IRINA ADJUDEANU

Codes correcteurs d'erreurs LDPC structurés

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en génie électrique
pour l'obtention du grade de Maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2010

Résumé

Dans ce travail nous nous intéressons aux problèmes liés à l'amélioration des performances d'erreur des codes LDPC. Cette amélioration peut être faite pendant la construction des codes LDPC ou pendant leur processus de décodage. Nous concentrons notre attention vers l'amélioration du décodage. Nous proposons un algorithme de décodage itératif log-SPA modifié, qui minimise l'erreur du décodage pour les mots-codes qui ne peuvent pas être décodés par l'algorithme SPA connu. L'algorithme proposé est basé sur une dépendance entre le poids d'un syndrome et le poids de la séquence d'erreurs généré au cours des itérations du décodage. Nous analysons plus en détail cette dépendance et l'effet du décodage log-SPA moyenné sur les poids des vecteurs d'erreur et les poids du syndrome, et ce, sur les ensembles de piégeage.

Concernant la construction des codes LDPC, on s'intéresse aux différents paramètres qui posent des problèmes dans la performance d'erreur : au plancher d'erreur, au périmètre du graphe de Tanner, à la distance minimale du code, aux ensembles de piégeage et aux ensembles d'arrêt. On présente différentes méthodes appliquées dans la construction des différents types de codes, pour éviter l'apparition des structures non-souhaitables dans le code. Nous faisons une comparaison entre la performance des

codes pseudo-aléatoires et les codes structurés, basés sur les géométries finies euclidienne et géométrique. On présente aussi différentes méthodes de construction des codes LDPC quasi-cycliques, qui peuvent éviter la formation de certains types d'ensembles de piégeage, d'ensembles d'arrêt ainsi que les cycles courts dans le graphe de Tanner. Les méthodes proposées donnent de très bas planchers d'erreur et une faible complexité du codage.

Remerciements

Je souhaiterais remercier mon directeur de recherche, Dr. Jean-Yves Chouinard et mon co-directeur de recherche, Dr. Paul Fortier, pour l'aide fournie pour la réalisation de cette étude.

Je remercie le Laboratoire de Radiocommunications et de Traitement de Signal (LRTS) de l'Université Laval, pour le soutien financier accordé.

Je remercie aussi mes parents et mon frère pour leur soutien et leurs encouragements.

Table des matières

Liste des tableaux

Liste des figures

Chapitre 1

Introduction

1.1 Mise en contexte sur le codage et les communications numériques

Nous vivons dans l'ère des télécommunications et de l'information. Lors des deux dernières décennies, les communications numériques ont beaucoup évolué. De nos jours, l'information est dans la plupart des cas véhiculée sous forme numérique, que ce soit sur support filaire (fibres optiques), ou en radio, réseaux cellulaires ou réseaux locaux sans fil ou bien des systèmes de stockage de l'information. Cette évolution a été déclenchée et entretenue par une forte demande de transmission et de traitement fiable, rapide et efficient de l'information de tous les types (traitement de la voix, des données ou des images). Et ce phénomène est présent dans tous les domaines (militaire, gouvernemental, commercial, etc.). De plus en plus, les communications fusionnent avec les technologies de traitement des données par l'ordinateur.

Les éléments d'un système de communications consistent en la transformation analogique / numérique de l'information, la réduction de la redondance, le cryptage et la protection contre les erreurs. Dans la transmission analogique, le signal est transmis tel quel, ce qui peut causer une perte de l'information utile à la réception à cause des interférences et du bruit existant sur le canal de transmission. En plus, la quantité d'information transmise est très grande et la transmission est faite dans des circuits d'une grande complexité et donc lents. Les communications numériques ont permis de contrer ces désavantages. En appliquant des méthodes de traitement numérique du signal, on réussit à protéger le signal de manière plus efficace. Les enjeux sont : une transmission à haut débit, en temps réel, à bas taux d'erreur, sécuritaire et avec une basse consommation énergétique. Dans un système de transmission, le traitement numérique du signal peut être appliqué à plusieurs reprises.

Le codage du canal numérique transforme la séquence d'information utile en une séquence discrète codée nommée mot de code. Le mot de code peut être binaire ou non-binaire. Dans ce mémoire, on étudie seulement les codes binaires. Le défi du codage de l'information numérique est de réussir à bien récupérer l'information à la réception, le moins possible affectée par les bruits du canal de transmission. Le récepteur transforme la séquence reçue codée en une séquence estimée d'information. Cette séquence doit être idéalement la même séquence discrète transmise, mais en réalité elle est affectée par des erreurs de transmission. La séquence discrète est ensuite transformée en une séquence continue et elle est livrée à la sortie.

En 1948, Shannon a démontré que lorsque le taux de transmission du système est inférieur à la capacité du canal de transmission, les erreurs causées par le bruit du canal

peuvent être réduites à un niveau arbitrairement bas par l'utilisation d'un codage et d'un décodage approprié. À partir de ce moment-là, les chercheurs ont commencé à étudier différentes méthodes de construction des codes correcteurs d'erreur. Le but de la théorie des codes correcteurs d'erreurs est de minimiser le plus possible les erreurs de décodage, en assurant en même temps de très grandes vitesses de transmission et de faibles coûts du codeur et du décodeur. De nos jours, il existe une multitude de méthodes visant à produire de bons codes correcteurs. Les années 1980s et 1990s ont amené d'autres dimensions de développement des ces codes, en ajoutant les codes non-binaires et les méthodes itératives de décodage à décision souple.

Les deux types principaux de codes utilisés sont les codes blocs et les codes convolutifs. Les codes bloc peuvent être linéaires ou non-linéaires. Les codes blocs linéaires peuvent être cycliques ou non-cycliques. La première classe de codes blocs linéaires a été découverte par Richard W. Hamming en 1950 : les codes de Hamming. Ils corrigent toutes les erreurs simples. La deuxième classe est formée par les codes linéaires Reed-Muller découverts par David E. Muller et Irwin S. Reed en 1954 : ils sont efficaces pour la correction des erreurs aléatoires multiples. Une autre classe importante de codes blocs est les codes Golay, qui ont beaucoup été utilisés dans les communications par satellite. Les codes BCH sont une généralisation des codes de Hamming et sont de codes cycliques. Un cas particulier est le code de Reed-Solomon.

Les codes convolutifs peuvent être non-récursifs ou récursifs. Les codes turbo sont en général des codes convolutifs récursifs ; ils conduisent à d'excellentes performances. Les codes LDPC font partie de la classe des codes blocs linéaires et s'approchent davantage de la limite de Shannon (capacité d'un canal). Ce sont les codes que nous allons étudier

en détail dans ce mémoire. Leurs performances peuvent dépasser les performances des autres types de codes présentés, même des codes turbo. Par exemple, pour différents types de codes, le rapport signal à bruit nécessaire pour atteindre une probabilité d'erreur inférieure à 10^{-5} est de 4.5 dB pour les codes convolutifs, 5.4 dB pour les codes BCH et Reed-Solomon, de valeurs inférieures à 1 dB pour les codes turbo et 0.005 dB pour les codes LDPC.

1.2 Motivation

Le problème majeur dans la théorie de codage est la construction des codes qui s'approchent de la capacité du canal et la conception d'algorithmes de codage et de décodage efficaces.

Les codes LDPC sont des codes correcteurs qui approchent la limite de Shannon. Il a été démontré que les codes LDPC longs avec un décodage itératif basé sur la propagation de la confiance atteignent une performance d'erreur à une fraction de décibel de la limite du Shannon [1], [2], [3], [4], [5], [6]. Les codes LDPC sont en grande compétition avec les codes turbo dans les systèmes de communications numériques qui demandent une fiabilité élevée. Aussi, les codes LDPC ont quelques avantages par rapport aux codes turbo :

(a) ils ne nécessitent pas d'entrelaceur pour réaliser une bonne performance d'erreur,

(b) ils ont une meilleure performance par trame,

(c) leur plancher d'erreur se produit à un niveau de BER de beaucoup inférieur,

(d) leur décodage n'est pas basé sur un treillis et peut être réalisé par un processus

parallèle.

Pour des grands taux et longueurs, les codes LDPC ont des meilleures performances que les codes turbo. Ils sont appliqués aux systèmes CDMA (Code Division Multiple Access), OFDM (Orthogonal Frequency Division Multiplexing) et aux systèmes de codage espace-temps. Récemment, ils ont été sélectionnés pour la norme de transmission vidéo numérique (DVB) et pour des applications en temps réel comme le stockage magnétique, l'Ethernet à 10 GB et les réseaux locaux sans fil avec débit élevé (WLAN).

1.3 Contributions

Dans ce mémoire, nous allons concentrer notre attention vers le processus du décodage des codes LDPC. Actuellement, les codes LDPC donnent de très bons résultats avec l'algorithme de décodage itératif SPA, basé sur la propagation de la confiance - BP (belief propagation). Pour améliorer l'algorithme log-SPA existant, nous avons proposé un algorithme de décodage log-SPA modifié, qui minimise l'erreur du décodage pour les mots-codes qui ne peuvent pas être décodés. Cet algorithme est basé sur une dépendance entre le poids d'un syndrome et le poids de la séquence d'erreurs générées au cours des itérations du décodage. On a analysé plus en détail cette corrélation entre les vecteurs de poids et on a étudié l'effet du décodage log-SPA moyenné sur les poids des vecteurs d'erreur et les poids du syndrome, et ce, sur les ensembles de piégeage. On s'intéresse aux problèmes spécifiques d'ensembles d'arrêt et surtout d'ensembles de piégeage.

1.4 Structure du mémoire

Le chapitre 2 présente la théorie des codes LDPC. On y retrouve différentes méthodes de construction et de décodage existantes. Le chapitre 3 présente les codes structurés basés sur les géométries finies (euclidienne et projective), les codes pseudo-aléatoires et un algorithme de décodage SPA modifié que l'on propose. Le quatrième chapitre porte sur la performance des codes LDPC. On s'intéresse au plancher d'erreur, à la distance minimale, aux ensembles de piégeage et aux ensembles d'arrêt, aux pseudo-mots codes et aux pseudo-poids. Dans ce chapitre, on présente aussi l'effet du décodage SPA moyenné sur les ensembles de piégeage. Dans le chapitre 5, nous étudions la corrélation entre les poids des vecteurs d'erreur et les poids des syndromes en fonction des itérations du décodage itératif log-SPA, ainsi que l'effet de l'algorithme SPA moyenné sur les deux poids. Le chapitre 6 présente les codes LDPC quasi-cycliques. La conclusion au chapitre 7 rappelle les contributions de ce mémoire et suggère des avenues de recherche.

Chapitre 2

Les codes correcteurs d'erreurs LDPC

2.1 Introduction

Les codes LDPC ont été découverts par Gallager [7], [8] dans les années '60, mais il a proposé seulement une méthode générale pour construire des codes LDPC pseudo-aléatoires ; les bons codes LDPC sont générés par ordinateur (en particulier les codes longs) et leur décodage est très complexe dû au manque de structure. Ces codes ont été ignorés jusqu'à 1981 quand Tanner leur a donné une nouvelle interprétation d'un point de vue graphique [9]. Sa théorie a été aussi ignorée pour les prochaines 14 années jusqu'au jour où quelques chercheurs en codage ont commencé à étudier les codes en graphes et le décodage itératif.

La première construction algébrique et systématique de codes LDPC basée sur les géométries finies a été proposée par Kou, Lin et Fossorier dans les années 2000 [10], [11],

[12], [5], [13]. La classe de codes LDPC à géométrie finie possède une bonne distance minimale et les graphes de Tanner n'ont pas de cycles courts. Leur structure est cyclique ou quasi-cyclique, ce qui fait que leur codage est simple et peut être réalisé avec des registres à décalage linéaire. Avec ce type de codes de grande longueur, on obtient une très bonne performance d'erreur.

La construction et le décodage des codes LDPC peuvent être fait de plusieurs manières. Un code LDPC est caractérisé par sa matrice de parité.

Définition des codes LDPC [7], [8] : Un code LDPC *régulier* est défini comme l'espace nul d'une matrice de contrôle de parité **H**, qui a les propriétés suivantes :

(1) chaque ligne a ρ valeurs de 1 ;

(2) chaque colonne a γ valeurs de 1 ;

(3) le nombre de 1 en commun entre deux colonnes quelconques, désigné par λ, n'est pas plus grand que 1 (donc $\lambda = 0$ ou $\lambda = 1$) ;

(4) ρ et γ ont des valeurs petites en comparaison avec la longueur du code et avec le nombre de lignes de la matrice **H**.

La matrice **H** est une **matrice creuse** : elle a une faible densité de valeurs de 1.

Si toutes les colonnes ou toutes les lignes de **H** n'ont pas le même poids, le code LDPC s'appelle *code LDPC irrégulier.*

On doit observer que les lignes de **H** ne sont pas nécessairement linéairement indépendantes sur le corps de Galois binaire $CG(2)$. Dans ce cas, pour déterminer la dimension du code, on doit calculer le rang de la matrice **H**.

2.2 Les graphes de Tanner pour les codes en blocs linéaires

Les graphes de Tanner [9] sont très utiles pour la représentation des codes en blocs linéaires, parce qu'ils affichent la relation entre les bits des mots-codes et les noeuds de contrôle de parité. Ils ont été proposés pour la première fois par Tanner pour le décodage itératif des codes LDPC. Les codes LDPC sont représentés sous forme de graphes bipartites. Un graphe est *bipartite* si son ensemble de noeuds ($\mathcal{V}$) peut être divisé en deux sous-ensembles disjoints $\mathcal{V}_1$ et $\mathcal{V}_2$, tels que chaque arête joint un noeud dans $\mathcal{V}_1$ avec un noeud dans $\mathcal{V}_2$ et sans que deux noeuds dans $\mathcal{V}_1$ ou dans $\mathcal{V}_2$ soient connectés. Pour un code LDPC, les deux ensembles de noeuds $\mathcal{V}_1$ et $\mathcal{V}_2$ représentent les n bits du mot-code (noeuds des variables) et les J équations de contrôle de parité qui doivent être satisfaites par les bits codés (noeuds de contrôle). Toutes les connexions sont faites entre chaque noeud de bit codé et ses noeuds de contrôle correspondants (qu'on vérifie dans la somme de contrôle).

Un *chemin* dans un graphe est une séquence de noeuds et d'arêtes, qui commence et finit par des noeuds, tel que chaque arête est incidente avec les noeuds qui la précèdent et la suivent et aucun noeud n'apparaît plus d'une fois. La *longueur* d'un chemin est donnée par le nombre d'arêtes formant le chemin. Un chemin qui commence et finit avec le même noeud s'appelle *cycle*. Si un graphe bipartite contient des cycles, ces cycles ont des longueurs paires. La longueur du plus court cycle dans le graphe s'appelle *le périmètre du graphe.*

Pour un code LDPC régulier, les degrés de tous les noeuds de bits codés sont égaux

à γ (le poids de chaque colonne de la matrice **H**) et les degrés de tous les noeuds de contrôle sont égaux à ρ (le poids de chaque rangée de la matrice **H**). Ainsi, le graphe de Tanner est *régulier*. Il ne peut pas y avoir deux bits codés qui peuvent être vérifiés par deux équations différentes, donc le graphe ne va contenir aucun cycle de longueur 4. Pour décoder un graphe d'un code LDPC avec le décodage itératif, il est important que le graphe de Tanner du code ne contienne aucun cycle de courte longueur (4 ou 6, mais en particulier 4), parce que les cycles courts limitent la performance d'erreur et empêchent le processus de décodage de converger vers la performance obtenue avec le décodage à vraisemblance maximale MLD (Maximum Likelihood Decoding) [14], [15].

2.3 Construction des codes LDPC

2.3.1 Une construction géométrique des codes LDPC

Les codes LDPC peuvent être construits de manière algébrique à partir de points et de lignes de la géométrie finie, comme la géométrie Euclidienne et la géométrie projective définies sur des champs finis [14]. Une géométrie finie est formée de points et lignes, qui ont les propriétés suivantes :

(1) chaque ligne a ρ points ;

(2) chaque point appartient à γ lignes ;

(3) deux points sont connectés par juste une ligne ;

(4) deux lignes sont soit disjointes, soit leur intersection est un seul point.

Pour chaque type de géométrie, on peut construire des codes LDPC de type 1 et de type 2, qui sont vraiment connexes et leurs graphes sont conjugués entre eux : les noeuds des bits codés d'un graphe sont les noeuds de contrôle pour l'autre graphe. Pour le type 1, on peut former une matrice de parité **H** dont les rangées sont les vecteurs d'incidence des lignes existantes dans la géométrie finie et les colonnes sont les points. Le graphe correspondant à cette matrice n'a aucun cycle de longueur 4. Pour un code LDPC de type 2, la matrice de parité **H** est la transposée de la matrice correspondante de type 1 : les rangées sont les vecteurs d'incidence des points et les colonnes sont les lignes de la géométrie finie. Les graphes correspondants à ces matrices n'ont aucun cycle de longueur 4.

Les codes EG-LDPC (basés sur la géométrie euclidienne) et les codes PG-LDPC (basés sur la géométrie projective) ont des performances d'erreur presque identiques [14]. Dans la construction des codes EG-LDPC de type 1 et 2, on peut éliminer le point d'origine de la géométrie et toutes les lignes qui passent par l'origine. Ainsi, on met le code EG-LDPC de type 1 dans une forme cyclique et le code EG-LDPC de type 2 dans une forme quasi-cyclique. Ceci simplifie le circuit de codage ; on doit garder en mémoire juste la première rangée de **H** et les autres rangées seront ajoutées à l'aide d'un registre à décalage linéaire. La différence entre les 4 types de codes (EG-LDPC et PG-LDPC de type 1 et 2) est comment on trouve les positions des valeurs de 1 dans la première rangée de la matrice de parité et les différents paramètres des codes obtenus. Les formules pour calculer ces paramètres sont données en [14].

Ces codes peuvent être raccourcis par élimination des colonnes de la matrice de parité [14]. Pour les codes EG-LDPC de type 1, les colonnes éliminées correspondent

aux points d'un ensemble de paquets parallèles qui ne passent pas par l'origine de la géométrie finie. On obtient une nouvelle matrice irrégulière dans laquelle les colonnes ont le même poids, mais les lignes ont des poids inférieurs. Son noyau donne un code LDPC irrégulier plus court avec la même distance minimale que le code initial. Pour les codes EG-LDPC de type 2, il faut mettre la matrice de parité dans une forme circulaire et on élimine les colonnes correspondant à un ensemble de lignes. On obtient un code quasi-cyclique plus court. On doit mentionner que plus on réduit la matrice de parité, plus on obtient une meilleure performance d'erreur du code.

2.3.2 La construction des codes LDPC de Gallager

Pour construire la matrice de parité $\mathbf{H}$ d'un code LDPC de Gallager, il faut d'abord construire une sous-matrice $\mathbf{H_i}$ ayant un poids des colonnes égal à 1 et un poids de lignes ρ. Ensuite on doit trouver de permutations des colonnes de cette sous-matrice pour former les autres sous-matrices avec lesquelles on forme la matrice de Gallager de manière suivante :

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_\gamma \end{bmatrix} \tag{2.1}$$

Lorsqu'on choisit les permutations des colonnes des sous-matrices, on doit garder une bonne distance minimale de la matrice de parité $\mathbf{H}$ et il faut éviter les cycles courts dans son graphe de Tanner. Gallager ne donne aucune méthode de construction pour ce

type de codes. Dans [14], [16], [17] on retrouve une telle méthode de construction basée sur la structure parallèle des lignes dans la géométrie Euclidienne $EG(m, 2^s)$ sur le corps de Galois $CG(2^s)$. Il y a beaucoup de manières possibles pour choisir les paquets des lignes parallèles et selon ce choix on obtient des codes LDPC EG-Gallager de dimensions différentes. Pour chacune des valeurs de m et s, on obtient un code LDPC de Gallager de longueur 2^{ms} pour différents valeurs de γ. Ces codes ont différents rendements et distances minimales.

Si on compare la performance d'erreur d'un tel code LDPC avec d'autres codes LDPC obtenus par ordinateur, qui ont le même rendement et presque la même longueur, on peut observer que le code EG-LDPC est meilleur.

On ne peut pas construire de codes LDPC dans la forme de Gallager basés sur la géométrie projective, parce que les lignes dans une géométrie projective n'ont pas la même structure parallèle simple que les lignes de la géométrie Euclidienne.

2.3.3 Les codes LDPC quasi-cycliques

2.3.3.1 Construction des codes LDPC quasi-cycliques par décomposition circulaire

Cette méthode consiste en la décomposition d'une matrice carrée, régulière et circulaire en plusieurs matrices circulaires de mêmes dimensions, mais avec des poids différents [14]. On obtient ces nouvelles matrices à partir de chaque colonne de la matrice de parité initiale, qui est décomposée en plusieurs colonnes de même longueur. Le poids de la colonne initiale est partagé parmi les différentes colonnes. À partir de chaque

nouvelle colonne ainsi formée, on forme une matrice circulaire par permutations circulaires successives de la colonne en bas. La méthode présentée s'appelle *la décomposition des colonnes* d'une matrice de parité.

De même, on peut décomposer la matrice initiale en descendants, en décomposant sa première ligne en plusieurs lignes et ensuite en faisant des permutations circulaires à la droite de chaque nouvelle ligne. Cette méthode s'appelle *la décomposition de rangée.* Si la matrice initiale est une matrice creuse, la matrice obtenue est aussi une matrice creuse de densité plus faible, qui donne un code LDPC quasi-cyclique dont le graphe de Tanner n'a pas de cycles de longueur 4.

Des matrices creuses circulaires peuvent être construites à partir des vecteurs d'incidence des lignes dans une géométrie Euclidienne ou projective. De plus, il n'existe pas deux lignes (ou deux colonnes) dans la même matrice ou dans différentes matrices circulaires ayant plus d'une valeur de 1 en commun. En conséquence, les codes LDPC quasi-cycliques peuvent être construits en décomposant une ou un groupe de ces matrices circulaires géométriques [18], [19].

2.3.3.2 Les codes LDPC quasi-cycliques (QC) pour un codage rapide et pratique

Par rapport aux codes LDPC construits de manière aléatoire, la matrice de parité d'un code QC-LDPC consiste en plusieurs matrices de permutation ou matrices nulles. Ainsi, la mémoire de stockage peut être réduite considérablement (il suffit de mémoriser les premières rangées de chaque sous-matrice : les autres rangées sont formées par permutations circulaires). Cette structure est orientée sur la réalisation d'un décodeur

plus efficace. Ce qui est important est que cette nouvelle structure de matrice de code LDPC ne diminue pas les performances du code. Ceci a été démontré par des simulations faites par ordinateur [20].

Il a été démontré que le périmètre d'un code QC-LDPC est limité en haut par un certain nombre qui est déterminé par les positions des matrices circulaires de permutations [21]. On peut donc obtenir une bonne performance du code en contrôlant la structure des cycles.

Une classe spéciale de codes LDPC quasi-cycliques sont *les codes LDPC de type bloc (B-LDPC)* qui possèdent un algorithme de codage efficace en raison de la structure simple de leurs matrices de parité. Avec l'implémentation efficace du codeur et du décodeur des codes B-LDPC et leur bonne performance d'erreur, ils représentent un axe très prometteur pour l'implémentation des *systèmes de codage LDPC en temps réel.* Dans [20], on trouve les estimations de valeurs de sortie et de la complexité matériel pour leur décodeur et leur codeur. Les codes B-LDPC sont construits comme des codes QC-LDPC irréguliers avec des matrices de parité presque triangulaires inférieures, de sorte qu'ils aient un algorithme de codage efficace, un bon palier de bruit et un plancher d'erreur bas. Leur complexité ne dépend pas de manière linéaire de la taille des matrices de parité.

Pour la construction des codes B-LDPC, au lieu de faire la multiplication entre la matrice de parité et le vecteur de message, on construit un système d'équations composé par des multiplications de matrices et de vecteurs de taille inférieure. Ainsi, on obtient un codage plus rapide et une structure plus simple.

Quand il n'y a pas de sous-matrices nulles dans la structure de $\mathbf{H}$, le code est

régulier et le rendement du code est supérieur au rendement d'un code irrégulier parce que dans ce cas la matrice **H** possède un rang maximal (si **H** est triangulaire inférieure ou supérieure avec des éléments non-nuls sur la diagonale principale, **H** a le rang maximum).

Dans [21], les auteurs proposent une méthode efficace de construction des codes B-LDPC, nommé le *Principe A* : on choisit d'abord les blocs sur les colonnes avec un petit poids tels que la longueur minimale des cycles entre les noeuds de degré inférieur soit maximum. Ensuite, on choisit les blocs des colonnes correspondants aux noeuds avec un degré supérieur tels que le périmètre du graphe de Tanner soit maximum. Les codes qui sont ainsi construits ont un plancher d'erreur bas et des meilleures valeurs des rapports de taux d'erreurs binaires (BER) et de taux d'erreurs par trame (FER).

La performance d'erreur de contrôle peut être améliorée si on impose que le degré des petits cycles inévitables soit le plus grand possible.

On doit mentionner que les noeuds de degré élevé sont plus sensibles aux erreurs (parce qu'ils ont plusieurs chemins pour la mise à jour des messages entre les noeuds variables et les noeuds de parité). Richardson [3] a montré par la méthode *d'évolution de la densité (density evolution)* que les codes de très grande longueur (qui convergent vers l'infini) s'approchent de très près de la limite de Shannon. Mais ceci n'est pas valide pour les codes courts. La performance des codes LDPC dépend aussi de cycles existants dans le graphe de Tanner et de la distance minimale du code. Dans [22], les auteurs ont analysé les codes LDPC de longueur finie (courts) pour un canal de transmission BEC (binary erasure channel).

2.3.4 Les codes LDPC aléatoires

On peut construire de codes LDPC pseudo-aléatoires à l'aide de l'ordinateur d'une manière basée sur un ensemble de conditions données par la définition des codes LDPC donnée à la section 2.1. Ces codes fournissent de bonnes performances [23]. La construction de la matrice de parité **H** est faite en plusieurs étapes. On choisit la première colonne de la matrice de manière aléatoire, ayant un certain poids. À chaque étape, une nouvelle colonne de même poids est ajoutée à une matrice formée partiellement. La colonne ajoutée est choisie entre plusieurs colonnes selon les conditions à respecter par la matrice de parité d'un code LDPC [14].

En raison des conditions imposées pendant la construction du code, son graphe de Tanner ne contient pas de cycles d'ordre 4 et alors son périmètre est égal à au moins 6. Cette construction n'est efficace que pour des petites valeurs du poids des colonnes (3 ou 4). Pour des valeurs de poids plus grandes, les ordinateurs n'arrivent pas à effectuer efficacement les calculs (en un temps raisonnable).

Le code construit par cette méthode n'est pas unique parce que les colonnes sont choisies aléatoirement et il y a une multitude de choix possibles. Ainsi, ce type de construction génère un ensemble de codes LDPC aléatoires.

Il est très difficile de déterminer la distance minimale du code et pour de petites valeurs des poids, la limite inférieure pour la distance minimale peut être très petite. Ces codes n'ont pas les mêmes propriétés structurales que les codes de géométrie finie (pas de structure cyclique ou quasi-cyclique). En conséquence, l'implémentation matérielle de ces codes est beaucoup plus complexe et ne peut pas être réalisée avec des registres

à décalage linéaires.

Ces codes ne peuvent pas être décodés par le décodage logique majoritaire ou par le décodage avec basculement de bit (BF). Avec le décodage SPA, la solution ne converge pas aussi vite que dans le cas des codes LDPC de géométrie finie.

En terme de performance d'erreur, ces codes s'approchent beaucoup de la limite de Shannon.

2.3.5 Les codes LDPC irréguliers

Un code **irrégulier** a une matrice de parité **H** avec différents poids de colonnes et de rangées. Ceci signifie que dans son graphe de Tanner, les noeuds de bits codés ont des degrés multiples ainsi que les noeuds de parité [14]. Les codes LDPC irréguliers sont construits le plus généralement sur la base de leurs graphes de Tanner selon la distribution $\gamma(X)$ des degrés des noeuds variables et la distribution $\rho(X)$ des noeuds de contrôle (leur performance d'erreur est fortement liée aux valeurs de ces deux paramètres) [3], [24].

Il y a des algorithmes d'optimisation des deux degrés (des rangées et des colonnes) pour un rendement de code fixé et ceci seulement quand la longueur du code approche l'infini et le graphe du code n'a pas de cycles et avec un nombre infini d'itérations [3]. Ces algorithmes sont centrés sur l'évolution des densités de probabilités des messages passés entre les noeuds dans un décodeur basé sur la propagation de la confiance (BP). Mais les graphes de Tanner pour les codes de longueur finie ne peuvent pas être faits sans cycle. Donc, les distributions optimales des degrés conçus pour les codes de longueur infinie ne sont pas optimales pour les codes de longueur finie. Si on construit un code

LDPC irrégulier de longueur finie, sur la base des distributions asymptotiques optimales de degrés, on peut obtenir un plancher d'erreur élevé et le code pourrait avoir une faible performance d'erreur par bit à cause du grand nombre de noeuds variables de degré 2 dans le graphe de Tanner. Un grand nombre de noeuds variables de degré 2 donnent une faible distance minimale. Pour résoudre ces problèmes, les conditions suivantes sont proposées dans [3] :

1. les noeuds de degré 2 sont construits de sorte qu'ils ne possèdent pas de cycles,
2. les noeuds variables de degré 2 doivent correspondre aux bits de parité du mot codé,
3. le graphe de Tanner ne possède pas de cycles de longueur 4,
4. pour améliorer le plancher d'erreur du code, le nombre de noeuds variables de degré 2 doit être plus petit que le nombre de bits de parité du code (ou le nombre de colonnes de poids 2 de la matrice de parité $\mathbf{H}$ doit être plus petit que le nombre de lignes de la matrice $\mathbf{H}$).

Respectant les conditions mentionnées auparavant, on construit le graphe de Tanner en connectant les noeuds variables avec les noeuds de contrôle par des lignes. Le choix des lignes de connexion n'est pas unique et on utilise l'ordinateur pour le faire. Ensuite, on construit la matrice de parité $\mathbf{H}$ selon le graphe de Tanner. On obtient un code irrégulier aléatoire de longueur et de rendements désirés.

2.4 Décodage des codes LDPC

Par rapport aux autres types de codes, le décodage des codes LDPC ne pose pas autant de problèmes pour les chercheurs que leur construction. Le travail le plus difficile est de trouver les meilleures méthodes pour construire des codes LDPC efficaces. Un code LDPC peut être décodé par plusieurs méthodes, telles que :

1. décodage avec des décisions fermes
   - décodage avec la logique majoritaire (MLG)
   - décodage avec basculement de bit (BF)
2. décodage avec des décisions pondérées
   - décodage basé sur la probabilité *a posteriori* (APP)
   - décodage itératif basé sur la propagation de la confiance (somme-produit SPA)
3. décodage mixte (ferme et pondéré)
   - décodage BF pondéré

La méthode MLG est la plus simple du point de vue de la complexité du circuit. La méthode BF demande un peu plus de complexité du circuit, mais elle donne des meilleures performances d'erreur que la méthode MLG. Les méthodes APP et SPA donnent des meilleures performances d'erreur, mais nécessitent aussi une plus grande complexité du circuit. Le décodage BF pondéré représente un bon compromis entre les deux caractéristiques. La méthode SPA donne la meilleure performance d'erreur entre les 5 types de décodage.

2.4.1 Décodage MLG des codes LDPC

La méthode MLG à une seule étape [14] peut être appliquée au décodage des 4 types de codes LDPC.

On calcule les syndromes

$$s_j^{(l)} = e * h_j^{(l)} = \sum_{i=0}^{n-1} e_i h_{j,i}^{(l)} \tag{2.2}$$

où $h_j^{(l)}$ $(1 \leq j \leq \gamma)$ sont les γ lignes de $\mathbf{H}$ qui sont orthogonales sur le bit de la l-ème position. L'ensemble des sommes de contrôle $s_j^{(l)}$ sont orthogonales sur le bit d'erreur e_l et on peut les utiliser pour l'estimation de e_l. Le bit d'erreur est bien corrigé si dans le vecteur d'erreur il y a moins de $\gamma/2$ erreurs.

2.4.2 Algorithme de décodage à basculement de bit (BF) des codes LDPC

Cette méthode est basée sur l'échange du nombre d'échecs de parité quand un bit de la séquence reçue est basculé. Il s'agit d'une méthode itérative de décodage [7], [8], [14]. Le décodeur calcule toutes les sommes de parité et ensuite il change chaque bit dans la séquence reçue s'il fait partie de plus de δ équations de parité échouées. La valeur de δ est un seuil fixé et dépend des paramètres du code (ρ, γ, d_{min}) et du SNR. À partir de la séquence modifiée, le décodeur recalcule les sommes de parité et le processus est répété jusqu'à ce que toutes les sommes de parité soient nulles.

Le nombre d'itérations du décodage est une variable aléatoire et dépend du rapport signal à bruit du canal. Pour des meilleures performances on peut utiliser des seuils adaptatifs δ et utiliser aussi une méthode hybride entre BF et MLG. Le décodage BF

corrige beaucoup de séquences d'erreur qui possèdent plus de bits erronés que la capacité du code pour corriger les erreurs.

2.4.3 Décodages MLG et BF pondérés

Une meilleure performance du décodage est obtenue si on ajoute de l'information de fiabilité (qui demande une complexité du circuit additionnelle). Pour un canal AWGN, une mesure de fiabilité est l'amplitude du symbole reçu ($|y_l|$) (plus l'amplitude est grande, plus la fiabilité de la décision ferme est grande). Pour un code LDPC caractérisé par sa matrice de parité **H** de dimension $J \times n$, on définit :

$$|y_j|^l_{min} = \{min\{|y_i|\} : 0 \leq i \leq n-1, h_{j,i} = 1\} \tag{2.3}$$

, et

$$E_l = \sum_{s_j^{(l)} \in S_l} (2s_j^{(l)} - 1)\,|y_j|^l_{min} \tag{2.4}$$

. E_l est une somme orthogonale pondérée sur la l-ème position du bit codé. S_l est l'ensemble de syndromes. La décision est : $e_l = 1$ si $E_l > 0$ et $e_l = 0$ si $E_l \leq 0$ qui modifie le décodage MLG à une étape en un décodage MLG pondéré. Cette relation de décision peut être utilisée aussi pour le décodage BF, donnant l'algorithme de décodage BF pondéré [14].

2.4.4 Décodage itératif basé sur la propagation de la confiance - SPA

Le décodage SPA [25], [3], [23], [26], [4] est une méthode itérative de décodage qui est très efficace pour les codes LDPC. Il converge relativement vite pour les codes EG-LDPC

et PG-LDPC. La performance d'erreur dépend de quelques paramètres importants du code, tels que :

(1) le périmètre du graphe de Tanner - qui doit être suffisamment grand pour qu'il n'y ait pas des cycles courts, en particulier de longueur 4 ;

(2) la distance minimale - qui varie de manière inversement proportionnelle avec le périmètre. Ceci nous amène à un compromis car elle doit être suffisamment grande pour une bonne identification ;

(3) les poids des lignes et des colonnes ;

(4) le coefficient d'erreur - le nombre de mots-code de poids minimal ; pour des valeurs faibles, il donne une meilleure performance d'erreur si le rapport signal à bruit est faible et les erreurs sont grandes.

2.5 Conclusion

Dans ce chapitre, nous avons décrit les notions de base sur les codes LDPC, quelques méthodes de construction et différentes méthodes de décodage. On a présenté les graphes de Tanner, qui sont une représentation utile des codes blocs linéaires et en particulier des codes LDPC.

On a expliqué comment on peut construire différents types de codes LDPC, comme les codes LDPC aléatoires, les codes LDPC pseudo-aléatoires et les codes LDPC structurés (i.e. les codes basés sur les géométries finies et les codes quasi-cycliques).

Concernant le décodage des codes LDPC, on a présenté plusieurs méthodes : le décodage avec la logique majoritaire (MLG), le décodage avec basculement de bit (BF),

le décodage avec des décisions pondérées, le décodage basé sur la probabilité *a posteriori* (APP) et le décodage itératif basé sur la propagation de la confiance (somme-produit SPA), qui en fait jusqu'à présent est le meilleur algorithme de décodage pour les codes LDPC et le plus utilisé.

Chapitre 3

Étude comparative des codes LDPC structurés

3.1 Introduction

Ce chapitre présente une étude comparative des performances de trois types de codes LDPC : les codes LDPC construits pseudo-aléatoirement proposés par Gallager, les codes LDPC construits sur la base de la géométrie finie Euclidienne $EG(m, 2^s)$ sur un corps de Galois $CG(2^s)$ et les codes LDPC construits sur la base de la géométrie finie projective $PG(m, 2^s)$ sur un corps de Galois $CG(2^s)$.

Nous avons choisi ces trois types de codes pour pouvoir faire une comparaison entre les performances des codes LDPC avec une structure bien définie et les codes non-structurés. Les communications numériques (sans fil, téléphonie mobile, communications par satellite, enregistrement magnétique, etc.) demandent de plus en plus de grandes vitesses de traitement et de faibles complexités des circuits. C'est une des rai-

sons pour laquelle les codes avec une structure bien définie sont plus avantageux. De plus, leur distance minimale peut être déterminée. Mais il faut déterminer si leur performance est aussi bonne que celle des codes aléatoires, sans structure. Sinon, on doit faire un compromis entre la complexité du circuit, la vitesse de codage et décodage et la performance d'erreur. Les codes basés sur les géométries finies ont une très bonne distance minimale et leurs graphes de Tanner n'ont pas de cycles courts. Leur structure est cyclique ou quasi-cyclique, ce qui fait que leur codage est simple et peut être réalisé avec des registres à décalage linéaire. On prévoit améliorer les performances d'erreur des codes structurés (en améliorant le codeur ou le décodeur) et étudier des autres caractéristiques de ces codes qui peuvent nous aider dans le processus d'amélioration.

La performance des trois types de codes est analysée pour différents paramètres. Pour les trois types de codes, on transmet 1000 ou 10000 mots codes. Le codage est fait à l'aide de la formule $\bar{c} = \bar{m}\mathbf{G}$, où $\bar{m}$ est le message transmis, $\mathbf{G}$ est la matrice génératrice du code et $\bar{c}$ est le mot-code. Les bits codés sont modulés avec une modulation bipolaire BPSK et envoyés sur un canal à bruit additif blanc et gaussien (AWGN). Les séquences reçues à la réception ont la même longueur que les mot-codes envoyés, mais les symboles reçus ont une valeur réelle quelconque en raison du canal AWGN.

À la réception, le signal reçu est démodulé et ensuite décodé pour estimer le message initial. Pour la méthode de décodage on a utilisé l'algorithme de décodage SPA (algorithme somme-produit) dans sa version logarithmique [27]. L'algorithme de décodage s'arrête quand le nombre d'itérations prédéfini est atteint ou quand le syndrome $\bar{s} = \bar{c}\mathbf{H}'$

est nul.

La construction et le décodage des codes ont été effectués avec le logiciel MATLAB. On a calculé la probabilité d'erreur par bit (BER) pour chaque type de code et on a tracé les graphiques de BER en fonction de rapport signal à bruit (E_b/N_o) pour analyser leur performances et les comparer.

3.2 Performance des codes LDPC pseudo-aléatoires

Les codes LDPC pseudo-aléatoires sont construits tel que les trois conditions de la définition de codes LDPC donnée par Gallager soient satisfaites [14], [7]. Des analyses avec l'ordinateur sont nécessaires afin de trouver de tels codes (en particulier pour les codes très longs). Cette construction produit un ensemble de codes aléatoires LDPC et il a été démontré que ceux-ci conduisent à de bonnes performances [23].

On considère la construction d'un code LDPC de longueur n et de rendement k/n [14]. Pour construire la matrice de parité, on doit choisir un poids approprié de colonne γ et un nombre approprié de rangées J (dans la construction des codes avec l'ordinateur, J est choisi égal à $n-k$). Pour avoir un poids de rangée ρ constant pour toutes les lignes, on impose la condition :

$$\gamma \times n = \rho \times (n-k) \tag{3.1}$$

Si n est divisible par $n-k$, alors ρ est un multiple de γ et on peut construire une matrice de parité régulière $\mathbf{H}$ de faible densité avec la densité de colonne γ et la densité de ligne ρ.

Si n n'est pas divisible par $n-k$ alors :

$$\gamma \times n = \rho \times (n-k) + b \tag{3.2}$$

où b est le reste et ρ le quotient. Cette équation est équivalente à :

$$\gamma \times n = (n-k-b)\rho + b(\rho+1) \tag{3.3}$$

ce qui suggère que la matrice $\mathbf{H}$ peut avoir deux poids de rangée différents : ρ et $\rho+1$ (les b premières lignes ont un poids $(\rho+1)$ et les dernières $(n-k-b)$ lignes sont de poids ρ). Si b est égal à zéro, $\mathbf{H}$ sera une matrice régulière et on retrouve l'équation (3.1).

La construction de la matrice $\mathbf{H}$ est faite en plusieurs étapes. À chaque étape, une nouvelle colonne est ajoutée à une matrice formée partiellement. La colonne ajoutée est choisie entre plusieurs colonnes selon les conditions à respecter par la matrice de parité d'un code LDPC.

Si le rang de $\mathbf{H}$ est exactement $(n-k)$, le noyau de $\mathbf{H}$ donne un code LDPC (n,k) de rendement k/n. Si le rang de $\mathbf{H}$ est inférieur à $n-k$, son noyau donne un code LDPC $(n-k')$ de rendement supérieur à k/n $(k'/n > k/n)$. Le rendement du code construit a comme limite inférieure : $R \geq 1-\gamma/\rho$.

En raison des conditions imposées pendant la construction du code, son graphe de Tanner ne contient pas de cycles d'ordre 4 et alors son périmètre est égal à au moins 6. Cette construction n'est efficace que pour des petites valeurs de γ (3 ou 4).

Pour cette construction, il est très difficile de déterminer la distance minimale du code et pour $\gamma = 3$ ou 4, la limite inférieure $\gamma+1$ pour la distance minimale peut être très petite.

Les codes générés pseudo-aléatoirement par ordinateur n'ont pas les mêmes propriétés structurales que les codes à géométrie finie. En conséquence, l'implémentation matérielle est beaucoup plus complexe.

Ces codes peuvent être décodés avec l'algorithme somme-produit (SPA), mais la solution ne converge pas aussi rapidement que dans le cas des codes LDPC à géométrie finie.

Bien qu'ils aient ces inconvénients, ces codes conduisent à des performances très proches de la limite de Shannon. Par exemple, de très longs codes (10^7) ont été construits et il est déjà prouvé qu'ils ont des performances proches de seulement quelques millièmes de décibel de la limite du Shannon [28]. Il y a des codes (en particulier les codes aléatoires de grande longueur) qui conduisent à une meilleure performance d'erreur que les codes basés sur la géométrie finie.

3.3 Exemple numérique de code LDPC pseudo-aléatoire

Pour notre étude, on a choisi en premier un code LDPC pseudo-aléatoire avec les paramètres suivants : la longueur du code $n = 255$, la dimension $k = 170$, le rendement $R = 0.67$, le poids de colonne $\gamma = 3$ et et le poids de ligne $\rho = 9$.

Supposons que l'on veuille construire un code pseudo-aléatoire régulier de longueur $n = 255$, de poids de colonne $\gamma = 3$ et de rendement $R = 0.67$. On calcule la dimension du code (le nombre de bits d'information) k et le poids de ligne ρ. Pour avoir un code

régulier, on impose la condition suivante :

$$\gamma \times n = \rho \times (n - k) \tag{3.4}$$

Le nombre de bits d'information sera donc $k = R \times n = 0.67 \times 255 = 170$. Le poids de ligne sera :

$$\rho = \frac{n \times \gamma}{n - k} = \frac{n \times \gamma}{n(1 - \frac{k}{n})} = \frac{\gamma}{1 - R} = \frac{3}{0.33} = 9 \tag{3.5}$$

On construit alors la matrice de parité **H** selon l'algorithme itératif présenté auparavant :

1. on choisit une colonne de poids $\gamma = 3$ de manière aléatoire ;
2. à chaque étape, on génère de manière aléatoire une nouvelle colonne de poids 3 (formée par une permutation aléatoire des éléments de la première colonne) ;
3. si cette colonne respecte les 3 conditions de Gallager, elle est ajoutée à la matrice **H** formée partiellement.

On remarque pendant ce processus que cette méthode de générer la matrice **H** a pour effet de produire une "explosion" du nombre de calculs pour des dimensions élevées de **H**, en essayant de trouver des colonnes qui doivent satisfaire toutes les contraintes imposées. Parfois l'algorithme ne réussit pas à trouver une matrice régulière, en ajoutant des nouvelles colonnes aléatoires à chaque étape. On a donc amélioré l'algorithme, en générant les colonnes de manière *pseudo*-aléatoire, c'est-à-dire en imposant des contraintes de poids pendant la génération. Par exemple, dans la génération d'une colonne, on vérifie

si le poids de chaque ligne est déjà atteint. Si c'est le cas, on met dans la nouvelle colonne les valeurs 1 (de manière aléatoire) sur des positions qui correspondent aux lignes qui n'ont pas encore le poids ρ. Ainsi, on a réduit le nombre de calculs du programme, qui faisait la validation d'une nouvelle colonne après qu'elle soit ajoutée à la matrice **H**.

La matrice **H** ainsi formée a des lignes linéairement dépendantes et donc le rang de **H** sera plus petit que $n - k$. On a rencontré des difficultés dans la construction de la matrice génératrice du code **G** à partir de la matrice **H**. La matrice **G** est obtenue à partir de la forme systématique de la matrice de parité $\mathbf{H} = [\mathbf{I}_{n-k}|\mathbf{P}] => \mathbf{G} = [\mathbf{P}'|\mathbf{I}_k]$. Mais la fonction *rref* existant dans Matlab,qui calcule la forme systématique d'une matrice, n'est pas adaptée pour les calculs en binaire (cela veut dire qu'elle ne fait pas tous les calculs, à chaque étape, modulo 2). Donc on a reprogrammé la fonction *rref* de Matlab, pour faire la forme systématique d'une matrice binaire (comme la matrice de contrôle de parité).

On a fait des analyses des performances de ce type de codes en fonction de différents paramètres :

- la *longueur du code* ($n = 255$, $n = 510$ et $n = 1020$) (Fig. 3.1) en gardant les autres paramètres constants (le rendement du code, le poids de ligne et le poids de colonne, le nombre d'itérations faits pour le décodage log-SPA et le nombre de mots-codes envoyés). Plus la longueur du code est grande, plus on obtient des meilleures performances. Cependant, ces codes demandent plus de temps pour le décodage et pour la génération de la matrice **H** ;
- *le poids de la colonne* ($\gamma = 3$ ou 4) (Fig. 3.2), (Fig. 3.3) - plus le poids de la colonne

est grand, plus on obtient de mauvaises performances. La distance minimale pour ce type de codes a comme limite inférieure $\gamma + 1$ et donc pour des petites valeurs de γ on obtient des codes avec une petite distance minimale entre les mots-codes et ceci n'est pas avantageux pour le décodage. On doit donc faire un compromis en choisissant la valeur du poids de la colonne ;

- *le nombre maximum d'itérations* pour le décodage SPA (améliorations des performances avec l'augmentation du nombre d'itérations permis). Nous avons fait des simulations pour 10, 30 et 100 itérations (Fig. 3.4).

On a essayé de mettre en évidence les effets du décodage SPA dans la reconstruction d'une image à chaque itération. Pour ce faire, on a construit une image de dimension 90×170, codée avec un code LDPC pseudo-aléatoire (255,170), i.e. de poids de colonne $\gamma = 3$ et de rendement $R = 0.67$. Cette image est ensuite modulé en BPSK et transmise sur un canal AWGN avec un rapport signal à bruit de 3 dB. À la réception, l'image reçue était décodée après 30 itérations du décodage SPA dans sa version logarithmique. La figure 3.5 montre les images à chaque itération du décodage.

3.4 L'algorithme de décodage SPA modifié

L'algorithme de décodage peut être arrêté dans deux situations : soit que l'on trouve un syndrome nul, soit qu'il s'arrête au nombre maximum d'itérations imposé (même si on n'a pas trouvé le bon mot-code). On a remarqué que les performances ne deviennent pas nécessairement meilleures en augmentant le nombre d'itérations faites

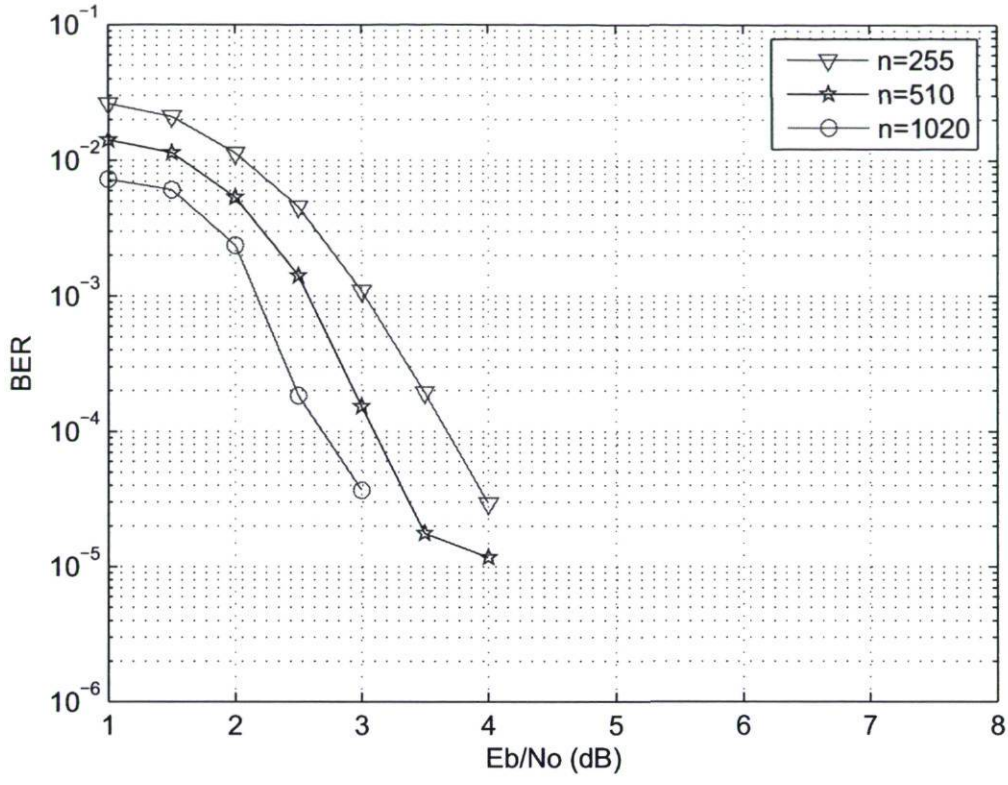


FIG. 3.1 – Probabilités d'erreur par bit des trois codes LDPC pseudo-aléatoires de même rendement ($R = 0.67$), de même poids sur la colonne ($\gamma = 3$), mais de longueurs différentes : $n = 255$, $n = 510$ et $n = 1020$. On transmets 1000 mot-codes qui sont décodés avec l'algorithme log-SPA, après 50 itérations.

dans le décodage SPA. Après un certain nombre d'itérations, le nombre de bits erronés dans le syndrome peut augmenter, ce qui peut aussi augmenter le nombre de bits erronés dans le mot-code estimé. Ceci signifie qu'à la fin des itérations, on ne prend pas nécessairement la meilleure décision (le mot-code estimé le plus proche du mot-code envoyé). Pour ce faire, on a modifié l'algorithme de décodage pour le cas où il est arrêté par le nombre maximum d'itérations imposé. On a imposé la condition que lorsqu'il

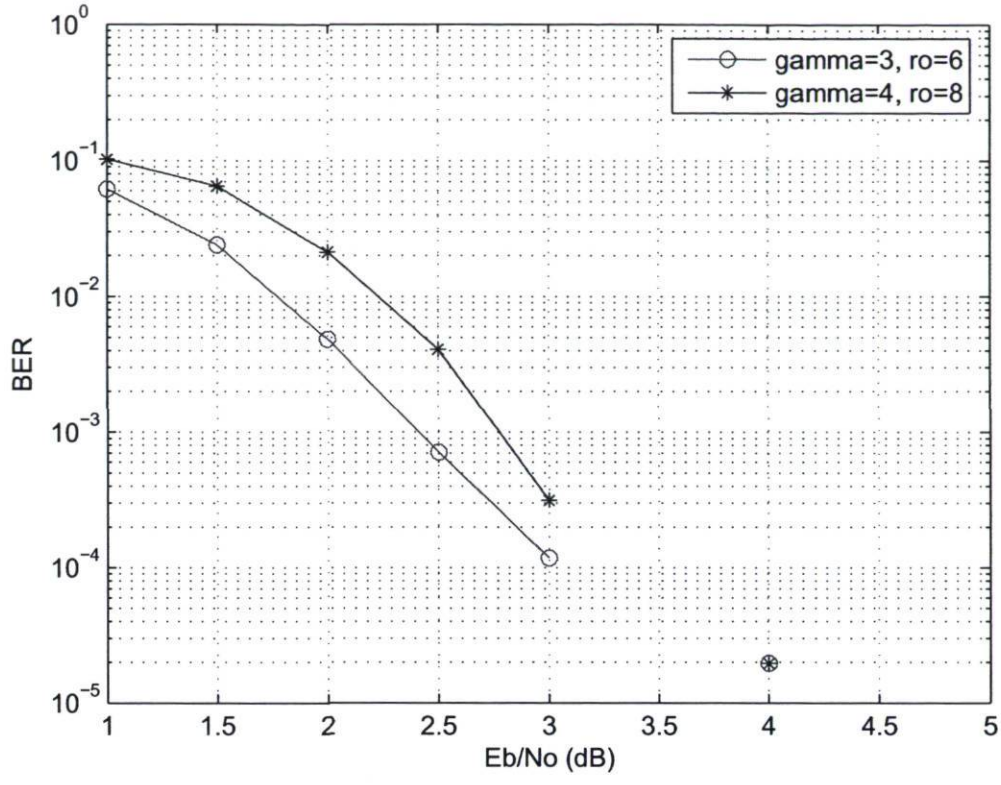


FIG. 3.2 – Influence du poids de la colonne ($\gamma = 3$ et 4) sur la performance des codes pseudo-aléatoires (simulation avec un code de longueur $n = 510$, rendement $R = 1/2$). On transmets 1000 mots-codes qui sont décodés après 30 itérations avec l'algorithme SPA.

arrive à la fin des itérations, le mot-code estimé est celui qui correspond au syndrome le plus proche (au sens de la distance de Hamming) du syndrome nul (c'est-à-dire le syndrome de poids minimal). On a supposé qu'il y a une liaison entre le poids du vecteur d'erreur et le poids du syndrome. La dépendance entre les poids des vecteurs syndromes et des vecteurs d'erreur fait l'objet du chapitre 5.

L'algorithme de décodage modifié permet d'obtenir de meilleures performances d'er-

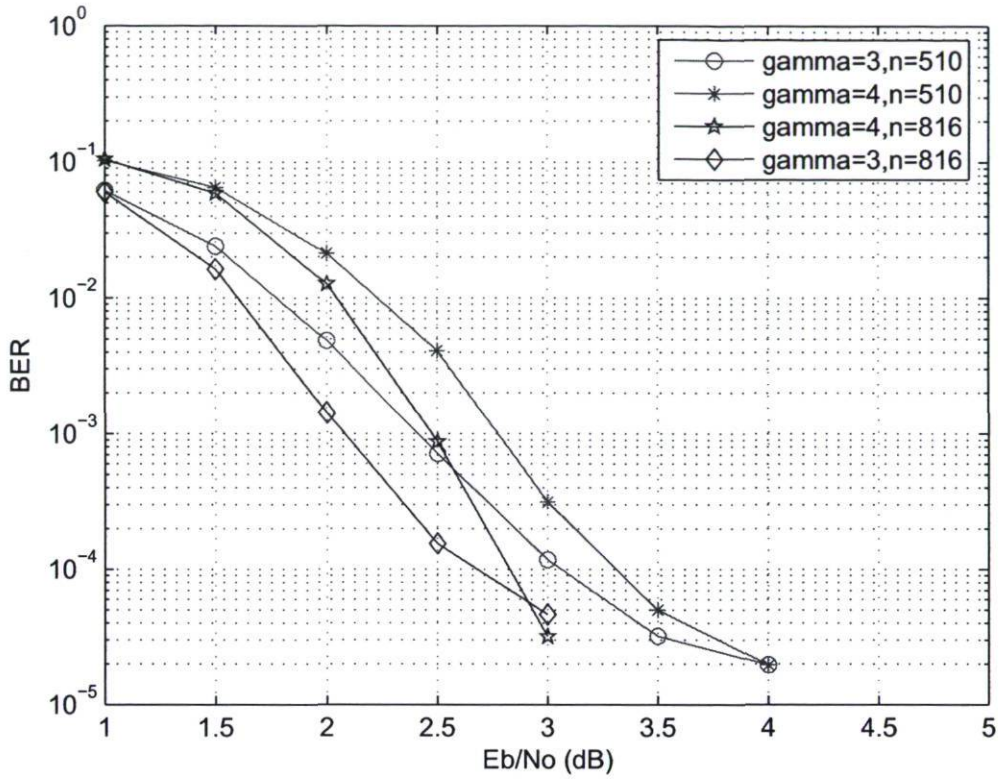


FIG. 3.3 – Influence du poids de la colonne ($\gamma = 3$ et 4) sur la performance des codes pseudo-aléatoires (comparaison pour deux codes de longueurs différents $n = 816$ et $n = 510$ et rendement $R = 1/2$. On transmets 1000 mots-codes qui sont décodés après 50 itérations de l'algorithme SPA.

reur pour les codes LDPC étudiés. On peut observer dans la figure (Fig. 3.6) les différences entre les courbes des performances avec l'algorithme log-SPA et l'algorithme log-SPA modifié. On a décodé un code pseudo-aléatoire de longueur $n = 255$ par les deux types d'algorithmes, en gardant les autres paramètres pareils. Pour toute la plage des valeurs du rapport signal à bruit, les probabilités d'erreur par bit obtenues avec l'algorithme log-SPA modifié sont plus faibles que celles obtenues avec l'algorithme

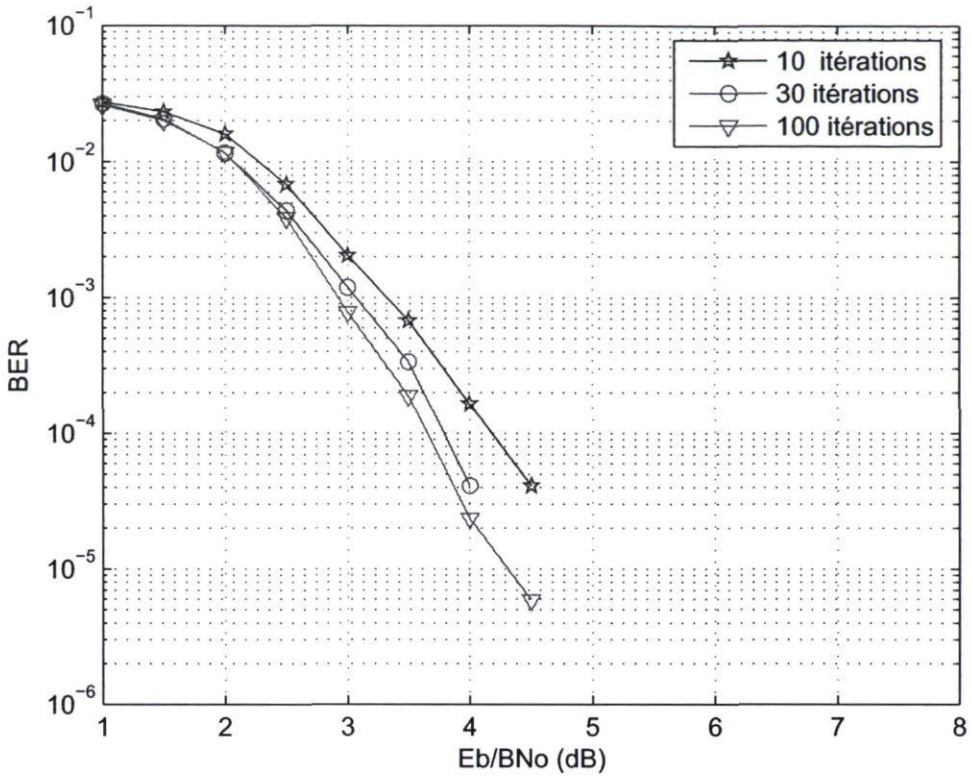


FIG. 3.4 – Code LDPC pseudo-aléatoire (255,170) : comparaison entre le décodage SPA-log modifié, fait en 10, 30 et 100 itérations.

log-SPA normal.

3.5 Codes LDPC basés sur la géométrie finie euclidienne

Les codes LDPC basés sur la géométrie finie présentent, pour des rendements élevés (i.e. près de l'unité), de très bonnes performances avec l'algorithme de décodage SPA. Aucun plancher d'erreur n'est observé pour des taux d'erreur $\geq 10^{-6}$ (Fig. 3.8).

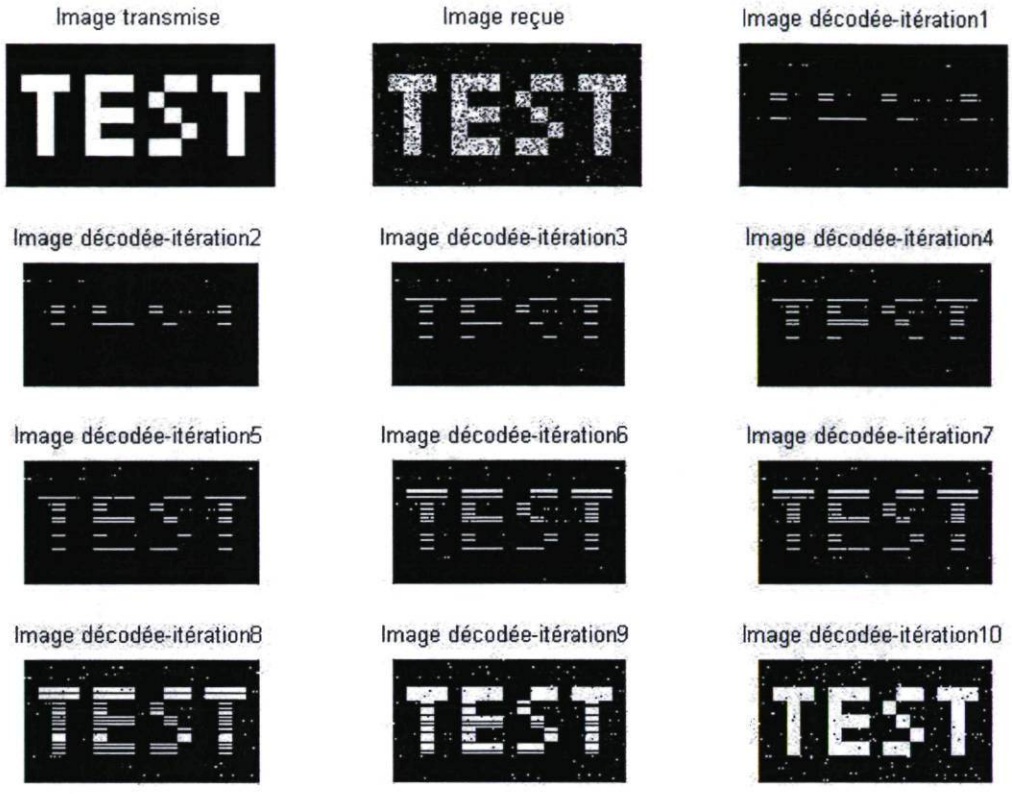


FIG. 3.5 – Le décodage SPA-log d'une image codée LDPC - l'évolution du décodage à chaque itération.

La géométrie finie $EG(m, 2^s)$ consiste en 2^{ms} points et chaque point est représenté par un m-tuple sur $CG(2^s)$. Par chaque point passent γ lignes. Chaque ligne comprends 2^s points (poids de ligne : $\rho = 2^s$). À partir de cette géométrie finie, on peut construire un code avec la matrice de parité $\mathbf{H}$, ayant ses éléments choisis selon les points et les lignes dans cette géométrie finie. Il existe deux types de codes basés sur la géométrie finie $EG(m, 2^s)$: type I et type II (en fonction du choix de correspondance de points et de lignes de la géométrie avec les colonnes et les lignes de la matrice de parité $\mathbf{H}$ du code).

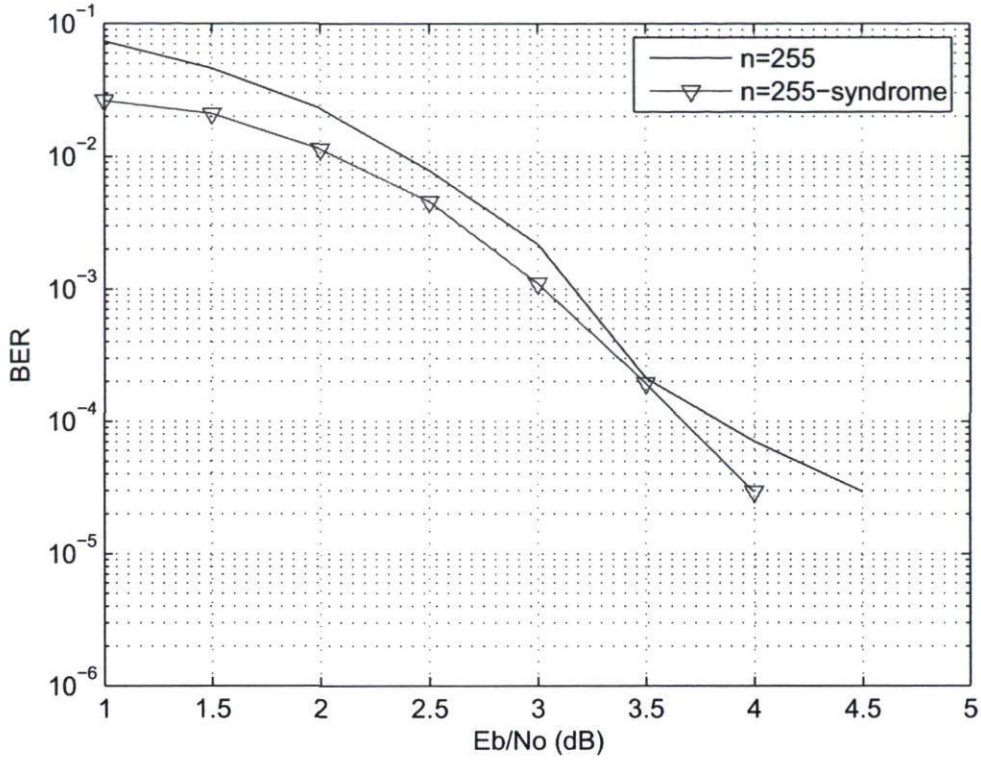


FIG. 3.6 – Influence de la modification de l'algorithme log-SPA sur la performance des codes pseudo-aléatoires de longueur $n = 255$, de poids $\gamma = 3$ et $\rho = 9$ et de rendement $R = 0.67$. On transmet 1000 mots-codes qui sont décodés après 30 itérations de l'algorithme SPA.

Pour construire un code LDPC à géométrie euclidienne finie de type I basé sur $EG(m, 2^s)$, les lignes de la matrice de contrôle de parité sont les vecteurs d'incidence de toutes les lignes dans $EG(m, 2^s)$ et les colonnes du $\mathbf{H}$ correspondent aux points dans $EG(m, 2^s)$. Le poids de ligne de $\mathbf{H}$ sera ρ et le poids de colonne γ .

Dans la construction des codes LDPC à géométrie euclidienne basés sur $EG(m, 2^s)$,

on peut éliminer le point d'origine de la géométrie et toutes les lignes qui passent par l'origine. Ainsi on met le code EG-LDPC de type 1 dans une **forme cyclique**. Cela simplifie la conception du circuit de codage. Les paramètres d'un code EG-LDPC cyclique de type 1 bi-dimensionnel sont [14] :

- longueur : $n = 2^{2s} - 1$
- nombre de bits de parité : $n - k = 3^s - 1$
- dimension : $k = 2^{2s} - 3^s$
- distance minimale : $d_{min} = 2^s + 1$
- densité : $r = \frac{2^s}{2^{2s}-1}$

Pour notre étude on a choisi un code EG-LDPC de type 1 avec $m = 2$ et $s = 4$. On obtient donc un code LDPC construit dans la géométrie euclidienne $EG(2, 2^4)$ sur le corps de Galois $CG(2^4)$. Ce code EG-LDPC de type 1, cyclique et bi-dimensionnel $C_{EG}^{(1)}(2, 0, s)$ a les paramètres suivants :

- longueur : $n = 2^{2\times4} - 1 = 255$
- nombre de bits de parité : $n - k = 3^4 - 1 = 80$
- dimension : $k = 2^{2\times4} - 3^4$=175
- distance minimale : $d_{min} = 2^4 + 1 = 17$
- densité : $r = \frac{2^4}{2^{2\times4}-1} = 0.0627$

Le graphe de Tanner correspondant à ce code a un périmètre égal à 6, ce qui veut dire qu'il ne contient pas de cycles courts (d'ordre 4). Le rendement de ce code est $R = k/n = 0.68$. La matrice de parité $\mathbf{H}$ du code est une matrice régulière et carrée (facile à implémenter en pratique!), de dimensions $(2^{2s} - 1) \times (2^{2s} - 1)$. Le poids de

chaque ligne est le même que le poids de chaque colonne ($\gamma = \rho = 2^s$). Dans notre cas, $\gamma = \rho = 2^4 = 16$. Tenant compte que la matrice $\mathbf{H}$ d'un tel code est carrée, on peut remarquer qu'on ne doit garder en mémoire que la première rangée de $\mathbf{H}$, les autres rangées seront ajoutées à l'aide d'un registre à décalage linéaire (par des permutations circulaires de la première rangée). Cela implique une faible complexité du codeur et du décodeur, ce qui représente un grand avantage dans l'implémentation pratique.

La construction de la matrice de parité de notre code est réalisée comme suit. Avec $m = 2$ et $s = 4$, notre code est basé sur la géométrie finie euclidienne $EG(2, 2^4)$ sur le corps du Galois $CG(2^{ms}) = CG(256)$, qui est généré par le polynôme primitif suivant :

$$p(X) = 1 + X + X^2 + X^3 + X^4 + X^8 \tag{3.6}$$

On génère le corps de Galois $CG(256)$ sous la forme d'une matrice de parité $\mathbf{H}$ de dimensions 256×8 (tous les vecteurs binaires de longueur 8). On identifie les éléments du corps de Galois étendu $CG(2^{2\times4})$, qui sont aussi dans le corps $CG(2^s) = CG(2^4) = CG(16)$, en imposant la condition :

$$(\alpha^x)^{16} = (\alpha^x) \mod 255 = \alpha^{x+255} \tag{3.7}$$

où α est un élément primitif de $CG(2^{2\times4})$. Donc $x = 17$ et soit $\beta = \alpha^{17}$. L'ensemble $E = \{0, 1, \beta, \beta^2, \beta^3, ..., \beta^{14}\}$ forme le sous-espace du corps de Galois $CG(2^4)$. Chaque ligne dans $EG(2, 2^4)$ a 16 points. L'ensemble $\{\alpha^{254} + \gamma\alpha\}$, avec $\gamma \in E$, constitue dans $EG(2, 2^4)$ une ligne qui ne passe pas par l'origine. Les positions des valeurs de 1 dans la première ligne sont indiquées par les puissances des éléments de l'ensemble $\{\alpha^{254} + \gamma\alpha\}$. La matrice $\mathbf{H}$ est ensuite formée par 254 permutations circulaires de cette ligne.

Dans la construction de la matrice génératrice du code, on rencontre des difficultés, dues à la forme carrée de la matrice de parité. On doit calculer le rang de la matrice **H** pour déterminer si le nombre de bits de parité coïncide bien avec le nombre calculé par la formule donnée auparavant : $n - k = 3^4 - 1 = 80$. Les dimensions de la matrice **H** étant très grandes, on a utilisé Matlab pour les calculer, mais aussi comme pour la fonction *rref* de Matlab, la fonction *rank* ne fait pas les calculs modulo 2 (juste avec de nombres binaires). Pour éliminer ce problème, on a transformé en premier la matrice **H** dans sa forme systématique par la fonction qu'on créé (*rref* modulo 2), et ensuite on calcule le rang avec la fonction *rank*.

La figure 3.7 montre, pour un code EG-LDPC, la différence entre les courbes de performances qu'on obtient avec 1000 et 10000 mots-codes transmis. Avec 1000 mots-codes la courbe de probabilité d'erreur s'arrête à $7.7 * 10^{-5}$ pour $E_b/N_0 = 3.5$ dB et pour 10000 mots-codes elle arrive jusqu'à $2.8 * 10^{-6}$ pour un rapport signal à bruit de 4 dB. Donc pour une bonne estimation du plancher d'errer il faut envoyer un nombre suffisamment grand de mot-codes.

3.6 Codes LDPC basés sur la géométrie projective

Comme pour la géométrie Euclidienne, la géométrie projective aussi peut être construite à partir d'un corps de Galois [14]. On considère l'élément primitif du corps de Galois $CG(2^{(m+1)s})$ qui contient comme sous-corps le corps de Galois $CG(2^s)$. Les puissances de α : $\alpha^0, \alpha^1, ..., \alpha^{2^{(m+1)s}}$ forment tous les éléments non-nuls du corps de Galois

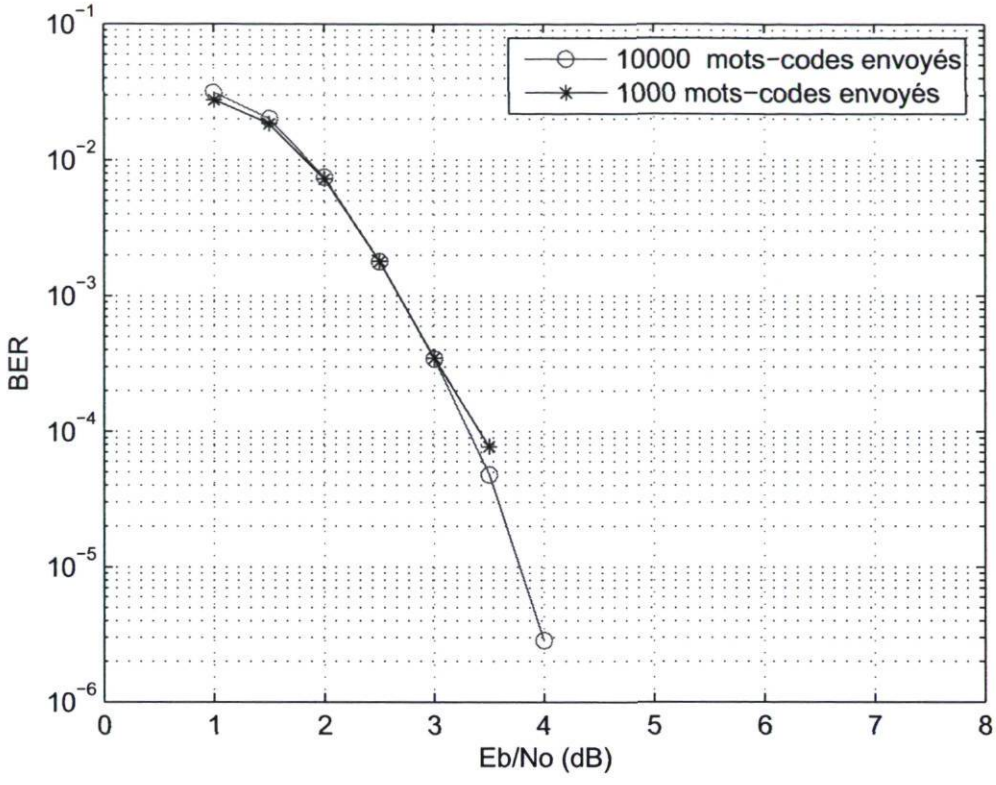


FIG. 3.7 – Influence de nombre de mots-codes transmis pour les courbes de performances d'un code EG-LDPC.

étendu $CG(2^{(m+1)s}$. Soit

$$n = \frac{2^{(m+1)s} - 1}{2^s - 1} = 2^{ms} + 2^{(m-1)s} + ... + 2^s + 1 \tag{3.8}$$

L'ordre de l'élément $\beta = \alpha^n$ est $2^s - 1$. Les 2^s éléments $0, 1, \beta, \beta^2, ..., \beta^{2^s-1}$ forment le corps du Galois $CG(2^s)$. On prend juste l'ensemble des premières puissances de α :

$$\Gamma = \left\{\alpha^0, \alpha^1, \alpha^2, ..., \alpha^{n-1}\right\} \tag{3.9}$$

On regroupe tous les éléments non-nuls de $CG(2^{(m+1)s}$ dans n sous-ensembles

différents :

$$(\alpha^i) \triangleq \left\{\alpha^i, \beta\alpha^i, ..., \beta^{2^s-2}\alpha^i\right\}$$

avec $0 \leq i < n$. Le $(m+1)$-tuple sur $CG(2^s)$ représentant (α^i) peut être vu comme un point dans la géométrie finie sur $CG(2^s)$. Donc les n points

$$(\alpha^0), (\alpha^1), (\alpha^2), ..., (\alpha^{n-1})$$

forment une géométrie projective m-dimensionnelle $PG(m, 2^s)$ sur le corps de Galois $CG(2^s)$. Dans cette géométrie, les $2^s - 1$ éléments dans $\left\{\alpha^i, \beta\alpha^i, ..., \beta^{2^s-2}\alpha^i\right\}$ sont considérés comme le même point dans $PG(m, 2^s)$. Cette propriété constitue la grande différence entre la géométrie projective et la géométrie euclidienne. Une géométrie projective n'a pas de point d'origine.

Il y a J lignes dans $PG(m, 2^s)$, où

$$J = \frac{(2^{(m-1)s} + ... + 2^s + 1)(2^{ms} + ... + 2^s + 1)}{2^s + 1}$$

Par chaque point (α^i) passent $\gamma = \frac{2^{ms}-1}{2^s-1}$ lignes.

On forme une matrice de parité $\mathbf{H}_{PG}^{(1)}$ dont les lignes sont les vecteurs d'incidence des lignes dans $PG(m, 2^s)$ et les colonnes correspondent aux points de $PG(m, 2^s)$. La matrice est donc de dimensions $(n \times J)$, son poids par colonne est γ et le poids par rangée est $\rho = 2^s + 1$. La densité de la matrice $\mathbf{H}_{PG}^{(1)}$ sera

$$r = \frac{\rho}{n} = \frac{(2^s - 1)(2^s + 1)}{2^{(m+1)s} - 1}.$$

Le noyau de la matrice $\mathbf{H}_{PG}^{(1)}$ donne un code PG-LDPC m-dimensionnel de type I $C_{PG}^{(1)}(m, 0, s)$ de longueur n et une distance minimale $d_{min} \geq \gamma + 1$. Le code formé est

cyclique.

Les paramètres d'un code PG-LDPC bi-dimensionnel de type I $C_{PG}^{(1)}(2,0,s)$ sont :

- longueur : $n = 2^{2s} + 2^s + 1$
- nombre de bits de parité : $n - k = 3^s + 1$
- dimension : $k = 2^{2s} + 2^s - 3^s$
- distance minimale : $d_{min} = 2^s + 2$
- densité : $r = \frac{2^s+1}{2^{2s}+2^s+1}$

La matrice de contrôle de parité $\mathbf{H}_{PG}^{(1)}$ de ce code est une matrice carrée, qui peut être obtenue en choisissant le vecteur d'incidence d'une ligne dans $PG(2,2^s)$ et en faisant ses $2^{2s} + 2^s$ permutations circulaires pour obtenir les lignes de la matrice.

Pour notre étude, on a choisi un code PG-LDPC de type 1 avec $m = 2$ et $s = 4$. Ce code LDPC est construit avec la géométrie projective $PG(2,2^4)$ sur le corps de Galois $CG(2^4)$. On obtient un code PG-LDPC de type I, cyclique et bi-dimensionnel $C_{PG}^{(1)}(2,0,s)$ ayant les paramètres suivants :

- longueur : $n = 2^{2\times4} + 2^4 + 1 = 273$
- nombre de bits de parité : $n - k = 3^4 + 1 = 82$
- dimension : $k = 2^{2\times4} + 2^4 - 3^4 = 191$
- distance minimale : $d_{min} = 2^4 + 2 = 18$
- densité : $r = \frac{2^4+1}{2^{2\times4}+2^4+1} = 0.0623$

3.7 Comparaison entre les trois types de codes

En comparant les codes pseudo-aléatoires avec les codes basés sur les géométries finies euclidienne ou projective (le code EG-LDPC et le code PG-LDPC), pour des longueurs courtes, avec des rendements semblables, 10000 mots-codes transmis et décodage avec l'algorithme SPA, on obtient de meilleurs résultats avec les codes basés sur la géométrie finie (Fig. 3.8). Les codes basés sur les géométries finies ont un plancher d'erreur beaucoup plus bas. On doit préciser que pour les codes pseudo-aléatoires, il est toujours préférable d'imposer un nombre d'itérations beaucoup plus grand (jusqu'à 200 itérations), comparativement au nombre d'itérations nécessaires pour obtenir de bonnes performances avec les codes basés sur les géométries finies EG-LDPC ou PG-LDPC. Par exemple, pour un code EG-LDPC de dimensions (4097, 3367) la différence de performance entre 5 et 100 itérations est de 0.1 dB. Donc, en pratique, si la vitesse de décodage est petite, 5 itérations peuvent être suffisants. Dans nos simulations, pour des codes FG-LDPC (basés sur la géométrie projective) de longueurs de quelques centaines de bits on utilise 50 itérations. Le décodage des codes basés sur la géométrie finie a donc l'avantage de pouvoir être réalisé plus rapidement, en raison d'un plus petit nombre d'itérations nécessaires pour le décodage de chaque mot-code envoyé.

Pour de grandes longueurs des mots-codes, les codes pseudo-aléatoires ont de meilleures performances que les codes basés sur la géométrie finie avec le décodage SPA, mais leur codage est très complexe à cause de leur manque de structure (e.g. ni cyclique, ni quasi-cyclique). Un autre inconvénient des codes pseudo-aléatoires réside dans la difficulté de trouver leurs distances minimales.

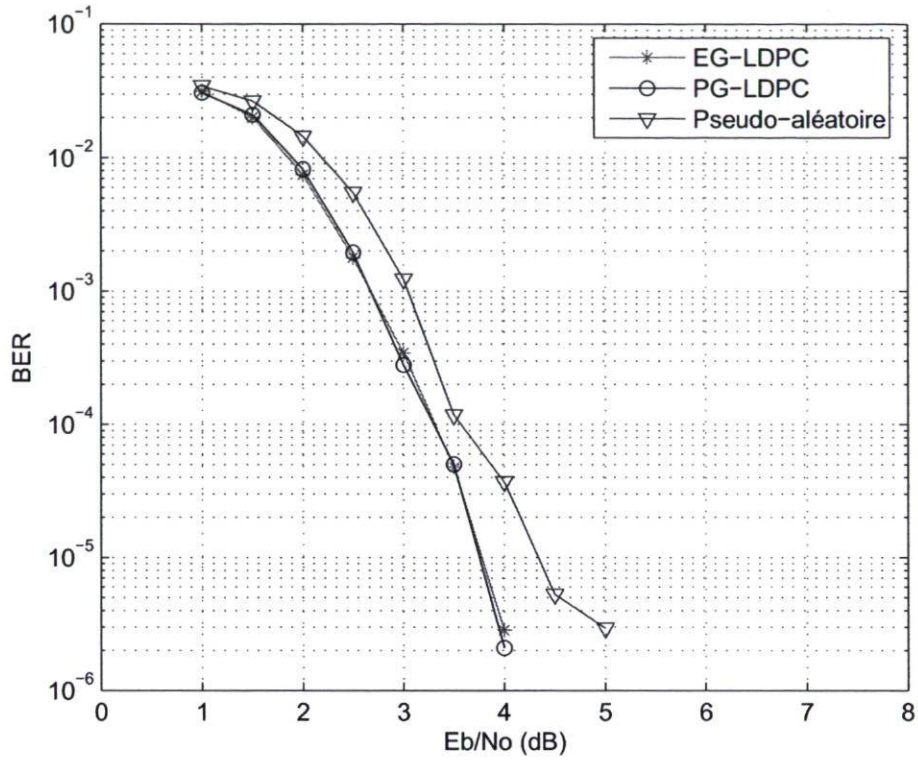


FIG. 3.8 – Comparaison entre les performances des trois types de codes LDPC : pseudo-aléatoire, EG-LDPC type I et PG-LDPC type I - Un code LDPC pseudo-aléatoire (255, 175) avec $\gamma = 3$, $\rho = 9$, $R = 0.67$, un code EG-LDPC (255, 170) et un code PG-LDPC (273, 191) ; 10000 mots-codes, SPA-log, 100 itérations.

3.8 Codes EG-LDPC prolongés

Un code LDPC basé sur la géométrie finie et ayant la longueur n peut être prolongé en dédoublant chaque colonne de sa matrice $\mathbf{H}$ dans plusieurs colonnes. On obtient alors une nouvelle matrice de contrôle de parité $\mathbf{H}_{ext}$ de densité plus petite, qui donne un nouveau code LDPC. On obtient des codes de longueurs, de rendements et de performances différents. Si les colonnes sont dédoublées adéquatement, on peut obtenir

de très bons codes LDPC (quelques codes LDPC prolongés arrivent à de très bonnes performances d'erreur avec le décodage SPA - à juste quelques fractions d'un décibel de la limite de Shannon). La raison pour laquelle le dédoublement des colonnes conduit à de meilleures performances d'erreur est que ce dédoublement des colonnes réduit le degré de chaque noeud de bit codé du graphe de Tanner et ainsi réduit le nombre de cycles dans le graphe. On a appliqué cette méthode pour le code EG-LDPC (255,175), avec différents facteurs d'extension du code (2, 3, 4, 5). On voit les résultats obtenus à la figure Fig. 3.9.

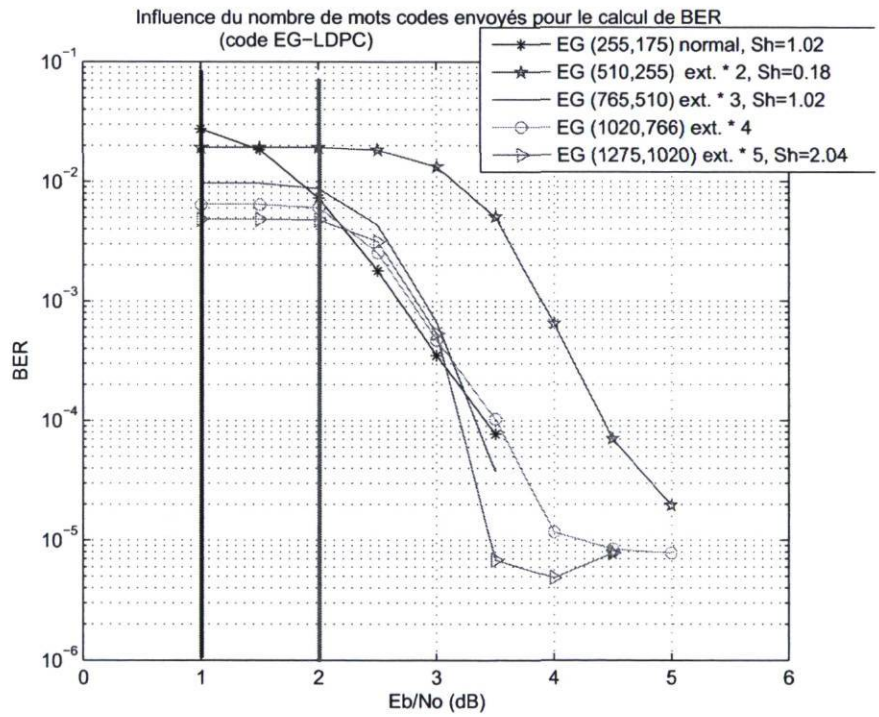


FIG. 3.9 – Performance des différents codes EG-LDPC obtenus par dédoublement de colonnes.

En fonction du rendement du code, on a des limites de Shannon différentes et donc

on juge différemment les performances. Par exemple le code prolongé avec le facteur 5 a sa limite de Shannon à 2.045 dB. Donc il est plus performant que le code initial (qui avait la limite de Shannon de 1.02 dB), parce que la courbe de probabilité d'erreur est plus proche de la limite de Shannon correspondante. On choisit le meilleur code selon les critères prioritaires en chaque cas (taux du code, longueur du code, temps de traitement, etc.). On obtient le plancher d'erreur le plus bas avec le facteur d'extension 5, parce qu'ainsi, on détruit le plus de cycles dans le graphe de Tanner du code. Cette courbe de probabilité d'erreur monte à la fin parce que nous n'avons pas envoyé suffisamment des mots-codes dans notre simulation pour cette valeur du rapport signal à bruit.

3.9 Conclusion

Dans ce chapitre, on a présenté les résultats comparatifs obtenus en étudiant trois types de codes LDPC de longueur courte et on a proposé un algorithme de décodage SPA modifié qui aide à obtenir de meilleures performance d'erreur dans le cas des codes pseudo-aléatoires étudiés.

On a fait la comparaison entre un code LDPC pseudo-aléatoire proposé par Gallager et deux types de codes LDPC structurés : un code LDPC basé sur la géométrie euclidienne et un code LDPC basé sur la géométrie projective. Les trois codes étaient de dimensions et rendements semblables pour pouvoir comparer leur performance. On a conclu que les codes étudiés, basés sur les géométries finies, ont une meilleure performance d'erreur que les codes pseudo-aléatoires de type Gallager. Cela est valide pour les codes LDPC courts. En plus, les codes basés sur les géométrie finies permettent

d'être décodés beaucoup plus rapidement et nécessitent des circuits beaucoup moins complexes pour leur codage. Cela sera la raison pour laquelle on va focaliser notre recherche sur les codes courts structurés, qui peuvent être utiles dans les communications numériques de nos jours qui demandent de petits circuits, de grandes vitesses et de bonnes performances (i.e. les communications cellulaires).

On a aussi analysé les performances des codes obtenues par dédoublement de colonnes d'un code EG-LDPC avec différents facteurs d'extension. On obtient des codes de différents rendements et dimensions, qu'on peut choisir d'utiliser en fonction de l'application.

Chapitre 4

Codes LDPC quasi-cycliques

4.1 Introduction

Les codes LDPC structurés construits par des sous-matrices de permutation circulaires sont de plus en plus étudiés par les chercheurs. Leur structure simple se traduit par une importante réduction de la complexité des codeurs et décodeurs. Différentes méthodes de construction des codes quasi-cycliques LDPC irréguliers sont proposées et analysées dans le but d'améliorer les performances d'erreur et en même temps de garder le plus possible une faible complexité des circuits.

4.2 Construction des codes LDPC quasi-cycliques

Dans [29] et [30], les auteurs ont proposé des codes LDPC quasi-cycliques irréguliers avec une structure triangulaire et doublement-diagonale afin d'obtenir un plancher d'erreur très bas (au dessous de 10^{-9}) et une basse complexité du codage (seulement

quelques centaines de portes XOR). Les codes proposés ont aussi une très bonne performance dans la région de rapports signal à bruit petits. Due à la faible complexité de codage linéaire, les codeurs de ces codes peuvent atteindre 10 Gbit/s. Ils sont utilisés dans les applications haute-vitesse comme les canaux optiques de taux de transmission de 20 Gbits/s ou les applications Ethernet Gigabit.

La matrice de parité est formée de deux parties : $\mathbf{H} = [\mathbf{H}_s|\mathbf{H}_p]$. $\mathbf{H}_s$ contient des matrices circulaires carrées, nulles ou non-nulles, et la matrice $\mathbf{H}_p$ a une forme doublement-diagonale et triangulaire (cette forme augmente la distance minimale du code).

$$\mathbf{H}_s = \begin{bmatrix} \mathbf{H}_{1,1} & \mathbf{H}_{1,2} & \cdots & \mathbf{H}_{1,L-J} \\ \mathbf{H}_{2,1} & \mathbf{H}_{2,2} & \cdots & \mathbf{H}_{2,L-J} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{J,1} & \mathbf{H}_{J,2} & \cdots & \mathbf{H}_{J,L-J} \end{bmatrix} \tag{4.1}$$

et

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \\ \mathbf{H}_{J-1,L-J+1} & \cdots & \mathbf{H}_{J-1,L-2} & \mathbf{I} & \mathbf{I}(q-1) \\ \mathbf{H}_{J,L-J+1} & \cdots & \mathbf{H}_{J,L-2} & \mathbf{I} & \mathbf{I} \end{bmatrix} \tag{4.2}$$

où $\mathbf{I}$ et $\mathbf{0}$ sont respectivement la matrice identité et la matrice nulle de dimension $(q \times q)$. $\mathbf{I}(q-1)$ est une matrice circulaire $(q \times q)$ obtenue par les permutations circulaires de $\mathbf{I}$ à droite avec $(q-1)$ positions et en enlevant le bit de la première rangée dans le but de garder le poids de la dernière colonne égal à 1. Pour $1 \leq j \leq J$ et $1 \leq l \leq L$, la sous-matrice $\mathbf{H}_{j,l}$ située dans la position (j,l) dans la matrice $\mathbf{H}$ est soit une matrice nulle, soit une matrice circulaire obtenue par des permutations circulaires à droite des

lignes de la matrice $\mathbf{I}$ avec $C_{j,l}$ positions. Le code défini par la matrice $\mathbf{I}$ a la longueur $n = q \times L$ et un rendement $R = 1 - J/L$. Les coefficients de permutation sont choisis aléatoirement mais en évitant les cycles courts dans le graphe de Tanner. Les cycles courts déterminent le fonctionnement sous-optimal de l'algorithme du décodage SPA. Pour éviter les cycles de longueur $2i$, une condition nécessaire et suffisante est [29] :

$$\sum_{k=1}^{m} (C_{j_k,l_k} - C_{j_{k+1},l_k}) \neq 0 \mod q \tag{4.3}$$

pour tous les m, $2 \leq m \leq i$, tous les j_k, $1 \leq j_k \leq J$, tous les j_{k+1}, $1 \leq j_{k+1} \leq J$ et tous les l_k, $1 \leq l_k \leq L$, avec $j_0 = j_m$.

Les quatre sous-matrices $\begin{bmatrix} \mathbf{I} & \mathbf{I}(q-1) \\ \mathbf{I} & \mathbf{I} \end{bmatrix}$ peuvent former une matrice double-diagonale par permutations des lignes et des colonnes.

Le plancher d'erreur est influencé par la distribution de poids des colonnes et par les coefficients de permutation. Afin de baisser le plancher d'erreur, on applique les règles suivantes :

1. on met des colonnes avec grands poids dans la matrice $\mathbf{H}_s$ (où on n'accepte pas de poids 1 ou 2) ;
2. les coefficients de permutation sont choisis à partir des colonnes de petits poids jusqu'à des colonnes avec grands poids. Les coefficients pour les colonnes de poids 2 ou 3 sont choisis en évitant les cycles de longueurs petite et moyenne (i.e. 4, 6, 8 et 10) et les coefficients pour les colonnes de poids supérieur à 3 sont choisis en évitant les cycles de petite longueur (i.e., 4 et 6).

Il est très important de bien choisir aussi les positions des sous-matrices non-nulles

dans la matrice **H** avant de choisir les coefficients de permutation de chaque sous-matrice. Leurs positions sont choisies en considérant les colonnes-blocs (formés par chaque colonne des sous-matrices). Entre chaque deux colonnes-blocs aléatoires de degré 3, il faut éviter les cycles-blocs de longueur 4 et entre chaque deux colonnes-blocs aléatoires de degré 4 il faut éviter les cycles-blocs doubles de longueur 4. Pour éliminer les cycles-blocs de longueur 4 ou les cycles-blocs doubles de longueur 4 entre deux colonnes-blocs arbitraires avec petit degré, il faut avoir $J > 10$ rangées-blocs. Il est aussi plus avantageux d'avoir plus de sous-matrices de dimension petite, que d'avoir moins de sous-matrices plus grandes [30].

Par l'optimisation de la distribution des degrés et de la structure des cycles simultanément pendant la construction du code, on élimine les cycles courts du graphe de Tanner et ainsi on élimine plusieurs ensembles de piégeage et ensembles d'arrêt dans le graphe, ce qui aide à réduire le plancher d'erreur.

La performance d'erreur pour la région de cascade de la courbe peut être améliorée par l'augmentation des poids de colonne-bloc et par la réduction du nombre des colonnes-blocs de grand degré.

Dans [31], Doré et al. ont aussi étudié les codes quasi-cycliques LDPC (QC-LDPC) basés sur les matrices de permutations circulaires. Ils proposent un algorithme de construction des codes avec de très grands périmètres et structures de cycles du graphe de Tanner. Ils trouvent aussi une limite pour la distance minimale de ce type de code et aussi une relation entre la distance minimale, le poids des colonnes et le nombre des sous-matrices-blocs. On peut tenir compte de ces paramètres lorsqu'on veut construire un code quasi-cyclique avec une certaine distance minimale. Leur matrice est aussi

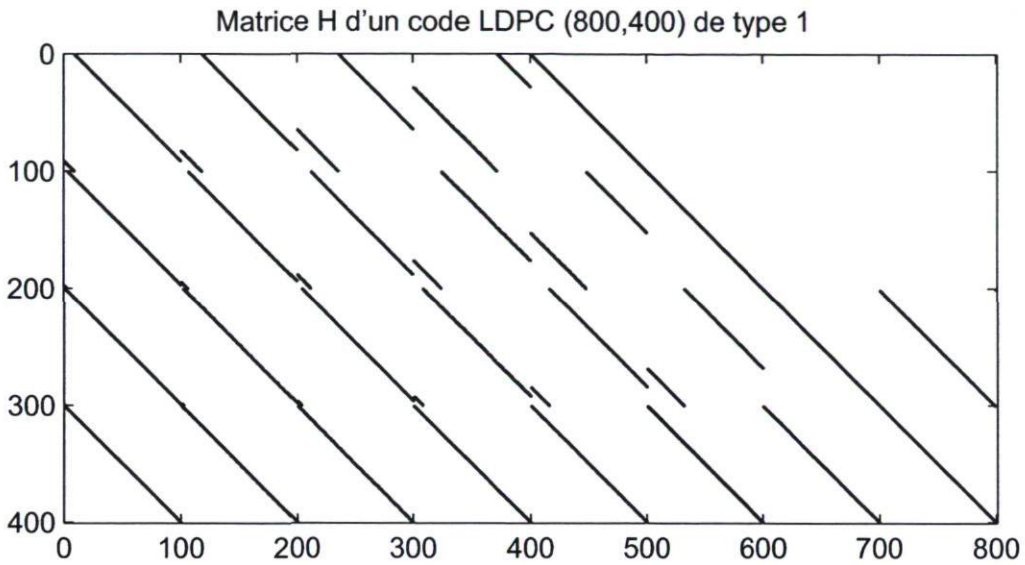


FIG. 4.1 – Matrice de parité d'un code QC-LDPC de type 1.

composée des deux parties : $\mathbf{H} = [\mathbf{H}_s|\mathbf{H}_p]$.

Pour construire la matrice $\mathbf{H}_s$, ils commencent aussi par le choix des positions des sous-matrices avec coefficients de permutation positifs. Un coefficient positif δ_i indique une permutation circulaire vers la gauche avec δ_i positions des lignes de la sous-matrice, un coefficient négatif indique une sous-matrice nulle et les coefficients nuls indiquent la matrice identité. Afin d'éviter les cycles courts, il faut s'assurer que dans la matrice formée avec les coefficients de permutation, il n'y ait pas de cycles courts entre les coefficients positifs. Les auteurs ont démontré que le périmètre du graphe de Tanner de la matrice des coefficients représente une limite inférieure pour le périmètre de la matrice de parité de notre code QC-LDPC. On peut ainsi s'assurer de la structure des cycles de la matrice $\mathbf{H}_s$ en imposant une certaine structure de la matrice des coefficients

qui est plus simple à étudier. On observe que cette méthode est similaire à la méthode de [29] qui cherche à choisir les positions des sous-matrices non-nulles afin d'éliminer les cycles courts dans la grande matrice de parité. Ils ont démontré aussi que, pour la matrice de coefficients, il est impossible de créer des graphes de Tanner sans cycles de longueur 4 si

$$d_v^{max} > m/2 \tag{4.4}$$

, où d_v^{max} est le poids maximal des colonnes de la matrice $\mathbf{H}$ et m est le nombre de sous-matrices blocs sur la verticale.

La deuxième étape pour construire le code consiste à choisir les coefficients de permutation de manière pseudo-aléatoire et graduellement en évitant les cycles courts dans le graphe entier de la matrice $\mathbf{H}$ (et pas sur des parties de la matrice $\mathbf{H}$ ou juste en ne considérant que quelques colonnes de la matrice comme dans la méthode de [29]. La méthode peut être vue comme un algorithme PEG (Progressive Edge Algorithm) qui ajoute à chaque étape un autre noeud dans le graphe de Tanner qui vérifie les conditions imposées (dans ce cas, la condition étant mise sur les valeurs des coefficients de permutation dans chaque noeud du graphe de Tanner et non pas sur les positions des noeuds comme dans la méthode PEG originale). Le but de cette construction est de réussir à atteindre un grand périmètre.

La formule pour les coefficients de permutation trouvée par des méthodes géométriques par Doré [31] est similaire à la formule existante dans [29]. Dans [31], les auteurs ont donné une limite pour la distance minimale d'un code de ce type :

$$d_{min} \leq 2 + mq_{min} \tag{4.5}$$

où q_{min} est le poids minimal de colonne dans $\mathbf{H}_s$ et m le nombre de sous-matrices sur la verticale.

La matrice $\mathbf{H}_p$ n'est pas triangulaire et doublement-diagonale comme dans le cas décrit par [29]. Elle est seulement doublement-diagonale avec un élément ajouté sur la première ligne et la dernière colonne de la matrice pour éviter d'avoir le poids d'une colonne égal à 1 (ce qui peut augmenter le plancher d'erreur).

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{I} & & & & \mathbf{I}(q-1) \\ \mathbf{I} & \mathbf{I} & & & \\ & \mathbf{I} & & & \\ & & \ddots & \mathbf{I} & \\ \mathbf{0} & & & \mathbf{I} & \mathbf{I} \end{bmatrix} \tag{4.6}$$

Nous avons pris comme exemple numérique la construction de deux matrices de parité d'un code QC-LDPC de même dimension (800, 400) et de même rendement ($R = 0.5$) par les deux méthodes présentées. La seule différence entre les deux matrice est dans la deuxième partie de la matrice de parité $\mathbf{H}$, c'est-à-dire dans la manière de construction de $\mathbf{H}_p$. On peut voir la structure de ces deux matrices de parités dans les figures 4.1 et 4.2. Comme futurs travaux, il serait intéressant de faire une comparaison de performance entre les deux types de codes QC-LDPC présentés.

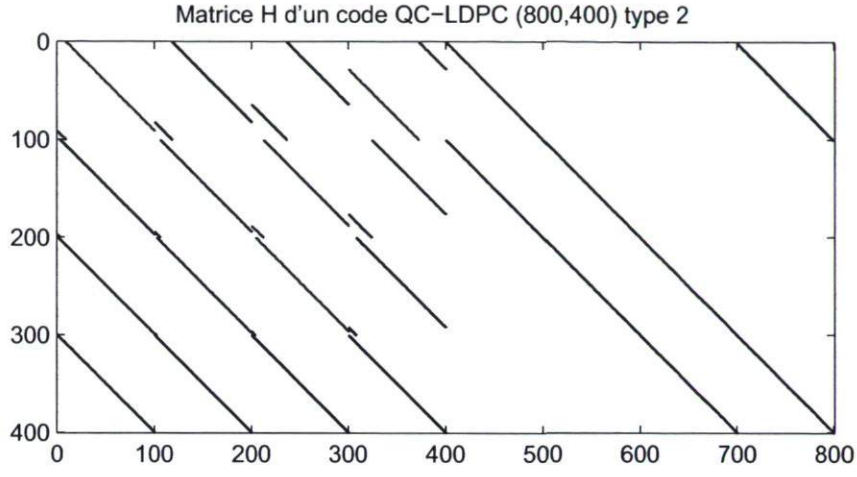


FIG. 4.2 – Matrice de parité d'un code QC-LDPC de type 2.

4.3 Codes LDPC-quasi-cycliques basés sur les codes protographes

Thorpe [32] a introduit le concept de *codes protographes*, une classe de codes LDPC construits à partir d'un protographe (une matrice incidente) de la manière suivante : les valeurs de 1 dans la matrice du protographe sont remplacées par des matrices de permutation de dimension $p \times p$ et les valeurs de 0 par des matrices nulles $p \times p$. Si ces sous-matrices de permutation sont circulaires, on parle alors de codes QC-LDPC. Il est important de trouver des codes avec de grands périmètres. On sait qu'un code QC-LDPC conventionnel ne peut pas avoir un périmètre plus grande que 12. À part le périmètre, il y a aussi des cycles plus grands que lui dans le graphe, nommés *cycles inévitables*. Un cycle inévitable est formé par la combinaison de deux ou trois cycles simples (un cycle simple est un cycle qui ne contient pas d'autres cycles de longueur

plus petite). Si r arrêtes sont partagées par deux cycles simples de longueur $2k$ et $2l$ dans le protographe, alors il y a un cycle inévitable de longueur $2(2l+2k-r)$ dans son code protographe.

Dans [33], les auteurs essayent de trouver la relation entre le périmètre d'un protographe et la longueur des cycles inévitables du code protographe correspondant. Ils identifient toutes les structures des sous-graphes des protographes qui génèrent les cycles inévitables $2i$, (avec $i = 6, 7, 8, 9, 10$) indépendamment des valeurs des coefficients de permutation.

Ils ont présentées des méthodes de construction des protographes dont les codes protographes correspondants ont un périmètre de 14 ou 18. Ils proposent aussi des règles d'assignation des coefficients de permutation d'un code QC-LDPC qui garantissent un périmètre 14, même s'il n'y a pas encore de méthodes très pratiques.

La longueur d'un cycle dans un code protographe peut être calculée en fonction de la longueur des cycles dans le protographe correspondant et les valeurs des coefficients de permutations circulaires [34].

Tanner [35, 36] a proposé une méthode algébrique d'assignation des coefficients de permutation pour les codes QC-LDPC pour avoir exactement le périmètre 12.

Si un protographe a le périmètre $2i$, $i \geq 2$, la longueur d'un cycle inévitable de son code protographe est plus grand ou égal à $6i$. Ceci implique que le code protographe peut avoir le périmètre plus grand ou égal à $6i$ avec un choix approprié des valeurs des coefficients de permutation. Cela est le même cas que notre type de codes LDPC étudiés auparavant qui avaient des cycles de longueur 4 dans la matrice de base, donc

le périmètre de la matrice de parité du code étant égal ou plus grand que 12. Mais si on essaye de choisir des bons coefficients de permutation, on peut augmenter la longueur du périmètre du code (qui amène une amélioration des performances du code).

Si un protographe a un cycle de longueur $i \geq 6$, son code protographe ne peut pas avoir un périmètre plus grand que $2i$.

4.4 Autres méthodes pour construire des codes LDPC-quasi-cycliques

La méthode la plus générale de construire les codes QC-LDPC est à l'aide de sous-matrices obtenues par des permutations circulaires de la matrice identité. On appelle cette méthode de type I. Dans [37], une autre méthode est présentée, qui utilise aussi des sous-matrices obtenues par la somme (modulo 2) des deux sous-matrices identité permutées. Cette méthode (de type II) augmente la distance minimale du code et garde la même régularité du code. Par exemple, un code QC-LDPC régulier (3,4) construit par la méthode de type I a la limite inférieure de la distance minimale égale à 24. Mais pour le type II, la limite inférieure de la distance minimale est 32. À une matrice formée par des sous-matrices circulaires on peut associer une matrice de parité polynomiale $\mathbf{H}(\mathbf{X})$. Un code QC-LDPC est de type I s'il est donné par la matrice de parité $\mathbf{H}(\mathbf{X})$ avec toutes les entrées soit des monômes, soit des matrices zéros. Un code QC-LDPC est de type II s'il est donné par une matrice de parité $\mathbf{H}(\mathbf{X})$ dont toutes les entrées sont soit des binômes, soit des monômes, soit des matrices zéros. Un exemple de matrice de type

I est donné ci-dessous :

$$\mathbf{H}(\mathbf{X}) = \begin{bmatrix} X & X^2 & X^4 & X^8 \\ X^5 & X^{10} & X^{20} & X^9 \\ X^{25} & X^{19} & X^7 & X^{14} \end{bmatrix} \tag{4.7}$$

La matrice $\mathbf{H}(\mathbf{X})$ qui génère des codes de type II pourrait avoir la forme suivante :

$$\mathbf{H}(\mathbf{X}) = \begin{bmatrix} X + X^2 & 0 & X^4 & X^8 \\ X^5 & X^9 & X^{10} + X^{20} & 0 \\ 0 & X^{25} + X^{19} & 0 & X^7 + X^{14} \end{bmatrix} \tag{4.8}$$

Cette matrice a été formée à partir de la matrice initiale (de type I) en faisant la somme de certaines sous-matrices en prenant soin de garder la même régularité $(3, 4)$ de la matrice de parité.

Il y a un lien entre le périmètre du graphe de Tanner d'un code QC-LDPC de type I et les mineurs de la matrice de parité polynomiale $\mathbf{H}(\mathbf{X})$: le graphe du code n'a pas de cycles de longueur $2i$ $(i = 2, 3)$ si et seulement si tous les mineurs de ordre i de $\mathbf{H}(\mathbf{X})$ n'ont pas de perte de poids (aucune annulation des monômes dans la somme de déterminant).

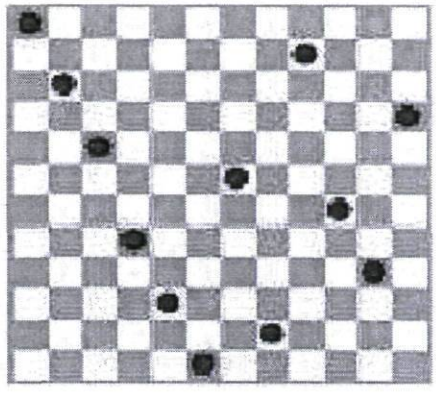
La distance minimale d'un code quasi-cyclique peut être augmentée aussi en remplaçant la matrice identité par une matrice construite selon l'algorithme Queen [38,39]. Elle a les propriétés suivantes : elle a un poids $w_r = 1$ sur chaque ligne, un poids $w_c = 1$ sur chaque colonne et le poids $w_t = 1$ sur chaque diagonale. À la Fig. 4.3, on a donné deux exemples de telles matrices de dimensions différentes ($n = 12$ et $n = 26$). Pour des petites valeurs de la dimension de la sous-matrice, il est facile de trouver les solutions.

Mais dès que la dimension dépasse la valeur 8, il faut recourir à des recherches exhaustives faites par l'ordinateur pour trouver la ou les solutions, parce que les solutions ne sont pas uniques et cela peut prendre parfois un grand temps de calcul.

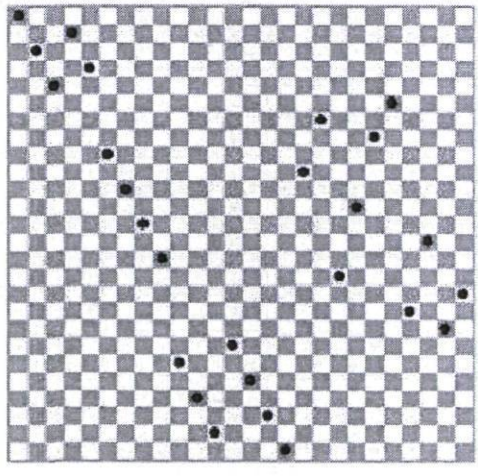
Dans les méthodes de construction des codes QC-LDPC proposés par Z. He [29] et Doré [31], on jongle beaucoup avec les poids des colonnes, les coefficients de permutations pour les sous-matrices, dans le but d'éviter les cycles courts et les structures désavantageuses pour le code, mais ils n'essayent pas d'augmenter la distance minimale du code en travaillant à l'intérieur des sous-matrices. C'est pour cette raison qu'on veut proposer comme travail futur l'étude de la performance du code QC-LDPC triangulaire et doublement-diagonale proposé par Zhi Yong He, en remplacent la matrice identité par la matrice Queen. On prévoit obtenir de meilleurs résultats de cette manière, en augmentant la distance minimale du code.

4.5 Conclusion

Dans ce chapitre, nous avons présenté différentes méthodes de construction pour les codes LDPC quasi-cycliques QC-LDPC. Ils donnent d'excellentes performances et ne requièrent qu'une faible complexité de calcul au codage. On présente les codes QC-LDPC triangulaires et doublement-diagonales, les codes QC-LDPC basés sur les protographes, les codes QC-LDCP ayant une matrice de parité polynomiale et les codes QC-LDPC basés sur la sous-matrice Queen. Nous avons proposé aussi deux avenues possibles de travaux futurs qui peuvent améliorer les performances des codes existantes.



(a) Matrice de parité avec $n = 12$



(b) Matrice de parité avec $n = 26$

FIG. 4.3 – Illustration de matrices de parité construites à l'aide de l'algorithme Queen.

Chapitre 5

Performances des codes LDPC - problèmes actuels

5.1 Introduction

Le plancher d'erreur et la distance minimale des codes demeurent des problèmes ouverts (même s'il y a déjà plus de 50 ans de recherche centralisée sur la distance minimale des codes). Pour détecter par simulations le plancher d'erreur pour les courbes de probabilité d'erreur, il faut transmettre beaucoup de mots-codes, et ce processus prend beaucoup de temps, surtout pour les codes qui ont un plancher d'erreur très bas. Une autre méthode d'analyse de la performance des codes LDPC consiste à analyser certaines structures spécifiques qui existent dans le graphe de Tanner du code ou déterminées par la matrice de parité du code, qui ont une grande influence sur le comportement des codes sur certains canaux.

Ensuite, on peut essayer d'éviter ces structures dans la construction du code ou

d'estimer le plancher d'erreur ou la courbe de performance en fonction de la distribution de ces ensembles pour chaque code ou type de codes LDPC.

Les ensembles de piégeage (en anglais : trapping sets) et les ensembles d'arrêt (en anglais : stopping sets) sont des structures existant dans le graphe de Tanner, qui influencent le plancher d'erreur des codes LDPC.

Dans les communications qui demandent un très bon rendement d'erreur pour de bons rapports signal à bruit (i.e. les communications optiques, les communications par satellite, Ethernet ou les DVD) il est important d'analyser ces structures qui influencent le comportement du code.

5.2 Périmètre du graphe de Tanner

Même si les contraintes de la construction des trois types de codes imposent qu'ils n'ont pas de cycles de longueur 4, nous avons construit un programme Matlab pour détecter de manière exhaustive tous les cycles de différentes longueurs dans les graphes des codes. Le programme permet de vérifier si les codes sont bien construits (sans cycles de longueur 4) et aussi l'analyse des autres propriétés du graphe concernant les cycles plus grands. On peut vérifier aussi si les codes ainsi construits ont des cycles de longueur 6, 8, 10, etc. Le périmètre du graphe de Tanner, i.e. la longueur du plus petit cycle dans le graphe, doit être aussi grand que possible pour éliminer les cycles de courtes longueurs (en particulier de longueur 4) et obtenir de bonnes performances asymptotiques du code. Cependant, il ne doit pas être très grand parce que dans ce cas, il en résulte une petite distance minimale du code, ce qui augmente le plancher d'erreur

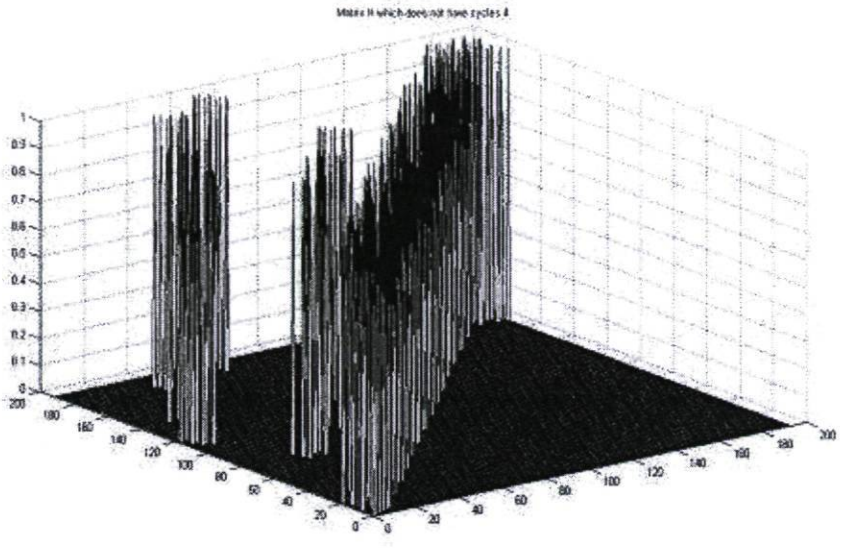
irréductible. On doit donc faire un compromis dans le processus d'élimination de cycles courts.

Concernant l'étude de la structure des cycles dans un graphe de Tanner pour les codes QC-LDPC, pour les cycles de longueur 4, on peut utiliser un critère plus simple : si la matrice $\mathbf{Q} = \mathbf{H} * \mathbf{H}'$ a des valeurs plus petites que 1, la matrice $\mathbf{H}$ n'a pas de cycles de longueur 4 (ici $\mathbf{H}'$ est la transposée de la matrice $\mathbf{H}$). Si la matrice produit $\mathbf{Q}$ a des valeurs plus grandes que 1, cela veut dire que la matrice $\mathbf{H}$ a un périmètre de 4. À la figure 5.1, on montre deux matrices de parité (Fig. 5.1(a) et 5.1(c)) ainsi que les matrices $\mathbf{Q}$ correspondantes (Fig. 5.1(b) et 5.1(d)). Ainsi la matrice $\mathbf{Q}$ de la figure 5.1(b) indique que la matrice de parité $\mathbf{H}$ de la figure 5.1(a) conduit à un code LDPC sans cycle de longueur 4, alors que la matrice de parité à la figure 5.1(c) indique un code LDPC avec des cycles de 4 (au périmètre de 4).

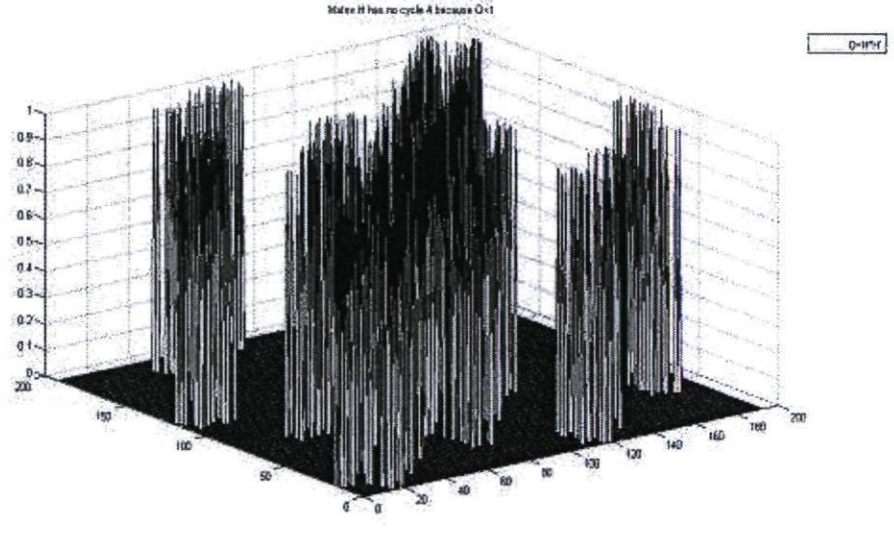
5.3 Ensembles d'arrêt et ensembles de piégeage

Les *ensembles d'arrêt* et *de piégeage* représentent des sous-graphes dans le graphe de Tanner d'un code LDPC qui, pour certaines classes de canaux, ont une grande influence sur le plancher d'erreur.

Pour le décodage ferme, borné par la distance minimale du code, la performance est complètement déterminée par l'énumérateur de poids et le paramètre de transition (la probabilité de croisement) du canal symétrique binaire (BSC). D'une manière similaire, la performance et la complexité de certains algorithmes de décodage souple sur un canal de bruit additif gaussien blanc (AWGN) peut être caractérisée par leurs *mots-codes*

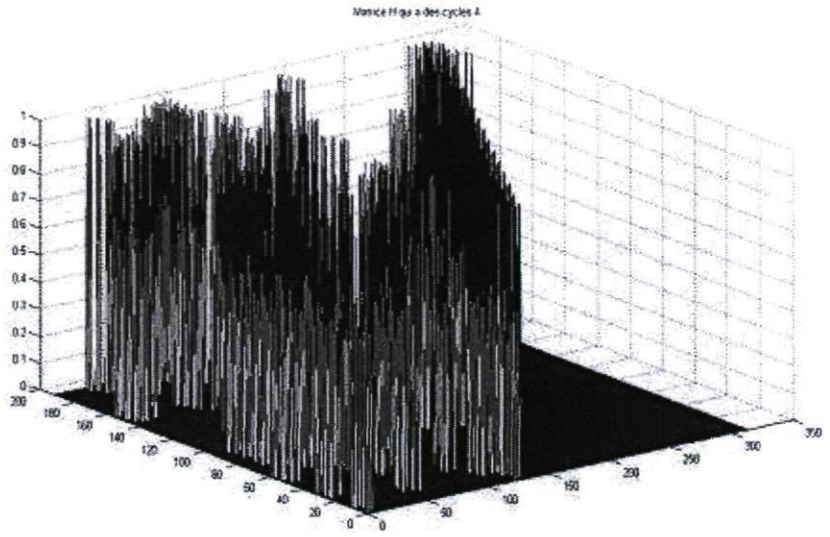


(a) Matrice de parité sans cycles de longueur 4

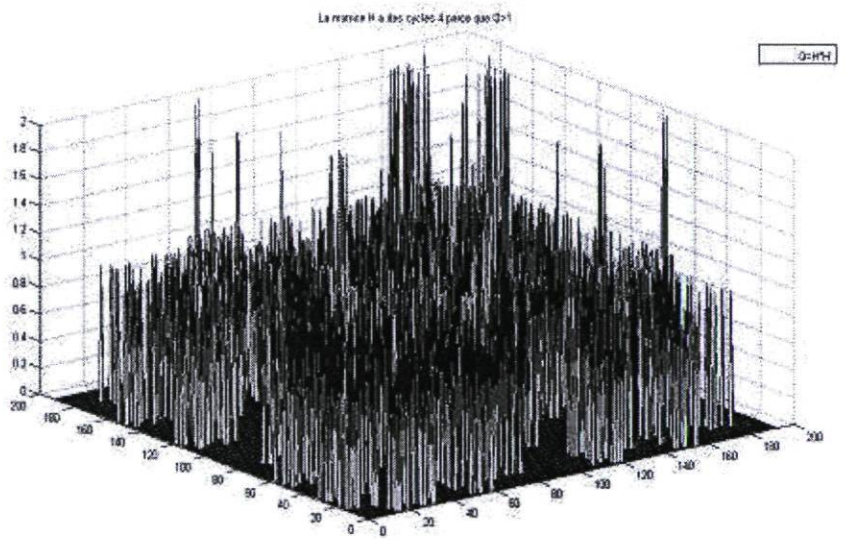


(b) $\mathbf{Q} = \mathbf{H} * \mathbf{H}'$

FIG. 5.1 – Méthode de visualisation pour déterminer s'il existe des cycles de longueur 4 dans la matrice de parité $\mathbf{H}$ (début).



(c) Matrice de parité avec cycles de longueur 4



(d) $\mathbf{Q} = \mathbf{H} * \mathbf{H}'$

FIG. 5.1 – Méthode de visualisation pour déterminer s'il existe des cycles de longueur 4 dans la matrice de parité $\mathbf{H}$ (suite).

minimaux et les mots-codes minimaux de leurs codes duals. Pour un canal binaire avec effacement (BEC), l'algorithme de décodage itératif échoue si les erreurs sont confinées dans les sous-ensembles de noeuds variables qui forment des *ensembles d'arrêt* [22]. Pour un canal AWGN, on peut observer un phénomène similaire donné par les ensembles de piégeage [40].

Un ensemble d'arrêt est un ensemble de noeuds variables dans le graphe, pour lesquels le sous-graphe correspondant ne contient pas de noeuds de contrôle de degré 1. Ils ont été introduits en 2002 par C. Di [22].

En 2003 Richardson a introduit les ensembles de piégeage [40]. *Un ensemble de piégeage général* (a, b) est un ensemble de a noeuds de variables pour lesquels le sous-graphe induit contient $b > 0$ noeuds de contrôle de degré impair.

Un ensemble de piégeage élémentaire (a, b) est un ensemble de piégeage pour lequel tous les noeuds de contrôle dans le sous-graphe induit ont soit un degré 1 soit un degré 2 et il y a exactement b noeuds de degré 1. Les petits ensembles de piégeage ($a \leq \sqrt{n}$ et $b \leq 4a$) ont une plus grande influence sur le plancher d'erreur et sont appelées ensembles de piégeage dominants.

Explication du phénomène : Dans les premières itérations du décodage de propagation des convictions (BP), dans certains noeuds de variables d'un ensemble de piégeage, les estimés de fiabilité pour les bits non-corrects augmentent. Cette information est propagée ensuite dans tous les noeuds de variables de l'ensemble de piégeage. Parmi eux, il y a déjà des noeuds avec une très faible fiabilité du canal, et

donc leur situation empire. Après cette étape, les noeuds de variables externes au sous-graphe commencent à ajuster les estimés non-corrects vers les estimés corrects. Mais, à ce moment là, les noeuds de variables dans l'ensemble de piégeage ont déjà bien établi leurs valeurs vers les décisions non-correctes et cette mauvaise information reste bloquée à l'intérieur de cet ensemble de noeuds jusqu'à la fin du processus de décodage. Les ensembles de piégeage sont appelés aussi *problèmes non-mot-code.*

Dans [41], les auteurs proposent une méthode de recherche des ensembles de piégeage dominants pour les codes de longueur moyenne. La méthode est appelée "*La recherche du décodeur*" et ne garantit pas une liste complète de tous les ensembles de piégeage possibles. Ils proposent aussi une méthode de construction de codes LDPC réguliers avec un plancher d'erreur très bas en utilisant cette méthode de recherche du décodeur.

Il est très difficile, voire impossible, de trouver toutes les structures combinatoires d'un code. Mais il existe des méthodes pour trouver les ensembles de piégeage d'un sous-graphe et ensuite de généraliser pour tout le graphe. Dans [42], Cole et al. proposent une nouvelle méthode de détermination complète de la performance d'erreur du code à partir d'ensembles de piégeage dominants trouvés. On peut ainsi faire aussi une estimation du plancher d'erreur du code entier (par une procédure appelée "*l'importance du prélèvement*"). La méthode est appelée "*La recherche localisée du décodeur*" et ne demande pas une connaissance totale de tous les ensembles de piégeage pour une bonne estimation du plancher d'erreur.

Dans [42] les auteurs spécifient l'importance de donner une définition des ensembles de piégeage qui prend en considération toute l'histoire du processus de décodage

et pas seulement l'état final. Il faut garder en mémoire toutes les décisions fermes après chaque itération et si à la fin des itérations aucun mot-code valide n'est trouvé, alors on définit l'ensemble de piégeage comme étant le mot-code qui correspond au syndrome de poids minimal.

Les ensembles de piégeage dépendent de la structure du code (la structure du graphe de Tanner) et de l'algorithme de décodage utilisé. Par exemple, une telle structure peut être un ensemble de piégeage pour un algorithme de décodage itératif, mais pas pour un autre (par exemple, les variantes de l'algorithme MPA (*message passing algorithm*)). Il y a deux méthodes pour améliorer l'influence des ensembles de piégeage sur le plancher d'erreur :

- identification des algorithmes de décodage qui peuvent éviter que l'information erronée ne soit bloquée dans le graphe du code ;
- identification par méthode analytique des classes des codes LDPC qui ne contiennent pas de petits ensembles de piégeage.

Dans [43] on trouve une classification des ensembles de piégeage en trois types :

1. *ponctuels* - ils sont responsables de toutes les décisions erronées à la fin du processus de décodage ;
2. *apériodiques* - ils contiennent un ensemble de piégeage à partir duquel les messages erronés sont propagés à plusieurs variables externes ;
3. *périodiques* - ils donnent une forme périodique au processus de décision.

L'algorithme de décodage peut exploiter la structure des ensembles de piégeage,

en observant à chaque itération le mot-code décodé. Si le décodeur produit des oscillations avec une période fixe, une solution qui pourrait amener à la convergence consiste à réinitialiser le décodeur utilisant un des mot-codes décodés comme le mot-code initialement reçu [44]. Les erreurs ponctuelles peuvent être décodées par renversement de certains bits connectés aux noeuds de parité non-satisfaits. Cependant, pour des ensembles de piégeage très persistants, la convergence peut ne pas se produire : dans ce cas on garde en mémoire les ensembles de piégeage et leurs syndromes correspondants [44]. Ce décodeur amélioré est utilisé uniquement quand le décodeur standard échoue (événement qui arrive très rarement). Ainsi, pour la région du plancher d'erreur où le taux d'erreur du trame est de l'ordre 10^{-6}, le décodeur amélioré est nécessaire seulement pour un mot-code parmi un million de mots-codes.

Créer une liste avec tous les ensembles de piégeage possibles et leurs sous-graphes correspondants serait utile pour analyser les graphes de Tanner de tous les codes. Pour un ensemble de piégeage, chaque noeud de variable du sous-graphe est connecté à plus de noeuds de parité de degré pair qu'à des noeuds de parité de degré impair. Le problème de créer une liste des ensembles de piégeage est apparenté au problème de la coloration dans la théorie de graphes. Le problème de la coloration des graphes est très bien étudié dans la littérature et on pourra utiliser les techniques de ce domaine pour résoudre les problèmes dans la théorie du codage. Ainsi les noeuds variables vont correspondre aux lignes et les noeuds de parité aux points (les noeuds de parité de degré pair seront représentés par des points blancs, sinon par de points noirs). Donc un sous-graphe correspondra alors à un possible ensemble de piégeage si chaque ligne

du sous-graphe a plus de points blancs que de points noirs. Si le nombre de points blancs sur chaque ligne est restreint au nombre minimal (seulement un point de plus que les points noirs), le sous-graphe correspondra à une structure *d'ensemble de piégeage minimale*. Dans [44], on trouve plusieurs de ces structures d'ensembles de piégeage et leurs sous-graphes correspondants, trouvés par des techniques simples de coloration de graphes. Pour trouver une liste exhaustive d'ensembles de piégeage, il faut utiliser des techniques plus sophistiquées.

Il y a aussi un lien entre la structure des cycles du graphe et les ensembles de piégeage. Dans [44], les auteurs ont présenté quelques structures des ensembles de piégeage qui correspondent aux différentes valeurs de poids de colonne γ et pour des périmètres (la longueur du cycle le plus court du graphe) de 6 ou 8.

Une classe particulière de codes est constituée par les codes basés sur la géométrie projective, pour lesquels on croit pouvoir obtenir des planchers d'erreur très bas (en dessous de 10^{-23}). Ceci peut être expliqué par la non-existence de certains ensembles de piégeage petits dans le graphe du code. Dans [43], Laendner et al. démontrent que les codes PG n'ont pas de petits ensembles de piégeage dans leurs graphes d'incidence ligne-point.

Le problème du plancher d'erreur est un problème différent pour chaque code. Le comportement du plancher d'erreur des codes courts est plus imprévisible que dans le cas des codes longs. Il faut donc examiner chaque code en particulier pour être sûr de

sa performance d'erreur aux grands rapports signal à bruit.

L'augmentation du périmètre du graphe de Tanner détermine un meilleur spectre des ensembles de piégeage. L'augmentation du degré de noeuds de variables (le poids des colonnes) va diminuer le nombre des ensembles de piégeage de petits poids, mais en même temps affecte négativement le seuil de décodage (la chute de la courbe).

Dans [45], les auteurs font une étude de 2 codes irréguliers construits par une combinaison de méthodes (codes irréguliers, graphes de Tanner "progressive edge-growth" et structure escalier). Les codes sont construits à partir du graphe de Tanner : à chaque étape on ajoute des noeuds en imposant quelques conditions (sur la distribution des degrés, le périmètre, etc).

Ils proposent une méthode pour trouver les ensembles de piégeage dominants par la transmission des impulsions de bruit (une méthode très efficace pour des codes de longueur moyenne et courte). Il faut compter combien de fois on rencontre un ensemble de piégeage, pour déterminer les ensembles de piégeage dominants (ils sont normalement directement proportionnels). La structure en "escalier" de la matrice de parité facilite l'implémentation, mais elle détermine l'apparition des ensembles de piégeage de type $(x, 1)$ (groupe de noeuds de variables avec 1 noeud de contrôle non-satisfait). Le plancher d'erreur est causé par des mots-codes de poids petit ($w_H < 20$) et par les non-mots-codes (les ensembles de piégeage) de type $(x, 1)$ et $(x, 2)$. Si on ajoute des valeurs de 1 sous la diagonale de la structure en escalier, on peut réduire les ensembles de piégeage de type $(x, 1)$ et $(x, 2)$, mais le degré des noeuds variables va augmenter et donc il va affecter le seuil de décodage (la chute de la courbe). Il y a 3 types d'évènement erreur dans ces codes :

1. erreurs de type 1 - les mots-codes de petit poids. Ils sont déterminés par la structure irrégulière de la matrice de parité (la première partie de **H**) et déterminent le comportement du code pour des rapports signal à bruit petits.

2. erreurs de type 2 - les ensembles de piégeage $(x, 1)$ et $(x, 2)$ avec x petit, où tous les bits des ensembles de piégeage sont parmi les bits de l'escalier de la matrice **H**. Ils déterminent le comportement du code pour de grands rapports signal à bruit (ils sont la cause du plancher d'erreur).

3. erreurs de type 3 - similaires aux erreurs de type 2, mais les bits des ensembles de piégeage sont une combinaison des bits consécutifs de l'escalier et un petit nombre (1 ou 2) de noeuds variables de degré 4 ou 5. Ils déterminent la performance du code dans la région de chute et ils ont aussi une influence pour des rapports signal à bruit plus grands.

Pour des grands rapports signal à bruit, les ensembles de piégeage de type 2 deviennent dominants. Étant donné que la structure en escalier qui détermine ce type d'erreurs est pareille pour les deux codes, les performances d'erreur vont se confondre au delà d'une certaine valeur du rapport signal à bruit. Les courbes des performances de ces deux codes analysés sont tracés par des simulations Monte-Carlo jusqu'à $Eb/No = 2$ dB [45] et pour des rapports signal à bruit plus grands, les auteurs ont utilisé leur méthode d'estimation des performances. Cette façon de faire prend beaucoup moins de temps en comparaison avec les simulations usuelles. La méthode proposée détermine les ensembles de piégeage dominants, estime ensuite l'influence de chaque ensemble de piégeage sur le plancher d'erreur et à la fin on fait une "addition" des influences des

ensembles de piégeage pour déterminer la courbe de performance d'erreur.

Pour les réseaux sans fils, un taux d'erreur de 10^{-6} est suffisant : donc il n'est pas pertinent d'étudier les ensembles de piégeage et le plancher d'erreur aux rapports signal à bruit très grands. Cependant, il est important de les analyser pour les communications optiques, pour les communications par satellite, Ethernet ou pour les DVD, où un taux de 10^{-10} est demandé.

D. Mackay [46] souligne qu'un code correcteur d'erreurs typique qui est décodé par échange de messages peut faire deux types d'erreurs :

1. des erreurs non-détectées, qui apparaissent lorsque le décodeur s'arrête en trouvant un mot-code qui n'est pas le mot-code original. Typiquement, une erreur non-détectée implique un mot-code de petit poids (ce qui veut dire que seulement un petit nombre de bits sont erronés). Quelques codes (par exemple des codes longs avec une longueur plus grande que 200, des codes Gallager réguliers) n'ont pas de mot-codes de petit poids. Donc, les erreurs non-détectées n'apparaissent jamais dans les applications pratiques. D'autres codes (comme les codes turbo et quelques codes irréguliers) ont des mot-codes de petit poids et les chercheurs font l'erreur de considérer ces erreurs normales.

2. des erreurs détectées, qui apparaissent lorsque le décodeur ne peut pas trouver un mot-code valide. Typiquement, les erreurs détectées impliquent un grand nombre de bits erronés. C'est un type d'erreur qu'on prévoit pour un très bon code (avec une performance proche de la limite de Shannon). Selon le théorème de Shannon, à

un débit proche de la capacité, les mots-codes qui se confondent sont à très longue distance (Hamming) l'un de l'autre. Un bon code de Gallager fait seulement des erreurs de grands poids.

Afin de distinguer les erreurs non détectées et détectées, il est important d'utiliser un critère adéquat pour arrêter le décodage. Si on inspecte les mots-codes de petit poids, on peut améliorer notre code en les éliminant.

5.3.1 Résultats de notre étude sur les performances en fonction des ensembles de piégeage

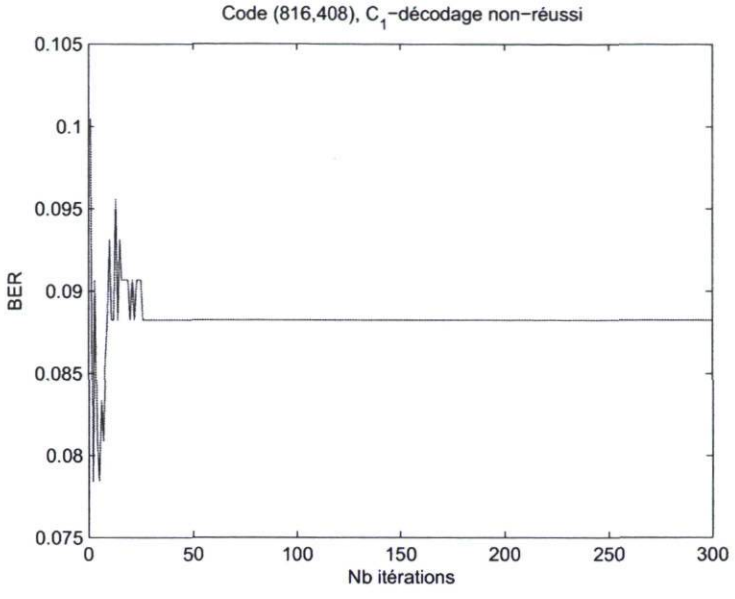
Nous avons commencé l'étude des ensembles de piégeage en essayant de trouver des mots-codes qui ne peuvent pas être décodés à la fin du nombre d'itérations préétabli du décodage SPA. En général, la cause de ce phénomène est les ensembles de piégeage qui déterminent une erreur dans leur sous-graphe correspondant : cette erreur se propage dans tout le graphe et le mot-code correspondant ne peut pas être décodé correctement même après un très grand nombre d'itérations.

Dans ce travail de recherche, on a choisi un code pseudo-aléatoire de dimensions $(816, 408)$, de rendement $R = 1/2$, de poids de colonne $\gamma = 4$ et de poids de rangée $\rho = 8$. Parmi les 100 mots-codes transmis aléatoirement, avec un rapport signal à bruit sur le canal AWGN de $Eb/No = 2$ dB, on a trouvé 21 mots-codes qui à la fin des itérations ne peuvent pas être bien décodés. On a considéré pour cette étude 300 itérations pour le décodage. On a retrouvé le classement de différents types de formes des ensembles

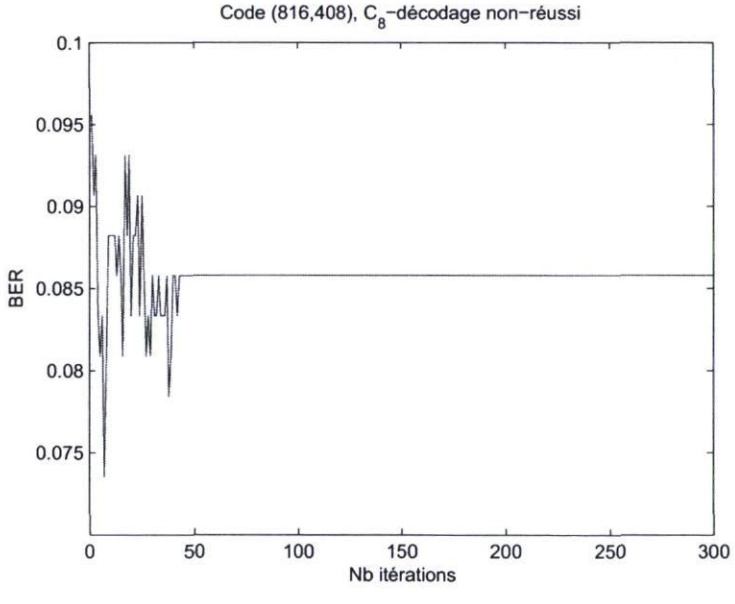
de piégeage fait dans [43] : les ensembles de piégeage ponctuels - avec un seuil d'erreur (Fig. 5.1), les ensembles de piégeage apériodiques (Fig. 5.2) et les ensembles de piégeage périodiques (Fig. 5.3). Chaque figure représente le décodage d'un mot-code différent qui n'a pas pu être décodé. Dans la figure 5.1 on peut remarquer qu'après environ 50 itérations, les oscillations du décodage convergent vers un seuil fixe qui reste le même jusqu'à la fin d'itérations. En fait, il est causé par l'existence d'une structure d'ensemble de piégeage ponctuelle dans le graphe de Tanner, qui propage un niveau d'erreur fixe dans le graphe et détermine l'erreur du décodage que l'on trouve à la fin des itérations. À la figure 5.2, on voit que l'erreur causée par l'existence d'un autre type d'ensemble de piégeage (i.e., apériodique), détermine l'oscillation du décodeur autour d'une solution erronée. Les messages échangés entre les noeuds du graphe de Tanner contiennent une erreur, qui dès qu'on avance dans le graphe pendant le décodage, peut déterminer des erreurs variées. La figure 5.3 montre le résultat du troisième type d'ensemble de piégeage existant dans le graphe de Tanner, qui détermine une oscillation périodique du décodeur autour d'une solution erronée.

5.3.2 Effets du décodage SPA moyenné sur les ensembles de piégeage et la performance des codes LDPC

Parmi les algorithmes de décodage connus à ce jour, il y a le *décodage mesuré* basé sur l'algorithme BP et le *décodage moyenné* basé aussi sur l'algorithme BP [43]. Les deux décodeurs évitent beaucoup les effets dus aux ensembles de piégeage et ont comme résultat une réduction du nombre de trames mal décodées dans la région du plancher

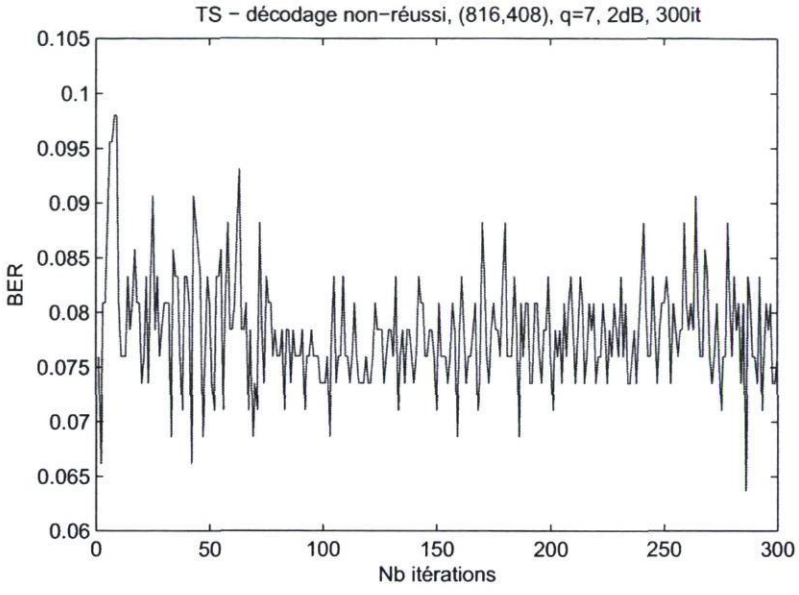


(e) Mot-code C_1, code $(816, 408)$, décodage non-réussi

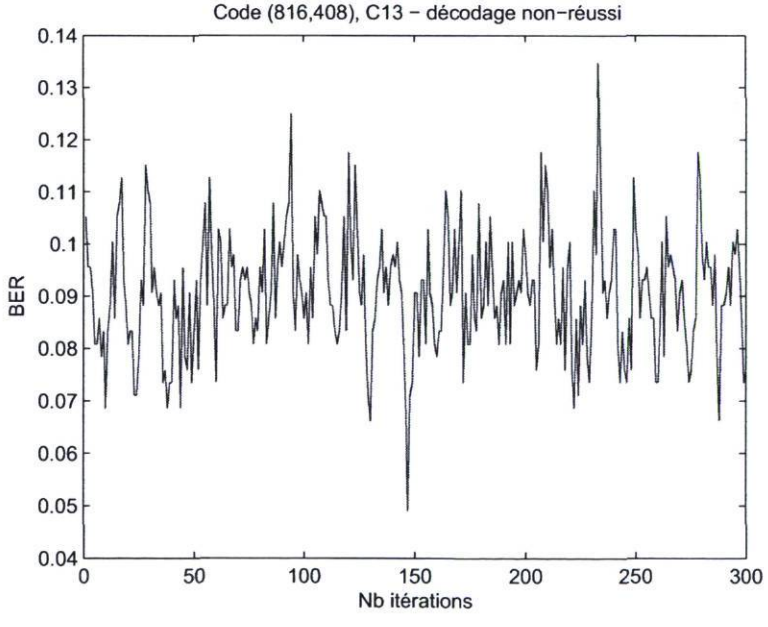


(f) Mot-code C_8, code $(816, 408)$, décodage non-réussi

FIG. 5.1 – Décodage SPA - Trames avec erreur de type ensemble de piégeage ponctuel (mots-codes qui ne peuvent pas être décodés) : a) C_1 ; b) C_8.



(a) Mot-code C_7, code $(816, 408)$, décodage non-réussi, 2dB



(b) Mot-code C_{13}, code $(816, 408)$, décodage non-réussi

FIG. 5.2 – Décodage SPA - Trames avec erreur de type ensemble de piégeage apériodique : a) C_7 ; b) C_{13}.

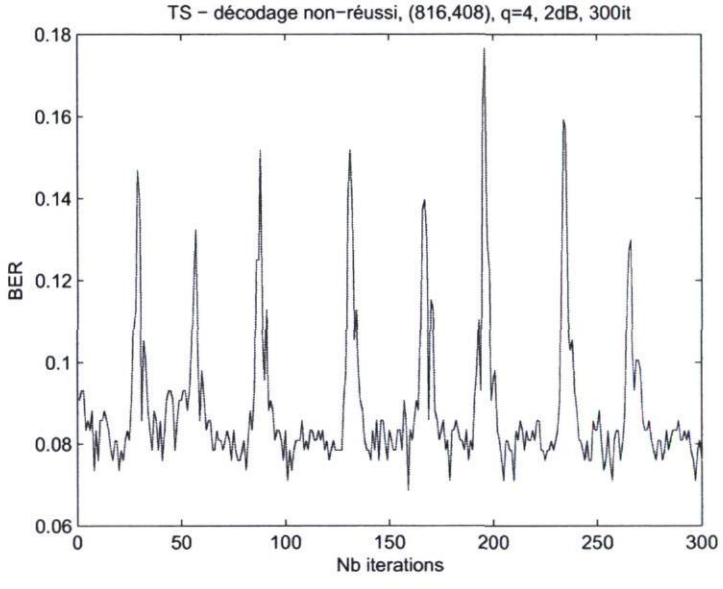


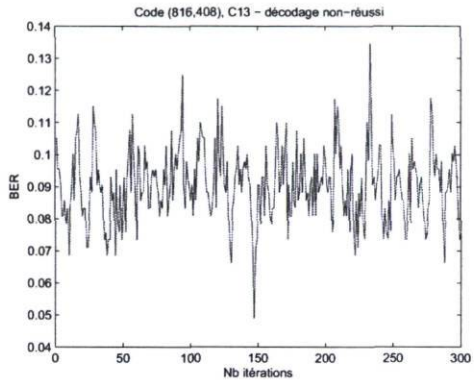
FIG. 5.3 – Décodage SPA - Trame avec erreur de type ensemble de piégeage périodique : code $(816, 408)$, 2 dB, 300 itérations)

d'erreur. Les deux méthodes ont d'autres inconvénients comme un temps de calcul prolongé (pour le décodage des mots-codes qui sont décodés sans problèmes jusqu'à la fin des itérations). Le décodage moyenné peut être amélioré (au niveau du temps de calcul) si on le transforme en un algorithme de *décodage moyenné adaptatif*, qui va initialiser le moyennage seulement si, entre deux itérations successives, il y a un très grand changement des probabilités extrinsèques, au dessus d'un seuil établi.

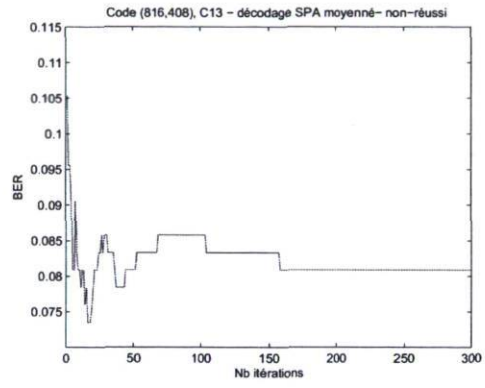
On a utilisé le décodage SPA moyenné aussi pour les codes étudiés dans la section précédente. Cet algorithme garde en mémoire, à chaque itération, les coefficients extrinsèques des q dernières itérations et fait un moyennage entre eux et le coefficient présent [43]. Les auteurs ont découvert que chaque ensemble de piégeage correspond à un saut de valeur d'un coefficient extrinsèque en comparaison avec tous les autres autour de lui : pour éliminer cette grande différence entre les coefficients, ils font un moyennage entre eux. Cette algorithme est conçu pour éliminer les oscillations de décodage dues aux ensembles de piégeage, en dirigeant la décision vers une solution moyenne (un plancher d'erreur stable qui correspond à un moyennage des amplitudes des oscillations). On peut observer les résultats en comparant les figures suivantes : Fig. 5.4(a) avec Fig. 5.4(b), Fig. 5.4(c) avec Fig. 5.4(d), Fig. 5.5(a) avec Fig. 5.5(b).

5.4 Pseudo mots-codes et pseudo-poids

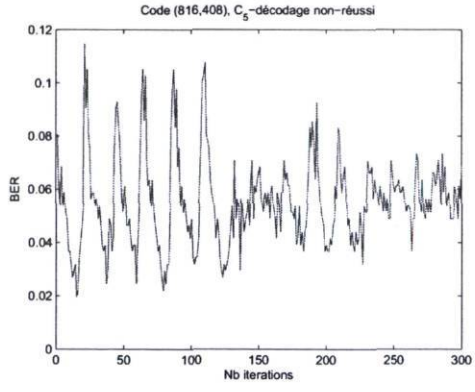
Pour comprendre la performance des codes avec le décodage à vraisemblance maximale ML (maximum-likelihood decoding) on étudie les mots-codes et en particulier les mots-codes minimaux et leurs poids de Hamming. Dans le contexte de LP (programma-



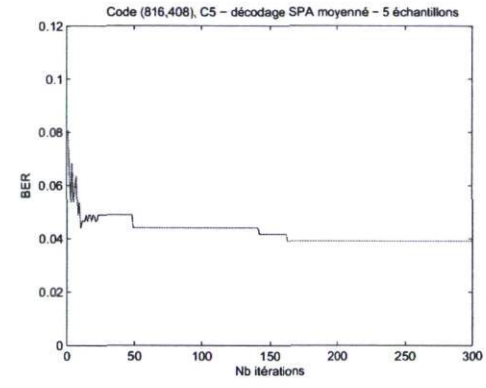
(a) décodage SPA normal non-réussi, code (816, 408), mot-code $C_1 3$



(b) décodage SPA moyenné non-réussi, code (816, 408), mot-code $C_1 3$

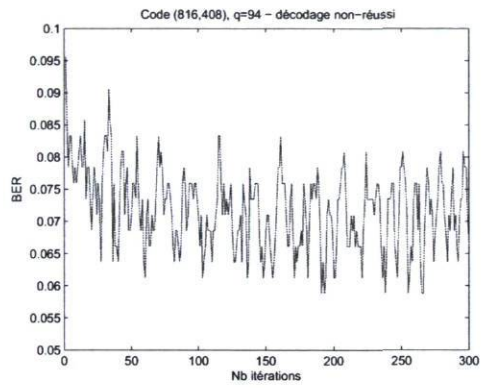


(c) décodage SPA normal non-réussi, code (816, 408), mot-code C_5

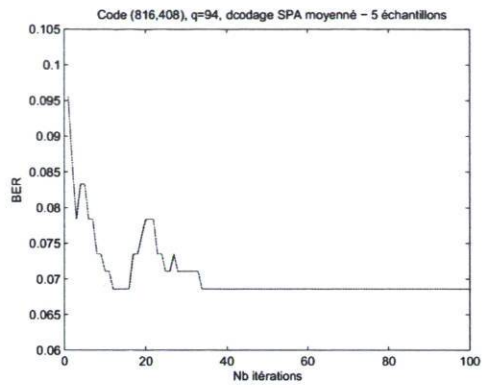


(d) décodage SPA moyenné non-réussi, code (816, 408), mot-code C_5

FIG. 5.4 – Les effets du décodage SPA moyenné ("averaged belief propagation") pour différents mots-codes qui ne peuvent pas être décodés.



(a) décodage SPA normal non-réussi, code (816, 408), mot-code C_8



(b) décodage SPA moyenné non-réussi, code (816, 408), mot-code C_8

FIG. 5.5 – Les effets du décodage SPA moyenné ("averaged belief propagation") pour différents mots-codes qui ne peuvent pas être décodés.

tion linéaire) on doit étudier les pseudo-mots-codes et en particulier les pseudo-mots-codes minimaux et leurs pseudo-poids. La programmation linéaire a des implications immédiates sur le décodage MPI (message-passing-iterative decoding). Un pseudo mot-code est un vecteur de valeurs intègres non-négatives $h = (h_1, h_2, ..., h_n)$ ayant la propriété suivante : pour chaque noeud de contrôle $j \in J$ il existe un voisinage de pseudo mots-codes $h_i : i \in N(j)$, qui est la somme des mots-codes locaux (vecteurs d'incidence des ensembles de même taille). $N(j)$ est l'ensemble de tous les noeuds voisins du noeud j. Le pseudo-poids est le poids d'un pseudo mot-code. Les pseudo mots-codes minimaux sont les pseudo mots-codes de poids minimal. L'étude des propriétés des pseudo-mots-codes est crucial pour l'analyse de la performance du décodage itérative des codes LDPC.

Dans [47], R. Smarandache et al. présentent une étude analytique du comportement des codes FG$(2, q)$ avec le décodage LP et le décodage itératif. Les auteurs ont étudié les pseudo-mots-codes minimaux des codes LDPC basés sur les géométries finies (EG-LDPC et PG-LDPC) et ont donné les limites inférieures de leurs pseudo-poids. Il est beaucoup plus facile de faire cette analyse pour ces types de codes en raison de leur structure bien définie. Le pseudo-poids pour un canal AWGN est défini par [47] :

$$w_p^{AWGN} = (\frac{|w_1| + |w_2| + ... + |w_n|}{\sqrt{w_1^2 + w_2^2 + ... + w_n^2}})^2 \tag{5.1}$$

Pour un code $PG(2, q)$ avec les pseudo-mots-codes contenant seulement des valeurs de 0 et 1, les pseudo-poids sont limités par :

- pour $q = 2$ et $q = 4$:

$$w_p \geq q + 4 \tag{5.2}$$

- pour $q \geq 8$:

$$w_p \geq q + 5 \tag{5.3}$$

Pour un code $PG(2, q)$ avec les pseudo-mots-codes non-mesurés contenant seulement des valeurs de 0, 1 et 2, les pseudo-poids sont limités par :

$$w_p \geq 1.185(q + 2) \tag{5.4}$$

Pour un code $PG(2, q)$ avec les pseudo-mots-codes non-mesurés contenant seulement des valeurs de 0, 1, 2 et 3, les pseudo-poids sont limités par :

$$w_p \geq 1.125(q + 2) \tag{5.5}$$

Les pseudo-mots-codes non-mesurés ω ont la propriété que : ω mod 2 est un

mot-code.

Pour des vecteurs binaires x : $w_p^{AWGN}(x) = w_p^{BSC}(x) = w_p^{BEC}(x) = w_p(x)$.

Pour les codes PG$(2, q)$ et EG$(2, q)$ avec $q = 2$, $q = 4$ et $q = 8$ les auteurs [47] ont trouvé tous les mots-codes, les mots-codes minimaux et les pseudo mots-codes minimaux et ont donné des polynômes énumérateurs des poids pour les trois types de canaux possibles : AWGN, BSC, BEC.

Ils ont introduit une nouvelle notion nommée *intervalle du spectre des pseudo-poids* [47] pour une matrice de parité, qui représente la différence entre le pseudo-poids minimum du code et le poids minimal des mots-codes (le poids de Hamming). Plus l'intervalle du spectre est grand, plus la performance du décodage LP (et potentiellement du décodage MPA) est proche de la performance du décodage ML quand le SNR tend vers l'infini. Pour les codes construits aléatoirement, l'intervalle du spectre pour un canal AWGN est toujours strictement négatif. Par contre, pour les codes basés sur les géométries finies PG$(2, q)$ et EG$(2, q)$, cet intervalle est non-négatif (en fait, pour les codes étudiés, les auteurs ont trouvé des valeurs positives très significatives), ce qui prouve de manière analytique aussi que les codes FG-LDPC sont meilleurs que les codes LDPC aléatoires. Ils ont étudié l'intervalle du spectre des mêmes codes, mais pour une matrice de parité réduite (en considérant seulement les premières p lignes de $\mathbf{H}$) et ils ont découvert que plus la matrice considérée a plus de lignes linéairement dépendantes, plus les histogrammes obtenus sont meilleurs

(c'est-à-dire que l'intervalle du spectre a des valeurs plus grandes).

Pour un code binaire linéaire (n, k) représenté par sa matrice de parité $\mathbf{H}$ et ayant seulement des pseudo mots-codes minimaux binaires ω, $w_p(\omega) = w_p(H) \leq n - k + 1$.

Dans [48], les auteurs font aussi une analyse des pseudo-mots-codes minimaux et leurs pseudo-poids.

Pour certains codes LDPC (comme les codes basés sur les géométries finies), les seuls pseudo mots-codes minimaux sont les multiples des mots-codes minimaux [48].

Étant donné un code binaire linéaire caractérisé par sa matrice de parité $\mathbf{H}$, si son graphe de Tanner a le périmètre $g \geq 6$ et la matrice $\mathbf{H}$ a le poids du colonne γ, alors la distance minimale du code $d \geq d_L$ [9], où :

$$d_L = \begin{cases} 1 + \gamma + ... + \gamma(\gamma - 1)^{\frac{g-6}{4}}, & \text{si } \frac{g}{2} \text{ est impair} \\ 1 + \gamma + ... + \gamma(\gamma - 1)^{\frac{g-8}{4}} + ((\gamma - 1)^{\frac{g-4}{4}}), & \text{si } \frac{g}{2} \text{ est impair.} \end{cases} \tag{5.6}$$

L'ensemble d'arrêt minimal $s(\mathbf{H}) \geq d_L$ [49] et, de plus, le pseudo-poids minimal $d_p(\mathbf{H}) \geq d_L$ [50].

Étant donné un code binaire linéaire (n, k, d) et $\mathbf{H}$ sa matrice de parité avec un poids de colonne γ, si toutes les paires de colonnes de $\mathbf{H}$ ont au plus un '1' en commun et γ/λ est un entier et $d = \gamma/\lambda + 1$, alors :

$$s(\mathbf{H}) = d_p(\mathbf{H}) = d \tag{5.7}$$

et

$$T_s(\mathbf{H}) = B_p(\mathbf{H}) = A_d \tag{5.8}$$

où $s(\mathbf{H})$ est la distance d'arrêt, $d_p(\mathbf{H})$ est le pseudo-poids minimal, d la distance minimale du code, A_d le nombre de mots-codes minimaux, $T_s(\mathbf{H})$ est le nombre des plus petits ensembles d'arrêt, et $B_p(\mathbf{H})$ le nombre des lignes avec pseudo-poids minimaux. Dans ce cas le décodage LP est optimal pour ce code.

Pour les codes LDPC, la seule condition nécessaire et suffisante pour avoir ces relations est $d = \gamma + 1$ (ce qui est le cas aussi des codes EG(m, q) et PG(m, q) avec $m = 2$ et aussi pour d'autres valeurs du paramètre m). *Donc le décodage LP est optimal de manière asymptotique pour les codes basés sur les deux types de géométries finies.*

Un autre théorème dit que : étant donné un code binaire linéaire (n, k, d) et $\mathbf{H}$ sa matrice de parité avec le périmètre $g \geq 6$ et le poids de colonne minimal γ, si $d = d_L$ alors $s(\mathbf{H}) = d_p(\mathbf{H}) = d$ et $T_s(\mathbf{H}) = B_p(\mathbf{H}) = A_d$.

Chaque mot-code est un pseudo-mot-code aussi et le support d'un pseudo-mot-code est un ensemble d'arrêt [51], donc $d_p(\mathbf{H}) \leq s(\mathbf{H}) \leq d$ pour n'importe quelle matrice de parité $\mathbf{H}$.

Pour obtenir une bonne performance on doit trouver de bonnes matrices de parité $\mathbf{H}$ pour maximiser le $d_p(\mathbf{H})$ et minimiser le $B_p(\mathbf{H})$.

Les auteurs proposent le problème de recherche suivant : construire pour un code

LDPC binaire une matrice de parité **H** avec un nombre minimal de lignes tel que le pseudo-poids minimal du code est égal à la distance minimale du code et le nombre de mots de pseudo-poids minimal est égal au nombre des mots-codes minimaux du code. Dans ces conditions, le décodage LP sera optimal de manière asymptotique pour ce code. Jusqu'à maintenant on ne sait pas s'il existe une telle matrice pour chaque code binaire linéaire.

5.5 Conclusion

Dans ce chapitre, on a présenté en détails les ensembles de piégeage. On a présenté une classification et on a donné des exemples obtenus par simulations. Les ensembles de piégeage déterminent les mots-codes non-décodés à la fin des itérations du décodage SPA. On a présenté différentes méthodes pour obtenir les ensembles de piégeage (i.e. par simulation, par le problème de la coloration des graphes, par la transmission des impulsions de bruit etc.) et pour analyser la performance d'erreur du code en fonction d'elles. On a introduit la notion du décodage SPA moyenné et on a montré son influence sur les ensembles de piégeage (mots-codes qui ne peuvent pas être décodés). Cet algorithme élimine les oscillations de décodage dues aux ensembles de piégeage, en dirigeant la décision vers une solution moyenne (un plancher d'erreur stable) pour chaque mot-code.

On a présenté aussi les pseudo-mots-codes et leurs pseudo-poids et on a donné les limites pour les pseudo-poids des codes EG-LDPC et PG-LDPC. Pour les codes PG$(2, q)$ et EG$(2, q)$ avec $q = 2$, $q = 4$ et $q = 8$ les auteurs [47] ont trouvé tous les mots-codes, les

mots-codes minimaux et les pseudo mots-codes minimaux et ont donné des polynômes énumérateurs des poids pour les trois types de canaux possibles : AWGN, BSC, BEC. On a conclu que le décodage linéaire est de manière asymptotique le meilleur pour les codes basés sur les géométries finies.

Dans la dernière partie, on a présenté une revue de travaux récents sur les relations entre le périmètre et la distance minimale d'un code LDPC et aussi entre la distance d'arrêt, le pseudo-poids minimal, la distance minimale, le nombre de mots-codes minimaux, le nombre des plus petits ensembles d'arrêt et le nombre des lignes avec pseudo-poids minimaux.

Chapitre 6

Sur la dépendance des poids des vecteurs d'erreur et des poids des syndromes

6.1 Introduction

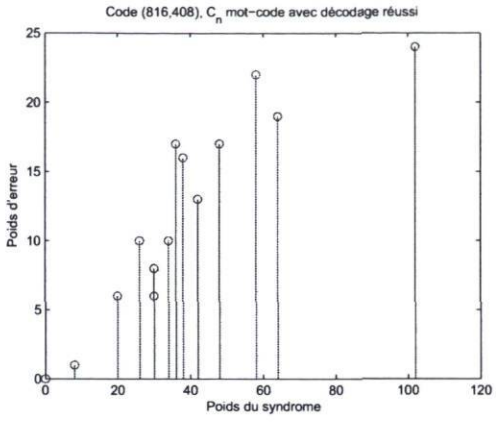
Dans ce chapitre, on étudie la dépendance entre le poids des vecteurs d'erreur et le poids de syndromes des mots-codes des codes LDPC, afin de pouvoir améliorer la performance ou la vitesse des algorithmes de décodage SPA. On analyse l'effet du décodage moyenné sur les poids des vecteurs d'erreur et les poids des syndromes du code et on vérifie si l'algorithme de décodage SPA modifié (Section 3.4) prend vraiment la meilleure décision. On fait aussi l'étude de la corrélation entre les deux vecteurs de poids et on calcule le coefficient de corrélation entre eux. À la fin du chapitre, on exprime la distribution des poids des vecteurs d'erreur et du syndrome pour chaque mot-code par un

polynôme énumérateur de poids.

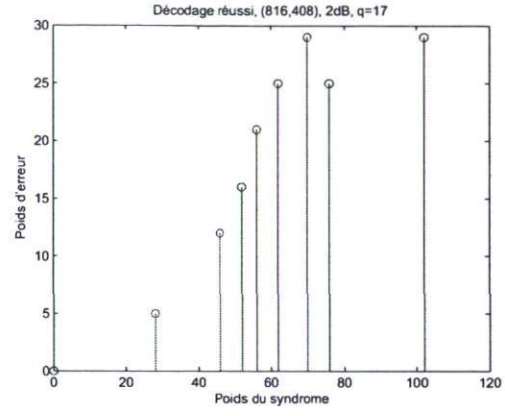
6.2 Exemple numérique et simulations

Pour notre travail de recherche, on a choisi le même code que dans la section 5.3.1, i.e. un code pseudo-aléatoire de dimensions $(816, 408)$, de rendement $R = 1/2$, de poids de colonne $\gamma = 4$ et de poids de ligne $\rho = 8$. Parmi les 100 mots-codes choisis aléatoirement, avec un rapport signal à bruit sur le canal AWGN $Eb/No = 2$ dB, on a trouvé 21 mots-codes qui, à la fin des itérations, ne peuvent pas être bien décodés. On a considéré pour cette étude 300 itérations pour le décodage.

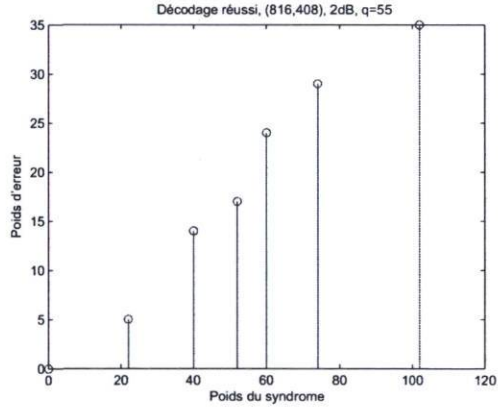
On a illustré sur des graphiques la dépendance du poids du vecteur d'erreur en fonction du poids du syndrome correspondant pour des mots-codes qui peuvent bien être décodés (Fig. 6.1) et aussi pour des mots-codes qui ne peuvent pas être décodés après un nombre élevé d'itérations (Fig. 6.2). On peut voir qu'à une valeur du poids du syndrome peuvent correspondre plusieurs valeurs du poids d'erreur et à une valeur du poids d'erreur peuvent correspondre plusieurs valeurs du poids du syndrome. On observe une dépendance presque proportionnelle pour les poids des vecteurs d'erreur et les poids des syndromes pour les mots-codes qui convergent vers la bonne solution, c'est-à-dire vers des poids du syndrome et d'erreur nuls (Fig. 6.1). Or, ce n'est pas la situation pour les mots-codes qui ne convergent pas vers le bon mot-code : le poids du vecteur d'erreur n'augmente plus avec le poids du syndrome au cours des itérations du décodage (Fig. 6.2). Les poids restent dans la même gamme de valeurs à travers le processus de décodage itératif.



(a) Code $(816, 408)$, C_2 mot-code avec décodage réussi, 2 dB

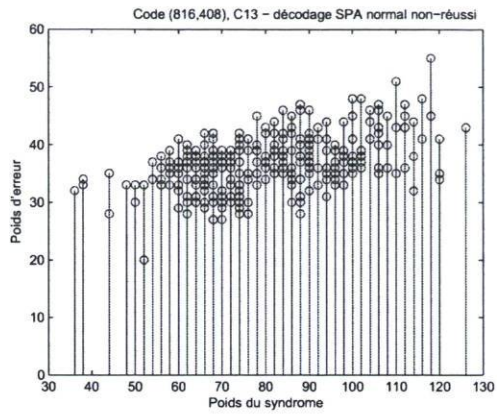


(b) Code $(816, 408)$, C_{17} mot-code avec décodage réussi, 2 dB

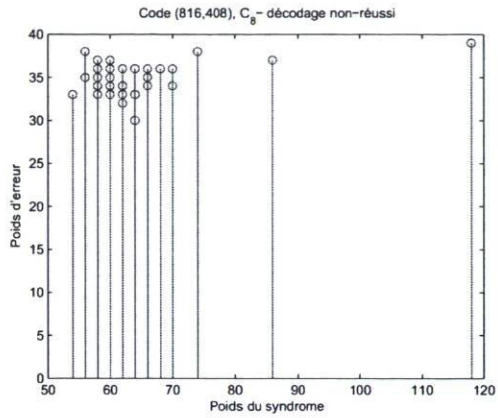


(c) Code $(816, 408)$, C_{55} mot-code avec décodage réussi, 2 dB

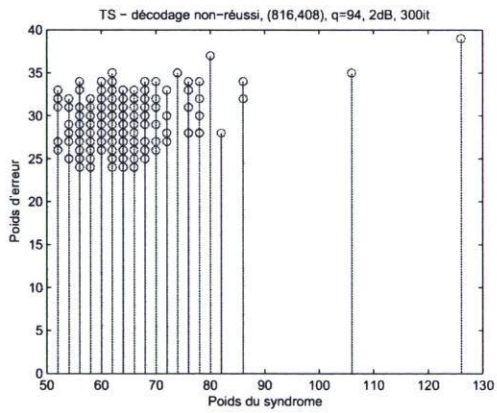
FIG. 6.1 – Évolution des poids d'erreur en fonction des poids des syndromes pendant le décodage SPA pour différents mots-codes qui sont décodés jusqu'à la fin des itérations : a) C_2 , b) C_{17} , c) C_{55}.



(a) Code $(816, 408)$, C_{13} mot-code avec décodage non-réussi



(b) Code $(816, 408)$, C_8 mot-code avec décodage non-réussi



(c) Code $(816, 408)$, C_{11} mot-code avec décodage non-réussi

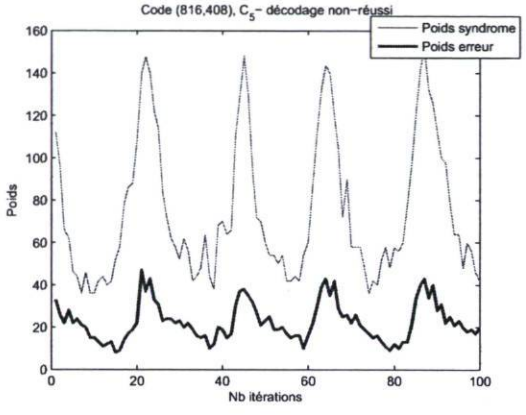
FIG. 6.2 – Évolution des poids d'erreur en fonction des poids des syndromes pendant le décodage SPA pour différents mots-codes qui ne peuvent pas être décodés : a) C_{13} , b) C_8 , c) C_{11}.

Avec la version améliorée de l'algorithme de décodage SPA proposée, à la fin des itérations on prend une décision correspondant au dernier syndrome de poids minimal. On a observé que pour les mots-codes qui ne peuvent pas être décodés on minimise l'erreur par cette version de l'algorithme SPA. Cependant, on ne prend pas toujours la meilleure décision : un syndrome de poids minimal ne correspond pas nécessairement au poids du vecteur d'erreur le plus petit.

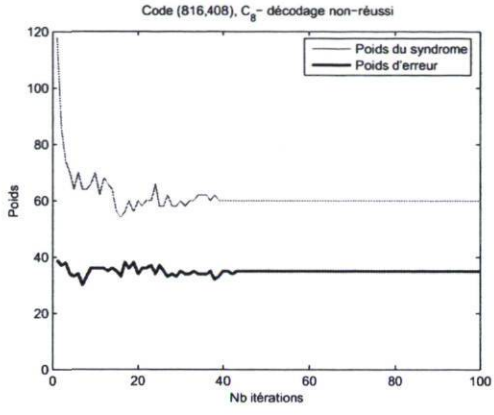
Pour trouver une relation entre les poids des syndromes et les poids des vecteurs d'erreur correspondants, on les a représenté sur la même figure en fonction du nombre d'itérations (Fig. 6.3). On observe que parfois, il n'y a pas de dépendance claire entre l'évolution de poids d'erreur et le poids du syndrome lui-même.

6.3 Effet de l'algorithme de décodage SPA moyenné sur les poids des vecteurs d'erreur et les poids des syndromes

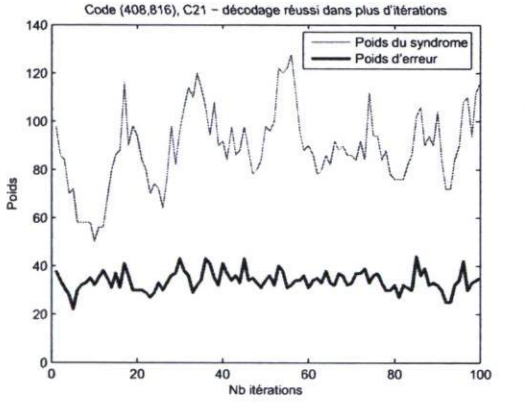
On a étudié l'effet du décodage SPA moyenné sur le poids des vecteurs d'erreur et le poids du syndrome : on observe une réduction significative du nombre des valeurs possibles des poids pour les mots-codes avec grandes oscillations, parce qu'ils convergent très rapidement vers le plancher d'erreur à partir duquel les poids vont commencer à avoir une valeur constante jusqu'à la dernière itération. On peut observer ces résultats à la figure 6.4, qui montre l'évolution du poids d'erreur en fonction du poids de syndrome



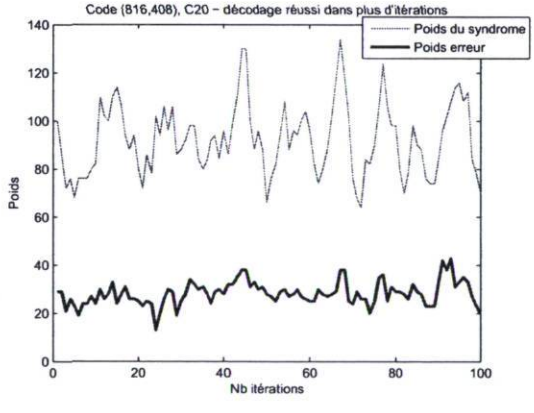
(a) C5, code (816, 408), décodage non-réussi



(b) C8, code (816, 408), décodage non-réussi



(c) C21, code (816, 408), décodage réussi après plusieurs itérations



(d) C20, code (816, 408), décodage réussi après plusieurs itérations

FIG. 6.3 – Évolution des poids des vecteurs d'erreur et des poids des syndromes pendant le décodage de différents mots-codes qui ne peuvent pas être décodés.

pour les mêmes deux mot-codes d'un code LDPC pseudo-aléatoire $(816, 408)$ pendant le décodage SPA normal et le décodage SPA moyenné.

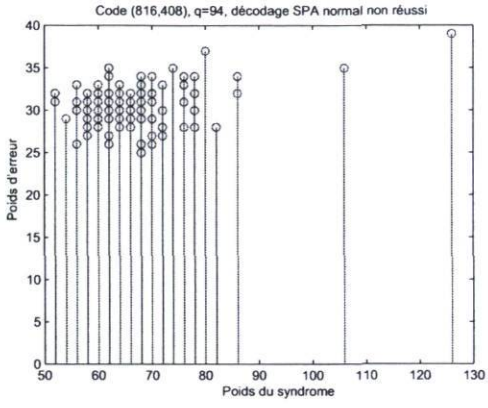
Comme exemple, nous avons choisi un code de Gallager $(20, 15)$ et on a fait la même recherche. Pour un rapport signal à bruit $Eb/No = 3$ dB et 50 itérations de décodage SPA on a transmis tous les $2^5 = 32$ mots-codes possibles et on a trouvé 4 mots-codes qui ne peuvent pas être décodés. Ce résultat dépend beaucoup de la séquence de bruit ajoutée à chaque mot-code. On a suivi l'évolution des probabilités d'erreurs (Fig. 6.5(a), 6.6(a)), les poids du syndrome et les poids d'erreur pendant les itérations du décodage SPA (Fig. 6.5(b)), 6.6(b), 6.5(c), 6.6(c)). Une représentation 3D de la distribution des poids est montrée dans (Fig. 6.5(d), 6.6(d), 6.5(e), 6.6(e)). Le polynôme énumérateur de poids des erreurs $T(X, Y)$ en deux variables, pour exprimer cette distribution pour chaque mot-code, est donné par :

$$T(X, Y) = \sum a_{ij} * X^i * Y^j \tag{6.1}$$

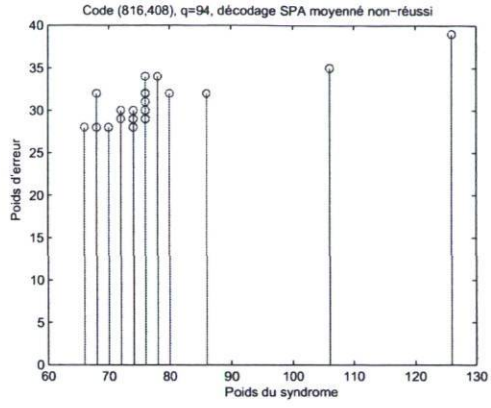
où i est le poids des vecteurs d'erreur, j le poids du syndrome et a_{ij} représente le nombre de séquences d'erreur de poids i correspondant à un syndrome de poids j pour un mot-code donné.

Dans nos tests, pour le 23-ème mot-code, C_{23}, on obtient le polynôme suivant :

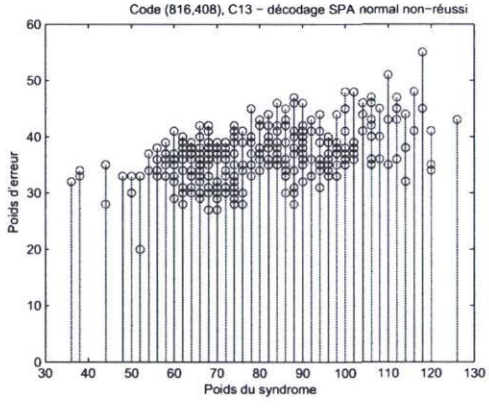
$$T_{23}(X, Y) = 8X^2Y^3 + X^2Y^4 + 4X^2Y^5 + 2X^2Y^6 + 34X^3Y^2 + X^3Y^3 \tag{6.2}$$



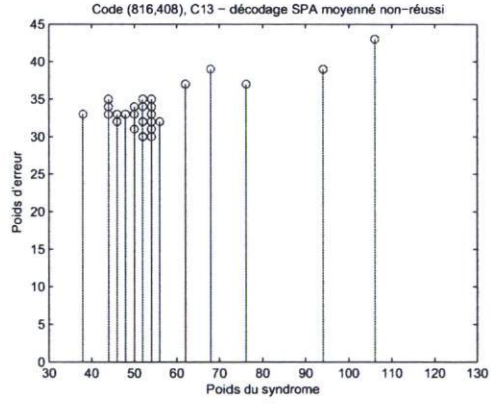
(a) Code $(816, 408)$, $C_1 1$, décodage SPA normal non-réussi



(b) Code $(816, 408)$, $C_1 1$, décodage SPA moyenné non-réussi



(c) Code $(816, 408)$, $C_1 3$, décodage SPA normal non-réussi



(d) Code $(816, 408)$, $C_1 3$, décodage SPA moyenné non-réussi

FIG. 6.4 – Effets du décodage SPA moyenné sur la distribution des poids d'erreur et des poids des syndromes pour différents mots-codes qui ne peuvent pas être décodés.

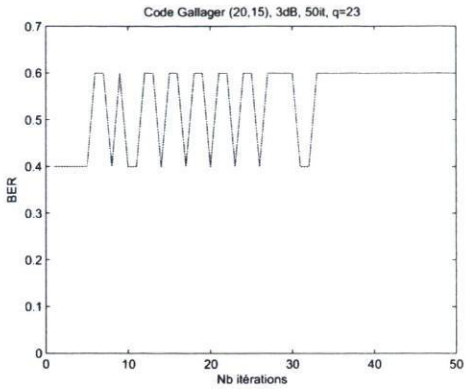
Pour le 24-ème mot code, C_{24}, on obtient la distribution des poids suivante :

$$T_{24}(X,Y) = XY^3 + XY^4 + 4XY^6 + 2XY^8 + 5XY^9 + 6X^2Y^2 + 4X^2Y^3 + 3X^2Y^5 + 15X^2Y^6 + 2X^2Y^8 + X^2Y^9 + 2X^2Y^{10} + 3X^3Y^{10} + X^3Y^{11} \tag{6.3}$$

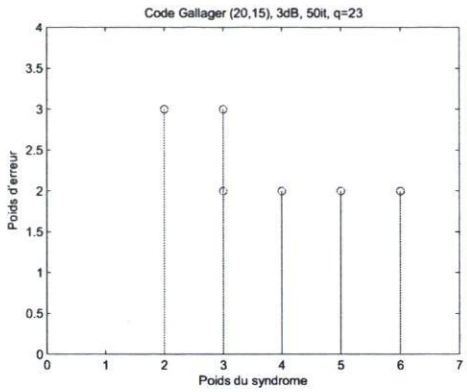
Pour mieux visualiser la dépendance entre le poids des vecteurs d'erreur et le poids des syndromes, on a fait aussi une représentation 3D de la distribution des poids pour 100 itérations d'un décodage SPA normal (Fig. 6.7(a)), et d'un décodage SPA moyenné (Fig. 6.7(b)). En analysant aussi les figures Fig. 6.8(b) et la Fig. 6.9(b), on peut voir que, grâce au moyennage, les gammes des valeurs des poids des vecteurs d'erreur et des poids du syndrome sont beaucoup réduites. En comparant la Fig. 6.8(a) avec la Fig. 6.9(a) on peut remarquer que, par moyennage, les oscillations de la probabilité d'erreur par bit convergent vers une valeur constante après un nombre relativement petit d'itérations.

6.4 Corrélation entre le vecteur des poids d'erreur et le vecteur des poids des syndromes pendant les itérations du décodage SPA

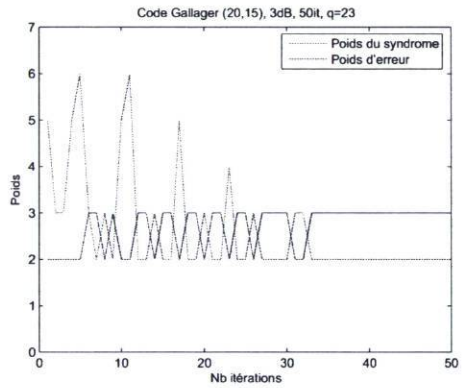
On désire trouver une méthode pour augmenter la vitesse de calcul pour la version modifiée du décodage SPA, en trouvant une règle selon laquelle le mot code détecté s'approche ou s'éloigne du mot code original transmis (ce qui veut dire une règle selon laquelle le syndrome calculé s'approche ou s'éloigne en termes de distance de Hamming



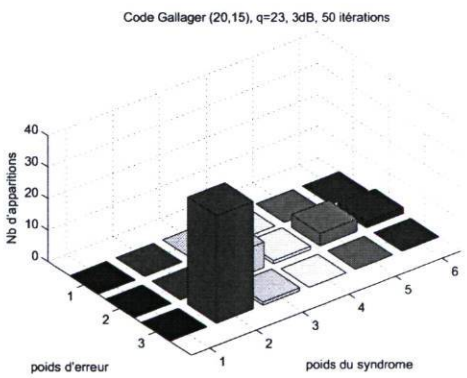
(a)



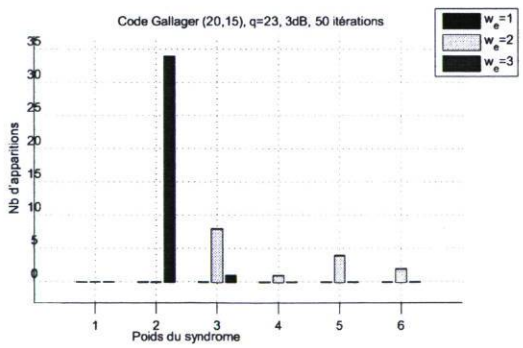
(b)



(c)

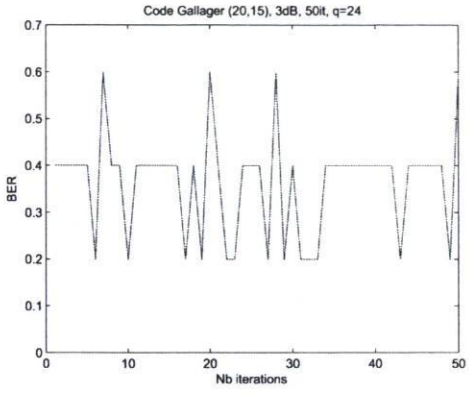


(d)

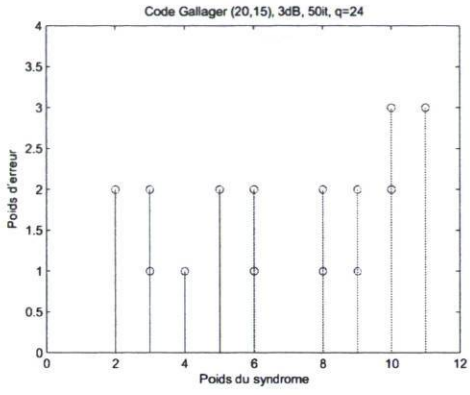


(e)

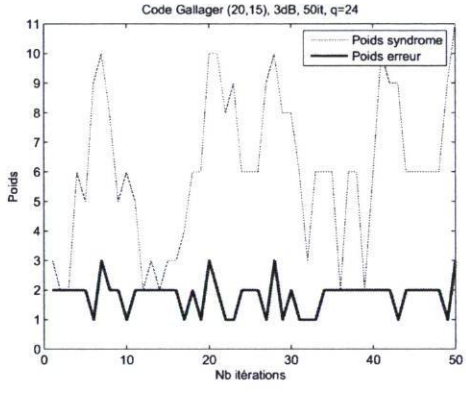
FIG. 6.5 – Effets du décodage SPA moyenné sur la distribution des poids d'erreur et des poids des syndromes pour différents mots-codes qui ne peuvent pas être décodés.



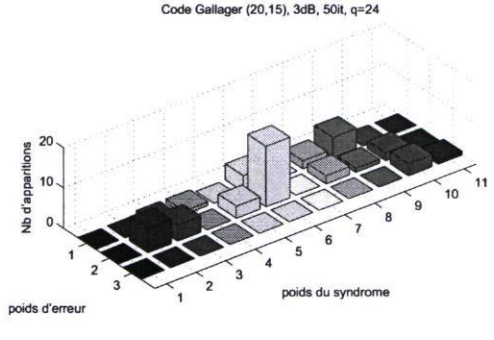
(a)



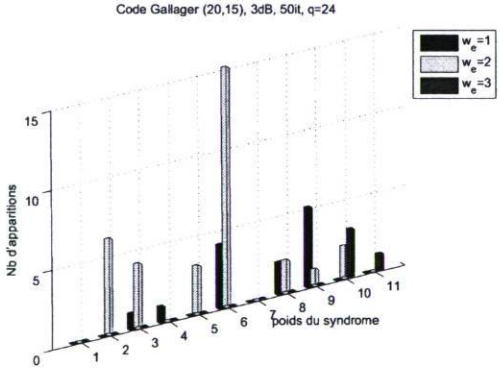
(b)



(c)

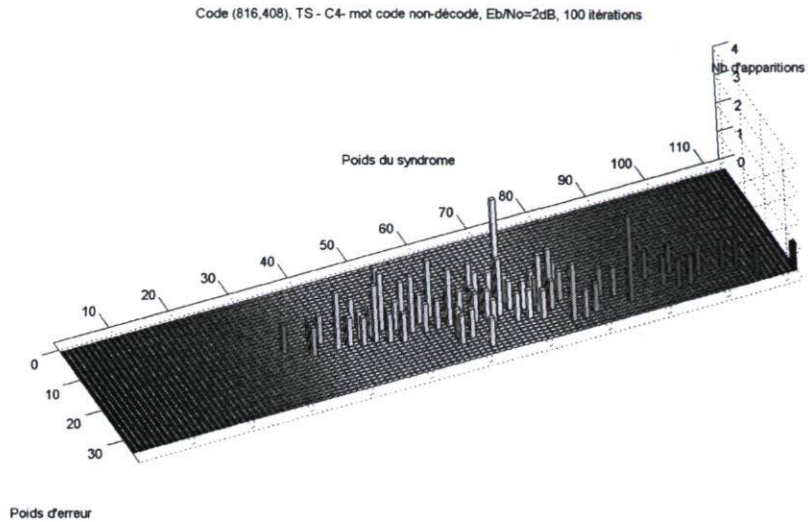


(d)

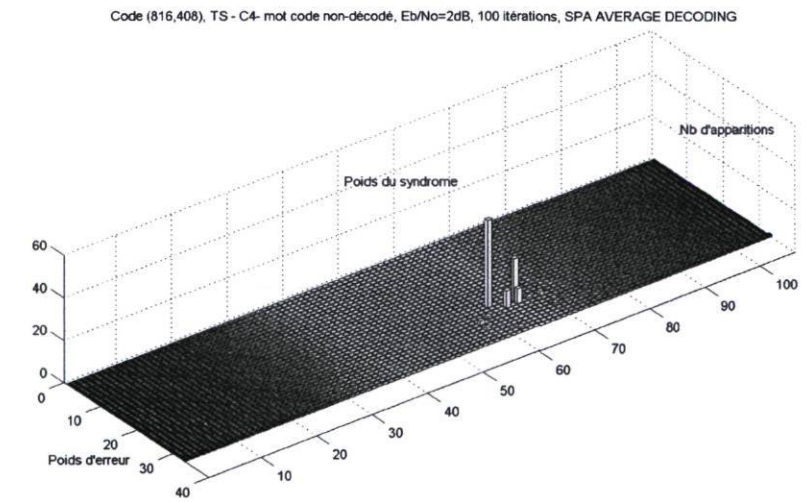


(e)

FIG. 6.6 – Effets du décodage SPA moyenné sur la distribution des poids d'erreur et des poids des syndromes pour différents mots-codes qui ne peuvent pas être décodés.

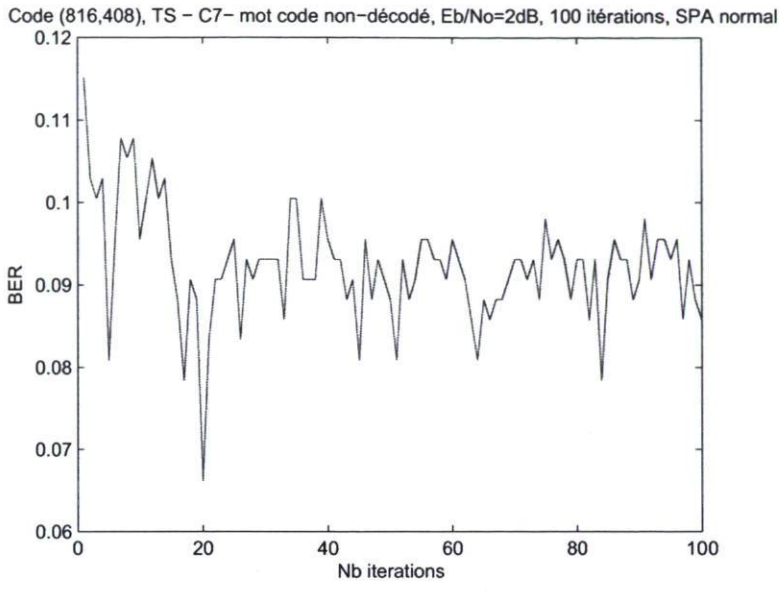


(a)

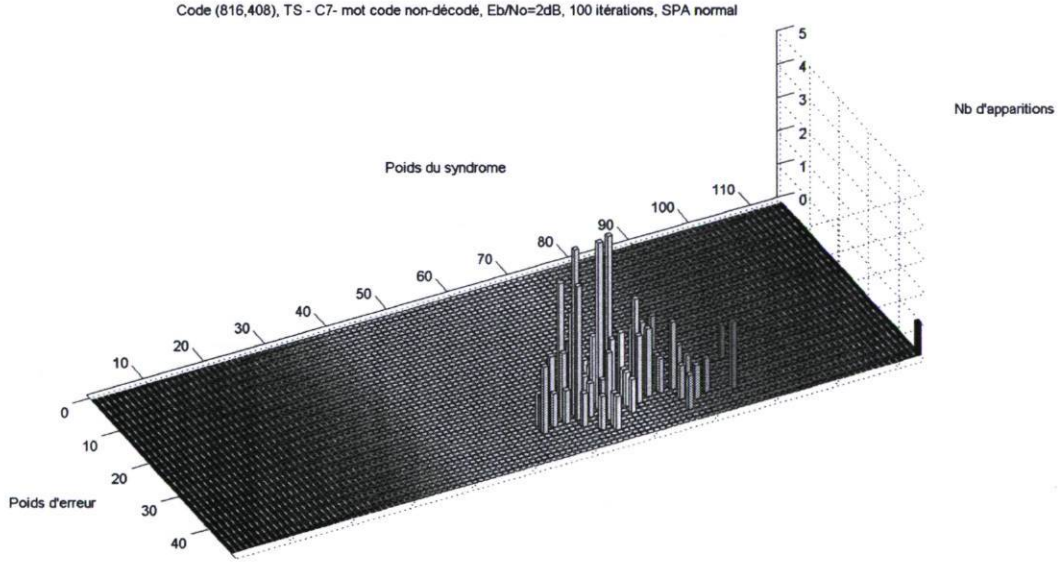


(b)

FIG. 6.7 – Effets du décodage SPA moyenné sur la distribution des poids d'erreur et des poids des syndromes pour le mot-code C4 qui ne peut pas être décodé.

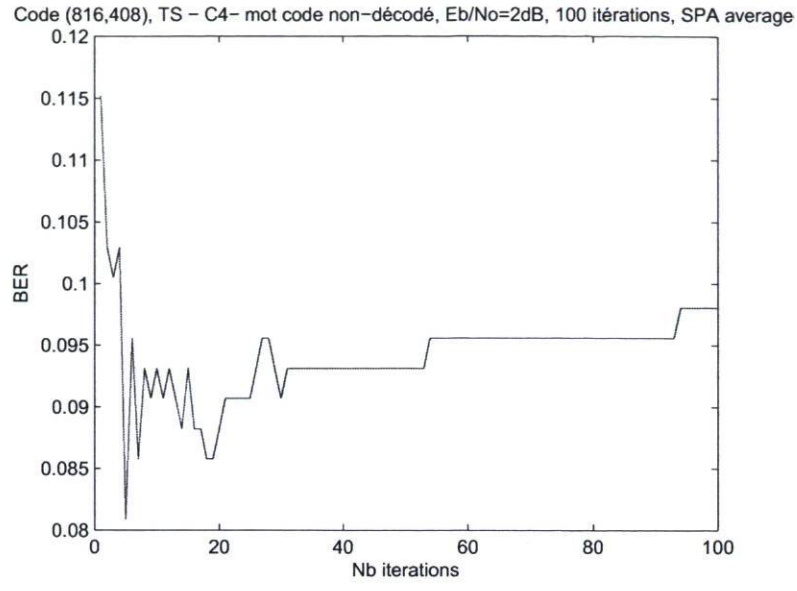


(a)

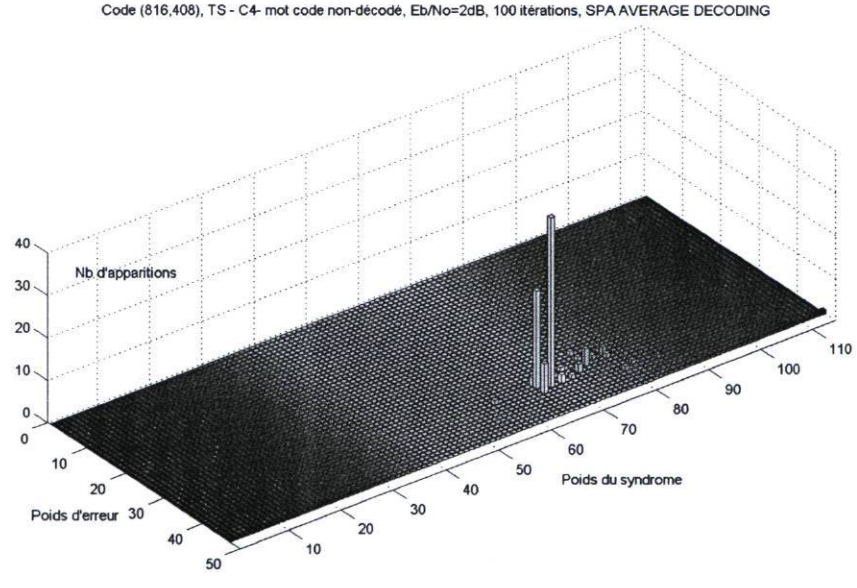


(b)

FIG. 6.8 – Effets du décodage SPA sur la distribution des poids d'erreur et des poids des syndromes pour le mot-code C7 qui ne peut pas être décodé.



(a)



(b)

FIG. 6.9 – Effets du décodage SPA moyenné sur la distribution des poids d'erreur et des poids des syndromes pour le mot-code C4 qui ne peut pas être décodé.

du vecteur nul). Ce problème a été étudié pour un code géométrie projective PG-LDPC $(273, 191)$. Parmi les 3000 mots-codes transmis, 14 mots-codes n'ont pu être décodés en 50 itérations. Parmi ceux-ci, 4 mots-codes ont pu être éventuellement décodés (en augmentant le nombre de itérations), mais les autres ne peuvent toujours pas être décodés après 500 itérations. Ce problème est vraisemblablement causé par l'existence des ensembles de piégeage.

Les graphiques dans la figure 6.10 illustrent les poids d'erreur et les poids du syndrome. Les coefficients de la droite qui passe parmi les points (au sens des moindres carrés) sont déterminés par régression linéaire : on a représenté sur les figures la droite de moindres carrés, dont la pente indique la corrélation entre les valeurs sur l'axe x (le poids du syndrome) et les valeurs sur l'axe y (le poids d'erreur). Si la droite est horizontale il n'y a pas de corrélation significative entre les variables. Si la droite croît, il y a une corrélation positive entre les deux variables et si la droite décroît, alors la corrélation existante est négative. Plus la pente de la droite est élevée, plus la corrélation entre les deux variables est forte. Il faut cependant faire attention à l'interprétation des résultats parce que cette pente nous indique une dépendance linéaire entre les deux variables, mais on ne peut pas déceler une relation non-linéaire (par exemple s'ils sont corrélées de manière exponentielle), mais ici, ce n'est pas le cas. On doit aussi mentionner que même si la pente de la droite et les coefficients de corrélation indiquent une grande corrélation entre les variables, il n'y a pas nécessairement une relation de causalité entre elles.

Selon la formule $w_p \geq q + 5$, le pseudo-poids est limité par $q + 5$. Pour le code PG-LDPC $(273, 191)$ choisi, $q = 2^4 = 16$ et donc $w_p \geq 21$. La distance minimale du code est $d_{min} = \gamma + 1 = 18$.

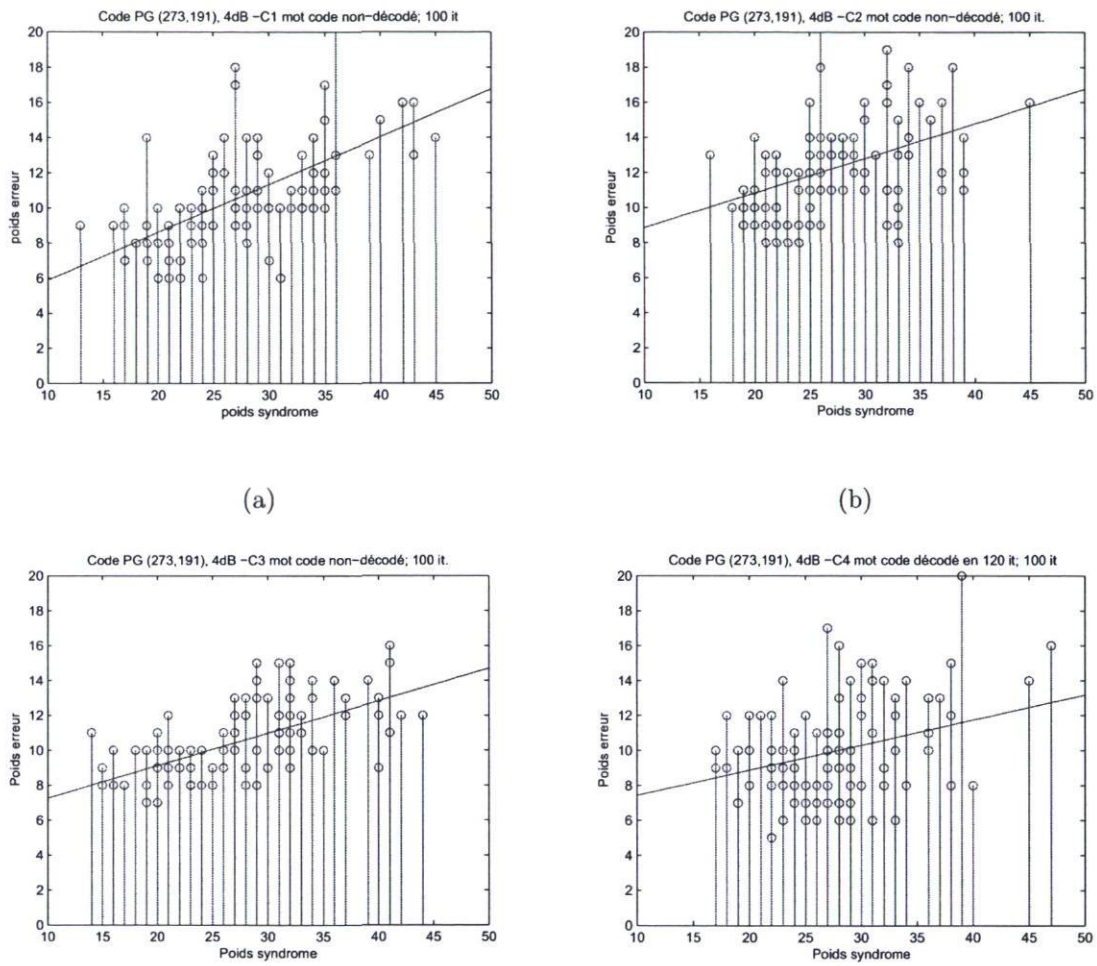
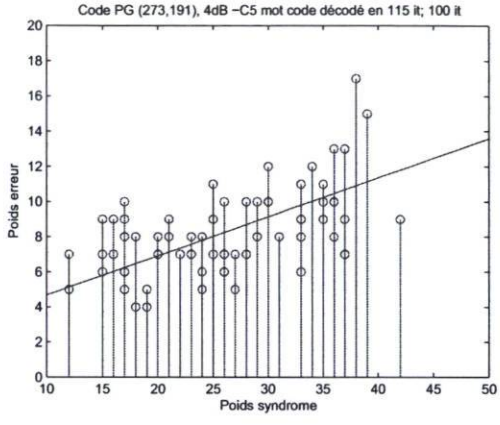
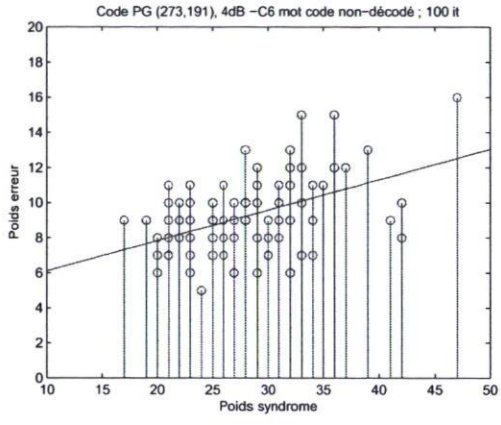


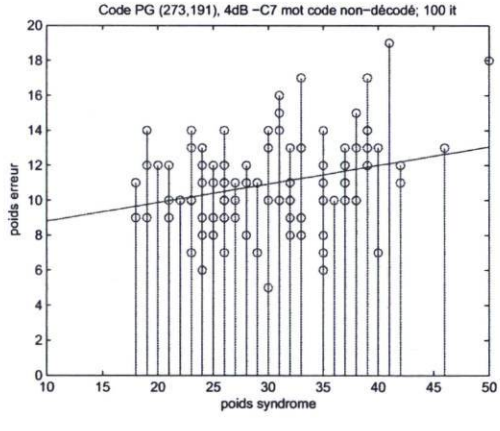
FIG. 6.10 – Corrélation entre le poids des vecteurs d'erreur et le poids du syndrome pendant le décodage SPA pour différents mots-codes qui ne peuvent pas être décodés en 50 itérations (début).



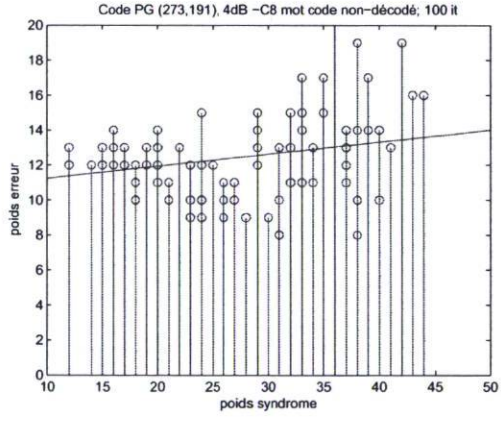
(e)



(f)

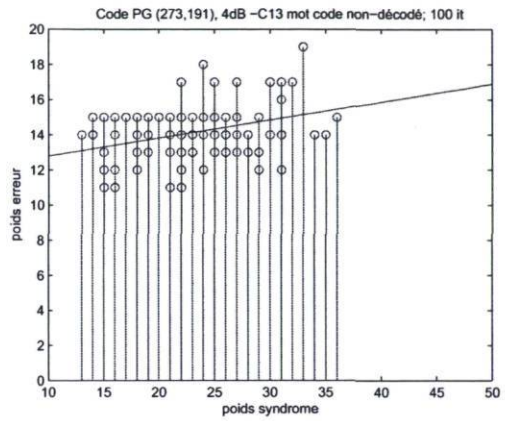


(g)

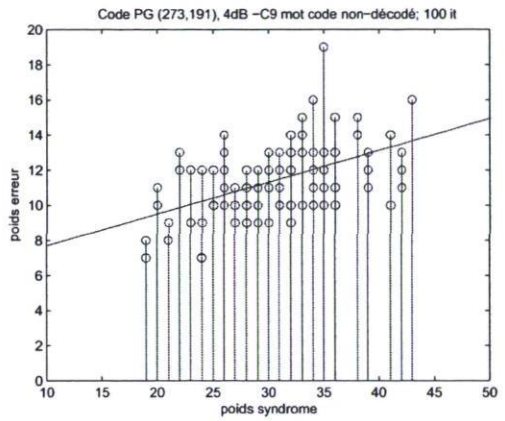


(h)

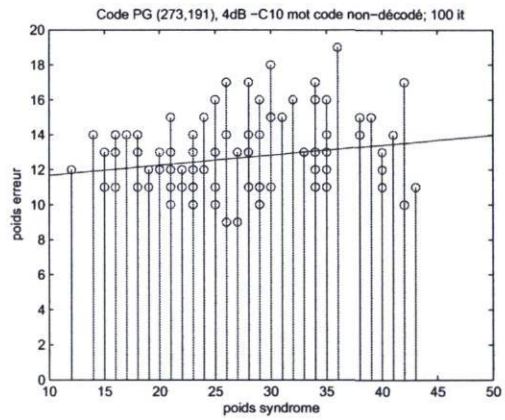
FIG. 6.10 – Corrélation entre le poids des vecteurs d'erreur et le poids du syndrome pendant le décodage SPA pour différents mots-codes qui ne peuvent pas être décodés en 50 itérations (suite).



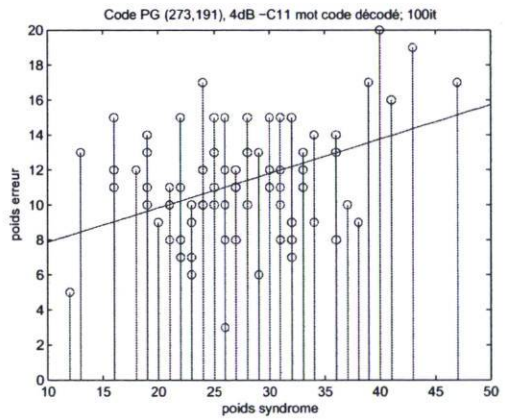
(i)



(j)

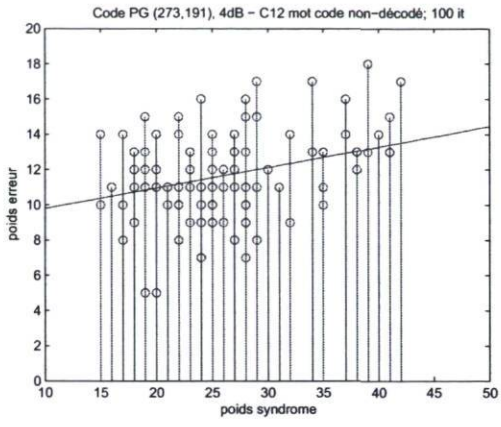


(k)

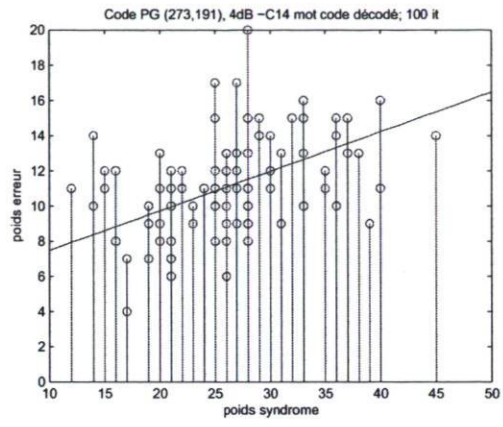


(l)

FIG. 6.10 – Corrélation entre le poids des vecteurs d'erreur et le poids du syndrome pendant le décodage SPA pour différents mots-codes qui ne peuvent pas être décodés en 50 itérations (suite).



(m)



(n)

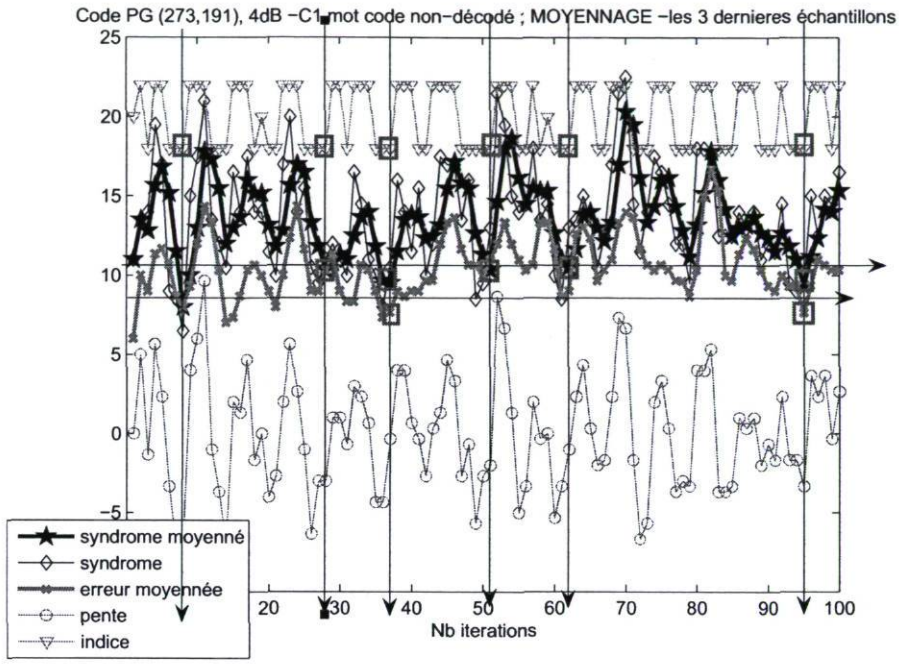
FIG. 6.10 – Corrélation entre le poids des vecteurs d'erreur et le poids du syndrome pendant le décodage SPA pour différents mots-codes qui ne peuvent pas être décodés en 50 itérations (fin).

On a représenté sur la même figure (Fig. 6.11) en fonction du nombre d'itérations, l'évolution du poids du vecteur d'erreur et du poids de syndrome. On a ajouté la moyenne avec les 3 derniers échantillons pour les courbes du syndrome et du vecteur d'erreur et la courbe de la pente du syndrome moyenné. Ensuite on donne un indice indiquant les oscillations de la pente entre valeurs négatives, nulles ou positives. À l'aide de ces paramètres, on a essayé de trouver une règle permettant de prendre la meilleure décision pour les mots-codes qui ne peuvent pas être décodés. Encore une fois avec le critère proposé, i.e. de choisir l'erreur qui correspond au syndrome minimum, le résultat est amélioré, mais on ne prends pas toujours la meilleure décision. Parfois la corrélation entre le poids du syndrome et le poids d'erreur est plus grand, parfois plus petit : on ne peut donc pas utiliser un critère général de décision basé sur ces paramètres, même si pour quelques mots-codes cela fonctionne (par exemple en prenant un minimum de la

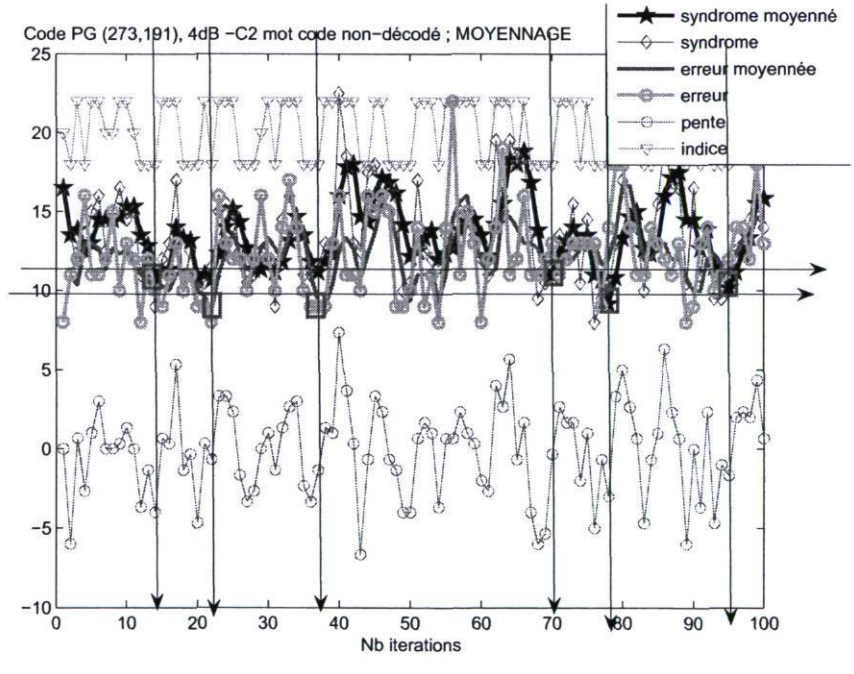
courbe du syndrome moyenné qui correspond parfois aussi au minimum local du poids d'erreur).

On a calculé un coefficient de corrélation qui reflète la corrélation existant entre deux vecteurs de même longueur. Dans notre cas, les deux vecteurs analysés sont le poids du vecteur d'erreur et le poids du syndrome pendant un nombre d'itérations donnés. Ce coefficient a des valeurs comprises entre 0 et 1 (0 indiquant aucune corrélation entre les vecteurs et 1 indiquant une corrélation parfaite). Il y a aussi une valeur de probabilité p qui indique si le coefficient de corrélation calculé correspond au niveau de corrélation réel entre les deux vecteurs ou si on pourra obtenir le même coefficient aussi pour deux vecteurs qui sont aléatoires et il n'y a aucune corrélation entre eux. Si cette valeur est en dessous de 0.05, ceci implique que le coefficient de corrélation obtenu est très fiable (il y a une probabilité très petite d'obtenir le même coefficient de corrélation pour deux vecteurs non-corrélés).

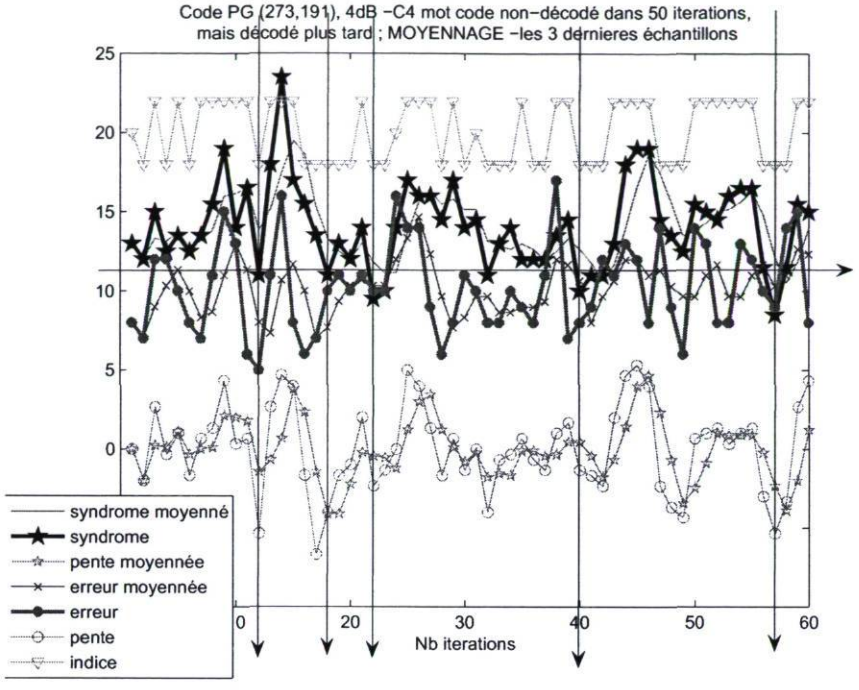
Ces deux coefficients ont été calculés pour tous les 14 mots-codes non-décodés (voir le Tableau 6.1). On peut voir que pour les mots-codes qui convergent (C_4, C_5, C_{11}, C_{14}), il y a une corrélation relativement grande entre les vecteurs et cette corrélation augmente avec le nombre d'itérations. Pour les mots-codes qui ne convergent pas, plus on augmente le nombre d'itérations, plus la valeur de la corrélation diminue (les vecteurs ne convergent pas). On peut expliquer ce phénomène par l'existence des ensembles de piégeage dans le code : après quelques itérations une erreur fixe est présente dans le graphe de Tanner et commence à se propager dans tout le graphe ; on s'éloigne alors de la bonne décision.


Code PG (273,191), 4dB −C1 mot code non−décodé ; MOYENNAGE −les 3 dernieres échantillons
25
20
15
10
5
0
−5
20
30
40
50
60
70
80
90
100
Nb iterations
syndrome moyenné
syndrome
erreur moyennée
pente
indice

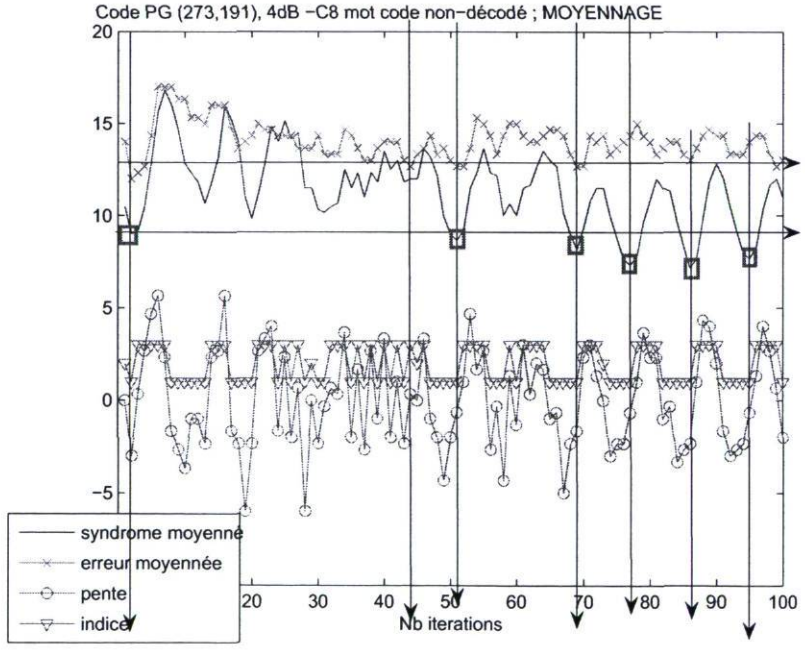
(a)


Code PG (273,191), 4dB −C2 mot code non−décodé ; MOYENNAGE
25
20
15
10
5
0
−5
−10
0
10
20
30
40
50
60
70
80
90
100
Nb iterations
syndrome moyenné
syndrome
erreur moyennée
erreur
pente
indice

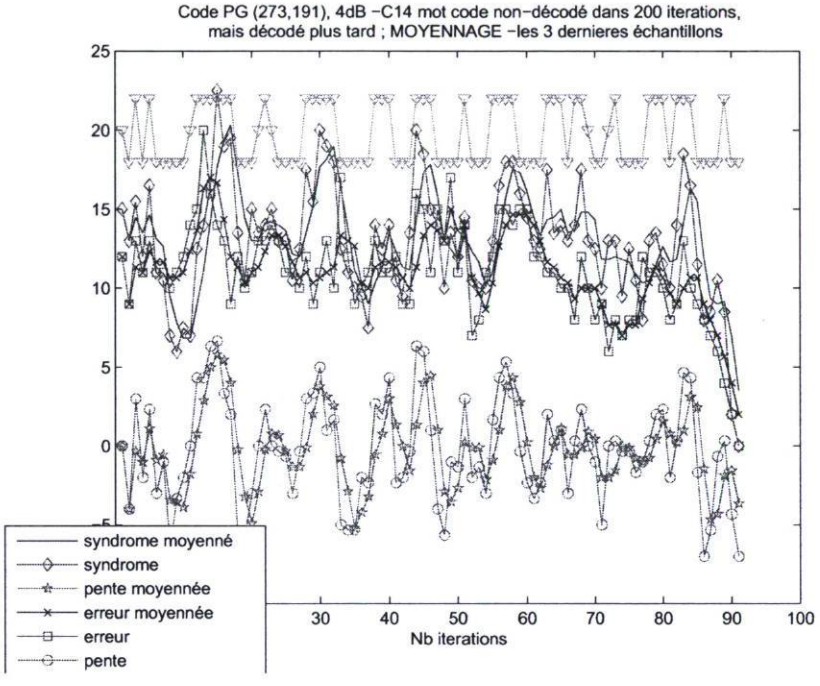
(b)


Code PG (273,191), 4dB −C4 mot code non−décodé dans 50 iterations,
mais décodé plus tard ; MOYENNAGE −les 3 dernieres échantillons
25
20
15
10
5
0
0
20
30
40
50
60
Nb iterations
syndrome moyenné
syndrome
pente moyennée
erreur moyennée
erreur
pente
indice

(c)


Code PG (273,191), 4dB −C8 mot code non−décodé ; MOYENNAGE
20
15
10
5
0
−5
20
30
40
50
60
70
80
90
100
Nb iterations
syndrome moyenné
erreur moyennée
pente
indice

(d)



(e)

FIG. 6.11 – Analyse de différents paramètres des différents mots-codes non-décodés pendant les itérations de l'algorithme du décodage SPA.

$Mot-code$	50 it.	100 it.	200 it.
C_1	0.6411	0.5933	0.5866
C_2	0.5379	0.4166	0.4619
C_3	0.5809	0.6164	0.6109
C_4	0.3500	0.2934	0.4861
C_5	0.5288	0.6758	0.6758
C_6	0.6128	0.4816	0.4145
C_7	0.3833	0.2737	0.1969
C_8	0.3887	0.2649	0.2704
C_9	0.5701	0.5162	0.5383
C_{10}	0.0951	0.2025	0.1908
C_{11}	0.2700	0.4284	0.4284
C_{12}	0.2706	0.3090	0.3170
C_{13}	0.4710	0.3557	0.3170
C_{14}	0.2474	0.5496	0.5496

TAB. 6.1 – Valeurs de coefficients de corrélation entre les vecteurs des poids d'erreur et les vecteurs des poids de syndromes pour différents mot-codes du code LDPC à géométrie projective PG-LDPC $(273, 191)$.

6.5 Conclusion

Dans ce chapitre, on conclut que pour les mots-codes qui peuvent être bien décodés, il y a une corrélation relativement grande entre le vecteur des poids d'erreur et le vecteur des poids des syndromes. La valeur de cette corrélation augmente avec le nombre d'itérations, c'est-à-dire lorsqu'ils convergent vers un syndrome nul (vers le bon mot-code). Pour les mots-codes qui ne convergent pas, après un certain nombre d'itérations du décodage SPA, plus on ajoute des itérations, plus la corrélation baisse (et les vecteurs s'éloignent de la bonne décision).

Par le décodage SPA modifié (section 3.4), on minimise l'erreur pour les mots-codes qui ne peuvent pas être décodés, mais on ne prend pas toujours la meilleure décision : un syndrome de poids minimal ne correspond pas nécessairement au poids du vecteur d'erreur le plus petit.

L'effet du décodage SPA moyenné sur la distribution des poids des vecteurs d'erreur et des poids des syndromes est étudié également. Grâce au moyennage, la gamme de valeurs possibles prises par les poids est considérablement réduite. Pour mieux la visualiser, on a fait une représentation 3D de la distribution des poids pendant les itérations d'un décodage SPA normal et d'un décodage SPA moyenné.

Le polynôme énumérateur de poids permet d'exprimer la distribution des poids du vecteur d'erreur et du syndrome pour chaque mot-code pendant le décodage.

Pour trouver la relation entre le vecteur de poids d'erreur et le vecteur de poids des syndromes, on a étudié l'évolution des plusieurs paramètres (le coefficient de corrélation, la pente de la droite de corrélation, le syndrome moyenné, l'erreur moyennée, etc.).

On conclut que l'on ne peut pas utiliser un critère général de décision basé sur ces paramètres.

Chapitre 7

Conclusions

7.1 Synthèse du mémoire

Dans ce travail, nous avons étudié les codes LDPC. Nous avons présenté en premier une revue de l'état de l'art sur différentes méthodes de construction et décodage de ce type de codes. Nous avons poursuivi avec une étude pratique sur l'amélioration des algorithmes de décodage des codes LDPC. À la fin, nous avons proposé différentes suggestions de travaux futurs pour améliorer le performances des codes LDPC.

Premièrement, nous nous sommes concentrés sur la comparaison entre les codes LDPC structurés, basés sur les géométries finies euclidienne ou géométrique et les codes pseudo-aléatoires, de longueur courte. On a conclu que les codes structurés courts ont une meilleure performance, en terme de probabilité d'erreur, que les codes pseudo-aléatoires. En même temps, leurs vitesses de décodage sont plus grandes et ils demandent une complexité des circuits beaucoup plus faible. Pour ces raisons, nous avons concentré notre attention sur les codes courts structurés.

Les codes LDPC ont une performance remarquable sur les canaux AWGN avec le décodage itératif SPA basé sur la propagation de la confiance (BP). Cependant, il reste de mots-codes qui ne peuvent pas être décodés. Une solution déjà existante pour ce problème est l'algorithme de décodage SPA moyenné, qui assure un meilleur décodage des mot-codes qui ne peuvent pas être décodés par l'algorithme SPA conventionnel. Cependant, pour les mots-codes qui sont normalement bien décodés, l'algorithme SPA moyenné produit un décodage moins performant. De plus, même s'il élimine les oscillations dans les poids du syndrome, cet algorithme donne un plancher de probabilité d'erreur plus élevé. Compte tenu de ces problèmes, nous avons proposé un algorithme de décodage log-SPA amélioré basé sur l'idée d'une possible corrélation existante entre le vecteur du poids de syndrome et le vecteur du poids d'erreur. Après un certain nombre d'itérations, le nombre de bits erronés dans le syndrome peut augmenter, ce qui peut aussi augmenter le nombre de bits erronés dans le mot-code estimé. Ceci signifie qu'à la fin des itérations de l'algorithme de décodage log-SPA conventionnel, on ne prend pas nécessairement la meilleure décision. Pour ce faire, nous avons modifié l'algorithme : lorsqu'il doit arrêter après avoir atteint le nombre maximum d'itérations fixé, on choisit le mot-code estimé comme étant celui qui correspond au syndrome le plus proche, au sens de la distance de Hamming, du syndrome nul (c'est-à-dire le syndrome de poids minimal). Nous avons étudié par des simulations l'effet de cette modification dans la performance des différents types de codes LDPC courts. Nous avons observé que cet algorithme réduit les erreurs de décodage pour les mots-codes qui ne peuvent pas être décodés, même s'il ne prend pas toujours la meilleure décision. Toutefois, il conduit à un plancher d'erreur plus bas. Il faut cependant poursuivre l'étude pour arriver à une

règle de décision plus générale.

Pour cette raison, nous avons investigué plus en détail la corrélation entre le vecteur de poids d'erreur et le vecteur de poids du syndrome pendant les itérations de l'algorithme log-SPA. Nous avons utilisé un polynôme énumérateur des poids et nous avons étudié l'évolution des coefficients de corrélation pour représenter la distribution des poids des vecteurs du syndrome et des poids des vecteurs d'erreur pour chaque mot-code pendant le décodage. Pour les mot-codes qui peuvent être bien décodés, il existe une forte corrélation entre les deux vecteurs de poids et la corrélation augmente avec le nombre d'itérations. Cependant, pour les mots-codes qui ne convergent pas, après un certain nombre d'itérations du décodage, la corrélation commence à baisser avec le nombre d'itérations : les vecteurs décodés deviennent de plus en plus éloignés de la solution correcte.

7.2 Contributions et suggestions de travaux futurs

Ce travail a fait l'objet d'un article intitulé "On the correlation between error weights and syndrome weights for belief propagation decoding" présenté à la conférence CWIT2009 (Canadian Workshop for Information Theory) à Ottawa [52].

Des investigations plus avancées sont proposées comme travaux futurs pour trouver un décodage optimal. Comme notre recherche est basée sur les codes LDPC courts, il serait intéressant de faire aussi l'étude du comportement des codes LDPC longs dans ce contexte.

Nous avons aussi abordé les problèmes actuels existants dans la performances d'er-

reur des codes LDPC : le périmètre du graphe de Tanner, la distance minimale du code et le plancher d'erreur. La cause de l'existence des mots-codes ne pouvant pas être décodés, qui résultent en un haut plancher d'erreur, sont les ensembles de piégeage et les ensembles d'arrêt. Les décodages avec l'algorithme SPA avec moyennage ou avec l'algorithme SPA mesuré permettent de réduire le nombre de trames mal-décodées dans la région du plancher d'erreur. Le problème des ensembles de piégeage peut être réduit au cours de la construction des codes.

Le chapitre 6 présente différentes méthodes de construction des codes LDPC quasi-cycliques. Ceux-ci peuvent éviter la formation des certains types d'ensembles de piégeage, d'ensembles d'arrêt ainsi que les cycles courts dans le graphe de Tanner. Les méthodes proposées donnent de très bas planchers d'erreur et une faible complexité du codage. On suggère également comme travaux de recherche futurs une comparaison entre les deux types de codes : quasi-cycliques irréguliers, triangulaires et doublement-diagonaux et les codes basés sur PEG. Pour augmenter la distance minimale du code on propose de remplacer la sous-matrice identité du code proposé par Z. He [29] par une matrice Queen pour accroître la distance minimale du code. Une combinaison de ces méthodes de construction des bons codes QC-LDPC, produira de meilleurs codes QC-LDPC avec des planchers d'erreur plus bas. Il serait aussi très intéressant de combiner les méthodes de décodage amélioré présentées dans le chapitres antérieurs avec les méthodes de construction améliorée des codes QC-LDPC.

Bibliographie

[1] D. J. MacKay et R. M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes", *Electron. Lett.*, vol. 32(18), pp. 1645–46, 1962.

[2] D. J. MacKay, "Gallager Codes That Are Better Than Turbo Codes", in *Proc. 36th Allerton Conf. Commun., Control, and Computing*, 1998. Monticello, III.

[3] T. Richardson, M. Shokrollahi et R. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes", *IEEE Trans. Inform. Theory*, vol. 47(2), pp. 619–637, 2001.

[4] M. C. Davey et D. J. MacKay, "Low Density Parity Check Codes over GF(q)", *IEEE Commun. Lett.*, vol. 2(6), pp. 165–167, 1998.

[5] S. Lin, Y. Kou et M. Fossorier, "Low Density Parity Check Codes Construction Based on Finite Geometries", in *Proc.GLOBECOM 2000*, 2000. San Francisco, Calif., Novembre 27-Decembre 1.

[6] Y. Kou, S. Lin et M. Fossorier, "Construction of Low Density Parity Check Codes-A Geometric Approach", in *Proc. 2nd Intl. Symp. Turbo Codes and Related Topics, Brest, France*, Septembre 2000.

[7] R. Gallager, "Low Density Parity Check Codes", *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–29, 1962.

[8] R. Gallager, *Low Density Parity Check Codes.* MIT Press, Cambridge, 1963.

[9] R. M. Tanner, "A Recursive Approach to Low Complexity Codes", *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, 1981.

[10] Y. Kou, S. Lin et M. Fossorier, "Low Density Parity Check Codes Based on Finite Geometries : A Redicovery", *Proc. IEEE Intl. Symp. Inform. Theory*, 2000. Sorrento, Italy, Juin 25-30.

[11] Y. Kou, S. Lin et M. Fossorier, "Low Density Parity Check Codes Based on Finite Geometries : A Redicovery and More", *IEEE Trans. Inform. Theory*, vol. 47(6), pp. 2711–2736, 2001.

[12] S. Lin et Y. Kou, "A Geometric Approach to the Construction of Low Density Parity Check Codes", *IEEE Trans. Inform. Theory*, 2000. presented at the IEEE 29th Communication Theory Workshop, Haines City, Fla., Mai 7-10.

[13] S. Lin, Y. Kou et M. Fossorier, "Finite Geometry Low Density Parity Check Code : Construction, Structure and Decoding", in *Proc. of the ForneyFest.* Kluwer Academic, Boston, Mass.

[14] S. Lin et D. Costello, *Error control coding.* Prentice Hall, 2004.

[15] N. Wiberg, H.-A. Loeliger et R. Kotter, "Codes and Iterative Decoding on General Graphs", *Eur. Trans. Telecommun.*, vol. 6, pp. 513–26, 1955.

[16] H. Tang, J. Xu, Y.Kou, S. Lin et K. Abdel-Ghaffar, "On Algebraic Construction of Low Density Parity Check Codes", in *Proc. 2002 IEEE Intl. Symp. Inform. Theory*, 2002. p.482, Laussane, Switzerland, Juin 30-Julliet 5.

[17] H. Tang, J. Xu, Y.Kou, S. Lin et K. Abdel-Ghaffar, "On Algebraic Construction of Gallager and Circulant Low Density Parity Check Codes", *IEEE Trans. Inform. Theory*, vol. 50(6), pp. 1269 – 1279, 2004.

[18] S. Liu, L. Chen, J. Xu et I. Djurdjevic, "Near Shannon Limit Quasi-Cyclic Low Density Parity-Check Codes", in *Proc. IEEE GLOBECOM 2003*, 2003. San Francisco, Calif., Decembre 1-5.

[19] L. Chen, J. Xu, I. Djurdjevic et S. Lin, "Near Shannon Limit Quasi-Cyclic Low Density Parity-Check Codes", *IEEE Trans. Commun.*, vol. 52(7), pp. 1038 – 1042, 2004.

[20] H. Zhong et T. Zhang, "Block-LDPC : A Practical LDPC Coding System Design Approach", *IEEE Transactions on Circuits and Systems*, vol. 52(4), pp. 766–775, 2005.

[21] S. Myung, K. Yang et J. Kim, "Low density parity-check codes", *IEEE Trans. Inform. Theory*, vol. 51, pp. 2894 – 2901, 2005.

[22] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson et R. Urbanke, "Finite length analysis of low-density parity-check codes on the binary erasure channel", *IEEE Trans. Inform. Theory*, vol. 48(6), pp. 1570–1579, 2002.

[23] D. J. MacKay, "Good error-correcting codes based on very sparse matrices", *IEEE Trans. Inform. Theory*, vol. 45(2), pp. 399–431, 1999.

[24] T. Richardson et R. Urbanke, "The capacity of low-density-parity-check codes under message-passing decoding", *IEEE Trans. Inform. Theory*, vol. 47(2), pp. 619–637, 2001.

[25] J.-Y. Chouinard, *Théorie et pratique des codes correcteurs.* Université Laval, 2005.

[26] H. Wymeersch, H. Steendam et M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)", *IEEE Communications Society*, vol. 45(2), pp. 399–431, 2004.

[27] T. K. Moon, *Error Correction Coding - Mathematical Methods and Algorithms.* Wiley, 2005.

[28] S. Chung, G. Forney, T. J. Richardson et R. Urbanke, "On The Design of Low Density Parity Check Codes within 0.0045 dB of the Shannon Limit", *IEEE Commun. Lett.*, vol. 5 (2), pp. 58–60, 2001.

[29] Z. He, P. Fortier et S. Roy, "A Class of Irregular LDPC Codes with Low Error Floor and Low Error Encoding Complexity", *IEEE Communications Letters*, vol. 10(5), 2006.

[30] Z. He, S. Roy et P. Fortier, "Capacity-approaching LDPC codes with Low Error Floors for High-speed Digital Communications", in *23rd Biennal Symposium on Communications, Ontario, Canada*, pp. 10–13, 2006.

[31] J.-B. Doré, M.-H. Hamon et P. Pénard, "Cycle and Distance Properties of Structured LDPC Codes Based on Circulant Pemutation Matrices", in *CWIT '07. 10th Canadian Workshop on Information Theory*, juin 2007.

[32] J. Thorpe, "Low-Density Parity-Check (LDPC)Codes constructed from protograph", *IPN Progr.Rep.*, pp. 42–154, 2003.

[33] S. Kim, J.-S. No, H. Chung et D.-J. Shin, "Quasi-cyclic Low-Density Parity-Check Codes With Girth Larger Than 12", *IEEE Trans. Inform. Theory*, vol. 53(8), pp. 2885–2891, 2007.

[34] J. Kang, P. Fan et Z. Cao, "Flexible construction of Irregular Partitioned Permutation LDPC codes with Low Error Floors", *IEEE Commun. Lett.*, vol. 9(6), pp. 534–536, 2005.

[35] S. Kim, J.-S. No, H. Chung et Z. Cao, "On the girth of Tanner's (3,5) quasi-cyclic LDPC codes", *IEEE Trans. Inform. Theory*, vol. 52(4), pp. 1739–1744, 2006.

[36] R. M. Tanner, D. Sridhara et T. Fuja, "A Class of Group-Structured LDPC Codes", in *Int. Symp. Commun. Theory Appl. (ISCTA)*, juillet 2001. Ambleside, U.K.

[37] R. Smarandache et P. Vontobel, "On Regular Quasi-Cyclic LDPC Codes from Binomials", *IEEE International Symposium on Information Theory*, p. 274.

[38] W. Zhan, G. Zhu, L. Peng et X. Yan, "Capacity-approaching LDPC codes with Low Error Floors for High-speed Digital Communications", in *2007 International Symposium on Intelligent Signal Processing and Communication Systems*, pp. 12–15, Dec. 2007. Xiamen, China.

[39] L. Peng, G. Zhu et X. Liu, "The Low-Density Parity Check Codes Based on The n-Queen Problem", in *ACM Conference NRBC*, pp. 37–41, Oct. 2004. New-York, NY, USA.

[40] T. Richardson, "Error Floors of LDPC Codes", *41st Annual Alerton Conf. on Communications, Control and Computing, Monticello, IL, USA*, pp. 1426–1435, 2003.

[41] C. Cole, S. Wilson, E. Hall et T. Giallorenzi, "Analysis and Design of Moderate Length Regular LDPC Codes with Low Error Floors", in *40th, Conference on Information Sciences and Systems, Princeton, NJ.*, Mai 2006.

[42] C. Cole, S. Wilson, E. Hall et T. Giallorenzi, "A General Method for Finding Low Error Rates of LDPC Codes", in *eprint arXiv :cs/0605051, 05/2006*, Juin 2006.

[43] S. Laendner et O. Milenkovic, "Algorithmic and Combinatorial Analysis of Trapping Sets in Structured LDPC Codes", in *International Conference on Wireless Networks, Communications and Mobile Computing, Hawaii*, pp. 630–635, 2005.

[44] B. Vasic, "Low-density Parity Check Codes With Low Error-Floors", in *available online at http ://www.ece.arizona.edu/ vasic/Projects.html*, 2006.

[45] C. Cole, "Error Floor Analysis for an Ensemble of Easily Implementable Irregular (2048,1024) LDPC Codes", in *Military Communications Conference, IEEE* , pp. 1–5, Nov. 2008. San Diego, CA.

[46] D. J. MacKay, *Information Theory, Inference, and Learning Algorithms.* CAMBRIDGE, 2003.

[47] R. Smarandache et P. Vontobel, "Pseudo-Codeword Analysis of Tanner Graphs from Projective and Euclidean Planes", *IEEE Transactions on Information Theory*, vol. 53(7), pp. 2376 – 2393, juillet 2007.

[48] S. Xia et F. Fu, "Minimum Pseudo-Weight and Minimum Pseudo-Codewords of LDPC Codes", in *available online at http ://arxiv.org/PS_cache/cs/pdf/0606/0606051*, Juin 2006.

[49] A. Orlitsky, R. Urbanke, K. Vishwanathan et J. Zhang, "Stopping sets and the girth of Tanner graphs ", in *IEEE Int. Symp. Inform. Theory, Lausanne, Switzerland*, p. 2, Juin-Julliet 2002.

[50] C. Kelley, D. Sridhara, J. Xu et J. Rosenthal, "Pseudocodeword weights and stopping sets", in *IEEE Int. Symp. Inform. Theory, Chicago, USA*, p. 68, Juin-Julliet 2004.

[51] R. Koetter et P. Vontobel, "Graph covers and iterative decoding of finite-length codes ", in *3rd Int. Conf. Turbo Codes and Related Topics, Brest France*, pp. 75–82, Sep. 2003.

[52] I. Adjudeanu, J.-Y. Chouinard et P. Fortier, "On the correlation between error weights and syndrome weights for belief propagation decoding of LDPC codes", in *Proc. 11th Canadian Workshop on Information Theory*, 2009. Ottawa, Canada, Mai 13-15.