'impact de plusieurs facteurs sur les performances des codes GSCB/GPCB, à savoir : le code élémentaire, le nombre d'itérations, la structure et la taille de l'entrelaceur. Nous avons évalués les performances de ces codes sur un canal gaussien utilisant la modulation BPSK. D'après les simulations, nous remarquons que les performances des codes GSCB/GPCB s'améliorent avec l'augmentation du nombre de multi-blocs, ainsi que le nombre d'itérations, ces performances sont encore mieux si on utilise un entrelaceur aléatoires. Pour un $TEB = 10^{-5}$, les performances du code GSCB-RS(63,39) sont à 2.1 dB par rapport à la limite de Shannon, et pour le cas du GPCB, les performances du code GPCB-RS(67,59) sont à 2.3 dB par rapport à la limite de Shannon. Les paramètres α et β que nous avons adoptés donnent des performances qui se comparent favorablement avec celles obtenus en utilisant les paramètres prédéterminés. La comparaison de ces deux codes montre que les codes GSCB est légèrement meilleurs que GPCB. Dans la deuxième contribution, nous avons proposé une version à sortie soft du décodeur Dual-R-m. Ce dernier propose des formules permettant de calculer la fiabilité de chaque bit avec une complexité réduite par rapport à celle de Pyndiah. Puis, nous avons proposé un schéma de décodage itératif pour les codes en blocs produits. Ensuite, nous avons évalué les performances des codes produits à base des codes BCH par simulation, en utilisant la modulation BPSK sous un canal à bruit blanc additif gaussien (AWGN). Les résultats de simulation montrent que le décodeur proposé dépasse ses concurrents présentés dans la littérature. Dans ce contexte, notre décodeur dépasse celui de Pyndiah de 0.6 dB pour certain codes. Dans La troisième contribution, nous avons mis le point sur l'utilisation des codes correcteurs d'erreur en cryptographie à clef publique. En effet, nous avons comparé deux instances du cryptosys-

tème de McEliece : la première instance se base sur les codes de Goppa et la deuxième se base sur les codes QC-MDPC. D'après les simulations, cette dernière minimise la taille de la clé publique, de plus ces deux instances sont parmi les candidats qui résisteraient aux attaques quantiques.

Mots clés :
Théorie d'information, codage canal, Codes RS, Codes produit, Codes concaténée en parallèles généralisés, Codes concaténée en séries généralisés, décodage itératif, décodage Turbo, cryptosystème McEliece , code goppa, QC-MDPC, Algorithme de Bit Fliping.

Abstract

This thesis, which focuses on error-correcting codes, yields three main contributions, in the first contribution, we have proposed two iterative decoding schemes, one for generalized serial concatenated block codes GSCB based on RS codes and the other for generalized parallel, concatenated block codes GPCB. We have also adapted the weighting parameters α and β that are empirically used in the Pyndiah scheme. This adaptation is carried out using a formula that gives the value of α as a function of the variance of the normalized extrinsic information of the code word delivered by the elementary decoder. We then studied the impact of several factors on the performance of GSCB/GPCB codes, namely the elementary code, the number of iterations, the structure and the size of the interleaver. We have also evaluated the performance of these codes on a Gaussian channel using BPSK modulation. According to the simulations, we have noticed that the performances of the GSCB/GPCB codes improve with the increase of the number of multi-blocks as well as the number of iterations. These performances are even better if we use a random interleaver. For a $BER = 10^{-5}$, the performance of the GSCB-RS(63,39) code is 2.1 dB with respect to the Shannon limit, and for the case of the GPCB, the performance of the GPCB-RS(67 .59) is within 2.3 dB of the Shannon limit. The parameters α and β that we have adopted yield performances which are similar to those obtained by using the predetermined parameters. The comparison of these two codes shows that the codes GSCB are slightly better than GPCB. In the second contribution, we have proposed a soft output version of the Dual-R-m decoder. This latter offers formulas to calculate the reliability of each bit with a reduced complexity compared to that of Pyndiah. Then, we have proposed an iterative decoding scheme for the product block codes. After that, we have evaluated the performances of the product codes based on the BCH codes by simulation using the BPSK modulation under an Additive White Gaussian Noise (AWGN) channel. The simulation results show that the proposed decoder outperforms its competitors presented in the literature. In this context, our decoder exceeds that of Pyndiah by 0.6 dB for certain codes. In the third contribution, we have focused on the use of error-correcting codes in public-key cryptography. Indeed, we compared two instances of the McEliece cryptosystem : the first is based on Goppa codes and the second is based on QC-MDPC codes. This latter, according to the simulations, minimizes the size of the public key. Moreover,

these two instances are among the candidates that would resist quantum attacks.

KeyWords :
RS codes, iterative decoding,, generalized parallel concatenated codes, Chase decoding,generalized serial concatenated block, Turbo decoding, McEliece cryptosystem, QC-MDPC codes, Goppa code, postquantum cryptography, Bit fliipping algorithm.

Liste des figures

Liste des tableaux

Liste des Acronymes

TURBO	Toggle Until Regenerations Bring Optimality
MLD	Maximum Liklyhood Decoding
BCH	Bose Chaudhuri Hoquengame
llr	Log Likelihood Ratio
AWGN	Additive White Gaussian Noise
HIHO	Hard-input Hard-output
SIHO	Soft-input Hard-output
SISO	Soft-input Soft-output
SCB	Serially Concatenated Block Codes
BER	Bit Error Rate
SNR	Signal to Noise Ratio
BPSK	Binary Phase Shift Keying
BTC	Block Turbo Codes
CTC	Convolutionel Turbo Codes
ITD	Iiterative Threshold Decoding
RS	Reed-Solmon
TEB	Taux d'Erreur Binaire
LDPC	Low Density Parity Cheek
MDPC	Moderate Density Parity-Check
QC-MDPC	Quasi-Cyclic MDPC
erfc	Fonction d'erreur complémentaire
MDS	Minimum Distance Separable
WEF	Weight Enumerator Function
IRWEF	Input Redundancy Weight Enumerator Function
PPCM	Plus Petit Commun Multiple
GSCB	Generalized Serially Concatenated Block Codes
GPCB	Generalized Parallel Concatenated Block Codes
PCB	Parallel Concatenated Block Codes
BMA	Berlekamp Massey Algorithm

Introduction générale

0.1 Contexte Général

Au cours du XXème siècle jusqu'à nos jours, les technologies de communication sont en évolution rapide et la quantité d'information à transmettre devient de plus en plus importante. Cette évolution concerne en premier lieu les systèmes de communication qui ont pour objectif soit l'échange ou le stockage des informations digitales tous en assurant la qualité des services de ces systèmes, ce qui favorise le besoin intensif d'outils de la théorie des codes pendant le traitement de l'information.
Dans la majorité des cas, l'échange de données se fait en utilisant un canal de communication non fiable et ces données sont susceptibles d'être altérées par le bruit. Pour faire face aux problèmes introduits par les canaux de transmission, les concepteurs ont envisagé deux grandes solutions. La première solution pour lutter contre ces erreurs est l'augmentation de la puissance émise (énergie d'émission), cette solution est très coûteuse, alors la solution alternative se base sur la détection et la correction des erreurs par codage de canal qui consiste à ajouter de la redondance de façon intelligente dans les données à transmettre ou à sauvegarder, afin de les protéger des effets indésirables. Pour répondre à ces exigences, la communauté scientifique tente d'établir des systèmes de communication toujours plus performants et innovants.
En 1948, la théorie de l'information a été établie par Claude Shannon [1] sous la forme d'une modélisation mathématique, en se basant sur les travaux de Hamming qui a inventé les premiers codes correcteurs non triviaux en 1947. Dans cet article, Shannon a prouvé que si le taux de transmission est plus petit que la capacité du canal, il existe des codes qui peuvent assurer la transmission des informations avec une probabilité d'erreur arbitrairement petite. Bien que ce travail ait montré l'existence de bons codes, il n'a pas indiqué comment ces codes peuvent être construits. Depuis l'établissement de cette approche mathématique de l'information, les chercheurs ont découvert de nombreuses classes de codes et étudié divers aspects de codage.

La chronologie de l'apparition des codes mérite de citer quelques dates clefs qui ont marquées l'histoire de la théorie des codes, parmi les travaux célèbres :

En 1954, Müller invente une nouvelle classe de codes porte son nom[2], pour la détection d'erreurs pour lesquels propose ensuite un algorithme de décodage pour la correction d'erreurs [3].
En 1955, Elias invente les codes convolutifs [4]. Il invente aussi presque en même temps les codes produits [5] que Pyndiah réussit à décoder de façon itérative quasi-optimale et peu complexe en 1994 [6].
En 1957 Prange invente les codes cycliques [7]. Deux ans plus tard et en 1959 Bose, Chaudhury et Hockenghem inventent une nouvelle classe des codes cycliques, appelés BCH [8][9]. Dans la même année les codes de Golay ont vu le jour. En 1960 Reed et Solomon inventent les codes Reed-Solomon [10] qui sont la classe des codes cycliques.
En 1963, Gallager [11] publie sa thèse sur les codes LDPC, qui vont être oubliés pendant 30 ans malgré leurs performances et l'importance théorique de ces travaux car les calculateurs numériques de l'époque n'étaient pas assez rapides et trop coûteuses.
En 1966 on voit apparaitre les premiers codes concaténés avec les travaux de Forney.[12], il propose un schéma de codage en concaténant en série deux codes en bloc : un code de Reed-Solomon et un code binaire.
Les Codes correcteurs d'erreur peuvent être divisés en deux classes : les codes en bloc et les codes convolutifs. Les codes en bloc comprennent aussi les codes linéaires et les codes non linéaires. Les codes cycliques sont une sous-classe importante des codes en blocs linéaires qui ont une structure algébrique inhérente de décodage. Par exemple, cette classe comprend les codes de Hamming, de Golay, de BCH (Bose Chaudhuri Hoc-quenghem) et de RS (Reed-Solomon). Les techniques de décodage aussi ont évolué en parallèle de l'invention de ces familles de codes de plus en plus complexes et puissantes pour corriger des erreurs. On peut citer les plus célèbres, en 1969, Viterbi [13] applique une technique de programmation dynamique au calcul de bornes théoriques de taux d'erreurs et de vitesse de convergence au décodage des codes convolutifs sur un treillis. Plusieurs travaux ont abouti à un algorithme de décodage à décision douce appelé aussi algorithme de type Chase [14] qui génère une liste des mots candidats à partir des positions les moins fiables du mot reçu et en sélectionne ensuite le mot à distance euclidienne minimale du mot reçu. Les performances de cet algorithme sont très efficases, mais sa complexité devient très grande le rendant impraticable pour des codes de grande longueur.
Après les années 1960, les progrès dans le domaine des codes correcteurs d'erreur se faisant d'une manière lente au niveau des performances. Il a fallu attendre l'année 1993 dont Claude Berrou, Alain Glavieux et Thitimajshima [15] présentent les turbo codes avec un décodage itératif permettant d'atteindre la limite de Shannon à quelques fractions de dB près. Ils ont vraiment été une révolution pour le domaine des télécommunications. L'in-

vention des turbo codes a ensuite suscité la redécouverte des codes LDPC par MacKay en 1995 [16] après avoir été oublié dès leurs apparition à cause de la complexité des calculs de décodage des codes LDPC pour les technologies disponibles à cette époque. Dans ce sens on peut citer les travaux de Pyndiah et al. [6] qui ont traité les codes produits à base des codes BCH, ils ont utilisé les paramètres de pondérations α et β avec des valeurs empiriques ou expérimentales. Mais le principe de turbo décodage n'a été appliqué aux codes en bloc concaténés en série et en parallèle qu'après une décennie et ceci dans les travaux [17, 18, 19]. Ces dernier travaux ont été réalisé sur les codes BCH et se base sur une version modifié du schéma de pyndiah, qui utilisent des paramètres de pondération α et β prédéterminés. Ces paramètre doivent être changés pour chaque code ce qu'est pénible. Pour pallier à ce problème, les auteurs de [19] ont proposé une formule permettant de calculer de façon adaptée les paramètres précités α et β. Dans cette thèse nous étendons les travaux [17, 18, 19] pour les codes concaténés à base des codes RS. Ensuite nous proposons une version SISO du décodeur Dua-R-M [20] dont le but est de l'utiliser comme brique de base pour décoder les turbo-codes à base des codes BCH [20]. Ce turbo décodeur est un concurrent de celui de Pyndiah. D'ailleurs il le dépasse en termes de performance TEB vis-à-vis SNR.

L'application des codes correcteurs d'erreur ne se limite pas dans la protection de l'information contre la perturbation des canaux mais il peut aussi être utilisé pour préserver l'information contre les modifications intentionnelles. En effet McEliece [21] a proposé en 1978 un cryptosystème basé sur les codes de Goppa pour préserver la confidentialité de l'information, mais son cryptosystème n'a pas été mis en œuvre à cause de la taille de la clé public qui est assez importante et difficile à gérer. Pour contourner ce problème, la communauté scientifique ont proposé plusieurs instances pour le cryptosystème de McEiece en remplaçant les codes goppa par plusieurs autres codes comme les codes RS, BCH, et Reed Muller et d'autres, mais ces instances ont été cassés par les cryptanalystes à cause du fait que ces codes sont plus structurés ce qui facilitent leurs attaques. Néanmoins, récemment d'autres familles ont été proposées qui semblent plus efficaces parce qu'elles sont peu structurés et permettent aussi de réduire efficacement la taille de la clé, ce sont les codes MDPC quasi-cyclique. Cette dernière instance est parmi les candidats présentés à NIST pour remplacer le cryptosystème actuels (RSA,...) pour faire face aux attaques quantiques. Dans ce contexte nous avons réalisé une étude comparative entre deux instances du cryptosystème de McEliece , la première est basée sur le code de Goppa et la deuxième basé sur les codes QC- MDPC. Cette étude montre l'efficacité de l'instance basée sur les QC-MDPC

0.2 Organisation du manuscrit

Le **premier chapitre** présente le cadre théorique de ce travail, les notions de base de la théorie d'informations en particulier la notion de la capacité d'un canal de communication numérique. Il introduit également les différents éléments d'une chaîne de transmission numérique à savoir l'émetteur, le récepteur et le canal de transmission il présente aussi quelques types de canaux de communication.Nous présentons des généralités sur les codes correcteurs d'erreurs en introduisant la classification des différentes familles de codes correcteurs d'erreurs, ainsi que les critères généraux du codage et décodage. En abordant les différentes familles des algorithmes de décodage avec une description détaillée pour quelques décodeurs. Ensuite il présente une discussion sur les trois familles des algorithmes de décodage HIHO, SIHO et SISO avec des exemples pour chacune de ces familles.
Pour **le chapitre 2** nous présentons les algorithmes de construction des codes composés. Ensuite, nous présentons les schémas de codage étudiés : La construction des codes concaténés en parallèle généralisés, et des codes concaténés en sérié généralisés. Notre contribution dans ce travail est de faire adapté les paramètres α et β pour les codes concaténés en série à base des codes RS et aussi pour les codes concaténés parallèle à base RS. Nous clôturons par la présentation des résultats de simulations et leurs interprétations, basé sur l'étude de l'effet de plusieurs variables sur les performances de ces codes, à savoir : le code élémentaire, le nombre d'itérations, la taille et la structure de l'entrelaceur . Finalement, nous faisons une comparaison entre les codes GSCB et GPCB d'un point de vue performances.
Ensuite, le **chapitre 3** est consacré à la présentation d'un nouveau décodeur des codes en bloc produits. Tout d'abord, nous introduisons le décodeur SIHO appelé Dual-R-m [20]. Ensuite, nous développons une version SISO appelé Dual-R-m. Ce dernier est utilisé pour développer un schéma de décodage turbo plus efficace. les performances du décodeur proposé sont évaluées par simulation, puis elles sont comparées avec les décodeurs concurrent se trouvant dans la littérature.
Dans le **chapitre 4**, nous introduisons l'application des codes correcteurs d'erreurs pour la sécurité. En particulier, nous mettons le point sur l'utilisation des codes en cryptographie asymétrique. Il s'agit du cryptosystème de McEliece basé sur les codes de Goppa, ainsi que celui basé sur les codes MDPC quasi-cyclique. Nous présentons les algorithme de génération de clés, de chiffrement et déchiffrement des deux cryptosystèmes. Ensuite, nous évaluons la complexité de chacun de ces cryptosystème en se basant sur la simulation.
Finalement, nous terminons ce manuscrit par la synthèse des travaux effectués, des principaux résultats obtenus ainsi que des perspectives sur les futurs travaux en soulignant

les points importants qui n'ont pas été développés.

Chapitre 1

Éléments de communication numérique et de codage

1.1 Introduction

Un système de communication numérique est un moyen de transport de l'information d'un usager qui est la source à un autre qui est le destinataire. En général, géographiquement éloigné, en utilisant un support physique comme les câbles, la fibre optique ou encore la propagation sur un canal radio-électrique avec une fiabilité possible, Le système est appelé numérique, ce qui signifie que l'information est représentée par une séquence de symboles choisis parmi un alphabet fini (binaire, par exemple).
Dans ce chapitre, nous présentons les éléments de base d'un système de communication numérique, nous introduisons également des définitions des outils et notions utilisées dans le domaine des codes correcteurs d'erreurs, des rôles qu'ils peuvent jouer dans les systèmes de télécommunication et de sauvegarde de données. Ainsi, la première partie de ce chapitre porte sur les éléments de base de la chaîne de transmission et des modèles de canaux qui permettent de simuler les performances des codes. La deuxième partie de ce chapitre détaille les caractéristiques et les classes des codes correcteurs d'erreurs, en particulier les codes étudiés dans les autres chapitres de cette thèse. Nous détaillerons ensuite le processus de décodage et ses critères et un ensemble de décodeurs que nous améliorons ou que nous comparons avec nos travaux.

1.2 Chaîne de transmission numérique

Le schéma classique d'une chaîne de transmission numérique est représenté sur la figure 1.1 permet de mieux comprendre le cheminement de l'information de la source au destinataire.

1.2.1 Codage/décodage de source

Permet de limiter le nombre d'éléments binaires nécessaires à la représentation de l'information contenue dans le message de la source. C'est une opération de compression qui fait appel à des algorithmes bien connus (Huffman, par exemple). Dans le cas où la

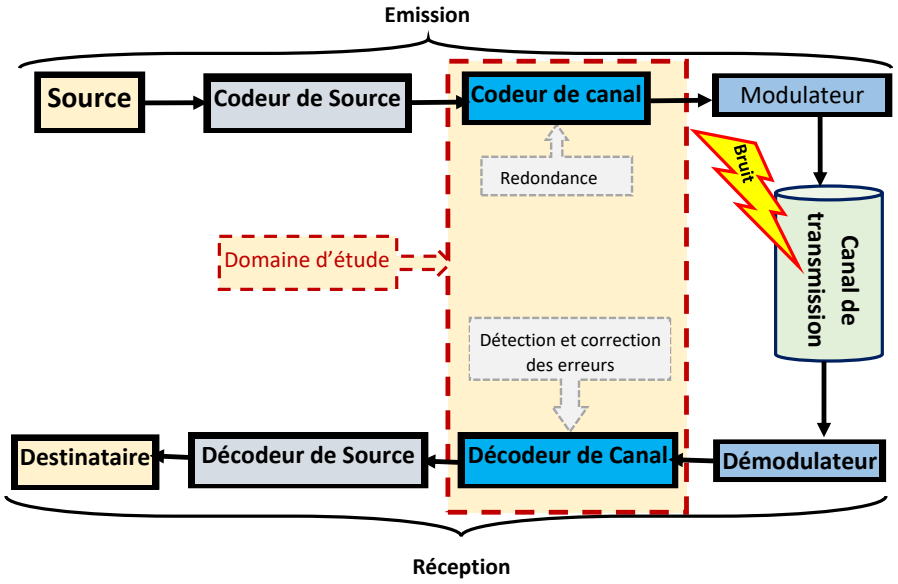


FIGURE 1.1 – Modèle d'un système de communication numérique.

source de données est analogique, le signal doit d'abord être échantillonné, ensuite, quantifié et enfin, transformé en une suite de 0 et 1 par les techniques de conversion analogique numérique comme la modulation à impulsion codée (MIC). Le décodeur source permet alors l'interprétation des données compressées et la reconstruction du média émis. Les limites du codage de source sont fixées par la théorie de l'information (premier théorème de Shannon)[1]. Au-delà de cette limite, le codage s'effectue avec perte, c'est-à-dire qu'à partir des données codées nous ne sommes plus en mesure de restituer exactement le message d'origine. Le codage source n'étant pas l'objet de notre étude, la séquence issue de ce codage sera considérée comme étant une source de message à éléments binaires indépendants avec une probabilité d'occurrence de 1=2 pour chacun des éléments 0 et 1.

1.2.2 Codage/décodage canal

Ces deux bloques sont bien détaillés dans la suite de ce chapitre. Aussi tous nos travaux objets de cette thèse, y sont focalisés. Une brève description qu'on peut leur donner est la suivante :

- Le codeur consiste à protéger les informations contre les erreurs dues au support de transmission par l'ajout de la redondance d'une manière rigoureuse pour contrôler l'information utile.
- le décodeur est basé sur des algorithmes permettant de reconstituer l'information émise à partir des données reçues même si ces dernières contiennent des erreurs liées au canal de transmission.

1.2.3 La Modulation/ Démodulation

Le modulateur adapte la séquence codée au canal physique (liaison satellite, filaire ou sans fils). Plus particulièrement, on gère l'ordre et le type de la modulation utilisée (Modulation de phase, de fréquence ou d'amplitude à plusieurs états), la puissance transmise des symboles, ou encore l'étalement, quand on est en présence d'un système à accès multiple par étalement de codes. A la réception le démodulateur permet de transformer les symboles reçus en une séquence binaire et de séparer les utilisateurs dans le cas d'un système d'accès multiple.

1.2.3.1 Modulation

Les signaux transmis sont construits à partir d'un signal sinusoïdal, dit porteur, qui est capable de se propager dans le milieux de transmission et qui est de la forme [22] :

$$p(t) = \alpha h(t) cos(2\Pi f_0 t + \phi) \tag{1.1}$$

où f_0 est la fréquence de la porteur, α et ϕ désignent respectivement l'amplitude et la phase du signal et h(t) est la fonction porte définie par h(t)=1 si $t \in [0, T_s]$ =0 sinon T_s représente la durée d'émission du signal élémentaire et s'appelle la durée symbole.
Dans le cas de la modulation de phase PSK, seule la phase du signal porteur varie. la modulation de phase la plus simple est BPSK. A chaque bit en entrée, le modulateur délivre l'un des deux signaux

$$s(t) = \pm\alpha cos(2\Pi f_0 t) h(t) \tag{1.2}$$

une représentation des signaux dans le plan de Fresnel, appelée constellation, est illustrée sur la figure 1.2.

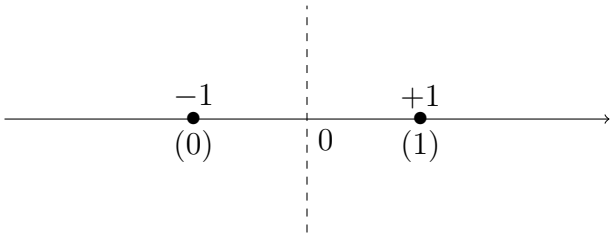


FIGURE 1.2 – Constellation associée à la modulation de phase à 2 états (BPSK).

Remarque 1.1. *Il existe d'autres types de modulation :*
- *Modulation de phase à quatre états QPSK(Quadratic Phase Shift Keying).*
- *Modulation de fréquence FSK(Frequency Shift Keying).*
- *Modulation d'amplitude ASK(Amplitude Shift Keying).*

Les modulations BPSK et QPSK sont équivalente [23], et nous adoptons la première dans cette thèse.

1.2.4 Modèles de canaux de transmission

1.2.4.1 Généralités

Un canal de transmission reçoit un message d'entrée et restitue un message de sortie. D'un point de vue abstrait nous le considèrerons comme une entité qui fait le lien entre deux alphabets comme montre la figure 1.3.

$$X = \{x_1, ..., x_K\} \xrightarrow{Canal} Y = \{y_1, ..., y_J\}$$

$$\boxed{X} \longrightarrow T_r \longrightarrow \boxed{Y}$$

FIGURE 1.3 – Modélisation du canal discret sans mémoire

- **Canal discret :**
  Les deux alphabets d'entrée et de sortie sont des alphabets discrets qui comportent un nombre fini de symboles.
- **Canal sans mémoire :**
  Le symbole courant de sortie ne dépend que du symbole courant d'entrée et ne dépend pas des précédents ni des suivants.
- **Canal discret sans mémoire :**
  Le canal effectue donc le couplage entre deux alphabets. S'il est sans mémoire, lorsqu'il reçoit le symbole x_j de l'alphabet d'entrée, il a une probabilité de transmettre le symbole y_j de l'alphabet de sortie. Le fonctionnement du canal est une matrice T_r de $K \times J$ probabilités conditionnelles $p(y_j|x_k)$:

$$T_r = \begin{pmatrix} p(y_1|x_1) & \dots & p(y_J|x_1) \\ \vdots & \ddots & \vdots \\ p(y_1|x_K) & \dots & p(y_J|x_K) \end{pmatrix}$$

  Le canal est sans mémoire si pour tout $(x_1, ..., x_n)$ transmis et $(y_1, ..., y_n)$ reçu, on a :

$$P(y_1, ..., y_n|x_1, ..., x_n) = P(y_1|x_1)...P(y_n|x_n) \tag{1.3}$$

  Dans ce travail, nous utilisons principalement les canaux discrets sans mémoire.

1.2.4.2 Canal binaire symétrique

Le canal binaire symétrique est le canal le plus simple possible puisque l'entrée et la sortie du canal sont binaires. Ce canal est représenté par la figure 1.4.

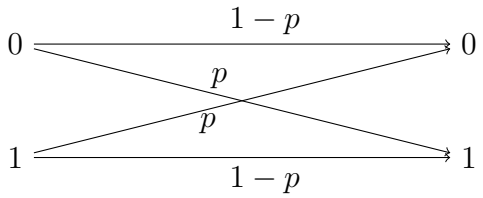


FIGURE 1.4 – Canal binaire symétrique

Ce canal est caractérisé par les 2 probabilités de transition suivantes :

$$P(Y = 0|X = 1) = P(Y = 1|X = 0) = p$$

$$P(Y = 0|X = 0) = P(Y = 1|X = 1) = 1 - p$$

p est appelé la probabilité d'inversion.

1.2.4.3 Canal binaire à effacement

C'est un modèle dérivé du canal binaire sans mémoire dans lequel une erreur de transmission génère une forme d'onde différentes de celles associées aux deux symboles binaires ainsi, l'alphabet de sortie comprend trois caractères, le troisième étant l'effacement "E" qui correspond à une erreur de transmission. La figure 1.5 représente le canal binaire à effacement.

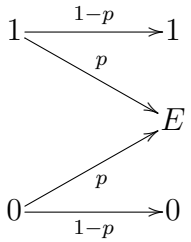


FIGURE 1.5 – Canal binaire à effacement

1.2.4.4 Canal à bruit blanc additif gaussien

Le canal à bruit blanc additif gaussien est le canal à alphabet de sortie continu le plus utilisé. Il permet de modéliser les canaux dont le bruit prédominant est le bruit thermique. Nous considérerons que la bande occupée par le signal en entrée du canal est en bande de base limitée à B et que le bruit additif est stationnaire, blanc, gaussien et de densité spectrale de puissance unilatérale N_0. La puissance du bruit N est égale à : $N = N_0B$. A la réception, le démodulateur optimal comprend un filtre adapté limitant la bande de bruit à B. Le théorème de l'échantillonnage implique que $2BT$ échantillons suffisent pour représenter les signaux en entrée et en sortie du démodulateur pendant

une durée T. Considérons une modulation bipodale où les symboles émis x_i à l'instant i peuvent prendre les valeurs $+\sqrt{E_b}$ ou $-\sqrt{E_b}$. A l'instant i, on peut exprimer la sortie y_i du démodulateur optimal comme suit :

$$y_i = x_i + n_i \tag{1.4}$$

n_i est l'échantillon réel de bruit blanc centré dont la densité de probabilité est gaussienne :

$$p(n_i) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{n_i^2}{N_0}\right) \tag{1.5}$$

La variance de l'échantillon de bruit n_i est égale à :

$$\sigma^2 = \frac{N}{2B} = \frac{N_0}{2} \tag{1.6}$$

Ainsi la densité de probabilité de y_i conditionnellement à x_i est :

$$p(y_i|x_i) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{|y_i - x_i|^2}{N_0}\right) \tag{1.7}$$

Le canal à bruit additif blanc gaussien peut être représenté symboliquement par le diagramme de la figure 1.6.

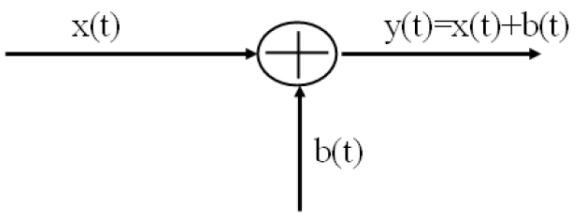


FIGURE 1.6 – Représentation d'un canal à bruit additif blanc gaussien

On peut montré que la probabilité d'erreur binaire P_{eb} pour la BPSK s'exprime sous la forme [22]

$$P_{eb} = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{1.8}$$

où $erfc$ est la fonction d'erreur complémentaire $erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^{+\infty} e^{-t^2} dt$ et $E_b/N0$ est le rapport signal sur bruit. Cette probabilité d'erreur binaire est tracée en fonction de $E_b/N0$(exprimé en dB) sur la figue 1.7.

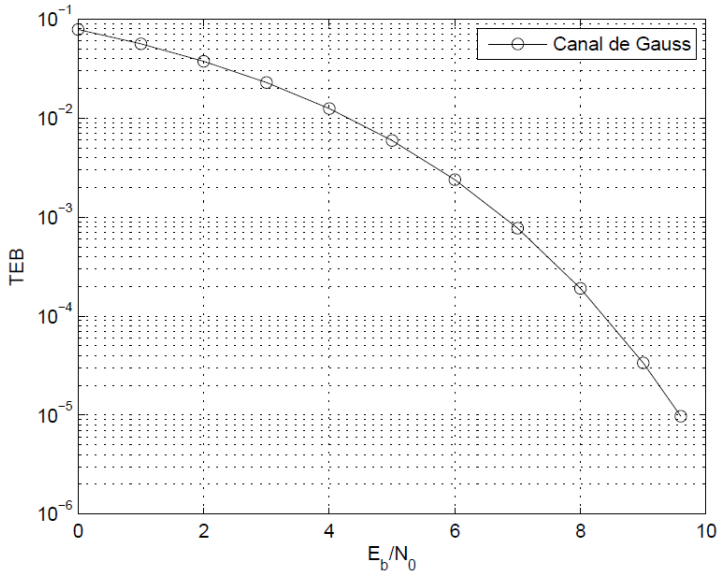


FIGURE 1.7 – Canal de Gauss

1.2.4.5 Canal de Rayleigh

Le canal de Rayleigh est un modèle plus élaboré que le canal gaussien et qui est représentatif des communications sur le canal radio-mobile. Ce modèle de canal porte le nom de canal à trajet multiples. Il est utilisé lorsqu'on se trouve en présence d'obstacles qui prennent des formes diverses. Le canal de Rayleigh peut être modélisé par :

$$Y = AX + B \tag{1.9}$$

où X caractérise l'entrée du canal, B est un bruit additif gaussien, et A est une variable aléatoire qui suit une loi de Rayleigh de variance σ^2, dont les échantillons α sont mutuellement indépendants :

$$P_A(\alpha) = \frac{\alpha}{\sigma^2} \exp\left(\frac{-\alpha^2}{2\sigma^2}\right) \tag{1.10}$$

On montre que la probabilité d'erreur binaire pour une BPSK sur le canal de Rayleigh s'écrit comme [22] :

$$P_{eb} = \frac{1}{2}\left[1 - \frac{1}{\sqrt{1+2\sigma^2}}\right] \tag{1.11}$$

La figure 1.8 représente la probabilité, P_{eb}, d'erreur en fonction E_b/N_0 pour le canal de Rayleigh. D'après les courbes représentant la probabilité d'erreur pour les deux canaux, on constate que la probabilité se dégrade lorsque on passe du canal de Gauss au canal de Rayleigh.

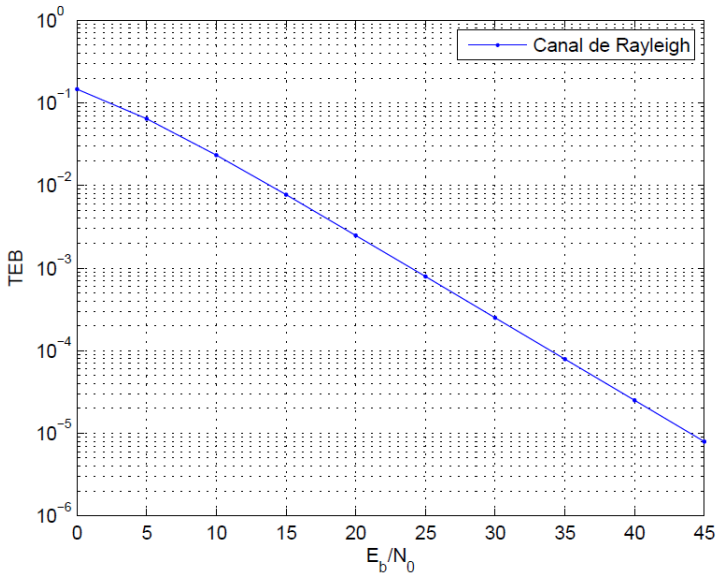


FIGURE 1.8 – Canal de Rayleigh

1.2.5 Capacité d'un canal de transmission et théorème fondamental de codage

La théorie de codage doit son origine 'a la théorie de l'information introduite en 1948 par ingénieur américaine Claude Elwood Shannon [1]. Cette dernière fixe les limites ultimes en terme de codage correcteur d'erreurs par l'intermédiaire du théorème fondamental de codage. Celui-ci énonce qu'en codant l'information correctement, celle-ci peut être transmise a travers un canal parasité sans perte de son contenu 'a la réception. Elle se base sur une mesure quantitative de l'information, c'est a dire qu'elle ne s'intéresse pas au sens du message du point de vue sémantique, mais seulement 'a la quantité d'information que le message véhicule.
L'un des principaux résultats de la théorie de l'information réside dans le théorème de codage formulé par Shannon [1] et dont l'énoncé est la suivante :

Théorème 1.1. *Pour un taux $R \leq C$, ou C est la capacité du canal, Il existe un code de longueur N tel que sa probabilité d'erreur est arbitrairement petite lorsque N tend vers l'infini. Inversement, sur un canal sans mémoire, si $R > C$; alors quel que soit le code considéré et quelle que soit sa longueur N, la probabilité d'erreur moyenne par mot ne peut pas tendre vers 0.*

Dans ses travaux, Shannon [1] a évalué la capacité d'un canal de propagation gaussien en fonction du rapport signal sur bruit et de sa largeur de bande. La capacité (exprimée en bit/s) peut se définir par le débit d'information maximal que peut supporter le canal tout en assurant une transmission sans erreur. l'étude de la capacité du canal permet donc de fixer les limites asymptotiques d'un système de codage. En effet, dans le cas d'un canal

AWGN binaire symétrique la capacité peut être exprimée par :

$$C = \frac{1}{2}\log_2\left(1 + R\frac{E_b}{N_0}\right) \tag{1.12}$$

On voit que la capacité, borne supérieure sur R pour une transmission fiable, dépend elle-même du taux de codage. La limite de Shannon correspond 'a l'inégalité suivante :

$$C \leq \frac{1}{2}\log_2\left(1 + R\frac{E_b}{N_0}\right) \tag{1.13}$$

qu'il faut résoudre en R pour trouver le compromis optimal entre le taux de codage et le rapport signal sur bruit $\frac{E_b}{N_0}$. On trouve que :

$$\frac{E_b}{N_0} \geq \frac{2^{2R} - 1}{2R} \tag{1.14}$$

qui donne le SNR par bit d'information minimal en fonction du taux de codage ; c'est une condition nécessaire et suffisante pour assurer une transmission numérique sans erreurs. A largeur de bande constante, une augmentation de R nécessite un niveau de puissance par bit d'informations plus élevé, ou d'une manière équivalente une puissance transmise plus élevée. L'équation (1.2.4) montre aussi l'existence d'une valeur seuil du rapport $\frac{E_b}{N_0}$ au dessous de laquelle il est théoriquement impossible d'assurer une communication sans erreurs, et ceci quelle que soit la largeur du canal dont on dispose. Cette valeur est obtenue quand le taux de codage R tend vers zéro, auquel cas on trouve facilement : $\frac{E_b}{N_0} \simeq \ln 2$; ce qui correspond 'a une valeur en décibels égale à -1.591. Cette valeur représente la limite de Shannon sur les performances d'un système codé. Le but dans tout système de communication est de se rapprocher le plus possible de la capacité du canal de transmission. Or, le théorème fondamental n'explicite pas le procédé de codage permettant d'atteindre cette limite.

1.3 Classification et caractérisation des codes correcteurs d'erreurs

1.3.1 Introduction

Le principe des codes correcteurs d'erreurs, est de rajouter une information supplémentaire redondante de manière à détecter et éventuellement corriger de possibles erreurs de transmission. Dans cette section, nous allons présenter les grandes classes de codes correcteurs d'erreurs et les techniques de décodage pour chaque types. Comme le montre La figure 1.9 les codes correcteurs d'erreur peuvent être divisés en deux classes : les codes en bloc et les codes convolutifs. Les codes en bloc comprennent aussi les codes linéaires et les codes non linéaires. Les codes cycliques sont une sous-classe importante des codes en blocs linéaires qui ont une structure algébrique inhérente de décodage. Par exemple, cette classe comprend les codes de Hamming, de Golay, de BCH (Bose Chaudhuri Hocquenghem) et de RS (Reed-Solomon).

Au début des années 90, une nouvelle famille de codes correcteurs d'erreurs a été découverte par C. Berrou [24] : les turbo codes. Cette famille de codes correcteurs d'erreurs est construite par concaténation de codes convolutifs. Les turbo codes convolutifs sont le résultat de deux innovations majeures : la concaténation de deux codes pour le codage et le décodage itératif. Le décodage itératif est appliqué à des décodeurs élémentaires à entrées et à sorties pondérées. Le turbo décodage a de bonnes performances et une complexité calculatoire raisonnable. Le concept général de décodage itératif appliqué à des décodeurs élémentaires a ensuite été étendu aux codes produits [6] et aux codes LDPC [11].

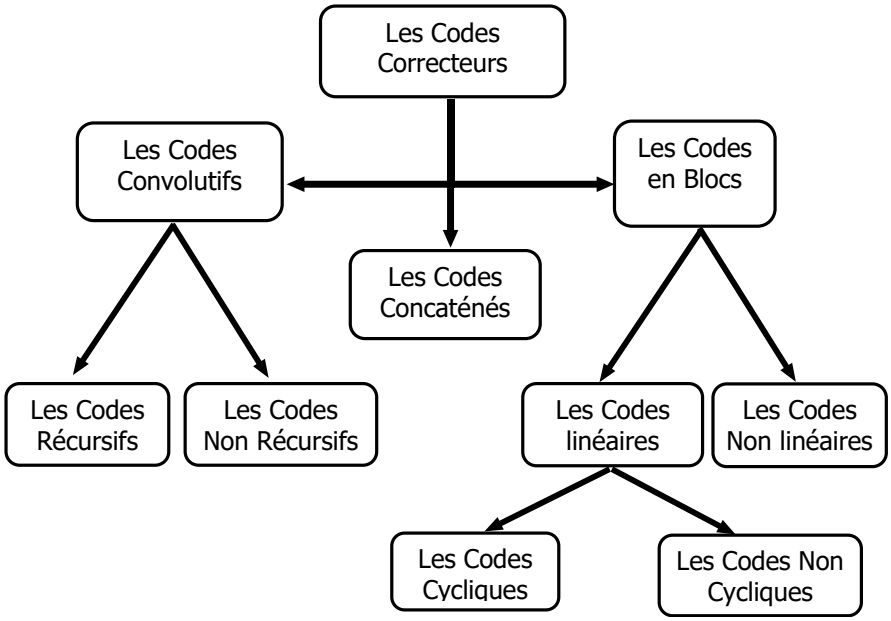


FIGURE 1.9 – Famille des codes correcteurs d'erreur

1.3.2 Mesure des performances des codes correcteurs d'erreurs

Les codes correcteurs d'erreurs sont une des solutions permettant d'améliorer la qualité des communications numériques. Le principe du codage canal est d'introduire de la redondance dans la séquence d'information binaire afin de corriger les erreurs de transmission durant la réception de l'information. Un bon code doit avoir :

- un bon rendement (taux) c'est-à-dire un grand nombre de bits d'information par rapport aux bits codés,
- une bonne capacité de détection et correction d'erreurs,
- une procédure de décodage (et de codage) suffisamment simple et rapide.

On mesure les performance d'un code correcteur d'erreurs par l'évaluation du Taux d'Erreur Binaires (TEB), à la sortie du décodeur au niveau du récepteur, en fonction du rapport signal à bruit ($SNR = \frac{E_b}{N_0}$ où E_b étant l'énergie reçue par symbole binaire d'information transmis et N_0 la densité spectrale de puissance du bruit contenue dans la bande).

$$TEB = \frac{\text{nombre de symboles binaires erronés}}{\text{nombre de symboles binaires transmis}}$$

L'efficacité d'un codeur de canal se reflète dans le gain de codage comme le montre cette

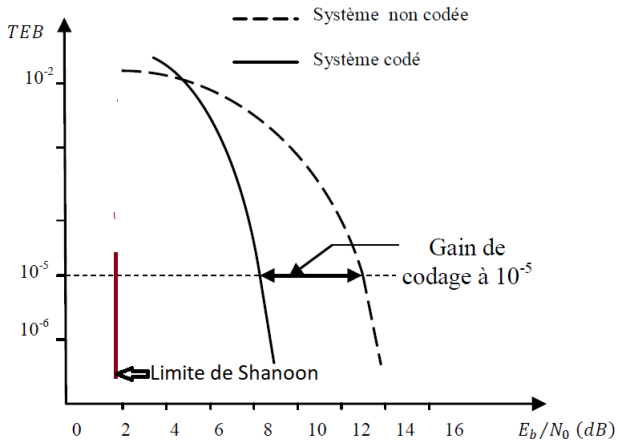


FIGURE 1.10 – Illustration du gains de codage pour un $TEB = 10^{-5}$

figure 1.10).

1.4 Codes en bloc linéaires

1.4.1 Définitions et propriétés

Soit F_q un corps fini à q éléments, où q est une puissance d'un nombre premier. Nous listons dans cette sous section les propriétés les plus populaires concernant les codes linéaires,telles que le poids de Hamming, la distance de Hamming et la distance minimale qui permette de déterminer les capacités de détection et de correction du code.

Définition 1.1. *Les codes linéaires sont des codes dont chaque mot du code C est obtenu après une transformation linéaire des bits du mot initial du message M. En d'autres termes, un code linéaire est un ensemble de vecteurs de longueur n, qui est stable par addition et par multiplication par un scalaire, dont une base contient k vecteurs (indépendants).*
Donc c'est un code correcteur (n, k) caractérisé par une application linéaire φ tel que :

$$\begin{aligned} \varphi : F^k &\to F^n \\ M &\to C = \varphi(M) \end{aligned}$$

k est la dimension du code et n est la longueur du code, où n sera évidemment choisi supérieur à k.

Définition 1.2. ***(Rendement)*** *: Le rendement R d'un code en bloc (N,K) est égal à :* $R = \frac{K}{N}$

Définition 1.3. ***(Poids de Hamming)*** *:*
Soit $A = (a_1, \cdots, a_n)$ un mot de code C, le poids de Hamming de A est le nombre de coordonnées non nulles de celui-ci. On le note $W_h(A)$. Autrement dit, on a :

$$W_h(A) = |\{i, a_i \neq 0\}|$$

Exemple : $x = [001100],\ w(x) = 2$

Définition 1.4. ***(Distance de Hamming)*** *:*
La distance de Hamming d_h entre deux mots, $A = (a_1, \cdots, a_n)$ et $B = (b_1, \cdots, b_n)$ est le nombre de coordonnées distinctes entre les deux. Autrement dit, on a :

$$d_h(A, B) = |\{\ i,\ a_i \neq b_i\ \}| \tag{1.15}$$

Exemple : $x = [001100]$ et $y = [001111]$, $d_H(x, y) = 2$

Définition 1.5. ***(Distance minimale)*** *:*
La distance minimale d est la plus petite distance non nulle entre deux mots de code différents.

Définition 1.6. ***(Capacité de détection)*** *:*
Soit C un code linéaire de paramètres $(n, k, d)_q$. La capacité de détection de C est la quantité $d - 1$. On dit alors que C est $(d-1)-$détecteur.

Définition 1.7. ***(Capacité de correction)*** *:*
Soit C un code linéaire de paramètres $(n, k, d)_q$. La capacité de correction de C, notée t, est définie par :

$$t = \left[\frac{(d-1)}{2}\right] \tag{1.16}$$

1.4.1.1 Bornes sur les paramètres d'un code

Proposition 1.1. ***Borne de Gilbert-Varshamov*** *:*
Il existe un code linéaire de longueur n, de dimension k et de distance minimale $\geq d$ sur F_q lorsque l'inégalité suivante est vérifiée :

$$q^{n-k} - 1 > \sum_{i=1}^{d-2} \binom{n-1}{i} (q-1)^i \tag{1.17}$$

Proposition 1.2. ***Borne de Singleton*** *:*
Notons $|C|$ le cardinal du code C, alors :

$$|C| \leq q^{n-d+1} \tag{1.18}$$

Dans le cas d'un code linéaire C de paramètres [n, k, d] on a $|C| = q^k$ et ce qui précède implique

$$d \leq n - k + 1 \tag{1.19}$$

Lorsque d = n - k + 1 on dit que le code est parfait ou MDS (Maximum Distance Separable).[25]

1.4.2 Représentation matricielle des Codes en bloc linéaires

1.4.2.1 Matrice génératrice

Les mots d'information et les mots de code sont représentés par des vecteurs. Soit $U = [u_1, u_2, ..., u_K]$ un mot d'information composé de K symboles d'information et $V = [v_1, v_2, ..., v_N]$ le mot de code associé composé de N symboles. On a la relation matricielle suivante entre le mot d'information U et le mot de code associé V :

$$V = U.G \tag{1.20}$$

G est la matrice génératrice du codeur de dimension $K \times N$.

$$G = \begin{pmatrix} g_{00} & \cdots & g_{0N} \\ \vdots & \ddots & \vdots \\ g_{K0} & \cdots & g_{KN} \end{pmatrix} \tag{1.21}$$

Un code en bloc linéaire peut être défini comme un sous espace vectoriel à $K < N$ dimensions construit suivant 1.21.
Il est toujours possible -en combinant les lignes entre elles- de mettre la matrice génératrice G sous la forme systématique suivante :

$$G = [I_K \ P] = \begin{pmatrix} 1 & 0 & \dots & 0 & p_{11} & p_{12} & \cdots & p_{1N-K} \\ 0 & 1 & \dots & 0 & p_{21} & p_{22} & \cdots & p_{2N-K} \\ \vdots & \vdots & \ddots & 0 & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & p_{K1} & p_{K2} & \cdots & p_{KN-K} \end{pmatrix} \tag{1.22}$$

Exemple 3 : Code de parité $C_1(3, 2)$:

$$G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

1.4.2.2 Matrice de contrôle

Pour chaque code C linéaire en bloc (N,K) on associé un code linéaire en bloc dual (N,N - K). Soit H la matrice génératrice de ce code dual. Chacun des mots de code c du

code C est orthogonal à tous les mots de code du code dual

$$cH^T = 0$$

Puisque cette relation est valide pour tous les mots de code du code C, on a la relation entre la matrice génératrice G du code C et H :

$$GH^T = 0$$

Si la matrice génératrice G est systématique de la forme 1.22, est de la forme suivante :

$$H = [P^T \ I_{N-K}] = \begin{pmatrix} p_{11} & p_{21} & \dots & p_{K1} & 1 & 0 & \dots & 0 \\ p_{12} & p_{22} & \dots & p_{K2} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{1N-K} & p_{2N-K} & \dots & p_{KN-K} & 0 & 0 & \dots & 1 \end{pmatrix} \tag{1.23}$$

La matrice H est appelée matrice de contrôle ou matrice de parité du code C.

Définition 1.8. ***(Fonction syndrome)*** *:*
Soient C un code linéaire et H une matrice de contrôle de C. On définit la fonction syndrome associée à H de la manière suivante :

$$\begin{aligned} S : F_{q^n} &\to F_{q^{n-k}} \\ c &\to S(c) = c.H^T \end{aligned}$$

1.4.3 Codes cycliques

Les codes cycliques représentent la famille de codes la plus importante. D'un point de vue pratique, ce sont les codes les plus utilisées car leur mise en œuvre est facile et ils admettent de bons algorithmes de décodage. D'un point de vue théorique, ils possèdent une structure algébrique intéressante. Les codes cycliques les plus connus sont les codes de Hamming, BCH, Reed-Solomon, Résidus quadratiques, etc.

Définition 1.9. *Les codes cycliques bénéficient de toutes les propriétés des codes en blocs linéaires en plus de la propriété cyclique. C'est-à-dire si* $C = (c_0, c_1, \dots, c_{n-2}, c_{n-1})$ *est un mot de code alors le décalage cyclique de i produit* $C = (c_{n-i}, c_{n-i+1}, \dots, c_0, \dots, c_{n-i-1})$ *est aussi un mot de code.*

1.4.3.1 Propriétés

Parmi les propriétés des codes cycliques on trouve :
– La linéarité : la somme de deux mots de code est un mot de code.
– Toute rotation circulaire d'un mot de code est un mot de code.

– Le générateur G(x) de degré n-k factorise tous les mots de code et est décrit :

$$G(X) = g_0 + g_1 x + \ldots + g_{n-k} x^{n-k}$$

1.4.3.2 Principe de codage

Deux modes de codage existent : le codage par multiplication et l'autre par division.

Codage par multiplication :

Tout mot de code C(x) peut donc s'écrire sous la forme :
$C(x) = D(x).G(x)$, où $D(x) = d_0 + d_1 x + \ldots + d_{k-1} x^{k-1}$est le message d'information à coder. G(x) et également un facteur de $x^n + 1$.

Codage par division :

Est utilisé pour le codage systématique, et consiste à coder le message D(x) à partir du polynôme générateur G(x), de la manière suivante :
- Multiplication de D(x) par x^{n-k}.
- Division de $x^{n-k}D(x)$ par G(x) : $x^{n-k}D(x) = Q(x).G(x) + R(x)$ où R(x) est le reste de la division.
- Addition de $x^{n-k}D(x)$ et R(x) : $C(x) = x^{n-k}D(x) + R(x)$
L'équation ci-dessus donne le mot de code C(x) sous forme systématique et les composantes de R(x) sont les symboles de redondance (ou de parité). [26] **Exemple :** code de Hamming de paramètres $[7, 4, 3]_2$ Les codes de Hamming ont été introduits en 1950 par Richard Hamming. Il a construit le premier code correcteur véritablement efficace, capable de corriger une erreur. Le code de paramètres $[7, 4, 3]_2$ est souvent pris comme exemple, car il est à la fois une illustration amusante du principe d'un code correcteur d'erreur et est un peu plus élaboré que la simple répétition de l'information. [27]

Exemple 1.1. *m=1011*
Les positions de bits de contrôle sont les positions 2^n *où* $n = 0, 1, 2, \ldots$
Le reste sont des bits de message.
- k_1= *parité* (m_1, m_2, m_4) = *parité (1,1,1) = 1*
- k_2= *parité* (m_1, m_3, m_4)= *parité (1,0,1) = 0*
- k_4= *parité* (m_2, m_3, m_4) = *parité (1,0,1) = 0*

TABLE 1.1 – Codage de Hamming

7	6	5	4	3	2	1
m_4	m_3	m_2	k_4	m_1	k_2	k_1
1	0	1	0	1	0	1

Cas d'une erreur :

Pour détecter une erreur, il suffit de recalculer les bits de contrôle et les comparer avec les bits de contrôle du message reçu.
Supposons que nous avons reçu c = 1110101. Les vérifications donnent :
- k_1= *parité (1,1,1) =1*
- k_2= *parité (1,1,1) =1*
- k_4= *parité (1,1,1) =1*

k_2 *et* k_4 *sont différents sur les bits de contrôle de message c. Donc le message reçu est erroné.*

1.4.4 Les codes BCH

Les codes BCH sont des codes cycliques binaires. Ils ont été découverts par Hocquenghem [8] puis par Bose et Ray-Chaudhuri [9]. Ils permettent de construire des codes avec les paramètres suivants :

$$N = 2^m - 1$$

$$N - K \leq mt$$

$$d_{min} \geq 2t + 1$$

un code BCH est défini par son polynôme générateur $g(p)$. Le polynôme générateur $g(p)$ possède parmi ses racines les $2t$ puissances consécutives de α, où $\alpha \in GF(2^m)$. Ainsi, le polynôme générateur $g(p)$ est le produit des polynômes minimaux associés à α, α^2, α^3, . . . , α^{2t} sans qu'aucun des polynômes ne soit répété :

$$g(p) = PPCM\left[m_1(p), m_2(p), m_3(p), ..., m_{2t}(p)\right] \tag{1.24}$$

avec PPCM : plus petit commun multiple.
Comme dans $GF(2^m)$, α^i et α^{2i} sont racines du même polynôme minimal, pour la détermination de g(p), il suffit de ne considérer que les puissances impaires de α.

Soit un mot de code $c(p) = c_0 + c_1 p + ... + c_{N-1}p^{N-1}$. Tout mot de code aura comme racines α, α^2, α^3, . . . , α^{2t}. Cette propriété peut s'écrire comme suit : $c(\alpha^i) = c_0 + c_1\alpha^i + .. + c_{N-1}\alpha^{(N-1)i} = 0, \quad \forall i\ 1 \leq i \leq 2t$.
Cette relation peut s'écrire aussi sous la forme matricielle suivante :

$$\begin{pmatrix} c_0 & c_1 & \dots & c_{N-1} \end{pmatrix} \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha & \alpha^2 & \dots & \alpha^{2t} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{N-1} & \alpha^{2(N-1)} & \dots & \alpha^{2t(N-1)} \end{pmatrix} = 0$$

La matrice de parité s'écrit donc :

$$H = \begin{pmatrix} 1 & \alpha & \dots & \alpha^{N-1} \\ 1 & \alpha^2 & \dots & \alpha^{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2t} & \dots & \alpha^{2t(N-1)} \end{pmatrix}$$

1.4.4.1 Décodage des codes BCH

Le décodage classique des codes BCH se déroule en 3 étapes distinctes :

1. Calcul des $2t$ composantes du syndrome $S = (s_1\ s_2 ... s_{2t})$
2. Recherche du polynôme localisateur d'erreurs $\sigma(p)$
3. Détermination des racines du polynôme $\sigma(p)$ et correction des erreurs.

Syndrome
Soit $c(p) = c_0 + c_1 p + ... + c_{N-1}p^{N-1}$ le mot de code , $y(p) = y_0 + yp + ... + y_{N-1}p^{N-1}$ le mot reçu et $e(p) = e_0 + e_1 p + ... + e_{N-2}p^{N-2} + e_{N-1}p^{N-1}$ le mot d'erreur. On a alors :

$$y(p) = c(p) + e(p) \tag{1.25}$$

La première étape consiste à calculer le syndrome S défini par un vecteur à $2t$ éléments. Par définition le syndrome est égal au produit du vecteur reçu par la matrice de contrôle transposée :

$$S = (s_1, s_2, ..., s_{2t}) = yH^T = (y_0, y_1, ..., y_{N-1})H^T$$

Les $2t$ composantes du syndrome sont calculées comme suit :

$$s_i = y(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i) \ \ \forall\ i\ 1 \leq i \leq 2t$$

Il faut noter que s'il n'y a pas d'erreur ($e(p) = 0$), les $2t$ composantes du syndrome sont nulles. Supposons que le mot d'erreur soit composé de ν erreurs (*avec* $v \leq t$) : $e(p) = p^{j_1} + p^{j_2} + ... + p^{j_\nu}$ avec $0 \leq j_1 \leq j_2 ... \leq j_\nu \leq N-1$.
Localisateur d'erreur
Définissons le polynôme suivant :

$$\sigma(p) = \prod_{l=1}^{\nu}(1 + \alpha^{j_l}p) = \sigma_0 + \sigma_1 p + \sigma_2 p^2 + ...\sigma_\nu p^\nu \tag{1.26}$$

Pour déterminer les coefficients σ_l on utilisera les relations dites de Newton :

$$s_i = \sum_{j=1}^{\nu} \sigma_j s_{i-j} \quad pour\ i > \nu \tag{1.27}$$

cette relation peut être écrite sous la forme matricielle suivante :

$$\begin{pmatrix} s_{\nu+1} \\ s_{\nu+2} \\ \vdots \\ s_{2\nu} \end{pmatrix} = \begin{pmatrix} s_1 & s_2 & \dots & s_\nu \\ s_2 & s_3 & \dots & s_{\nu+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_\nu & s_{\nu+1} & \dots & s_{2\nu-1} \end{pmatrix} \begin{pmatrix} \sigma_\nu \\ \sigma_{\nu-1} \\ \vdots \\ \sigma_1 \end{pmatrix}$$

La résolution de ce système permet de déterminer le polynôme localisateur d'erreurs et par la suite les positions d'erreurs.
Dans la littérature, on trouve plusieurs algorithmes pour résoudre ce système. Parmi ses algorithmes on cite :

- L'algorithme PGZ : permet de calculer le vecteur (σ_j) ; solution du système linéaire. Cette méthode est utilisé seulement pour des valeurs de $t \leq 6$.

- L'algorithme de Berlekamp-Messey : c'est la méthode la plus efficace pour résoudre ce système linéaire.
- L'algorithme d'Euclide : utilisé généralement dans l'implémentation matérielle des décodeurs des codes BCH.

La figure 1.11 présente le diagramme du décodeur BCH.

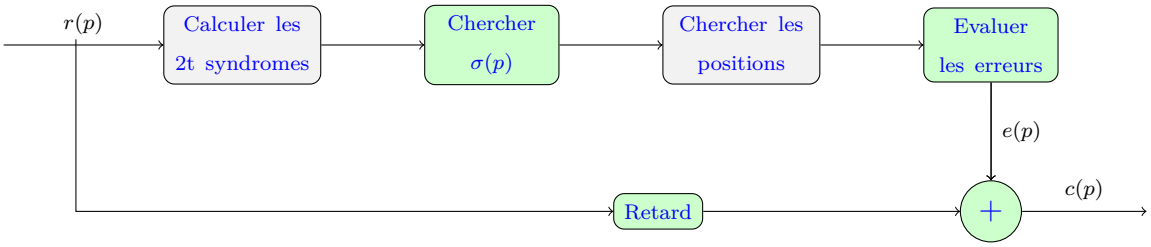


FIGURE 1.11 – Décodeur des codes BCH

1.4.5 Codes RS

Les codes de RS sont des codes cycliques et plus précisément des codes BCH non binaires, ils portent le nom de leurs inventeurs Irving Reed et Gustave Solomon. Ils ont été introduits en 1960. Il s'agit de codes adaptés à la correction de paquets d'erreurs.

Définition 1.10. ***(code RS)*** *:*
Soit F_q un corps fini. Soient k et n deux entiers tels que $k \leq n$. Soient $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ des éléments distincts de F_q . Un code de Reed-Solomon $RS_{(n,k)}(\alpha)$ sur F_q est défini par :

$$RS_{(n,k)}(\alpha) = \{(P(\alpha_1), P(\alpha_2), \ldots, P(\alpha_n)) \in F_{q^n};\ P \in F_q[X],\quad deg(P) < k\} \tag{1.28}$$

Ce code est de longueur n, de dimension k et de distance minimale d = n - k + 1

Propriété : Une matrice génératrice G du code de Reed-Solomon est donnée par :

$$\mathrm{G} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & \ldots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \ldots & \alpha_n^{k-1} \end{pmatrix}$$

Théorème 1.2. *Soit F_q un corps fini et F_{q^m} son extension de degré m. Le code $RS_{(n,k)}$ de longueur $n = q^m - 1$ est défini par :*

$$RS_{(n,k)} = \{(v_1, v_2, \ldots, v_n) \in F_{q^m}^n, \forall i \in [\,1, 2t\,] \sum_{j=1}^{n} \equiv v_j \alpha^{(j-1)i} = 0 \in F_{q^m}\} \tag{1.29}$$

où α est une racine primitive fixée de F_{q^m}.

Il existe aussi le code de Reed Solomon Généralisé GRS, il correspond donc à un code RS dont on multiplie les colonnes par des éléments non nuls de F_{q^m}. La multiplication d'une colonne par un élément non nul ne modifiant pas la distance minimale, les codes GRS ont les mêmes paramètres que les codes RS [27].

Définition 1.11. ***(Code GRS)*** *:*
Un code de Reed-Solomon généralisé $GRS_{(\alpha,v)}$ sur F_{q^m} de longueur $n \leq q^m - 1$ et de dimension k est défini par un ensemble appelé son support $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ qui est un n-uplet d'éléments non nuls ordonnés de F_{q^m} deux à deux distincts, et par un vecteur appelé son multiplicateur $v = (v_1, v_2, \ldots, v_n)$, qui est une suite d'éléments non nuls de F_{q^m}. On a alors :

$$GRS_k(\alpha; v) = \{(v_1.P(\alpha_1), v_2.P(\alpha_2), \ldots, v_n.P(\alpha_n));\ P \in F_{q^m}[X];\ deg(P) < k\} \quad (1.30)$$

1.4.5.1 Décodage des codes RS

Les codes RS sont des codes BCH non binaires. Dans le cas d'un code BCH binaire il suffit de connaitre la position de l'erreur pour inverser le bit trouvé. Cependant, la connaissance de la valeur de l'erreur e_{i_j} est indispensable pour décoder les codes RS. Le polynôme évaluateur d'erreurs est défini par :

$$\Omega(p) = \sigma(p)S(p)\ mod\ p^{2t} \quad (1.31)$$

où S est le polynôme syndrome défini par :

$$S(p) = \sum_{j=1}^{2t} S_j p^{j-1} \quad (1.32)$$

Pour calculer la valeur de l'erreur on utilise l'algorithme de Forney [28], [29], [30] :

$$e_{i_j} = \frac{\Omega(\alpha^{-i_j})}{\dot{\sigma}(\alpha^{-i_j})} \quad (1.33)$$

La procédure de décodage des erreurs des codes RS peut se résumer en trois étapes :
- Calculer les $2t$ syndromes à partir du mot $y(p)$ $(s_i = y(\alpha^i))$;
- Calculer le polynôme localisateur d'erreurs $\sigma(p)$, et rechercher ses zéros et déduire les positions des erreurs ;
- Evaluer les erreurs.

* **Algorithme de Berlekamp-Messey**

– **Correction des erreurs**
Cet algorithme a été proposé par Berlekamp [31] et son fonctionnement a été éclairci par Massey [32].
La première étape consiste à déterminer le polynôme de degré minimal $\sigma^{(1)}(p)$ dont les coefficients vérifient la première égalité de Newton 1.27. Si les coefficients vérifient aussi la seconde égalité de Newton, on pose alors $\sigma^{(2)}(p) = \sigma^{(1)}(p)$. Sinon, on ajoute un terme correctif à $\sigma^{(1)}(p)$ pour obtenir $\sigma^{(2)}(p)$. Le polynôme $\sigma^{(2)}(p)$

est alors le polynôme à degré minimum dont les coefficients vérifient les deux premières égalités de Newton. On continue cette procédure pour former $\sigma^{(3)}(p)$ puis $\sigma^{(4)}(p)$ jusqu'à obtenir $\sigma^{(2t)}(p)$. Ce dernier polynôme est le polynôme localisateur $\sigma(p)$.
Soit $\sigma^{(i)}(p) = 1+\sigma_1^{(i)}p+\sigma_2^{(i)}p^2+...+\sigma_{l_i}^{(i)}p^{l_i}$ le polynôme de degré l_i déterminé à la i^{eme} étape et vérifiant donc les i premières équations. Pour déterminer $\sigma^{(i+1)}(p)$ on applique la procédure suivante :
on commence par calculer d_i comme suit :

$$d_i = s_{i+1} + \sigma_1^{(i)}s_i + \sigma_2^{(i)}s_{i-1} + ... + \sigma_{l_i}^{(i)}s_{i+1-l_i}$$

Si $d_i = 0$, alors $\sigma^{(i+1)}(p) = \sigma^{(i)}(p)$. Sinon, on remonte à l'itération ρ telle que $d_\rho \neq 0$ et $\rho - l_\rho$ soit maximal (avec l_ρ degré du polynôme $\sigma^{(\rho)}(p)$). On obtient alors :

$$\sigma^{(i+1)}(p) = \sigma^{(i)}(p) + d_i d_\rho^{-1} p^{i-\rho}\sigma^{(\rho)}(p)$$

– **Correction des erreurs et des effacements**
Etant donné d la distance minimale du code, ν le nombre d'erreurs et μ le nombre d'effacements contenu dans le mot reçu. Alors la distance minimale de Hamming entre les mots du code est réduite à $d-\mu$ dans les positions non effacées. Il s'en suit que la capacité du code égale $\lfloor (d-\mu-1)/2 \rfloor$, d'où la relation en ν et μ est :

$$d > 2\nu + \mu.$$

Pour la correction des effacements, le principal changement à apporter à la procédure de décodage des codes RS décrite auparavant, est l'introduction d'un polynôme localisateur des effacements, $\tau(p)$, défini comme suit :

$$\tau(p) = \prod_{l=1}^{\mu}(1+y_{i_l}p), \qquad 1 \leq l \leq \mu$$

où $y_{i_l} = \alpha^{i_l}$ et l dénote la position d'effacement.
La procédure de décodage est similaire au décodage avec erreur seulement avec quelques modifications. Un polynôme syndrome modifie est nécessaire

$$T(p) = S(p)\tau(p) + 1 \mod p^{2t+1}$$

L'algorithme de Berlekamp peut être appliqué pour trouver $\sigma(p)$ avec les modifications suivantes :

1. Le discriminant est définie par :

$$d_i = T_{i+\mu+1} + \sum_{j=1}^{l_i}\sigma_j^{(i)}T_{i+\mu+1-j}, \quad avec \quad d_0 = T_{\mu+1}.$$

2. L'algorithme s'arrête si la condition suivante est vérifiée

$$i \geq l_{i+1} + t - 1 - \frac{\mu}{2}.$$

Après avoir déterminé $\sigma(p)$, un polynôme évaluateur d'erreurs et d'effacements, $W(p)$, est calculer comme suit :

$$W(p) = [1 + T(p)]\,\sigma(p)\ mod\ p^{2t+1}. \tag{1.34}$$

de plus, un polynôme localisateur d'errata, $\phi(p)$, est calculé,

$$\phi(p) = \tau(p)\sigma(p). \tag{1.35}$$

En utilisant l'algorithme de Forney modifié on détermine la valeur d'errata par la relation

$$e_{j_l} = \frac{(\alpha^{j_l})^{2-b} W(\alpha^{-j_l})}{\phi\prime(\alpha^{-j_l})}, \tag{1.36}$$

$1 \leq l \leq \nu$, pour les erreurs, et

$$f_{i_l} = \frac{(y_{i_l})^{-2-b} W(y_{j_l}^{-1})}{\phi\prime(y_{j_l}^{-1})}, \tag{1.37}$$

$1 \leq l \leq \mu$, pour les effacements.

1.4.6 Codes de Goppa

Les codes de Goppa sont une classe de codes correcteurs d'erreurs linéaires qui a été proposée par Valery Denlsovlch Goppa en 1970. Tous comme les codes cycliques sont spécifiés en termes de polynôme générateur, les codes de Goppa sont décrits en terme de polynôme de Goppa G(z),ils proviennent de la famille des codes dits alternants eux-mêmes définis à partir des codes de Reed-Solomon généralisés, Ils ont fait une apparition marginale en cryptographie dans le cryptosystème de McEliece. Généralement, les codes de Goppa sont considérés comme de ≪ *bons* ≫ codes linéaires puisqu'ils permettent de corriger jusqu'à $\binom{n^k}{lb(n)}$ erreurs, tel que $lb(n) = ln(n)/ln(2)$.

1.4.6.1 Construction du codes de Goppa

Définition 1.12. ***(Code de Goppa) :***
Soit q une puissance d'un nombre premier p, $m > 0$ et $n \leq q^m$. Les codes de Goppa peuvent être définis sur le corps fini F_{q^m} à partir d'un polynôme unitaire, dit de Goppa, $G(x) \in F_{q^m}$ de degré t et d'un ensemble $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ appelé support, tel que $\forall i = \{1, 2, \ldots, n\}$, $G(\alpha_i) \neq 0$. Le code de Goppa de longueur n est défini par :

$$\Gamma(L, G) = \{C = (c_1, c_2, \ldots, c_n), \in F_q^n\ ;\ \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} \equiv 0\, mod\, G(x)\} \tag{1.38}$$

Le théorème suivant sert à calculer la matrice de contrôle du code de Goppa à partir de deux matrices :

Théorème 1.3. *$L = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ un sous-ensemble du corps à q^m éléments et $G(X)$*

un polynôme de degré t sur F_{q^m}. Une matrice de contrôle du code H_q de Goppa $\Gamma(L,G)$ peut être construite à partir d'une matrice H dans F_{q^m} de la façon suivante :

$$Y = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \vdots & \vdots & \dots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \dots & \alpha_n^{t-1} \end{pmatrix} \quad ; \quad Z = \begin{pmatrix} \frac{1}{G(\alpha_1)} & 0 & \dots & 0 \\ 0 & \frac{1}{G(\alpha_2)} & \ddots & \vdots \\ \vdots & \ddots & \frac{1}{G(\alpha_3)} & 0 \\ 0 & \dots & 0 & \frac{1}{G(\alpha_4)} \end{pmatrix}$$

$$H = Y.Z = \begin{pmatrix} \frac{1}{G(\alpha_1)} & \frac{1}{G(\alpha_2)} & \dots & \frac{1}{G(\alpha_n)} \\ \frac{\alpha_1}{G(\alpha_1)} & \frac{\alpha_2}{G(\alpha_2)} & \dots & \frac{\alpha_n}{G(\alpha_n)} \\ \dots & \dots & \dots & \dots \\ \frac{\alpha_1^{t-1}}{G(\alpha_1)} & \frac{\alpha_2^{t-1}}{G(\alpha_2)} & \dots & \frac{\alpha_n^{t-1}}{G(\alpha_n)} \end{pmatrix} \tag{1.39}$$

En considérant une base de F_{q^m} sur F_q, le code de Goppa $\Gamma(L,G)$ est bien défini comme étant l'ensemble des vecteurs $x \in F_{q^n}$ tels que $H_q.x^T = 0$.

Remarque 1.2. *Suivant [27], la construction d'une autre matrice de contrôle de ce code avec la forme $H_{G^2} = Y\,.\,Z \in M_{2t,n}(F_{2^3})$ calculée avec G^2 donne le même résultat.*

Exemple 1.2. *Soit $\Gamma(L,G)$ un code de Goppa classique binaire irréductible de paramètres $[8,2,5]_2$ avec $L = F_2[X]/(X^3+X+1)$ et $G(X) = X^2+X+1$.*

Regardons comment construire ce code :

Rappel :

$F_{p^m} = F_{p[X]}/P_m(X)$ où P est un polynôme irréductible sur $F_p[X]$ de degré m.
Soient $F_8 = F_2[X]/(X^3+X+1)$ et $\beta = X \bmod (X^3+X+1)$ une racine primitive de F_8.
On considère le code de Goppa classique binaire irréductible $\Gamma(L,G)$, où

$$L = \{\alpha_1, \alpha_2, \dots, \alpha_8\} \;=\; \{0, 1, \beta_1, \beta^2, \dots, \beta^6\} \;=\; F_{2^3}$$

On a $t = deg(G(X)) = 2$ (le code est 2-correcteur), car G est bien irréductible. De plus, on a $m = 3$, $n = 2^3 = 8$, $k \geqslant n - m.t = 8 - 3 \times 2 = 2$ et $d \geq t+1 = 3$.
On peut également pré-calculer les tables suivantes :

TABLE 1.2 – Somme de deux éléments dans F_8

$\oplus$	0	1	β	β^2	β^3	β^4	β^5	β^6
0	0	1	β	β^2	β^3	β^4	β^5	β^6
1	1	0	β^3	β^6	β	β^5	β^4	β^2
β	β	β^3	0	β^4	1	β^2	β^6	β^5
β^2	β^2	β^6	β^4	0	β^5	β	β^3	1
β^3	β^3	β	1	β^5	0	β^6	β^2	β^4
β^4	β^4	β^5	β^2	β	β^6	0	1	β^3
β^5	β^5	β^4	β^6	β^3	β^2	1	0	β
β^6	β^6	β^2	β^5	1	β^4	β^3	β	0

TABLE 1.3 – Produit de deux éléments dans F_8

$\cdot$	0	1	β	β^2	β^3	β^4	β^5	β^6
0	0	0	0	0	0	0	0	0
1	0	1	β	β^2	β^3	β^4	β^5	β^6
β	0	β	β^2	β^3	β^4	β^5	β^6	1
β^2	0	β^2	β^3	β^4	β^5	β^6	1	β
β^3	0	β^3	β^4	β^5	β^6	1	β	β^2
β^4	0	β^4	β^5	β^6	1	β	β^2	β^3
β^5	0	β^5	β^6	1	β	β^2	β^3	β^4
β^6	0	β^6	1	β	β^2	β^3	β^4	β^5

TABLE 1.4 – Calcul des coefficients de la matrice H_G

X	X	X	$(1, \beta, \beta^2)$	G(X)	$G^{-1}(X)$	$X.G^{-1}(X)$
α_1	0	0	(0, 0, 0)	1	1	0
α_2	1	1	(1,0,0)	1	1	1
α_3	β	β	(0,1,0)	$\beta^2+\beta+1$	β^2	$\beta+1$
α_4	β^2	β^2	(0,0,1)	$\beta+1$	$\beta^2+\beta$	β^2+1
α_5	β^3	$\beta+1$	(1,1,0)	$\beta^2+\beta+1$	β^2	$\beta^2+\beta+1$
α_6	β^4	$\beta^2+\beta$	(0,1,1)	β^2+1	β	$\beta^2+\beta+1$
α_7	β^5	$\beta^2+\beta+1$	(1,1,1)	β^2+1	β	β^2+1
α_8	β^6	β^2+1	(1,0,1)	$\beta+1$	$\beta^2+\beta$	$\beta^2+\beta$

Construisons une matrice de contrôle de ce code en utilisant la forme $H_G = Y.Z \in M_{t,n}(F_{2^3})$ *calculée avec G :*

$$H_G = \begin{pmatrix} G(\alpha_1)^{-1} & G(\alpha_2)^{-1} & G(\alpha_3)^{-1} & \cdots & G(\alpha_8)^{-1} \\ \alpha_1.G(\alpha_1)^{-1} & \alpha_2.G(\alpha_2)^{-1} & \alpha_3.G(\alpha_3)^{-1} & \cdots & \alpha_8.G(\alpha_8)^{-1} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & \beta^2 & \beta^4 & \beta^2 & \beta & \beta & \beta^4 \\ 0 & 1 & \beta^3 & \beta^6 & \beta^5 & \beta^5 & \beta^6 & \beta^3 \end{pmatrix}$$

Chaque élément de cette matrice, qui est un élément de F_{2^3}, est déroulé, c'est-à-dire qu'il est projeté sur F_{2^3} puis écrit comme une colonne de 3 éléments de F_2 en utilisant la base $B = (1, \beta, \beta^2)$, pour finalement former une matrice $H_{G,2}$ de taille 6×8 comme illustré ci-dessous :

$$H_{G,2} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Pour trouver G, il faut mettre $H_{G,2}$ sous forme systématique (notée H), grâce à la méthode du pivot de Gauss, suivie d'une permutation des colonnes 6 et 7. On obtient alors :

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Comme $H = [I_6|A]$, on aura alors $G' = [A^T|I_2]$, d'où :

$$G' = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Maintenant, retrouvons $G \in M_{k,n}(F_2)$ en appliquant la permutation inverse (de celle appliquée lors du Pivot de Gauss). On obtient donc :

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

On obtient alors comme code de Goppa :
$\Gamma(L, G) = \{(0,0,0,0,0,0,0,0), (1,1,1,1,0,1,0,0), (1,1,0,0,1,0,1,1), (0,0,1,1,1,1,1,1)\}$.
On peut rapidement vérifier que l'on a $k = 2$ et $d = 5$.

Un deuxième théorème existe permette de calculer la matrice de parité avec une autre façon en ce basant sur le résultat du théorème 1.3

Théorème 1.4. *Soient $L = (\alpha_1, \alpha_2, \cdots, \alpha_n) \subseteq F_{q^m}$ un sous-ensemble du corps à q^m éléments et $G(X) = g_t.X^t + g_{t-1}.X^{t-1} + \ldots + g_1.X + g_0$ un polynôme de degré t sur F_{q^m}. Une matrice de contrôle du code de Goppa $\Gamma(L\ ,\ G)$ peut être construite dans F_{q^m} de la façon suivante :*

$$H = X.Y.Z = \begin{pmatrix} g_t & 0 & \cdots & 0 \\ g_{t-1} & g_t & \cdots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ g_1 & \cdots & g_{t-1} & g_t \end{pmatrix} . \begin{pmatrix} \frac{1}{G(\alpha_1)} & \frac{1}{G(\alpha_2)} & \cdots & \frac{1}{G(\alpha_n)} \\ \frac{\alpha_1}{G(\alpha_1)} & \frac{\alpha_2}{G(\alpha_2)} & \cdots & \frac{\alpha_n}{G(\alpha_n)} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\alpha_1^{t-1}}{G(\alpha_1)} & \frac{\alpha_2^{t-1}}{G(\alpha_2)} & \cdots & \frac{\alpha_n^{t-1}}{G(\alpha_n)} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{g_t}{G(\alpha_1)} & \frac{g_t}{G(\alpha_2)} & \cdots & \frac{g_t}{G(\alpha_n)} \\ \frac{g_{t-1}+\alpha_1.g_t}{G(\alpha_1)} & \frac{g_{t-1}+\alpha_2.g_t}{G(\alpha_2)} & \cdots & \frac{g_{t-1}+\alpha_n.g_t}{G(\alpha_n)} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\sum_{i=1}^{t} \alpha_1^{i-1}.g_i}{G(\alpha_1)} & \frac{\sum_{i=1}^{t} \alpha_2^{i-1}.g_i}{G(\alpha_2)} & \cdots & \frac{\sum_{i=1}^{t} \alpha_n^{i-1}.g_i}{G(\alpha_n)} \end{pmatrix} \tag{1.40}$$

1.4.6.2 Décodage du code de Goppa

Nous décrivons dans cette partie un algorithme de décodage efficace pour les codes de Goppa. Le décodage se fait en 3 étapes :

1. trouver le syndrome.
2. rechercher les polynômes de localisateur d'erreur (PLE) et d'évaluateur d'erreur (PEE).
3. echercher les emplacements et les valeurs d'erreurs.

Avant de présenter cet algorithme, nous avons besoin des définitions suivantes :

Définition 1.13. *Le polynôme localisateur d'erreur $E = (e_1, e_2, \ldots, e_n)$ dans $C' = C + E$ est défini par :*

$$\sigma_E = \prod_{\substack{i=1 \\ e_i=1}}^{n} (X - \alpha_i) \tag{1.41}$$

Définition 1.14. *Le polynôme évaluateur d'erreur E dans $C' = C + E$ est défini par :*

$$\eta_E(X) = \frac{d\sigma_E(X)}{dX} = \sum_{\substack{i=1 \\ e_i \neq 0}}^{n} \prod_{\substack{j=1 \\ e_j \neq 0 \\ j \neq i}}^{n} (X - \alpha_j) \tag{1.42}$$

Théorème 1.5. *Le polynôme syndrome d'un mot reçu C est donné par :*

$$S_C(X) = C.H^T.(X^{t-1}, \ldots, X, 1) \tag{1.43}$$

Remarque 1.3. *Comme X est inversible, les matrices H = Y.Z et H = X.Y.Z représentent deux fonctions syndromes ayant le même noyau, à savoir le code de Goppa Γ(L, G) [27].*

Proposition 1.3. *Pour corriger C' en C, il suffit de connaître le polynôme syndrome $S_{c'}(X)$ et d'utiliser les définitions de PLE et PEE pour résoudre ce que l'on appelle l'équation clé :*

$$S_{c'}(X) \cdot \sigma_E(X) \equiv \eta_E(X) \, mod \, G(X) \tag{1.44}$$

Algorithm 1 Décodage du code de Goppa

1: **Entrées** : $C' = C+E$ un mot reçu tel que $C \in \Gamma(L, G)$ un code de Goppa t-correcteur, E un vecteur erreur de poids $w_H(E) \leq t$, $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ le support du code et G le polynôme de Goppa.
2: **Sorties** : $C \in \Gamma(L, G)$.
3: Calculer le syndrome de C' : $S_{c'}(X)$.
4: Résoudre l'équation-clé : $S_{c'}(X) \cdot \sigma_E(X) \equiv \eta_E(X) \, mod$ G(X) pour obtenir $\sigma_E(X)$;
5: Chercher les racines de $\sigma_E(X)$ sur le support L ;
6: Construire le vecteur erreur E(X) tel que $e_i = 1$ si $\sigma_E(\alpha_i) = 0$ et $e_i = 0$ si $\sigma_E(\alpha_i) = 1$;
7: **Retourner** C = C' - E .

Pour la résolution de l'équation clé, il existe plusieurs algorithmes tels que d'Euclide étendu, de Berlekamp-Massey et de Patterson. Nous choisissons à détailler celle d'Euclide étendu car c'est simple à comprendre et direct. Il permet de trouver le plus grand diviseur commun d'entiers ou de polynômes. Nous décrirons l'algorithme tel qu'il s'applique aux polynômes, car c'est le cas qui nous intéresse ici.

1.4.6.3 Décodage par Euclide étendu

L'algorithme d'Euclide étendu détermine le plus grand diviseur commun (pgcd) de deux polynômes A(X) et B(X) appartenant à F_{2^m} avec $B(X) \neq 0$ et deg(A) > deg(B), ainsi que deux polynômes U(X) et V(X), en vérifiant la relation de Bézout de la manière suivante :

$$pgcd(A(X), B(X)) = U(X).A(X) + V(X).B(X) \tag{1.45}$$

Dans notre cas :

$$pgcd(G(X), S_{c'}(X)) = U(X).G(X) + V(X).S_{c'}(X), \; avec \; deg(S_{c'}) < deg(G) = t$$

$$pgcd(G(X), S_{c'}(X)) \, mod \, G(X) = V(X).S_{c'}(X)$$

Nous avons :

$$S_{c'}(X) = \frac{\eta_E(X)}{\sigma_E(X)} \, mod \, G(X)$$

Donc

$$pgcd(G(X), S_{c'}(X)) = \eta_E(X) \, mod \, G(X) \; et \; V(X) = \sigma_E(X) \, mod \, G(X) \tag{1.46}$$

Algorithm 2 Décodage par Euclide étendu

1: **Entrées** : $R_0 = S_{c'}(X)$; $R_1 = \mathrm{G(X)}$;
$U_0 = 1$; $V_0 = 0$; $U_1 = 0$; $V_1 = 1$; $i = 1$;
2: **Tant Que** $R_i > 0$
$Q_{i+1}(X) = quotient(R_{i-1}(X), R_i(X))$;
$R_{i+1}(X) = reste(R_{i-1}(X), R_i(X))$;
$U_{i+1}(X)$=$U_{i-1}(X) - Q_{i+1}(X).U_i(X)$;
$V_{i+1}(X)$=$V_{i-1}(X) - Q_{i+1}(X).V_i(X)$;
i++ ;
3: **Sortie** : $U_{i-1}(X), V_{i-1}(X), D(X) = R_{i-1}(X)$

Exemple 1.3. *En utilisant l'exemple 1.2 étudié précédemment, en calculant cette fois ci la matrice de contrôle avec la formule $H = X.Y.Z$ calculé avec $G^2(X) = X^4 + X^2 + 1$. Nous avons $t = 4$, donc suivant le théorème (1.4) :*

$$H = \begin{pmatrix} 1 & 1 & \beta^4 & \beta & \beta^4 & \beta^2 & \beta^2 & \beta \\ 0 & 1 & \beta^5 & \beta^3 & 1 & \beta^6 & 1 & 1 \\ 1 & 0 & \beta^3 & \beta^6 & \beta^6 & \beta^5 & \beta^3 & \beta^5 \\ 0 & 0 & \beta^4 & \beta & \beta^2 & \beta^2 & \beta & \beta^4 \end{pmatrix}$$

Supposons que nous avons reçu le mot de code erroné C' = (0,1,0,1,0,1,0,0).
Pour décoder ce mot, nous suivons ces étapes :
1 : Le polynôme syndrome de C' est $S_{C'}(X) = \beta^5.X^3 + \beta^5.X^2 + \beta.X + \beta^4$
2 : Nous cherchons les polynômes de Bézout U(X) et V(X) en utilisant l'algorithme d'Euclide étendu qui prend en entrées $S_{C'}(X) = \beta^5.X^3 + \beta^5.X^2 + \beta.X + \beta^4$ et $G^2(X) = X^4 + X^2 + 1$.
le résultat de cet algorithme est $\beta^5 = G^2(X).(\beta^2.X + \beta^5) + S_{C'}(X).(\beta^4.X^2 + \beta^5.X)$.
Donc $\sigma_E(X) = \beta^4.X^2 + \beta^5.X$
3 : Nous calculons les racines de $\sigma_E(X)$, les résultats sont 0 et β.
4 : Nous obtenons comme vecteur d'erreur : E = (1, 0, 1, 0, 0, 0, 0, 0).
Donc le mot de code (sans erreur) est C = C'- E =(1, 1, 1, 1, 0, 1, 0, 0).

1.4.7 Les codes QC-LDPC/MDPC

1.4.7.1 Les codes QC-LDPC

Les codes LDPC (Low Density Parity Check) ont été introduits en 1963 par Gallager [33]. Il a même proposé un algorithme de décodage efficace pour ces codes. Les codes LDPC sont basés sur des matrices de contrôle pseudo aléatoires de faible densité.
Un code LPDC est un code dont la matrice de contrôle de parité H est de faible densité (matrice creuse). C'est à dire que les éléments de cette matrice sont majoritairement nuls.

1.4.7.2 Propriétés

Définition 1.15. *On note T l'opérateur de décalage standard sur $\mathbb{F}^n$.*
Un code linéaire est dit l-quasi-cyclique si et seulement s'il est invariant par rapport à T^l.

Maintenant, nous donnons une définition des codes quasi-cycliques LDPC/MDPC.

Définition 1.16. *Soit w le poids de chaque ligne d'une matrice de contrôle de parité $H \in \mathbb{F}_2^{r\times n}$. La densité est le taux $\frac{w}{n}$. Un $(n,k,w)-LDPC$ code C est un code linéaire de longueur n et de dimension k qui admet une matrice de contrôle de parité H à faible densité d'un poids des lignes w constant. Quand C est quasi-cyclique, il est appelé un $(n,k,w)-QC-LDPC$ code avec $n = n_0.r$.*

La matrice de contrôle de parité d'un code QC-LDPC est définie par la concaténation de blocs H_i tels que :

$$H = [H_0, \cdots, H_{n_0-1}]$$

où H_i (avec $i = 0, \cdots, n-1$)est une matrice circulante de taille $r \times r$ donnée par :

$$H_i = \begin{bmatrix} h_0 & h_1 & h_2 & \dots & h_{n_0-1} \\ h_{n_0-1} & h_0 & h_1 & \dots & h_{n_0-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & h_3 & \dots & h_0 \end{bmatrix} \tag{1.47}$$

La matrice génératrice d'un code QC-LDPC sous forme systématique est donnée par :

$$G = \left[\begin{array}{c|c} \mathrm{I} & \begin{matrix} \left(H_{n_0-1}^{-1}H_0\right)^T \\ \left(H_{n_0-1}^{-1}H_1\right)^T \\ \vdots \\ \left(H_{n_0-1}^{-1}H_{n_0-2}\right)^T \end{matrix} \end{array}\right] \tag{1.48}$$

1.4.8 QC/MDPC

Les codes MDPC (Moderate Density Parity Check) sont des codes à densité peu élevé par rapport aux codes LDPC. Nous reprondront en détail ce type de code et nous détaillerons l'algorithme de décodage bit flipping, utilisé comme décodeurs pour les codes MDPC en cryptographie, dans le chapitre 4.

1.5 Codes produits et Concaténés

1.5.1 Codes produits

Les performances d'un code augmentent avec sa longueur de bloc N, tandis que la complexité de décodage d'un code croit avec la dimension K . Une approche de ce problème est de réduire la complexité de décodage. Une façon d'aborder le problème, est de construire de bons codes qui présentent une complexité de décodage raisonnable, est d'utiliser des codes produit ou concaténés.

Les codes produits, inventés par P.Elias en 1954 [34], sont des codes correcteurs d'erreurs puissants. Ils sont construits par concaténation de deux ou plusieurs codes en blocs linéaires de faible capacité de correction.

1.5.1.1 Schéma de codage

Considérons deux codes en blocs élémentaires systématiques C_1 et C_2 ayant respectivement pour paramètres (n_1, k_1, d_1) et (n_2, k_2, d_2).avec :
- n_i : la longueur des mots du code C_i.
- k_i : la dimension du code C_i.
- d_i : la distance minimale du code C_i.

Un mot du code produit $Cp = C_1 \otimes C_2$ se présente sous forme d'une matrice M_p à n_1 lignes et n_2 colonnes dans laquelle :
- Les bits d'information défnissent une sous matrice I de k_1 lignes et de k_2 colonnes.
- Chacune de k_1 lignes de la matrice I est codée par le code C_2.
- les n_2 colonnes de la matrice ainsi formée sont codées en utilisant le code C_1

La figure 1.12 montre le principe de construction d'un code produit.

1.5.1.2 Paramètres de codes produits

Les paramètres du code produit $Cp = C_1 \otimes C_2$ sont déduits à partir de ceux des codes élémentaires, comme suit :
- La longueur : $n_p = n_1 \times n_2$.
- La dimension : $k_p = k_1 \times k_2$.
- La distance minimale : $d_p = d_1 \times d_2$.

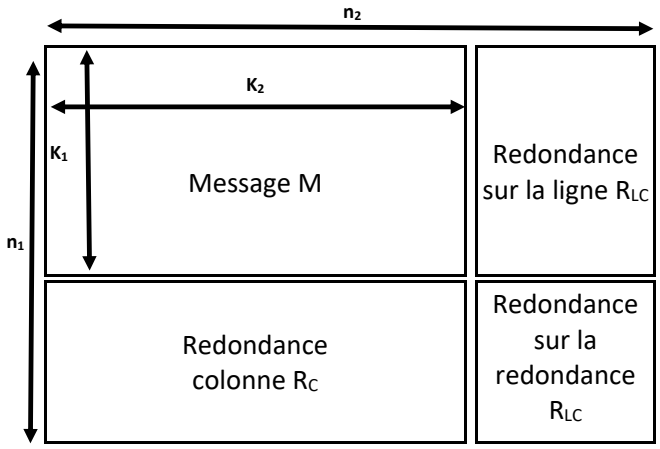


FIGURE 1.12 – Mot du code produit.

– Taux de code : $R_p = R_1 \times R_2 = \dfrac{k_1 \times k_2}{n_1 \times n_2}$.

1.5.1.3 Propriétés déductibles des codes produits

Proposition 1.4. *Soit G_1 la matrice génératrice de C_1 , et G_2 celle de C_1. $I(k_1; k_2)$ le mot d'information à transmettre. Alors la matrice génératrice du mot de code produit est : $M_p = G_1^T \times I \times G_2$.*

Proposition 1.5. *Si C_1 et C_2 sont linéaires alors le code produit $C_p = C_1 \otimes C_2$ est aussi linéaire.*

Proposition 1.6. *Si C_1 et C_2 sont systématiques, alors le code produit $C_p = C_1 \otimes C_2$ est aussi systématique.*

1.5.2 Les codes concaténés

En 1993 Berrou a montré que les performances des codes concaténés peuvent être amélioré avec une technique itérative utilisée pour leur décodage. Ce nouveau schéma de codage, appelé turbo-code, permet de s'approcher de la limite de Shannon. Les turbo-codes peuvent être des turbo-codes en bloc ou des turbo-codes convolutifs selon le type des codes concaténés. Ainsi, selon le type de la concaténation, parallèle ou série, on peut avoir des turbo-codes parallèles ou séries.
C'est vrais que les codes produit décrits ci-dessus sont des code concaténés (CC), mais ils présentent deux cas particuliers de la forme concaténation des codes qui est en général présentée dans deux formes à savoir la concaténation parallèle(CCP) et la concaténation série (CCS) ou hybride. Cette concaténation, sous sa forme générale nécessite un entrelaceur placé entre les codeurs objet de concaténation. Ce dernier joue un rôle important.

1.5.3 L'entrelaceur

Le rôle de 1'entrelacement et du désentrelacement est de décorréler les erreurs qui pourraient arriver dans un même temps en salve. Cela permet donc au récepteur de considérer les symboles codés subséquemment comme étant statistiquement indépendants. Une hypothèse de base est considérée dans l'utilisation de cette technique, le canal est à mémoire courte ou sans mémoire. L'entrelaceur doit aussi étaler les symboles sur une longueur de quelques fois la contrainte du codeur [35], d'où l'augmentation de la latence. Il existe plusieurs types d'entrelaceurs dont voici les plus usuels :

- 1'entrelaceur en bloc,
- 1'entrelaceur convolutionnel,
- l'entrelaceur aléatoire.
- l'entrelaceur Helical.

pour mieux de détails sur l'entrelaceur voir l'annexe.

1.5.4 Codes concaténés en série simples et généralisés

1.5.4.1 Codes concaténés en série simples

La concaténation série de codes est introduite dans [15], la figure 1.13 schématise cette concaténation dans sa forme simple, en effet l'information I est codée par un codeur dit externe $C(n_1, k_1, d_1)$ de taux R_1, les symboles de mot de code résultat de ce codage sont ensuite permuté entre eux par un entrelaceur, puis après intervient le deuxième codeur $C(n_2, k_2, d_2)$ de taux R_1 dit interne pour coder la sortie de l'entrelaceur, la sortie finale est un mot de code. Dans ce cas $k_2 = n_1$, et donc Le taux de ce type de concaténation est $R = \frac{k_1}{n_2}$.

1.5.4.2 Codes concaténés en série généralisés (GSCB)

la forme généralisée connue sous le nom de code concaténés en série généralisé (GSCB) peut varier selon la description. Farchane et al. [18] ont proposé une forme généralisée de la concaténation série, cette façon de construction leur a permis d'obtenir des codes de taux intéressant.

La figure 1.14 décrit cette construction, deux codeurs en bloc systématiques sont utilisés comme codeurs élémentaires. Le codeur interne est une troncature de codeur élémentaire usuel, le codeur externe est exactement le code sous sa forme habituelle, et entre les deux codes est placé un entrelaceur. Un bloc de $M.k$ bits parvenant à l'entrée du codeur et subdivisé en M sous blocs de k bits pour chacun. Chaque k bits est codé pour produire r bits de redondances. Les blocs codés sont permutés par l'entrelaceur avant d'arriver à l'entrée de deuxième codeur. Les $M.n$ bits sont subdivisé en sous bloc de n bits, chaque bloc est codé pour produire r bits de redondance.

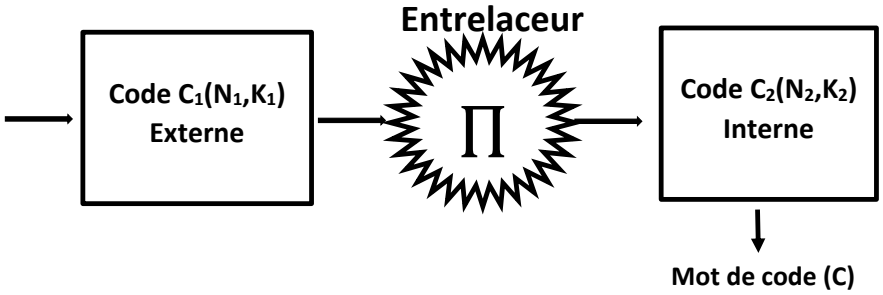


FIGURE 1.13 – La concaténation série simple

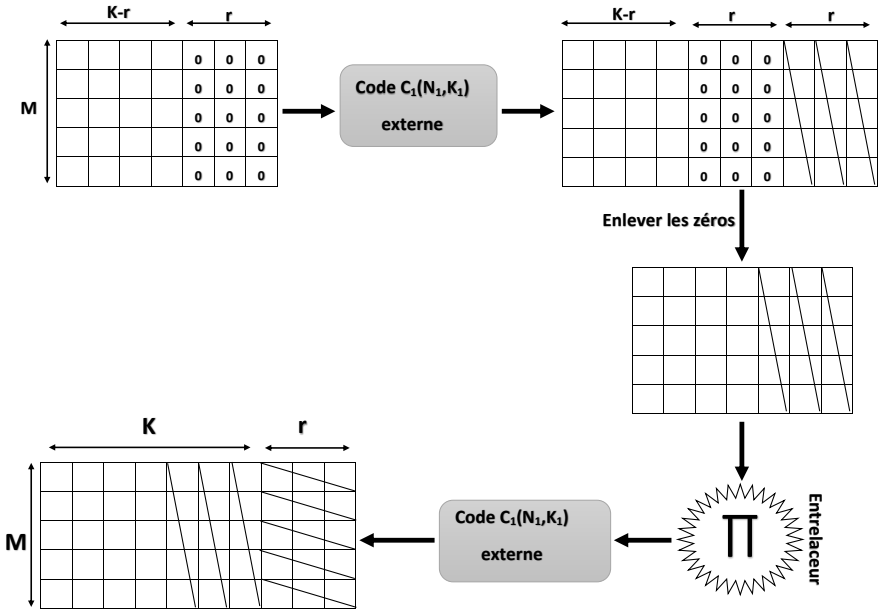


FIGURE 1.14 – La concaténation sérié généralisée :GSCB

1.5.5 Codes concaténés en parallèle simple et généralisée

En 1990, les codes concaténés en parallèle ont vu le jours avec les turbo code [15]. La figure 1.15 schématise cette famille des codes sous ça forme simple (i.e lorsque M = 1). L'information I de taille k est codé la première fois par le premier code C_1 en générant la redondance r_1, les k symbole de l'information I sont ensuite entrelacés puis codés par le deuxième codeur C_2 pour générer la redondance r_2, le bloque (Mux) est le multiplexeur

qui rassemble les trois parties à savoir le message (I), la redondance r_1 et la redondance r_2 pour produire le mot de codes résultant qui est une simple concaténation de I , r_1 et r_2. le taux de code résultant est $\frac{k_1}{r_1 + r_2 + k_1}$

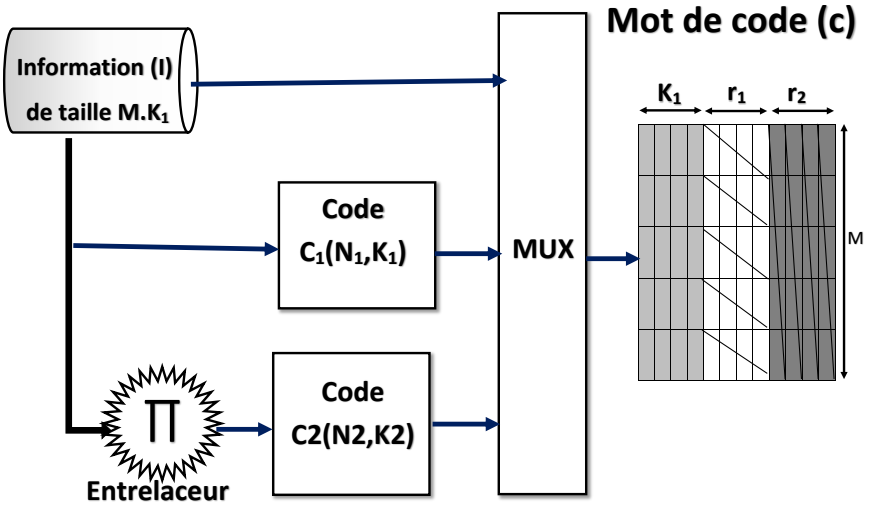


FIGURE 1.15 – La concaténation parallèle généralisée :GPCB

1.6 Classification des Algorithmes de décodage

1.6.1 Introduction

Les algorithmes de décodage (ou décodeurs)sont conçu pour répondre à la problématique de détection et correction des erreurs liés au canal de transmission. Ils se basent pratiquement sur la nature de codes (codeurs). L'efficacité du décodeur dépend du besoin pour lequel il est conçu. Le dilemme preformances/complexité pose toujours un souci pratiquement pour toutes les applications. La classiffication d'un décodeur se base sur le type de valeurs présentées à son entrée, et leur type produit au niveau de sa sortie.

1.6.2 Les types entrée et sortie d'un décodeur

Comme illustré dans la figure 1.16 ci-dessus, la sortie du canal est une valeur réelle. Après la démodulation, le décodeur peut recevoir dans son entrée des valeurs réelles, dans ce cas on dit que le décodeur est "Soft In". Il se peut aussi recevoir des valeurs binaires (0 ou 1) après seuillage, et on dit que le décodeur est "Hard In". En ce qui concerne la sortie du décodeur, il se peut qu'elle soit de type réelle, cette sortie est appelée " sortie Soft ",

on parle alors de décodeur "Soft Out", dans le cas où cette sortie est binaire (1 ou 0) à ce moment, on dit que la sortie est "Hard", et on qualifie le décodeur "Hard Out ". Il est à noter aussi que tous ces types d'entées et sorties sont liées à ce qu'on appelle les critères de décodage.

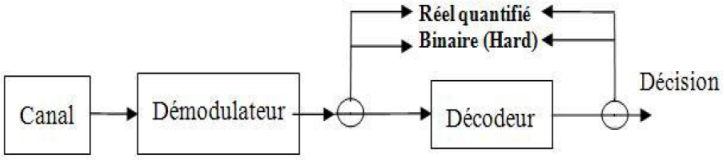


FIGURE 1.16 – Les différentes entrées et sorties possibles pour un décodeur

Dans ce paragraphe on va traiter plusieurs notions liées au décodage, puis citer les différentes familles de décodage en détaillant quelques algorithmes de décodage pour chaque famille.

1.6.3 Les critères du décodage

La conception des algorithmes du décodage diffère selon le type de la valeurs présentée au niveau de la sortie du démodulateur qui est soit binaires ou réelles, ainsi nous formulons quelques rappels sur les critères de décodage suivant que l'on considère le décodage ferme (dur ou hard) qui utilise des entrées binaires résultant du seuillage de la sortie du canal, ou le décodage pondéré (souple ou soft) qui utilise des entrées réelles. On fait également l'hypothèse que le décodage ferme est réalisé sur un canal sans mémoire (on peut ainsi se ramener à un modèle de canal binaire symétrique) et le décodage pondéré sur un canal gaussien.

1.6.4 Le décodage ferme(dur ou hard)

On suppose que la probabilité qu'une erreur se produise est égale à p. On considère alors un code binaire C de longueur n. On note $c = (c_0, c_1, ..., c_{n-1})$ un mot de code et $r = (r_0, r_1, ..., r_{n-1})$ l'observation qui lui est associée. Le décodeur cherche à maximiser la vraisemblance à posteriori donc la probabilité $P(c|r)$. Les mots de code étant équiprobables, cela revient par conséquent à maximiser $P(r|c)$ L'hypothèse d'un canal sans mémoire permet de dire que les observations r_i sont indépendantes et par conséquent :

$$\prod_{i=0}^{n} P(r_i|c_i)$$

De plus, selon qu'il y ait égalité ou non entre r_i et c_i, $P(r_i|c_i)$ prend la valeur $(1-p)$ ou p. Nous pouvons donc écrire :

$$P(r|c) = p^{d(r,c)} \times (1-p)^{n-d(r,c)} \tag{1.49}$$

Où $d(.,.)$ désigne la distance de Hamming.
p étant inférieure à $\frac{1}{2}$, maximiser $P(r|c)$ revient donc à minimiser la distance de Hamming entre c et r. Le décodage ferme consiste donc à rechercher le mot de code c de C qui minimise la distance de Hamming avec l'observation r.

1.6.5 Le décodage pondéré (souple ou soft)

Deux aspects sont à envisager. On peut considérer une minimisation de la probabilité d'erreur par mot (on maximise la vraisemblance à posteriori) ou une minimisation de la probabilité d'erreur par bit (on maximise le rapport de vraisemblance a posteriori).

1.6.5.1 Probabilité d'erreur par mot

On note $c = (c_0, c_1, ..., c_{n-1})$ un mot de longueur n du code C et $r = (r_0, r_1, ..., r_{n-1}))$ l'observation qui lui est associée. On cherche à maximiser $P(c|r)$. Les mots de codes étant équiprobables, cela équivaut à maximiser $P(r|c)$. Le canal étant gaussien, les observations sont indépendantes et nous pouvons écrire :

$$P(r|c) = \prod_{i=0}^{n} P\left(r_i|c_i\right) = \left(\frac{1}{2\Pi\sigma^2}\right)^{\frac{n}{2}} \exp\left(-\frac{d^2(r,c)}{2\sigma^2}\right) \tag{1.50}$$

Où $d(r,c)$ est la distance euclidienne entre le mot c et l'observation r.
Maximiser $P(r|c))$ revient donc à minimiser la distance euclidienne entre l'observation r et le mot c de C . Un tel décodeur fournit donc le mot de code le plus probable.
Probabilité d'erreur par bit
Pour chacune des composantes c_i du mot de code, on détermine le logarithme du rapport de vraisemblance défini par :

$$L(c_i) = \frac{P(c_i = 1|r)}{P(c_i = 0|r)} \tag{1.51}$$

Si L(ci) est supérieure à 1, on décide que le bit vaut 1 et 0 dans le cas contraire.
Ce décodeur traite donc les éléments bits à bits ce qui implique que le mot obtenu après décodage n'appartient pas nécessairement au code. Donc selon l'observation de l'entrée et de la sortie d'un décodeur, on peut distinguer trois familles de décodage à savoir :

- Le décodage à entrées et à sorties binaires (HIHO).
- Le décodage à entrées pondérées et à sorties binaires (SIHO).
- Le décodage à entrées et à sorties pondérées (SISO).

Indépendamment de l'entrée et de la sortie des décodeurs, nous pouvons aussi les catégoriser comme décodeurs itératifs ou non itératifs.

1.6.6 Le décodage itératif

Le décodage itératif est un bon compromis entre performance et complexité, qui a comme idée de base, de partager le problème de décodage en des étapes (itérations) successives telles que chaque étape utilise la sortie de l'étape précédente pour fournir sa propre sortie. La complexité et les ressources matérielles demandées à chaque étape doivent être minimales et rester au dessous d'un certain seuil qui est fonction de l'application, en même temps la performance assurée doit être supérieure à chaque fois qu'on augmente le nombre d'étapes (itérations) jusqu'à un nombre maximal d'itérations, au delà de ce seuil le gain de décodage est sans importance. Ce nombre est obtenu expérimentalement.
De manière générale, le décodage itératif consiste à décoder un code par étapes successives, à l'aide de plusieurs décodeurs à faible coût au lieu de décoder le code avec un seul décodeur complexe. Récemment, il a été démontré [15] que les systèmes de décodage itératif peuvent fonctionner à des rendements très proches de la limite de Shannon, imposée par le théorème du codage sur canaux bruités, avec toutefois une complexité raisonnable.

1.6.6.1 L'information extrinsèque :

Comme son nom l'indique l'information extrinsèque w_j est extérieur à l'objet étudier, n'appartient pas à son essence. Elle dépend de tous les symboles du mot de code, excepté le symbole en position j pour lequel est évaluée la décision pondérée. Puisque tous les symboles d'un mot de code sont corrélés par le codage, une combinaison linéaire de ces symboles autres que d_j apporte une information sur d_j . L'information extrinsèque est interprétée comme un terme de correction pour l'entrée R.

1.6.7 Le décodage à entrées et à sorties binaires (HIHO)

Dans le cas où les données présentées à l'entrée du décodeur sont à valeurs binaires, le décodage revient à trouver le mot le plus proche en terme de distance de Hamming de l'observation. Ce type de décodage est appelé décodage binaire ou décodage "hard". Cette approche de décodage exige l'existence d'un circuit de décision, permettant le seuillage de l'information réelle avant l'opération de décodage selon le principe suivant :
Soit un mot de code x transmit à travers un canal AWGN. La séquence reçue est $y \in \mathbb{R}^n$. Le mot de code estimé à la réception est noté $x^i \in X^n$ et $c^{'} \in (GF^2)^n$. La décision binaire pour un réel r donné est :

$$H(r) = \begin{cases} 1, & r >= 0 \\ 0, & r < 0 \end{cases} \tag{1.52}$$

Après cette opération, le décodage proprement dit est effectué par un décodeur algébrique. Il est bien évident que la sortie du décodeur est binaire, puisque l'entrée est binaire.
Plusieurs décodeur de ce type existent dans la littérature, chacun est conçu pour une famille de codes, le plus populaire c'est **l'algorithme de Berlekamp-Massey**

1.6.8 Le décodage à entrées Pondérées et sorties binaires (SIHO)

Le principe d'un décodage SIHO est basé sur l'exploitation de l'information réelle à l'entrée du décodeur pour fiabiliser davantage la décision binaire de la sortie, on va voir par la suite que dans [14],[36], l'entré réelle (soft) du canal est exploitée pour calculer la distance euclidienne sur laquelle ces décodeurs basent leur décision, et que dans [37] cette entrée est utilisée pour calculer la probabilité d'erreur et dans la règle de décodage.

1.6.8.1 L'algorithme de décodage de Chase

L'algorithme de Chase [14] est une méthode de décodage des codes en blocs, qui utilise une liste de modèles des erreurs les plus probables. Ces modèles d'erreur sont sélectionnés en se basant sur la puissance (fiabilité) des symboles reçus. Chaque modèle d'erreur est ajouté à la décision binaire " Hard-Decision " du mot reçu, le mot est ensuite décodé en utilisant un décodeur à décision binaire. Chaque mot décodé (séquence binaire) est classé en calculant sa métrique en terme de distance euclidienne par rapport à la séquence reçue à l'entrée du décodeur (séquence réelle). Le mot de code avec la meilleure métrique est sélectionné comme le mot le plus vraisemblable au mot reçu. Dans son travail [14] qui date de 1972, David Chase a montré que pour les grandes valeurs de rapport signal sur bruit, son algorithme peut atteindre une capacité de correction égale à $d-1$.

***Les variantes de l'algorithme de Chase**

Si l'on considère un canal gaussien, cela revient à minimiser la distance euclidienne entre l'observation et un mot de code. Soient r l'observation réelle, y l'observation binaire $y \in F_2^n$ et c un mot de code quelconque d'un code binaire $C(n,k)$. La distance euclidienne entre r et c est alors donnée par :

$$\begin{aligned} d_E^2(r,c) &= \sum_{i=0}^{n-1} (r_i - c_i)^2 = \|r\|^2 + n - 2\sum_{i=0}^{n-1} |r_i|\, y_i \cdot c_i \\ &= \|r\|^2 + n - 2\sum_{i=0}^{n-1} |r_i| + 4 \sum_{i=0, y_i \neq c_i}^{n-1} |r_i| \end{aligned}$$

En mappant l'ensemble $[-1,1]$; sur l'ensemble $(0,1)$, l'égalité ci-dessus devient :

$$d_E^2(r,c) = \|r\|^2 + n - 2\sum_{i=0}^{n-1} |r_i| + 4\sum_{i=0}^{n-1} (|r_i|\, y_i \oplus c_i)$$

Minimiser cette distance euclidienne revient donc à minimiser le terme $\sum_{i=0}^{n-1} (|r_i|\, y_i \oplus c_i)$ par rapport à l'ensemble des mots du code. Le problème majeur d'une telle méthode étant le nombre élevé de mots de Code. Chase a proposé de restreindre la recherche du mot de code le plus vraisemblable à un simple sous ensemble de mots de code. Dans un premier temps, l'observation réelle r est seuillée pour fournir l'observation Binaire y. On détermine le mot de code c^0 le plus proche de y à l'aide d'un décodeur algébrique corrigeant au plus $t = \frac{d-1}{2}$ erreurs. c^0 est l'unique mot de code situé à l'intérieur de la boule de centre y et de rayon t.

La méthode de Chase consiste à étendre la recherche à l'ensemble des mots de code appartenant à la sphère de centre y et de rayon $2t = d-1$ le mot de code émis s'y trouvant

forcement si le nombre d'erreurs commises ne dépasse pas la capacité correctrice du code. Cela revient à augmenter le pouvoir de correction du décodeur. On note t^i les séquences de longueur n et de poids $\frac{d}{2}$ utilisées pour " atteindre " les mots de codes contenus dans la sphère de centre y et de rayon $\frac{d}{2}$. Pour chacun des t^i on calcule la séquence $y^i = t^i + y$ à laquelle on associe un mot de code c^i qui est déterminé à l'aide du décodeur algébrique. Parmi tous les mots de codes ainsi construits, on choisit finalement celui qui vérifie :

$$min_j \sum_{i=0}^{n-1} \left(|r_i|\, y_i \oplus c_i^j \right)$$

Chase a décliné son algorithme en 3 variantes distinctes permettant ainsi de diminuer le nombre de séquences de test nécessaires, réduisant donc la complexité, mais dégradant les performances du décodage :

- **Variante 1** : Il s'agit de la version originale de l'algorithme explicitée ci-dessus. Il implique de former toutes les séquences test de poids $\left[\frac{d}{2}\right]$ pour obtenir tous les mots de code contenus dans la sphère $S(y, d-1)$. Le coût de calcul est malheureusement inacceptable lorsque la distance minimale (ou la longueur) est importante car il nécessite la formation de $C_n^{\frac{d}{2}}$ séquences de test.
- **Variante 2** :On considère les $\left[\frac{d}{2}\right]$ indices correspondant aux valeurs les moins vraisemblables des $|r_i|$. On ne forme que les motifs d'erreur dont les 1 sont positionnés en ces indices. Le nombre de séquences de test est alors $2^{\frac{d}{2}}$

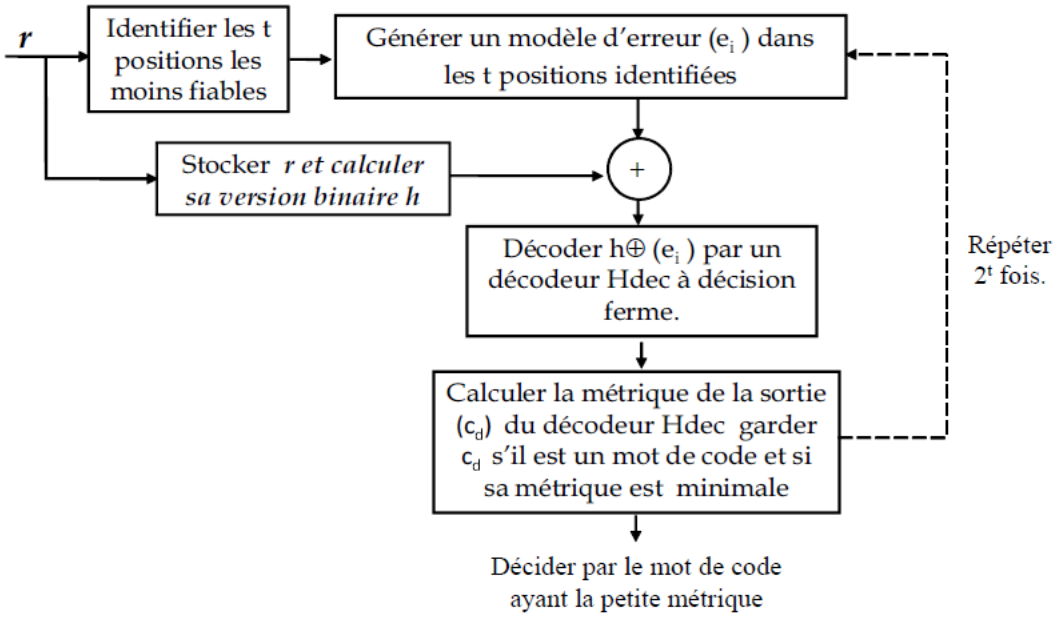


FIGURE 1.17 – Algorithme de Chase variante 2

- **Variante 3** : Dans ce cas, le nombre de séquences de test est $\left[\frac{d}{2}+1\right]$ où chacune d'entre elles possèdent exactement i positions contenant 1. Ces i positions correspondent aux i valeurs $|r_i|$ de vraisemblances les plus faibles pour $i \in 0, 1, 3, ..., d-1$ si d est pair et $i \in 0, 2, 4, ..., d-1$ dans le cas contraire.

La figure 1.17 présente la variante 2 de l'algorithme de Chase,qui est la plus utilisée, pour son compromis Performance/complexité, elle est très utilisée dans des algorithme itératifs, cette problématiques est traitée dans plusieurs travaux par exemple celui Ait Sab et al.[38] qui ont réduit le nombre de séquence de tests à 18 pour les code BCH, aussi Gazelle et Snyders[39] ont donné une méthode pour générer les séquence de test, et également Kabat et al.[40] ont décodé les codes RS de longues tailles par l'algorithme de l'OSD.

1.6.9 Le décodage à entrées et sorties Pondérées (SISO)

Ces types de décodeurs peuvent être divisés en deux familles. La première contient des algorithmes à entrés et sorties pondérées appliqués pour les codes simples, nous appelons cette famille décodeur SISO simple. La deuxième famille est appelée algorithme de décodage turbo, cette dernière est appliquée pour les codes concaténés. La clef de ce type de décodeur est le calcul d'une information pondérée (souple ou soft) qui est considérée comme information corrective appelée **l'information extrinsèque**.
Plusieurs exemples pour les deux familles ont été présentés dans la littérature, on trouve l'algorithme de Lucas et al [41], l'algorithme de Belkasmi et al. [42] pour les codes simple. Pour la famille turbo on peut citer les algorithmes de Chase-Pyndiah [43], Chase/Soliemani, L'IDA de Lucas [41] pour les code concaténés en parallèle et les codes produits, et le turbo décodeur de de Massey/Belkasmi [44, 45]

1.6.9.1 Le décodeur Itératif de Chase/Pyndiah

L'algorithme de Chase fournit un mot de code décidé $D = \{d_0, d_1, ..., d_n\}$ et une liste de mots de codes du code $C(n, k)$ les plus vraisemblables à l'entrée pondérée. Le décodeur affecte une pondération à chaque composante d_j du mot de code décidé D pour mesurer la fiabilité de chaque décision binaire. Cette fiabilité est évaluée par le Logarithme du rapport de vraisemblance associé à la décision $d_j \in +1, -1$ en sortie du décodeur, défini par :

$$\Lambda_{\mathrm{d_j}} = \ln \frac{\Pr\left(\mathrm{d_j} = +1/\mathrm{R}\right)}{\Pr\left(\mathrm{d_j} = -1/\mathrm{R}\right)} \tag{1.53}$$

De même que précédemment, le signe de $\Lambda_{\mathrm{d_j}}$ donne la décision d_j , et la valeur absolue de $\Lambda_{\mathrm{d_j}}$ est une mesure de la fiabilité de cette décision. Le numérateur du rapport de vraisemblance est égale à :

$$\Pr\left[d_j = +1/R\right] = \sum_{C^i \in S^{+j}} \left[\Pr C = C^i/R\right] \tag{1.54}$$

Où C_i est un mot de code dont la composante j est égale à $+1$: S^{+j} est l'ensemble de tous les mots de code dont la composante j est égale à $+1$. De même, le dénumérateur du rapport de vraisemblance est égale à :

$$\Pr\left(d_j = -1/R\right) = \sum_{C^i} \in S^{-j} \Pr\left(\mathrm{C} = \mathrm{C^i/R}\right) \tag{1.55}$$

S^{-j} représente l'ensemble de tous les mots de code dont la composante j est égale à -1 : $C_j^i = -1$ En utilisant la formule de Bayes, on modifie l'écriture de $Pr[C = C^i/R]$:

$$\Pr\left[C = C^i/R\right] = \frac{p\left(R/C^i\right)\Pr\left(C^i\right)}{p(R)} \tag{1.56}$$

En tenant compte du fait que tous les mots de code C^i sont équiprobables et que la densité de probabilité $p(R)$ de l'observation disparaît dans le rapport de vraisemblance, le seul terme utile est la densité de probabilité conditionnelle $p\left(R/C^i\right)$. Sur le canal gaussien, perturbé par le bruit blanc de puissance moyenne σ^2 , la densité de probabilité conditionnelle $p\left(R/C^i\right)$ suit une loi gaussienne centrée sur C^i :

$$p\left(R/\mathrm{C^i}\right) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^{\mathrm{n}} \exp\left(-\frac{1}{2\sigma^2}\sum_{1=1}^{\mathrm{n}}\left(\mathrm{r}_1 - c_1^{\mathrm{i}}\right)^2\right) \tag{1.57}$$

Ou encore

$$p\left(R/C^i\right) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^{\mathrm{n}} \exp\left(-\frac{1}{2\sigma^2}\left\|\mathrm{R} - \mathrm{C^i}\right\|^2\right) \tag{1.58}$$

Rappelons que le mot de code décidé est celui pour lequel la distance euclidienne par rapport à l'observation R est minimale, c'est-à-dire que : $D = C^i$ si $\|R - D\|^2$ est minimale. En tenant compte des expressions 1.53) et 1.58), le Logarithme du Rapport de Vraisemblance (LRV) en anglais (LLR) (log-likelihood ratio) en sortie du décodeur est égale à :

$$\Lambda_{d_j} = \ln \frac{\sum_{C^i \in S^{+j}} \exp\left(-\frac{1}{2\sigma^2}\left\|R - C^i\right\|^2\right)}{\sum_{C^i \in S^{-j}} \exp\left(-\frac{1}{2\sigma^2}\left\|R - C^i\right\|^2\right)} \tag{1.59}$$

Si le rapport signal sur bruit est fort, la distance du vecteur d'observation R aux différents mots de code C^i est assez grande pour que l'on puisse ne retenir que le mot de code le plus proche C^{I+} au numérateur et le mot de code le plus proche C^{I-} au dénominateur, ces deux mots de code ayant respectivement en position j un symbole égal à $+1$ et -1 : $c_j^{I+} = +1$ et $c_j^{I-} = -1$.
Le LRV du symbole décidé d_j est approximé par :

$$\Lambda_{d_j} = \ln \frac{\exp\left(-\frac{1}{2\sigma^2}\left\|\mathrm{R} - \mathrm{C^{I+}}\right\|^2\right)}{\exp\left(-\frac{1}{2\sigma^2}\left\|\mathrm{R} - \mathrm{C^{I-}}\right\|^2\right)} \tag{1.60}$$

Ce qui, après simplification, s'écrit sous forme :

$$\Lambda_{\mathrm{d_j}} = \frac{2}{\sigma^2}\mathrm{r_j} + \frac{2}{\sigma^2}\sum_{\substack{\mathrm{l}\neq\mathrm{j} \\ \mathrm{c_1^{I+}}=-\mathrm{c_1^{I-}}}} \mathrm{r_1 c_1^{I+}} \tag{1.61}$$

En normalisant par $\frac{\sigma^2}{2}$, la fiabilité de la décision binaire en sortie du décodeur en blocs est évaluée par :

$$r_{\mathrm{d_j}} = \frac{\sigma^2}{2}\Lambda_{\mathrm{d_j}} = \mathrm{r_j} + \sum_{\substack{1\neq \mathrm{j} \\ \mathrm{c_1^{I+}}=-\mathrm{c_1^{1-}}}} \mathrm{r_1 c_1^{I+}} \tag{1.62}$$

Notons :

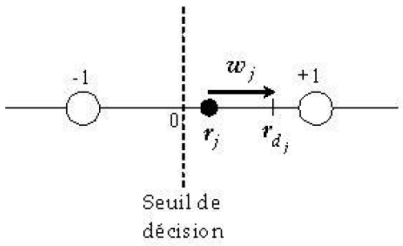
$$w_j = \sum_{\substack{l\neq j \\ c_l^{I+}=-c_l^{I-}}} r_l c_l^{I+} \tag{1.63}$$

Ce terme w_j , indépendant du symbole en position j, est une information extrinsèque apportée par le décodeur, elle joue le même rôle que pour les turbo codes convolutionnels. Le Logarithme du Rapport de Vraisemblance normalisé de la décision d_j s'écrit finalement :

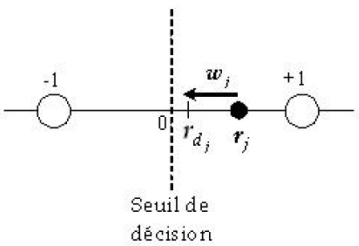
$$r_{d_j} = r_j + w_j \tag{1.64}$$

Le LRV de la décision est la somme du LRV à l'entrée du décodeur et d'une information extrinsèque apportée par le décodeur. L'information extrinsèque est interprétée comme un terme de correction pour l'entrée R, qui dépend des deux mots de code modulés les plus proches de R. L'information extrinsèque w_j peut s'interpréter, d'après la relation (3.25), comme la différence entre l'information disponible après le décodage et l'information connue avant le décodage. L'information extrinsèque w_j mesure donc l'apport d'information inédite fournie par le décodeur. L'information extrinsèque w_j a donc tendance à modifier l'observation r_j de façon à :

- soit confirmer la décision d_j , si w_j et r_j sont de même signe Figure 1.18a.
- soit à inverser la décision d_j , si w_j et r_j sont de signes opposés Figure 1.18b.



(a) Confirmation de la décision



(b) infirmation de la décision

1.6.9.2 L'évaluation de l'information extrinsèque

Le décodeur itératif ne sera efficace que si l'information extrinsèque est correctement utilisée à l'entrée du décodeur suivant. En effet, ce n'est pas la décision pondérée rdj qui est le plus efficacement exploitée par un décodeur itératif, mais l'information extrinsèque que l'on extrait de la décision pondérée.
Si on revient sur l'expression de r_{d_j} , on constate qu'il peut s'écrire sous la forme :

$$r_{d_j} = \frac{\sigma^2}{2}\Lambda_{d_j} = \frac{1}{4}\left[\left\|R - C^{I-}\right\|^2 - \left\|R - C^{I+}\right\|^2\right] \tag{1.65}$$

Si D est égale au mot de code C^{I^+}, dj est égale +1. Le mot de code C^{I^-} est le mot de code concurrent le plus proche qui à un symbole opposé à d_j en position j,et inversement.

En définissant les métriques du mot décidé D et du mot concurrent C^c respectivement par :

$$\begin{aligned} \mathrm{M^d} &= \|\mathrm{R} - \mathrm{D}\|^2 \\ \mathrm{M^c} &= \|\mathrm{R} - \mathrm{C^c}\|^2 \end{aligned} \tag{1.66}$$

Le LRV normalisé est donné par :

$$r_{d_j} = \left[\frac{M^c - M^d}{4}\right] d_j \tag{1.67}$$

Compte tenu de l'expression (II.44) et (II.46) de la dernière expression du text LRV normalisé, l'information extrinsèque est évaluée en sortie du décodeur par :

$$w_j = \left[\frac{M^c - M^d}{4}\right] d_j - r_j \tag{1.68}$$

Cette expression de w_j suppose que l'un des mots concurrent C^c ait en position j un signe opposé à d_j . Un tel mot de code existe bien dans l'ensemble des 2k mots de code. Mais, en sortie du décodeur, on ne dispose que de$2^{d/2}$ mots de code proposés par l'algorithme de Chase. La recherche du mot de code concurrent C^c se fait dans l'ensemble de ces $2^{d/2}$ mots les plus vraisemblables. Lorsqu'il existe dans cet ensemble, un mot de code concurrent tel que $C^c = -d_j$, l'information extrinsèque est calculée à partir de (II.47). Par contre, lorsque tous les mots de code potentiels proposés par le décodeur élémentaire ont un symbole d_j égal à la décision c_j , il est impossible d'évaluer le LRV défini par la formule (II.40). Dans ce cas, l'information extrinsèque est utilisée pour confirmer la décision d_j . Lorsqu'il y a pas de concurrent pour le symbole d_j , w_j sera donc évaluée par :

$$w_j = \beta \times d_j \tag{1.69}$$

Où β est un coefficient positif choisi expérimentalement.

1.6.9.3 Le turbo décodage de Pyndiah

Après un décodage de toutes les lignes et de toutes les colonnes du code produit, certaines erreurs résiduelles peuvent être corrigées en itérant le processus de décodage. L'information extrinsèque disponible en sortie d'un décodeur à l'itération $(p-1)$ est utilisée pour modifier le vecteur à l'entrée du décodeur suivant, en fonction de la décision qui vient d'être prise. Le vecteur $R(p)$ à l'entrée du p^{ime} décodeur est donné par :

$$R(p) = R(1) + \alpha(p)W(p-1) \tag{1.70}$$

Où $R(1)$ est le vecteur des LRV à l'entrée du premier décodeur, c'est-à-dire : $R(1) = R$ le vecteur reçu.
Le coefficient $\alpha(p)$ est introduit pour tenir en compte du fait que l'estimation de l'information extrinsèque $W(p-1)$ est peu fiable lors des premières itérations. Sa fiabilité augmente au fil des itérations, et le coefficient $\alpha(p)$ augmente avec p.
Au p^{ime} décodage, l'information extrinsèque $w_j(p)$ est évaluée par l'une des deux expressions suivantes, selon qu'un concurrent ayant un symbole c_j de signe opposé à d_j est trouvé

ou non, si le concurrent existe :

$$w_j(p) = \left[\frac{M^c - M^d}{4}\right] d_j - r_j(p) \tag{1.71}$$

Où r_j est la j^{ime} composante du vecteur d'entrée défini en (II.49). Si aucun mot de code en sortie du décodeur, n'a pas un symbole c_j opposé à d_j , l'information extrinsèque est calculée par :
Les valeurs de $\alpha(p)$ et $\beta(p)$ augmentent avec le nombre des itérations p, et sont déterminées expérimentalement. Le décodage itératif d'un code produit selon l'algorithme de Chase-

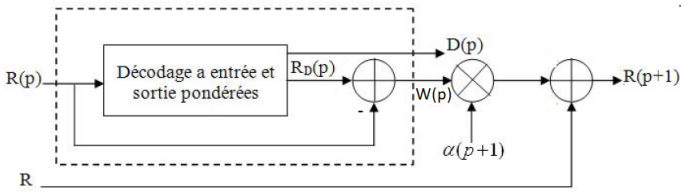


FIGURE 1.18 – Le turbo décodage de Pyndiah

Pyndiah, est effectué en décodant alternativement les lignes puis les colonnes de la matrice reçue selon l'algorithme de Chase. Le décodeur fournit une décision pondérée dont on extrait l'information extrinsèque selon l'algorithme de Pyndiah. L'information extrinsèque est ensuite pondérée par le facteur α. Le paramètre β est utilisé lorsque la décision de Chase sur un bit n'a pas de concurrent. Ces deux paramètres évoluent en fonction des itérations, ils sont également très importants et ils ont une influence considérable sur les performances. Ils sont fixés arbitrairement, expérimentalement et à priori pour chaque itération. Cependant il est difficile de les justifier.

1.7 Conclusion

Dans ce chapitre, premièrement, nous avons introduit les éléments de base d'un système de communication numérique. Deuxièmement Nous avons rappelé les notions de base et les définitions relatives aux codes correcteurs d'erreur. Nous avons également évoqué plusieurs types des codes que nous avons utilisés à savoir les codes BCH, RS, Goppa, QC-LDPC/MDPC et les différentes formes de concaténation de codes.
Nous avons étudié les trois familles de décodeurs à savoir HIHO, SIHO et SISO en donnant un exemple pour chaque famille . Pour les décodeurs de type HIHO, on a donné comme exemple le décodeur algébriques de Berlekam-Massay, pour le cas du décodeurs de type SIHO on a entamé l'exemple de l'algorithme de Chase. Nous avons également vu que le turbo décodage et le décodage itératif nécessitent le calcul de l'information extrinsèque à chaque demi-itération pour la communiquer au décodeur selon un schéma de décodage. Il y a des types qui pondèrent l'information extrinsèque à la fin de chaque demi-itération, comme le cas de l'algorithme de Pyndiah. Le chapitre suivant sera consacré au décodage itératif des codes concaténés en série et parallèle basé sur les codes RS.

Chapitre 2

Turbo décodage des codes GSCB et GPCB : Cas des codes RS

2.1 Introduction

En 1993, Berrou [15] a construit les turbo codes convolutionels. Ces codes consistent à concaténer deux codes convolutionels récursifs séparés par un entrelaceur (non-uniform interleaver). Ils ont donné des performances exceptionnelles. Une année plus tard, Pyndiah [43] à proposé un nouveau algorithme de décodage itératif basé sur la version SISO du décodeur de Chase [14]. Les résultats obtenus sont similaires à ceux des turbo codes convolutionels.
Le concept de turbo décodage peut aussi être appliqué aux autres codes concaténés [46][47][17]. Dans ce chapitre,les codes concaténés généralisés, en serie et en parallèle sont similaires aux turbo-codes convolutionnels en terme de décodage. Ces concaténations de codes permettent d'obtenir des codes de longue taille et qui sont puissants, tout en maintenant un décodage simple. Les applications dont la sûreté est critique bénéficient de ces codes puissants.
Dans ce chapitre, nous avons étendu cette nouvelle construction des codes GSCB[19] et GPCB[48] pour les codes RS (Reed Solomon). Ensuite, nous avons étudié l'effet de plusieurs paramètres sur les performances des codes GSCB et GPCB à base des codes RS : le code élémentaire, le nombre d'itérations, la taille et la structure de l'entrelaceur en utilisant la simulation.
Finalement, nous avons comparé les performances des codes GSCB et GPCB.

2.2 Les codes en bloc concaténés en série généralisés

En 1996, Benedetto [49] a évalué théoriquement les performances des codes concaténés, en particulier les codes SCB. Néanmoins, Les codes SCB à base des codes en blocs, ont un faible taux de codage ce qui est indésirable pour les systèmes de communication numériques. Donc, pour surmonter ce problème,les auteurs de [18] ont présenté une nouvelle construction permettant d'obtenir des codes concaténés avec un taux de codage très élevés. Cette construction est basée sur des codes raccourcis. Elle nous donne le choix entre plusieurs taux de codage en utilisant le même code pour les codes élémentaires. Dans son travail, Benedetto a évalué les performances en utilisant union-bounding. Notre travail est basé sur la simulation pour évaluer les performances de ces codes.

2.2.1 Construction des codes GSCB

2.2.1.1 Construction classique

La structure du codeur SCB est montrée sur la figure 2.1. Deux codeurs en bloc systématiques sont utilisés comme codeurs élémentaires avec un entrelaceur Π placé avant le second codeur élémentaire. Un bloc de $M.k_1$ symboles parvenant à l'entrée du décodeur est subdivisé en M sous blocs de k_1 symboles pour chacun, où M représente le nombre de multi-blocs. Chaque k_1 symboles est encodé pour produire n_1 symboles. Les blocs encodés sont brouillés par un entrelaceur avant d'arriver à l'entrée du deuxième codeur. Les $M.n_1$ symboles sont subdivisés en sous blocs de n_1 symboles. Chaque n_1 symboles est encodé pour produire n_2 symboles.
Le code SCB systématique est basé sur deux codes en blocs systématiques. $C_1(n_1, k_1)$

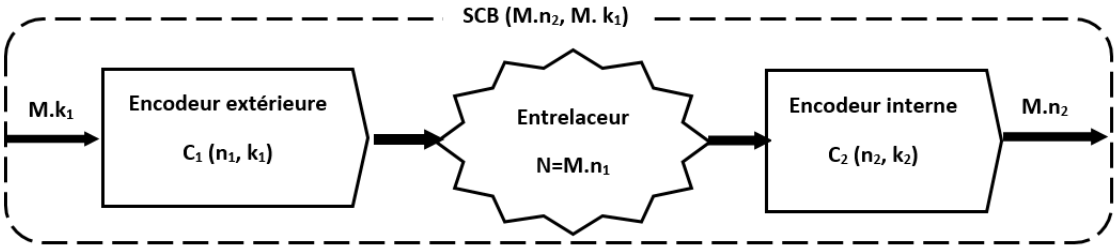


FIGURE 2.1 – Codeur classique des codes SCB

et $C_2(n_2, k_2)$. La longueur de l'entrelaceur est donnée par la longueur du premier codeur $N = M.n_1$. Considérons le schéma de codage de la figure 2.1 comme un seul codeur SCB. La longueur du mot d'information à coder par le codeur SCB est $M.k_1$. Le premier codeur produit $p_1 = M(n_1 - k_1)$ symboles de parité. Le second codeur produit $p_2 = M.(n_2 - n_1)$ symboles de parité. Alors le nombre total de symboles de parité généré par le codeur SCB est $p = p_1 + p_2 = M.(n_2 - k_1)$. La longueur du mot de code SCB est donné par : $L = M.k_1 + P = M.n_2$. Par conséquent, le taux de codage du code SCB peut être calculé par $R = \frac{M.k_1}{L} = \frac{k_1}{n_2}$. Ceci veut dire que le taux de codage du code SCB est indépendant de la longueur de l'entrelaceur N.
Par nature, les codes concaténés en série ont un taux de codage faible. Néanmoins, les systèmes de communication numériques ont besoin d'un taux de codage élevé. Pour atteindre cet objectif, une nouvelle construction basée sur les codes raccourcis à été proposé dans [18]. Cette construction est présentée dans la sous-section suivantes puis étendu pour être utilisé pour les codes RS.

2.2.1.2 Nouvelle construction

La concaténation des codes SCB en utilisant des codes en blocs rencontre des contraintes, qui proviennent du fait qu'on ne peut pas avoir deux codes en blocs dont la longueur du premier égale à la dimension du deuxième code. Pour résoudre ce problème, les auteurs

de [18] ont proposé une construction basé sur les codes BCH raccourcis. Dans ce sens, nous étendons ce travail aux codes en blocs RS.
La concaténation en serie généralisée des codes RS consiste à prendre deux codes en bloc linéaires $C_1(n_1, k_1)$ et $C_2(n_2, k_2)$ de longueur n_1 (respectivement n_2) et de dimension k_1 (respectivement k_2). Nous allons, maintenant raccourcir les C_1 et C_2 par s_1 symboles (respectivement par s_2 symboles). En d'autre terme, pour avoir la longueur du code tronqué $\bar{C}_1$ coïncide avec la dimension du code $\bar{C}_2$. Donc, le code extérieur $\bar{C}_1(n_1 - s_1, k_1 - s_1)$ est obtenu en tronquant C_1 par s_1 symboles, et le code intérieur $\bar{C}_2(n_2 - s_2, k_2 - s_2)$ est obtenu en tronquant C_2 par s_2 symboles, pour pouvoir finalement satisfaire la contrainte :

$$k_2 - s_2 = n_1 - s_1$$

Alors, $s_1 - s_2 = n_1 - k_2$. Ces deux dernier décodeurs sont reliés par un entrelaceur de taille $M.(n_1 - s_1)$, où M est un entier positive appelé le multi-blocs. Le code construit résultant est $GSCB(n_2 - s_2, k_1 - s_1)$ son taux de codage est $R = \frac{k_1 - s_1}{n_2 - s_2}$.
Plusieurs cas partiuliers peuvent être dérivés du cas générale :

- Le premier cas consiste à prendre $c_1 = c_2$, d'où $\bar{C}_1$ et $\bar{C}_2$ deviennent $\bar{C}_1(n_1 - s_1, k_1 - s_1)$ et $\bar{C}_2(n_1 - s_2, k_1 - s_2)$. Pour le cas du code $RS(127, 123)$, $s_1 = 6$ et $s_2 = 2$ on aura $\bar{C}_1 = RS(121, 117)$ et $\bar{C}_1 = RS(125, 121)$. Le code global obtenu est $GSCB(125, 117)$ de taux $R = 0.936$.
- Le deuxième cas particulier peut être obtenu lorsque $s_1 = s_2 = s$. Ce cas implique que $n_1 = k_2$. Donc, le code global devient $GSCB(n_2 - s, k_1 - s)$ de taux $R_s = \frac{k_1 - s}{n_2 - s}$. En particulier si $s = n_1 - k_1$, le code devient $GSCB(n_2 + k_1 - n_1, 2k_1 - n_1)$ de taux $R = \frac{2k_1 - n_1}{n_2 + k_1 - n_1}$.
- Le troisième cas particulier est plus intéressant avec $s_1 = n_2 - k_2$ et $s_2 = 0$. La dernière relation implique que $s_1 = n_1 - k_2$. Ceci signifie que $n_2 - k_2 = n_1 - k_2$. D'où les deux codes ont la même langueur ($n_1 = n_2$). De plus, on a $s_1 = n_1 - k_1$ ceci signifie que les deux codes ont la même dimension ($k_1 = k_2$). Finalement, le code concaténé en série devient $GSCB(n_1, 2k_1 - n_1)$ et de taux de codage $R = \frac{2k_1 - n_1}{n_1}$. La taille de l'entrelaceur est $M.k_1$.

La figure 2.2 présente le schéma synoptique de la construction série généralisée des codes en blocs. Cette construction donne un eventail de choix de code GSCB, dans ce sens, le tableau 2.1 présente quelques exemples de code construit GSCB à base des codes RS.

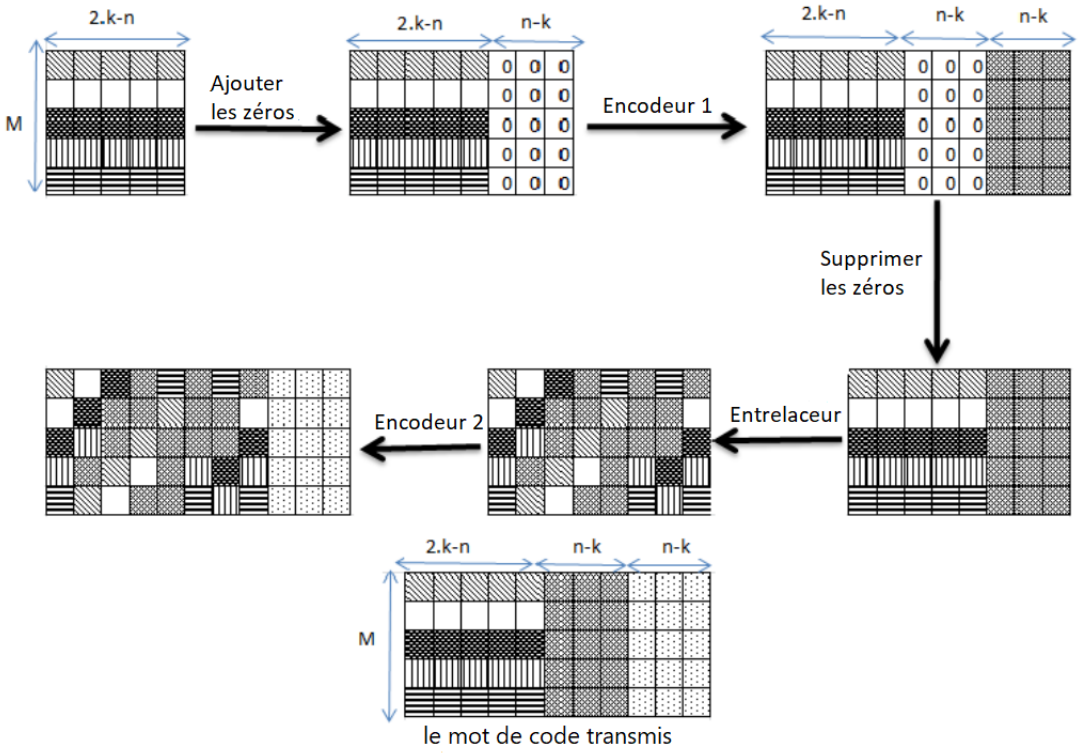


FIGURE 2.2 – Processus d'encodage des codes GCSB

Le tableau 2.1 donne quelques exemples des codes GSCB basés sur la famille de codes RS (Reed-Solomon).

TABLE 2.1 – Exemples de codes GSCB

Code élémentaire 1	Code élémentaire 2	Code GSCB	Taux du code
RS(15,13)	RS(13,11)	GSCB-RS(15,11)	0.73
RS(31,29)	RS(29,27)	GSCB-RS(31,27)	0.87
RS(31,27)	RS(27,25)	GSCB-RS(31,25)	0.81
RS(31,25)	RS(25,23)	GSCB-RS(31,23)	0.74
RS(63,61)	RS(61,59)	GSCB-RS(63,59)	0.94
RS(63,59)	RS(59,57)	GSCB-RS(63,57)	0.90
RS(63,57)	RS(57,55)	GSCB-RS(63,55)	0.87
RS(63,55)	RS(55,53)	GSCB-RS(63,53)	0.84
RS(63,53)	RS(53,51)	GSCB-RS(63,51)	0.81
RS(63,51)	RS(51,49)	GSCB-RS(63,49)	0.78
RS(127,125)	RS(125,123)	GSCB-RS(127,123)	0.97
RS(127,123)	RS(123,121)	GSCB-RS(127,121)	0.95
RS(127,121)	RS(121,119)	GSCB-RS(127,119)	0.94
RS(127,119)	RS(119,117)	GSCB-RS(127,117)	0.92

Le principe de codage des codes GSCB est comme suit :
Le premier codeur prend une matrice de taille $M \times (2k-n)$ représentant le message à coder. Puis, on concatène aux dernière matrice une autre matrice nulle de taille $M \times (n-k)$. Le résultat intermédiaire sera codé par le premier codeur. La sortie du premier codeur est

formée par les deux matrices d'entrée plus une matrice redondante. Ensuite, la matrice nulle sera écrasée. Le résultat obtenu sera entrelacé et aliment le deuxième codeur, ce dernier à son tour générera une deuxième redondance.

2.2.2 Décodeur élémentaire

Considérons une transmission utilisant une modulation de phase binaire codée par un code en bloc de rendement k_i/n ; $i = 1$ *ou* 2. l'entrée du décodeur lorsque le canal est perturbé par un bruit gaussien blanc additif, est égale à :

$$R = E + B$$

, où

$$R = \begin{pmatrix} r_{11} & \cdots & r_{1j} & \cdots & r_{1n} \\ \vdots & \vdots & r_{ij} & \vdots & \vdots \\ r_{m1} & \cdots & r_{mj} & \cdots & r_{mn} \end{pmatrix}$$

est le vecteur d'observation,

$$E = \begin{pmatrix} e_{11} & \cdots & e_{1j} & \cdots & e_{1n} \\ \vdots & \vdots & e_{ij} & \vdots & \vdots \\ e_{m1} & \cdots & e_{mj} & \cdots & e_{mn} \end{pmatrix}$$

$E = (E_j ... E_n)$ $E_j = \pm 1$ le mot émis correspondant à la matrice du mot codé.

$$B = \begin{pmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1n} \\ \vdots & \vdots & b_{ij} & \vdots & \vdots \\ b_{l1} & \cdots & b_{lj} & \cdots & b_{ln} \end{pmatrix}$$

est le bruit blanc dont les composantes b_{ij} sont à moyenne nulle et de même variance σ^2. Le décodage la séquence reçue R est réalisé selon le critère du maximum de vraisemblance est donné par :

$$D = C^i \quad \text{if} \quad Pr(E = C^i/R) > Pr(E = C^l/R) \quad \forall l \neq i \tag{2.1}$$

avec :

$$C^i = \begin{pmatrix} C^i_{11} & \cdots & C^i_{1j} & \cdots & C^i_{1n} \\ \vdots & \vdots & C^i_{ij} & \vdots & \vdots \\ C^i_{m1} & \cdots & C^i_{mj} & \cdots & C^i_{mn} \end{pmatrix}$$

est le $i^{ème}$ mot de code C avec des paramètres (n, i) et

$$D = \begin{pmatrix} d_{11} & \cdots & d_{1j} & \cdots & d_{1n} \\ \vdots & \vdots & d_{ij} & \vdots & \vdots \\ d_{m1} & \cdots & d_{mj} & \cdots & d_{mn} \end{pmatrix}$$

La décision correspondante au maximum de vraisemblance transmise de la sequence R. la régle (3.15) peut être simplifié par :

$$D = C^i \ si : \ |R - C^i|^2 < |R - C^l|^2 \quad \forall l \neq i \tag{2.2}$$

telle que :

$$|R - C^i|^2 = \sum_{j=1}^{n} \sum_{f=1}^{m} (r_{jf} - c_{jf}^i)^2 \tag{2.3}$$

Le décodage est réalisé en utilisant l'algorithme de Chase-Pyndiah[43] qui permet de déterminer les mots de code les plus vraisemblables. Parmi ces mots il sélectionne celui qui est à la distance euclidienne minimale de R. La mesure de fiabilité associée à chaque symbole $c_{jf},\ 1 \leq j \leq n,\ 1 \leq f \leq m$ appartenant au mot décodé peut être déterminée à partir du logarithme du rapport de vraisemblance (LLR) de la decision D_{if} donnée par :

$$\Lambda(C_{jf}) = \ln \left(\frac{P(e_{jf} = +1/R)}{P(e_{jf} = -1/R)} \right) \tag{2.4}$$

avec : e_{jf} l'élement binaire dans la position (j, f) du mot de code transmis E,$1 \leq j \leq n$ et $1 \leq f \leq m$.
l'expression de $\Lambda(C_{jf})$ approximativement dans le cas de AWGN donner par :

$$\Lambda(C_{jf}) = \ln \left(\frac{e^{-\frac{1}{2\sigma^2}\|R - C^{min(+1)}\|^2}}{e^{-\frac{1}{2\sigma^2}\|R - C^{min(-1)}\|^2}} \right) \tag{2.5}$$

$$\Lambda(C_{jf}) = \frac{1}{2\sigma^2} \left[|R - C^{min(-1)}|^2 - |R - C^{min(+1)}|^2 \right] \tag{2.6}$$

Ou $C_{jf}^{min(+1)}$ et $C_{jf}^{min(-1)}$ sont des mots de code ayant la distance euclidienne minimale par rapport R et choisis parmi les sous-ensembles des mot de code donné par l'algorithme de chase-pyndiah telque : $C_{jf}^{min(+1)} = +1$ et $C_{jf}^{min(-1)} = -1$. en devellopant la relation 2.6 on obtient :

$$\Lambda(C_{jf}) = \frac{2}{\sigma^2} \left(r_{jf} + \sum_{x=1, x \neq j}^{n} \sum_{z=1, z \neq f}^{n} r_{xz} c_{xz}^{min(+1)} \rho_{xz} \right) \tag{2.7}$$

avec :

$$((x, z) \neq (j, f)) \quad \rho_{xz} = \begin{cases} 0, & \text{if} \quad c_{xz}^{min(+1)} = c_{xz}^{min(-1)} \\ 1, & \text{if} \quad c_{xz}^{min(+1)} \neq c_{xz}^{min(-1)} \end{cases}$$

En normalisant par $\frac{\sigma^2}{2}$, la fiabilité de la décision binaire en sortie du décodeur est évaluée par :

$$r'_{jf} = (\frac{\sigma^2}{2}).\Lambda(C_{jf}) = r_{jf} + w_{jf}$$

- w_{jf} indépendante de r_{jf} et analogue à l'information extrinsèque du décodeur RS.
- l'expression simplifiée du LLR déterminer à partir des deux mots de code $C^{min(+1)}$ et $C^{min(-1)}$ en utilisant l'algorithme de Chase qui permet de déterminer un sous-ensemble de mots du code parmi lesquels on peut trouver à priori les deux mots recherchés.

$C^{min(+1)}$ doit avoir $-i$ comme élément binaire a la position (j, f).
si on trouve le mot de code $C^{min(+1)}$ on peut écrire la décision $r^{'}_{jf}$ de d_{jf} sous forme :

$$r^{'}_{jf} = \left(\frac{(M^{min(-i)} - M^{min(i)})}{4} \right) c^{min(i)}_{jf}$$

avec $M^{min(-i)}$ et $M^{min(i)}$ représentent respectivement les distances euclidiennes par rapport à R de $c^{min(-i)}_{jf}$ et $c^{min(i)}_{jf}$
Parfois, il arrive que l'on ne puisse pas trouver dans le sous-ensemble de mots de code retourné par l'algorithme de Chase, deux mots à la distance minimale de R ayant des symboles de signes opposées en position $j(1 \leq j \leq n)$. Dans ce cas, le LLR du symbole décidé est simplement donné par la relation :

$$r^{'}_{jf} = (\frac{1}{2}\sigma_R + |r_{ij}|)c^{min(i)}_{jf}$$

tel que σ_R est l'écart-type de la séquence d'entrée du décodeur R. c'est a dire :

$$r^{'}_{jf} = \beta d_{jf}$$

Où d_{jf} est le bits en position j appartenant au mot de code à la distance minimale de R et β est une constante prédéterminée.

2.2.3 Décodage itératif des codes GSCB

La structure du décodeur GSCB est montrée sur la figure 2.3. Pour décoder un mot de code du code GSCB, on aura besoin non seulement de l'information extrinsèque sur les symboles d'information mais, aussi, de l'information extrinsèque sur les symboles de redondances, contrairement aux codes GPCB, on aura besoin de l'information extrinsèque sur les symboles d'information. Le décodeur GSCB est construit de deux décodeurs élémentaires. Le premier décodeur est similaire à celui du décodeur GPCB. Le deuxième est un peu différent du premier. Initialement, le décodeur élémentaire 1 n'a aucune connaissance sur les bits d'information à l'exception de l'information observé sur le canal. Ensuite, il calcule l'information extrinsèque sur les symboles d'information en utilisant le décodeur de Chase-Pyndiah. Cette information extrinsèque est désentrelacée et alimente le décodeur élémentaire 2 conjointement avec la séquence reçue par le canal. Ce dernier génère l'information extrinsèque sur les symboles d'information et les symboles de redondances. Cette procédure détermine une itération. L'information extrinsèque obtenue par le second décodeur est entrelacée, puis alimente le premier décodeur si les itérations sont à poursuivre. Ce processus recommence jusqu'à un nombre d'itérations maximal est atteint. Le décodeur de Chase-Pyndiah utilise deux facteurs α et β. Le choix de ces deux facteurs est déterminé expérimentalement. Nous décrirons ce processus dans la section suivante.

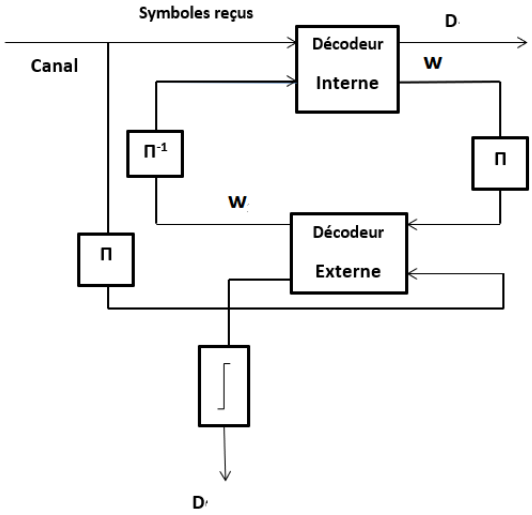


FIGURE 2.3 – Décodage itératifs pour les codes GSCB-RS

2.2.4 Choix des Paramètres α et β

Les paramètres α et β appelé aussi paramètres de pondérations ou de fiabilités, jouent un rôle important dans le décodage itératif des codes produits et peuvent avoir une influence considérable sur les performances des codes GSCB. Ils dépendent bien entendu de l'itération, mais également d'autres critères tels que le rapport signal à bruit, le code élémentaires ou la modulation utilisée. D'un point de vue pratique, il est toujours très difficile de régler et d'optimiser ces coefficients . Dans son travail pyndiah [43] a adopté une solution pour ces paramètres en choisissant des valeurs qui permettent d'atteindre un taux d'erreur de 10^{-5} en un minimum d'itérations.
Plusieurs travaux ont utilisés ses paramètres d'une manière expérimentales ou empiriques, cependant il est difficile de justifier le choix de ces valeurs et de l'évaluer au fil des itérations même si les valeurs utilisées donnent en moyenne de bons résultats.
Dans notre contribution, nous tentons de comprendre des critères dans le choix de ces paramètres α et β et nous proposons une évolution "adaptative" de ces coefficient au fil des itérations. Le terme "adaptatif" est à prendre ici dans le sens d'une évolution en fonction de certains paramètres évalués à chaque itération.

2.2.4.1 α et β prédéterminés

Nous choisissons un code de longueur moyenne et nous prenons une valeur du paramètre M relativement élevée (par exemple M=100). Nous jugeons que ces paramètres sont optimaux, car nous avons fait un compromis entre la longueur du code et la complexité de calculs.
Nous commençons notre processus de tâtonnage par fixer le nombre d'itérations à 1 et nous changeons la valeur de α, avec $0 \leq \alpha \leq 1$, nous retenons la valeur de α donnant la bonne performance. Ensuite, nous changeons, de la même façon, la valeur de β, avec

$0 < \beta \leq 1$, une fois les valeurs de α et β sont choisis pour la première itération nous incrémentons le nombre d'itération, nous cherchons de nouveau les valeurs de α et de β pour la deuxième itération. Ensuite, nous reviendrons en arrière, sans décrémenter le nombre d'itérations, pour ajuster les valeurs de α et de β pour une éventuelle amélioration des performances. Enfin, nous incrémentons le nombre d'itérations et nous répétons le même processus jusqu'à la dernière itération.

2.2.4.2 α et β adaptés

2.2.4.3 Paramètre α

Dans [43], le paramètre α est choisi empiriquement. Sa valeur est celle qui donne un $TEB = 10^{-5}$ avec un minimum d'itérations pour un SNR donné. Cette méthode empiriques est pénible et gourmande en temps de calculs, pour remédier à ce problème nous avons adapté ce paramètre pour les codes produits et les codes en bloc concaténés généralisés. La formule suivante donne l'expression de $\alpha(p)$:

$$\alpha(p) = \frac{1}{\sigma^2_{w(p-1)}} \tag{2.8}$$

où $\sigma^2_{w(p-1)}$ dénote la variance de l'information extrinsèque normalisée du mot de code délivrée par le décodeur précédent. Les performances obtenues en utilisant le paramètre $\alpha(p)$ adapté sont comparables à celles obtenues par le paramètre pré-déterminé.

2.2.4.4 Paramètre β

Le paramètre de pondération β a été utilisé par le decodeur de chase-pyndiah [43] dans le cas d'absence d'un concurent. La valeur de ce paramètre est prédéterminé quelque soit la situation, en d'autre terme la valeur de β ne tient pas compte des autres grandeurs comme les fiabilités de l'entré du décodeur et l'information à priori, la chose qui influence sur les performances du décodeur. Pour pallier à ce problème, nous proposons une formule permutant de calculer la valeur de β en fonction des autres grandeurs influençant sur la décision du décodeur. Donc si le concurrent est absent, tous les mots de code ont un élément c_{ij} égal à d_{ij}. Cela signifie que tous les mots de code ont la même décision concernant l'élément d_{ij}. Par conséquent, le LLR à la sortie du décodeur devrait confirmer cette décision. Dans ce cas, la fiabilité produite par le décodeur doit suivre le fait que tous les mots se mettent d'accord sur la même décision d_{ij}. Le LLR normalisé noté par r'_{d_j} peut être traduit par la relation suivante :

$$r'_{d_j} = \beta . d_j \tag{2.9}$$

avec :

$$\beta = (\frac{1}{2}\sigma_R + |r_{ij}|)$$

est l'écart type de la séquence d'entrée du décodeur R.

2.2.5 Simulation et résultats

Dans cette section, les performances des codes RS concaténés en série sont évaluées. Le canal de transmission est un canal de Gauss à bruit blanc additif(AWGN). Nous avons utilisé une modulation binaire antipodale BPSK. Nous sommes intéressés au taux d'erreur binaire (TEB) pour différents rapports signal-à-bruit par bit d'information $\frac{E_b}{N_0}$ en dB. Plusieurs paramètres affectent les performances des codes GSCB. Nous avons étudié l'influence des paramètres suivants sur le décodeur GSCB : le nombre d'itérations, le code élémentaire, la structure et la longueur de l'entrelaceur. Les paramètres utilisés dans la simulation figurent dans le tableau 2.2.

TABLE 2.2 – Les paramètres de simulation

Paramètre	Valeur
Modulation	BPSK
Environnement	Langage C
Canal	AWGN
Entrelaceur	Random interleaver (par défaut) Entrelaceur diagonale Entrelaceur cyclique Entrelaceur en bloc
α	0.0,0.25,0.3,0.45,0.55, 0.55,0.6,0.65, 0.65, 0.75, 0.75, 0.8, 0.85, 0.9, 0.9, 0.95, 0.95, 0.95, 1.0, 1.05
β	0.3,0.3, 0.35, 0.4, 0.4, 0.45, 0.5, 0.55,0.6, 0.65, 0.7, 0.75, 0.8, 0.8, 0.85, 0.85, 0.9,0.95, 1.0, 1.0
décodeur élémentaire	Chase-Pyndiah(CPA)/ Modified Chase-Pyndiah (MCPA)
Itérations	de 1 à 15 (par défaut)
Taille de l entrelaceur	$1 \times k$, $10 \times k$, $100 \times k$, $300 \times k$

2.2.5.1 Effet des itérations

La figure 2.4 illustre les performances du code GSCB-RS(63, 39). Ici, nous fixons le paramètre multi-bloc à la valeur 300. Sur cette figure, nous remarquons que le TEB s'améliore avec les itérations. Néanmoins, après la 10^{th} itération, le gain de codage devient négligeable pour le décodeur CPA ; par contre, le décodeur MCPA peut aller jusqu'à la 15^{th} itération.

2.2.5.2 Effet de la taille de l'entrelaceur

La figure 2.5 montre l'effet de la taille de l'entrelaceur M sur les performances du code $GSCB - RS(63, 39)$. Le gain atteint $1,4dB$ en passant de $M = 1$ à $M = 10$, diminue à $0,4dB$ entre $M = 10$ et $M = 100$ et devient négligeable au-delà de $M = 100$. Ceci démontre l'efficacité du multi-bloc M.

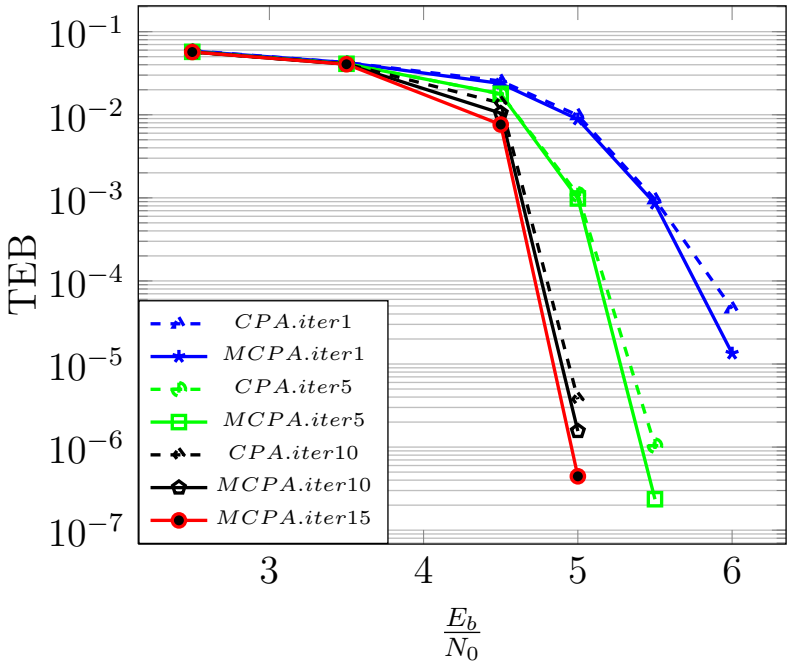


FIGURE 2.4 – Effet du nombre d'itérations sur les performances du code GSCB-RS(63 , 39) Code, avec M=300

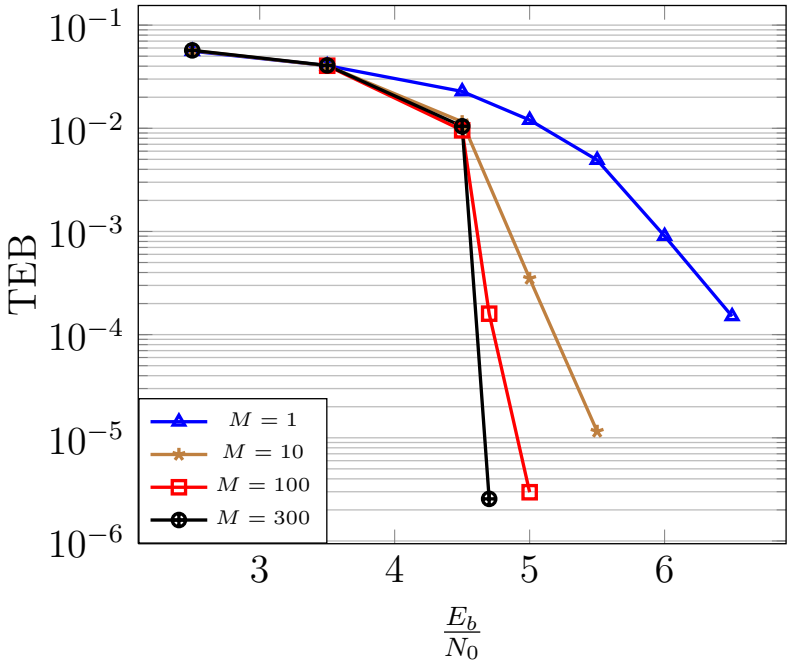


FIGURE 2.5 – Effet de la taille de l'entrelaceur sur les performances du code GSCB-RS(63, 39)

2.2.5.3 Effet de la structure de l'entrelacement sur les performances des codes GSCB

L'entrelaceur est un élément essentiel dans les schémas de codage turbo ou les codes concaténés, en son absence, les performances se trouvent fortement affectée. Cependant,

il est possible de réaliser un gain supplémentaire grâce à l'optimisation de l'entrelaceur. Plusieurs entrelaceurs ont été implémentés dont l'objectif d'étudier l'effet de la structure de l'entrelaceur sur les performances des codes GSCB.

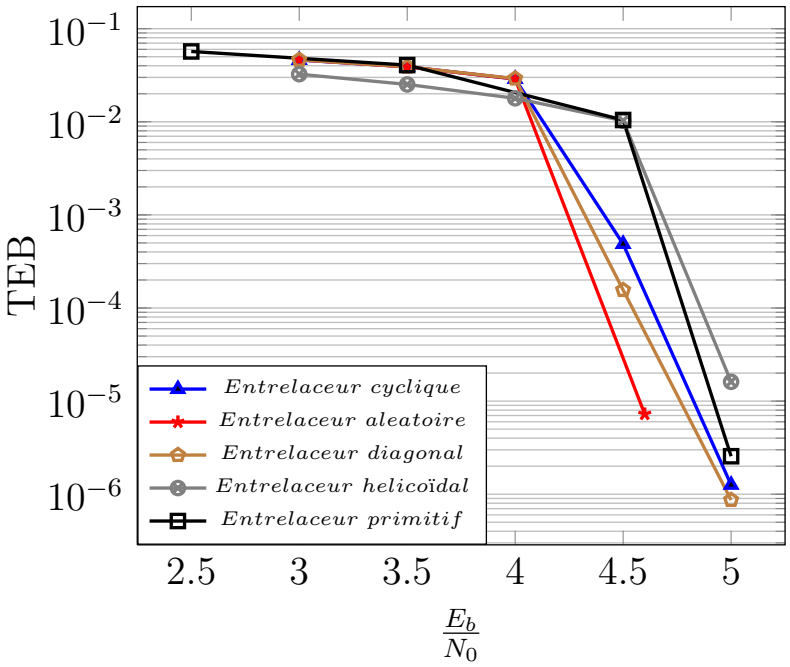


FIGURE 2.6 – Effet de la structure de l'entrelaceur sur les performances des codes GSCB-RS(63,39), avec $M = 300$

Afin d'évaluer l'effet de la structure de l'entrelaceur sur la performance des codes GSCB-RS, nous représentons dans la même figure le TEB en fonction du SNR du code $GSCB - RS(63, 39)$ pour différents entrelaceurs, à savoir l'entrelaceur aléatoire, primitif, cyclique, diagonal et hélicoïdal. Il est à noter que le multi-bloc utilisé ici est $M = 300$. La figure 2.6 représente les résultats de performance. D'après cette figure, nous observons que l'entrelaceur aléatoire surpasse les autres d'environ $0,5$ dB à TEB=10^{-5}.

2.2.5.4 Évaluation des performances des codes GSCB

Dans cette partie nous évaluons les performances des codes GSCB.Une comparaison des performances des codes GSCB-RS (127.85) et GSCB-RS (63.39) est effectuée . Les deux derniers codes ont un taux de codage similaire, dont la valeur R est égale à 0,82 et le nombre de multi-blocs utilisés M est égal à 300. Les performances des derniers codes sont présentées dans la figure 2.7. A partir de ces résultats, nous observons que lorsque nous augmentons la longueur du code élémentaire, la performance se dégrade. Les codes GSCB-RS (63,39) et GSCB-RS (127,85) sont respectivement à 2.1 dB et 2.6 dB de leur limite de Shannon.

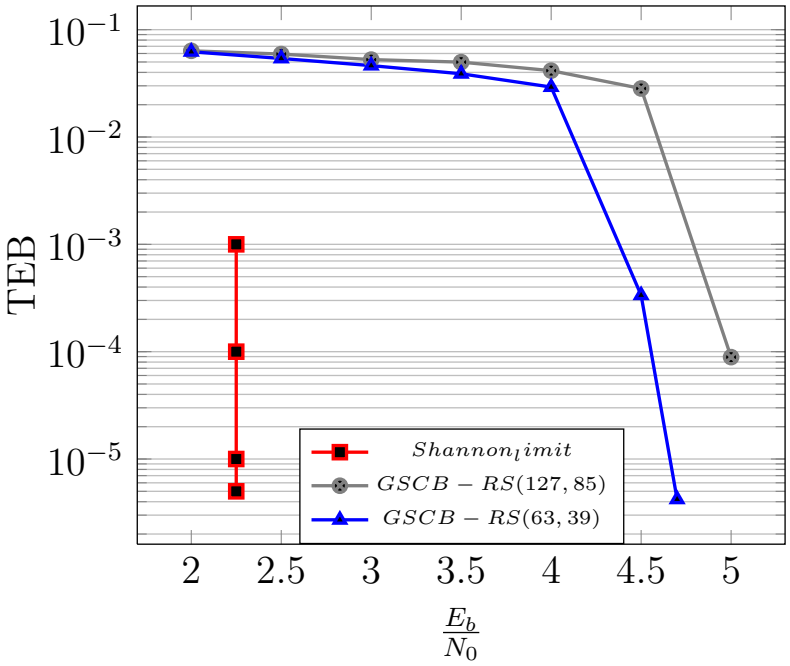


FIGURE 2.7 – Comparaison des performances du deux codes GSCB-RS(127,85) et GSCB-RS(63,39)

2.3 Les codes en bloc concaténés en parallèle généralisés GPCB

La construction des codes PCB à été introduite indépendamment par Nilsson et al.[50],et Benedetto et al.[51]. Pour évaluer ce nouveau schéma, Nilsson et al. Ont utilisé le décodage itératif basé sur le décodeur MAP, alors que Benedetto et al ont évalué théoriquement les performances des codes PCB. En plus la construction donnée par Nilsson est un peu plus générale que celle de Benedetto. Plus tard, Ng et al. [52] ont étudié les codes PCB spécialement pour la famille des codes BCH en utilisant le décodeur de Viterbi à entrée et à sortie pondérés(SISO). Dans leur travail, ils ont évalué les performances en utilisant "union-bounding" et la simulation. Dans ce sens, les auteurs de [45] ont évalué les performances des codes GPCB basé sur les codes BCH, par simulation, en utilisant l'algorithme de Chase-Pyndiah (CPA). Ensuite, ces mêmes auteurs dans leur travail [19], ont décodé les codes en blocs concaténés généralisés tout en modifiant l'algorithme de chase-pyndiah. Il l'on appelé MCPA(Modified Chase-Pyndiah Algorithm).
Dans notre travail, nous avons fait une extension de ces travaux pour les codes GPCB basé sur les codes RS : Nous avons adapté le décodeur MCPA pour décoder les codes RS concaténé en parallèle.
Dans notre étude, basé sur les codes RS, nous avons adapté les facteurs de pondération α et β pour les codes GPCB-RS. De plus, nous avons verifié la validité de l'algorithme MCPA pour ces codes, ainsi que l'investigation de l'effet de plusieurs paramètres à savoir : le code élémentaire, le nombre d'itération, la taille et la structure de l'entrelaceur.
La section suivante présente le principe de codage des codes GPCB à base des codes RS

2.3.1 Construction des codes GPCB

Le schéma de codage des codes GPCB est illustré par la figure 2.8. Deux codeurs en blocs systématiques sont utilisés comme codeurs élémentaires avec un entrelaceur placé avant le second codeur élémentaire. Un bloc de M.k symboles est présenté à l'entrée du codeur. Il est subdivisé en M sous blocs de k symboles pour chacun. Chaque k symbole est encodé pour produire n symboles. Le bloc précédant est brouillé par un entrelaceur avant d'arriver à l'entrée du deuxième codeur. Le mot de code du codes GPCB comme le montre la figure 2.9 est constitué du bloc d'entrée, les symboles de parité produites par le premier et le deuxième codeurs. L'entrelaceur est un élément essentiel dans les schémas de codage turbo, en son absence, les performances se trouvent fortement affectées. Dans cette construction, plusieurs techniques d'entrelacement ont été utilisées : l'entrelaceur aléatoire (random), en bloc, diagonal, et cyclique.

Un code GPCB systématique est basé sur deux codes en bloc systématiques, C_1 de

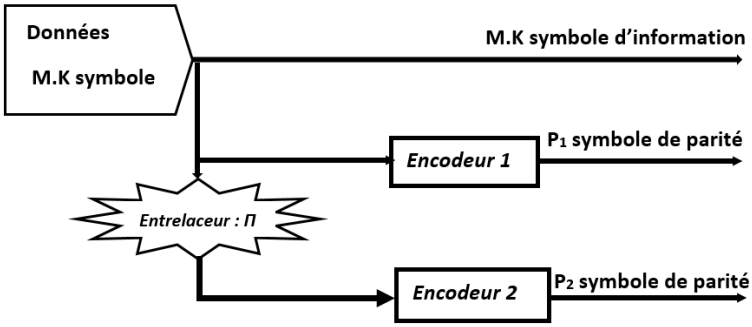


FIGURE 2.8 – Concaténation parallèle de deux codes en bloc

paramètres (n_1, k) et C_2 de paramètres (n_2, k). Considérons le schéma de la figure 2.8 comme un seul codeur GPCB, la longueur du mot d'information à coder est donnée par la taille de l'entrelaceur $N = M.k$. Le premier codeur produit $P_1 = M(n_1 - k)$ symboles de parité. Le second codeur produit $P_2 = M(n_2 - k)$ symboles de parité. Ainsi le nombre total de symboles de parité générés par le codeur GPCB est $P = P_1 + P_2 = M(n_1 + n_2 - 2k)$. La longueur du mot du code GPCB est $L = M.k + P = M(n_1 + n_2 - k)$. Par conséquent, le taux de codage du code GPCB peut être calculer par $R = \frac{N}{L} = \frac{k}{n_1+n_2-k}$. Ceci implique que le taux de codage des codes GPCB est indépendant de la taille de l'entrelaceur.

La table 2.3 donne quelques exemples des codes basés sur cette construction. Les codes en bloc concaténés en parallèle généralisés sont noté par GPCB.

2.3.2 Décodage itératif des codes GPCB

La structure du décodeur GPCB est montrée sur la figure 2.10. à chaque itération deux décodeurs sont utilisés. Le premier prend comme entrée l'information systématiques et les premiers symboles de parité pour générer l'information extrinsèque en utilisant l'algorithme de Chase-Pyndiah. Cette information extrinsèque est utilisée pour mettre à jour

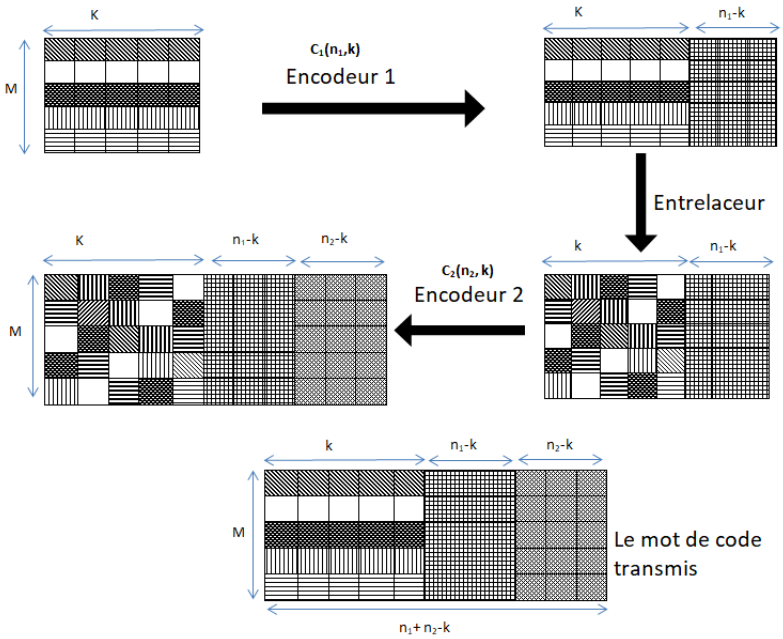


FIGURE 2.9 – Schéma de codage des codes GPCB

TABLE 2.3 – Exemples des codes GPCB

Code élémentaire 1	Code élémentaire 2	Code GPCB	Taux de codage
RS(15,11)	RS(15,11)	GPCB-RS(19,11)	0.578
RS(31,29)	RS(31,29)	GPCB-RS(33,29)	0.878
RS(31,27)	RS(31,27)	GPCB-RS(35,27)	0.771
RS(31,25)	RS(31,25)	GPCB-RS(37,25)	0.675
RS(63,61)	RS(63,61)	GPCB-RS(65,61)	0.938
RS(63, 59)	RS(63, 59)	GPCB-RS(67,59)	0.880
RS(63,57)	RS(63,57)	GPCB-RS(69,57)	0.82
RS(127,125)	RS(127,125)	GPCB-RS(129,125)	0.968
RS(127,123)	RS(127,123)	GPCB-RS(131,123)	0.938
RS(127, 121)	RS(127, 121)	GPCB-RS(133, 121)	0.909
RS(127,119)	RS(127,119)	GPCB-RS(135,119)	0.881
RS(255, 253)	RS(255, 253)	GPCB-RS(257,253)	0.939
RS(255,251)	RS(255,251)	GPCB-RS(259,251)	0.969
RS(255,249)	RS(255,249)	GPCB-RS(261,249)	0.954
RS(255,247)	RS(255,247)	GPCB-RS(263,247)	0.939
RS(255, 245)	RS(255, 245)	GPCB-RS(265,245)	0.924
RS(511,509)	RS(511,509)	GPCB-RS(513,509)	0.992
RS(511,507)	RS(511,507)	GPCB-RS(515,507)	0.984
RS(511,505)	RS(511,505)	GPCB-RS(517,505)	0.976
RS(511,503)	RS(511,503)	GPCB-RS(519,503)	0.969
RS(511,501)	RS(511,501)	GPCB-RS(521,501)	0.961

les fiabilités des bits d'information systématiques. Cette dernière va être entrelacée pour alimenter le deuxième décodeur élémentaire. Dans l'étape suivante, on utilise les seconds symboles de parité. Le deuxième décodeur produit, à son tour, l'information extrinsèque pour mettre à jour de nouveau l'information systématique. Cette procédure définit une itération. L'information extrinsèque obtenue par le second décodeur est désentrelacée,

puis alimente le premier décodeur si les itérations sont à poursuivre. Ce processus recommence jusqu'à un nombre d'itérations maximal est atteint. Le décodeur élémentaire de Chase-Pyndiah utilise deux paramètres α et β. Le choix de ces deux paramètres se fait de la même façon que dans la sous section 2.2.4. Les valeurs obtenues par ce processus sont montrées sur le tableau 2.4. Nous avons utilisé un nouveau schéma de décodage qui est déférents à celui de Pyndiah. Dans ce schéma, la mise à jour des fiabilités concerne juste la partie des symboles d'information. Mais, la partie des symboles de parité reste intact [53, 54]

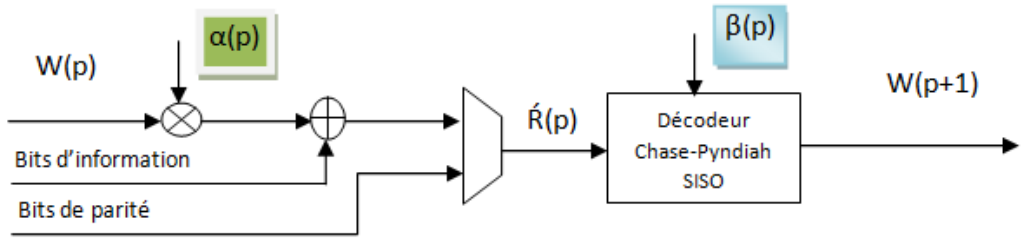


FIGURE 2.10 – Schéma de décodage itératif des codes GPCB

TABLE 2.4 – Les paramètres de simulation des codes GPCB

Paramètre	Valeur
Modulation	BPSK
Environnement	Langage C
Canal	AWGN
Entrelaceur	aléaloire (par défaut) diagonale cyclique Primitif hélicoïdal
α	0, 0.15, 0.25, 0.35, 0.4, 0.5, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.80, 0.85
β	0.2, 0.25, 0.4, 0.45, 0.45, 0.5, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.80, 0.85
décodeur élémentaire	Chase-Pyndiah (CPA) et Modified Chase-Pyndiah (MCPA)
Itérations	de 1 à 7 (par défaut)
Taille de l'entrelaceur	$1 \times k$, $10 \times k$, $100 \times k$, $300 \times k$

2.3.3 Performances des codes GPCB

2.3.3.1 Résultas de simulation

Dans cette section, les performances des codes en bloc concaténés en parallèles généralisés basés sur les codes RS sont évaluées. On utilise la transmission sur un canal à bruit blanc gaussien additif (AWGN) et une modulation binaire antipodale. Nous nous intéressons au taux d'erreur sur les bits d'information (TEB) pour différents rapports

signal/bruit par bit d'information $\frac{E_b}{N_0}$ en dB. De nombreux paramètres affectent les performances des codes GPCB-RS lorsqu'ils sont décodés avec un décodeur itératif. Par conséquent, nous avons étudié les effets des paramètres suivants sur les performances du décodeur, à savoir le nombre d'itérations de décodage, les codes composants et la taille et la structure de l'entrelaceur.

2.3.3.2 Effet itératif

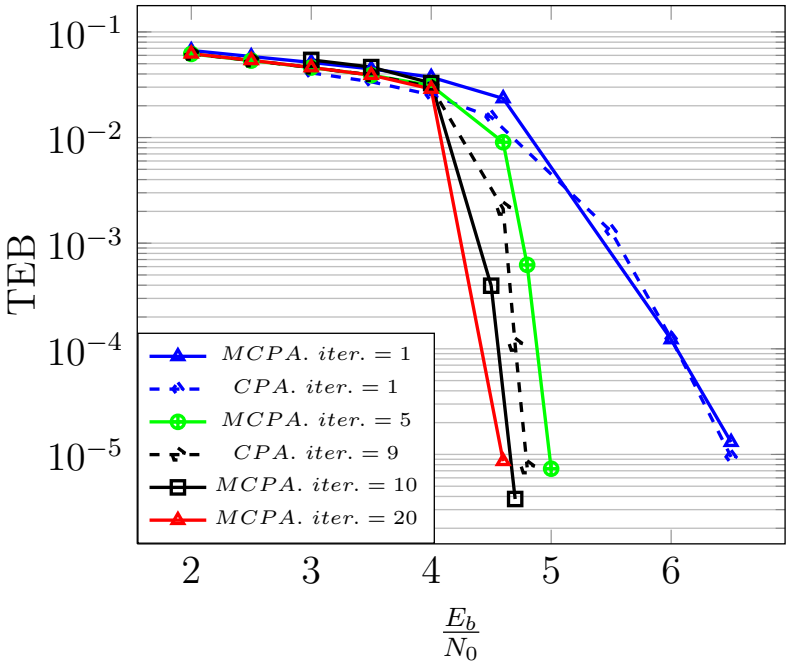


FIGURE 2.11 – l'effet iteratif sur le décodage des codes GPCB-RS(67 , 59) pour un canal AWGN avec M=100

La figure 2.11 présente les performances du code GPCB-RS (67 , 59), avec$M = 100$. Cette figure montre que la pente des courbes et le gain de codage sont améliorés en augmentant le nombre d'itérations. Après la $10^{ième}$ itération, l'amélioration du gain de codage devient négligeable pour le décodeur Chase-Pyndiah (CPA), alors que le décodeur Chase-Pyndiah modifié (MCPA) peut aller jusqu'à la $20^{ième}$ itération. Le décodeur Chase-Pyndiah modifié (MCPA) se compare favorablement avec le décodeur CPA.

2.3.3.3 Effet de multibloc M

Le nombre de multi-bloc codé en même temps améliore les performances des codes concaténés généralisés que ça soit pour la concaténation série ou parallèle.
D'aprés la figure 2.12, le gain atteint $1,4dB$ en passant de $M = 1$ à $M = 10$,diminue à $0,4dB$ entre $M = 10$ et $M = 100$ et devient négligeable au-delà de $M = 100$.Ceci démontre l'efficacité du multi-bloc M. Dans le reste de la simulation nous fixons le nombre de multi-bloc M à 100.

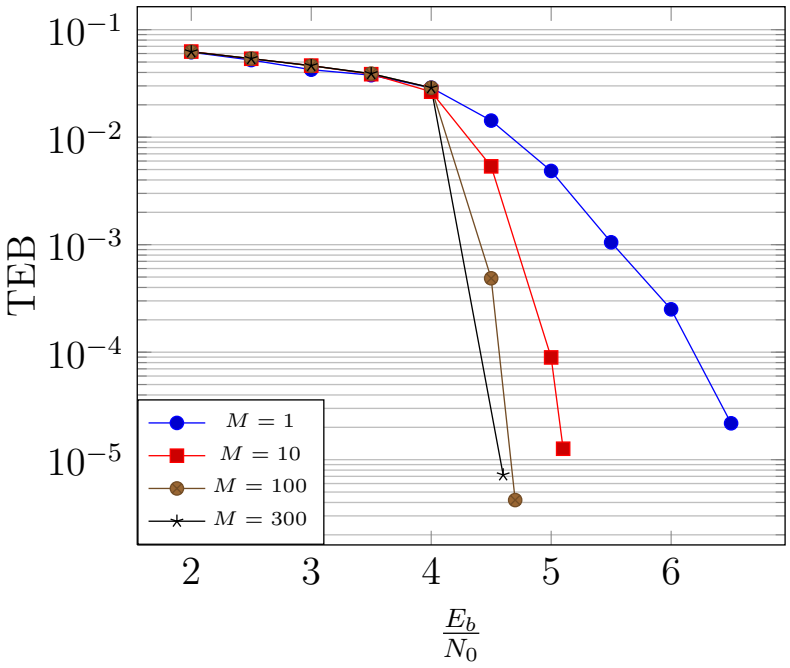


FIGURE 2.12 – Effect of the parameter M on iterative decoding of GPCB-RS(67 , 59) code, over AWGN channel

2.3.3.4 Effet de structure d'entrelaceur

Pour étudier l'influence du motif d'entrelacement sur les performances des codes GPCB-RS, nous avons évalué le TEB en fonction de $\frac{E_b}{N_0}$ du code GPCB-RS $(67, 59)$ en utilisant différentes structures d'entrelacement telles que l'entrelacement diagonal, hélicoïdal, primitif et aléatoire avec le paramètre M=100.
La figure 2.13 montre les résultats de performance. Selon cette figure, nous observons que l'entrelaceur aléatoire surpasse les autres d'environ 0,5dB à TEB=10^{-5}.

2.3.3.5 Évaluation des performances des codes GPCB

Pour évaluer les performances des codes en blocs concaténés parallèles généralisés, nous comparons le gain de codage à la $20^{ième}$ itération des codes suivants GPCB-RS (67, 59), GPCB-RS (131, 123), avec le même taux de codage $0,82$ et le paramètre $M = 100$. Les performances sont présentées dans la figure 2.14. Sur cette figure, nous observons que la performance se dégrade avec l'augmentation de la longueur du code de la composante. Les codes GPCB-RS (69,57), GPCB-RS (131, 123) sont respectivement à 2.3 dB, 2.8 dB de leurs limites de Shannon.

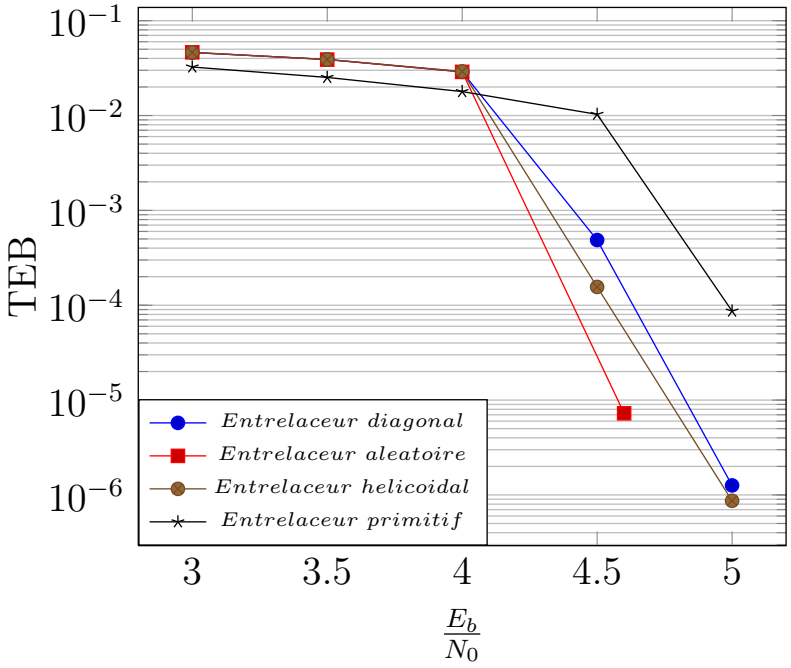


FIGURE 2.13 – Effet de la structure de l'entrelaceur sur les performances du code GPCB-RS(67 , 59), $M = 100$ sur un canal AWGN.

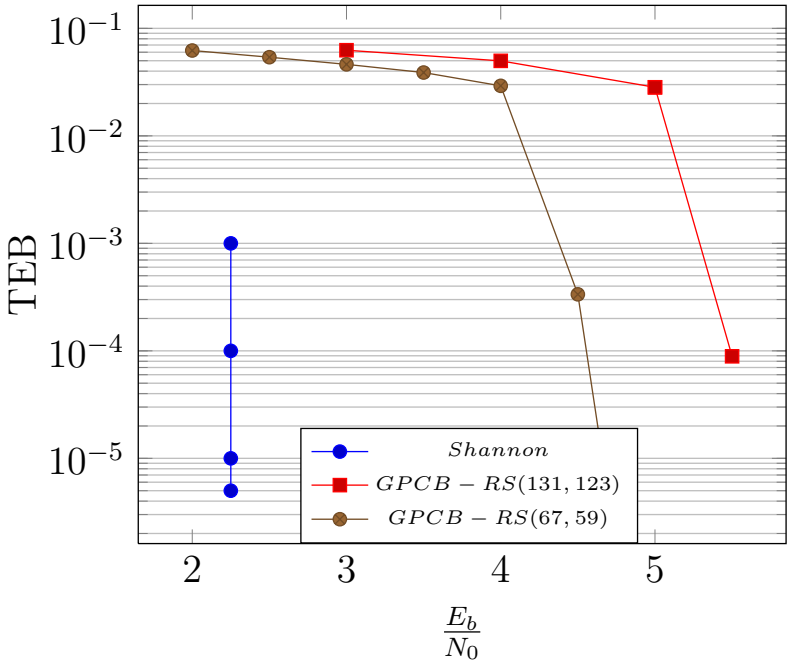


FIGURE 2.14 – Effet du multi-blocs sur le décodage des codes GPCB.

2.3.4 Comparaison entre les codes GSCB et GPCB

Dans cette section, nous comparons les codes concaténés en série et en parallèle généralisés. Pour faire cette comparaison plus honnête, nous avons choisi des codes de longueurs

et de taux de codage proches. Néanmoins nous avons évalué les performances des codes GSCB et les codes GPCB avec 10 itérations.
Les codes $GSCB - RS(63, 39)$ et $GPCB - RS(67, 59)$ ont des taux de codage respectivement, 0.62 et 0.88 avec M=300.Sur la figure 2.15 nous remarquons le code GPCB-RS(67,59) est légèrement meilleur que le code GSCB-RS(63,39) pourtant le dernier code à taux faible par rapport au premier.

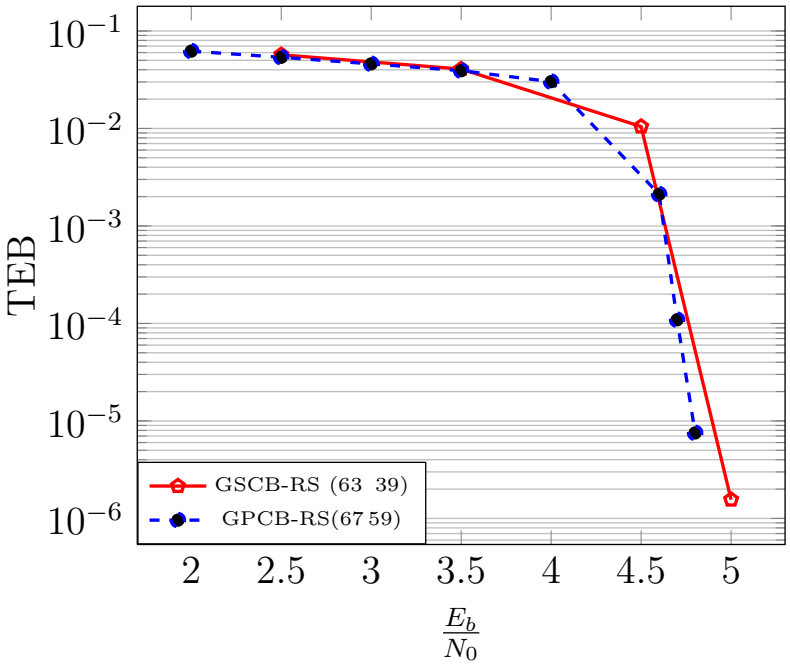


FIGURE 2.15 – Comparaison entre les codes GSCB et les codes GPCB, sur un canal AWGN

2.4 Conclusion

Dans ce chapitre, nous avons réalisé une extension de construction particulière des codes concaténés série (GSCB) et parallèle (GPCB) généralisés basé sur les codes RS.
La construction des codes GSCB consiste à cascader deux codeurs élémentaires symétriques, c'est à dire, nous utilisons le même code comme code interne et externe. Néanmoins le premier codeur est tronqué et le deuxième codeur utilisé de façon normale. Cette manière de construire les codes GSCB permet d'obtenir des codes de taux de codage très élevé.
Nous avons, aussi, démontré l'application de l'algorithme de Chase-Pyndiah pour le décodage des codes GSCB et GPCB en utilisant deux nouveaux schémas de décodage. Nous avons, également, examiné l'effet de plusieurs facteurs sur les performances de ces codes à savoir : le code élémentaire, le nombre d'itérations, la structure et la taille de l'entrelaceur en utilisant la simulation.
La comparaison des codes GSCB et GPCB montre que certain codes GSCB sont meilleurs

que leurs homologues (codes GPCB). Mais, il y en a d'autres codes GSCB qui sont similaires à leurs homologues GPCB.
Les résultats obtenus, en appliquant les constructions précitées, sont prometteurs et ouvrent de nouvelles perspectives.

Chapitre 3

Décodage des codes en blocs produits

3.1 Introduction

Plusieurs travaux ont été faits sur les codes en blocs linéaires spécialement le décodage MLD (décodage à maximum de vraisemblance). Ce dernier n'exploite pas seulement l'information soft des symboles reçus du démodulateur mais il teste toutes les configurations d'erreurs. On peut distinguer entre deux types de décodage souple (soft) : ceux qui traitent les bits d'information les moins faibles comme le décodeur de Chase [14], et ceux qui traitent les bits les plus fiables. Le travail de Dorsch [55] est le pionner de la deuxième catégorie. Dans le décodeur de Dorsch, utilisant un code de paramètres $C(n, k, d_{min})$ et de matrice de parité H_0, une décision dure est obtenue à partir de l'information soft reçue, les bits obtenus, par la décision dure, sont ordonnés en se basant sur leurs fiabilités. Puis, une liste de mots du code est dérivée d'un ensemble de k bits indépendants et plus fiables. Ce traitement est réalisé en transformant la matrice de parité originale H_0 en une autre matrice H obtenue en réarrangeant les colonnes de H_0 selon les fiabilités de chaque position, ensuite on réduit la matrice résultante à sa forme échelonnée. Un décodeur similaire à été proposé par Fossorier [56], mais il ne quantifie pas la décision. Dans ce sens, les auteurs de [20] ont proposé un décodeur similaire . Ce dernier est basé sur le traitement des bits les moins fiables, son principe est comme suit : il trie le vecteur reçu dans l'ordre croissant des fiabilités des bits. Après, il détermine les premiers p colonnes linéairement indépendantes de la matrice H_0. Ensuite, il résout le système d'équations formé par les p equations de parité, mais d'abord on doit s'assurer que les $(n-k-p)$ équations restantes soient satisfaites. Si ces dernières ne sont pas satisfaites le décodeur déclare un échec de décodage sinon il retourne le mot du code trouvé. La valeur de p peut être forcée pour atteindre la valeur $(n-k)$. Dans ce cas, quelques colonnes dépendantes peuvent être ignorées afin de former $(n-k)$ colonnes linéairement indépendantes. Dans tous les cas les performances, en terme de TEB contre SNR, ne se dégradent pas.
Pour construire de long codes qui sont puissants tout en gardant la complexité de décodage relativement faible, Elias [34] a proposé en 1954 les codes produits. Ces codes sont plus résistants aux bursts d'erreurs. Généralement, ces codes sont algébriquement décodable, néanmoins ce décodage algébrique n'améliore pas significativement les performances sous un décodage itératif. En 1965 Forney [57] a proposé une généralisation des codes produits en insérant un entrelaceur entre le code internes et le code externes. Mais, les codes interne et externe se décodent seulement une fois. Toutefois, pour les décoder itérativement, Berrou et al. [15] ont inventé les turbo codes, en 1993. Les décodeurs élé-

mentaires ont des entrée souples et sortie souples siso (pour soft-input soft-ouput). Les codes élémentaires utilisés par Berrou sont des codes convolutionels. Les performances des turbo-codes s'approchent de la limite de Shannon. Une année plus tard, Pyndiah [6, 58] a introduit les turbo codes en blocs. Ces derniers sont basés sur les codes en blocs. Dans ce chapitre, on s'intéresse aux turbo-codes en blocs BTC. En décodage itératif présenté dans [6, 58], chaque décodeur élémentaire d'un code BTC est décodé en deux étapes. Premièrement, l'algorithme de Chase est employé [14]. Deuxièmement, l'information extrinsèque est extraite de la décision soft générée pendant la première étape. Dans ce travail, nous utilisons le décodeur décrit dans [20] à la place de celui de Chase. Dans la deuxième étape, le décodeur de Chase-Pyndiah peut être utilisé. Il utilise deux formules pour calculer la fiabilité de la décision. Mais, dans notre décodeur, nous développons et simplifions la première formule du décodeur de Chase-Pyndiah [6, 58]. Quant à la deuxième formule est remplacée par une autre formule proposé par Farchane et Belkasmi [59]. Le décodeur proposé dépasse celui de Chase-Pyndiah et offre un alternatif pour décoder les turbo-codes en blocs.
Le reste de ce chapitre est organisé comme suit : la section 2 présente le décodeur élémentaire dans sa version de base. Puis, nous introduisons les variantes qui sont des améliorations du décodeur de base, dans la section 3. La section 4, expose une version soft du décodeur élémentaire présenté dans les sections précédentes. Puis, nous donnons une méthode rapide de génération de configurations d'erreurs. Ensuite, nous introduisons le schéma de décodage que nous avons utilisé pour le décodeur proposé. Avant de conclure, nous exposons et discutons les performances du décodeur proposé.

3.2 Décodeur élémentaire

3.2.1 Décodeur de base

Considérons un code en bloc linéaire C de paramètres $C(n, k, d_{min})$. Pour une transmission utilisant la modulation BPSK avec l'état 0 mappé à -1 et l'état 1 mappé à $+1$. Le mot du code transmis est noté $X = (x_1, x_2, ..., x_n)$. Le mot reçu contient n coordonnés $R = (r_1, r_1, ..., r_n)$ avec $r_i = x_i + b_i$, où $1 \leq i \leq n$ et $B = (b_1, ..., b_n)$ est un bruit blanc gaussien (AWGN) de variance σ^2. Les fiabilités des symboles reçus sont proportionnelles aux LLR. Ainsi, les valeurs de $|r_i|$ peuvent être utilisées comme mesures de fiabilités des bits reçus. Soit $\Pi(I_1, I_2, ..., I_n)$ la permutation permettant de réordonner les composantes de R du bit moins fiable au bit le plus fiable. Ainsi, nous aurons $|r_{I_1}| \leq |r_{I_2}| \leq ... \leq |r_{I_n}|$. Soit H_0 la matrice de parité du code C, tel que $H_0 = [PI_{n-k}]$ ait la forme suivante :

$$H_0 = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1k} & 1 & 0 & \dots & 0 \\ p_{21} & p_{22} & \dots & p_{2k} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{(n-k)1} & p_{(n-k)2} & \dots & p_{(n-k)k} & 0 & 0 & \dots & 1 \end{bmatrix} \tag{3.1}$$

Soit p le nombre de premières colonnes linéairement indépendantes qui correspond aux positions des bits les moins fiables tel que $d_{min} \leq p \leq n - k$. En appliquant des opéra-

tions élémentaires sur les lignes de la sous-matrice formée par les p premières colonnes linéairement indépendantes mais sans faire de permutations sur les colonnes. Ainsi, nous obtenons une matrice de parité sous la forme suivante :

$$H=\begin{bmatrix} h_{11} & \dots & 0 & \dots & h_{1I_j} & 1 & \dots & 0 & h_{1(p+1)} & \dots & h_{1n} \\ h_{21} & \dots & 1 & \dots & h_{2I_j} & 0 & \dots & 0 & h_{2(p+1)} & \dots & h_{2n} \\ h_{31} & \dots & 0 & \dots & h_{3I_j} & 0 & \dots & 0 & h_{3(p+1)} & \dots & h_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{p1} & \dots & 0 & \dots & h_{pI_j} & 0 & \dots & 1 & h_{p(p+1)} & \dots & h_{pn} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{(n-k)1} & \dots & 0 & \dots & h_{(n-k)I_j} & 0 & \dots & 0 & h_{(n-k)(p+1)} & \dots & h_{(n-k)n} \end{bmatrix} \tag{3.2}$$

Où $1 \leq i \leq n-k$
Aucune permutation n'est nécessaire pour la matrice de parité, contrairement aux travaux [55, 60, 56] qui ont besoin d'effectuer deux permutations sur la matrice génératrice où de parité, cela dépend de l'espace de travail "Espace G" ou "Espace H".
Maintenant, supposant que $Z = (z_1, z_2, ..., z_n)$ dénote la décision ferme de la sequence reçue R tel que $z_i = 0.5(1 + sgn(r_i))$ avec $z_i \in \{0, 1\}$. Puis, nous calculons le vecteur syndrome $S = (s_1, ..., s_{n-k})$ par l'équation suivante :

$$S = H.Z^T.$$

Dénotons par $E = (e_1, e_2, ..., e_n)$ le vecteur d'erreur, et supposons que les $(n-p)$ plus fiables ne contient pas d'erreur. Ceci vaut dire que

$$(e_{I_{p+1}} = e_{I_{p+2}} = ... = e_{I_n} = 0).$$

Alors, le vecteur d'erreur E peut être écrite sous la forme :

$$E = (0, ..., e_{I_2}, 0, e_{I_1}, ..., e_{I_p}, ..., 0).$$

Le vecteur E contient p composantes inconnus et les $(n-p)$ autres composantes sont mises à zéro. Les inconnus peuvent être déterminés en utilisant les équations de parité suivantes.

$$\begin{aligned} (X \oplus E) \cdot H^T &= S \\ X \cdot H^T \oplus E \cdot H^T &= S \\ E \cdot H^T &= S \end{aligned} \tag{3.3}$$

le terme $H.X^T$ est nul puisque X est un mot du code qui a pour matrice de parité H.

Si nous développons la relation (3.3) nous obtenons le système linéaire suivant :

$$\begin{cases} e_{I_1} = s_1 \\ e_{I_2} = s_2 \\ ... = ... \\ e_{I_p} = s_p \\ 0 = s_{p+1} \\ ... = ... \\ 0 = s_{n-k} \end{cases} \tag{3.4}$$

si les équations suivantes sont satisfaites $s_{p+1} = s_{p+2} = ...s_{n-k} = 0$, le système linéaire admit la solution suivante :

$$e_{I_j} = \begin{cases} s_j, & \text{pour } 1 \leq j \leq p\,; \\ 0, & \text{pour } j > p. \end{cases}$$

Sinon, le system (3.4) n'admit pas de solution pour cette configuration d'erreur. Alors, les erreurs ne sont pas localisées seulement dans les p positions moins fiables. Par conséquent, les $(n-p)$ positions les plus fiables contiennent aussi des erreurs. Dans ce cas, il n'y a pas de solution , et par conséquent un échec de décodage est déclaré par le décodeur. Le mot décodé est considéré comme une décision ferme de la séquence reçue. Dans la suite, nous appelons ce traitement "traitement d'ordre zero" et nous le notons Dual-R-0. Contrairement aux décodeurs publié dans [56, 60, 61], le décodeur proposé n'a pas besoin d'appliquer des permutation inverse sur le mot décodé.
Pour corriger des erreurs contenues dans la partie fiable, nous proposons une variante de cet algorithme. Cette dernière permet de traiter les erreurs résiduelle. Ce point sera traité dans la section suivante.

3.2.2 Traitement des erreurs résiduelles

Le décodeur décrit dans la sous-section (3.2.1) peut corriger p effacements (erasures), où $d_{min} - 1 \leq p \leq n - k$. Néanmoins, si nous voulons corriger des erreurs dans la partie fiable nous générons des configurations d'erreurs en se basant sur les $(n-p)$ positions fiables, ensuite nous utilisons ces configurations pour résoudre le système d'équations suivant :

$$S = H.E^T.$$

Supposons que nous voulions corriger une erreur de poids 1, dans les positions fiables, ainsi que les effacements contenues dans la partie moins fiable. Soit e_j la j^{eme} composante de l'erreur à corriger avec $j > I_p$. Alors, le vecteur E peut être écrit sous la forme suivante :

$$E = (0, ..., e_{I_2}, ..., e_j, ..., 0, ..., e_{I_1}, ..., e_{I_p}, ..., 0).$$

en se basant sur ces suppositions nous pouvons récrire les équations de parité sous la forme suivante :

$$\begin{cases} e_{I_1} \oplus h_{1j}.e_j = s_1 \\ e_{I_2} \oplus h_{2j}.e_j = s_2 \\ \qquad\qquad ..... = ... \\ e_{I_p} \oplus h_{pj}.e_j = s_p \\ h_{(p+1)j}.e_j = s_{p+1} \\ \qquad\qquad ..... = ... \\ h_{(n-k)j}.e_j = s_{n-k} \end{cases} \tag{3.5}$$

le système linéaire (3.5) devient :

$$\begin{cases} e_{I_1} = s_1 \oplus h_{1j}.e_j \\ e_{I_2} = s_2 \oplus h_{2j}.e_j \\ ..... = ... \\ e_{I_p} = s_p \oplus h_{pj}.e_j \\ 0 = s_{p+1} \oplus h_{(p+1)j}.e_j \\ ..... = ... \\ 0 = s_{n-k} \oplus h_{(n-k)j}.e_j \end{cases} \tag{3.6}$$

Le système (3.6) a une solution si et seulement si les contraintes suivantes sont satisfaites :

$$\begin{cases} s_{p+1} \oplus h_{(p+1)j}.e_j = 0 \\ s_{p+2} \oplus h_{(p+2)j}.e_j = 0 \\ \qquad\qquad \cdots \\ s_{n-k} \oplus h_{(n-k)j}.e_j = 0. \end{cases}$$

sinon le système (3.6) n'a pas de solution, donc cette configuration d'erreur n'est pas valide. Ensuite, on traitera la configuration suivante. Le nombre de configurations de poids 1 est $\binom{n-p}{1}$. Une fois toutes les configurations de poids 1 sont traitées on obtient une liste de mots du code à partir de laquelle on choisit un mot du code. Ce traitement sera noté Dual-R-1. Si on veut améliorer davantage les performances on traite les configurations de poids 2, ce traitement sera noté Dual-R-2. De manière générale, on peut traiter les configurations de poids m, dans ce cas le décodeur sera noter Dual-R-m.
Sur un canal AWGN, on peut montrer que la décision optimale $D = (d_1, d_2, ..., d_n)$, correspondant au mot transmis X, est celui qui minimise la distance euclidienne par rapport à la séquence reçue. Ceci est exprimé par la formule suivante :

$$D = \underset{c \in \mathsf{C}}{\text{Argmin}} \sum_{l=1}^{l=n} (r_l - c_l)^2 \tag{3.7}$$

où $C = (c_1, c_2, ..., c_n)$ est un mot du code C et $\sum_{l=1}^{l=n}(r_l - c_l)^2$ est la distance euclidienne

entre la séquence reçue R et le mot du code C.
L'équation (3.7) peut être développée pour avoir la relation observée dans (3.8).

$$D = \underset{c\in\mathsf{C}}{\text{Argmin}} \sum_{l=1}^{l=n} r_l^2 - 2.r_l.c_l + c_l^2 \tag{3.8}$$

Les termes r_l^2 et c_l^2 sont constants pour $C \in \mathsf{C}$. Nous remarquons que le terme $-2.r_l c_l$ est le seul qui affecte le problème de minimisation de l'équation (3.8). Si nous supprimons le coefficient -2 le problème de minimisation devient un problème de maximisation de l'équation (3.9).

$$D = \underset{c\in\mathsf{C}}{\text{Argmax}} \sum_{l=1}^{l=n} r_l.c_l \tag{3.9}$$

Supposons que $\mathbb{E}$ dénote l'ensemble de vecteurs d'erreur, Z dénote la décision ferme de la séquence reçue R, et C dénote le mot décodé modulé tel que $c_l = 2.(z_l \oplus e_l) - 1$, où E est un vecteur d'erreur appartenant à l'ensemble des erreurs $\mathbb{E}$. maintenant, nous remplaçons C avec son expression dans la relation (3.9), puis nous développons cette dernière pour avoir la relation observée dans (3.10).

$$D \quad = \quad \underset{E\in\mathbb{E}}{\text{Argmax}} \sum_{l=1}^{l=n} r_l.(2.(z_l \oplus e_l) - 1) \tag{3.10}$$

Si nous développons le terme $(2.(z_l \oplus e_l) - 1)$ nous aurons la relation suivante :

$$(2.(z_l \oplus e_l) - 1) \quad = \quad (2.z_l - 1) - 2.(2.z_l - 1).e_l \tag{3.11}$$

Maintenant, nous injectons l'équation (3.11) dans l'équation (3.10). Alors, cette dernière peut être développée pour donner la relation suivante (3.12) :

$$D \quad = \quad \underset{E\in\mathbb{E}}{\text{Argmax}} \sum_{l=1}^{l=n} r_l.(2.z_l - 1) - 2.r_l(2.z_l - 1).e_l \tag{3.12}$$

Les termes $(2.z_l - 1)$ et r_l, de l'équation (3.12), ont le même signe puisque z_l est la décision ferme de r_l. Donc $r_l.(2.z_l - 1)$ égal à $|r_l|$. Nous remplaçons le dernier terme dans l'équation (3.12). Ainsi nous obtenons la relation suivante :

$$D \quad = \quad \underset{E\in\mathbb{E}}{\text{Argmax}} \sum_{l=1}^{l=n} |r_l| - 2.|r_l|.e_l \tag{3.13}$$

La composante $|r_l|$ de l'équation (3.13) est la même pour tous les mots du code de la liste, alors il n'affecte pas le problème de maximisation. Le coefficient -2 peut être supprimer de la relation (3.13), en revanche le problème de maximisation devient un problème de minimisation de l'équation (3.14) :

$$D \quad = \quad \underset{E\in\mathbb{E}}{\text{Argmin}} \sum_{l=1}^{l=n} |r_l|.e_l \tag{3.14}$$

Finalement, la minimisation de la métrique observée sur l'équation (3.14) est équivalente à la minimisation de la distance euclidienne. Ainsi, pour chaque solution nous calculons la métrique définie dans l'expression (3.14) et nous choisissons le mot du code qui minimise cette métrique. Ce traitement constitue ce que l'on appelle retraitement d'ordre 1 et va être noté Dual-R-1 ou Dual-R-m pour un ordre supérieur où $1 \leq m \leq n - p$.

3.2.3 Heuristiques et Critère d'arrêt

La complexité de l'algorithme Dual-R-m est dominée par le calcul de la métrique associée à chaque configurations d'erreur générée et la métrique du mot décodé. Alors, réduire la complexité de l'algorithme revient à réduire la taille de la liste de configurations et/ou à simplifier le calcul de la métrique associée. Les auteurs de [61, 62] ont développé des fonctions d'évaluation ou encore appelées fonctions heuristiques $\triangle(t_i)$, $f(t_i)$ qui guident la recherche de la configuration d'erreur la plus vraisemblable, où $e_j = t_{ij}$ pour $j > p$. Soit $F(t_i)$ une fonction d'évaluation arbitraire. Si $F(t_i) > L(\hat{E})$, où $L(.)$ dénote la métrique utilisée, alors il est inutile de déterminer le vecteur d'erreur E à partir du vecteur de test t_i, car le vecteur d'erreur qui en résulte ne peut pas être mieux que l'erreur le plus vraisemblable $\hat{E}$. Dans [56, 61, 62], ils ont donné deux fonctions heuristiques que l'on a adapté à notre décodeur Dual-R-m :

1. La première est définie comme suit :

$$\triangle(t_i) = \sum_{j=p+1}^{n} t_{iI_j} \mid r_{I_j} \mid \tag{3.15}$$

   Où p représente les premières colonnes indépendantes, i est le poids de la configuration d'erreur.

2. La deuxième fonction heuristique a été utilisé par les algorithmes MLD décrètent dans [56, 61, 62]. L'heuristique $f(t_i)$ est une borne inférieure serrée à la métrique dite "correlation discrepancy". La fonction $f(t_i)$ utilise le faite que la distance de Hamming entre deux mots du code c_i et c_{seed} est supérieure ou égal à d. Les auteurs de [56, 62] mettent $c_{seed} = \hat{c}$, avec $\hat{c} = Z + \hat{E}$. Soit $B(\hat{c})$ l'ensemble qui dénote les positions des éléments nuls de $\hat{E}$. Soit $A(\hat{c}, E)$ l'ensemble de $d - w_H(\hat{E}) - w_H(E)$ positions moins fiables de $B(\hat{c})$. Enfin, la fonction $f(E)$ est définie comme suit :

$$f(E) = \triangle(E) + \sum_{j \in A(\hat{c},E)} \mid r_{I_j} \mid \tag{3.16}$$

   Cette fonction a été utilisée par l'algorithme A^* dans [62]. Pour chaque vecteur de test généré, le décodeur calcule l'heuristique associe en utilisant l'un de ces deux fonctions heuristiques (3.15) et (3.16). Si la valeur de l'heuristique est supérieur ou égal à la métrique du mot du code le plus proche du mot reçu, le vecteur de test est rejeté.

3.2.4 Génération rapide de configurations d'erreurs

On peut réduire le nombre d'opérations nécessaire pour générer un vecteur d'erreur. Cela se fait en réutilisant les vecteurs de test déjà générés. Soit E^i un vecteur d'erreur de poids i, dont les positions d'erreurs sont $j_1 < ... < j_i$ avec $I_p < j_1$, et E^{i+1} un vecteur d'erreur de poids $i+1$, dont les positions d'erreurs sont $j_1 < ... < j_i < j_{i+1}$. Les équations de parité du vecteur E^i sont données par l'équation 3.17

$$\begin{cases} e^i_{I_1} = s_1 \oplus (h_{1j_1}.e^i_{j_1} \oplus h_{1j_2}.e^i_{j_2} \oplus ... \oplus h_{1j_i}.e^i_{j_i}) \\ e^i_{I_2} = s_2 \oplus (h_{2j_1}.e^i_{j_1} \oplus h_{2j_2}.e^i_{j_2} \oplus ... \oplus h_{2j_i}.e^i_{j_i}) \\ .... = ... \\ e^i_{I_p} = s_p \oplus (h_{pj_1}.e^i_{j_1} \oplus h_{pj_2}.e^i_{j_2} \oplus ... \oplus h_{pj_i}.e^i_{j_i}) \\ 0 = s_{p+1} \oplus (h_{(p+1)j_1}.e^i_{j_1} \oplus h_{(p+1)j_2}.e^i_{j_2} \oplus ... \oplus h_{(p+1)j_i}.e^i_{j_i}) \\ .... = ... \\ 0 = s_{n-k} \oplus (h_{(n-k)j_1}.e^i_{j_1} \oplus h_{(n-k)j_2}.e^i_{j_2} \oplus ... \oplus h_{(n-k)j_i}.e^i_{j_i}) \end{cases} \tag{3.17}$$

et celles du vecteur E^{i+1} sont données par l'équation 3.18 suivante :

$$\begin{cases} e^{i+1}_{I_1} = s_1 \oplus (h_{1j_1}.e^{i+1}_{j_1} \oplus h_{1j_2}.e^{i+1}_{j_2} \oplus ... \oplus h_{1j_i}.e^{i+1}_{j_i}) \oplus h_{1j_{i+1}}.e^{i+1}_{j_{i+1}} \\ e^{i+1}_{I_2} = s_2 \oplus (h_{2j_1}.e^{i+1}_{j_1} \oplus h_{2j_2}.e^{i+1}_{j_2} \oplus ... \oplus h_{2j_i}.e^{i+1}_{j_i}) \oplus h_{2j_{i+1}}.e^{i+1}_{j_{i+1}} \\ .... = ... \\ e^{i+1}_{I_p} = s_p \oplus (h_{pj_1}.e^{i+1}_{j_1} \oplus h_{pj_2}.e^{i+1}_{j_2} \oplus ... \oplus h_{pj_i}.e^{i+1}_{j_i}) \oplus h_{pj_{i+1}}.e^{i+1}_{j_{i+1}} \\ 0 = s_{p+1} \oplus (h_{(p+1)j_1}.e^{i+1}_{j_1} \oplus h_{(p+1)j_2}.e^{i+1}_{j_2} \oplus ... \oplus h_{(p+1)j_i}.e^{i+1}_{j_i}) \oplus h_{(p+1)j_{i+1}}.e^{i+1}_{j_{i+1}} \\ .... = ... \\ 0 = s_{n-k} \oplus (h_{(n-k)j_1}.e^{i+1}_{j_1} \oplus h_{(n-k)j_2}.e^{i+1}_{j_2} \oplus ... \oplus h_{(n-k)j_i}.e^{i+1}_{j_i}) \oplus h_{(n-k)j_{i+1}}.e^{i+1}_{j_{i+1}} \end{cases} \tag{3.18}$$

Si $e^i_{j_l} = e^{i+1}_{j_l}$ pour $I_p < l \leq j_i$, alors l'équation 3.18 devient :

$$\begin{cases} e^{i+1}_{I_1} = e^i_{I_1} \oplus h_{1j_{i+1}}.e^{i+1}_{j_{i+1}} \\ ..... \\ e^{i+1}_{I_p} = e^i_{I_p} \oplus h_{pj_{i+1}}.e^{i+1}_{j_{i+1}} \\ s^{i+1}_{p+1} = s^i_{p+1} \oplus h_{(p+1)j_{i+1}}.e^{i+1}_{j_{i+1}} = 0 \\ ..... \\ s^{i+1}_{n-k} = s^i_{n-k} \oplus h_{(n-k)j_{i+1}}.e_{j_{i+1}} \end{cases} \tag{3.19}$$

Pour profiter des calculs déjà effectués, on peut utiliser les configurations d'erreur E^i et E^{i+1} de sorte que elles ne diffères que d'un seul élément. Dans ce cas, nous pouvons calculer E^{i+1} à partir de E^i avec moins de opération binaire. A partir de l'équation 3.19, on peut dire que le calcul d'une configuration d'erreur peut se faire en $(n-k)$ opérations binaires. Par contre, ce calcul se fait en $\bigcirc(k.n)$ opérations binaires dans [56]. Quant à [63] il peut se faire en $\bigcirc(n)$ opérations binaires. Donc, l'algorithme proposé présente une complexité réduite par rapport à ceux présentés dans [56, 63].

3.3 Décodeur Dual-R-m à sortie souple

Dual-R-m est un décodeur en liste. Cette liste est de taille variable mais on peut lui définir une borne supérieure. Ceci peut être fait en rejetant les mots du code non désirés. Cela veut dire que nous ne gardons que les v mots du code les plus proches de la séquence reçue R, où v désigne la borne supérieure désirée. Quand le décodeur Dual-R-m est alimenté par le mot reçu R, il retourne le mot décidé D, qui est le mot le plus vraisemblable. La fiabilité de chaque élément d_j doit être générée en utilisant LLR du mot transmis, cette fiabilité est donnée par la relation suivante :

$$\Lambda(d_j) = \ln(\frac{Pr\{x_j = +1/R\}}{Pr\{x_j = -1/R\}}) \tag{3.20}$$

L'équation (3.20) peut être simplifiée et devient la relation observée en (3.21). Pour plus de détailles, le lecteur intéressé peut consulter la référence [58] :

$$\Lambda(d_j) = \frac{1}{2.\sigma^2}(||R - C||^2 - ||R - D||^2) \tag{3.21}$$

où C est le mot du code le plus proche de R et qui a un élément opposé au composant d_j en position j. Cela vaut dire que $c_j \neq d_j$. C peut être vu comme le mot du code concurrent de D. Si nous supposons que l'écart type σ soit constant, nous pouvons normaliser $\Lambda(d_j)$ par la constante $\frac{2}{\sigma^2}$. Ainsi, nous obtenons l'équation suivante :

$$\acute{\Lambda}(d_j) = \frac{1}{4}(||R - C||^2 - ||R - D||^2) \tag{3.22}$$

En développant la relation (3.22) nous obtenons l'équation (3.23)

$$\acute{\Lambda}(d_j) = \frac{1}{2}\sum_{l=1}^{l=n} r_l.d_l - r_l.c_l \tag{3.23}$$

En remplaçant d_l et c_l dans (3.23) par leurs expressions $d_l = 2.(z_l \oplus e_l^{(d)}) - 1$ et $c_l = 2.(z_l \oplus e_l^{(c)}) - 1$, nous pouvons calculer $\acute{\Lambda}(d_j)$ par l'équation (3.24).

$$\begin{aligned}
\acute{\Lambda}(d_j) &= \frac{1}{2}\Big(\sum_{l=1}^{l=n} r_l.(2.(z_l \oplus e_l^{(d)}) - 1) - r_l.(2.(z_l \oplus e_l^{(c)}) - 1)\Big) \\
&= \sum_{l=1}^{l=n}\Big(r_l(2.z_l - 1).e_l^{(c)} - r_l(2.z_l - 1).e_l^{(d)}\Big) \\
&= \sum_{l=1}^{l=n}|r_l|.e_l^{(c)} - \sum_{l=1}^{l=n}|r_l|.e_l^{(d)}
\end{aligned} \tag{3.24}$$

où $E^{(c)} = (e_1^{(c)}, ...e_n^{(c)})$ (respectivement $E^{(d)} = (e_1^{(d)}, ...e_n^{(d)})$) est le vecteur d'erreur associé au mot du code C (respectivement D). A partir de l'équation (3.24) on peut remarquer que la fiabilité de l'élément d_j est la déférence entre les métriques associe au

vecteur d'erreur concurrent $E^{(c)}$ et au vecteur d'erreur décidé $E^{(d)}$. Une fois le vecteur d'erreur concurrent $E^{(c)}$ est trouvé tel que $e_j^{(c)} \neq e_j^{(d)}$, on peut déterminer la fiabilité de d_j à la sortie du décodeur. La valeur de la sortie est donnée par l'expression suivante :

$$\acute{r}_j = \left(\sum_{l=1}^{l=n} |r_l| . e_l^{(c)} - \sum_{l=1}^{l=n} |r_l| . e_l^{(d)} \right) . d_j \tag{3.25}$$

Parfois, on ne peut pas remplir toutes les contraintes sur le vecteur d'erreur concurrent, c'est-à-dire $e_j^{(c)} \neq e_j^{(d)}$. Dans ce cas, on applique la formule (3.25).
la sortie pondérée du décodeur peut être vue comme la déférence entre la métrique associé aux $E^{(c)}$ et $E^{(d)}$, où $E^{(d)}$ est le vecteur d'erreur décidé et $E^{(c)}$ est le vecteur d'erreur tel que $E^{(c)} \neq E^{(d)}$. De plus, $E^{(c)}$ doit réaliser la métrique la plus petite par rapport au mot reçu.
Par fois, le mot reçu est le même que le mot transmis alors le décodeur retourne un seul mot du code et la formule précédente (3.25) ne peut pas être appliquée. Dans ce cas, on peut utiliser la formule (3.26). Cette formule a été proposée dans les travaux [64, 65]. On peut aussi utiliser cette formule dans le cas où le vecteur d'erreur concurrent n'a pas une composantes $e_j^{(c)}$ telle que $e_j^{(c)} \neq e_j^{(d)}$.

$$\acute{r}_j = (\sigma_R + |r_j|) . d_j \tag{3.26}$$

où σ_R est l'écart type du vecteur présenté à l'entrée du décodeur.
Maintenant que nous avons développé une sortie pondérée du décodeur Dual-R-m, nous pouvons l'utiliser comme une brique de base pour la mise en ouvre de turbo-décodage des codes produit. La section suivante abordera ce point.

3.4 Décodage des codes produits

3.4.1 Turbo décodage des codes en blocs produits

Un turbo décodeur est basé sur un décodeur à entrée et sortie pondérée. Dans notre cas, nous utilisons le décodeur Dual-R-m. Le turbo décodage des codes en blocs produits commence par décoder les lignes (ou les colonnes) de la matrice reçue $[R(1)]$, où $[R(1)]$ correspond à la matrice transmise $[X]$ sur un canal AWGN utilisant une modulation BPSK. L'information extrinsèque disponible à la sortie du $q^{ème}$ décodeur peut être calculée en retranchant l'entrée $[R(q)]$ de la sortie $[\acute{R}(q)]$ du décodeur comme suit :

$$[W(q)] = [\acute{R}(q)] - [R(q)] \tag{3.27}$$

l'information extrinsèque $[W(q)]$ est pondérée par un facteur α. Ce facteur est utilisé pour modifier LLR à l'entrée $(q+1)^{ème}$ décodeur comme montré par la formule suivante :

$$[R(q+1)] = [R(1)] + \alpha . [W(q)] \tag{3.28}$$

La valeur de α est déterminée de façon empirique. Nous exposerons, dans la section suivante (3.4.2), la méthode que nous avons utilisée pour déterminer la valeur optimale du

paramètre α.
La matrice $[R(q+1)]$ est entrelacée par un entrelaceur en bloc qui est, en fait, une transposition de la matrice $[R(q+1)]$. Le deuxième décodeur décode tous les lignes (respectivement les colonnes) de la matrice transposée précédente, $[R(q+1)]^T$. Il produit la sortie $[\acute{R}(q+1)]^T$. Cette dernière est utilisée pour calculer de nouveau l'information extrinsèque produite par le deuxième décodeur élémentaire. La formule (3.29) exprime cette quantité :

$$[W(q+1)]^T = [\acute{R}(q+1)]^T - [R(q+1)]^T \tag{3.29}$$

L'information extrinsèque générée par (3.29) sera utilisée pour mettre à jour l'entrée du décodeur de la demi-itération à venir selon la formule suivante :

$$[R(q+2)] = [R(1)] + \alpha.[W(q+1)] \tag{3.30}$$

Toutes ces étapes constituent ce que l'on appelle itération. Ainsi, le processus continu jusqu'à un nombre d'itérations maximales est atteint. L'effet du nombre d'itérations sera aussi étudie dans la partie simulation (section suivante (3.4.2)). Le processus de décodage décrit plus haut est illustré par la figure 3.1.

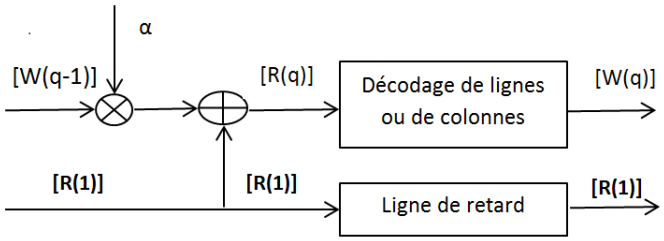


FIGURE 3.1 – Schéma de décodage des turbo-codes en blocs

3.4.2 Résultat et discussion

Dans cette partie, nous nous intéressons aux performances des turbo codes en blocs en terme de TEB vis-à-vis de SNR. Ces performances sont évaluées en utilisant la modulation BPSK sous un canal à bruit blanc additif gaussien (AWGN). Nous considérons des turbo codes en blocs en combinant seulement des codes BCH identiques parce que, selon [58], le fait de combiner différents codes n'améliore pas significativement les performances. Maintenant, nous donnons les paramètres utilisés dans la simulation dans le tableau (3.1) suivant : Le décodeur élémentaire utilisé dans cette étude est Dual-R-m avec le paramètre

TABLE 3.1 – paramètres de simulation

Code produit	d_{min}	R
$BCH(63,51,5)^2$	25	0.66
$BCH(127,113,5)^2$	25	0.79
$BCH(255,247,3)^2$	9	0.94
$BCH(511,502,3)^2$	9	0.96

$m = 2$. Plusieurs valeurs de ce paramètre ont été testées mais au-delà de $m = 2$ on

n'obtient qu'une amélioration minime. Ce décodeur produit une liste de taille variable mais nous avons plafonné cette liste à 40 mots du code. Il est à noter que l'algorithme proposé ne nécessite aucune normalisation de l'information extrinsèque, contrairement au décodeur de Pyndiah [58, 6] qui nécessite une normalisation à la fin de chaque demi-itération.

Le paramètre α joue un rôle vital pour obtenir de bonne performances. Nous essayons, alors de déterminer la valeur optimale de ce paramètre. Pour ce faire, nous suivons les étapes suivantes : Nous choisissons un code de taille moyenne puis nous fixons le SNR ; Sa valeur est choisie de tel sorte que la valeur du TEB soit à l'environ de 10^{-5}. Ensuite, nous fixons le nombre d'itérations à une valeur au-delà de laquelle il n'aurait pas d'amélioration en terme de TEB. Après cela, nous varions la valeur de α dans l'intervalle $]0,\ 1]$ avec un pas très petit, par exemple 0.05 ; Pour chaque valeur de α nous déterminons la valeur du TEB correspondante. Finalement, nous traçons la courbe donnant le TEB contre le SNR. Cette courbe sera analysé pour déterminer la valeur optimale de α. La figure 3.2 présente

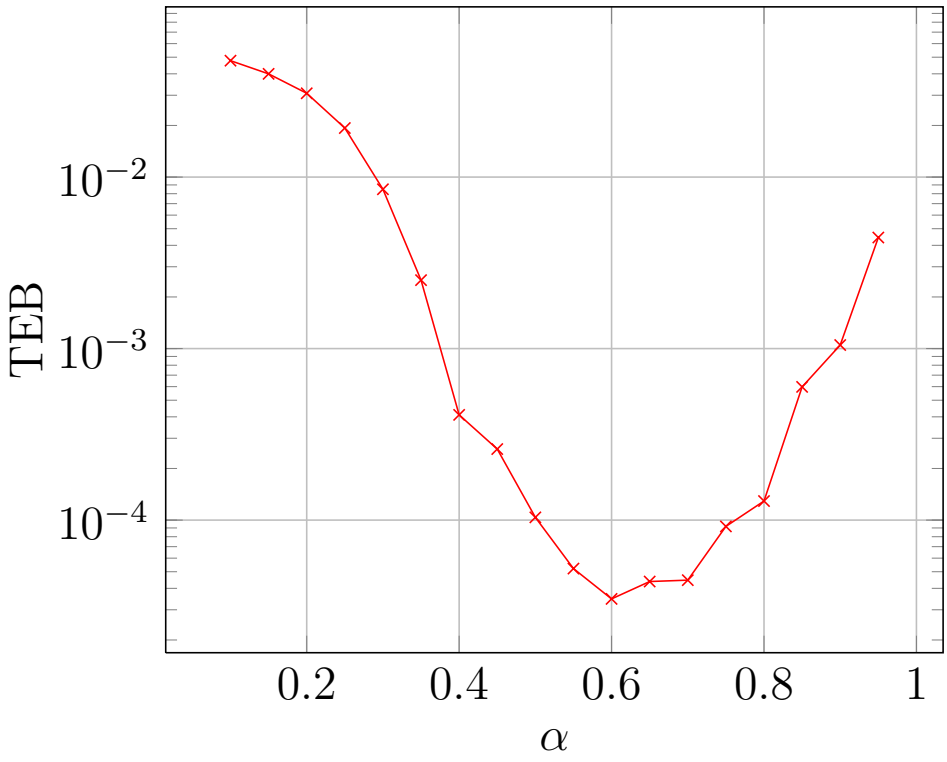


FIGURE 3.2 – TEB en fonction de α du code $BCH(63, 51, 5)^2$, sur un canal AWGN

l'évolution du TEB en fonction du paramètre α. Selon cette figure, nous observons que le TEB diminue en allant de $\alpha = 0.1$ à $\alpha = 0.6$. Au-delà de $\alpha = 0.6$ le TEB augmente. Alors, la valeur $\alpha = 0.6$ donne la plus petite valeur du TEB. Dans la suite de la simulation, nous fixons α à 0.6. Les performances du code $BCH(63, 51, 5)^2$ sont données par la figure 3.3. D'après cette figure, nous observons que ce code présente l'effet turbo comme les codes ctc [15]. Nous remarquons aussi que en passant de l'itération 1 à l'itération 2 nous gagnons $1.4dB$, et en passant de l'itération 2 à l'itération 3 nous gagnons $0.4dB$. Néanmoins, nous ne gagnons que $0.2dB$ en passant de l'itération 3 à 4. Pour chaque itération additionnelle, nous obtenons une réduction du TEB. Néanmoins, pour un TEB égal à 10^{-5}, la réduction en terme du SNR devient négligeable pour les itérations au-delà de l'itération 6. Alors,

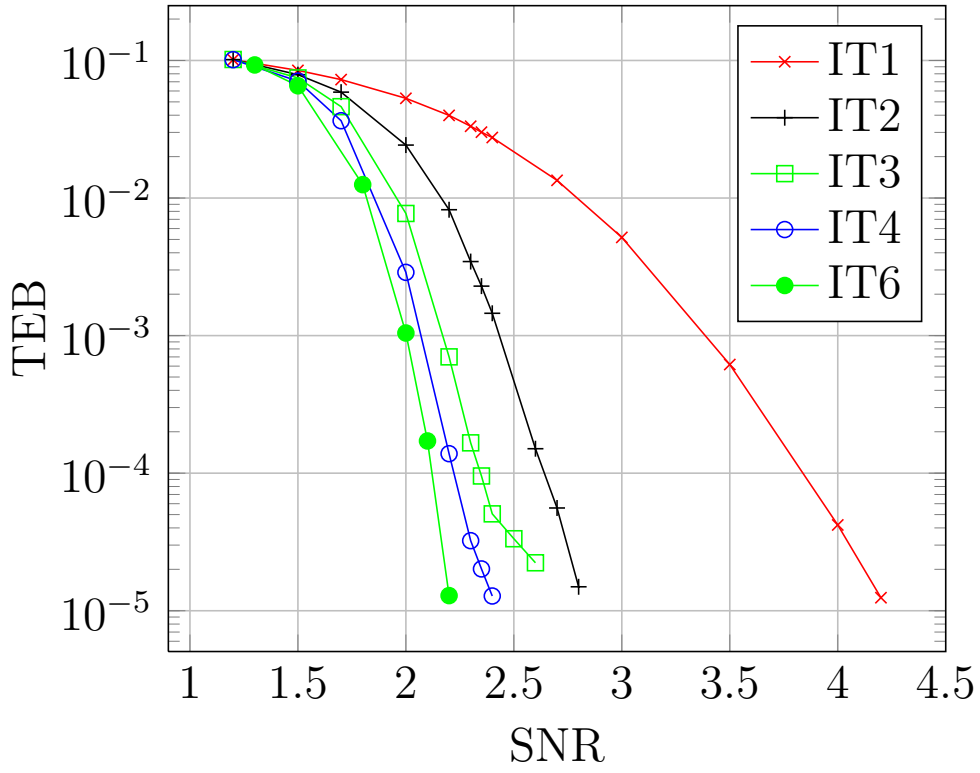


FIGURE 3.3 – TEB en fonction dea SNR du code $BCH(63, 51, 5)^2$, sur un canal AWGN

nous fixons le nombre d'itérations maximale à 4 itérations pour reduire la complexité des calculs.

La figure 3.4 présente les performances du code $BCH(127, 113, 5)^2$. Selon cette figure, nous remarquons que en passant de l'itération 1 à l'itération 2 nous gagnons $1dB$, et en passant de l'itération 2 à l'itération 3 nous gagnons $0.2dB$. Néanmoins, nous ne gagnons que $0.1dB$ en passant de l'itération 3 à 4.

Dans la figure 3.5, nous avons tracé les courbes TEB en fonction de SNR pour différents turbo codes en blocs, sur un canal AWGN, en utilisant la modulation BPSK à l'itération 4. Nous remarquons que la pente des courbes augmente avec le paramètre n et aussi avec la distance minimale d_{min}. D'après les résultats de la simulation, on remarque que notre décodeur dépasse celui de chase-Pyndiah, par exemple, le décodeur proposé dépasse celui de Chase-Pyndiah de 0,6 dB pour le code $BCH(63, 51, 5)^2$. Les performances des turbo codes en blocs pour certaines distances se rapprochent de plus en plus de la limite de Shannon avec l'augmentation du paramètre k. Ceci est cohérent avec le théorème de Gallager [66]. Nous observons que pour le code $BCH(511, 502, 3)^2$, pour un TEB de 10^{-5}, est à 0,55 dB de la limite de Shannon. À notre connaissance, le résultat le plus proche de la limite de Shannon est à 0,35dB de sa limite. Il a été obtenu par Nickl et Hagenaure [67] en utilisant une concaténation parallèle du code $BCH(511, 502, 3)^2$. Mais, la complexité de l'algorithme utilisé augmente exponentiellement avec le nombre de bits de redondance.

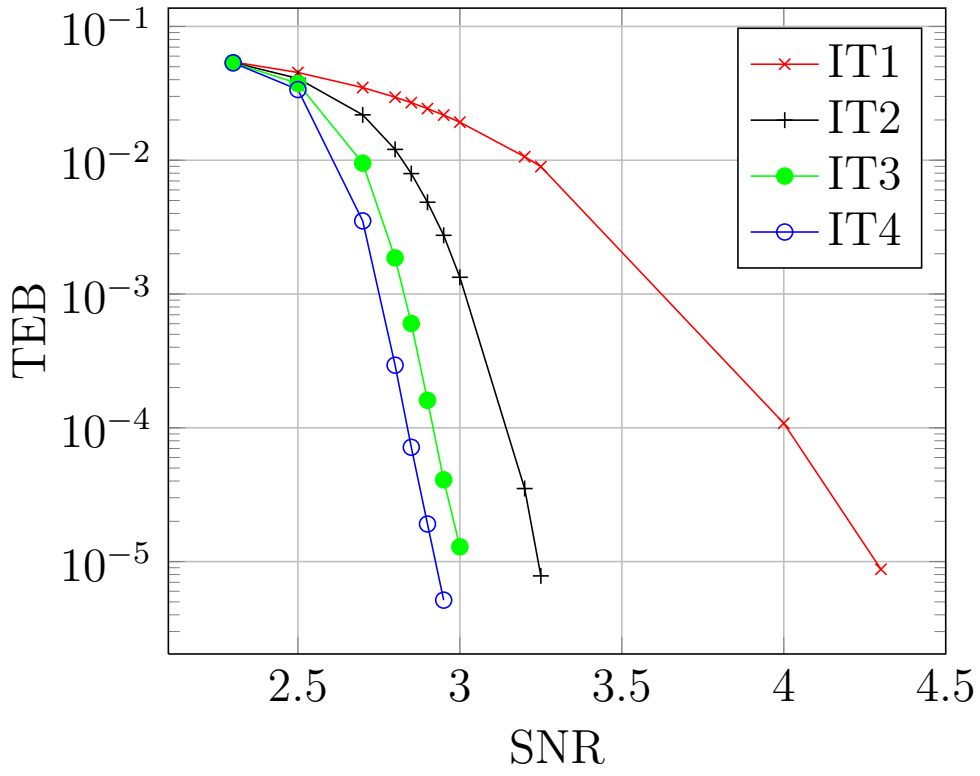


FIGURE 3.4 – TEB contre SNR du code $BCH(127, 113, 5)^2$, sur un canal AWGN

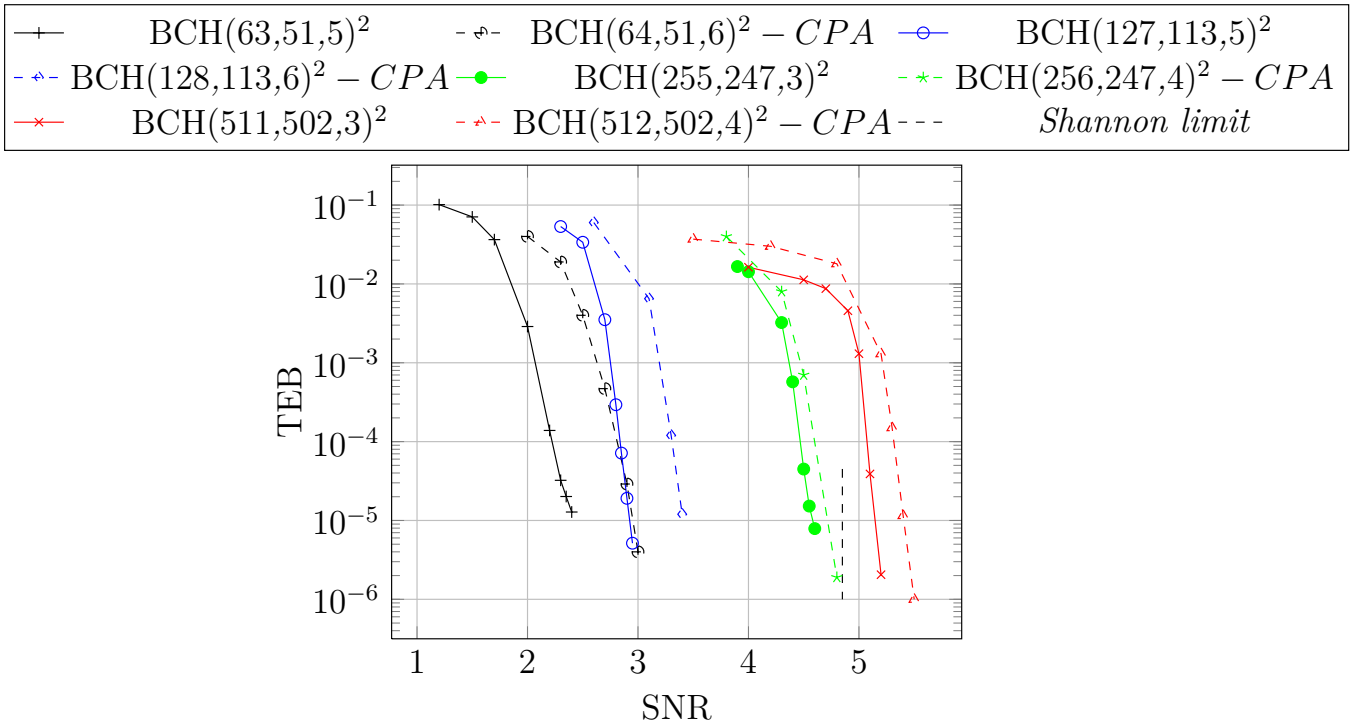


FIGURE 3.5 – TEB contre SNR de codes produits basés sur les codes BCH, sur un canal AWGN.

3.5 Conclusion

Dans ce chapitre, nous avons proposé un algorithme [68] pour décoder les codes en blocs produit. Cet algorithme est basé sur le décodeur Dual-R-m. Ce dernier a été aussi

modifié pour offrir une sortie pondérée (Soft-output). De plus, nous avons utilisé le schéma de Pyndiah pour décoder itérativement les codes en blocs produits. Les résultats de simulation montrent que notre décodeur dépasse celui de chase-Pyndiah. Pour le code $BCH(511, 502, 3)^2$, la performance notre décodeur est de $0.55dB$ de la limite de Shannon [1]. Il opère près de la capacité du canal.
Plusieurs travaux méritent d'être effectués. Dans ce sens, nous citons la réduction de la complexité du décodeur élémentaire, l'utilisation du décodeur proposé pour décoder les codes en blocs concaténés en parallèle comme dans [53, 54] et aussi les codes concaténés en série présentés dans [69].

Chapitre 4

Application des codes à la cryptographie : Cryptosystème de McEliece

4.1 Introduction

En 1994, Peter Shor de Bell Laboratories a montré dans ces travaux [70, 71] que les ordinateurs quantiques, rend vulnérable les systèmes de chiffrement à clé publique les plus populaires tels que RSA [72] ou DSA [73], dont leur sécurité est basée sur les problèmes difficiles de la théorie des nombres (factorisation, logarithme discret,... etc). La cryptographie basée sur la difficulté de décoder un code linéaire aléatoire, en revanche, résiste aux attaques quantiques et donc considérée comme un substitut viable de ces systèmes dans les applications futures. Pour cette raison, nous nous intéressons aux algorithmes cryptographiques post-quantiques qui résistent aux attaques quantiques connues.
En 1978, Robert J. McEliece [21] a inventé le premier système de chiffrement postquantique basé sur la théorie des codes correcteurs d'erreurs. Sa sécurité est basée sur deux problèmes difficiles : le problème du décodage des codes linéaires aléatoires (qui est prouvé NP-complet [10]) et l'indistinguabilité des codes de Goppa des codes aléatoires. Niederreiter [74], en 1986, a introduit un nouveau schéma basé sur les codes de Reed-Solomon généralisés (GRS) en utilisant une matrice de contrôle de parité. Les deux systèmes ont une sécurité équivalente [75].
Le principal inconvénient d'un cryptosystème de McEliece réside dans la taille de la clé publique qui est coûteuse à stocker a cause de la taille de la matrice quelle a constitué. C'est l'une des principales raisons pour lesquelles les cryptosystèmes de McEliece ne sont plus utilisés en pratique.
Pour réduire la taille de la clé publique, plusieurs travaux sont menés en utilisant la famille des codes LDPC. Ils ont été suggérés à plusieurs reprises par le schéma de McEliece [76],[77],[78]. L'inconvénient de l'utilisation des codes LDPC réside dans le fait que le poids du mot de code est faible ce qui peut constituer un point d'attaques des cryptosystème de McEliece. Pour surmonter ce problème,en 2013, l'utilisation de codes quasi-cycliques MDPC (Moderate Density Parity Check) a été suggérée pour être adopter par le schéma de McEliece [79]. Cette version bénéficie de tailles de clé relativement petites (quelques milliers de bits). La sécurité de cette instance réside dans la difficulté de décryptage d'un mot de code de petit poids, de plus le décryptage consiste essentiellement en un décodage réalisé avec les mêmes algorithmes itératifs que pour les codes LDPC. En particulier, une implémentation à faible coût, adaptée aux systèmes embarqués, peut être réalisée en uti-

lisant un algorithme itératif appelé bit flipping, comme a été démontré dans [80]. D'autres travaux [80], [81]et [82] se sont concentrés sur la façon de choisir le seuil afin de réduire le nombre moyen d'itérations pour réussir à décoder un nombre donné d'erreurs en utilisant l'algorithme de décodage par retournement de bits. Malheureusement, ces codes ont été attaqués. Pour lutter contre ces attaques, [83][84] [85] ont utilisé une approche dans laquelle elle a déterminé une variante de l'algorithme Bit flipping. Si cette variante est combinée à une simulation intensive, elle peut fournir quelques directives pour concevoir le déchiffrement QC-MDPC-McEliece qui peut résister aux attaques de synchronisation. Dans ce chapitre, nous allons essayer d'introduire un cryptosystème à clef publique basé sur les codes QC-MDPC, nous expliquons dans un premier temps son fonctionnement en le comparant avec le cryptosystème de McEliece [21] classique basé sur le code de Goppa.

4.2 Cryptosystème de McEliece

En 1978, Robert McEliece eut l'idée d'utiliser la théorie des codes correcteurs d'erreurs pour un algorithme de chiffrement asymétrique. L'idée du schéma de McEliece est d'utiliser un code correcteur dont la structure est masquée. En envoyant un message contenant un grand nombre d'erreurs, que seul le destinataire sait détecter et corriger, rendant aussi le décodage de ce code difficile pour toute personne ne connaissant pas cette structure.
McEliece propose une fonction à sens unique qui code le message clair puis bruite le mot de code obtenu. Si le code est aléatoire et a des paramètres non triviaux, cette fonction est évaluable efficacement pour tout message. Donc, trouver une pré-image revient à décoder un code aléatoire en utilisant une trappe. Pour cela, McEliece propose d'utiliser la famille des codes de Goppa car sa matrice de parité est semblable à une matrice aléatoire pour une personne ignorant la structure algébrique du code.
Suite à cela, plusieurs familles de code ont été suggérées pour remplacer les codes de Goppa dans ce schéma : les codes de Reed-Solomon généralisés (GRS)[86], des codes de Reed-Muller [87], des codes algébriques géométriques [88], des codes LDPC [78], des codes MDPC [89] ou plus récemment des codes convolutifs [90].

4.2.1 Algorithme de McEliece

Comme tous les systèmes à clé publique, ce cryptosystème est constitué de trois algorithmes génération de clés, chiffrement et déchiffrement sont constitués de la façon suivante : [27]

Algorithm 3 Génération de clés

1: **Entrées** : Deux entiers n, k et t.
2: **Sorties** : La clé publique p_k = (G', t) et la clé privée associée s_k = (S, G, P, C)
3: Choisir un code linéaire C t-correcteur de dimension k et de longueur n
4: Prendre la matrice génératrice $G \in M_{k,n}(F_2)$ de C ;
5: Choisir aléatoirement une matrice inversible $S \in M_{k,k}(F_2)$;
6: Choisir aléatoirement une matrice de permutation $P \in M_{n,n}(F_2)$;
7: Calculer la matrice génératrice $G' = S.G.P$;
8: **Retourner** (p_k, s_k)

Algorithm 4 Chiffrement

1: **Entrées** : la clé publique p_k = (G', t) et un message à chiffrer $m \in F_2^k$.
2: **Sortie** : $c \in F_2^n$ le texte chiffré associé à m.
3: Encoder le message $c' = m.G'$;
4: Générer aléatoirement un vecteur erreur $e \in F_2^n$ de poids $w_H(e) = t$;
5: Calculer $c = c' \oplus e$;
6: **Retourner** c.

Algorithm 5

1: Déchiffrement
2: **Entrées** : La clé privée s_k = (S, G, P, C) et le texte chiffré $c \in F_2$.
3: **Sortie** : $m \in F_2^k$ le texte clair associé à c.
4: Calculer $c'_p = c.P^{-1}.$;
5: Décoder c'_p pour retrouver $m.S.G$;
6: Récupérer $m' = m.S$ à partir de $m.S.G$;
7: Calculer $m = m'.S^{-1}$.
8: **Retourner** m.

La figure suivante résume les trois algorithmes de système de McEliece lors de la transmission d'un message m d'Alice vers Bob.
La clé publique G' du système de McEliece n'est autre qu'une matrice génératrice d'un

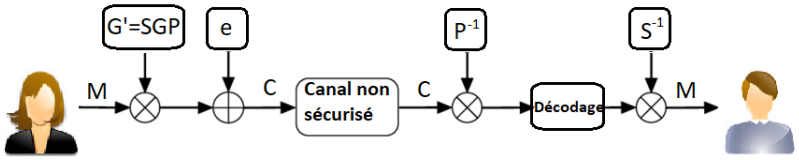


FIGURE 4.1 – Schéma de McEliece.

autre code linéaire C' équivalent au code C, qui ont la même distance minimale d. En effet, effectuer le produit $G' = S.G.P$ revient simplement à modifier la base choisie pour représenter le code sous forme d'une matrice génératrice (combinaison linéaire des lignes par la matrice S) et à permuter les coordonnées du code (permutations des colonnes par

la matrice P). Un message chiffré par le système, s'écrit donc sous la forme d'un mot du code public dont t positions sont erronées. Afin de retrouver le texte clair, ou de façon équivalente le vecteur d'erreurs e, on décode c relativement au code public jusqu'à $w = t$.

4.2.2 Construction des paramètres du code

Pour construire la clé publique G' de cryptosystème de McEliece de taille $nk\,log_2\,(q^m)$, on doit obligatoirement débuter par la construction de la matrice génératrice G qui dépend du code utilisé, puis la matrice de permutation P et la matrice inversible S. Pour ce qui ait les matrices S et P nous proposons les méthodes suivantes :

- Construction de la matrice S

Il s'agit ici de construire une matrice carrée de taille k inversible et aléatoire. La première idée consiste à générer une première matrice aléatoire. On regarde ensuite si elle est inversible. Si c'est le cas, on a trouvé une matrice S convenable, sinon on réitère le procédé jusqu'à la réussite. Cette méthode contribue à un gaspillage du temps et de l'espace mémoire lorsque le nombre de matrices que l'on va générer soit trop grand et surtout pour des valeurs de k élevées. Nous proposons la méthode[91] suivante comme solution :
- Générer aléatoirement une matrice triangulaire supérieur inversible $S1 \in F_2$.
- Générer aléatoirement une matrice triangulaire inférieur inversible $S2 \in F_2$.
- Calculer $S = S1.S2$.

- Construction de la matrice P

Construire la matrice P revient à générer une permutation de l'ensemble $\{1, 2, \ldots, n\}$. La première idée simpliste consisterait à engendrer un nombre aléatoire entre 1 et n. On le compare ensuite à ceux engendrés précédemment : s'il est distinct de tous, on passe à l'élément suivant, sinon on essaye avec un autre nombre. Il est clair que cette méthode devient rapidement très lente, donc on peut proposer l'algorithme suivant qui est plus efficace :
- On définit la liste LST par $LST(i) = i$ et r par $r = 0$;
- On génère un nombre aléatoire a entre 1 et n-r (r définit l'étape en cours) ;
- On échange les positions de LST(a) et LST(r+1) ;
- On incrémente r par 1. On recommence l'étape 2 jusqu'à ce que la permutation soit toute générée (i.e. $r = n - 1$).

4.2.3 Avantages et inconvénients de cryptosystème de McEliece

Le cryptosystème de McEliece est rarement utilisé en pratique car il est basé sur des codes correcteurs d'erreurs ont une structure tellement forte qu'il est impossible de la

masquer. Le cryptosystème de McEliece a été introduit initialement en utilisant les codes de Goppa qui, étant indistinguables de codes aléatoires, offrent une bonne résistance aux attaques structurelles. Cependant ces codes ont une capacité de correction très faible et nécessitent des clés publiques de taille très importantes pour être viables. Prenons ainsi l'exemple des codes de Reed-Solomon, Ils sont de bons candidats en théorie pour le cryptosystème de McEliece mais leur forte structure rend les cryptosystèmes très vulnérables [92] [93] [94].
Le tableau 4.1 résume quelques principaux avantages et inconvénients du cryptosystème de McEliece .

TABLE 4.1 – Avantages & Inconvénients de système de McEliece

Avantages	Inconvénients
▪ Implémentation facile. ▪ La sécurité croît beaucoup plus avec la taille des clés. ▪ Bon résistant aux cryptanalyses et bon candidat pour la cryptographie post-quantique. ▪ La rapidité du chiffrement et de déchiffrement.	▪ Consomme plus de mémoire. ▪ Rarement utilisé en pratique du fait de la grande taille des clés. ▪ Sensibilité aux erreurs (car le message chiffré est plus long que le message clair). ▪ Le cryptosystème n'est pas employé pour la signature ou l'authentification car il est asymétrique et l'algorithme de chiffrage n'est pas linéaire.

Pour réduire la taille de la clé publique, plusieurs travaux sont réalisés sur la base de la famille des codes LDPC (Low Density Parity Check)[95] et MDPC (Moderate Density Parity Check codes). ces codes semblent donc être une proposition très intéressante puisqu'ils sont très peu structurés et sont munis d'algorithmes de décodage très efficaces et performants. Cependant, ces codes possèdent de nombreux mots de poids faible.La seule différence étant que dans ce cas, la matrice de parité des codes LDPC considérée est très creuse (le poids des lignes étant constant en fonction de la longueur du code).ce qui facilite l'attaques du le cryptosystème de McEliece adoptant ces codes.pour remédier ce probleme Certaines propositions ont donc porté sur l'utilisation des codes quasi-cycliques puisque ceux-ci sont entièrement décrits grâce à une seule ligne d'une de leur matrice génératrice (ou de parité).

4.3 Sécurité

Le thème de la sécurité est de plus en plus intéresse les individus comme les entreprises. La sécurité du cryptosystème de McEliece (ou de Niederreiter) repose en premier lieu sur la difficulté qu'un adversaire soit capable de résoudre les problèmes de décodage avec les outils dont nous disposons aujourd'hui. Dans un second temps, il faut utiliser des conversions adaptées et éviter les codes ayant une faible structure.
McEliece proposa une attaque contre son système. Elle consiste à choisir aléatoirement

k bits du mot reçu x en espérant qu'il n'y ait pas d'erreurs sur ces bits-là. Soient x_k le vecteur uniquement formé par les k bits choisis et G'_k la matrice G' dont on ne garde que les k colonnes correspondants aux k bits choisis. S'il n'y a effectivement pas d'erreurs dans les k bits choisis, alors le véritable message est : m = $x_k.G'^{-1}_k$.
Il est possible de distinguer deux grandes catégories d'attaques contre le cryptosystème de McEliece : les attaques par décodage (ou attaques génériques) et les attaques structurelles (ou attaques algébriques).

4.3.1 Attaques génériques

Une attaque générique (par décodage) consiste à voir la matrice génératrice publique G' comme une matrice génératrice d'un code aléatoire et d'essayer de décoder une instance du cryptosystème en utilisant un algorithme général.

Exemple 4.1.
▪ *G' est par hypothèse de rang maximal. Donc on peut trouver k colonnes qui forment une famille libre. Autrement dit, on peut trouver P' une matrice de permutation telle que : G'.P' = (G1|G2) où G1 est une matrice inversible de taille k et G2 une matrice quelconque de taille (k,n-k).*
▪ *Les valeurs que peut prendre le vecteur d'erreur sont : $\sum_{i=0}^{t} C_n^i$. On a en général une approximation suffisante de ce nombre en prenant la valeur C_n^t.*

Algorithm 6

1: **Entrée :** la clé G' et le message chiffré c'.
2: **Sortie :** le message clair m et le vecteur d'erreur e.
3: On dresse un tableau de tous les vecteurs d'erreur possibles et leur syndrome.
4: On calcule le syndrome du message c'= m.G'+ e , (S(c') = S(e) car m.G' est un mot de code) ;
5: On cherche dans le tableau le vecteur d'erreur correspondant, et on obtient le message corrigé c ;
6: On peut écrire c = m.G' = $m.(G1\,|\,G2).P'^{-1} = (m.G1.P'^{-1}|\ m.G2.P'^{-1}) = (C1\,|\,C2)$;
7: On trouve m = $C1.P'.G1^{-1}$;
8: **Retourner** le couple (m, e)

Cet algorithme est très efficace pour des codes de valeurs de n et de t suffisamment petites, ce qui est différent dans la pratique.
En prenant par exemple les valeurs suggérées par McEliece : n=1024 et t=50. Alors, le nombre de vecteurs d'erreur possibles est de l'ordre 10^{86}*. Donc, il est impossible de calculer tous les éléments du tableau car il est trop long et le stockage de toutes ces informations demanderait une mémoire d'un milliard de milliards de disques durs de 10 Go.*

Exemple 4.2. ***(Décodage par ensemble d'informations)*** *:*

C'est l'algorithme efficace qui possède une petite complexité. De manière générale, pour un [n, k]-code linéaire, un ensemble d'informations est un ensemble de taille k de positions

du code qui forment un espace vectoriel de dimension k.
Supposons que nous avons trouvé $I = \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$ un tel ensemble que son support est disjoint avec le support d'erreur. Notons $P \in F_2^n$ une matrice carrée de permutation qui, $\forall i \in I$, a une image appartient à l'ensemble $\{(n-k)+1, \ldots, n\}$. La matrice HP ainsi construite est ensuite mise sous forme systématique, en utilisant une matrice inversible $U \in F_2^r$.
Nous obtenons donc un système équivalent :

$$eH^T = s \ avec \ w_t(e) = t \ \Leftrightarrow \ eP(UHP)^T = sU^T \ avec \ w_t(eP) = t$$

Dans la suite, nous noterons

$$e' = eP, \ \ s' = sU^T \ et \ H' = UHP = [I_r \mid H'_R]$$

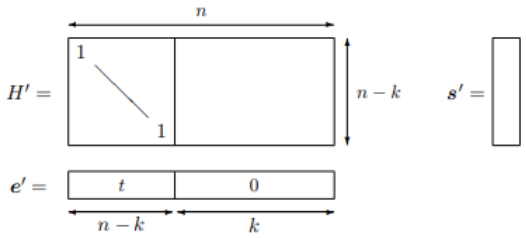


FIGURE 4.2 – Décodage par ensemble d'informations

Comme il est montré dans la figure ci-dessus, trouver une solution du système correspond à trouver $e' \in F_2^n$ de la forme $e' = (e'_L|e'_R) \in F_2^n$ avec $e'_L \in F_2^r$ et $w_t(e'_L) = t$, $e'_R \in F_2^k$ et $w_t(e'_R) = 0$, tel que $e'[I_r \mid H'_R]^T = e'_L = s'$
Ainsi, une fois que le choix de P et U est effectué, si $w_t(sU^T) = t$ alors $e = e'P^{-1}$ est une solution du système. En effet, nous avons alors trouvé $e' = (sU^T \mid 0 \ldots 0)$ avec $w_t(e') = t$ qui vérifie $e'H' = s'$ et comme la multiplication par une matrice de permutation ne change pas le poids d'un vecteur, $w_t(e) = t$ et e vérifie $eH^T = s$
L'algorithme suivant résume ce que nous venons de présenter, c'est le premier algorithme de décodage par ensemble d'informations proposé par Eugene Prange.

Algorithm 7

1: **Entrée :** $H \in F_2^{(n-k)\times n}$, $s \in F_2^{n-k}$ et $t \in N$.
2: **Sortie :** $e \in F_2^n$ tel que wt(e) = t et $eH^T = s$.
3: Choisir $P \in F_2^{n\times n}$ une matrice de permutation aléatoire.
4: Chercher une matrice inversible $U \in F_2^{(n-k)}$ telle que UHP soit sous forme systématique ;
5: **Tant Que** $w_t(sU^T) \neq t$ **Faire** :
Tirer une autre matrice de permutation P et en déduire la nouvelle valeur de U.
6: **Fin Tant Que** $e' \leftarrow (sU^T \mid 0 \ldots 0)$
7: **Retourner** $e'P^{-1}$

La complexité de cet algorithme dépend en partie de la probabilité de trouver une matrice inversible P telle que les k dernières colonnes de la matrice HP forment un ensemble

d'informations.

4.3.2 Attaques algébriques

Une attaque algébrique (ou structurelle) consiste à retrouver tout ou partie de la structure de la clé secrète à partir de la clé publique. Par exemple, l'adversaire cherche à estimer la matrice de permutation P à partir de la matrice G'.

Exemple 4.3. ***(Attaque par renvoi de message)*** *:*
A cause d'un accident, ou du fait de l'action d'un cryptanalyste, un message peut envoyé deux fois sous forme $c_1 = m.G + e_1$ *et* $c_2 = m.G + e_2$ *avec* $e_1 \neq e_2$*. Dans ce cas, il est facile pour un cryptanalyste de résoudre un système de* c_i *afin de retrouver m surtout dans le cas où* $i = 2$*, remarquons aussi que* $c_1 + c_2 = e_1 + e_2 (mod\, 2)$*.*

Pour détecter un renvoi de message, on observe le poids de Hamming de la somme de deux cryptogrammes. Quand les messages sont différents, le poids attendu de la somme est k (pour les paramètres originaux de McEliece, le poids attendu est d'environ 512). Alors que quand les deux messages sont identiques, le poids de la somme ne peut pas dépasser 2t (ou 100 pour les paramètres originaux de McEliece)

4.3.3 Attaques existantes

Comme déjà mentionné nous avons deux types d'attaques possibles et chaque type de son rôle contient différentes méthodes d'attaque qui dépendent de la famille de codes choisie pour le schéma.

TABLE 4.2 – Comparaison des familles de codes proposés avec le chiffrement de McEliece

Code	Paramètres	Clé	Sécurité	Attaques
Code de Goppa Binaires McEliece 1978.	$[1024, 524, 101]_2$	67 Ko	2^{62}	
	$[2048, 1608, 48]_2$	412 Ko	2^{96}	
Code RSG Niederreiter 1986.	$[256, 128, 129]_2$	67 Ko	2^{95}	Attaques en θ^3
Code de Reed Muller Binaires Sidelnikov 1994.	$[1024, 176, 128]_2$	22.5 Ko	2^{72}	Attaques sous exponentielles
	$[2048, 232, 256]_2$	59.4 Ko	2^{93}	
Codes Géométriques Janwa Moreno 1996	$[171, 109, 61]_2$	16 Ko	2^{66}	Attaques polynomiales

La table 4.2 décrit quelques codes utilisés avec le cryptosystème de McEliece et les

attaques qu'ils ont subies. La première famille de codes proposée par McEliece, est celle des codes de Goppa binaires classiques qui reste actuellement un bon résistant aux attaques. Après, plusieurs tentatives ont été faites pour surmonter les inconvénients du système d'origine. En 1986, Niederreiter propose son cryptosystème équivalent au schéma de McEliece, il a travaillé sur les codes de Reed Solomon généralisés (GRS). Cette famille de codes n'est pas résistante aux attaques structurelles puisque en 1992 Sidelnikov et Shestakov ont proposé un algorithme polynomial qui permet de retrouver la structure des codes GRS. En 1994, Sidelnikov a crée un schéma utilisant les codes de Reed-Muller pour une clé publique de taille relativement faible, mais cette famille a aussi été cassé par une attaque algébrique introduit par Minder et Shokrollahi [96] . En 1996, Janwa et Moreno ont proposé un schéma basé sur les codes algébriques géométriques (AG). Ce schéma a été complètement attaqué par Couvreur, Marquez-Corbella et Pellikaan[97]. Faugère et al. [98] ont cassé presque tous les systèmes basés sur des structures quasi cycliques ou quasi-dyadiques (à l'exception du cas binaire de [99]), Landais et Tillich [100] ont réalisé une attaque efficace sur le McEliece PKC basé sur les codes convolutionnels [90], et très récemment Bardet et al. [101] ont réussi à briser la variante de McElice basée sur des codes polaires [102].
Aujourd'hui, seuls deux systèmes sont sécurisés (à notre connaissance) : celui basé sur les codes Goppa [21] et celui basé sur les codes QC-MDPC [89].
par la suite on va décrit l'application de McEliece sur une nouvelle famille des codes MDPC quasi-cycliques.

4.4 Codes MDPC / QC-MDPC

4.4.1 Généralités sur les codes quasi-cycliques

Définition 4.1. ***(Code quasi-cyclique)***
Soit n un entier positif non nul et l un diviseur de n différent de n.
Un $[n, k, d]_q$ code linéaire C est dit quasi-cyclique d'indice l (l-quasi-cyclique) si et seulement s'il est stable par la puissance l du shift T, c'est à dire : pour tout
$v = (v_0, v_1, \ldots, v_{n-1}) \in C$, *on a* $v' = (v_{n-l}, v_{n-l+1}, \ldots, v_{n-1}, v_0, v_1, \ldots, v_{n-l-1}) \in C$.

Définition 4.2. ***(Matrice circulante)***
Une matrice carrée est dite circulante si elle est construite de sorte que ses lignes soient les décalages cycliques successifs de la première ligne.

Remarque 4.1.
• *Une matrice circulante est entièrement définie par une seule de ses lignes. Dans la suite, une matrice circulante $A \in F_2^{n\times n}$ sera notée $A = C(a_1, a_2, \ldots, a_n)$ où $(a_1, a_2, \ldots, a_n) \in F_2^n$ présente par convention la première ligne de A.*
• *L'ensemble des matrices circulantes de $F_2^{n\times n}$ sera noté $Circ(F_2^n)$.*
$A = C(a_1, a_2, \ldots, a_n) \in Circ(F_2^n)$ si, et seulement si, sa matrice est de la forme suivante :

$$\begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ a_n & a_1 & a_2 & \dots & a_{n-1} \\ a_{n-1} & a_n & a_1 & \dots & a_{n-2} \\ & & & \ddots & \\ a_2 & a_3 & a_4 & \dots & a_1 \end{pmatrix}$$

Définition 4.3. ***(Matrice quasi-circulante) :*** *Une matrice quasi-circulante est une matrice formée de blocs circulants*

Exemple 4.4. *la matrice suivante est quasi-circulante car elle est composée de deux blocs circulants :*

$$\left(\begin{array}{ccccc|ccccc} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{array}\right)$$

Proposition 4.1. *Un code quasi-cyclique admet au moins une matrice de parité quasi-circulante.*

Définition 4.4. ***(Ordre et indice)*** *:*
L'ordre p d'une matrice quasi-circulante est la taille des blocs circulants qui la composent. L'indice d'une matrice quasi-circulante de dimension $n \times k$ et d'ordre p désigne le rapport $\frac{n}{p}$.

En 1962, Gallager a inventé les codes LDPC. Ces codes sont basés sur des matrices de contrôle de parité pseudo aléatoires de faible densité. A cause de leur complexité d'encodage, de décodage et des moyens matériels de l'époque, ils ont été négligés. En 1995, David J.C. MacKay redécouvre ces codes et montre tout d'abord que l'algorithme de décodage développé par Gallager peut également se décrire comme l'algorithme de propagation de croyance (Belief Propagation) de Pearl. Par la suite, la communauté scientiflque a été motivée de poursuivre les travaux dans le domaine des codes LDPC. En 2004, le code LDPC a été pour la première fois normalisé dans un contexte de diffusion par satellite : DVB-S2. Plus récemment, les codes LDPC ont été introduits dans les standards IEEE 802.16e (Wimax mobile) et IEEE 802.11n (Wifl). Ces différents facteurs, ainsi que les nouveaux problèmes posées, ont contribué à accroitre la popularité des codes LDPC dans le domaine industriel et scientiflque. [103]
En 2008, dans l'article [78], les auteurs Baldi, Barreto et Chiaraluce proposent une nouvelle variante du cryptosystème de McEliece avec des codes LDPC quasi cycliques (QC-LDPC). Ils sont une classe particulière de codes LDPC, capables d'associer un codage peu complexe de codes quasi-cycliques à des techniques de décodage hautement performantes basées sur le principe de propagation de croyance. Afin d'augmenter la capacité de correction, les auteurs de [89] suggèrent en 2012 l'utilisation des codes MDPC (Moderate Density Parity Check codes) quasi-cycliques dont la structure est quasiment la même que celle des codes QC-LDPC dans le cryptosystème de McEliece.

4.4.1.1 Définitions

Définition 4.5. ***(Codes MDPC/QC-MDPC)***
Un code (n, r, w)-MDPC est un code linéaire de longueur n, de codimension r qui a une matrice de contrôle de parité dont les lignes ont un poids fixe w, où $w = O(\sqrt{n})$.
Lorsque ces codes sont également quasi-cycliques, ils sont appelés codes [n, r, w]-QC-MDPC.

Proposition 4.2. *un code [n, r, w]-QC-MDPC admet une matrice de parité de la forme suivante :*

$$\begin{pmatrix} H_{1,1} & H_{1,2} & \dots & H_{1,n_0} \\ H_{2,1} & H_{2,2} & \dots & H_{2,n_0} \\ & & \vdots & \\ H_{r_0,1} & H_{r_0,2} & \dots & H_{r_0,n_0} \end{pmatrix}$$

où, pour $1 \leq i \leq r_0$ *et* $1 \leq j \leq n_0$, $H_{i,j} = C(h_{i,j}) \in Circ\, F_2^{r/r_0}$ *et* $\sum_{j=1}^{n_0} w_t(h_{i,j}) = w$

Définition 4.6. ***(Représentation de la matrice de parité sous forme polynomiale)*** *:*

Soient $H = [C(h_L) \mid C(h_R)] \in F_2^{r \times n}$ *et C un code [n, r, w]-QC-MDPC.*

Soient $(h_L(X), h_R(X)) \in (F_2[X]/(X^r - 1))^2$ *tels que* $h_L(X) = \varphi(h_L)$ *et* $h_R(X) = \varphi(h_R)$.

H est une matrice de parité de $C \Leftrightarrow (h_L(X), h_R(X))$ *est une description de C.*
Tel que φ *est définit comme l'isomorphisme suivant :*

$$\varphi : \ Circ(F_2^r) \to F_2[X]/(X^r - 1)$$
$$C(a_1 \ a_2 \ \dots \ a_r) \to a_1 + a_2 X + \dots + a_r X^{r-1}$$

4.4.2 Décodage des codes MDPC

Les algorithmes de décodage pour les codes MDPC sont principalement divisés en deux familles. La première classe proposée par Berlekamp et Al en 1978 offre une meilleure capacité de correction d'erreur mais elle est plus complexe en calcul que la deuxième famille. En particulier lors du traitement de gros codes, la deuxième famille, appelée algorithmes de bit flipping introduit par Gallager en 1962, semble être plus approprié [82]. L'idée générale de cet algorithme de bit flipping de Gallager est la suivante. Étant donnés $y \in F_2^n$ un mot de code bruité et C un code [2r, r, 2d]-MDPC dont une matrice de parité creuse est $H \in F_2^{r \times n}$, nous allons utiliser $s = y.H^T \in F_2^r$ pour identifier les positions du vecteur y qui ne vérifient pas les équations de parité de H. Rappelons que, si une coordonnée du syndrome, disons i pour $1 \leq i \leq r$, est non nulle, c'est que $< y, h_i >= 1$. Autrement dit, y ne vérifie pas la i-ème équation de parité définie par H. Ceci se produit, si et seulement si, un nombre impair de coordonnées du vecteur y qui interviennent dans

cette équation sont erronées. La spécificité des codes LDPC et MDPC réside dans le fait que ces équations de parité ont un poids très faible (ou modéré). En utilisant le support de ces équations de parité, nous pouvons déterminer les positions de y susceptibles d'être fausses.

4.4.2.1 Convention

▪ Pour v un vecteur binaire de longueur quelconque, nous noterons $j \in v \Leftrightarrow v_j = 1$. Autrement dit,

$$pour\ \ v \in F_2^n, v = j : 1 \leq j \leq n\ \ et\ \ v_j = 1 \tag{4.1}$$

▪ $|v|$: le poids de Hamming de v.
▪ Pour v et v' deux vecteurs binaires de F_2^n, $v \cap v'$ est le AND binaire usuel entre ces deux vecteurs. Autrement dit,

$$v \cap v' = j : 1 \leq j \leq n, j \in v\ \ et\ \ j \in v' \tag{4.2}$$

▪ Soit $H = (h_{i,j})_{1 \leq i \leq r, 1 \leq j \leq n} \in F_2^{r \times n}$ une matrice de parité d'un code [2r, r, 2d]-MDPC , nous désignerons par h_i pour $1 \leq i \leq r$ les équations de parité définies par cette matrice. Par convention, nous supposerons que :

$$h_i = (h_{i,1}, h_{i,2}, \ldots, h_{i,n}) \tag{4.3}$$

▪ De même, nous noterons $h_{*,j} \in F_2^r$, pour $1 \leq i \leq r$, la transposée de la j-ème colonne de H. Autrement dit,

$$h_{*,j} = (h_{1,j}, \ldots, h_{r,j}) \tag{4.4}$$

▪ Pour $e \in F_2^n$ et $H \in F_{r \times n}$ qui définie $h_i : 1 \leq i \leq r\ et\ |h_i| = w$ un ensemble d'équations de parité, nous posons pour $1 \leq j \leq n$,

$$\sigma_j = |\{i : 1 \leq i \leq r, j \in h_i\ \ et\ \ < h_i, e >= 1\}|$$

Soit $s = eH^T$ alors

$$\sigma_j = |h_{*,j} \cap s| \tag{4.5}$$

σ_j correspond au nombre d'équations de parité insatisfaite faisant intervenir la j-ème position de l'erreur. Autrement dit,

$$\sigma_j = \sum_{i:j \in h_i} < h_i, e > \tag{4.6}$$

D'un autre côté, s = $(s_i)\quad 1 \leq i \leq r$ où $s_i =< h_i, e >$.

4.4.2.2 Algorithme de Bit Flipping

Les algorithme de bit flipping sont des algorithmes itératifs et probabilistes, puisque ils sont constitués d'une série d'opérations qui se répètent plusieurs fois afin de retourner la bonne solution dans une certaine probabilité. Ils sont basés sur le principe suivant :

1 : Calculer le syndrome du texte chiffré reçu s = xH^T.
2 : Compter le nombre d'équations de contrôle de parité non satisfaites associées à chaque bit de texte chiffré.
3 : Inverser chaque bit de texte chiffré qui provoque plus de b équations de parité non satisfaites.
4 : Mettre à jour le calcul de syndrome.
Ce processus est répété jusqu'à ce que le syndrome devient nul ou qu'un nombre maximum prédéfini d'itérations sur lequel une erreur de décodage soit renvoyée soit atteint.

La différence entre les algorithmes de bit flipping réside dans la manière dont ils déterminent le seuil b. L'estimation de la capacité de correction d'erreur des codes MDPC est influencée par le choix du seuil b. Par conséquent, plusieurs algorithmes de bit flipping, évaluent leur capacité de correction d'erreur et comptent le nombre d'itérations nécessaires en moyenne pour décoder un mot de code de sorte que la probabilité d'échec soit négligeable. Gallager en 1962, pré-calcule les seuils b_i pour chaque itération i. Huffman et Pless en 2010, fixent le seuil au nombre maximum des équations de contrôle de parité non satisfaites b = max (σ_j) et Misoczki et al en 2013 proposent d'utiliser b = max $(\sigma_j) - \delta$, pour certains petits δ afin d'accélérer le décodage. Si le décodage échoue après un nombre maximal d'itérations défini, δ est réduit à $\delta - 1$ et le processus de décodage est redémarré. Ceci est répété jusqu'à ce que $\delta = 0$ où le décodeur devient comme celui proposé par Huffman et Pless.
Il existe un autre type de décodeur qui calcule le syndrome, puis vérifie le nombre d'équations de parité non satisfaites et si un bit de mot de code j est inversé, la ligne correspondante h_j de la matrice de contrôle de parité est directement ajoutée au syndrome courant. Cette méthode permet de travailler toujours avec un syndrome à jour et non avec celui de la dernière itération. En cas d'erreur de décodage, δ est décrémenté comme dans le décodeur de Misoczki et Al [82].

Algorithm 8 Exemple d'algorithme de bit flipping

1: **Entrées** : $H \in F_2^{r \times n}$ une matrice de parité sous sa représentation creuse de C, un code [2r, r, 2d]-MDPC, dont les équations de parité seront notées h_i pour $1 \leq i \leq r$ et $y \in F^n$
2: **Sortie** : le mot y décode.
3: **Traitement**
s = $y.H^T$
while $|s| \neq 0$ **do**
for $j \in 1, \ldots, n$ **do**
if $|h_{*,j} \cap s| > b$ **then**
$y_j \leftarrow y_j \oplus 1$
end if
end for
s = $y.H^T$
end while
4: **Retourner** y.

4.5 Cryptosystème de McEliece sur des Codes MDPC/ QC-MDPC

4.5.1 Introduction

Le cryptosystème de McEliece est plus rapide que les solutions concurrentes, telles que RSA, qui fait partie des algorithmes de la clé publique les plus répandus actuellement. Malgré cela, le cryptosystème de McEliece a rarement été pris en compte dans les applications pratiques car ils nécessitent des clés de taille très importante, et selon le code choisi, il peut être vulnérables aux attaques structurelles.
Récemment, une nouvelle famille de codes appelés codes MDPC (Moderate Density Parity-Check) est apparue. Les codes MDPC sont des codes LDPC (Low Density Parity Check) de densité supérieure et de capacité de correction d'erreur plus grande. Elle est résistante aux attaques structurelles. Elle utilise des matrices quasi-cycliques (QC-MDPC) afin d'augmenter le débit de transmission et d'obtenir des clés de taille compacte.
Dans ce chapitre nous allons présenter la version du cryptosystème de McEleice instancié avec les codes QC-MDPC et sa sécurité.

4.5.2 Choix de paramètres des codes QC-MDPC

Afin d'aboutir le niveau de sécurité souhaité, il faut bien choisir les paramètres des codes QC-MDPC en prenant en compte les faiblesses structurelles de certains jeux de paramètres.

Plus récemment, les codes quasi-cycliques dont la taille des blocs est paire ou plus généralement, n'est pas un nombre premier facilitent la résolution du décodage générique. Pour éviter ce type d'attaque, nous choisissons des matrices ayant des blocs dont la taille est un nombre premier. De plus, l'emploi de codes quasi-cycliques d'indice 2 et de taux de transmission $\frac{1}{2}$ permet de simplifier la mise en oeuvre des cryptosystèmes proposés et l'analyse de leur sécurité.

D'autre part, les algorithmes de générations de clés que nous allons présenter par la suite nécessitent le choix d'une matrice circulante inversible dont les lignes de poids impair d. Or, si d est pair, une telle matrice n'est pas inversible.
Enfin, nous restreignons notre choix à celui des matrices quasi-circulantes régulières, c'est-à-dire dont les blocs ont des lignes de même poids. Nous fixerons w = 2d (car la matrice contient deux blocs), où w est le poids des lignes de la matrice de parité .

Donc par la suite, nous considérerons des codes [n, r, w]-QC-MDPC tels que :
- r soit premier,
- n = 2r,
- w = 2d avec d impair et $w = O(\sqrt{n})$.

Nous noterons R = $F_2[X]/(X^r - 1)$

4.5.3 algorithmes McEliece basé sur QC-MDPC

Algorithm 9 Génération de clés

1: **Entrées** : C un code [2r, r, 2d]-QC-MDPC capable de corriger jusqu'à t erreurs.
2: **Sorties** : $p_k = h \in R$ est la clé publique et $s_k = (h_L, h_R) \in R^2$ est la clé privée associée.
3: Choisir aléatoirement $h_L, h_R \in R$ avec $w_t(h_L) = w_t(h_R) = d$ et tel que h_L soit inversible dans R.
4: Calculer h = $h_R.h_L^{-1}$
5: **Retourner** (p_k, s_k)

Algorithm 10 Chiffrement

1: **Entrées** : La clé publique p_k = h et un message à chiffrer $m \in R$.
2: **Sortie** : $c = (c_L, c_R) \in R^2$ le texte chiffré associé à m.
3: Choisir aléatoirement $e = (e_L, e_R) \in R^2$
4: Calculer $c = (c_L, c_R) = (m.h + e_L, m + e_R)$;
5: **Retourner** c

Algorithm 11 Déchiffrement

1: **Entrées** : la clé privée $s_k = (h_L, h_R)$ et le texte chiffré $c \in R^2$;
2: **Sortie** : $m \in R$ le texte clair associé à c ;
3: Retrouver $(e_L, e_R) \in R^2$ tel que $c_L.h_L + c_R.h_R = e_L.h_L + e_R.h_R$ et $w_t(e_L) + w_t(e_R) \leq t$ en utilisant un algorithme de décodage ;
4: Déduire le message m à partir de $(c_L - e_L, c_R - e_R) = (mh, m)$.
5: **Retourner** m

Remarque 4.2. *La donnée de* $(h_L, h_R) \in H_{QC}(n, r, w)$ *est équivalente à la donnée de* $(h_L^T, h_R^T) \in H_{QC}(n, r, w)$.

Comme nous avons vu précédemment la description d'un code C [2r, r, 2d]- QC-MDPC est donnée grâce à $(h_L, h_R) \in H_{QC}(n, r, w)$ tel que $(h_L^T, h_R^T) \in H_{QC}(n, r, w)$ soit une matrice de parité du code C dans lequel on peut déduire sa forme systématique $(1, h_R^T(h_L^{-1})^T) \in R^2$. Ainsi, nous obtenons $(h_R^T((h_L^{-1})^T)^T, 1) = (h_R h_L^{-1}, 1) \in R^2$ une représentation d'une matrice génératrice de C sous forme systématique .
La première étape dans le déchiffrement du cryptosystème de McEliece est de trouver un algorithme de décodage capable de corriger t erreurs grâce à la donnée $(h_L, h_R) \in H_{QC}(n, r, w)$.

4.5.4 Sécurité

La sécurité du cryptosystème de McEliece repose : d'une part, sur la difficulté de résoudre le problème du décodage générique et d'autre part, sur la difficulté de la résolution du problème de distinguabilité de la clé publique.

4.5.4.1 Sécurité du message

Actuellement, les algorithmes de décodage par ensemble d'informations sont les meilleurs algorithmes pour résoudre le problème du décodage générique. On notera par WF_{ISD} (pour Work Factor of the Information Set Decoding) la complexité de l'algorithme le plus efficace. Cependant, dans notre contexte, nous nous intéressons par la difficulté de résoudre le problème de décodage générique dans un code quasi-cyclique d'indice 2 et de taux de transmission 1/2.

Problème 1 : Décodage générique dans un code quasi-cyclique
Paramètres : $r, t \in N$ et $R = F_2[X]/(X^r - 1)$
Instance : $h, s \in R$
Question : Trouver $(e_L, e_R) \in R^2$ tel que

$$e_L + e_R h = s \ \ et \ \ w_t(e_L) + w_t(e_R) \leq t$$

Pour les codes quasi-cycliques, la donnée d'une instance du problème 1 permet de construire r instances ayant la même solution, c'est-à-dire si on considère le décalage cyclique à droite d'une instance, on va former une nouvelle instance ayant la même solution . En effet, pour $r, t \in N$ et $R = F_2[X]/(X^r - 1)$ fixés, la donnée de $(h, s) \in R^2$ permet de construire les instances équivalentes dont l'entrée est $(h, s.X_j) \in R^2$ pour $1 \leq j \leq r - 1$.
En effet chercher $(e_L, e_R) \in R^2$ comme solution du problème 1 est équivalent à chercher $(e'_L, e'_R) \in R^2$ comme solution du problème suivant :

$$e'_L + e'_R h = s.X^j \ \ et \ \ w_t(e'_L) + w_t(e'_R) \leq t \tag{4.7}$$

Si un tel $(e_L, e_R) \in R^2$ existe alors $(e'_L, e'_R) = (e_L.X_j, e_R.X_j)$ est une solution du problème équivalent mentionné ci-dessus.
Dans ce cas, la complexité du décodage générique lorsque le cryptosystème de McEliece est instancié grâce aux codes [2r, r, 2d]-QC-MDPC et un poids d'erreur t est la complexité de l'algorithme divisée par la racine du nombre d'instances disponibles du problème.

$$\frac{WF_{ISD}(2r, r, t)}{\sqrt{r}} \tag{4.8}$$

4.5.4.2 Sécurité de la clé

le problème de distinguabilité d'un code [2r, r, 2d]-QC-MDPC revient à décider de l'existence d'un mot de poids w dans le dual du code défini par la matrice publique. Pour retrouver la clé privée à partir de la clé publique, il faut résoudre le problème suivant.

Problème 2 : (Distinguabilité d'un code QC-MDPC - Recherche).
Paramètres : $r, w \in N \ et \ R = F_2[X]/(X^r - 1)$
Instance : $h \in R$
Question : Trouver $(h_L, h_R) \in R^2$ tels que

$$h_L h + h_R = 0 \ et \ w_t(h_L) = w_t(h_R) = d \tag{4.9}$$

Pour retrouver un mot du dual de poids w d'un code [2r, r, 2d]-QC-MDPC, c'est-à-dire pour résoudre le problème 2, la meilleure stratégie est d'appliquer un algorithme de décodage par ensemble d'informations ayant pour entrée un syndrome nul et une matrice génératrice de ce code avec w comme borne sur le poids de la solution à chercher. La seule différence avec un code quelconque est que, dans le cas des codes MDPC, nous savons que ce problème a r solutions (qui forment une matrice de parité du code). La probabilité de trouver un mot de poids faible dans ces codes est donc r fois celle de trouver un tel mot dans un code aléatoire. Sa complexité est donc

$$\frac{WF_{ISD}(2r, r, 2d)}{r} \tag{4.10}$$

4.5.4.3 paramètres proposés

Les auteurs de [79] proposent, à titre indicatif, une série de paramètres en fonction du niveau de sécurité souhaitée comme illustré dans le tableau suivant :

TABLE 4.3 – Choix des paramètres des codes [2r, r, w]-QC-MDPC pour le cryptosystème de McEliece avec un poids d'erreur initial t

Sécurité	r	w	t	Clé publique	Clé privéé	Clair	Chiffré
80	4801	90	84	4801	9602	4801	9602
128	9857	142	134	9857	19714	9857	19714
256	32771	274	264	32771	65542	32771	65542

Comme nous considérons des codes de taux de transmissions R = 1/2, la codimension d'un code [n, r, w]-QC-MDPC est égale à sa dimension. Autrement dit, k = r = n/2 . Dans ce cas,

$$WF_{ISD}(2r, k, t) = WF_{ISD}(2r, r, t) = 2^{ct}(1 + o(1)) avec \ c = log_2(\frac{1}{1-R}) = 1$$

et

$$WF_{ISD}(2r, r, w) = 2^{cw}(1 + o(1)) \tag{4.11}$$

fournissent une bonne approximation de la complexité des algorithmes de décodage par ensemble d'informations. Ainsi,

- la sécurité de la clé est donnée par $\frac{2^w(1+ro(1))}{r}$,
- la sécurité du message est égale à $\frac{2^t(1+\sqrt{r}o(1))}{\sqrt{r}}$

4.5.5 Resultats et simulations

Dans cette partie nous avons mis en œuvre l'algorithme de cryptage de McEliece classique basé sur les codes de Goppa et celui des codes QC-MDPC. En utilisant des fichiers de taille 1Ko, 2Ko, 3Ko, 4Ko et 5Ko nous a amené à ces résultats illustré par ces deux figures

4.5.5.1 Chiffrement

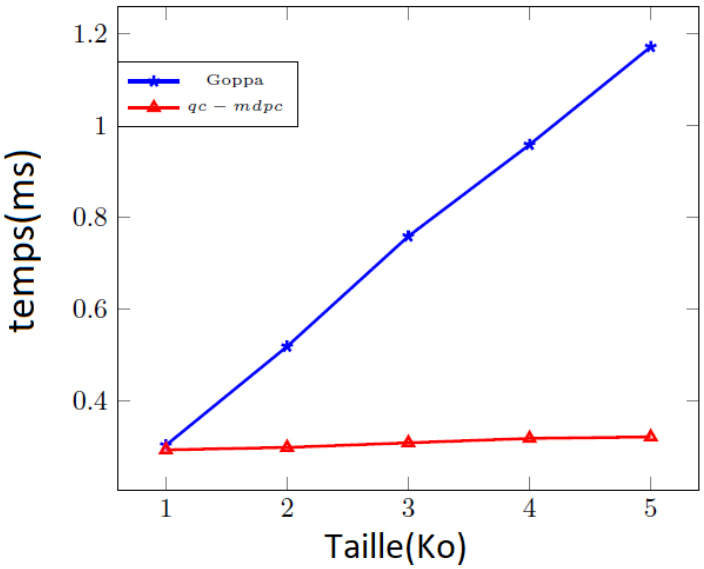


FIGURE 4.3 – temps de chiffrement des fichiers en fonction de la taille

On voit clairement que le chiffrement de McEliece par les deux codes est presque avec un temps linéaire égal ce fait est due à la même procédure de cryptage qui est la multiplication de message par la matrice génératrice de code.

4.5.5.2 Déchiffrement

Pour le déchiffrement, on observe que le code de Goppa prend un temps de plus que le code de QC-MDPC, donc on peut considérer le déchiffrement de McEliece avec le code de QC-MDPC est meilleur que le code de Goppa car le premier rapide et sécurisé, alors que le deuxième est lente vue son algorithme de décodage.
L'avantage des cryptosystèmes basés sur QC-MDPC est que le chiffrement et le déchiffrement sont plus rapides ce qui réduit la taille de la clé publique. Dans la perspective de

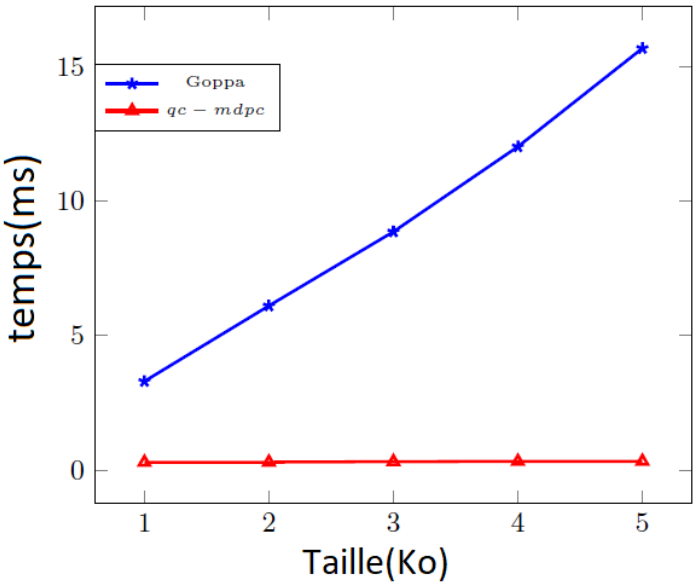


FIGURE 4.4 – temps de déchiffrement des fichiers en fonction de la taille

ce travail, nous allons améliorer l'algorithme de décodage de bit flipping en cherchant à réduire le taux d'échec du décodage.

4.6 Conclusion

Dans ce chapitre nous avons exploré les propriétés du cryptosystème McEliece. Nous avons commencé par présenter les outils nécessaires à la mise en œuvre d'un cryptosystème McEliece, son avantage principal est la rapidité de l'opération de chiffrement et de déchiffrement, mais il n'est pas encore utilisé dans la pratique car il nécessite la generation de clés de taille importantes.
Nous avons menés une comparaison en terme de temps de chiffrement et déchiffrement sur des fichiers de tailles différentes tailles entre les deux cryptosysteme de McEliece, l'instance basé sur les codes de Goppa et l'instance basé sur les codes QC-MDPC.
l'utilisation des codes de Goppa en conjointement avec l'algorithme d'Euclide étendu en tant que mécanisme de décodage pour le déchiffrement semble être un bon choix, car les codes de Goppa sont relativement faciles à comprendre et à construire, et l'algorithme de décodage s'adapte assez bien malgré la taille importante de la clé public . Le cryptosystème basé sur les codes QC-MDPC est performant par rapport à celui basé sur les codes de Goppa, d'une part il est sécurisé contre les attaques basées sur les matrices creuses, d'autre part, la taille de sa clef est réduite.
Nous concluons que le cryptosystème de McEliece instancié par les codes QC-MDPC peut être adopté par des organisme de la sécurité informatique car c'est un candidat efficace pour les cryprosystème post-quantique de l'avenir .

Conclusion et Perspectives

Cette thèse porte sur les codes correcteurs d'erreurs, et plus précisément sur les codes concaténés. Tout système de correction d'erreurs obtenu par la concaténation de codes simples, séparés par un entrelaceur est un turbo-code s'il est associé à un décodeur itératif. Les turbo-codes sont considérés actuellement parmi les meilleures solutions pour protéger l'information lors de la transmission ou le stockage sur des supports .
Nous avons présenter dans le premier chapitre le cadre théorique de cette thèse. Une partie aborde les notions de base de la communication numérique, puis les différents éléments de la chaîne de transmission à savoir l'émetteur, le récepteur, quelques types de canaux de communication et les performances de la modulation BPSK . La deuxième partie du chapitre aborde les notions de base des codes correcteurs d'erreurs et leurs classifications.
Dans la première contribution , nous avons présenté une nouvelle construction des codes en bloc concaténés en série et en parallèle généralisés. La première construction possède un taux de codage élevé. L'évaluation de performances montre qu'ils sont à 1.8 dB de la limite Shannon pour $TEB = 10^{-5}$. Nous avons aussi mis en œuvre l'algorithme de Chase-Pyndiah pour les codes en bloc concaténés en parallèle. D'après les résultats de simulation, on montre que ces codes sont à 2.1 dB de la limite de Shannon. Ils sont moins performants que les codes concaténés en série.
Dans la deuxième contribution, nous avons présenté le décodeur Dual-R-m pour les codes en bloc linéaires en résolvant un système d'équations. Dans un premier temps, on a abordé l'algorithme de base. Dans un deuxième temps, nous avons développé une version soft du décodeur Dual-R-m pour fournir une sortie pondérée. Cette dernière est utilisée pour développer un nouvel algorithme pour décoder les codes en blocs produits. Les résultats de simulation montrent que le décodeur proposé dépasse ses concurrents présentés dans la littérature. Dans ce contexte, notre décodeur dépasse celui de Pyndiah de 0.6 dB pour certain codes. De plus, le code BCH(511, 502, 3)2 est de 0.55dB de la limite de Shannon. Il opère près de la limite de Shannon.
Ensuite, la troisième contribution, est consacré à l'étude du cryptosystème à clé publique, introduit par Robert McEliece ,en 1978 , basé sur les codes de Goppa. Ce cryptosystème possède des propriétés intéressantes en terme de la sécurité et la rapidité de chiffrement et déchiffrement. Cependant, il n'est jamais utilisé en pratique du fait de la grande taille des clés, il a subi des attaques efficaces mais plusieurs améliorations ont été faites afin d'éviter ces attaques et réduire la taille de clé. Dans notre contribution on a fait une étude comparative entre le cryptosystème McEliece classique basé sur les codes de Goppa et celui basé sur les codes QC-MDPC, ces deux systèmes sont sécurisés et résiste aux attaques quantiques, et le deuxième est plus performant que l'autre et aussi par rapport aux schémas existants, les résultats de simulations qu'on a effectué montrent que le cryptosystème basé sur les QC-MDPC est plus performant que celui basé sur les codes de Goppa. De plus, les codes QC-MDPC minimise la taille de la clé publique.
Comme perspectives de notre travail, il serait intéressant d'appliquer les deux algorithmes proposés aux codes concaténés construits à base des codes hybride (série et parallèle). Il

sera intéressant aussi d'analyser le comportement de convergence des décodeurs itératif des cas étudiés en utilisant la technique dite EXIT chart.
Pour l'utilisation des codes correcteur en cryptographie, il serait important d'instancier le cryptosystème de McEliece en utilisant les autres familles de codes dont l'objectif d'augmenter le niveau de sécurité et/ou réduire la complexité du cryptosystème.

Bibliographie

[1] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948. 1, 7, 13, 84

[2] D. E. Muller, "Application of boolean algebra to switching circuit design and to error detection," *Transactions of the IRE professional group on electronic computers*, no. 3, pp. 6–12, 1954. 2

[3] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," vol. 4, no. 4, pp. 38–49. 2

[4] P. Elias, "Coding for two noisy channels," in *Information Theory, 3rd London Symposium, London, England, Sept. 1955*, 1955. 2

[5] P. Elias, "Error-free coding," vol. 4, pp. 29–39, 1954. 2

[6] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of products codes," *in Proc. GLOBECOM94 Conf.*, vol. 1/3, pp. 339–343, November 1994. 2, 3, 15, 71, 81

[7] E. Prange, *Cyclic error-correcting codes in two symbols*. Air force Cambridge research center, 1957. 2

[8] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffers*, vol. 2, pp. 147–156, 1959. 2, 21

[9] 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. 2, 21

[10] 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. 2

[11] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962. 2, 15

[12] F. G.D., "Concatenated codes," *Cambridge,MA :MIT Press*, 1966. 2

[13] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967. 2

[14] D. Chase, "Class of algorithms for decoding block codes with channel measurement information," *IEEE Transactions on Information theory*, vol. 18, no. 1, pp. 170–182, 1972. 2, 42, 49, 70, 71

[15] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding : Turbo-codes. 1," in *Proceedings of ICC'93-IEEE International Conference on Communications*, vol. 2, pp. 1064–1070, IEEE, 1993. 2, 36, 37, 41, 49, 70, 81

[16] D. J. MacKay and R. M. Neal, "Good codes based on very sparse matrices," in *IMA International Conference on Cryptography and Coding*, pp. 100–111, Springer, 1995. 3

[17] M. Belkasmi and A. Farchane, "Iterative decoding of parallel concatenated block codes," in *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pp. 230–235, IEEE, 2008. 3, 49

[18] A. Farchane and M. Belkasmi, "Generalized serially concatenated codes : construction and iterative decoding," *International Journal of Mathematical and Computer Sciences*, vol. 6, no. 2, 2010. 3, 36, 49, 50, 51

[19] A. Farchane and M. Belkasmi, "New efficient decoder for product and concatenated block codes," *Journal of Telecommunication*, vol. 12, pp. 17–22, 2012. 3, 49, 61

[20] A. Farchane and M. Belkasmi, "New decoder for linear block codes," in *2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, pp. 1–5, Oct 2016. 3, 4, 70, 71

[21] R. McEliece, "A public-key system based on algebraic coding theory, 114-116. deep sace network progress report, 44," *Jet Propulsion Laboratory, California Institute of Technology*, 1978. 3, 85, 86, 93

[22] A. Glavieux and M. Joindot, *Communications numériques : introduction.* Masson., 1996. 8, 11, 12

[23] J. G. Proakis and M. Salehi, *Digital communications*, vol. 4. McGraw-hill New York, 2001. 8

[24] C. Berrou, "Turbo codes ; general principles and applications," in *Proc 6^< th> Tirrenia Int. Workshop Digital Commun, Tirrenia, Italy, Sept. 1993.* 15

[25] G. Murat, "Résultants de polynômes de ore et cryptosystèmes de McEliece sur des codes rang faiblement structurés." 18

[26] H. Jaber, *Conception architecturale haut débit et sûre de fonctionnement pour les codes correcteurs d'erreurs*. PhD thesis, Université Paul Verlaine-Metz, 2009. 20

[27] T. Richmond, *Implantation sécurisée de protocoles cryptographiques basés sur les codes correcteurs d'erreurs*. PhD thesis, Université de Lyon, 2016. 20, 24, 27, 31, 86

[28] R. H. Morelos-Zaragoza, *The art of error correcting coding*. John Wiley & Sons, 2006. 24

[29] J. C. Moreira and P. G. Farrell, *Essentials of error-control coding*. John Wiley & Sons, 2006. 24

[30] G. Forney, "On decoding bch codes," *IEEE Transactions on information theory*, vol. 11, no. 4, pp. 549–557, 1965. 24

[31] R. B. Elwyn, "Algebraic coding theory," *McGraw-Hill, New York. MR*, vol. 38, p. 6873, 1968. 24

[32] J. Massey, "Shift-register synthesis and bch decoding," *IEEE Trans. Inform. Theory*, vol. 13, pp. 21–27, 1967. 24

[33] R. Gallager, "Low-density parity-check codes. ma," *Cambridge : M*, vol. 1, pp. 1–73, 1963. 33

[34] P. Elias, "Error free-coding," *IRE Trans. Information Theory*, vol. IT, pp. 29–37, September 1954. 34, 70

[35] C. Lee, *Convolutional coding : fundamentals and applications*. Artech House Communications Li, 1997. 36

[36] M. P. Fossorier and S. Lin, "Computationally efficient soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 738–750, 1996. 42

[37] C. Hartmann and L. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Transactions on Information Theory*, vol. 22, no. 5, pp. 514–517, 1976. 42

[38] O. Aitsab and R. Pyndiah, "Performance of concatenated reed-solomon/convolutional codes with iterative decoding," in *GLOBECOM 97. IEEE Global Telecommunications Conference. Conference Record*, vol. 2, pp. 934–938, IEEE, 1997. 44

[39] D. Gazelle and J. Snyders, "Reliability-based code-search algorithms for maximum-likelihood decoding of block codes," *IEEE Transactions on Information Theory*, vol. 43, no. 1, pp. 239–249, 1997. 44

[40] A. Kabat, F. Guilloud, and R. Pyndiah, "New approach to order statistics decoding of long linear block codes," in *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*, pp. 1467–1471, IEEE, 2007. 44

[41] R. Lucas, M. Bossert, and M. Breitbach, "On iterative soft-decision decoding of linear binary block codes and product codes," *IEEE Journal on selected areas in communications*, vol. 16, no. 2, pp. 276–296, 1998. 44

[42] M. Lahmer, M. Belkasmi, and F. Ayoub, "Iterative threshold decoding of one step majority logic decodable block codes," in *2007 IEEE International Symposium on Signal Processing and Information Technology*, pp. 668–673, IEEE, 2007. 44

[43] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes," in *1994 IEEE GLOBECOM. Communications : The Global Bridge*, pp. 339–343, IEEE, 1994. 44, 49, 54, 56, 57

[44] M. Belkasmi, M. Lahmer, and F. Ayoub, "Iterative threshold decoding of product codes constructed from majority logic decodable codes," in *2006 2nd International Conference on Information & Communication Technologies*, vol. 2, pp. 2376–2381, IEEE, 2006. 44

[45] M. Belkasmi and A. Farchane, "Iterative decoding of parallel concatenated block codes," in *2008 International Conference on Computer and Communication Engineering*, pp. 230–235, IEEE, 2008. 44, 61

[46] A. Farchane and M. Belkasmi, "Generalized serially concatenated codes : construction and iterative decoding," *International Journal of Mathematical and Computer Sciences*, vol. 6, no. 2, 2010. 49

[47] F. Ayoub, A. Farchane, A. Mohamed, M. Belkasmi, and M. M. Himmi, "Serially concatenated OSMLD codes : design and iterative decoding," *Applied Mathematical Sciences*, vol. 10, pp. 2179–2188, 2016. 49

[48] A. Farchane, M. Belkasmi, and N. Said, "Generalized parallel concatenated block codes based on BCH and RS codes, construction and Iterative decoding," *arXiv preprint arXiv :1303.4224*, 2013. 49

[49] S. Benedetto and G. Montorsi, "Serial concatenation of block and convolutional codes," *Electronics Letters*, vol. 32, no. 10, pp. 887–888, 1996. 49

[50] N. K. Foa, J. Nilsson, and R. Kotter, "Iterative decoding of product code constructions," in *in Proceedings of the International Symposium on Information Theory and its Applications*, Citeseer, 1994. 61

[51] S. Benedetto and G. Montorsi, "Average performance of parallel concatenated block codes," *Electronics Letters*, vol. 31, no. 3, pp. 156–158, 1995. 61

[52] S. Ng, T. Liew, L.-L. Yang, and L. Hanzo, "Binary bch turbo coding performance : union bound and simulation results," in *VTC2000-Spring. 2000 IEEE 51st Vehicular Technology Conference Proceedings (Cat. No. 00CH37026)*, vol. 2, pp. 849–853, IEEE, 2000. 61

[53] M. Belkasmi, M. Lahmer, and M. Benchrifa, "Iterative threshold decoding of parallel concatenated block codes," in *Turbo Coding Conf*, pp. 1–4, 2006. 64, 84

[54] A. Farchane, M. Belkasmi, and S. Nouh, "Generalized parallel concatenated block codes based on BCH and RS codes : construction and Iterative decoding," *Journal of Telecommunication*, vol. 12, pp. 1–9, JANUARY 2012. 64, 84

[55] B. Dorsch, "A decoding algorithm for binary block codes and J-ary output channels," *IEEE Trans. Info. Theory*, vol. 20, pp. 391–394, 1974. 70, 72

[56] M. Fossorier and S. lin, "Soft decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Info. theory*, vol. 41, pp. 1379–1396, sep 1995. 70, 72, 73, 76, 77

[57] J. G. D. Forney, *Concatenated codes*. MIT Press, Cambridge, Mass., 1966. 70

[58] R. M. Pyndiah, "Near-optimum decoding of product codes : block turbo codes," *IEEE Transactions on Communications*, vol. 46, pp. 1003–1010, Aug 1998. 71, 78, 80, 81

[59] M.Belkasmi and A. Farchane, "Iterative decoding of parallel concatenated block codes," *2008 International Conference on Computer and Communication Engineering*, pp. 230–235, May 2008. 71

[60] M. P. C. Fossorier, S. Lin, and J. Snyders, "Reliability-Based Syndrome Decoding of Linear Block Codes," *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 44, pp. 388–398, JANUARY 1998. 72, 73

[61] D. Gazelle and J. Snyders, "Reliability-based code-search algorithm for maximum likelihood decoding of block codes," *IEEE Trans. Inform. Theory*, vol. 43, pp. 239–249, Jan 1997. 73, 76

[62] Y. S. Han, C. R. Hartmann, and C.-C. Chen, "Efficient priority-first search maximum-likelihood soft-decision decoding of linear block codes," *IEEE transactions on information theory*, vol. 39, no. 5, pp. 1514–1523, 1993. 76

[63] H. Yagi, M. Kobayashi, T. Matsushima, and S. Hirasawa, "Complexity reduction of the Gazelle and Snyders decoding algorithm for maximum likelihood decoding," *IEICE Trans. Fundamentals*, vol. E86-A, pp. 2461–2472, Oct 2003. 77

[64] A. Farchane and M. Belkasmi, "New efficient decoder for product and concatenated block codes," *Journal of Telecommunication*, vol. 12, pp. 17–22, JANUARY 2012. 79

[65] A. Farchane, M. Belkasmi, and A.Azouaoui, "Exit Chart for Iterative Decoding of Product and Concatenated Block Codes," *International Journal of Computer Science Issues*, 07 2013. 79

[66] R. G. Gallager, "A simple derivation of the coding theorem and some applications," *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 3–18, jan 1965. 82

[67] H. Nickl, J. Hagenauer, and F.Burkert, "Approaching Shannon's capacity limit by 0.27 dB using simple Hamming codes," *IEEE Int. Commun. Letters*, vol. vol. 1, pp. 130–132, Sept 1997. 82

[68] A. Farchane and M. Belkasmi, "Turbo decoding of product codes based on Dual-Rm decoder ," *submitted to Security and Communication Networks*, —-. 83

[69] A. Farchane and M. Belkasmi, "Generalized serially concatenated codes : construction and iterative decoding," *International Journal of Mathematical and Computer Sciences*, vol. 6, no. 2, 2010. 84

[70] P. W. Shor, "Algorithms for quantum computation : discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134, Ieee, 1994. 85

[71] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999. 85

[72] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978. 85

[73] D. W. Kravitz, "Digital signature algorithm," July 27 1993. US Patent 5,231,668. 85

[74] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Prob. Contr. Inform. Theory*, vol. 15, no. 2, pp. 157–166, 1986. 85

[75] Y. X. Li, R. H. Deng, and X. M. Wang, "On the equivalence of mceliece's and niederreiter's public-key cryptosystems," *IEEE Transactions on Information Theory*, vol. 40, no. 1, pp. 271–273, 1994. 85

[76] C. M. J. Rosenthal and A. Shokrollahi, "Using low density parity check codes in the mceliece cryptosystem," in *IEEE International Symposium on Information Theory (ISIT'2000)*, 2000. 85

[77] M. Baldi and F. Chiaraluce, "Cryptanalysis of a new instance of mceliece cryptosystem based on qc-ldpc codes," in *2007 IEEE International Symposium on Information Theory*, pp. 2591–2595, IEEE, 2007. 85

[78] M. Baldi, M. Bodrato, and F. Chiaraluce, "A new analysis of the mceliece cryptosystem based on qc-ldpc codes," in *International Conference on Security and Cryptography for Networks*, pp. 246–262, Springer, 2008. 85, 86, 94

[79] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto, "MDPC-McEliece : New McEliece variants from moderate density parity-check codes," in *2013 IEEE International Symposium on Information Theory*, pp. 2069–2073, IEEE. 85, 101

[80] S. Heyse, I. Von Maurich, and T. Güneysu, "Smaller keys for code-based cryptography : Qc-mdpc mceliece implementations on embedded devices," in *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 273–292, Springer, 2013. 86

[81] I. Von Maurich and T. Güneysu, "Lightweight code-based cryptography : Qc-mdpc mceliece encryption on reconfigurable devices," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2014. 86

[82] I. V. Maurich, T. Oder, and T. Güneysu, "Implementing qc-mdpc mceliece encryption," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 3, pp. 1–27, 2015. 86, 95, 97

[83] J. Chaulet and N. Sendrier, "Worst case qc-mdpc decoder for mceliece cryptosystem," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1366–1370, IEEE, 2016. 86

[84] A. Janoska, "MDPC decoding algorithms and their impact on the McEliece cryptosystem," pp. 1085–1089. 86

[85] G. Liva and H. Bartz, "Protograph-based quasi-cyclic MDPC codes for McEliece cryptosystems," 86

[86] T. P. Berger and P. Loidreau, "How to mask the structure of codes for a cryptographic use," *Designs, Codes and Cryptography*, vol. 35, no. 1, pp. 63–79, 2005. 86

[87] V. M. Sidelnikov, "A public-key cryptosystem based on binary reed-muller codes," 1994. 86

[88] H. Janwa and O. Moreno, "Mceliece public key cryptosystems using algebraic-geometric codes," *Designs, Codes and Cryptography*, vol. 8, no. 3, pp. 293–307, 1996. 86

[89] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. Barreto, "Mdpc-mceliece : New mceliece variants from moderate density parity-check codes," in *2013 IEEE international symposium on information theory*, pp. 2069–2073, IEEE, 2013. 86, 93, 94

[90] C. Löndahl and T. Johansson, "A new version of mceliece pkc based on convolutional codes," in *International Conference on Information and Communications Security*, pp. 461–470, Springer, 2012. 86, 93

[91] B. Preneel, A. Bosselaers, R. Govaerts, and J. Vandewalle, "A software implementation of the mceliece public-key cryptosystem," in *Symposium on Information Theory in the BENELUX*, pp. 119–119, TECHNISCHE UNIVERSITEIT DELFT, 1992. 88

[92] G. Murat, *Résultants de polynômes de Ore et Cryptosystèmes de McEliece sur des Codes Rang faiblement structurés*. PhD thesis, Université de Limoges, 2014. 89

[93] A. BRAHIMI, *IMPLEMENTATION DES CRYPTOSYSTEMES BASES SUR LES CODES CORRECTEURS D'ERREURS*. PhD thesis, UNIVERSITE MOHAMED BOUDIAF M'SILA : FACULTE DES MATHEMATIQUES ET DE L . . . , 2014. 89

[94] N. Hadj-Said, A. Ali-Pacha, A. Belgoraf, and A. M'Hamed, "Crypto système à clé publique de mceliece basé sur les codes cycliques de hamming," in *MajecSTIC 2005 : Manifestation des Jeunes Chercheurs francophones dans les domaines des STIC*, pp. 384–388, 2005. 89

[95] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Transactions on information Theory*, vol. 47, no. 2, pp. 585–598, 2001. 89

[96] L. Minder and A. Shokrollahi, "Cryptanalysis of the sidelnikov cryptosystem," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 347–360, Springer, 2007. 93

[97] E. Barelli, *Étude de la sécurité de certaines clés compactes pour le schéma de McEliece utilisant des codes géométriques*. PhD thesis, Université Paris Saclay (COmUE), 2018. 93

[98] J.-C. Faugere, A. Otmani, L. Perret, and J.-P. Tillich, "Algebraic cryptanalysis of mceliece variants with compact keys," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 279–298, Springer, 2010. 93

[99] R. Misoczki and P. S. Barreto, "Compact mceliece keys from goppa codes," in *International Workshop on Selected Areas in Cryptography*, pp. 376–392, Springer, 2009. 93

[100] G. Landais and J.-P. Tillich, "An efficient attack of a mceliece cryptosystem variant based on convolutional codes," in *International Workshop on Post-Quantum Cryptography*, pp. 102–117, Springer, 2013. 93

[101] M. Bardet, J. Chaulet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Cryptanalysis of the mceliece public key cryptosystem based on polar codes," in *Post-Quantum Cryptography*, pp. 118–143, Springer, 2016. 93

[102] S. R. Shrestha and Y.-S. Kim, "New mceliece cryptosystem based on polar codes as a candidate for post-quantum cryptography," in *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*, pp. 368–372, IEEE, 2014. 93

[103] J.-B. Doré, *Optimisation conjointe de codes LDPC et de leurs architectures de décodage et mise en œuvre sur FPGA*. PhD thesis, INSA de Rennes, 2007. 94

Annexes

Annexe A : Calcul de la limite de Shannon

En 1948, Shannon a dérivé la formule qui exprime la capacité du canal en fonction du rapport signal à bruit comme suit : $C = W \log_2(1 + NS)(bits/second)$ Avec
W : La largeur de bande du canal S : La puissance de signal N :La puissance de bruit
Le tableau 1 et la figure 1 représentent la limite de Shannon en fonction du taux du code pour un canal AWGN avec une modulation.

Taux du Code	La limite de Shannon	Taux du Code	La limite de Shannon
0.068743	-1.382605	0.076696	-1.358051
0.085515	-1.330722	0.095280	-1.300324
0.106078	-1.266538	0.117997	-1.229019
0.131132	-1.187389	0.145577	-1.141239
0.161429	-1.090126	0.178784	-1.033574
0.197732	-0.971069	0.218361	-0.902058
0.240749	-0.825951	0.264961	-0.742115
0.291043	-0.649876	0.319022	-0.548513
0.348896	-0.437255	0.380625	-0.315279
0.414133	-0.181701	0.449292	-0.035583
0.485920	0.124056	0.523781	0.298204
0.562587	0.487801	0.602010	0.693666
0.641693	0.916430	0.681266	1.156535
0.720336	1.414350	0.758452	1.690418
0.795050	1.985754	0.829416	2.301974
0.860748	2.640939	0.888362	3.003799
0.911983	3.389831	0.931911	3.795956
0.948857	4.217693	0.963455	4.651386
0.975737	5.096374	0.985113	5.554839
0.991206	6.028060	0.994632	6.513076

Table 1 – Limites de Shannon

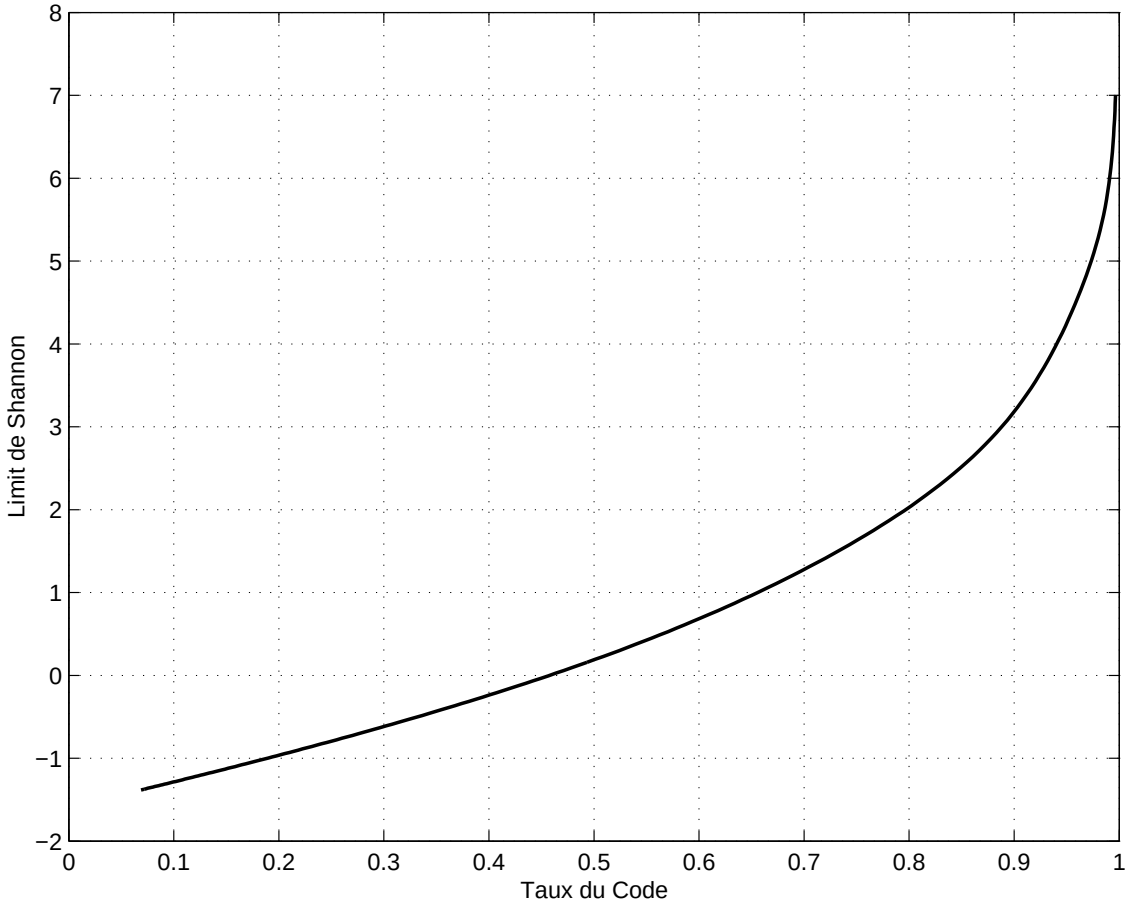


FIGURE 1 – Limite de Shannon en fonction du taux du code

Annexe B : Entrelacement

L'entrelacement est une technique couramment utilisée dans les systèmes de communication pour surmonter le bruit de canal en corrélation, comme une erreur d'éclatement ou de décoloration. L'entrelaceur réorganise les données d'entrée telles que les données consécutives sont espacées. À la fin du récepteur, les données entrelacées est disposé en arrière dans la séquence d'origine par le désentrelaceur. Plusieurs types d'entrelaceurs sont indiqués ci-après :

Procédure pseudo-aléatoire($indice, M, k$)

Entrée M ; k : entier
Sortie *indice* : tableau d'entiers
k, i, j, ind, S, min : entier
number : tableau d'entiers
$S \leftarrow M \times k$
Pour i allant de (0) à (S) faire
$number[i] \leftarrow rand()$
$index[i] \leftarrow i$
Fin Pour
Pour i allant de (0) à (S) faire
$min \leftarrow number[i]; ind \leftarrow i$
Pour j allant de $(i+1)$ à $(S-2)$ faire
Si($min > number[j]$)
$ind \leftarrow j$
$min \leftarrow number[j]$
Fin Si
Fin Pour
$min \leftarrow number[i]; number[i] \leftarrow number[ind]; number[ind] \leftarrow min$
$min \leftarrow index[i]; index[i] \leftarrow index[ind]; index[ind] \leftarrow min$
Fin Pour

Procédure Bloc($indice, M, k$)

Entrée M ;k : entier
Sortie *indice* : tableau d'entiers
$P; M; k; N_1; N_2; S$: entier
$N1 \leftarrow M; N_2 \leftarrow k$
$S \leftarrow M \times k$
Pour i allant de (0) à (S) faire
$P \leftarrow ((i \ mod \ N_1) \times N_2 + floor(\frac{i}{N_1}))$
$indice[i] \leftarrow P$
Fin Pour

Procédure DIAGONAL(indice,M,k)

Entrée $M; k$: entier
Sortie *indice* : tableau d'entiers
$P; M; k; N_1; N_2; S$: entier
$N1 \leftarrow M; N_2 \leftarrow k$
$S \leftarrow M \times k$
Pour i allant de (0) à (S) faire
$P \leftarrow ((i \times N_2) mod\ S + (\frac{i}{N_1} + i\ mod\ N_1) mod\ N_2)$
$indice[i] \leftarrow position$
Fin Pour

Procédure HILICAL($indice, M, k$)

Entrée $M; k$: entier
Sortie *indice* : tableau d'entiers
$P; M; k; N_1; N_2; S$: entier
$N1 \leftarrow M; N_2 \leftarrow k$
$S \leftarrow M \times k$
Pour i allant de (0) à (S) faire
$P \leftarrow ((i\ mod\ N_2) + (i \times N_2)) mod\ S))$
$indice[i] \times P$
Fin Pour

Procédure CYCLIC($indice, M, k, D$)

Entrée $M; k; D$: entier
Sortie *indice* : tableau d'entiers
$P; M; k; N_1; N_2; S$: entier ; tab1,tab2 : tableaux d'entiers
$D \leftarrow 20; N_1 \leftarrow M; N_2 \leftarrow k; S \leftarrow M \times k$
Pour i allant de (0) à (S) faire
$tab1[i\ mod\ M][\frac{i}{k}] \leftarrow i$
Fin Pour
Pour i allant de (0) à (M) faire
Pour j allant de (0) à (k) faire
$tab2[j] \leftarrow tab1[i][j]$
Fin Pour
Pour j allant de (0) à (k) faire
$tab1[i][(i \times D + j) mod\ N2] = tab2[j]$
Fin Pour
Fin Pour
Pour i allant de (0) à (S) faire
$P \leftarrow tab1[i\ mod\ M][\frac{i}{N_1}]$
$indice[i] \leftarrow P$
Fin Pour

Annexe C : Construction d'un corps de Galois à partir d'un élément primitif

Construction d'un corps de Galois

Les Corps de Galois font partie d'une branche particulière des mathématiques qui modélise les fonctions du monde numérique. Ils sont très utilisés dans la construction des codes correcteurs d'erreurs et en télécommunication en général.

La dénomination Corps de Galois (Galois field en anglais et désigné par GF) provient du mathématicien français E. Galois qui en a découvert leurs propriétés fondamentales.

Définition .7. *Un Corps de Galois consiste en un ensemble fini de nombres, ces nombres sont constitués à l'aide de l'élément base α a comme suit :* $0; 1; \alpha; \alpha^2; \alpha^3; ...; \alpha^{m-2}$ *avec* $m \in N$

Pour tout élément $\alpha \in GF$, il doit exister des éléments 0 (l'élément neutre additif) et 1 (l'élément neutre multiplicatif) tel que :

$$0 + \alpha = \alpha$$

$$1 \times \alpha = \alpha$$

$$(-\alpha) + \alpha = 0$$

$$0 \times \alpha = 0$$

$$si\ \alpha \neq 0; \alpha^{-1} \times \alpha = 1$$

Soient $p(x)$ le polynôme unitaire et irréductible en $GF(q^m)$ de degré d et v sa racine, donc on peut définir les éléments du corps $GF(q^m)$ comme expressions polynomiales en α de degré strictement inférieur à d. Noter que $p(x)$ est le seul polynôme irréductible en $GF(q^m)$ qui a α comme racine et est appelé polynôme irréductible de α sur $GF(q^m)$. Ce polynôme permet de construire le corps de Galois souhaité. Tous les éléments non nuls du corps peuvent être construits en utilisant l'élément α comme racine du polynôme primitif.

La table C.1 montre les polynômes primitifs pour les principaux corps de Galois. Pour chaque valeur de m, il peut y avoir plusieurs polynômes primitifs $p(x)$, mais dans cette table, on mentionne seulement les polynômes ayant le moins d'éléments.

Exemple .5. *On veut construire un corps binaire $GF(2)$ à 8 éléments. Pour représenter vectoriellement les 8 valeur binaires distinctes du corps il faut $log_2(8) = 3$ bits.*

On peut donc raisonner sur $GF(2^3)$. En consultant la table C.1 pour $m = 3$, on obtient le mot 1101 dont la représentation polynomiale est $p(x) = x^3 + x + 1$. Ce polynôme est le polynôme primitif du corps $GF(2^3)$. Soit α une racine du polynôme $p(x) = x^3 + x + 1$. On a alors :

$$p(\alpha) = \alpha^3 + \alpha + 1$$

Comme $p(\alpha) = 0$, alors : $0 = \alpha^3 + \alpha + 1$ $\alpha^3 = \alpha + 1$ Avec α^3, on peut construire tout les autres éléments du corps F_2^3 , comme par exemple :

$$\begin{aligned}\alpha^4 &= \alpha \times \alpha^3 \\ &= \alpha(\alpha + 1 \\ &= \alpha^2 + \alpha\end{aligned}$$

$$\begin{aligned}\alpha^5 &= \alpha \times \alpha^4 \\ &= \alpha(\alpha^2 + \alpha)) \\ &= \alpha^3 + \alpha^2 \\ &= (\alpha + 1) + \alpha^2 \\ &= \alpha^2 + \alpha + 1\end{aligned}$$

Et ainsi de suite, on construit le corps F_2. La table C.2 montre le corps sur F_2 avec $p(x) = x^3 + x + 1$. Soit r le nombre de coefficients d'un polynôme $p(x)$. Tout élément non nul du corps F_2^3 peut être représenté par un polynôme avec $r \leq m$.

m	Représentation binaire de $p(x)$
2	1 1 1
3	1 1 0 1
4	1 1 0 0 1
5	1 0 1 0 0 1
6	1 1 0 0 0 0 1
7	1 1 0 0 0 0 0 1
8	1 1 1 0 0 0 0 1 1
9	1 0 0 0 1 0 0 0 0 1
10	1 0 0 1 0 0 0 0 0 0 1
11	1 0 1 0 0 0 0 0 0 0 0 1
12	1 1 1 0 0 0 0 0 1 0 0 0 1
13	1 1 1 0 0 1 0 0 0 0 0 0 0 1
14	1 1 1 0 0 0 0 0 0 0 0 0 1 0 1
15	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
16	1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1
17	1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
18	1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
19	1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
20	1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

TABLE 2 – Polynômes primitifs pour les corps $GF(q^m)$ avec $q = 2$

Puissance de α	Élément du GF	Polynôme	α^2	α	1
0	1	1	0	0	1
1	α	α	0	1	0
2	α^2	α^2	1	0	0
3	α^3	$\alpha + 1$	0	1	1
4	α^4	$\alpha^2 + \alpha$	1	1	0
5	α^5	$\alpha^2 + \alpha + 1$	1	1	1
6	α^6	$\alpha^2 + 1$	1	0	1

TABLE 3 – Corps $GF(2^3)$ généré par le polynôme irréductible $x^3 + x + 1$

Annexe D : Classes cyclotomiques

Les classes cyclotomiques permettent de déterminer le nombre de facteurs irréductibles de $x^n - 1$. Elles permettent de trouver tous les polynômes minimaux de $x^n - 1$ (tous les facteurs de $x^n - 1$). On appelle classe cyclotomique q-aire modulo n de l'entier i, l'ensemble des entiers modulo n de la forme iq^j.

Pour $0 \leq i \leq n$ on note $C_i = \{i; iq; iq^2; ...; iq^j\}$ la classe cyclotomique de q modulo n qui contient l'entier i avec j le plus petit entier positif tel que :

$$iq^j = 1 \ mod \ n$$

Il existe une correspondance bijective entre les facteurs irréductibles de $x^n - 1$ et les classes cyclotomiques q-aires module n. En particulier, le nombre de facteurs irréductibles de $x^n - 1$ est égal au nombre de classes cyclotomiques. Le degré d'un facteur irréductible est égal au cardinal de la classe cyclotomique associée. L'entier i est souvent appelé chef de classe ou coset leader en anglais.

Exemple .6. *On se place sur le corps fini de caractéristique 2 et longueur 3, c'est-à-dire, sur un corps* $F2^3$. *On veut construire ses classes cyclotomiques :*
Pour $i = 0$
$j = 1; 2^1 \times 0 = 0 \ mod \ 7 = 0$
alors $C_0 = \{0\}$
Pour $i = 1$
$j = 0; 2^0 \times 1 = 1 \ mod \ 7 = 1$
$j = 1; 2^1 \times 1 = 2 \ mod \ 7 = 2$
$j = 2; 2^2 \times 1 = 4 \ mod \ 7 = 4$
Alors $C_1 = \{2; 4; 1\}$
Pour $i = 2$
$j = 0; 2^0 \times 2 = 2 \ mod \ 7 = 2$
$j = 1; 2^1 \times 2 = 4 \ mod \ 7 = 4$
$j = 2; 2^2 \times 2 = 8 \ mod \ 7 = 1$
Alors $C_2 = \{4; 1; 2\}$
Pour $i = 3$
$j = 0; 2^0 \times 3 = 3 \ mod \ 7 = 3$
$j = 1; 2^1 \times 3 = 6 \ mod \ 7 = 6$
$j = 2; 2^2 \times 3 = 12 \ mod \ 7 = 5$
Alors $C_3 = \{6; 5; 3\}$
Pour $i = 4$
$j = 0; 2^0 \times 4 = 4 \ mod \ 7 = 4$
$j = 1; 2^1 \times 4 = 8 \ mod \ 7 = 1$
$j = 2; 2^2 \times 4 = 16 \ mod \ 7 = 2$
Alors $C_4 = \{4; 2; 1\}$
Si on continuons ce calcule, nous pouvons facilement obtenir :
$C_0 = 0$
$C1 = C_2 = C_4 = \{1; 2; 4\}$
$C3 = C_5 = \{3; 5; 6\}$
La décomposition en facteurs irréductibles dans $GF(2^3)$ *de* $x^7 - 1$ *comporte donc un facteur*

de degré 1, et deux facteurs de degré 3.