Université de Limoges
École doctorale Science-Technologie-Santé
Faculté des Sciences et Techniques
XLIM-DMI Équipe PI2C

Thèse N° 29-2009

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Spécialité : **Mathématiques et Applications**

présentée et soutenue par

Christophe CHABOT

le 24 Septembre 2009, à 14h

Reconnaissance de codes, structure des codes quasi-cycliques

Thèse dirigée par **Thierry BERGER et Nicolas SENDRIER**

Jury :

Rapporteurs :		
	Pierre LOIDREAU	Ingénieur de l'Armement HDR Université Rennes 1
	Gilles ZEMOR	Professeur à l'Université de Bordeaux
Examinateurs :		
	Thierry BERGER	Professeur à l'Université de Limoges
	Pascale CHARPIN	Directrice de Recherche à l'INRIA Rocquencourt
	Philippe GABORIT	Professeur à l'Université de Limoges
	Nicolas SENDRIER	Directeur de Recherche à l'INRIA Rocquencourt

Remerciements

Ces quelques pages furent certainement pour moi les plus dures à écrire mais les personnes citées (et celle que j'aurais oubliées par mégarde) en valent la peine.

Je voudrais bien sûr remercier tout d'abord mes directeurs de thèse qui m'ont permis d'atteindre mon rêve de quelques années maintenant. Thierry Berger, qui m'a fait l'honneur de m'accueillir à Limoges et qui m'a fait tout de suite confiance. Cette confiance a été de suite réciproque. Je le remercie pour sa présence et ses conseils précieux.

Nicolas Sendrier, qui m'a accueilli au feu Projet CODES où j'ai passé un stage de découverte de la recherche très enrichissant et confortant mon envie de travailler dans ce domaine en vogue qu'est la Codecryptographie.

Je souhaite les remercier pour leur disponibilité, leur soutien et l'indépendance qu'ils m'ont laissé dans mes travaux. Les gens me cotoyant un minimum savent comme ceci est important pour moi.

Les remerciements suivants vont bien entendu à mes deux rapporteurs de thèse Pierre Loidreau et Gilles Zémor qui m'ont fait l'honneur de prendre du temps sur leurs vacances pour relire le manuscrit et me faire des remarques pertinentes ouvrant des perspectives.

Finalement, Pascale Charpin et Philippe Gaborit m'ont fait le plaisir de bien vouloir compléter le jury pour finalament constituer un jury que beaucoup m'envieraient :)

Avant de continuer l'énumération, je tiens à signaler que l'ordre d'apparition des personnages est tout à fait pseudo-aléatoire.

Lors de mon séjour de 3 ans dans les bureaux du DMI, j'ai pu cotoyer de nombreux gens sympathiques. Je tiens à remercier Thierry, Matthias, Abdelkader, François, Olivier, Philippe, Carlos, ... pour leur aide. Une mention particulière revient aux secrétaires qui ont été toujours disponibles : Sylvie, Patricia, Yolande, Christelle, Nadia, Aurélie et Marie-Paule. Je tiens à saluer tous les doctorants que j'ai pu croiser. Je commencerai tout naturellement par le bureau du bas avec Nicolas, Hassan, Julien et Aurore. Salut à Sandrine, Samuel, Daouda, Delphine, Elsa, Carole, Ainhoa, Guilhem, Pierre-Louis et Benjamin. Merci pous tous les bons moments passés ensemble entre autres à la pause café.

Je remercie toutes les personnes que j'ai cotoyées lors de mon stage de Master 2 et des visites ultérieures au Projet CODES : Nicolas, Anne, Pascale, Jean-Pierre, Daniel, Mathieu, Matthieu, Christelle, Céline, Benoit, Maria, Andréa, Yann, Scarab, Maxime, ...

Je tiens aussi à remercier les Rennais qui m'ont accueilli chaleureusement à plusieurs reprises et qui sont maintenant mes collègues : Johann, Delphine, Felix et Pierre.

Évidemment, le plus grand merci revient à toute ma famille qui m'a supporté pendant 27 ans dans tous les sens du terme. Plus particulièrement, je remercie papa et maman sans qui je ne serais pas là ; j'ai toujours voulu placer cette phrase dans un discours de remerciements, je crois que le moment est venu. Je voudrais les remercier pour les sacrifices qu'ils ont fait pour que le petit dernier qui a appris à compter avec les chiffres et les lettres puisse s'épanouir dans les Mathématiques, cette si belle science. Je sais combien cela représente pour une famille de "paysans" d'avoir un docteur dans la famille que j'ai du mal à quantifier ma fierté d'y être arrivé. Je ne peux pas oublier ma grande soeurette Titine qui a toujours été présente. Fanny, Flavien et Jimmy, je n'oublierai jamais les courses de charrettes et les parties de Wallball ! Merci aussi à mon frère Didier qui m'a donné goût à l'informatique dès mon plus jeune âge grâce notamment à son YAMAHA MSX avec quelque chose comme 16Ko de RAM (les spécialistes comprendront !). Une pensée aussi à Chantal, Mathilde, Tonton Omer, Tata Nicole, Cousin Éric et cousine Karine.

Une petite pensée aux gens du village ravis d'avoir un docteur dans leur entourage pour soigner leur sciatique ...

Un grand merci à Nico et Justine pour leur soutien de longue date durant mes séjours dans la jungle parisienne.

Une pensée très particulière va tout naturellement à P'tite Doune et Shanwouimp, si vous m'entendez. Je n'oublierai jamais votre joie de vivre tellement communicative.

Bien sûr, big up à tous les gâtinais que je ne citerai pas un à un par peur d'en oublier. Vive la Gâtine libre !

Je tiens à saluer aussi mes amis de longue date rencontrés à Poitiers sur les bancs de la Fac : Dorothée et Julien, Sébastien, Karen et Nathanaël. Un grand merci à M. Reis, Mlle Di Poï et M. Quitté pour m'avoir donné goût aux Mathématiques supérieures.

Petit salut à Antoine x2, Chérif et Gossar pour ces parties de raquettes endiablées, très utiles pour se changer les idées.

Une grosse pensée à mes deux grandes aventurières, copine Céline et copine Hélène et merci pour ce grand n'importe quoi qui rythme nos soirées apéro-ciné ou encore on-mélange-des-ingrédients-qui-ont-l'air-bon-et-on-voit-ce-que-ça-donne ! Si je parle de vous,

je ne peux pas oublier les deux acolytes copain Xav et copain Manu.

Une grosse pensée à la bande des cheminots and co et à toutes ces soirées de folie. Merci à Chacha, Aurélie, Émilie, Ludo, Marlène, Ben, Yvan, Karine, Seb, Marlène, Mylène et tout particulièrement à Pierre pour sa présence cette dernière année.

Je ne peux en aucun cas oublier Momo pour son soutien de tous les moments et cette complicité rare.

Merci.

Table des matières

Introduction

Cette thèse regroupe les travaux effectués lors de ma thèse au sein du projet PI2C à l'Université de Limoges dans la continuité de mon stage de Master Recherche [Cha06] effectué au Projet CODES de l'INRIA Rocquencourt.

Le thème principal de cette thèse était la reconnaissance de codes. Ce sujet est abordé dans la première moitié de ce manuscrit et a fait l'objet de publications [Cha07, Cha09] et de résultats soumis [CCN09, CB09]. La problème de reconnaissance de codes est assez récent dans le domaine académique. En effet les premiers résultats publics sont arrivés avec la thèse d'*A. Valembois* [Val00] en 2000. Brièvement (ceci sera détaillé ultérieurement) le but de la reconnaissance de codes est de retrouver le code utilisé lors d'une transmission en connaissant seulement les données interceptées, c'est-à-dire à partir de mots de codes bruités. Dans le cas des codes en blocs linéaires aléatoires, ce problème a été montré NP-complet par *A. Valembois* dans sa thèse. Cependant il donne une méthode pour reconstruire de tels codes. Elle est basée sur l'utilisation d'un algorithme de recherche de mots de poids faibles dans un certain code (différent du code que l'on veut reconstruire), qui malheureusement est de très grande longueur. Ceci apporte des limites à cet algorithme et ainsi, on ne peut atteindre de grandes longueurs de codes et/ou des taux de bruit raisonnables. Mais cette limitation était attendue après la démonstration de la NP-complétude du problème de reconnaissance de codes. Ensuite, *M. Cluzeau* et *M. Finiasz*[Clu06b, Clu06a, Clu09] se sont intéressés à l'amélioration de cette technique. Ils ont proposé un meilleur test statistique pour distinguer une séquence aléatoire d'une séquence codée bruitée. Ils ont aussi adapté cet algorithme à la reconnaissance de codes LDPC en ajoutant des techniques de décodage itératif. Grâce à cela ils atteignent les bornes théoriques pour les LDPC concernant le nombre de mots nécessaires obtenues par *M. Cluzeau* et *J-P. Tillich* dans [Clu08]. Ceci montre que cette méthode est optimale (du point de vue de la longueur de séquence nécessaire) pour la reconstruction des codes LDPC. Ils ont ensuite appliqué cela à la détection de la longueur du code utilisé et à la synchronisation.

Ici, je m'intéresse à des problèmes moins généraux. Je suppose en effet connaître des informations sur le code et utilise ces données pour élaborer un algorithme dédié. Tout d'abord, le problème le plus simple auquel je m'intéresse est le problème de reconnaissance d'un code [Cha07]. Peut-on distinguer une séquence aléatoire d'une séquence bruitée codée

par un code donné ? Je présente ici une méthode pour résoudre ce problème. Elle est simplement constituée de calculs de produits scalaires entre des mots d'une base du dual et les mots bruités de la séquence. Un test statistique servant de distingueur est élaboré et naturellement, il comporte des probabilités de fausse alarme et de non-détection qui font partie des paramètres de l'algorithme. Ces paramètres déterminent entre autres la longueur de la séquence à considérer. Grâce à cette méthode, nous sommes maintenant en possession d'une brique de base qui nous servira par la suite.

Ensuite, je me suis intéressé à un problème un peu plus général : la reconstruction de codes appartenant à une famille [Cha09]. Supposons maintenant que le code utilisé appartienne à une famille connue (typiquement des codes linéaires en blocs sur un même corps et de même longueur). Le principe général formulé ici est de faire une recherche exhaustive sur les codes maximaux de cette famille (en utilisant l'algorithme de reconnaissance d'un code) au lieu de faire une recherche exhaustive sur tous les éléments de la famille. Cette technique est optimale si on considère la famille des codes cycliques d'une certaine longueur. En effet, le nombre de codes maximaux est logarithmique en le nombre d'éléments de la famille. Et ainsi, revient à une recherche exhaustive sur une "base" plutôt que sur la famille entière. De cette façon, on obtient en pratique des temps de calcul bien meilleurs que ceux obtenus avec l'algorithme de reconstruction de codes aléatoires.

En ce qui concerne les codes convolutifs, les algorithmes de reconstruction de tels codes utilisent tous le fait que les paramètres de l'encodeur sont supposés connus. En pratique, on effectue une recherche exhaustive sur ces paramètres. Je me suis intéressé à la détection de ces paramètres [CB09], c'est-à-dire retrouver ces paramètres à partir d'une séquence codée bruitée sans passer par un algorithme de reconstruction. Cette étude est faite dans le but d'éviter cette recherche exhaustive sur les paramètres et ainsi accélérer la reconstruction. Pour cela, on utilise la transformée de *Burrows-Wheeler* utilisée dans l'algorithme de compression *bzip2*. Elle consiste en un réarrangement de la séquence. Et dans le cas d'une séquence codée par un code convolutif, elle fait apparaître des suites de mêmes bits (runs). On applique ensuite un test statistique sur le nombre de runs, afin de distinguer une telle suite codée d'une suite aléatoire. De plus elle permet de distinguer des suites codées par des codes de paramètres différents. Ainsi, on obtient un algorithme qui, à partir d'une séquence codée bruitée, permet de retrouver les paramètres du codeur (nombre d'entrées, de sorties, et de cases mémoire) ainsi que la probabilité d'erreur du canal. Pour des taux d'erreur du canal inférieur à 10^{-2}, la détection préalable des paramètres permet d'améliorer l'algorithme de reconstruction.

Dans la deuxième partie de ce manuscrit, je me suis intéressé à la structure des codes quasi-cycliques. Ce genre d'étude peut être intéressant dans plusieurs domaines. En effet, une meilleure connaissance de la structure de telles familles peut aider à élaborer des algorithmes de reconnaissance de codes dédiés (c'est le cas pour les codes cycliques), à exhiber de nouvelles familles de codes avec de bons paramètres ou encore mieux comprendre l'u-

tilisation des codes quasi-cycliques dans le cryptosystème de *McEliece*.

Tout d'abord, nous nous sommes intéressés à la construction d'une sous-famille de codes quasi-cycliques annulés par des polynômes à coefficients matriciels [CCN09]. La motivation de cette étude était de généraliser les résultats bien connus pour les suites récurrentes linéaires et les codes cycliques. Concernant les suites récurrentes linéaires, nous avons généralisé les résultats principaux tels que la rationalité des suites récurrentes linéaires aux suites récurrentes linéaires à coefficients matriciels. Ensuite, nous avons regardé les codes quasi-cycliques comme des codes cycliques sur l'anneau $\mathbb{F}_q^\ell$. Nous avons pu ainsi généraliser des résultats issus des codes cycliques tels que la formulation du dual d'un code. Ici, le dual d'un code annulé par un polynôme est encore un code annulé par un polynôme que nous déterminons. Ainsi, il nous a été possible de construire des codes quasi-cycliques auto-duaux annulés par un polynôme à coefficients matriciels. Parmi ces codes auto-duaux, certains atteignent les meilleures bornes connues pour les distances minimales.

Ensuite, toujours pour essayer de généraliser les résultats obtenus pour les codes cycliques, je me suis intéressé aux permutations laissant invariant la quasi-cyclicité. *W.C. Huffmann*, *V. Job* et *V. Pless* ont déterminé dans [HJP93] l'ensemble de ces permutations dans le cas où la longueur du code vaut pr ou p^2 pour p et r des nombres premiers distincts. En effet, il est difficile de tous les déterminer dans le cas général. Cependant, ils déterminent le normalisateur de T (le shift) dans $\mathscr{S}_n$ qui est toujours constitué de telles permutations. Ici, je détermine le normalisateur de T^ℓ (le quasi-shift) dans $\mathscr{S}_{m\ell}$. De même, ces permutations sont toujours des permutations conservant la quasi-cyclicité.

Chapitre 1

Notions de théorie des codes

Rappelons tout d'abord quelques notions essentielles et utiles pour la suite sur la théorie des codes correcteurs d'erreurs. Nous détaillerons dans un premier temps les bases ainsi que certaines structures particulières de codes en blocs linéaires. Ensuite, nous nous intéresserons à une classe de codes "à flot", les codes convolutifs. Ces rappels sont tirés des ouvrages classiques [HP03, MS77, PH88].

1.1 Codes en blocs linéaires

1.1.1 Définitions et propriétés

Considérons le corps de base $\mathbb{F}_q$ où $q = p^m$ est une puissance d'un nombre premier.

Définition 1 *Soient $k, n \in \mathbb{N}^*$ avec $k \leq n$.*
Un code en bloc linéaire de longueur n et de dimension k *est un sous-espace vectoriel de dimension k de $\mathbb{F}_q^n$.*

Dans la suite, par abus de langage et quand il n'y aura pas d'ambiguïté possible, nous parlerons de *code* au lieu de *code en bloc linéaire*. Un tel objet sera alors appelé un $[n,k]_q$-*code* ou encore un $[n,k]$-*code sur* $\mathbb{F}_q$. Les éléments d'un code sont appelés les *mots* de ce code.

Définition 2 *Soit $x = (x_1, \ldots, x_n)$ un vecteur de l'espace ambiant $\mathbb{F}_q^n$. On définit le support de ce vecteur par*

$$supp(x) := \{i \in \{1, \ldots, n\} |\ x_i \neq 0\}$$

Nous allons maintenant munir l'ensemble des vecteurs de $\mathbb{F}_q^n$ d'une *distance*. A cet effet, définissons le *poids* d'un mot.

Définition 3 *Soit $x \in \mathbb{F}_q^n$ un vecteur de l'espace ambiant. On définit le* poids de Hamming *de ce mot par la valeur*

$$w_H(x) = \#supp(x)$$

Le poids de Hamming *de x est le nombre de coordonnées non nulles du vecteur x.*

Définition 4 *Soient x et y des vecteurs de l'espace ambiant $\mathbb{F}_q^n$. La* distance de Hamming *entre ces deux vecteurs est donnée par*

$$d_H(x, y) = w_H(x - y)$$

La distance de Hamming *entre x et y est le nombre de coordonnées différentes entre x et y.*
Si C est un code sur $\mathbb{F}_q$, on définit la distance de x au code C de la façon suivante

$$d_H(x, C) := \min_{c \in C} d_H(x, c)$$

La *Distance de Hamming* est bien une distance au sens usuel.

Remarquons que le poids de x est en fait la distance entre x et le vecteur nul ($w_H(x) = d_H(x, 0)$).

Définition 5 *Soit C un $[n, k]$-code sur $\mathbb{F}_q$. On définit la* distance minimale *de C par*

$$d_{min}(C) := \min_{c, c' \in C,\ c \neq c'} d_H(c, c') = \min_{c \in C,\ c \neq 0} w_H(c)$$

Cette distance est habituellement notée d et ainsi, on parlera d'un $[n, k, d]$-code sur $\mathbb{F}_q$.

Proposition 1 *(Borne de* Singleton*)*
Soit C un code de paramètres $[n, k, d]$. Alors :

$$d \leq n - k + 1$$

Lorsque cette borne est atteinte, c'est-à-dire quand $d = n - k + 1$, on parle de *code parfait* ou *MDS* (Maximum Distance Separable).

Exemple 1 *Considérons le code de parité dont chaque mot $c = (c_1, \ldots, c_n)$ est tel que*

$$c_n = -\sum_{i=1}^{n-1} c_i.$$

Prenons comme exemple le code de parité binaire de longueur 4.

$$C := \{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$$

Il a pour dimension 3 *et a bien pour distance minimale* $2 = 4 - 3 + 1$.

La distance minimale d'un code est une quantité primordiale puisqu'elle caractérise la capacité de correction de code :

Proposition 2 *Soit C un $[n,k,d]$-code sur $\mathbb{F}_q$. Soit $t = \lfloor \frac{d-1}{2} \rfloor$. Alors, pour tout $x \in \mathbb{F}_q^n$ tel que $d_H(x,C) \leq t$, il existe un unique élément c de C tel que $d_H(x,c) = d_H(x,C)$.*

Soient C un code de longueur n sur $\mathbb{F}_q$, $c \in C$ un mot de code et $e \in \mathbb{F}_q^n$ un motif de bruit (c'est-à-dire un vecteur de poids faible). Notons $x = c \oplus e$. Le principe du décodage est le suivant :

A partir de x, peut-on retrouver c (ce qui est équivalent à retrouver e).

En théorie, il est possible de décoder de manière unique pour des motifs d'erreurs de poids inférieur à la capacité de correction t du code. Cependant, en pratique, on fait face à des problèmes de complexité en temps. Par exemple, dans le cas du décodage des codes aléatoires, les meilleures techniques connues à ce jour utilisent des algorithmes de recherche de mots de poids faible tels que les versions de *Stern*[Ste89], *Canteaut-Chabaud*[CC98] et *Bernstein*[BLP08]. La méthode est la suivante :

Soient C un code de distance minimale d, c un mot de ce code et e un motif d'erreur de poids inférieur à la capacité de correction $t = \lfloor \frac{d-1}{2} \rfloor$ de C. Considérons maintenant D le code obtenu en ajoutant $x = c \oplus e$ à une base de C ($D = C \oplus x$). Dans le code D, il y a tous les mots de code de C ainsi que tous les translatés de ces mots par e. En particulier le vecteur e appartient à D et il est le vecteur de plus petit poids de D. Ainsi un algorithme de recherche de mots de poids faible appliqué à D est susceptible de renvoyer exactement le vecteur e. Cependant cet algorithme reste exponentiel en la longueur n du code. En effet, dans [CC98], une approximation de la complexité de cette méthode dans le cas binaire est donnée par :

$$2^{naH(R)+b}$$

où $a = 5.511\ 10^{-2}$, $b = 12$, R est le rendement du code et H la fonction entropie

$$H(x) = -x\log_2(x) - (1-x)\log_2(1-x).$$

Ainsi, nous sommes amenés à étudier des familles particulières de codes telles que les codes BCH et les codes de Reed-Solomon par exemple. Pour de telles familles, des algorithmes de décodage dédiés utilisant la structure particulière de ces codes ont été exhibés. Ces spécifications permettent d'obtenir des complexités bien meilleures.

1.1.2 Matrice génératrice et matrice de parité

Soit C un $[n,k]$-code sur $\mathbb{F}_q$. Comme C est un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k, on peut le représenter par une de ses $\mathbb{F}_q$-base $(c_1, \ldots, c_k)$.

Définition 6 *Soit C un $[n,k]$-code sur $\mathbb{F}_q$. Soit $(c_1,\ldots,c_k)$ une base de C. Alors la matrice*

$$G := \begin{pmatrix} \boxed{c_1} \\ \boxed{c_2} \\ \vdots \\ \boxed{c_k} \end{pmatrix}$$

est appelée une matrice génératrice de C.

Propriété 1 *Soit G une matrice génératrice d'un code C. On dit qu'elle* génère *C, car en effet, tout mot de code $c \in C$ est obtenu par multiplication à gauche de G par un vecteur de $\mathbb{F}_q^k$.*

$$\textit{Pout tout } c \in C, \textit{il existe } m \in \mathbb{F}_q^k \textit{ tel que} \quad m.G = c$$

Du fait qu'un code est un sous-espace vectoriel, on peut effectuer n'importe quelle opération inversible sur les lignes de G, la matrice résultante sera toujours une matrice génératrice.

Propriété 2 *Soit G une matrice génératrice de C. Soit $S \in M_{k,k}(\mathbb{F}_q)$ une matrice inversible. Alors $G' = SG$ est toujours une matrice génératrice de C.*

Définition 7 *Soient C un $[n,k]$-code et G une matrice génératrice de C. On dit que G est sous forme systématique si elle est de la forme*

$$G = (I_k|R) \quad \textit{où } R \in M_{k,n-k}(\mathbb{F}_q).$$

Plus généralement, on peut toujours parler de forme systématique si la matrice identité I_k n'a pas pour support de colonnes l'ensemble $\{1,\ldots,k\}$ mais un ensemble I . Si on note J son complémentaire dans $\{1,\ldots,n\}$, on notera :

$$G = (I_k|R)_{I,J}\,.$$

Pour avoir l'unicité de la matrice, on garde l'ordre naturel sur les éléments de I et J.

Exemple 2 *Soit C le code linéaire binaire ayant pour matrice génératrice*

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

Alors $G = (I_k|R)_{I,J}$ avec $R = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$, $I = \{1,3,6\}$ et $J = \{2,4,5,7\}$.

Avec cette définition, chaque matrice génératrice G peut être mise sous forme systématique. Pour cela, il suffit de réaliser un pivot de Gauss sur un ensemble de colonnes linéairement indépendantes. Cela revient à multiplier G à gauche par l'inverse de la sous-matrice de G obtenue en ne gardant que ces colonnes.

Exemple 3 *Soit C le code linéaire binaire ayant pour matrice génératrice*

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

Si on prend $S = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, *alors* $G' := SG = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$ *est sous forme systématique.*

Un code étant un sous-espace vectoriel d'un certain $\mathbb{F}_q^n$, on peut s'intéresser à son espace dual.

Définition 8 *Soient $c = (c_1, \ldots, c_n)$ et $d = (d_1, \ldots, d_n)$ des vecteurs de $\mathbb{F}_q^n$. On appelle le* produit scalaire *entre c et d la quantité*

$$< c, d >:= \sum_{i=1}^{n} c_i d_i$$

Remarquons que le terme de produit scalaire est en fait un abus de langage (cette application bilinéaire n'est pas définie positive).

Définition 9 *Soit C un $[n,k]$-code sur $\mathbb{F}_q$. On définit son code dual noté $C^\perp$ par*

$$C^\perp := \{d \in \mathbb{F}_q^n |\ \textit{pour tout } c \in C, \quad < c, d >= 0\}$$

Ainsi, $C^\perp$ est un $[n, n-k]$-code sur $\mathbb{F}_q$.

De même manière que pour la matrice génératrice d'un code, on peut définir une matrice générant le code dual.

Définition 10 *Soit C un $[n,k]$-code sur $\mathbb{F}_q$. On appelle* matrice de parité *de C toute matrice génératrice H de son code dual $C^\perp$. On a donc*

$$\textit{pour tout } c \in C, \quad H.{}^t c = 0$$

Ainsi, si G est une matrice génératrice de C, on a

$$G.{}^t H = 0$$

Il est alors facile de construire une matrice de parité d'un code à partir d'un matrice génératrice de ce code.

Proposition 3 *Soient C un $[n,k]$-code et G une matrice génératrice de C sous forme systématique*

$$G = (I_k|R) \quad \text{où } R \in M_{k,n-k}(\mathbb{F}_q)$$

Alors la matrice $H = (-{}^tR|I_{n-k})$ est une matrice de parité de C.

Exemple 4 *Prenons pour exemple le code de* Hamming *de longueur* 7. *Les codes de* Hamming *forment une famille de codes linéaires binaires de longueur $2^m - 1$, de dimension $2^m - m - 1$ et de distance minimale* 3. *Ils permettent ainsi de corriger des erreurs de poids 1. Une matrice de parité de ces codes est obtenu en listant sur les colonnes tous les vecteurs binaires de longueur m non nuls.*

Dans notre exemple il s'agit d'un code C de paramètres $[7,4,3]$.

$$H = \left(\begin{array}{|cccc|}\hline 1&1&0&1\\ 1&0&1&1\\ 0&1&1&1\\ \hline\end{array}\begin{array}{ccc}1&0&0\\0&1&0\\0&0&1\end{array}\right)$$

Et ainsi C possède une matrice génératrice de la forme :

$$G = \left(\begin{array}{cccc|ccc}1&0&0&0&1&1&0\\0&1&0&0&1&0&1\\0&0&1&0&0&1&1\\0&0&0&1&1&1&1\end{array}\right)$$

Nous avons vu dans la Propriété 2 que toute action inversible sur les lignes ne changeait pas le code. Cependant, en général des transformations sur les colonnes donnent un code tout à fait différent. D'où la définition :

Définition 11 *Soit C un code de longueur n. On note $Aut(C)$ l'ensemble des permutations sur les colonnes laissant C invariant.*

$$Aut(C) := \{\sigma \in \mathscr{S}_n | \sigma(C) = C\}$$

$Aut(C)$ est appelé groupe d'automorphismes *de C.*

En général $Aut(C)$ est réduit à $\{Id\}$. En effet, un groupe d'automorphismes non trivial implique une structure dans le code. Intéressons-nous à des familles de codes ayant un groupe d'automorphismes non trivial et étudions leur structure.

1.1.3 Codes cycliques

Jusqu'à maintenant, un code était seulement un sous-espace vectoriel quelconque de $\mathbb{F}_q^n$. Cependant aucun algorithme efficace de décodage n'existe pour les codes aléatoires, alors des études ont été réalisées pour construire des familles de codes structurés, en espérant pouvoir trouver un algorithme de décodage dédié plus efficace.

Définition 12 *Soit*

$$\begin{array}{rccc} T: & \mathbb{F}_q^n & \to & \mathbb{F}_q^n \\ & (x_1, x_2, \ldots, x_n) & \mapsto & (x_n, x_1, \ldots, x_{n-1}) \end{array}$$

l'application décalage circulaire *aussi appelée* shift.
Soit C un code de longueur n sur $\mathbb{F}_q$. Par définition :

$$C \text{ est cyclique} \iff \text{Pour tout } c \in C, \quad T(c) \in C$$

En d'autres mots, C reste stable par l'action de la permutation sur les colonnes T ou $T \in Aut(C)$.

Représentation polynomiale

Par souci de facilité d'écriture et afin d'étudier les propriétés algébriques de ces codes, il est plus commode d'écrire les mots d'un code cyclique sous forme polynomiale grâce à l'identification suivante :

$$c = (c_0, c_1, \ldots, c_{n-1}) \longleftrightarrow c(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$$

Et ainsi l'action du décalage circulaire sur un mot revient à la multiplication par X modulo $X^n - 1$ sur le polynôme correspondant :

$$T(c) = (c_{n-1}, c_0, \ldots, c_{n-2}) \longleftrightarrow X.c(X) = c_{n-1} + c_0 X + \cdots + c_{n-2} X^{n-1} \bmod (X^n - 1)$$

Définition 13 *Soit R_n l'anneau quotient défini par*

$$R_n := \mathbb{F}_q[X]/\langle X^n - 1 \rangle$$

Ainsi, un décalage circulaire sur un mot de code c correspond à une multiplication par X de $c(X)$ dans cet anneau quotient R_n. Dorénavant, nous utiliserons librement les deux notations suivant le contexte.

Proposition 4 *Soit C un code de longueur n sur $\mathbb{F}_q$.*

$$C \text{ est cyclique} \iff C \text{ est un idéal de } R_n.$$

Exemple 5 *Soient $q = 2$ et $n = 7$. L'idéal $(X^3 + X + 1).\mathbb{F}_2[X]/\langle X^7 - 1 \rangle)$ est un code cyclique de longueur 7 sur $\mathbb{F}_q$.*

Propriété 3 *Soit C un code cyclique de longueur n sur $\mathbb{F}_q$ avec $pgcd(n,q)=1$. Alors :*
- *C est un idéal principal de R_n.*
- *Il existe un unique polynôme unitaire $g(X)$ de plus bas degré dans C.*
- *$C=\langle g(X)\rangle$.*
- *$g(X)$ divise X^n-1.*

Il vient alors que tous les codes cycliques d'une longueur n donnée sur $\mathbb{F}_q$ peuvent s'obtenir grâce à la factorisation de X^n-1 sur $\mathbb{F}_q$.

Exemple 6 *Les codes cycliques de longueur 7 sur $\mathbb{F}_q$ sont :*
- $\langle 1\rangle = R_n$.
- $\langle (X+1)\rangle$.
- $\langle (X^3+X+1)\rangle$.
- $\langle (X^3+X^2+1)\rangle$.
- $\langle (X+1)(X^3+X+1)\rangle$.
- $\langle (X+1)(X^3+X^2+1)\rangle$.
- $\langle (X^3+X+1)(X^3+X^2+1)\rangle$.
- $\langle 0\rangle = \{0\}$.

Matrice génératrice et matrice de parité

Proposition 5 *Soit $C=\langle g(X)\rangle$ un code cyclique de longueur n sur $\mathbb{F}_q$ (avec $g(X)$ divisant X^n-1). Alors C est un code de dimension $k=n-\deg(g)$.*

Remarque 1 *Tous les mots du code engendré par $g(X)$ s'écrivent sous la forme $(m_0+m_1X+\cdots+m_{n-\deg(g)-1}X^{n-\deg(g)-1})g(X)$.*

Grâce à cette notion d'être *engendré* par un polynôme, il est facile de construire une matrice génératrice d'un code cyclique.

Proposition 6 *Soit $C=\langle g(X)\rangle$ un code cyclique de longueur n sur $\mathbb{F}_q$ avec $g(X)=g_0+g_1X+\cdots g_{\deg(g)}X^{\deg(g)}$. Si de plus $g(X)$ divise X^n-1, on a*

$$G=\begin{pmatrix} g_0 & g_1 & \cdots & g_{\deg(g)} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{\deg(g)} & 0 & \cdots & 0 \\ & & \ddots & & & \ddots & & \\ 0 & 0 & 0 & \cdots & g_0 & g_1 & \cdots & g_{\deg(g)} \end{pmatrix}$$

est une matrice génératrice de C.

Exemple 7 *Une matrice génératrice du code cyclique sur $\mathbb{F}_2$ de longueur 7 engendré par X^3+X+1 est*

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

Précédemment, nous avons vu qu'il était facile de construire une matrice de parité d'un code à partir d'une de ses matrices génératrice. Dans le cas d'un code cyclique, il est facile de caractériser son dual. En effet, il s'agit toujours d'un code cyclique dont un générateur est facilement calculable.

Théorème 1 *Soit C un code cyclique de longueur n sur $\mathbb{F}_q$ engendré par $g(X)$ avec $X^n - 1 = g(X)h(X)$. Alors*

$$C^\perp = \langle h^*(X) \rangle$$

où $h^*(X) = \sum_{i=0}^{\deg(h)} h_i X^{\deg(h)-i}$ *si* $h(X) = \sum_{i=0}^{\deg(h)} h_i X^i$

Ainsi, la construction d'une matrice de parité est immédiate.

Exemple 8 *Reprenons $C = \langle X^3+X+1 \rangle$ le code cyclique de longueur 7 sur $\mathbb{F}_2$ de l'exemple précédent. On a la factorisation*

$$X^n - 1 = (X^3 + X + 1)(X^4 + X^2 + X + 1).$$

Ainsi, $C^\perp$ est un code cyclique engendré par $h^*(X) = X^4 + X^3 + X^2 + 1$. Et donc, une matrice de parité de C est donnée par

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

Codes BCH

Les codes BCH forment une sous-famille des codes cycliques qui ont été introduits indépendamment autour de 1960 par A.Hocquenghem et R.C.Bose, K.Ray-Chaudhuri. Ces codes cycliques particuliers sont intéressants puisqu'ils possèdent un algorithme de décodage utilisant la borne BCH. Ils sont décrits grâce aux racines du polynôme $g(X)$ engendrant le code.

Soient $n \in \mathbb{N}^*$ et $\mathbb{F}_q$ le corps de base. Soit m l'ordre de q modulo n. Les racines du polynôme $X^n - 1$ sont des éléments de F_{q^m}. De plus, elles sont des racines $n^{\text{ème}}$ de l'unité. En particulier, si on note α un élément primitif de $\mathbb{F}_{q^m}$ et $\beta = \alpha^{(q^m-1)/n}$, β sera une racine primitive $n^{\text{ème}}$ de l'unité et toutes les racines de $X^n - 1$ en seront des puissances.

$$X^n - 1 = \prod_{i=0}^{n-1} (X - \beta^i).$$

Considérons C un code cyclique de longueur n sur $\mathbb{F}_q$ engendré par $g(X)$ (de degré r) divisant $X^n - 1$. Le polynôme $g(X)$ aura pour racines $\beta^{i_1}, \ldots, \beta^{i_r}$.

Théorème 2 *Soit C le code cyclique de longueur n sur $\mathbb{F}_q$ engendré par $g(X)$. Si $g(X)$ a pour racines $\beta^b, \beta^{b+1}, \ldots, \beta^{b+\delta-2}$ ces $\delta - 1$ puissances consécutives de la racine primitive β, alors la distance minimale de C est au moins δ. Ce paramètre δ est appelé la* distance construite.

Théorème 3 *Pout tout entier n de la forme $q^m - 1$, $m \geq 3$ (q une puissance d'un premier), il existe un code cyclique t-correcteur C de paramètres $[n, k]$ tel que $k \geq n - 2tm$ si $q > 2$ ($k \geq n - tm$ si $q = 2$) dont le polynôme générateur est :*

$$g(X) = ppcm(m_1(X), m_2(X), \ldots, m_{2t-1}(X))$$

où $m_i(X)$ est le polynôme minimal de α^i, α étant un élément primitif de $\mathbb{F}_{q^m}$.

Définition 14

- *Un tel code est appelé un* code BCH.
- *Si de plus $b = 1$, on parle de* code BCH au sens strict.
- *Et si $n = q^m - 1$, on parle de* code BCH primitif.

En particulier, pour $n \in N^*$ et $q = p^r$ premiers entre-eux, et $\delta \in \{1, \ldots, n\}$, il est facile de construire un code BCH de longueur n sur $\mathbb{F}_q$ et de distance minimale au moins δ. Il est engendré par le polynôme

$$g(X) = ppcm(\mu_\beta(X), \mu_{\beta^2}(X), \ldots, \mu_{\beta^{\delta-1}}(X))$$

où $\mu_\gamma(X)$ désigne le polynôme minimal de γ à coefficients dans $\mathbb{F}_q$.

Exemple 9 *Soient $n = 15$, $q = 2$ et $\delta = 3$. L'ordre de $q = 2$ modulo $n = 15$ est $m = 4$. Soit β une racine primitive $15^{ème}$ de l'unité ($\beta \in \mathbb{F}_{q^m} = \mathbb{F}_{2^4}$ et $\beta^4 + \beta + 1 = 0$).*

Posons

$$\begin{aligned} g(X) &= ppcm(\mu_\beta(X), \mu_{\beta^2}(X), \mu_{\beta^3}(X)) \\ &= ppcm(X^4 + X + 1, X^4 + X + 1, X^4 + X^3 + X^2 + X + 1) \\ &= X^8 + X^7 + X^6 + X^4 + 1. \end{aligned}$$

Alors, le code cyclique engendré par $g(X)$ est une code BCH primitif au sens strict de longueur 15 sur $\mathbb{F}_2$ de distance minimale au moins 3.

Codes de Reed-Solomon

Nous allons maintenant considérer une famille de codes BCH primitifs avec $m = 1$.

Définition 15 *Soit $q = p^r$ une puissance d'un nombre premier. Un* code de Reed-Solomon *sur $\mathbb{F}_q$ de distance minimale d est un code BCH de longueur $n = q-1 = p^r-1$ et de distance construite d.*

La racine primitive $n^{\text{ème}}$ de l'unité α que l'on doit considérer pour construire le code BCH se trouve cette fois-ci dans $\mathbb{F}_q$. Ainsi les polynômes minimaux des puissances de ces racines sont de degré 1. Par conséquent le polynôme générateur $g(X) = (X - \alpha^b)(X - \alpha^{b+1}) \cdots (X - \alpha^{b+d-2})$ est de degré $d-1$ et le code est de dimension $k = n-(d-1) = n-d+1$. Un code de Reed-Solomon est donc MDS.

1.1.4 Codes quasi-cycliques

Les codes quasi-cycliques sont une généralisation des codes cycliques. En effet, dans le cas des codes cycliques on imposait la présence d'un *shift* dans le groupe d'automorphismes d'un code, ici, nous imposerons la présence d'un *quasi-shift*.

Définition 16 *Soit*

$$\begin{array}{rccc} T : & \mathbb{F}_q^n & \to & \mathbb{F}_q^n \\ & (x_1, x_2, \ldots, x_n) & \mapsto & (x_n, x_1, \ldots, x_{n-1}) \end{array}$$

le shift *défini précédemment.*
Soient C un code de longueur n sur $\mathbb{F}_q$ et $\ell \in \mathbb{N}^*$. Par définition :

$$C \text{ est } \ell\text{-quasi-cyclique} \iff \text{Pour tout } c \in C, \quad T^\ell(c) \in C.$$

En d'autres mots, C reste stable par la permutation sur les colonnes T^ℓ ou $T^\ell \in Aut(C)$.

La permutation T^ℓ est appelée *quasi-shift*.

Exemple 10 *Le code linéaire binaire ayant pour matrice génératrice*

$$G = \left(\begin{array}{cc|cc|cc} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{array}\right)$$

est 2-quasi-cyclique.

En fait, plus généralement on peut parler de code ℓ-quasi-cyclique dès que le groupe d'automorphismes du code contient une permutation ℓ-quasi-cyclique. Nous détaillerons cela dans la dernière partie.

1.1.5 Codes de Reed-Muller

Les Codes de *Reed-Muller* forment une famille de codes introduite dans les années 1950 par *D.E. Muller* en collaboration avec *I.S. Reed* qui en donna un algorithme de décodage. Depuis, ces codes ont été beaucoup étudiés en particulier pour leur représentation grâce à des fonctions booléennes.

Définition 17 *Soit m un entier naturel. Toute fonction*

$$\begin{array}{rccc} f: & \mathbb{F}_2^m & \longrightarrow & \mathbb{F}_2 \\ & (v_1,\ldots,v_m) & \longmapsto & f(v_1,\ldots,v_m) \end{array}$$

est appelée une fonction booléenne *à m variables.*

Définition 18 *Soit f une fonction booléenne à m variables. On note*

$$\mathbf{f} := (f(0,0,\ldots,0), f(1,0,\ldots,0), \ldots, f(1,1,\ldots,1))$$

le vecteur binaire de toutes les valeurs prises par f.

Exemple 11 *Soit $m = 2$. La fonction $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ à 2 variables définie par*

$$f(v_1, v_2) = v_1 v_2$$

est une fonction booléenne. Pour cette fonction booléenne,

$$\mathbf{f} = (f(0,0), f(1,0), f(0,1), f(1,1)) = (0,0,0,1).$$

Puisque $v_i^2 = v_i$, tout monôme de l'expression d'une fonction booléenne peut être écrit de manière unique comme produit de variables distinctes. On définit ainsi le *degré* d'une fonction booléenne comme étant le nombre maximal de variables distinctes apparaissant dans chaque monôme.

Définition 19 *Le* code de Reed-Muller *binaire d'ordre r en m variables $\mathscr{R}(r,m)$ de longueur $n = 2^m$, pour $0 \leq r \leq m$, est l'ensemble de tous les vecteurs $\mathbf{f}$ où $f(v_1,\ldots,v_m)$ est une fonction booléenne de degré au plus r.*

Exemple 12 *Soient $r = 1$ et $m = 2$. Le code de* Reed-Muller *binaire d'ordre 1 en 2 variables est obtenu de la façon suivante :*

f	$\mathbf{f}$			
0	0	0	0	0
1	1	1	1	1
v_1	0	1	0	1
v_2	0	0	1	1
$v_1 + v_2$	0	1	1	0
$1 + v_1$	1	0	1	0
$1 + v_2$	1	1	0	0
$1 + v_1 + v_2$	1	0	0	1

Proposition 7 *Le code de Reed-Muller binaire $\mathscr{R}(r,m)$ d'ordre r en m variables a pour*
- *longueur* $n = 2^m$,
- *dimension* $k = 1 + C_m^1 + \cdots C_m^r$,
- *distance minimale* $d = 2^{m-r}$.

1.2 Codes convolutifs

Les codes convolutifs ont été introduits par *P. Elias* [Eli55] dans les années 50. À cette époque-là, aucun algorithme de décodage efficace n'existait. Les premiers algorithmes de décodage efficaces apparurent au début des années 60 pour aboutir à l'algorithme de Viterbi [Vit67, Omu69, For73] en 1967. Cet algorithme repose sur la recherche de plus court chemin dans un treillis et permet d'atteindre des performances remarquables qui ont fait qu'ils sont devenus des standards dans les communications spatiales par exemple.

Un (n,k,m)-code convolutif est composé de k registres à décalage non rétroactifs comportant m cases mémoire. A chaque top d'horloge, pour chacune des n sorties, certains bits de l'état interne (dépendant de *polynômes générateurs*) sont additionnés (par un XOR). Cette somme donne le bit de sortie.

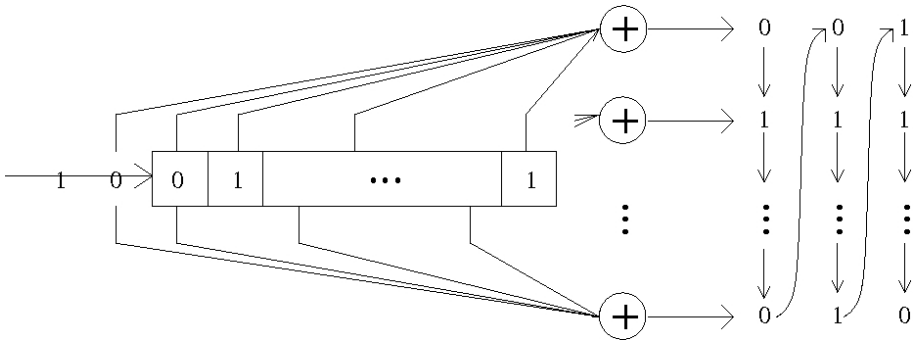


FIG. 1.1 – Code convolutif à 1 entrée et n sorties

La structure des codes convolutifs est riche et permet plusieurs approches. Dans un premier temps, nous détaillerons une approche polynomiale et ensuite une approche plutôt orientée algèbre linéaire.

1.2.1 Représentation polynomiale

Dans son chapitre du *Handbook of coding theory* [McE98], R.J. McEliece donne une formalisation très complète des codes convolutifs. Son approche algébrique polynomiale permet une étude poussée des propriétés de tels codes et est indispensable pour aborder les problèmes de reconstruction.

Définition 20 *Un (n,k)-code convolutif est une application linéaire qui associe à une suite $u(0), u(1), \ldots$ de mots de longueur k une suite $x(0), x(1), \ldots$ de mots de longueur n.*

À la différence des codes en blocs, un code convolutif possède une mémoire interne $s(i)$ de taille m et le $i^{\text{ème}}$ mot de code $x(i)$ ne dépend plus seulement du $i^{\text{ème}}$ mot d'entrée $u(i)$, mais aussi du $i^{\text{ème}}$ état interne $s(i)$.

On décrit ainsi un codeur convolutif de la façon suivante :

$$\begin{aligned} s(0) &= 0 \text{ et pour } i \geq 1, \\ s(i+1) &= s(i)A + u(i)B, \qquad (1.1) \\ x(i) &= s(i)C + u(i)D, \qquad (1.2) \end{aligned}$$

où les quatre matrices binaires A, B, C et D ont pour taille

$$\begin{aligned} A &: m \times m \\ B &: k \times m \\ C &: m \times n \\ D &: k \times n \end{aligned}$$

L'entier m est appelé *degré* du codeur.

Définition 21 *Le code convolutif associé aux matrices (A, B, C, D) est défini comme étant l'ensemble de toutes les séquences possibles $x(0), x(1), \ldots$ produites par les équations d'encodage (1.1) et (1.2).*

Définition 22 *À une suite $(a(i))_{i\geq 0}$, on associe sa* fonction génératrice

$$A(Z) = \sum_{i \geq 0} a(i) Z^i.$$

On peut ainsi définir l'encodage par un codeur convolutif de façon polynomiale :

Si on note $S(Z), X(Z)$ et $U(Z)$ les fonctions génératrices respectives de $(s_i)_i, (x_i)_i$ et $(u_i)_i$, on a alors les équation d'encodage

$$\begin{aligned} S(Z) &= U(Z)E(Z), \qquad (1.3) \\ X(Z) &= U(Z)G(Z), \qquad (1.4) \end{aligned}$$

où les matrices $E(Z)$ (qui est $k \times m$) et $G(Z)$ (qui est $k \times n$) sont données par :

$$\begin{aligned} E(Z) &= B(Z^{-1} I_m - A)^{-1}, \qquad (1.5) \\ G(Z) &= D + E(Z)C = D + B(Z^{-1} I_m - A)^{-1} C. \qquad (1.6) \end{aligned}$$

La matrice $G(Z)$ est appelée *matrice génératrice* du codeur convolutif.

1.2.2 Représentation binaire

Un code convolutif peut aussi être vu comme un endomorphisme de $\mathbb{F}_2^*$ l'espace vectoriel sur $\mathbb{F}_2$ des suites binaires. Un (n, k, m)-codeur est caractérisé par ses branchements sur les cases mémoire pour chaque sortie, c'est-à-dire les coefficients des polynômes générateurs. Notons $p_{1,0}, \ldots, p_{1,km-1}, \ldots, p_{n,0}, \ldots, p_{n,km-1}$ les coefficients de ces polynômes. Ainsi la matrice de cet endomorphisme vaut

$$M = \begin{pmatrix} p_{1,0} & \cdots & p_{1,k} & \cdots & p_{1,km-1} & 0 & 0 & \cdots \\ \vdots & & & & \vdots & 0 & 0 & \cdots \\ p_{n,0} & \cdots & p_{n,k} & \cdots & p_{n,km-1} & 0 & 0 & \cdots \\ 0 & 0 & p_{1,0} & \cdots & \cdots & p_{1,km-1} & 0 & \cdots \\ 0 & 0 & \vdots & & & \vdots & 0 & \cdots \\ 0 & 0 & p_{n,0} & \cdots & \cdots & p_{n,km-1} & 0 & \cdots \\ & & \ddots & & \ddots & & \ddots & \end{pmatrix}.$$

Si on note $E = (e_0, e_1, \ldots)$ la séquence binaire d'entrée et $S = (s_0, s_1, \ldots)$ la séquence de sortie, on a ainsi

$$M \begin{pmatrix} e_0 \\ e_1 \\ \vdots \end{pmatrix} = \begin{pmatrix} s_0 \\ s_1 \\ \vdots \end{pmatrix}.$$

Première partie

Reconnaissance de codes

Chapitre 2

Introduction

Dans cette partie, nous nous intéresserons à l'analyse d'un train binaire intercepté lors d'une transmission de données. Le but est dans l'idéal de reconstruire entièrement le système de transmission. Afin de réaliser cela, le problème est découpé en sous-problèmes (en considérant des sous-systèmes plus simples). En particulier, nous nous intéressons ici à la reconstruction du code correcteur utilisé lors de la transmission.

2.1 Transmission de données

Nous nous plaçons ici dans le contexte de transmission de données, ou plus précisément de transmission numérique de données. Par définition, une transmission numérique consiste à transférer de l'information sous forme de suite de symboles entre deux entités. Les études sur les transmissions numériques ont commencé avec les travaux de *C. Shannon* [Sha48] en 1948. Depuis, beaucoup d'études ont été réalisées, en particulier en théorie des codes et plus précisément en reconnaissance de codes.

Lors d'une transmission de données, un émetteur souhaite envoyer une certaine information à un récepteur. Cependant, si elle se fait à distance, ce premier ne peut pas envoyer l'information de manière brute pour des raisons physiques, de sécurité, d'intégrité. En effet, l'information doit transiter via un canal physique qui dans la plupart des cas est non sûr et sujet à des perturbations. Ainsi, l'émetteur doit appliquer des transformations à l'information telles qu'un système de chiffrement pour préserver la confidentialité, un code correcteur d'erreurs pour préserver l'intégrité, ou encore un brasseur, un entrelaceur, un modulateur.

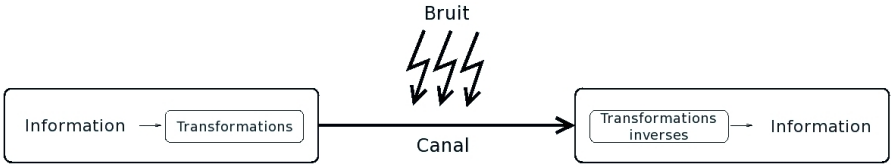


Fig. 2.1 – Schéma classique de transmission de données dans un canal

Bien entendu, toutes ces transformations doivent être inversibles afin que le récepteur puisse retrouver l'information initiale.

La transmission se faisant à distance, l'information transite par un canal physique sous forme par exemple d'un signal radio dans l'air ou encore d'une onde lumineuse dans une fibre optique. Considérons ici un canal dans le sens de la théorie de l'information. C'est-à-dire une boîte qui reçoit en entrée des éléments d'un alphabet $\mathscr{A}_1$ et qui renvoie des éléments d'un alphabet $\mathscr{A}_2$.

Ces deux alphabets peuvent être différents ; c'est le cas du canal à effacement pour lequel $\mathscr{A}_2 = \mathscr{A}_1 \cup \{\infty\}$ et où chaque élément $a \in \mathscr{A}_1$ est envoyé sur lui-même ou sur le caractère d'effacement ∞. Dans ce cas-là, il n'y a pas de notion d'*erreur* puisque chaque symbole reçu est identique au symbole envoyé ou alors un symbole d'effacement est reçu.

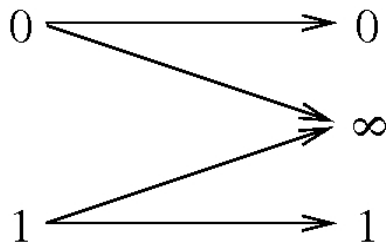


Fig. 2.2 – Canal binaire à effacement

Dans toute cette thèse, nous considérerons le canal binaire symétrique. Ici les deux alphabets sont identiques et réduits aux symboles binaires ($\mathscr{A}_1 = \mathscr{A}_2 = \{0, 1\}$). La symétrie provient du fait que la probabilité qu'un symbole change est la même pour 0 et 1. Elle s'appelle la *probabilité d'erreur du canal* et est notée p ($p < 1/2$). Ce canal est réaliste dans l'utilisation d'une modulation de phase à 2 états (BPSK) avec traitement numérique des données. Typiquement, lors de la modulation BPSK, le bit 0 est codé en 1 et le bit 1 en -1 et à la sortie du canal on reçoit des valeurs réelles plus ou moins proches de -1 et 1.

Pour retrouver des éléments de notre alphabet $\{0,1\}$, on décide lequel des éléments est le plus proche.

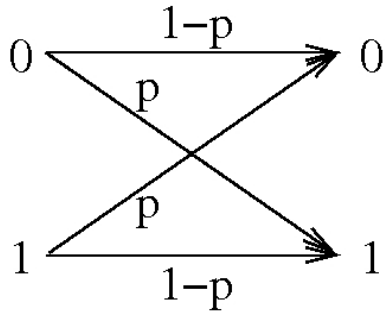


FIG. 2.3 – Canal binaire symétrique avec probabilité d'erreur p

Notations 1 *Nous notons $CBS(p)$ le canal binaire symétrique de probabilité d'erreur p.*

Ce modèle est pratique à étudier puisque la survenue d'une erreur revient à ajouter 1 modulo 2, ou à ajouter 1 dans le corps de Galois $\mathbb{F}_2$, ou encore à effectuer un OU EXCLUSIF $\oplus$ avec 1.

On peut facilement étendre la notion de canal symétrique à des entrées/sorties de taille supérieures. Soient q un entier et $\mathscr{A} = \{a_1, \ldots, a_q\}$ un alphabet à q éléments (typiquement il s'agit d'un corps à q éléments). Soit p la probabilité d'erreur du canal ($p < (q-1)/q$).

On appelle canal q-aire symétrique un canal avec :

- q entrées
- q sorties
- la probabilité de ne pas avoir d'erreur est de $1-p$
- la probabilité d'avoir une erreur est p
- toutes les erreurs sont équiprobablement réparties (chacune avec une probabilité $p/(q-1)$).

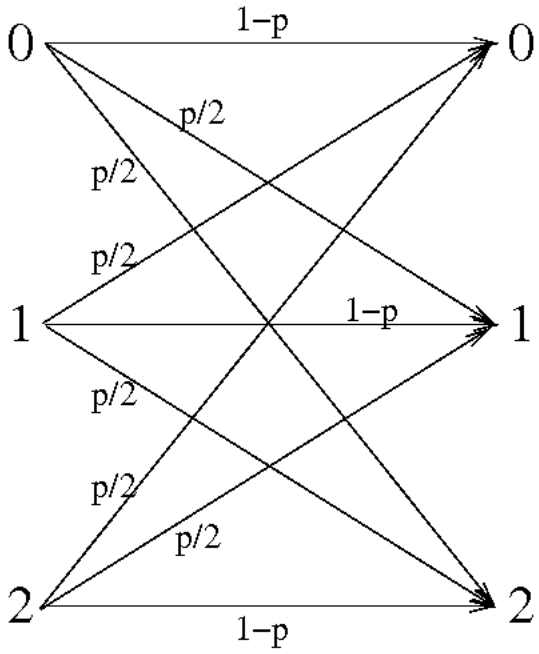


Fig. 2.4 – Canal 3-aire symétrique de probabilité d'erreur p

2.2 Problème de reconnaissance de codes

Notre but initial était de reconstruire le système de transmission en analysant le train binaire intercepté. Ici, nous nous intéressons à un sous-problème plus simple, le reconnaissance de codes. Nous supposons que la dernière transformation appliquée est un code correcteur d'erreurs ou alors que les couches supérieures ont déjà été reconstruites. En ce qui concerne les brasseurs, *M. Cluzeau* donne une méthode pour les reconstruire dans [Clu06b, Clu04, Clu07]. Cependant, il a besoin de connaître un biais statistique sur l'entrée ; ce qui est moral puisqu'un brasseur n'apporte aucune redondance à l'information.

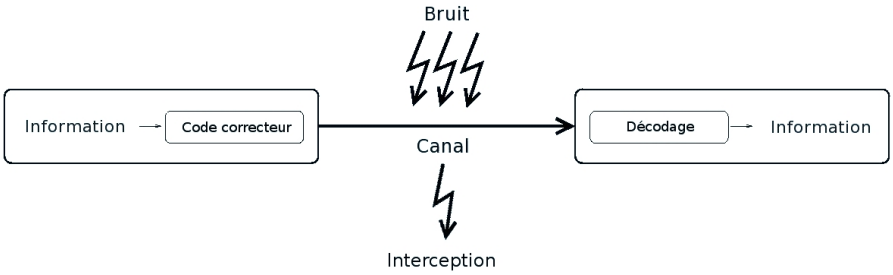


Fig. 2.5 – Schéma de transmission simplifié pour la reconnaissance de codes

Dans le cas de l'utilisation d'un code en blocs, le train binaire que nous avons en notre possession est composé de mots de code bruités. Le problème que l'on se pose est le suivant : étant donnée une séquence de mots d'un code C à la sortie d'un canal binaire symétrique, est-il possible de retrouver ce code C ? Formulé tel quel, ce problème a toujours une solution. En effet, tout motif d'erreur peut être apporté par un canal binaire symétrique de probabilité d'erreur non nulle, et ainsi, tout code conviendrait. *A. Valembois* s'est intéressé à ce problème dans sa thèse [Val00] et a établi le problème décisionnel suivant qu'il nomme *Réduction de rang* :

Soient N et n deux entiers positifs, et X une $N \times n$-matrice binaire.
Soit $k \leq rang(X)$ un entier positif.
Soit ω un entier.

DRR(X, k, ω) : Existe-t-il une $N \times n$-matrice binaire E vérifiant :

$$\begin{cases} rang(X + E) \leq k \\ w_H(E) \leq \omega \end{cases}$$

FIG. 2.6 – Problème décisionnel de Réduction de rang

Remarque 2 *Moralement, X représente la matrice des mots de code bruités interceptés (N mots de longueur n) et E la matrice de bruit. L'entier k représente la dimension du code que l'on souhaite reconstruire et ω le poids total de l'erreur.*

Quand ces conditions sont respectées, c'est-à-dire quand ce problème admet une réponse positive, on a donc que la matrice d'erreur E est de poids inférieur à ω et que la matrice $X+E$ est bien de rang inférieur à k puisque elle est composée uniquement de mots du code (le bruit a été enlevé).

Il a montré que ce problème décisionnel était dans la classe des problèmes NP-complet en utilisant la NP-complétude du problème décisionnel du calcul de la distance minimale d'un code montrée par *A. Vardy* [Var97].

Dans le cas d'un code convolutif, nous sommes en possession d'une séquence de sortie d'un codeur convolutif qui aura été bruitée par le canal.

Une notion importante apparaît dans ces deux cas : la différence entre code et codeur. En effet, si la reconstruction aboutit, on retrouve le *code* C qui a été utilisé, c'est-à-dire le sous-espace vectoriel dans le cas d'un code en blocs et un jeu de polynômes dans le cas d'un code convolutif. Cependant, cela ne nous donne aucun indice sur la manière dont a

été encodée l'information. Tandis que le codeur serait une application Enc_C de F^k dans F^n qui associerait à un message m un unique mot de code c.

Prenons le cas d'un code linéaire en bloc C de longueur n et de dimension k qui aurait été reconstruit. On en connaît donc une matrice génératrice G. Cependant, pour toute matrice inversible S de taille $k \times k$, SG est toujours une matrice génératrice de C. Maintenant, supposons que l'on ait un vecteur m de taille k et un mot de code c tels que $m.G = c$. Alors pour toute matrice inversible S de taille $k \times k$, $(mS^{-1}).SG = c$. Finalement, si on ne connaît que le code et rien sur la manière dont l'information est encodée, tout message m de taille k peut donner le mot de code c.

En conclusion, si on n'a aucune information sur la source, il est impossible (comme dans le cas des brasseurs) de retrouver le codeur à partir du code.

2.3 Étude statistique des séquences codées

Afin d'analyser un train binaire, une première étude possible est l'étude statistique de ces données. Cela revient à vérifier si la séquence présente des biais statistiques pour certaines notions telles que la fréquence des bits, l'entropie, ... Pour cela, nous avons appliqué la batterie de tests statistiques classiques du *NIST* [Nist01] (National Institute of Standards and Technology). Il s'agit d'une série de tests statistiques basiques permettant de vérifier le caractère aléatoire d'une séquence sortant d'un générateur pseudo-aléatoire. Chaque test renvoie une P_{value}, c'est-à-dire une valeur calculée à partir des données statistiques évaluant le caractère aléatoire de la séquence. Dans ces tests, si la P_{value} est au dessus d'un seuil (généralement 0.01), cela signifie que la séquence est considérée comme aléatoire. Cependant, en dehors de ce seuil, cette P_{value} est très peu exploitable.

Voici le principe de chacun des tests de cette batterie :

2.3.1 Tests du NIST

Test de fréquence

Le principe de ce test est de calculer la proportion de 0 et de 1 dans la séquence. Les proportions attendues pour une séquence aléatoire sont $1/2, 1/2$.

Test de fréquence par bloc

La séquence est découpée en blocs de taille M fixée. Le principe de ce test est de calculer la proportion de 1 dans ces blocs. La proportion attendue est $1/2$. Remarquons que si $M = 1$, on retombe sur le *Test de fréquence*.

Test des sommes cumulées

Grâce à ce test, nous étudions les excursions maximales partant de 0 définies par les sommes cumulées. Plus précisément, remplaçons les bits 0 (resp. 1) par la valeur -1 (resp. $+1$). On appelle *excursion* toute sous-séquence $(e_1, \ldots, e_m) \in \{-1, +1\}^m$ telle que

$$\sum_{i=1}^{m} e_i = 0 \text{ et } \sum_{i=1}^{k} e_i \neq 0 \text{ pour } k \in [\![1, m[\![$$

Le principe de ce test sera de chercher l'excursion maximale, c'est-à-dire la somme partielle la plus grande (en valeur absolue). Pour une séquence aléatoire, elle devrait être proche de 0.

Test des *Runs*

Le principe de ce test est de calculer le nombre total de *Runs*, c'est-à-dire les suites de même bit. Cela revient à calculer le nombre de fois où 2 bits consécutifs sont différents. Je parlerai plus en détail de ce test dans la section sur le *test de Burrows-Wheeler et Runs*.

Test des plus longs *Runs* dans un bloc

On considère une longueur de bloc M et on coupe la séquence en bloc de taille M. Ensuite, le test précédent est appliqué sur chaque bloc.

Test du rang

La séquence est découpée en blocs de taille M regroupés par groupes de Q pour former des matrices $Q \times M$. On calcule le rang de ces matrices et on compare les résultats obtenus avec une distribution référence χ^2

Test de la *transformée de Fourier* rapide

Dans ce test, on se focalise sur la hauteur des pics dans la *transformée de Fourier discrète* de la séquence. Ce test revient à un test d'autocorrélation : on cherche une corrélation entre la séquence et un décalé de cette même séquence. Pour cela, on s'intéresse au nombre de pics supérieurs au seuil de 95%. Une séquence non aléatoire sera détectée si ce nombre est significativement différent de 5%.

Test des motifs avec ou sans chevauchement

On considère une taille de fenêtre m. Le principe de ces tests est de rechercher les occurrences des motifs de taille m. Deux tests différents s'offrent à nous. Pour les deux

tests, la séquence est parcourue bit à bit jusqu'à apparition du motif. La différence réside dans la marche à suivre après découverte du motif :

Pour le premier, lorsqu'un motif est trouvé, on décale la fenêtre d'un bit vers la droite *(test des motifs avec chevauchement)*.

Pour le second, lorsqu'un motif est trouvé, on décale la fenêtre au premier bit suivant le motif trouvé *(test des motifs sans chevauchement)*.

Ensuite, les valeurs obtenues sont comparées à une distribution χ^2.

Ces tests servent à détecter des générateurs produisant de mêmes motifs de façon apériodique.

Test universel

La séquence est découpée en deux parties majeures : une première partie d'initialisation composée de $Q.L$ bits et une seconde partie de test composée de $K.L$ bits. Si la longueur de la séquence n'est pas un multiple de L, des bits finaux sont éliminés. Ensuite, les deux parties sont découpées en blocs de taille L.

On parcourt le segment d'initialisation et on note le numéro du bloc de la dernière occurrence de chaque motif de taille L. Si un motif n'apparaît pas, le numéro de bloc de sa dernière occurrence sera fixé à 0. Ensuite, on parcourt le segment de test par blocs de L bits. Pour chaque motif trouvé, on ajoute à la somme correspondante le $\log_2$ de la distance à la dernière occurrence.

Finalement, on compare les valeurs obtenues aux valeurs attendues si la séquence était aléatoire.

Test d'entropie

Ce test étudie la fréquence d'apparition de tous les motifs possibles de taille m dans la séquence. Plus précisément, il compare les fréquences pour des longueurs de bloc consécutives m et $m+1$.

Pour cela, il calcule l'entropie de l'ensemble des blocs de taille m (resp. de taille $m+1$) notée $\varphi^{(m)}$ (resp. $\varphi^{(m+1)}$). On considère finalement la valeur $ApEn(m) = \varphi^{(m)} - \varphi^{(m+1)}$.

Une valeur de $ApEn(m)$ proche de 0 signifiera une forte régularité et une grande valeur impliquera le caractère non aléatoire de la séquence.

Test d'excursion aléatoire

On considère toutes les excursions (cf. test des sommes cumulées) de la séquence. Pour chaque valeur x de somme partielle comprise entre -4 et $+4$, et pour chaque $k = 0, \ldots, 5$ on compte le nombre d'excursions dans lesquelles la valeur est apparue k fois, on notera cette valeur $v_k(x)$. Si une valeur x apparaît plus de 5 fois, on stocke sa fréquence dans $v_5(x)$.

On compare finalement ces valeurs avec les valeurs que l'on aurait obtenues avec une séquence aléatoire grâce à une distribution χ^2.

Test sériel

Comme pour le test d'entropie, ce test étudie la fréquence d'apparition de tous les motifs de taille m. Pour une taille m donnée, on calcule la fréquence d'apparition de tous les motifs de taille m (avec chevauchement). Si la séquence est aléatoire, la répartition est uniforme.

Test de compression *Lempel-Ziv*

Ce test est basé comme son nom l'indique sur l'algorithme de compression de *Lempel-Ziv*. Il cherche à déterminer si la séquence peut être compressée de manière significative par cet algorithme de compression.

L'algorithme de compression fonctionne comme suit :
On parcourt la séquence bit à bit et dès qu'un mot n'est pas dans le dictionnaire, on l'y ajoute.

Il suffit donc d'étudier le dictionnaire obtenu. Pour une séquence aléatoire, tous les mots de taille inférieure à la taille maximale devrait être représentés dans le dictionnaire. Pour ce test, on ne regarde que la taille du dictionnaire. Un dictionnaire de petite taille indique que la séquence est fortement compressible et donc qu'elle n'est pas aléatoire.

Test de complexité linéaire

Le but de ce test est de déterminer la taille du plus petit *LFSR* (Registre à décalage à rétroaction linéaire) générant cette séquence. Pour cela, ce test utilise l'algorithme de *Berlekamp-Massey*.

Plus la taille renvoyée par l'algorithme est petite, plus la séquence s'éloigne d'une séquence aléatoire.

Nous présentons ici les résultats de l'application de ces tests sur des séquences codées par des codes en blocs, des codes convolutifs et des turbo-codes avec ou sans bruit. Ces résultats sont généraux (ils sont appliqués à plusieurs codes d'une même famille) et ne présentent qu'une idée globale des résultats.

	Fréquence	Fréquence par bloc	Sommes cumulées	Runs	Plus longs Runs	Rang	Fourier rapide	Motifs	Universel	Entropie	Excursion	Sériel	Lempel-Ziv	Complexité linéaire
Sans erreur														
Codes en bloc														
Codes convolutifs					**x**	**x**		**x**					**x**	
Turbo codes					**x**			**x**					**x**	
Avec erreur 0.05														
Codes en bloc														
Codes convolutifs					**x**			**x**					**x**	
Turbo codes					**x**			**x**					**x**	
Avec erreur 0.1														
Codes en bloc														
Codes convolutifs													**x**	
Turbo codes													**x**	

Fig. 2.7 – Résultats des tests statistiques sur différentes séquences codées

Analyse des résultats :

Tout d'abord, pour les codes en blocs, même sans erreur, les séquences codées semblent aléatoires pour ces tests statistiques. Cependant cela n'est vrai qu'en général, puisque pour des codes de longueur relativement petite (quelques dizaines) certains tests réagissent et il est alors possible d'élaborer un test statistique dédié. Cela est dû fait que le nombre de mots d'un tel code est suffisamment petit pour qu'ils apparaissent tous dans une séquence. Nous utiliserons cela lors du test de distribution des poids.

Par contre, pour les codes convolutifs (et pour les turbo-codes), certains tests comme les runs, les motifs ou encore la compression semblent réagir. De plus, en présence de bruit ils réagissent toujours, mais si le bruit devient important, seule la compression donnent des résultats positifs. Moralement, cela est compréhensible, puisque comme dans le cas des codes en blocs de petite longueur, tous les mots de code sont susceptibles d'apparaître dans

une séquence. Finalement, la compression semble une voie intéressante pour reconnaître un code convolutif. Nous étudierons cela dans la section sur la détection des paramètres d'un code convolutif.

2.3.2 Test de distribution des poids

Un des buts de cette thèse était de construire des *distingueurs* pour les codes en blocs, ou bien de retrouver des *signatures* d'un code (un biais dans un test statistique permettant d'affirmer qu'un code avec certains paramètres a été utilisé) à partir de séquences codées bruitées. Mais au fur et à mesure des expérimentations, et avec confirmation des tests statistiques du NIST, pour des codes en blocs de longueur raisonnable, le nombre de mots de code est trop grand pour espérer trouver un biais statistique. En effet, les mots de code apparaissent être répartis uniformément dans l'espace $\mathbb{F}_2^n$. On pourrait s'attendre à ce que le fossé entre le mot nul et le(s) mot(s) de poids minimal d soit reconnaissable, mais le nombre de mots de poids faible dans un code est négligeable devant le nombre de mots de poids autour de $n/2$.

Cependant, pour des codes de longueur relativement petite (inférieure à 40), le nombre de mots étant raisonnable, il est possible de distinguer une séquence codée d'une séquence aléatoire. Pour cela, nous étudions la distribution des poids dans un tel code.

Principe

Supposons que la longueur n du code est connue (si ce n'est pas le cas, on fait une recherche exhaustive sur la longueur).

On découpe la séquence en mots de taille n et on calcule le poids de Hamming de ces mots. On obtient alors la répartition des poids de ces mots pour cette séquence. Il suffit alors de comparer cette répartition avec celle d'une séquence aléatoire, c'est-à-dire suivant la loi binomiale $\mathscr{B}(n, 1/2)$.

En théorie, ces deux répartitions sont différentes puisque pour un code en bloc, il y a la notion de distance minimale d. En effet, il n'y a pas de mots de code de poids strictement compris entre 0 et d. Ainsi, afin de pouvoir discriminer cela, nous serons amenés à considérer des séquences suffisamment grandes.

Implémentation

Dans cet algorithme, la complexité repose essentiellement sur le calcul du poids de Hamming des mots. Nous utilisons ici une astuce de *Cédric Lauradoux* permettant le calcul rapide du poids de Hamming d'un octet.

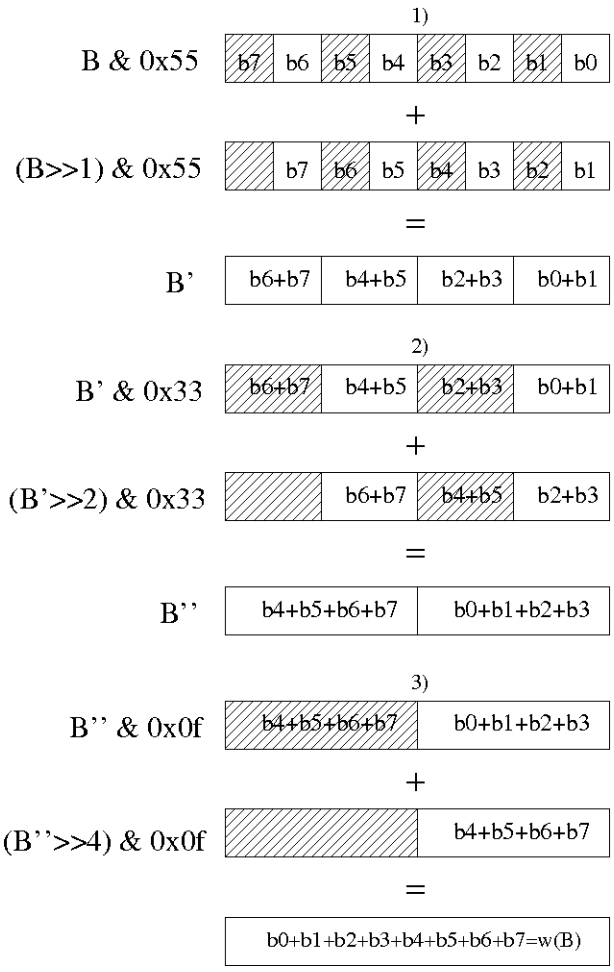


Fig. 2.8 – Calcul du poids de Hamming d'un octet

Un calcul naïf du poids de Hamming d'un octet coûterait 8 masques, 8 comparaisons et en moyenne 4 sommes alors qu'ici on n'a besoin que de 6 masques, 3 sommes et 3 décalages.

Ensuite pour la comparaison avec la répartition aléatoire, plusieurs choix sont possibles. Ici, nous utilisons la comparaison de la variance des poids (un test du χ^2 serait possible aussi). Ainsi pour chaque longueur testée, on obtient une P_{value} de la façon suivante :

$$P_{val} = \frac{Var(w_{obs}) - Var(w_{alea})}{Var(w_{alea})}$$

où w_{obs} est la répartition des poids observés et w_{alea} est la répartition des poids uniforme.

Ainsi, contrairement aux tests du NIST, plus cette valeur est proche de 0, plus la séquence est considérée comme aléatoire.

Résultats expérimentaux

Voici deux exemples de codes en blocs sur lesquels est appliquée notre méthode. Sont regroupées ici les courbes des P_{value} en fonction de la longueur testée et ceci pour différentes tailles de séquence. Pour l'instant considérons des séquences sans bruit.

• Un code en bloc de longueur 40 et de dimension 14 :

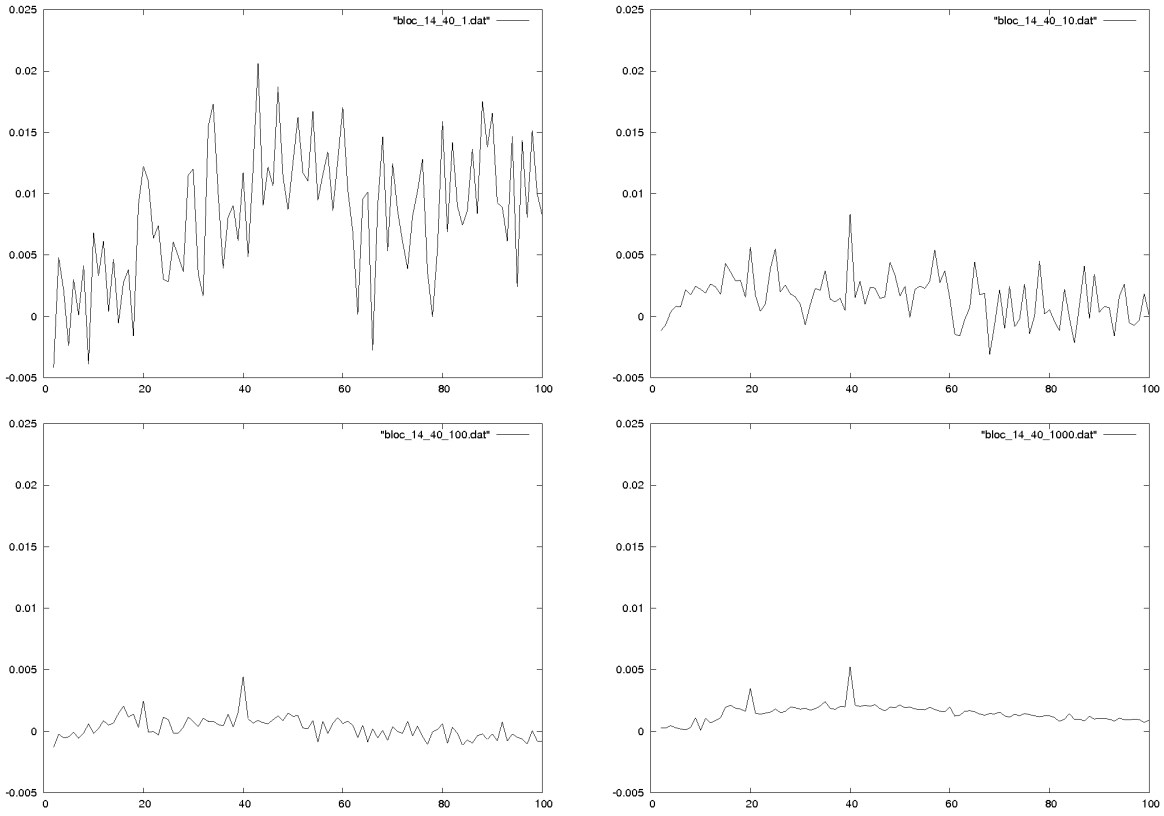


FIG. 2.9 – Test sur une séquence de 10^6, 10^7, 10^8 et 10^9 bits.

• Un code en bloc de longueur 20 et de dimension 10 :

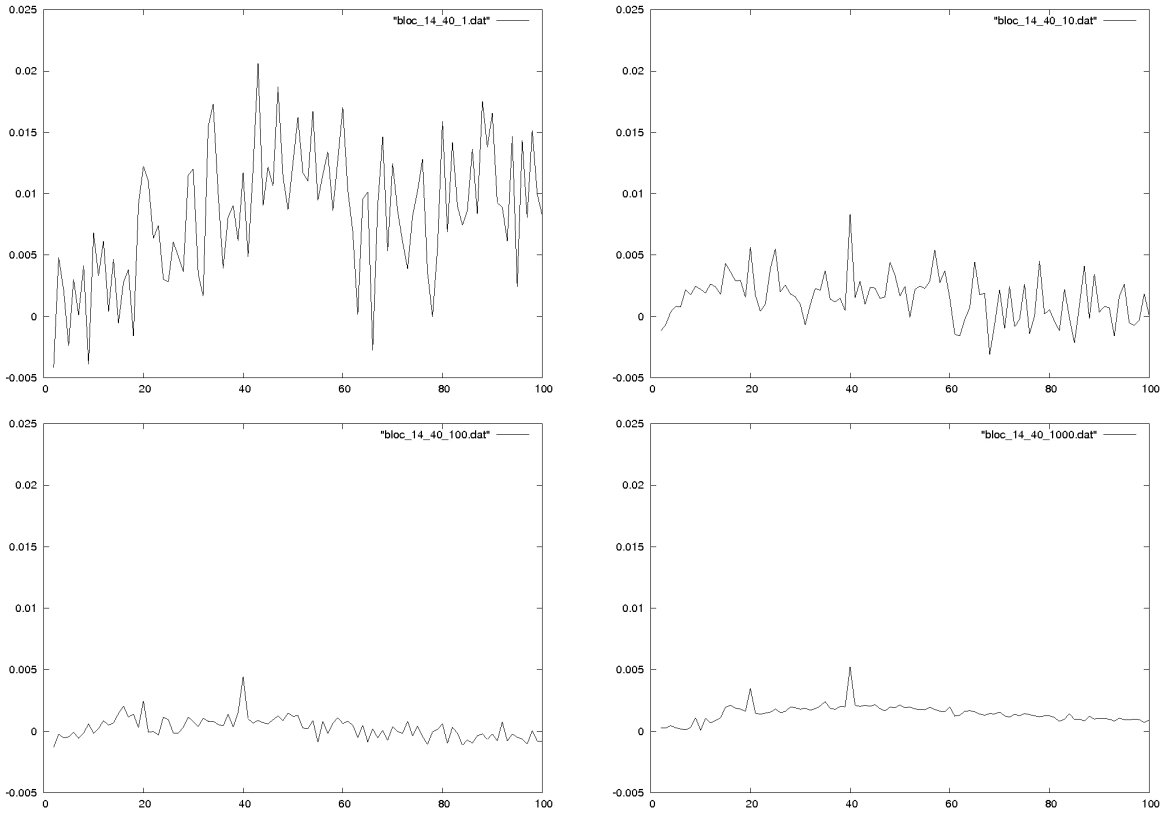


Fig. 2.10 – Test sur une séquence de 10^6, 10^7, 10^8 et 10^9 bits.

On remarque que dans ces graphiques, lorsque l'on augmente la taille de la séquence, un pic apparaît à la longueur réelle du code. Bien entendu, quand la dimension du code est plus petite, le pic apparaît plus tôt et est plus important. Remarquons tout de même que l'on ne peut atteindre que des paramètres relativement petits. Moralement, on s'attend à ce que la longueur de la séquence doive être supérieure à $n2^k$ pour un code de paramètres $[n, k]$ afin que tous les mots de code apparaissent. Cependant, en pratique, la séquence doit être bien plus grande pour commencer à voir apparaître le pic.

Donnons maintenant un exemple du comportement de cet algorithme face au bruit. Considérons à nouveau un code en bloc de longueur 20 et de dimension 10 et ajoutons du bruit petit à petit à notre séquence.

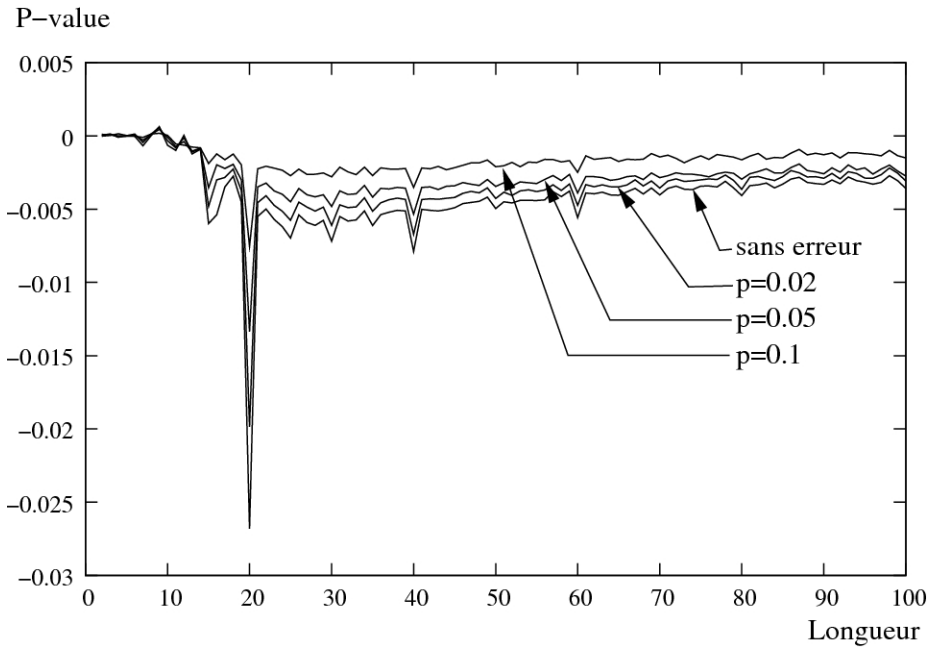


FIG. 2.11 – Test avec présence de bruit

Remarquons que le bruit ne perturbe pas énormément les résultats, il ne fait que lisser la courbe. Même avec un taux d'erreur de 0.1, le pic à la longueur 20 est encore visible.

On peut aussi noter que l'on retrouve des pics pour des longueurs multiples de 20. En effet, cette séquence pourrait provenir d'un code obtenu par concaténations de notre code initial de longueur 20.

Conclusion

Finalement, pour les codes en blocs, une étude purement statistique du train binaire ne permet pas d'atteindre des paramètres de codes réalistes. En effet le nombre de mots de code devient très rapidement trop élevé pour pouvoir espérer les parcourir. Dans la suite, nous utiliserons alors la structure de tels codes comme leur linéarité ou leur structure algébrique (codes cycliques) pour élaborer des algorithmes dédiés.

Chapitre 3

Codes en blocs

3.1 État de l'art

Ce chapitre a fait l'objet de deux publications [Cha07, Cha09]

De façon générale, le problème de reconnaissance de codes n'a pas beaucoup été étudié dans le domaine académique jusqu'aux années 2000 et les travaux de *A. Valembois* [Val00, Val01]. En effet, dans sa thèse, il formule pour la première fois le problème décisionnel de "Réduction de rang" (cf. Figure 2.6) qui formalise le problème de reconstruction d'un code linéaire en blocs. Il prouve que ce problème décisionnel est NP-complet et donne une technique pour reconstruire une matrice génératrice du code utilisé à partir d'une séquence codée interceptée bruitée. Nous présentons ici le principe de cette méthode.

Tout d'abord, au lieu de reconstruire directement une matrice génératrice, on reconstruit une matrice de parité (une matrice génératrice du code dual). Rappelons qu'il est facile de passer de l'une à l'autre (cf. Proposition 3). En effet, le but de la méthode est de retrouver des équations de parité, c'est-à-dire des équations annulant tous les mots de code. Cependant, ici, les mots de code reçus sont bruités, il faut donc retrouver des équations de parité probables. Regardons ce qu'il se passe dans le cas non bruité.

Soit $S = (s_i)_{1 \leq i \leq M}$ la suite interceptée découpée en mots de longueur n. Rangeons maintenant ces mots en colonnes dans une matrice L de taille $n \times M$:

$$L = \left(\begin{array}{|c|c|c|c|} \hline & & & \\ s_1 & s_2 & \cdots \quad \cdots & s_M \\ & & & \\ \hline \end{array} \right).$$

En absence de bruit :

Dans ce cas, les $s_i = c_i$ sont exactement des mots d'un code C de longueur n. Ainsi, une combinaison linéaire nulle des lignes de cette matrice donne exactement une équation de parité du code C.

$$(h_0, \ldots, h_{n-1}) \quad . \quad \left(\begin{array}{|c|c|c|c|} \hline c_1 & c_2 & \cdots & c_M \\ \hline \end{array} \right)$$

$$= \quad (\quad 0 \ , \ 0 \ , \ \ldots\ldots \ , \ 0 \quad)$$

Finalement, en absence de bruit, retrouver des équations de parité revient à chercher des combinaisons linéaires nulles des lignes de la matrice L.

En présence de bruit :

Dans ce cas, les $s_i = c_i \oplus e_i$ sont des mots de code bruités d'un code C de longueur n. Si on applique une équation de parité h du code C à de tels mots, on obtient :

$$\langle h, s_i \rangle = \langle h, c_i \rangle + \langle h, e_i \rangle = \langle h, e_i \rangle.$$

Mais comme nous le verrons plus tard dans la partie sur la reconnaissance d'un code, la probabilité que le produit scalaire $\langle h, e_i \rangle$ soit égal à 1 vaut

$$P(\langle h, e_i \rangle = 1) = \frac{1 - (1-2p)^{w_H(h)}}{2},$$

où p est la probabilité d'erreur du canal.

Ainsi, si on applique une équation de parité h de C à la matrice L, on obtient un mot de poids à peu près

$$\frac{M}{2} - \frac{M}{2}(1-2p)^{w_H(h)}.$$

Et cette valeur est loin de $\frac{M}{2}$ si la probabilité d'erreur du canal p n'est pas trop grande.

$$(h_0, \ldots, h_{n-1}) \quad . \quad \left(\begin{array}{|c|c|c|c|} \hline c_1 \oplus e_1 & c_2 \oplus e_2 & \cdots & c_N \oplus e_N \\ \hline \end{array} \right)$$

$$= \quad (\quad 0 \ , \ \mathbf{1} \ , \ \ldots\ldots \ , \ 0 \quad)$$

On est donc ramené à rechercher des mots de poids faible dans le code engendré par les lignes de la matrice L. Notons que ce code est très différent du code que l'on veut reconstruire. En effet, c'est un code de longueur M (qui est souvent très grande). En pratique, ce code est souvent de dimension n puisque la taille M de la séquence est grande devant n et ainsi le bruit perturbe suffisamment les mots de code pour rendre la matrice L de rang plein.

Pour la recherche de mots de poids faible, il est possible d'utiliser des algorithmes dédiés tels que ceux de *J.S. Leon* [Leo88], de *P.J. Lee* et *E.F. Brickell* [LB88], ou encore *J. Stern* [Ste89]. Ces algorithmes étaient à l'origine conçus pour attaquer le cryptosystème de *R.J. McEliece* [McE78], premier système de chiffrement à clef publique basé sur les codes. Cependant, *A. Valembois* et *M. Cluzeau* préconisent dans leurs thèses d'en utiliser la version la plus élaborée, celle de *A. Canteaut* et *F. Chabaud* [CC98]. Notons que récemment, *E.R. Bernstein*, *T. Lange* et *C. Peters* ont apporté quelques améliorations à cet algorithme dans [BLP08].

Cette méthode est ensuite reprise par *M. Cluzeau* [Clu06b], en particulier le test statistique permettant de construire le distingueur entre une séquence aléatoire et une séquence codée bruitée est affiné. Ensuite, dans [Clu06b, Clu06a], il donne une application aux codes LDPC (Low Density Parity Check) en utilisant des techniques de décodage itératif (adaptés au décodage de tels codes). Cet apport permet une amélioration significative de la complexité et permet d'atteindre la borne théorique du nombre de mots nécessaires dans la séquence pour reconstruire le code donnée par *M. Cluzeau* et *J.P. Tillich* dans [Clu08].

Toutes ces techniques supposent que le longueur du code et la synchronisation sont connues. Cependant, *M. Cluzeau* et *M. Finiasz* donnent dans [Clu09] une méthode pour les retrouver et ainsi éviter la recherche exhaustive sur ces paramètres.

Parallèlement, *J. Barbier* [BGH06, Bar07] donne une méthode de reconstruction des codes linéaires en bloc basée sur un algorithme de Gauss randomisé.

Dans cette thèse, nous ne nous intéressons pas au cas général de reconnaissance de codes, c'est-à-dire quand aucune information n'est connue sur le code. En effet, nous supposons connaître certaines informations sur le code utilisé et nous utilisons ces informations pour améliorer considérablement les performances des algorithmes. Nous développons aussi des algorithmes dédiés à ces cas particuliers que nous considérons.

Dans ce chapitre, nous ne considérons que des codes binaires ou sur des extensions de $\mathbb{F}_2$. En effet, en pratique, on utilise seulement des codes sur des corps de caractéristique 2. Ceci est dû en particulier aux implémentations efficaces de calcul dans de tels corps dans les ordinateurs. Par exemple, le corps $\mathbb{F}_{256} = \mathbb{F}_{2^8}$ est une extension de $\mathbb{F}_2$ qui est très utilisée puisque chaque élément de ce corps peut-être codé sur un caractère (8 bits). En général, les éléments facilement codables sur 16, 32 ou 64 bits sont privilégiés puisque l'on

peut utiliser des entiers pour les coder, et il existe des implémentations efficaces pour les opérations élémentaires sur ces objets-ci.

Exemple 13 *Par exemple, une addition d'éléments du corps revient à un XOR $\oplus$ sur les entiers les codant :*
Considérons $\mathbb{F}_{256} = \mathbb{F}_2[\alpha]$, où α est un élément primitif de $\mathbb{F}_{256}$. Ainsi, chaque élément x de $\mathbb{F}_{256}$ peut s'écrire de manière unique sous la forme $x = x_0 + x_1\alpha + \cdots x_7\alpha^7$ avec $x_0, \ldots, x_7 \in \mathbb{F}_2$. Cet élément sera codé sur un caractère (8 bits) de la forme $(x_0, \ldots, x_7)$. Soient $x = x_0 + x_1\alpha + \cdots x_7\alpha^7$ et $y = y_0 + y_1\alpha + \cdots y_7\alpha^7$ deux éléments de $\mathbb{F}_{256}$. Leurs représentations sont $X = (x_0, \ldots, x_7)$ et $Y = (y_0, \ldots, y_7)$. Leur somme donne $x+y = (x_0+y_0)+(x_1+y_1)\alpha+\cdots(x_7+y_7)\alpha^7$ et sa représentation est bien $X \oplus Y = (x_0 \oplus y_0, \ldots, x_7 \oplus y_7)$.

3.2 Reconnaissance d'un code

Tout d'abord, intéressons-nous à un problème basique qui nous servira de brique de base pour la suite : la reconnaissance d'un code. En d'autres mots, sommes-nous capables de décider si une séquence bruitée a bien été codée par un code donné ou non ? Tout d'abord, nous formalisons ce problème décisionnel et puis donnons un algorithme basé sur un test statistique permettant de répondre à ce problème. Ces travaux ont fait l'objet d'une publication dans les Proceedings d'ISIT'07 [Cha07].

3.2.1 Problème de décision

Définition 23
Considérons un canal binaire symétrique $CBS(p)$ de probabilité d'erreur p.
- *Soit $e \in \mathbb{F}_2^n$ un vecteur de n bits produit par ce canal.*
  e est appelé un motif d'erreur.
- *Soit C un code linéaire de longueur n. Soit c un mot de code de C.*
  Le vecteur $x = c \oplus e$ est appelé un mot de code bruité de C.

On note $DP(C, S, p, \epsilon)$ le problème décisionnel suivant :

Entrées :	Un code linéaire binaire C de paramètres $[n, k]$. La probabilité d'erreur p du canal binaire symétrique. Une séquence S de M mots de longueur n. $\epsilon \in [0, 1]$.
Problème :	Est-ce que cette séquence S est constituée de mots de code bruités de C avec une probabilité supérieure à $(1 - \epsilon)$?

Fig. 3.1 – Problème décisionnel de reconnaissance d'un code

Remarque 3 *Notons que si $p = 1/2$, peu importe la séquence entrant dans le canal, la séquence de sortie sera aléatoire. C'est exactement le problème du masque jetable (one-time pad). Dans ce cas-là, le problème est indécidable. Cependant, nous ne considérons que des probabilités d'erreur p petites devant $1/2$. En effet, le contexte d'interception de données lors d'une transmission de données impose que le taux d'erreur soit suffisamment bas pour que l'information soit décodable. Néanmoins, le problème de reconnaissance de codes semble plus facile que le décodage. En effet, le but principal est de retrouver la structure générale du code utilisé et non pas chaque mot de code particulier. Nous espérons donc atteindre des taux d'erreurs supérieurs à la capacité de correction du code.*

Remarque 4 *Notons qu'un tel canal peut produire tous les motifs d'erreur possibles. Cependant, ces motifs d'erreurs n'ont pas tous la même probabilité d'apparaître, et cette probabilité dépend entièrement de la probabilité d'erreur du canal. La probabilité qu'un motif d'erreur de longueur n et de poids ω apparaisse est $P(n,\omega) = p^\omega(1-p)^{n-\omega}C_n^\omega$. Cette notion nous permettra de définir un test statistique et d'évaluer sa probabilité de succès.*

3.2.2 Idée de la méthode

Pour répondre à ce problème décisionnel, il faut construire un distingueur entre une séquence aléatoire et une séquence de mots bruités. Une première approche est d'étudier cette différence en absence de bruit.

Soit C un code linéaire binaire de longueur n. Soit h un mot de son dual $C^\perp$.

$$\begin{array}{ll} \text{Pour tout } x \in \mathbb{F}_2^n, & P(\langle h, x\rangle = 1) = \frac{1}{2}. \\ \text{Pour tout } c \in C, & P(\langle h, c\rangle = 1) = 0. \end{array}$$

Preuve :

- L'espace dual de h est un hyperplan d'un espace vectoriel sur $\mathbb{F}_2$. Il a donc moitié moins d'éléments que l'espace ambiant.
- Par définition du dual, le deuxième point est vérifié.

⊡

Supposons maintenant que $e \in \mathbb{F}_2^n$ soit un motif d'erreur (avec une probabilité d'erreur relativement faible). L'idée principale est la suivante :

$$\begin{array}{ll} \text{Pour tout } x \in \mathbb{F}_2^n, & P(\langle h, x \oplus \mathbf{e}\rangle = 1) = \frac{1}{2}. \\ \text{Pour tout } c \in C, & P(\langle h, c \oplus \mathbf{e}\rangle = 1) \text{ est faible.} \end{array}$$

3.2.3 Reconnaissance d'un hyperplan

Intéressons nous tout d'abord au cas le plus simple, c'est-à-dire quand le code est un hyperplan (le dual est réduit à un seul mot).

Soit h un vecteur de $\mathbb{F}_2^n$. Soit $S = (x_i)_{i=1,\dots,M}$ une séquence de M mots de longueur n. D'après la remarque précédente, pour déterminer si S est une séquence de mots bruités de C, une méthode serait de calculer le nombre N de produits scalaires $\langle h, x_i \rangle$ égaux à 1.

Considérons les différents cas pouvant apparaître pour déterminer ce nombre.

• **Les x_i sont des mots aléatoires :**
Si x est un mot aléatoire, $x \oplus e$ est aussi un mot aléatoire et ainsi

$$P(\langle h, x \oplus e \rangle = 1) = \frac{1}{2}$$

Ceci signifie que N est une variable aléatoire suivant une loi binomiale de paramètres $\mathscr{B}(M, 1/2)$. Ainsi N est proche de $M/2$.

• **Les x_i sont des mots bruités de $h^\perp$:**

Dans ce cas, x_i peut s'écrire sous la forme $x_i = c_i \oplus e_i$ avec c un mot de code de $h^\perp$ et e_i un motif d'erreur généré par $CBS(p)$.

Puisque $c_i \in h^\perp$, $\langle h, x_i \rangle = \langle h, c_i \oplus e_i \rangle = \langle h, e_i \rangle$. Ainsi, il suffit de calculer la probabilité que $\langle h, e \rangle$ soit égal à 1 avec e un motif d'erreur.

Proposition 8 *Soit $e \in \mathbb{F}_2^n$ un motif d'erreur dû à un $CBS(p)$. Alors, la probabilité que son poids de Hamming $w_H(e)$ soit pair est*

$$P(w_H(e) \text{ est pair }) = \frac{1 + (1-2p)^n}{2}.$$

Preuve :
La probabilité que e soit de poids i vaut $C_n^i p^i (1-p)^{n-i}$. Ainsi la probabilité recherchée vaut

$$\sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} C_n^{2i} p^{2i} (1-p)^{n-2i}.$$

Tout d'abord, on sait que

$$(1-2p)^n = ((1-p)-p)^n = \sum_{i=0}^{n} C_n^i (-1)^i p^i (1-p)^{n-i}.$$

Et de même,

$$1 = ((1-p)+p)^n = \sum_{i=0}^{n} C_n^i p^i (1-p)^{n-i}.$$

Ainsi,

$$\frac{1+(1-2p)^n}{2} = \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} C_n^{2i} p^{2i} (1-p)^{n-2i}.$$

⊡

Grâce à ce résultat, on peut exprimer la probabilité que le produit scalaire $\langle h, e \rangle$ soit égal à 1 en fonction de p et de $w_H(h)$.

Corollaire 1 *Soient $h \in \mathbb{F}_2^n$ et $e \in \mathbb{F}_2^n$ un motif d'erreur dû à un $CBS(p)$. Alors :*

$$P(\langle h, e \rangle = 0) = \frac{1+(1-2p)^{w_H(h)}}{2}$$

et

$$P(\langle h, e \rangle = 1) = \frac{1-(1-2p)^{w_H(h)}}{2}.$$

Remarque 5 *Notons que cette probabilité augmente avec le poids de Hamming de h. Ainsi, si l'on veut une probabilité proche de 0 afin de mieux discriminer, il vaut mieux utiliser des mots du dual de poids les plus faibles possibles.*

Ainsi le nombre N de produits scalaires égaux à 1 est proche de $\frac{M}{2} - \frac{M}{2}(1-2p)^{w_H(h)}$ pour une séquence de M mots.

• Les x_i sont des mots bruités de $h_0^\perp$ avec $h_0 \in \mathbb{F}_2^n$ et $h_0 \neq h$:
Les espaces $h^\perp$ et $h_0^\perp$ sont deux hyperplans distincts, ainsi leur intersection est un sous-espace vectoriel de dimension $n-2$. Ainsi la probabilité qu'un mot de $h_0^\perp$ soit dans le dual de h vaut $1/2$. Finalement, nous sommes dans la même situation que pour le cas où les x_i sont des mots aléatoires et donc le nombre N de produits scalaires $\langle h, x_i \rangle$ égaux à 1 est proche de $M/2$.

• Autres cas :
Dans le cas où le code utilisé n'est pas de longueur n, heuristiquement, la séquence découpée en mots de longueur n se comporte comme une séquence de mots aléatoires et le nombre de produits scalaires égaux à 1 sera proche de $M/2$. On ne peut pas le montrer, mais les expérimentations vont dans ce sens. Cependant, il existe quelques cas marginaux, comme par exemple si C' est un code de longueur rn obtenu par concaténation de r fois le code C de longueur n. Dans ce cas, le code C sera reconnu à la place de C'. Mais l'utilisation

d'un tel code C' est sans intérêt.

Finalement, l'idée est confirmée puisque, pour une séquence de M mots :

Si les x_i sont des mots bruités de $h^\perp$,
le nombre N de produits scalaires égaux à 1 sera proche de $\frac{M}{2} - \frac{M}{2}(1-2p)^{w_H(h)}$.

Sinon,
le nombre N de produits scalaires égaux à 1 sera proche de $\frac{M}{2}$.

Dans cette section, nous élaborons un test statistique permettant de confirmer l'idée précédente et de discriminer les deux cas. Tout d'abord nous devons rappeler la notion de probabilité d'échec d'un tel test.

Notons $\mathscr{H}_0$ l'hypothèse nulle :

$$\mathscr{H}_0 = \text{“}S \text{ est une séquence aléatoire”}.$$

Définition 24 *Définissons les deux types de probabilités d'échec d'un test statistique tentant de vérifier $\mathscr{H}_0$.*
- *Probabilité de fausse alarme : $\alpha = P(\text{rejeter } \mathscr{H}_0 | \mathscr{H}_0 \text{ est vérifiée})$.*
- *Probabilité de non-détection : $\beta = P(\text{accepter } \mathscr{H}_0 | \mathscr{H}_0 \text{ n'est pas vérifiée})$.*

Nous allons donc définir le test statistique proposé pour résoudre notre problème et en évaluer les probabilités d'échec en fonction des paramètres.

Définition 25 *Soit h un vecteur de $\mathbb{F}_2^n$. Soit $S = (s_i)_{1 \leq i \leq M}$ une séquence de M mots de longueur n. Soit T un réel compris entre 0 et M. On définit le test statistique $ST1(h, S, T)$ de la façon suivante :*

On décide que S est une séquence de mots bruités de $h^\perp$
si et seulement si
$$\#\{\langle h, s_i \rangle = 1 | 1 \leq i \leq M\} \leq T.$$

Définition 26 *On note ϕ la fonction de répartition de la loi normale centrée réduite :*

$$\phi(x) = \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt.$$

Théorème 4 *Soient h un vecteur de $\mathbb{F}_2^n$ et p la probabilité d'erreur du canal que l'on considère.*
Soient α et β deux réels dans $[0,1]$ et $a = \phi^{-1}(\alpha), b = \phi^{-1}(1-\beta)$.
Fixons

$$M = \left(\frac{b\sqrt{1-(1-2p)^{2w_H(h)}} - a}{(1-2p)^{w_H(h)}} \right)^2 \text{ et } T = \frac{1}{2}\left(M + a\sqrt{M} \right).$$

Si $S = (s_i)_{1 \leq i \leq M}$ est une séquence de M mots de longueur n, alors la probabilité de fausse alarme (resp. de non-détection) du test statistique $ST1(h, S, T)$ est exactement α (resp. β).

Preuve :

Soit X_1 la variable aléatoire définie par $X_1 = \sum_{i=1}^{M} \langle h, s_i \rangle = \sum_{i=1}^{M} \langle h, e_i \rangle$ où S est une séquence de mots bruités de $h^\perp$ ($s_i = c_i + e_i$ avec $c_i \in h^\perp$ et e_i un motif d'erreur dû au $CBS(p)$). La variable aléatoire X_1 suit une loi binomiale de paramètres $\mathscr{B}(M, \tau = \frac{1-(1-2p)^{w_H(h)}}{2})$. Cependant, grâce au théorème central limite, pour des grandes valeurs de M et des valeurs de τ et $1-\tau$ proches, X_1 tend vers une loi normale de moyenne $\mu_1 = M\tau$ et de variance $\sigma_1^2 = M\tau(1-\tau)$.

La variable aléatoire définie par $X_0 = \sum_{i=1}^{M} \langle h, a_i \rangle$ où les a_i sont des mots aléatoires, suit une loi binomiale de paramètres $\mathscr{B}(M, 1/2)$; alors, pour les mêmes raisons, elle peut être approchée par une loi normale de moyenne $\mu_0 = M/2$ et de variance $\sigma_0^2 = M/4$.

La fonction de répartition de la loi normale de moyenne μ et de variance σ^2 peut s'exprimer en fonction de celle de la loi normale centrée réduite de la façon suivante :

$$\Pi_{\mu,\sigma}(x) = \phi\left(\frac{x-\mu}{\sigma} \right).$$

Ainsi, les relations $a = \phi^{-1}(\alpha)$ et $b = \phi^{-1}(1-\beta)$ donnent

$$\phi\left(\frac{T - M/2}{\sqrt{M/4}} \right) = \alpha \text{ et } \phi\left(\frac{T - M\tau}{\sqrt{M\tau(1-\tau)}} \right) = \beta.$$

Ces conditions sont équivalentes à

$$\begin{cases} T = \frac{M}{2} + \phi^{-1}(\alpha)\sqrt{\frac{M}{4}} \\ T = M\tau + \phi^{-1}(1-\beta)\sqrt{M\tau(1-\tau)}. \end{cases}$$

Et nous obtenons

$$\begin{cases} M = \left(\frac{\phi^{-1}(1-\beta)\sqrt{1-(1-2p)^{2w_H(h)}} - \phi^{-1}(\alpha)}{(1-2p)^{w_H(h)}} \right)^2 \\ T = \frac{1}{2}\left(M + \phi^{-1}(\alpha)\sqrt{M} \right). \end{cases}$$

⊡

Remarque 6 *On peut voir que le nombre M de mots nécessaires pour réaliser un tel test augmente exponentiellement en fonction du poids de Hamming du mot du dual h que nous considérons. Il serait donc préférable de prendre un mot du dual de poids relativement faible. Cependant le recherche d'un mot de poids faible dans un code reste un problème très difficile.*

Puisque le nombre de mots nécessaires et le seuil dépendent de p,α et β, on définit de la même façon le test statistique $ST2(h, S, p, \alpha, \beta)$.

Définition 27 *Soit h un vecteur de $\mathbb{F}_2^n$. Soient α, β et p des réels dans $[0, 1]$. Posons*

$$M = \left(\frac{\phi^{-1}(1-\beta)\sqrt{1-(1-2p)^{2w_H(h)}} - \phi^{-1}(\alpha)}{(1-2p)^{w_H(h)}} \right)^2$$
$$\text{et } T = \frac{1}{2}\left(M + \phi^{-1}(\alpha)\sqrt{M} \right).$$

Soit $S = (s_i)_{1 \leq i \leq M}$ une séquence de M mots de longueur n. On définit le test statistique $ST2(h, S, p, \alpha, \beta)$ de la façon suivante :

On décide que S est une séquence de mots bruités de $h^\perp$
si et seulement si
$$\#\{\langle h, s_i \rangle = 1 | 1 \leq i \leq M\} \leq T.$$

Corollaire 2 *La probabilité de fausse alarme (resp. de non-détection) de $ST2(h, S, p, \alpha, \beta)$ est α (resp. β).*

Exemple 14 *Voici quelques courbes du nombre M de mots nécessaires en fonction du poids de Hamming du mot du dual h pour illustrer cette remarque. Dans ces exemples, les probabilités de fausse alarme et de non-détection sont fixées à $\alpha = \beta = 10^{-3}$ puis à 10^{-6}.*

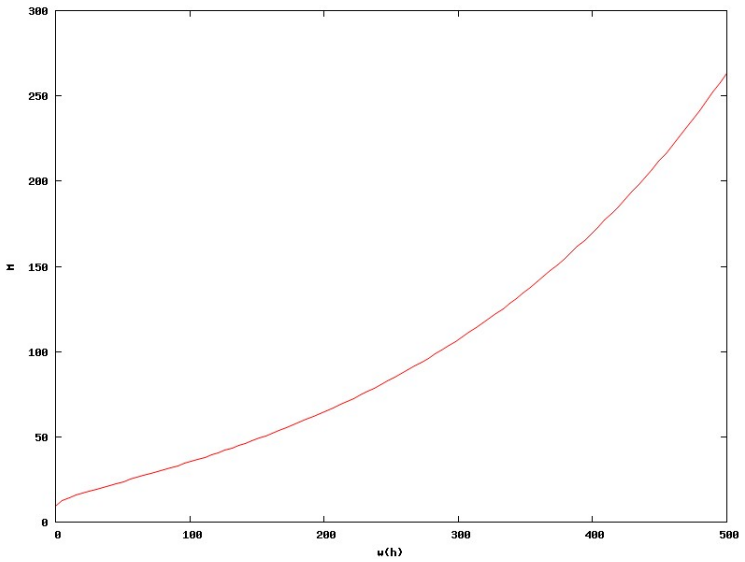


FIG. 3.2 – Valeurs de M en fonction de $w_H(h)$ pour $p = 0.001$ et $\alpha = \beta = 10^{-3}$.

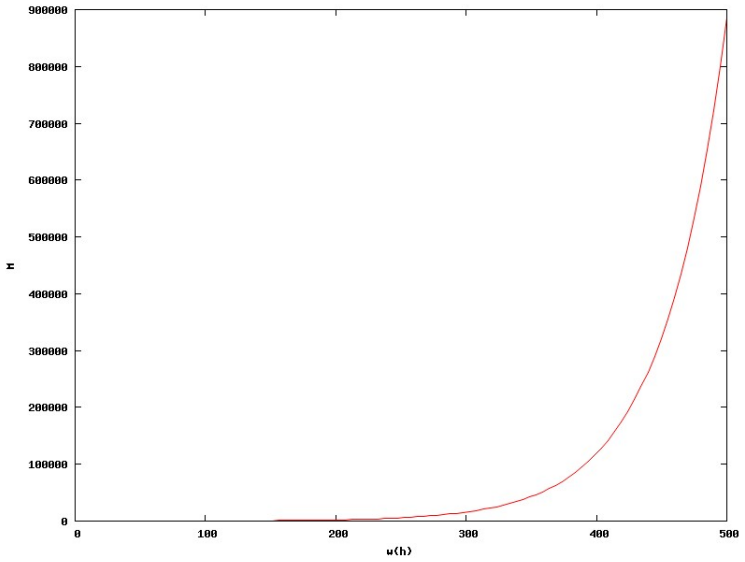


FIG. 3.3 – Valeurs de M en fonction de $w_H(h)$ pour $p = 0.005$ et $\alpha = \beta = 10^{-3}$.

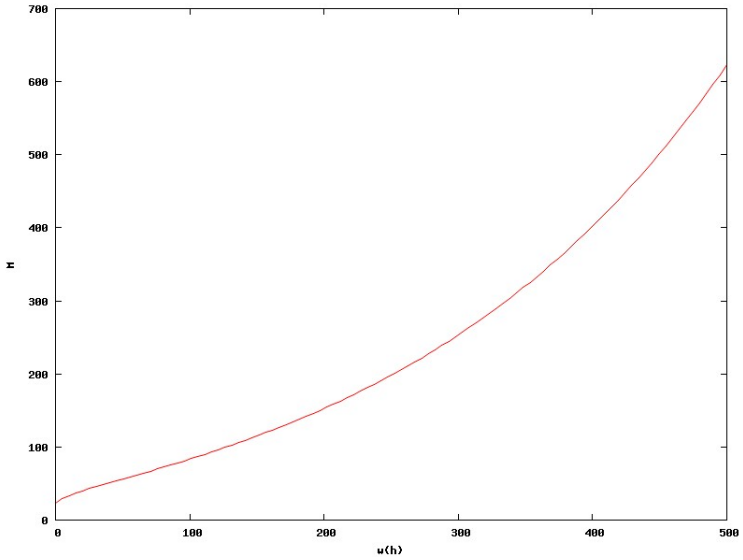


Fig. 3.4 – Valeurs de M en fonction de $w_H(h)$ pour $p = 0.001$ et $\alpha = \beta = 10^{-6}$.

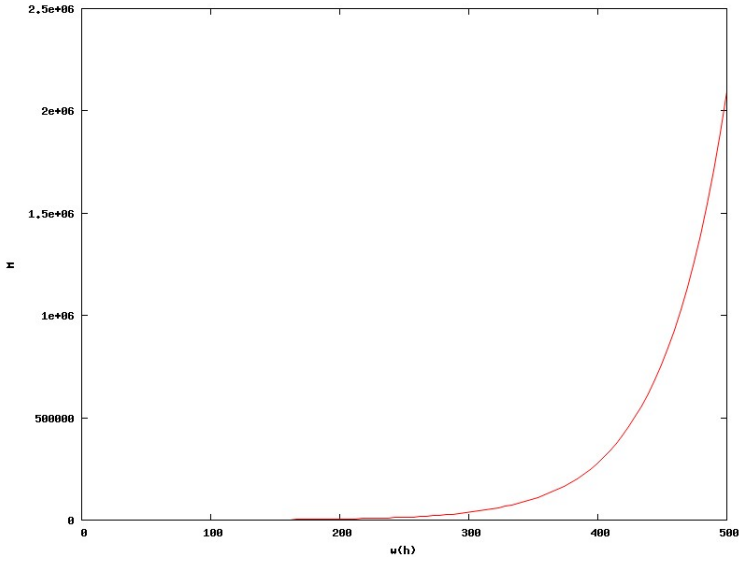


Fig. 3.5 – Valeurs de M en fonction de $w_H(h)$ pour $p = 0.005$ et $\alpha = \beta = 10^{-6}$.

Entrées :
Un canal binaire symétrique $CBS(p)$.
h un vecteur de $\mathbb{F}_2^n$.
$S = (s_i)$ une séquence de vecteurs de $\mathbb{F}_2^n$.
α,β les probabilités de fausse alarme et de non-détection.

Initialisation :
$N = 0$.
Calculer $M = \left(\frac{\phi^{-1}(1-\beta)\sqrt{1-(1-2p)^{2w_H(h)}} - \phi^{-1}(\alpha)}{(1-2p)^{w_H(h)}} \right)^2$ et $T = \frac{1}{2}\left(M + \phi^{-1}(\alpha)\sqrt{M} \right)$.

Algorithme :
Pour j allant de 1 à M
Si $\langle h, s_j \rangle = 1$, alors $N \longleftarrow N + 1$;
Sortie :
Si $N \leq T$, répondre "oui".
Si $N > T$, répondre "non".

FIG. 3.6 – Algorithme de reconnaissance d'un hyperplan $h^\perp$ avec probabilités de fausse alarme et de non détection α et β.

3.2.4 Reconnaissance d'un code

Intéressons-nous maintenant au cas général, c'est-à-dire quand le code C n'est pas forcément un hyperplan. Dans ce cas, on peut considérer $(h_1, \ldots, h_{n-k})$ une base de l'espace dual $C^\perp$ et ainsi :

$$C = \bigcap_{i=1}^{n-k} h_i^\perp.$$

Nous avons donc à vérifier que la séquence est bien composée de mots de codes bruités de chaque hyperplan. Cependant, cela affecte les probabilités d'échec du test statistique final.

En effet, Notons α_i (resp. β_i) les probabilités de fausse alarme (resp. de non-détection) des test statistiques relatifs à chacun des hyperplans $ST2(h_i, S, p, \alpha_i, \beta_i)$.

Proposition 9 *Notons $ST3((h_1, \ldots, h_{n-k}), S, p, (\alpha_1, \ldots, \alpha_{n-k}), (\beta_1, \ldots, \beta_{n-k}))$ le test statistique revenant à décider que S est une séquence de mots de code bruités de C si et seule-*

ment si tous les tests $ST2(h_i, S, p, \alpha_i, \beta_i)$ *sont vérifiés. Ce test a pour probabilités de fausse alarme* $\alpha = \prod_{i=1}^{n-k} \alpha_i$ *et de non détection* $\beta = 1 - \prod_{i=1}^{n-k}(1-\beta_i)$.

Preuve :
Notons α et β les probabilités de fausse alarme et de non-détection du test général.

Alors
$$\begin{aligned}
\alpha &= P(ST3 \text{ renvoie vrai} \mid \quad S \text{ est une séquence aléatoire}) \\
&= P(\forall i, \quad \#\{\langle h_i, s_j\rangle = 1 | 1 \le j \le M_i\} \le T_i | \quad S \text{ est une séquence aléatoire}) \\
&= \prod_{i=1}^{n-k} P(\#\{\langle h_i, s_j\rangle = 1 | 1 \le j \le M_i\} \le T_i | \quad S \text{ est une séquence aléatoire}) \\
&= \prod_{i=1}^{n-k} \alpha_i
\end{aligned}$$
Et
$$\begin{aligned}
\beta &= P(ST3 \text{ renvoie faux} \mid \quad S \text{ est une séquence de mots bruités de } C) \\
&= 1 - P(ST3 \text{ renvoie vrai} \mid \quad S \text{ est une séquence de mots bruités de } C) \\
&= 1 - P(\forall i, \#\{\langle h_i, s_j\rangle = 1 | 1 \le j \le M_i\} \le T_i | \quad S \text{ est une séquence de mots bruités de } C) \\
&= 1 - \prod_{i=1}^{n-k} P(\#\{\langle h_i, s_j\rangle = 1 | 1 \le j \le M_i\} \le T_i | \quad S \text{ est une séquence de mots bruités de } C) \\
&= 1 - \prod_{i=1}^{n-k}(1-\beta_i).
\end{aligned}$$

⊡

Corollaire 3 *Si on suppose de plus que* $\alpha_1 = \cdots = \alpha_{n-k}$ *et* $\beta_1 = \cdots = \beta_{n-k}$, *alors :*

$$\alpha = \alpha_1^{n-k} \text{ et } \beta = (n-k)\beta_1.$$

Définition 28 *Soit* $\mathscr{B}$ *une base du dual de* C.
On note $ST4(\mathscr{B}, S, p, \alpha, \beta)$ *le test statistique* $ST3(\mathscr{B}, S, p, (\alpha_1, \ldots, \alpha_1), (\beta_1, \ldots, \beta_1))$,
avec $\alpha_1 = \alpha^{1/(n-k)}$ *et* $\beta_1 = \beta/(n-k)$.

Remarque 7 *Le test statistique* $ST3(\mathscr{B}, S, p, \alpha, \beta)$ *a pour probabilités de fausse alarme et de non-détection* α *et* β.

Nous allons maintenant donner un algorithme de reconnaissance d'un code C à partir d'une séquence S de vecteurs de $\mathbb{F}_2^n$. Nous imposons que les probabilités d'échec pour la reconnaissance de chacun des hyperplans sont les mêmes ($\alpha_1 = \cdots = \alpha_{n-k}$ et $\beta_1 = \cdots = \beta_{n-k}$). Le nombre de mots nécessaires sera le nombre de mots maximal pour chacune des reconnaissances des hyperplans.

Entrées :
Un canal binaire symétrique $CBS(p)$.
Un code linéaire binaire C.
$S = (s_i)$ une séquence de vecteurs de $\mathbb{F}_2^n$.
α,β les probabilités de fausse alarme et de non-détection.

Initialisation :
$(N_1, \ldots, N_{n-k}) = (0, \ldots, 0)$.
Calculer $(h_1, \ldots, h_{n-k})$ une base de $C^\perp$.
Calculer $\alpha_1 = \alpha^{1/(n-k)}$ et $\beta_1 = \beta/(n-k)$.
Calculer $M = \left(\frac{\phi^{-1}(1-\beta_1)\sqrt{1-(1-2p)^{2\max_i w_H(h_i)}} - \phi^{-1}(\alpha_1)}{(1-2p)^{\max_i w_H(h_i)}} \right)^2$ et $T = \frac{1}{2}\left(M + \phi^{-1}(\alpha_1)\sqrt{M} \right)$.

Algorithme :
Pour i allant de 1 à $n-k$
  Pour j allant de 1 à M
    Si $\langle h_i, s_j \rangle = 1$, alors $N_i \longleftarrow N_i + 1$;
Sortie :
Si $N_i \leq T$ pour tout $i \in \{1, \ldots, n-k\}$, répondre "oui".
Si $N_i > T$ pour au moins un $i \in \{1, \ldots, n-k\}$, répondre "non".

FIG. 3.7 – Algorithme de reconnaissance d'un code C avec probabilités de fausse alarme et de non détection α et β.

Dans la phase d'initialisation, nous avons besoin de calculer une base $(h_1, \ldots, h_{n-k})$ du code dual $C^\perp$. D'après la Remarque 6, il est préférable de choisir des mots de poids relativement faible afin de limiter le nombre de mots nécessaires dans la séquence. Néanmoins, la recherche de mots de poids faibles ne possède pas d'algorithme efficace. Cependant, une manière simple de réduire le poids d'une base du dual est d'écrire la matrice de parité obtenue sous forme systématique. En effet, après cette opération le poids des mots obtenus est proche de $k/2$. Dans le cas où $k = n/2$, ce poids est proche de $n/4$ alors qu'un mot pris au hasard dans cet espace est de poids proche de $n/2$.

Ensuite, cet algorithme n'est pas coûteux en ressources puisqu'il ne demande que le calcul de produits scalaires sur des vecteurs de bits. Ceci est implémentable de manière efficace car il s'agit seulement de ET logiques sur des vecteurs de bits, et des calculs de poids sur des vecteurs de bits.

Proposition 10 *L'algorithme 3.7 nécessite au plus* $M(n-k)$ *masques et* $nM(n-k)$

additions.

Preuve :

Tout d'abord, les calculs préliminaires dans la phase d'initialisation sont négligeables devant la partie principale de l'algorithme.

Ensuite, la partie principale de l'algorithme est composée de $M(n-k)$ calculs d'un produit scalaire entre deux vecteurs de taille n. Le calcul d'un tel produit scalaire peut être effectué en réalisant un ET logique entre les deux vecteurs (1 masque) et en calculant le poids du vecteur résultant (n additions).

Ainsi le nombre d'opérations total est $M(n-k)$ masques et $nM(n-k)$ additions.

⊡

Cas où le corps de base est $\mathbb{F}_{2^m} (m \geq 2)$

L'algorithme présenté ci-dessus ne fonctionne que pour les codes binaires. Cependant, il est possible de l'adapter aux codes construits sur une extension de $\mathbb{F}_2$. En effet, $\mathbb{F}_{2^m}$ peut être vu comme un espace vectoriel de dimension m sur $\mathbb{F}_2$. On peut construire des isomorphismes d'espaces vectoriels entre $\mathbb{F}_{2^m}$ et $\mathbb{F}_2^m$ de la façon suivante :

Si $(1, \alpha, \ldots, \alpha^{m-1})$ est une $\mathbb{F}_2$-base de $\mathbb{F}_{2^m}$ $(\mathbb{F}_{2^m} = \mathbb{F}_2[\alpha] = \mathbb{F}_2 \oplus \mathbb{F}_2.\alpha \oplus \ldots \mathbb{F}_2.\alpha^{m-1})$

$$\begin{array}{rcl} \phi_\alpha : \mathbb{F}_{2^m} & \longrightarrow & \mathbb{F}_2^m \\ a_0 + a_1.\alpha + \cdots + a_{m-1}.\alpha^{m-1} & \longmapsto & (a_0, \ldots, a_{m-1}) \end{array}$$

est un isomorphisme de $\mathbb{F}_2$-espaces vectoriels.

Remarque 8 *En fait, α est une racine d'un polynôme irréductible P_α de degré m sur $\mathbb{F}_2$. Et $\mathbb{F}_{2^m} = \mathbb{F}_2[X]/\langle P_\alpha(X) \rangle$.*

Ainsi, si on fixe une représentation de l'extension, c'est-à-dire un α, il est facile de passer d'une écriture à l'autre. On peut donc écrire tout vecteur de $\mathbb{F}_{2^m}^n$ sous forme d'un vecteur binaire de $\mathbb{F}_2^{mn}$.

Remarque 9 *Si la probabilité d'erreur par symbole d'un vecteur $v \in \mathbb{F}_{2^m}^n$ vaut p, alors la probabilité d'erreur par symbole du vecteur étalé $\phi_\alpha(v) \in \mathbb{F}_2^{mn}$ vaut $p/2$.*

Proposition 11 *Soit $\alpha \in \mathbb{F}_{2^m}$ tel que $\mathbb{F}_{2^m} = \mathbb{F}[\alpha]$. Soit C un code de longueur n et de dimension k sur $\mathbb{F}_{2^m}$ ayant pour base $(b_1, \ldots, b_k)$. Alors C est un code de longueur nm et de dimension km sur $\mathbb{F}_2$ ayant pour base :*

$$\left(\phi_\alpha(b_1), \phi_\alpha(\alpha b_1), \ldots, \phi_\alpha(\alpha^{m-1} b_1), \ldots, \phi_\alpha(b_k), \phi_\alpha(\alpha b_k), \ldots, \phi_\alpha(\alpha^{m-1} b_k)\right).$$

On peut donc reconnaître un code C de longueur n sur $\mathbb{F}_{2^m}$ de la façon suivante :

Entrées :
Un canal 2^m-aire de probabilité d'erreur p.
C un code de longueur n sur $\mathbb{F}_{2^m}$.
$S = (s_i)$ une séquence de vecteurs de $\mathbb{F}_{2^m}^n$.
α,β les probabilités de fausse alarme et de non-détection.

Initialisation :
$(N_1, \ldots, N_{m(n-k)}) = (0, \ldots, 0)$.
Calculer $(h_1, \ldots, h_{n-k})$ une $\mathbb{F}_{2^m}$-base de $C^\perp$.
Choisir $\alpha \in \mathbb{F}_{2^m}^n$ tel que $\mathbb{F}_{2^m}^n = \mathbb{F}_2[\alpha]$.
Calculer $(h'_1, \ldots, h'_{m(n-k)})$ une $\mathbb{F}_2$-base de C (cf Prop. 11).
Calculer $S' = (s'_i)_i = (\phi_\alpha(s_i))_i$.
Calculer $\alpha_1 = \alpha^{1/m(n-k)}$ et $\beta_1 = \beta / m(n-k)$.
Calculer $M = \left(\frac{\phi^{-1}(1-\beta_1)\sqrt{1-(1-p)^{2\max_i w_H(h_i)}} - \phi^{-1}(\alpha_1)}{(1-p)^{\max_i w_H(h_i)}} \right)^2$ et $T = \frac{1}{2}\left(M + \phi^{-1}(\alpha_1)\sqrt{M}\right)$.
où w_H désigne le poids de Hamming du mot étalé sur $\mathbb{F}_2^{mn}$.

Algorithme :
Pour i allant de 1 à $m(n-k)$
  Pour j allant de 1 à M
    Si $\langle h'_i, s'_j \rangle = 1$, alors $N_i \longleftarrow N_i + 1$;
Sortie :
Si $N_i \leq T$ pour tout $i \in \{1, \ldots, m(n-k)\}$, répondre "oui".
Si $N_i > T$ pour au moins un $i \in \{1, \ldots, m(n-k)\}$, répondre "non".

FIG. 3.8 – Algorithme de reconnaissance d'un code C sur $\mathbb{F}_{2^m}$ avec probabilités de fausse alarme et de non détection α et β.

3.2.5 Résultats pratiques

Pour des soucis de faisabilité, la séquence ne doit pas être trop longue. En effet, il faut déjà qu'elle représente une quantité de données interceptable (quelques Mo). De plus, il faut que l'algorithme de reconnaissance renvoie une réponse en un temps raisonnable.

Nous donnons ici quelques tableaux de valeurs représentant le taux d'erreur du canal et le poids de Hamming des mots du dual atteignables avec une séquence de longueur donnée.

• **Avec une séquence de 2^{30} mots :**

p	0.001	0.002	0.005	0.01	0.02	0.05
$w(h)$ avec $\alpha = \beta = 10^{-3}$	4283	2139	853	424	210	81
$w(h)$ avec $\alpha = \beta = 10^{-10}$	3922	1959	781	388	192	74

• **Avec une séquence de 2^{20} mots :**

p	0.001	0.002	0.005	0.01	0.02	0.05
$w(h)$ avec $\alpha = \beta = 10^{-3}$	2552	1274	508	252	125	48
$w(h)$ avec $\alpha = \beta = 10^{-10}$	2191	1094	436	217	107	41

• **Avec une séquence de 2^{10} mots :**

p	0.001	0.002	0.005	0.01	0.02	0.05
$w(h)$ avec $\alpha = \beta = 10^{-3}$	825	412	164	81	40	15
$w(h)$ avec $\alpha = \beta = 10^{-10}$	480	239	95	47	23	9

Exemple 15 *Si l'on veut reconnaître un code de longueur $n = 860$ et de dimension $k = 430$ avec une probabilité d'erreur du canal de $p = 0.01$ et des probabilités d'échec $\alpha = \beta = 10^{-10}$, d'après une remarque précédente, on peut construire des mots du dual de poids $n/4 = 215$. Ainsi, d'après le tableau précédent, une séquence de $M = 2^{20}$ mots est suffisante. Ceci représente une séquence de $Mn = 860 * 2^{20}$ bits.*

L'algorithme codé en C a été testé sur différents codes avec différentes probabilités d'erreur du canal. Pour tous ces tests, les probabilités d'échec sont fixées à 10^{-3}. On utilise une base du dual sous forme systématique, ainsi ces mots sont de poids proche de $k/2$.

• **Pour un code BCH de paramètres** $[511, 250, 63]$ **:**

Le poids maximal des mots de la base considérée est 140.

Taux d'erreur p	Nombre de mots M	Seuil T	Temps en s
0.005	618	270	≤ 1
0.01	10914	5295.5	7

• **Pour un code de Reed-Muller de paramètres** $[1024, 638]$ **:**

Le poids maximal des mots de la base considérée est 136.

Taux d'erreur p	Nombre de mots M	Seuil T	Temps en s
0.005	569	247	≤ 1
0.01	9282	4492	13

• **Pour un code aléatoire de paramètres** $[2000, 1000]$ **:**

Le poids maximal des mots de la base considérée est 535.

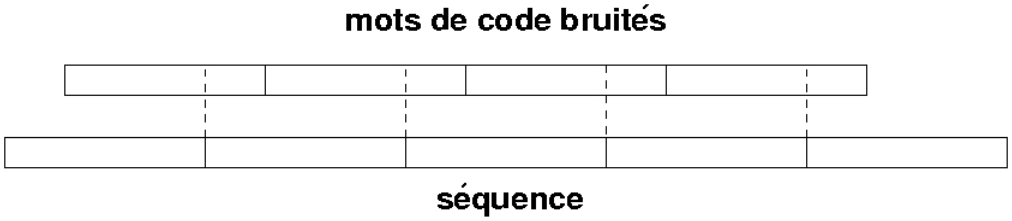
Taux d'erreur p	Nombre de mots M	Seuil T	Temps en s
0.001	306	126	3
0.002	2764	1300	28

3.2.6 Application à la synchronisation

Dans tout ce qui précède, on supposait que la séquence était synchronisée. Cependant, nous allons voir que nous pouvons utiliser notre algorithme de reconnaissance afin de retrouver la synchronisation.

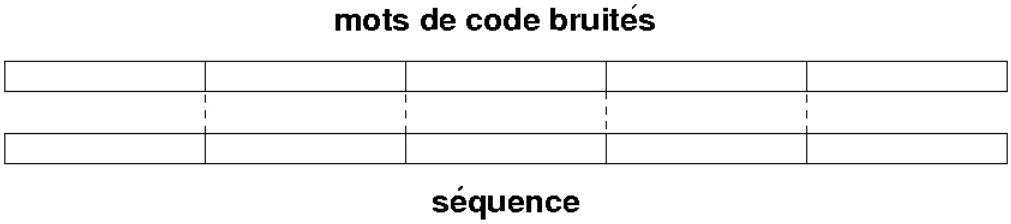
Soit C un code de longueur n. On suppose ici que la séquence S est une suite binaire composée de concaténations de mots de C bruités. On suppose de plus que S est découpée en mots de taille n, cependant, la synchronisation n'est pas forcément bonne.

• **Si la synchronisation est mauvaise :**
Dans ce cas, chaque mot de la séquence est composé de la fin d'un mot de code bruité et du début du suivant.



Dans le cas général, les équations de parité que l'on considère n'ont aucune chance d'être des équations de parité du code décalé. Ainsi, notre algorithme se comporte comme dans le cas d'une séquence aléatoire.

• **Si la synchronisation est bonne :**
Dans ce cas, on se trouve bien en possession d'une séquence de mots bruités du code considéré, et ainsi l'algorithme de reconnaissance peut s'appliquer.



Le problème que nous considérons ici est plus simple que celui de la reconnaissance d'un code. En effet, les mots de la séquence sont ici soit des mots de C bruités (quand la synchronisation est bonne) soit des mots aléatoires (quand la synchronisation n'est pas bonne). Nous avons donc juste à distinguer un code non trivial de l'espace ambiant. C'est-à-dire qu'il faut juste identifier un vecteur du dual du code. Pour cela, nous n'avons pas besoin de considérer toute une base du code dual. Seulement un est nécessaire.

Entrées :
Un canal binaire symétrique $CBS(p)$.
Un code linéaire binaire C de paramètres $[n, k]$.
$S = (s_i)$ une séquence de vecteurs de $\mathbb{F}_2^n$ de mots bruités de C mais peut-être mal synchronisée.
α,β les probabilités de fausse alarme et de non-détection.

Initialisation :
Calculer h un vecteur non nul de $C^\perp$.
Calculer $M = \left(\frac{\phi^{-1}(1-\beta)\sqrt{1-(1-2p)^{2w_H(h)}} - \phi^{-1}(\alpha)}{(1-2p)^{w_H(h)}} \right)^2$ et $T = \frac{1}{2}\left(M + \phi^{-1}(\alpha)\sqrt{M}\right)$.
$sync = 0$.

Algorithme :
Boucle
N=0 ;
Pour j allant de 1 à M
Si $\langle h, s_j \rangle = 1$, alors $N \longleftarrow N + 1$;
Si $N \leq T$, alors
Retourner $sync$;
Sinon
$sync \longleftarrow sync + 1$;
$S \longleftarrow S << 1$; (décalage circulaire à gauche de S)

Sortie :
$sync$ est la bonne synchronisation.

FIG. 3.9 – Algorithme de synchronisation d'un code C avec probabilités de fausse alarme et non détection α et β.

3.3 Reconstruction de familles de codes

Maintenant en possession d'un algorithme de reconnaissance d'un code dans un environnement bruité, nous allons nous intéresser à la reconnaissance d'un code appartenant à une famille connue. En d'autres mots, nous supposons que le code utilisé appartient à une famille de codes que l'on connaît. Ce problème est plus simple que le problème de reconstruction (cas d'un code aléatoire) et ainsi nous obtiendrons des résultats pratiques bien meilleurs. Ces résultats sont publiés dans les Proceedings de WCC'09 [Cha09].

3.3.1 Éléments maximaux d'une famille

Définitions

Dans cette partie, nous donnons des résultats applicables à un cas très général. En effet, nous considérons un corps fini $\mathbb{F}$ (pas nécessairement $\mathbb{F}_2$ ou une extension) et un entier naturel n. Soit $\mathscr{C}$ une famille de sous-ensembles de $\mathbb{F}^n$. Ici, nous n'imposons pas que les éléments sont des codes (des sous-espaces vectoriels). Par contre, nous imposons que $\mathbb{F}^n \in \mathscr{C}$.

Nous munissons cette famille de la relation d'ordre la plus naturelle, l'inclusion $\subset$. Nous obtenons ainsi une famille partiellement ordonnée $(\mathscr{C}, \subset)$.

Définition 29 *Soit C un élément de $\mathscr{C}\backslash\{\mathbb{F}^n\}$. C est appelé un élément maximal de $\mathscr{C}$ si :*

$$(C \subset D \text{ et } D \in \mathscr{C}) \Longrightarrow (D = C \text{ ou } D = \mathbb{F}^n).$$

En d'autres mots, C est un élément maximal de $\mathscr{C}$ s'il n'existe pas de code *entre* C et $\mathbb{F}^n$ dans $(\mathscr{C}, \subset)$.

Remarque 10 *Notons que, puisque $\mathscr{C}$ est une famille finie, le nombre d'éléments maximaux de $\mathscr{C}$ est fini.*

Définition 30 *On note $\mathscr{C}_{max} = \{C_1, \ldots, C_r\}$ l'ensemble des éléments maximaux de $(\mathscr{C}, \subset)$.*

Remarque 11 *Si $\mathscr{C}$ n'est pas réduit à $\{\mathbb{F}^n\}$, $\mathscr{C}$ possède au moins un élément maximal.*

Propriétés

Maintenant que nous avons exhibé des éléments particuliers de la famille : ses éléments maximaux, nous étudions leur rôle dans la description de cette famille.

Proposition 12 *Soit C un élément de $\mathscr{C}$ $(C \neq \mathbb{F}^n)$. Alors, il existe $i \in \{1, \ldots, r\}$ tel que :*

$$C \subset C_i.$$

Ce résultat découle de la définition d'élément maximal.

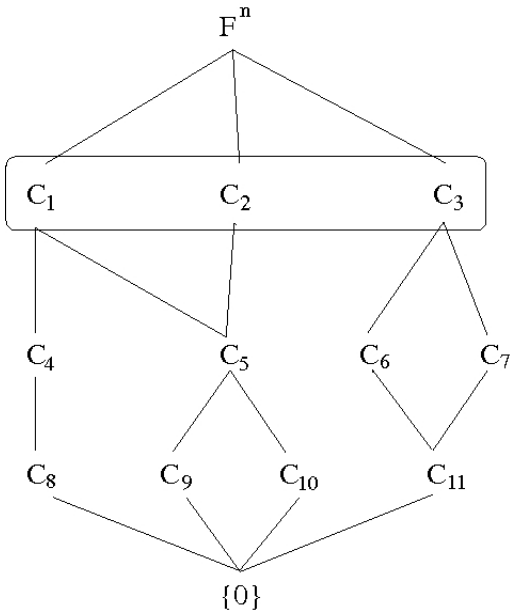


FIG. 3.10 – Exemple de structure d'une telle famille.

Exemple 16 *Ce schéma se lit de la manière suivante : un élément C est relié à un élément D placé plus haut si et seulement si $C \subset D$.*
Dans cet exemple, $\{0\} \subset C_8 \subset C_4 \subset C_1 \subset \mathbb{F}^n$.
D'après la définition d'élément maximaux, ces derniers sont exactement ceux qui sont encadrés (C_1, C_2, C_3). Ils se retrouvent au dernier niveau avant $\mathbb{F}^n$.
Notons que pour chaque élément, on peut vérifier s'il est inclus dans chacun des codes maximaux. Par exemple, $C_5 \subset C_1$ et $C_5 \subset C_2$, mais par contre, $C_5 \not\subset C_3$. D'où la définition suivante.

Définition 31 *Soit C un élément de $\mathscr{C}$. On définit la* signature *de C par*

$$I(C) := \{i \in \{1, \ldots, r\} \quad | \quad C \subset C_i\}.$$

Pour une famille quelconque, $I(\mathbb{F}^n) = \emptyset$ et $I(C_i) = \{i\}$.

Exemple 17 *Dans l'exemple précédent, $I(C_5) = \{1, 2\}$ et $I(\{0\}) = \{1, 2, 3\}$.*

Regroupons maintenant tous les éléments ayant la même signature.

Définition 32 *Soit I un sous-ensemble de $\{1, \ldots, r\}$. On note*

$$\mathscr{C}_I := \{C \in \mathscr{C} \quad | \quad I(C) = I\}$$

l'ensemble des éléments de $\mathscr{C}$ ayant pour signature I.

Exemple 18 *Dans l'exemple précédent,* $\mathscr{C}_{\{1,2\}} = \{C_5, C_9, C_{10}\}$.

Proposition 13 *Soit* $\mathscr{C}$ *une famille de sous-ensembles de* $\mathbb{F}^n$.
Les $\mathscr{C}_I$ *définis ci-dessus forment une partition de* $\mathscr{C}$ *:*

$$\mathscr{C} = \bigsqcup_{I \in \mathscr{P}(\{1,\ldots,r\})} \mathscr{C}_I,$$

où $\mathscr{P}(\{1,\ldots,r\})$ *représente l'ensemble des partitions de* $\{1,\ldots,r\}$.

Preuve :
• D'après la Définition 32, pour tout $I \in \mathscr{P}(\{1,\ldots,r\})$, $\quad \mathscr{C}_I \subset \mathscr{C}$.
Soit $C \in \mathscr{C}$. D'après la Définition 32, $C \in \mathscr{C}_{I(C)}$. Alors,

$$\mathscr{C} \subset \bigcup_{I \in \mathscr{P}(\{1,\ldots,r\})} \mathscr{C}_I$$

$$\text{et } \mathscr{C} = \bigcup_{I \in \mathscr{P}(\{1,\ldots,r\})} \mathscr{C}_I.$$

• Soient $I, J \in \mathscr{P}(\{1,\ldots,r\})$ tels que $\mathscr{C}_I$ et $\mathscr{C}_J$ ne sont pas vides.
Supposons qu'il existe $C \in \mathscr{C}_I \cap \mathscr{C}_J$.

$$\text{Comme } C \in \mathscr{C}_I,\ I(C) = I.$$
$$\text{Comme } C \in \mathscr{C}_J,\ I(C) = J.$$

Et ainsi, $I = J$.
Par conséquent, si I et J sont deux éléments distincts de $\mathscr{P}(\{1,\ldots,r\})$, alors $\mathscr{C}_I \cap \mathscr{C}_J = \emptyset$.

$$\text{D'où } \mathscr{C} = \bigsqcup_{I \in \mathscr{P}(\{1,\ldots,r\})} \mathscr{C}_I.$$

⊡

3.3.2 Application à la reconstruction

Nous considérons ici une famille $\mathscr{C}$ de codes linéaires de longueur n sur $\mathbb{F}_2$ (ou sur une extension $\mathbb{F}_{2^m}$, cf Algorithme 3.8). L'idée principale, afin de reconstruire le code C utilisé, est de calculer sa *signature* $I(C)$ à partir de la séquence de mots bruités interceptée. En effet, à partir de cette signature, on pourra construire la sous-famille $\mathscr{C}_{I(C)}$ et par définition, C appartient à cet ensemble. Ainsi, on réduit une recherche exhaustive sur la famille entière $\mathscr{C}$ à la sous-famille $\mathscr{C}_{I(C)}$. Bien entendu, plus cette sous-famille est petite, plus cette

méthode sera efficace.

Plus précisément, si nous possédons une séquence de mots bruités de C, il suffit d'appliquer la méthode de *reconnaissance d'un code* (binaire ou sur $\mathbb{F}_{2^m}$) pour chaque code maximal $C_1, \ldots, C_r$. De cette manière, pour chaque code maximal C_i, on sait si la séquence est composée de mots de code bruités de C_i. On calcule ainsi la signature $I(C)$ du code utilisé.

Entrées :
Un canal binaire symétrique $CBS(p)$.
Un code linéaire binaire C de paramètres $[n, k]$
appartenant à la famille $\mathscr{C}$ avec $\mathscr{C}_{max} = \{C_1, \ldots, C_r\}$.
$S = (s_i)$ une séquence de vecteurs de $\mathbb{F}_2^n$.
α,β les probabilités de fausse alarme et de non-détection.

Initialisation :
$I = \{\}$;
Calculer $\alpha_0 = \alpha^{1/r}$ et $\beta_0 = \beta/r$.
Calculer $\mathscr{B}_1, \ldots, \mathscr{B}_r$ des bases respectives de $C_1^\perp, \ldots, C_r^\perp$.

Algorithme :
Pour i allant de 1 à r
Si S est une séquence de mots bruités de C_i
(En utilisant $ST4(\mathscr{B}_i, S, p, \alpha_0, \beta_0)$)
$I \longleftarrow I \cup \{i\}$;
Calculer $\mathscr{C}_I$.

Sortie :
S est une séquence de mots bruités
d'un code appartenant à $\mathscr{C}_I$.

FIG. 3.11 – Algorithme de reconstruction d'un code C appartenant à une famille connue $\mathscr{C}$ avec probabilités de fausse alarme de de non-détection α et β.

Moralement, avec cette méthode, la recherche exhaustive sur la famille entière devient une recherche exhaustive sur l'ensemble des codes maximaux. Cette méthode est efficace si le cardinal de l'ensemble des éléments maximaux est petit devant la taille de la famille. Considérons différentes familles et étudions leur structure.

• **Codes de Reed-Muller de longueur 2^m :**

Comme nous l'avons vu dans le chapitre introductif, les codes de Reed-Muller d'une longueur donnée sont tous imbriqués les uns dans les autres. Ainsi, on obtient une structure de la forme suivante :

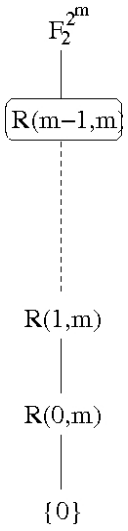


FIG. 3.12 – Structure de la famille des codes de Reed-Muller de longueur 2^m.

Dans ce cas, on n'obtient qu'un seul élément maximal $R(m-1,m)$ et deux sous-familles $\mathscr{C}_{\{\}} = \{\mathbb{F}_2^{2^m}\}$ et $\mathscr{C}_{\{1\}} = \mathscr{C} \backslash \{\mathbb{F}_2^{2^m}\}$.

Ainsi, notre algorithme de reconstruction permet seulement de distinguer un code non trivial de $\mathbb{F}^n$.

3.3.3 Application à la reconstruction des codes cycliques

Soient n un entier naturel impair et $\mathbb{F}$ un corps fini de caractéristique 2. On considère $\mathscr{C}$ la famille des codes cycliques de longueur n sur $\mathbb{F}$.

Rappelons que chaque code cyclique est généré par un facteur unitaire de $X^n - 1$. Nous avons donc la correspondance suivante :

$$\begin{array}{rcl} \phi : \{ \text{ facteurs de } X^n - 1\} & \longrightarrow & \mathscr{C} \\ g(X) & \longmapsto & \langle g(X) \rangle \end{array}$$

De plus, les codes cycliques maximaux de longueur n sont exactement les facteurs irréductibles de $X^n - 1$ dans $\mathbb{F}[X]$. Si on note $g_1(X), \ldots, g_r(X)$ les facteurs irréductibles de

X^n-1, alors les codes cycliques maximaux de longueur n sont exactement les $\langle g_i(X)\rangle$, $\quad i = 1,\ldots,r$.

Exemple 19 *Prenons $n = 7$ et $\mathbb{F} = \mathbb{F}_2$. $X^7 - 1$ se factorise de la façon suivante :*

$$X^7 - 1 = (X+1)(X^3+X^2+1)(X^3+X+1),$$

les trois facteurs étant irréductibles. Ainsi, les codes cycliques maximaux de longueur 7 sont exactement $\langle g_1(X)\rangle$, $\langle g_2(X)\rangle$ et $\langle g_3(X)\rangle$ avec

$g_1(X) = X+1,$
$g_2(X) = X^3+X^2+1$
et $g_3(X) = X^3+X+1.$

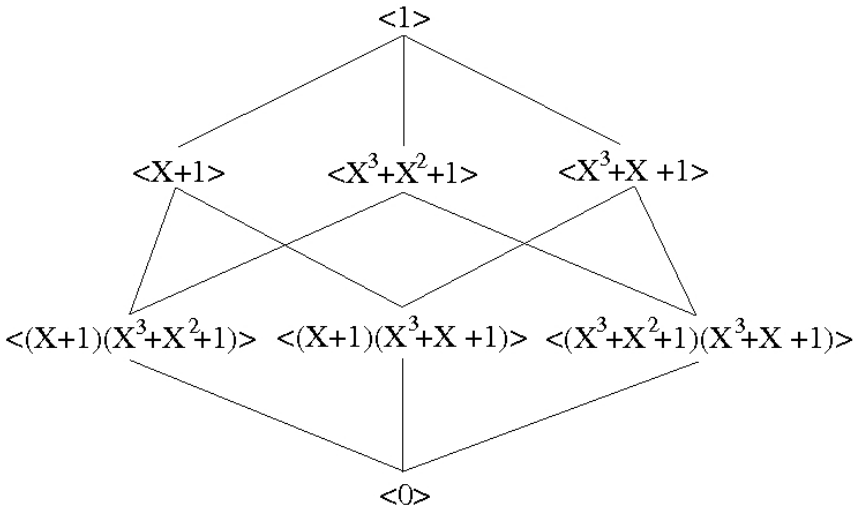


FIG. 3.13 – Structure de la famille des codes cycliques de longueur 7 sur $\mathbb{F}_2$.

Algorithme de reconstruction

La méthode de reconstruction d'un code appartenant à une famille connue de l'Algorithme 3.11 est basée sur le fait que

$$C \in \mathscr{C}_{I(C)}.$$

Cependant, dans le cas des codes cycliques, $\mathscr{C}_I$ est réduit à un unique élément :

$$\mathscr{C}_I = \left\{ \bigcap_{i \in I} \langle g_i(X) \rangle \right\} = \left\{ \langle \prod_{i \in I} g_i(X) \rangle \right\}.$$

Ainsi, on a l'égalité

$$\{C\} = \mathscr{C}_{I(C)},$$

et notre algorithme de reconstruction ne retournera pas seulement une sous-famille de codes possibles, mais le code recherché.

Dans l'algorithme de reconstruction, nous devons considérer les espaces duaux $C_i^\perp$. Mais dans le cas des codes cycliques, ils sont faciles à décrire.

$$C_i^\perp = \langle g_i(X) \rangle^\perp = \langle \left(\frac{X^n - 1}{g_i(X)} \right)^* \rangle =: \langle h_i(X) \rangle$$

où $g^*(X) = \sum_{i=0}^{\deg(g)} a_{\deg(g)-i} X^i$ est le polynôme réciproque de $g(X) = \sum_{i=0}^{\deg(g)} a_i X^i$.

Ainsi, une base de $C_i^\perp$ est

$$\left(h_i(X), X h_i(X), \ldots, X^{\deg(g)-1} h_i(X) \right),$$

ou en notation vectorielle

$$\left(h_i, \sigma(h_i), \ldots, \sigma^{\deg(g)-1}(h_i) \right),$$

où h_i est la représentation vectorielle de $h_i(X)$.

Exemple 20 *Dans l'exemple précédent des codes cycliques de longueur* 7 *sur* $\mathbb{F}_2$,
$h_1(X) = (X^3 + X^2 + 1)(X^3 + X + 1)$,
$h_2(X) = (X^3 + X^2 + 1)(X + 1)$
et $h_3(X) = (X + 1)(X^3 + X + 1)$.

Remarque 12 *Puisque le code considéré est cyclique, si l'algorithme retourne que* h_i *est bien un mot du dual, alors il retournera nécessairement que* $\sigma(h_i), \ldots, \sigma^{\deg(g_i)-1}(h_i)$ *le sont aussi. Ainsi, pout tout* i *allant de* 1 *à* r*, il suffit de vérifier que* S *est une séquence de mots bruités de* $h_i^\perp$ *(et non plus de* C_i*).*

Grâce à toutes ces remarques et aux algorithmes précédents, on obtient l'algorithme de reconstruction de codes cycliques suivant :

Entrées :
Un canal binaire symétrique $CBS(p)$.
$S = (s_i)$ une séquence de vecteurs de $\mathbb{F}_2^n$.
α,β les probabilités de fausse alarme et de non-détection.

Initialisation :
$g(X) := 1$;
Factoriser $X^n - 1 = \prod_{i=1}^{r} g_i(X)$ en facteurs irréductibles.
Calculer $h_i(X) := \left(\frac{X^n-1}{g_i(X)}\right)^*$;
Calculer $\alpha_0 = \alpha^{1/r}$ et $\beta_0 = \beta/r$.

Algorithme :
Pour i allant de 1 à r
Si S est une séquence de mots bruités de $h_i^\perp$
(En utilisant $ST2(h_i, S, p, \alpha_0, \beta_0)$)
$g(X) \longleftarrow g(X) \times g_i(X)$;

Sortie :
S est une séquence de mots bruités de $\langle g(X) \rangle$.

FIG. 3.14 – Algorithme de reconstruction d'un code cyclique binaire de longueur n avec probabilités de fausse alarme et de non-détection α et β.

Analyse de complexité

Proposition 14 *Soit $\mathscr{C}$ une famille de codes cycliques de longueur n comprenant r codes maximaux. Pour reconnaître un code cyclique dans $\mathscr{C}$ avec probabilités de fausse alarme et de non-détection α et β, il faut une séquence d'au moins M mots avec*

$$M = \left(\frac{s\sqrt{1-(1-2p)^{2\max_i(w_H(h_i))}} - a}{(1-2p)^{\max_i(w_H(h_i))}} \right)^2$$

où $a = \phi^{-1}(\alpha^{1/r})$ et $b = \phi^{-1}(1-\beta/r)$.

Le nombre d'opérations binaires est en $\mathscr{O}(rMn)$.

Preuve :
Soient $\alpha_0 = \alpha^{1/r}$ et $\beta_0 = \beta/r$.
Pour reconnaître un code $C \in \mathscr{C}$, il faut accomplir r tests statistiques $ST2(h_i, S, p, \alpha_0, \beta_0)$. D'après le Théorème 4, tous ces tests ont des probabilités de fausse alarme et de non-détection α_0 et β_0 et requièrent au plus ce nombre M de mots.

En itérant successivement r tests statistiques indépendants avec probabilités d'échec α_0 et β_0, le résultat final sera correct avec probabilités de fausse alarme $\alpha_0^r = (\alpha^{1/r})^r = \alpha$ et de non-détection $r\beta_0 = r\beta/r = \beta$.

Finalement, il faut calculer M produits scalaires sur des mots de longueur n pour chacun des r codes maximaux. Ainsi, le nombre d'opérations binaires est en $\mathscr{O}(rMn)$. ⊡

Résultats expérimentaux

Dans la table suivante sont comparés les temps de calcul pour reconstruire un code cycliques de notre algorithme avec celui de reconstruction d'un code aléatoire de *M. Cluzeau*.

Pour notre version, on utilise l'Algorithme 3.14 implémenté en *Magma* (Cf Annexe). Les temps de calcul correspondent au temps moyen de la reconstruction totale du code cyclique.

Pour la version de *M. Cluzeau*, l'algorithme de reconstruction d'un code aléatoire est implémenté en *C*. Les temps de calcul correspondent au temps moyen de reconstruction d'un mot du dual. En effet, dans la plupart des cas, un seul mot du dual est suffisant pour reconstruire le code entier. (En utilisant le fait que si h est un mot du dual, $\sigma(h), \ldots, \sigma^n(h)$ en sont aussi).

Longueur n	$\#\mathscr{C}_{max}$ r	Erreur p	Temps Cluzeau	Temps Notre algo.
123	8	0.01	$\leq$ 1 s	$\leq$ 1 s
123	8	0.02	7 min	3 s
511	59	0.001	$\leq$ 1 s	$\leq$ 1 s
511	59	0.002	$\geq$ 10 min	1 s
511	59	0.005		14 s
1011	34	0.001	$\geq$ 10 min	2 s
1011	34	0.002		9 s
1011	34	0.003		66 s

FIG. 3.15 – Résultats expérimentaux avec $\alpha_0 = \beta_0 = 10^{-6}$ ($\alpha = 10^{-6r}$ et $\beta = r10^{-6}$).

Chapitre 4

Codes convolutifs

4.1 Reconstruction de codes convolutifs

Comme dans le cas de la reconstruction des codes linéaires en blocs, la recherche académique sur la reconstruction des codes convolutifs est assez récente. En effet, les premiers résultats apparurent en 1995 avec *B. Rice* [Ric95] résolvant ce problème pour les $(n, 1)$-codes convolutifs dans des canaux sans bruit. Ensuite, *E. Filiol* donne une technique pour reconstruire des codes convolutifs dans des canaux bruités [Fil97, Fil00, Fil01]. Finalement, *J. Barbier*, s'appuyant sur les travaux de *E. Filiol*, améliora les performances en utilisant des algorithmes de reconstruction de codes en blocs linéaires [BGH06, Bar07]. Au même moment, *J. Dingel* [DH07] propose une solution pour la détection des codes convolutifs de taux $1/n$.

Le principe de l'algorithme de reconstruction est le suivant. Tout d'abord, étudions le phénomène dans le cas sans bruit. Nous sommes amenés à chercher des relations entre les bits de sortie d'un codeur convolutif et les polynômes composant sa matrice génératrice $G(Z)$. La méthode se décompose en deux étapes. Dans la première, on met en équation les bits de sortie, les polynômes générateurs et les bits d'entrée grâce à la définition de l'encodeur. Une manipulation de ces équations nous mènent à un système d'équations mettant en relation les bits de sortie et certains mineurs d'ordre k de $G(Z)$. Remarquons que les bits d'entrée n'entrent plus en jeu dans les équations.

La deuxième étape consiste en la résolution de ces équations afin de retrouver les polynômes générateurs.

Dans le cas avec erreur, il utilise les mêmes techniques que pour la reconstruction des codes en blocs linéaires. En effet, en utilisant l'algorithme de *Gauss* randomisé, il arrive à retrouver les équations à partir d'une matrice bruitée.

Proposition 15 *Dans un canal parfait (sans bruit), le nombre minimum de bits requis*

dans la séquence interceptée pour reconstruire un (n, k, m)-code convolutif est

$$n((k+2)(m+1)-1).$$

La complexité de calcul est de

$$O(n^5 m^4).$$

Dans le cas bruité, résoudre le système revient à reconstruire un code linéaire en bloc de longueur $(k+1)(m+1)$. Notons $N(p, P_{succ})$ la nombre d'itérations du pivot de *Gauss* nécessaires pour réussir avec une probabilité de succès P_{succ} et un taux d'erreur p. Ainsi, la complexité de l'algorithme de reconstruction devient

$$O(N(p, P_{succ})n^5 m^4).$$

Et la longueur nécessaire de la séquence devient

$$n(N(p, P_{succ}) + m).$$

4.2 Détection des paramètres

Jusqu'à maintenant, pour reconstruire un code convolutif, nous devions faire une recherche exhaustive sur les paramètres du code. Cependant, si la recherche est faite de manière subtile - les paramètres les plus utilisés sont vérifiés en premier - les résultats sont meilleurs. Mais de façon générale, tester des mauvais paramètres est une perte de temps. Ceci nous motive à nous intéresser à la détection de paramètres d'un code convolutif. De plus, ce problème semble plus facile, et sa résolution permettrait une accélération de la phase de reconstruction.

Plusieurs paramètres peuvent être étudiés, mais ici, nous nous intéressons à :

- k le nombre d'entrées du codeur
- n le nombre de sorties du codeur
- m le nombre de cases mémoire du codeur

Ainsi, notre but est de trouver un invariant sur des séquences codées bruitées pour chaque jeu de paramètres (n, k, m).

4.2.1 Observations

Ce travail a débuté par des observations statistiques de séquences codées bruitées. Pour cela, nous avons appliqué les tests du *NIST* [Nist01] sur des séquences codées par différents codes convolutifs avec différents niveaux de bruit. La conclusion de ces tests est

qu'une séquence codée par un code convolutif, même bruitée, n'est pas considérée comme aléatoire au regard de la compressions. En effet, sont regroupés dans ce tableau les taux de compression obtenus sur des séquences aléatoires et provenant d'un code convolutif grâce à différents algorithmes de compression. Ici, les séquences ne sont pas bruitées.

	gzip	bzip2 -1	bzip2 -9
Séquence aléatoire	1	1.008	1.004
Code convolutif $(1, 2, 8)$	0.999	0.986	0.804
Code convolutif $(1, 3, 8)$	0.982	0.937	0.682
Code convolutif $(1, 2, 5)$	0.974	0.676	0.556

FIG. 4.1 – Comparaison de taux de compression (sans bruit).

Remarque 13 gzip *correspond à l'algorithme de compression de* Lempel-Ziv *et* bzip2 *correspond à celui de* Burrows-Wheeler. *Les options "-1" et "-9" correspondent à la taille de la fenêtre utilisée dans l'algorithme de* Burrows-Wheeler *(nous détaillerons cela plus tard).*

Les séquences aléatoires ne peuvent pas être réduites par compression. Pour *bzip2*, les fichiers compressés sont même plus volumineux que les fichiers initiaux ; cela est dû à l'insertion d'en-têtes dans le fichier compressé. Mais pour les séquences codées par un code convolutif, la compression est efficace. On remarque même qu'elle est meilleure avec *bzip2* qu'avec *gzip* et qu'elle augmente avec la taille des fenêtres considérées. Pour le troisième code, on atteint même presque un taux de compression de 1/2, ce qui est le rendement du code ; la compression est donc presque complète dans ce cas-là. Finalement *Burrows-Wheeler* semble être un bon axe de recherche pour trouver des biais statistiques sur de telles séquences.

Regardons ce qui se passe en présence de bruit. Sont répertoriés dans ce tableau les taux de compression obtenus pour un code convolutif de paramètres $(1, 2, 5)$ avec différents niveaux de bruit.

Taux d'erreur	gzip	bzip2 -1	bzip2 -9
0	0.974	0.676	0.556
0.001	0.976	0.698	0.592
0.005	0.982	0.776	0.689
0.01	0.987	0.838	0.765
0.05	1.0002	0.994	0.977
0.1	1.0002	1.008	1.004

FIG. 4.2 – Comparaison de taux de compression (avec bruit).

Remarque 14 *On remarque que l'ajout du bruit dégrade progressivement l'efficacité de la compression jusqu'à ce que la compression ne soit plus possible. Ces résultats semblent pourtant montrer que le bruit est quantifiable en fonction du taux de compression obtenu. Et d'après le Tableau 4.1 les taux de compression semblent différents suivant les paramètres du code utilisé. Dans la suite, nous élaborons une méthode utilisant ce fait pour retrouver les paramètres du code utilisé.*

4.2.2 Transformée de Burrows-Wheeler

L'algorithme de compression de *Burrows-Wheeler* est composé de deux étapes. La première consiste à réarranger intelligemment la séquence afin d'exhiber des motifs redondants. La seconde consiste à effectuer un codage source, typiquement du Huffman adaptatif. Puisque notre but n'est pas de compresser mais d'exhiber des biais statistiques, nous considérons seulement la première étape appelée *Transformation de Burrows-Wheeler*.

Considérons $e = (e_0, e_1, \ldots, e_{N-1})$ une séquence binaire de longueur N et construisons la matrice circulante M_e suivante à partir de e :

$$M_e := \begin{pmatrix} e_0 & e_1 & \cdots & e_{N-1} \\ e_1 & e_2 & \cdots & e_0 \\ \vdots & & \ddots & \vdots \\ e_{N-1} & e_0 & \cdots & e_{N-2} \end{pmatrix}.$$

Maintenant, transformons M_e en M'_e en rangeant les lignes de M_e dans l'ordre lexicographique $\leq$. Si on note φ la permutation sur les lignes correspondante, on a :

$$M'_e := P_\varphi M_e = \begin{pmatrix} e_{\varphi(0)} & e_{\varphi(0)+1 \bmod N} & \cdots & \mathbf{e_{\varphi(0)+N-1 \bmod N}} \\ e_{\varphi(1)} & e_{\varphi(1)+1 \bmod N} & \cdots & \mathbf{e_{\varphi(1)+N-1 \bmod N}} \\ \vdots & & \ddots & \vdots \\ e_{\varphi(N-1)} & e_{\varphi(N-1)+1 \bmod N} & \cdots & \mathbf{e_{\varphi(N-1)+N-1 \bmod N}} \end{pmatrix}.$$

où

$$(e_{\varphi(0)}, \ldots, e_{\varphi(0)+N-1 \bmod N}) \leq \cdots \leq (e_{\varphi(N-1)}, \ldots, e_{\varphi(N-1)+N-1 \bmod N}).$$

Finalement, dans l'algorithme de compression, $\varphi^{-1}(0)$ et la dernière colonne de M'_e sont suffisants pour retrouver la séquence initiale. Mais ici, nous gardons seulement la dernière colonne, ce qui signifie qu'on réarrange seulement la séquence.

Exemple 21 *Prenons* $N = 7$ *et* $e = (1, 0, 0, 1, 0, 0, 1)$. *On a :*

$$M_e = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Et ainsi,

$$M'_e = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & \mathbf{1} \\ 0 & 0 & 1 & 1 & 0 & 0 & \mathbf{1} \\ 0 & 1 & 0 & 0 & 1 & 1 & \mathbf{0} \\ 0 & 1 & 1 & 0 & 0 & 1 & \mathbf{0} \\ 1 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} \\ 1 & 0 & 0 & 1 & 1 & 0 & \mathbf{0} \\ 1 & 1 & 0 & 0 & 1 & 0 & \mathbf{0} \end{pmatrix}.$$

Finalement, la séquence de sortie est $e' = (1, 1, 0, 0, 1, 0, 0)$.

Remarque 15 *Remarquons que dans cet exemple, le motif* 1001 *apparaît deux fois dans la séquence originale. En fait, il réapparaît aussi dans les deux premières lignes de* M'_e *:*

$$\begin{array}{ccccccc} \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 1 & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 1 & 0 & 0 & \mathbf{1} \end{array}$$

Et ainsi, deux '1' consécutifs apparaissent dans la séquence de sortie.

Finalement, les motifs redondants dans la séquence initiale impliquent des suites de bits identiques dans la séquence de sortie. C'est pour cela que dans l'algorithme de compression, on effectue un codage source aussitôt après. Nous utilisons ici ce fait pour construire notre test statistique.

4.2.3 Test des runs

Définition 33 *Soit $e = (e_0, e_1, \ldots, e_{N-1})$ une séquence binaire de longueur N. On appelle* run *de e de longueur ℓ toute sous-séquence maximale de ℓ bits identiques de e :*

$$(e_j, e_{j+1}, \ldots, e_{j+\ell-1}) \text{ est un run}$$

$$\iff \begin{pmatrix} e_j = e_{j+1} = \cdots = e_{j+\ell-1} \\ e_{j-1} \neq e_j \\ e_{j+\ell-1} \neq e_{j+\ell} \end{pmatrix}.$$

Proposition 16 *Soit e une séquence de N bits où le nombre de '0' (resp. de '1') vaut N_0 (resp. N_1). On note $\pi_0 = N_0/N$ et $\pi_1 = N_1/N$ les fréquences de '0' et de '1'. Alors, le nombre moyen de runs vaut :*

$$2(N-1)\pi_0\pi_1 + 1.$$

Si $\pi_0 \simeq \pi_1$,

$$2(N-1)\pi_0\pi_1 + 1 \simeq \frac{N}{2}.$$

Comme dans les tests statistiques du NIST, afin d'évaluer le caractère aléatoire d'une séquence, on utilise une P_{value}, c'est-à-dire une valeur numérique calculée à partir d'un test statistique comparant une valeur observée avec une valeur attendue dans le cas aléatoire. Dans notre cas, nous comparons le nombre de runs observés avec $2(N-1)\pi_0\pi_1 + 1$ (Cf Proposition 16) de la façon suivante :

$$P_{value} := \frac{|N_{obs} - (2(N-1)\pi_0\pi_1 + 1)|}{S\sqrt{2N}\pi_0\pi_1}.$$

Dans notre cas, contrairement aux tests du NIST, plus la P_{value} est proche de 0, plus la séquence est considérée aléatoire du point de vue des runs. Et plus la P_{value} est grande, plus la séquence est considérée comme non-aléatoire. Dans les tests du NIST, si la P_{value} est en-dessous d'un seuil fixé, la séquence est considérée comme non-aléatoire.

4.2.4 Test statistique

Test :
- Une séquence S de N bits.
- Obtenir S' en appliquant la transformation de *Burrows-Wheeler* de longueur N à S.
- Calculer la P_{value} (relative aux runs) de la séquence S'.

Une chose intéressante est que, si on fixe
- N la longueur de la séquence
- k le nombre d'entrées du codeur
- n le nombre de sorties du codeur
- m le nombre de cases mémoire du codeur

alors les courbes de P_{value} en fonction de p, le taux d'erreur du canal, sont toujours les mêmes peu importe les polynômes utilisés (on ne considère pas de codes dégénérés). On obtient des courbes telles que dans la Figure 4.3.

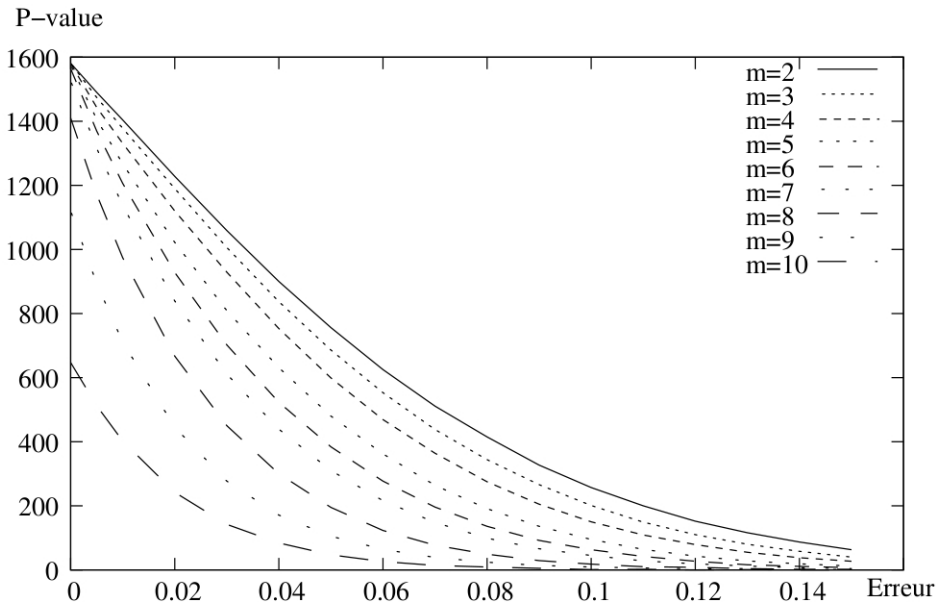


FIG. 4.3 – Courbes-type avec les paramètres $N = 10^7, k = 1, n = 2$.

Nous allons utiliser le fait que les courbes pour chaque jeu de paramètres sont toutes différentes pour construire notre algorithme de détection.

4.2.5 Algorithme de détection

Quand k, n et p sont connus

Dans ce cas le plus simple, le but est de retrouver m le nombre de cases mémoire du codeur. La Figure 4.3 illustre ce cas. Il est alors facile de retrouver m, puisque pour un couple (p, P_{value}) donné, il n'y a qu'un seul choix pour m.

Entrées :
Un canal binaire symétrique $CBS(p)$.
S une séquence de N bits codée par un code convolutif de paramètres (n, k, m) bruitée.
Algorithme :
Transformer S en S' par *Burrows-Wheeler*.
Calculer la P_{value} de S'.
Sortie :
Retourner le seul m correspondant à (p, P_{value}).

Fig. 4.4 – Algorithme pour retrouver m quand k, n et p sont connus.

Quand k et n sont connus

Dans ce cas, c'est un peu plus difficile puisque pour un couple (p, P_{value}), plusieurs valeurs de m sont possibles. Mais nous connaissons, à travers les courbes-type le comportement de notre algorithme quand le taux d'erreur augmente. En effet, pour chaque valeur de m, on a une courbe-type de la P_{value} en fonction de p. Ainsi, l'idée est d'ajouter de l'erreur artificielle à notre séquence et calculer une nouvelle P_{value}. Les différences entre les P_{value} calculées consécutivement donneront la bonne valeur de m.

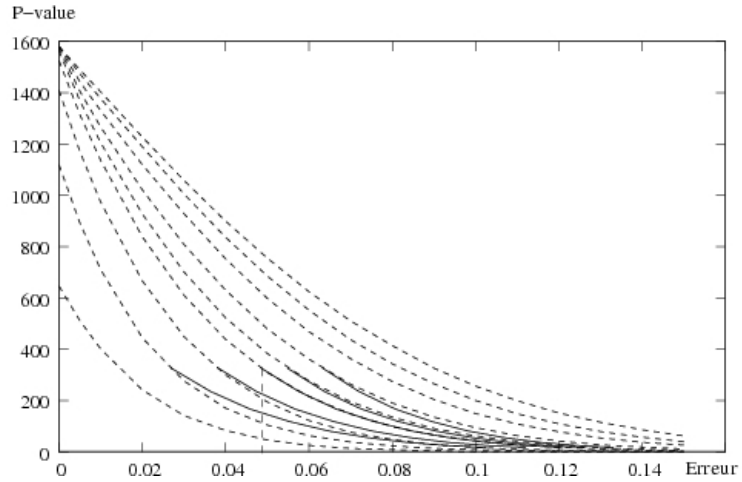
Remarque 16 *Remarquons qu'ajouter un motif d'erreur de probabilité p_1 à un motif d'erreur de probabilité p_0 est équivalent à un motif d'erreur de probabilité*

$$p_0 + p_1 - 2p_0p_1.$$

Entrées :
Un canal binaire symétrique $CBS(p)$.
S une séquence de N bits codée par un code convolutif de paramètres (n, k, m) bruitée.
Paramètre :
$step_p$.
Algorithme :
Transformer S en S' par *Burrows-Wheeler*.
Calculer la P_{value} de S'.
$list := \{[M, P] \quad | \quad (P, P_{value})$ est un point de la courbe correspondant à $m\}$;
$i := 0$;
Tant que $(\#list \neq 1)$ faire
$i := i + 1$;
$T := S$ avec de l'erreur ajoutée de probabilité $i * step_p$;
Transformer T en T' par *Burrows-Wheeler* ;
Calculer la P_{value} de T' ;
Pour $[M, P]$ dans $list$ faire
Si $(P + i * step_p - 2 * P * i * step_p, P_{value})$ n'est pas un point de la courbe correspondant à M alors
Ôter $[M, P]$ de $list$;
Fin si
Fin pour
Fin tant que
Sortie :
$list = \{[m, p]\}$.

FIG. 4.5 – Algorithme pour retrouver m et p quand k et n sont connus.

Exemple 22 *Ici,* $step_p = 0.005$.



- $i = 0$:

$$list = \{[5, 0.061], [6, 0.054], [7, 0.049], [8, 0.039], [9, 0.026]\}.$$

- $i = 1$:

$$list = \{[5, 0.061], [6, 0.054], [7, 0.049]\}.$$

- $i = 2$:

$$list = \{[7, 0.049]\}.$$

Ainsi,

$$m = 7 \ et \ p = 0.049.$$

Quand rien n'est connu

Comme montré dans la Figure 4.6, toutes les courbes sont différentes quand k et n varient. Il suffit donc d'appliquer l'algorithme 4.5 à un plus grand panel de courbes possibles.

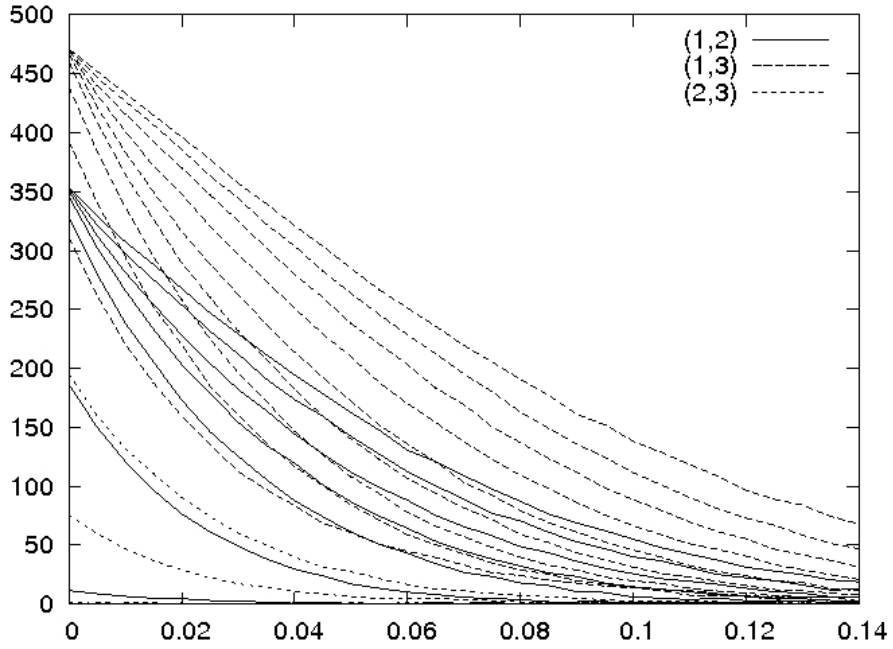


FIG. 4.6 – Courbes pour $N = 10^6$.

4.2.6 Analyse de complexité

Cet algorithme de détection est fait de deux parties : le réarrangement de la séquence et le calcul de la P_{value}. Le temps de calcul de la seconde partie est négligeable devant la première. Ainsi, nous considérons la complexité de l'algorithme de réarrangement. Plus précisément, nous avons à trier N vecteurs de bits, et le plus efficaces des algorithmes de tri connus le fait avec une complexité en $\mathcal{O}(N\log(N))$. Par conséquent, la complexité est constante (N est considéré comme un paramètre de l'algorithme) et ne dépend pas de la séquence que l'on considère. Cependant, le paramètre N doit être choisi en fonction des paramètres k, n, m, p que nous voulons être capables de détecter.

Un code convolutif peut être vu comme une application linéaire avec une matrice infinie

$$\begin{pmatrix} p_{1,0} & \cdots & p_{1,k} & \cdots & p_{1,km-1} & 0 & 0 & \cdots \\ \vdots & & & & \vdots & 0 & 0 & \cdots \\ p_{n,0} & \cdots & p_{n,k} & \cdots & p_{n,km-1} & 0 & 0 & \cdots \\ 0 & 0 & p_{1,0} & \cdots & \cdots & p_{1,km-1} & 0 & \cdots \\ 0 & 0 & \vdots & & & \vdots & 0 & \cdots \\ 0 & 0 & p_{n,0} & \cdots & \cdots & p_{n,km-1} & 0 & \cdots \\ & & \ddots & & \ddots & & & \ddots \end{pmatrix}.$$

Mais, si on ne considère ce code convolutif que durant une période de q tops d'horloge, il peut être vu comme une application linéaire de longueur nq et de dimension $k(m+q)$ (le m provient du fait que l'état interne du registre doit être rempli) de matrice

$$\begin{pmatrix}
p_{1,0} & \cdots & p_{1,k} & \cdots & p_{1,km} & 0 & \cdots & 0 \\
\vdots & & & & \vdots & \vdots & & \vdots \\
p_{n,0} & \cdots & p_{n,k} & \cdots & p_{n,km} & 0 & \cdots & 0 \\
0 & 0 & p_{1,0} & p_{1,1} & \cdots & p_{1,km} & 0 & 0 \\
\vdots & \vdots & \vdots & & & \vdots & \vdots & \vdots \\
0 & 0 & p_{n,0} & p_{n,1} & \cdots & p_{n,km} & 0 & 0 \\
 & \ddots & & \ddots & & & \ddots & \\
0 & \cdots & \cdots & 0 & p_{1,0} & p_{1,1} & \cdots & p_{1,km} \\
\vdots & & & \vdots & \vdots & & & \vdots \\
0 & \cdots & \cdots & 0 & p_{n,0} & p_{n,1} & \cdots & p_{n,km}
\end{pmatrix}.$$

Si on note C_q cette application et si on suppose que tous les polynômes générant ce code sont premiers entre-eux, alors le rang de la matrice est :

$$Rang(C_q) = \begin{cases} nq & \text{si } q \leq \frac{km}{n-k} \\ k(q+m) & \text{si } q \geq \frac{km}{n-k} \end{cases}$$

Cette caractéristique est bien connue pour les codes convolutifs et est appelée *Saut de rang*. En particulier, si $q \geq \frac{km}{n-k}$, cette application n'est plus surjective. Par conséquent, tous les motifs de taille au moins $\frac{knm}{n-k}$ n'apparaîtront pas dans une séquence générée par un tel code convolutif. Si on veut utiliser ce fait, on doit prendre une séquence de longueur en $O(2^{\frac{knm}{n-k}})$. Notons $L_{min} = 2^{\frac{knm}{n-k}}$ la longueur minimale de la séquence à considérer.

Ainsi, l'algorithme de détection a pour complexité $O(\frac{knm}{n-k}2^{\frac{knm}{n-k}})$.

En pratique, avec une séquence de longueur $10^6 \simeq 2^{20}$, on peut détecter des codes convolutifs avec $k = 1, n = 2$ et m jusqu'à 10 ($\frac{knm}{n-k} = 2m$).

4.3 Application à la reconstruction

Rappelons que l'idée principale de cette étude était d'améliorer la reconstruction de codes convolutifs en effectuant une détection des paramètres préalablement. Les algorithmes de reconstruction demandent une faible longueur de séquence, cependant il faut faire une recherche exhaustive sur les paramètres du code. Par contre, pour l'algorithme de détection des paramètres, il est nécessaire d'avoir une séquence relativement longue. Ainsi, nous nous intéressons ici au compromis possible entre ces deux méthodes afin d'optimiser la complexité de la reconstruction.

Les limites de l'algorithme de reconstruction apparaissent pour un taux d'erreur du canal supérieur à 2.10^{-2} alors que l'algorithme de détection fonctionne pour des taux d'erreur jusqu'à 10^{-1}. Ainsi, pour des taux d'erreur supérieurs à 2.10^{-2}, on peut seulement retrouver les paramètres du code.
Pour des taux d'erreur inférieurs à 2.10^{-2}, l'algorithme de reconstruction permet de retrouver les paramètres ainsi qu'une matrice canonique en des temps inférieurs à celui de l'algorithme de détection. Cependant, si p est inconnue, l'algorithme de détection permet de la retrouver. Ces remarques sont répertoriées dans le schéma suivant.

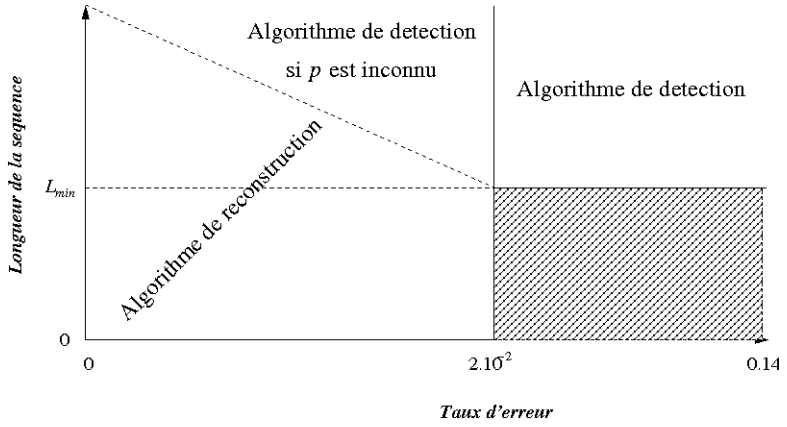


FIG. 4.7 – Complémentarité des algorithmes de détection et de reconstruction

Deuxième partie

Structure des codes quasi-cycliques

Chapitre 5

Introduction

Dans cette partie, nous nous intéressons à la structure des codes quasi-cycliques. Ces codes sont utilisés en particulier en cryptographie car ils permettent d'utiliser des clés plus petites. Ce genre d'étude permet donc une meilleure compréhension de cette famille de codes dans le but d'améliorer la sécurité de tels cryptosystèmes ou encore de construire des algorithmes de reconstruction dédiés.

5.1 Codes quasi-cycliques

Les codes quasi-cycliques étant une généralisation des codes cycliques, il est essentiel de définir un généralisation du *shift*, le *quasi-shift*.

Définition 34 *Soit $n = m\ell$. Une permutation $\sigma \in \mathscr{S}_n$ est un* ℓ-quasi-shift *si elle est le produit disjoint de ℓ cycles d'ordre m.*

Exemple 23
• *Si on considère T le décalage circulaire (utilisé dans la définition des codes cycliques), alors $\sigma = T^\ell \in \mathscr{S}_n$ est un ℓ-quasi-shift.*
• *Prenons $n = 6$, $\ell = 2$. Alors, la permutation*

$$\sigma = (1, 3, 6)(2, 5, 4)$$

est un 2-quasi-shift.

Nous pouvons ainsi donner la définition générale d'un code quasi-cyclique.

Définition 35 *Soient $n = m\ell$ et C un code de longueur n.*

C est un code ℓ-quasi-cyclique ***si et seulement si*** *$Aut(C)$ contient un ℓ-quasi-shift.*

*Le ℓ minimal tel que C est un code ℓ-quasi-cyclique est appelé l'*indice *du code C.*

Notons que dans le cas trivial où $\ell = 1$, on retombe exactement sur la définition d'un code cyclique.

5.2 Représentations des codes quasi-cycliques

La définition générale des codes quasi-cycliques n'impose que la présence d'un quasi-shift dans le groupe d'automorphisme du code. Cependant, il existe deux quasi-shifts particuliers qui donnent les représentations suivantes.

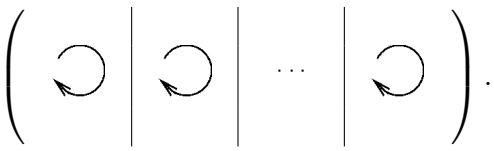
5.2.1 Comme concaténation de codes cycliques

Dans cette section, nous voyons les codes quasi-cycliques comme des concaténation de plusieurs codes cycliques. Nous utilisons le quasi-shift particulier suivant :

$$\tau_1 = (1, 2, \ldots, m)(m+1, m+2, \ldots, 2m) \cdots ((\ell-1)m+1, (\ell-1)m+2, \ldots, \ell m).$$

Et ainsi, un code C est dit quasi-cyclique **si et seulement si** $\tau_1 \in Aut(C)$.

Une matrice génératrice d'un tel code peut donc être mise sous la forme de concaténation de blocs circulants



Cette représentation est utilisée par *G. Skersys* [Ske97, Ske99], *P. Fitzpatrick* et *K. Lally* [FL01] et *N. Nardeau* et *W. Benmoussa* [NB07]. Elle permet une approche polynomiale et de façon similaire aux codes cycliques qui sont des idéaux de l'anneau quotient $\mathbb{F}[X]/(X^n - 1)$, les codes quasi-cycliques sont ici des R-sous-modules de R^ℓ où

$$R = \mathbb{F}[X]/(X^m - 1).$$

On peut ainsi écrire la matrice polynomiale suivante qui représente le code :

$$\begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1\ell} \\ 0 & g_{22} & \cdots & g_{2\ell} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & g_{\ell\ell} \end{pmatrix}$$

où les g_{ij} sont des polynômes engendrant des codes cycliques. De plus les éléments diagonaux g_{ii} divisent $X^m - 1$.

On obtient ainsi une matrice génératrice sous forme canonique :

$$\left(\begin{array}{c|c|c|c}
g_{11} & g_{12} & \cdots & g_{1\ell} \\
Xg_{11} & Xg_{12} & \cdots & Xg_{1\ell} \\
\vdots & \vdots & & \vdots \\
X^{m-\deg(g_{11})-1}g_{11} & X^{m-\deg(g_{11})-1}g_{12} & \cdots & X^{m-\deg(g_{11})-1}g_{1\ell} \\
\hline
0 & g_{22} & \cdots & g_{2\ell} \\
0 & Xg_{22} & \cdots & Xg_{2\ell} \\
\vdots & \vdots & & \vdots \\
0 & X^{m-\deg(g_{22})-1}g_{22} & \cdots & X^{m-\deg(g_{22})-1}g_{2\ell} \\
\hline
 & & \ddots & \\
\hline
0 & 0 & \cdots & g_{\ell\ell} \\
0 & 0 & \cdots & Xg_{\ell\ell} \\
\vdots & \vdots & & \vdots \\
0 & 0 & \cdots & X^{m-\deg(g_{\ell\ell})-1}g_{\ell\ell}
\end{array}\right)$$

où les éléments diagonaux g_{ii} divisent $X^m - 1$ et où les g_{ij}, avec $i < j$ sont réduits modulo g_{jj}. On obtient ainsi une écriture unique.

5.2.2 Comme code cyclique sur un anneau

Dans cette section, nous voyons les codes quasi-cycliques comme codes cycliques sur un anneau. Nous utilisons le quasi-shift particulier suivant :

$$\tau_2 = T^\ell = (1, \ell+1, \ldots, (m-1)\ell+1)(2, \ell+2, \ldots, (m-1)\ell+2)\cdots(\ell, 2\ell, \ldots, m\ell).$$

Et ainsi, un code C est dit quasi-cyclique **si et seulement si** $\tau_2 \in Aut(C)$.

Cette permutation correspond à un décalage circulaire par blocs de taille ℓ et on peut donc écrire une matrice génératrice de façon générale sous la forme :

$$\begin{pmatrix}
A_1 & A_2 & \cdots & A_m \\
A_m & A_1 & \cdots & A_{m-1} \\
 & \ddots & \ddots & \\
A_2 & A_3 & \cdots & A_1
\end{pmatrix} = \begin{pmatrix}
A_1 & A_2 & \cdots & A_m \\
 & & \circlearrowright &
\end{pmatrix}$$

où les A_i sont des matrices de taille $j \times \ell$ avec $j \leq \ell$.

Ici, le code est donc vu comme un code cyclique par blocs. Cette représentation est utilisée par *J. Conan* et *G. Séguin* [CS93] et *S. Ling* et *P. Solé* [LS01a, LS01b]. Par cette approche, ils voient les codes quasi-cycliques comme codes sur l'anneau de polynômes

$$F[Y]/(Y^m - 1).$$

Dans nos travaux et donc la suite de cette thèse, nous utilisons cette représentation. Elle nous permettra de munir les codes d'une structure de M-module à gauche où M est un anneau de polynômes à coefficients matriciels.

5.3 Codes quasi-cycliques en cryptographie

Les codes correcteurs d'erreurs sont apparus pour la première fois en cryptographie en 1978 lorsque *R.J. McEliece* propose son cryptosystème à clef publique basé sur les codes [McE78]. Ce cryptosystème a pour principal inconvénient de nécessiter une clef privée très grande : quelques milliers de bits contre un millier de bits avec RSA pour une sécurité équivalente. Ainsi, cette étude a été mise de côté pendant de nombreuses années. Cependant, un regain d'intérêt pour ce cryptosystème est apparu ces dernières années avec l'intérêt porté sur l'ordinateur quantique. En effet, les cryptosystèmes basés sur les codes feraient partie de la courte liste des cryptosystèmes résistant à ce dernier.

Ce cryptosystème est basé sur le problème de décodage [BMV78, JJ02] de la façon suivante :

Clef privée :
Un code binaire C de paramètres $[n, k, d]$
muni d'un algorithme de décodage Φ_C jusqu'à t erreurs.
G une matrice génératrice de C.
S un matrice inversible de taille $k \times k$.
P une matrice de permutation de taille $n \times n$.
Clef publique :
$G' = SGP$.
Chiffrement :
$x \longmapsto y = xG' + e$ où e est un mot d'erreur aléatoire de poids t.
Déchiffrement :
$y \longmapsto \Phi_C(yP^{-1})S^{-1}$.

Fig. 5.1 – Cryptosystème de *McEliece*

La famille de codes proposée par *McEliece* est la famille des codes de *Goppa* puisqu'il existe un algorithme de décodage efficace et que leur distance minimale est relativement grande. De plus les codes de *Goppa* sont indistinguables des codes aléatoires. Par cela, on entend qu'il est calculatoirement difficile de distinguer un code de *Goppa* d'un code aléatoire.

Récemment, en 2005, *P. Gaborit* [Gab05] propose d'utiliser des codes quasi-cycliques dans le système de *McEliece* afin de réduire la taille des clés. En effet, on ne stocke que les premières lignes d'une matrice génératrice, les suivantes étant obtenues par application du quasi-shift.

Si

$$G = \begin{pmatrix} A_1 & A_2 & \cdots & A_m \\ & \circlearrowleft & & \end{pmatrix}$$

est une matrice génératrice d'un code quasi-cyclique, cette matrice est entièrement déterminée par les sous matrices $A_1, \ldots, A_m$ de taille $r \times \ell$ avec $r \leq \ell$. Il applique ensuite une permutation π sur chaque bloc de taille ℓ. On obtient ainsi une matrice de la forme

$$G' = \begin{pmatrix} A_1\pi & A_2\pi & \cdots & A_m\pi \\ & \circlearrowleft & & \end{pmatrix}.$$

Cette matrice constituera la donnée publique. La clé publique peut ainsi être réduite aux r premières lignes de cette matrice.

Cependant, la petite taille de la permutation π est une faiblesse de cette idée. Elle a été utilisée dans [OTD09] pour élaborer une attaque permettant de retrouver le code initial.

Une nouvelle version utilisant des sous-codes de codes alternants sur des sous-corps est présentée dans [BCGO09].

Chapitre 6

Codes cycliques sur des anneaux de matrices

Dans cette partie, nous proposons la construction d'une sous-famille de codes quasi-cycliques. Ces travaux sont soumis au journal Finite Fields and Applications [CCN09]. Ils sont basés sur les travaux de *M. Camus* [Cam04] et *N. Nardeau, W. Benmoussa* [NB07]. Nous utilisons la représentation de la section 5.2.2. :

Soit $n = m\ell$ un entier naturel. On considère la permutation cyclique sur les colonnes $T \in \mathscr{S}_n$ définie par :

$$T((c_0, c_1, \ldots, c_{n-1})) = (c_{n-1}, c_1, \ldots, c_{n-2}).$$

On définit ensuite la permutation $\tau \in \mathscr{S}_n$ par :

$$\tau = T^{\ell}.$$

On utilise donc la définition de codes quasi-cycliques suivante.

Définition 36 *Un code C de longueur $n = m\ell$ est dit ℓ-quasi-cyclique si :*

$$\textit{Pour tout } c \in C, \quad \tau(c) \in C.$$

En d'autres mots, C est laissé stable par la permutation sur les colonnes τ.

Soit q une puissance d'un nombre premier, on définit $\mathbb{A} = \mathbb{F}_q^{\ell}$. Pour $v = (v_0, \ldots, v_{\ell-1})$ un vecteur ligne de $\mathbb{A}$, on note ${}^t(v_0, \ldots, v_{\ell-1})$ le vecteur colonne correspondant.

On considère l'isomorphisme de $\mathbb{F}_q$-espaces vectoriels de $\mathbb{F}_q^n$ dans $\mathbb{A}^m$ donné par :

$$\forall c = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n, \quad \Theta(c) = ({}^t(c_0, \ldots, c_{\ell-1}), {}^t(c_{\ell}, \ldots, c_{2\ell-1}), \ldots, {}^t(c_{(m-1)\ell}, \ldots, c_{m\ell-1})).$$

On étend la permutation circulaire T à $\mathbb{A}^m$:

$$\forall v = ({}^t v_0, {}^t v_1, \ldots, {}^t v_{m-1}) \in \mathbb{A}^m, \quad T(v) = ({}^t v_{m-1}, {}^t v_0, \ldots, {}^t v_{m-2}).$$

Grâce à cette formalisation, si C est un code de longueur $n = m\ell$ sur $\mathbb{F}_q$,

$$C \text{ est un code } \ell\text{-quasi-cyclique sur } \mathbb{F}_q \Longleftrightarrow \Theta(C) \text{ est un code cyclique sur } \mathbb{A}.$$

Comme $\Theta(C)$ est composé de mots de *vecteurs*, le formalisme matriciel est une interprétation naturelle. Dans cette partie, nous ferons agir des polynômes à coefficients matriciels sur les éléments de $\mathbb{A}^m$. Cette action permettra de construire de nouvelles familles de codes quasi-cycliques *annulés* par des polynômes.

Dans la suite, par abus de langage, on notera toujours C le code $\Theta(C)$.

6.1 Suites récurrentes linéaires à coefficients matriciels

On note $S(\mathbb{A}) = \mathbb{A}^{\mathbb{N}}$ l'ensemble des suites d'éléments de $\mathbb{A}$.

Soit $v = (v_n)_{n\geq 0}$ un élément de $S(\mathbb{A})$. On définit la séquence translatée $T(v)$ par $T(v) = (v_{n+1})_{n\geq 0}$.

Soit A une matrice carrée de taille ℓ sur $\mathbb{F}_q$. On définit la multiplication externe :

$$\begin{array}{ccccc} \mathbb{M}_\ell(\mathbb{F}_q) & \times & S(\mathbb{A}) & \longrightarrow & S(\mathbb{A}) \\ (A & , & (v_n)_{n\geq 0}) & \longmapsto & (Av_n)_{n\geq 0} \end{array}$$

On peut ainsi définir l'action des polynômes à coefficients matriciels $\mathbb{M}_\ell(\mathbb{F}_q)[X]$ sur les suites de $\mathbb{A}$ de la façon suivante :

$$\begin{array}{ccccc} \mathbb{M}_\ell(\mathbb{F}_q)[X] & \times & S(\mathbb{A}) & \longrightarrow & S(\mathbb{A}) \\ (P(X) & , & (v_n)_{n\geq 0}) & \longmapsto & P(X)v = P(T)v \end{array}$$

Exemple 24 *Soient $\ell = 1$ et $q = 2$. Considérons $\mathbb{M}_2(F_2)[X]$.*
Soit v la suite périodique $v = (\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ldots)$.
Et soit P le polynôme $P(X) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$.
Ainsi,

$$
\begin{aligned}
P(X)v &= P(T)v = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} Tv + \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} v \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} (\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ldots) \\
&\quad + \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} (\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ldots) \\
&= (\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ldots) \\
&\quad + (\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ldots) \\
&= (\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ldots) \\
&\quad + (\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \ldots) \\
&= (\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ldots).
\end{aligned}
$$

Proposition 17 *Avec les notations ci-dessus, $S(\mathbb{A})$ muni des opérations usuelles sur les suites et la multiplication à gauche par un polynôme à coefficients matriciels est un $\mathbb{M}_\ell(\mathbb{F}_q)[X]$-module à gauche.*

Remarque 17 *À partir de maintenant, tous les ensembles quotients que nous considérons sont des quotients à gauche.*

Définition 37 *Soit $v \in S(\mathbb{A})$. On définit l'annulateur de v par :*

$$Ann(v) = \{P \in (\mathbb{F}_q)[X] \quad | \quad Pv = (0)\}.$$

Définition 38 *Soit $v \in S(\mathbb{A})$. On dit que v est une suite récurrente linéaire (à coefficients matriciels) si son annulateur $Ann(v)$ contient un polynôme unitaire.*

Remarque 18

• *Il a été montré dans [Nec99] et [Nie95] qu'une suite sur $\mathbb{A}$ est récurrente linéaire si et seulement si les suite des composantes scalaires sont récurrentes linéaires. Le calcul de leurs périodes permet de trouver la période de la suite de vecteurs. Cependant, ici, nous donnerons une démonstration directe de ce résultat via le calcul de l'exposant d'un polynôme matriciel.*

• *L'annulateur d'une suite récurrente linéaire n'est pas forcément un idéal principal (sauf dans le cas classique $\ell = 1$). Néanmoins nous saurons dire quelque chose dans certains cas.*

Définition 39 *On dit qu'un polynôme $P \in \mathbb{M}_\ell(\mathbb{F}_q)[X]$ est réversible si son coefficient de tête et son terme constant sont des matrices inversibles.*

Exemple 25 *Dans* $M_2(\mathbb{F}_2)[X]$,
- $P(X) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ *est réversible.*
- $Q(X) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} X + \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ *ne l'est pas.*

Proposition 18 *Soit* $P \in \mathbb{M}_\ell(\mathbb{F}_q)[X]$. *On suppose que* P *est réversible. Alors, il existe* $e \in \mathbb{N}^*$ *tel que*

$$P \text{ divise } (X^e - 1).$$

Preuve :
On peut supposer sans perte de généralité que P est unitaire.

Considérons la séquence $(X^n)_{n\in\mathbb{N}}$. L'ensemble quotient $\mathbb{M}_\ell(\mathbb{F}_q)[X]/\langle P\rangle$ a une structure particulière. En effet, on le considère à la fois module et $\mathbb{F}_q$-espace vectoriel. De plus, c'est un $\mathbb{F}_q$-espace vectoriel de dimension finie. Ainsi, il existe deux entiers s et $t \in \mathbb{N}^*$ tels que $X^t = X^s \bmod P$. Par conséquent, P divise $X^t - X^s$. Si on suppose que $t > s$, alors P divise $X^s(X^{t-s} - 1)$.

Montrons que :

$$P \text{ divise } X^s(X^{t-s} - 1) \Longrightarrow P \text{ divise } (X^{t-s} - 1).$$

Supposons que $X^s - X^t = q(X)P(X)$ avec P réversible et $s < t$. Tout d'abord, si $s = 0$, il n'y a rien à démontrer. Supposons alors que $s \geq 1$. Alors :

$$X^s(1 - X^{t-s}) = q(X)P(X) \tag{6.1}$$

Ensuite, la division Euclidienne de $(1 - X^{t-s})$ par $P(X)$, qui est possible puisque P est unitaire, est :

$$(1 - X^{t-s}) = b(X)P(X) + r(X) \text{ avec } \deg(r) < \deg(P).$$

D'où

$$X^s(1 - X^{t-s}) = X^s b(X)P(X) + X^s r(X) \tag{6.2}$$

En évaluant cette expression en $X = 0$, on trouve $q(0)P(0) = 0$. Et puisque $P(0)$ est inversible, $q(0) = 0$.

Par conséquent, $q(X) = Xq'(X)$. Comme $s \geq 1$, on a :

$$X([X^{s-1}b(X) - q'(X)]P(X) + X^{s-1}r(X)) = 0.$$

D'où l'égalité :

$$[X^{s-1}b(X) - q'(X)]P(X) + X^{s-1}r(X) = 0.$$

Si $s = 1$, alors le résultat est démontré : $(1 - X^{t-s}) = q'(X)P(X)$.
Sinon, on réitère le raisonnement jusqu'à ce que $q(X) = X^s h(X)$. Ainsi, en mettant X^s en facteur, on obtient :

$$[b(X) - h(X)]P(X) + r(X) = 0.$$

Si $b(X) \neq h(X)$, alors on peut voir que $\deg([b(X) - h(X)]P(X)) \geq \deg(P)$ puisque P est unitaire (ainsi le terme de tête ne peut être annulé). Or $\deg(r) < \deg(P)$, ce qui est impossible.

Par conséquent, $b(X) = h(X)$ et $r(X) = 0$. Ainsi, on a :

$$P \text{ divise } (X^e - 1).$$

⊡

Définition 40 *L'*exposant *d'un polynôme matriciel réversible P est le plus petit entier non nul e tel que*

$$P \textit{ divise à droite } X^e - 1.$$

Remarque 19 *Remarquons que dans ce cas, P divise aussi à gauche $X^e - 1$.*

Définition 41 *Soit $P \in \mathbb{M}_\ell(\mathbb{F}_q)[X]$. On définit le socle de P comme étant l'ensemble*

$$\Omega(P) := \{u \in S(\mathbb{A}) \quad | Pu = 0\}.$$

Proposition 19 *Soit $P \in \mathbb{M}_\ell(\mathbb{F}_q)[X]$ un polynôme réversible, alors on a :*
- *Il existe un entier $m \in \mathbb{N}$ tel que chaque élément de $\Omega(P)$ est périodique de période m.*
- *L'ensemble $\mathscr{C}(P) = \{(u_0, \ldots, u_{m-1}) \in \mathbb{A}^m \quad | \quad u \in \Omega(P)\}$ est un code cyclique.*
- $\dim_{\mathbb{F}_q} \Omega(P) = \ell \deg(P)$.

Preuve :
Pour le premier point, il suffit de prendre m égal à l'exposant de P.
On vérifie facilement le deuxième et le troisième se montre en exhibant une base. ⊡

Remarque 20
- *L'ensemble $\Omega(P)$ est en fait un code ℓ-quasi-cyclique de longueur $m\ell$ sur $\mathbb{F}_q$.*
- *Le calcul de la période d'une suite récurrente linéaire à coefficients matriciels peut être obtenu, comme dans [Nie95], grâce au calcul du polynôme déterminant de la matrice compagnon de la suite récurrente linéaire.*

On sait, dans le cas scalaire, qu'une suite est récurrente linéaire si et seulement si sa série génératrice est une fraction rationnelle. On obtient un résultat similaire dans le cas matriciel :

Proposition 20 *Soient $u \in \mathbb{A}^{\mathbb{N}}$ et $U(X) \in \mathbb{A}[[X]]$ sa série génératrice. Alors*

$$u \text{ est une suite récurrente linéaire} \iff U(X) \text{ est une fraction rationnelle.}$$

Par fraction rationnelle, on entend :
Il existe $R(X) \in \mathbb{A}[X]$ et $Q(X) \in \mathbb{M}_\ell(\mathbb{F}_q)[X]$ avec $\deg(R) < \deg(Q)$ et $Q(0)$ inversible tels que :

$$Q(X)U(X) = R(X).$$

Une preuve de ce résultat se trouve dans [Cam04].

6.2 Codes quasi-cycliques

6.2.1 Codes quasi-cycliques comme codes cycliques sur un anneau

Soient $n = m\ell$ et $\mathbb{A} = \mathbb{F}_q^\ell$ comme précédemment.

On définit de la même façon que pour les suites l'action des polynômes à coefficients matriciels sur les vecteurs de $\mathbb{A}^m$.

$$\begin{array}{ccccc} \mathbb{M}_\ell(\mathbb{F}_q)[X] & \times & \mathbb{A}^m & \longrightarrow & \mathbb{A}^m \\ (P(X) & , & v = (v_0, \ldots, v_{m-1})) & \longmapsto & P(X)v = P(T)v \end{array} \tag{6.3}$$

où T est cette fois le décalage circulaire :

$$T((v_0, v_1, \ldots, v_{m-1})) = (v_{m-1}, v_0, \ldots, v_{m-2}).$$

Par souci de clarté, on notera $X^m - 1$ le polynôme $I_\ell X^m - I_\ell$ où I_ℓ est la matrice identité de taille ℓ. Soient $\mathscr{I}$ l'idéal bilatère de $\mathbb{M}_\ell(\mathbb{F}_q)[X]$ engendré par $X^m - 1$ et $\mathscr{B}$ l'anneau quotient $\mathbb{M}_\ell(\mathbb{F}_q)[X]/\mathscr{I}$.

Proposition 21 *Avec les opérations usuelles et le produit externe défini en 6.3, le $\mathbb{F}_q$-espace vectoriel $\mathbb{A}^m$ est un $\mathscr{B}$-module à gauche.*

Définition 42 *Soient $F \subset \mathscr{B}$ et $C \subset \mathbb{A}^m$.*
• *L'*annulateur *de C est l'ensemble*

$$Ann(C) = \{P \in \mathscr{B} \quad | \quad \forall c \in C, \quad Pc = 0\}.$$

• *Le* socle *de F est le sous-ensemble de $\mathbb{A}^m$ donné par*

$$\Omega(F) = \{y \in \mathbb{A}^m \quad | \quad \forall P \in F, \quad Py = 0\}.$$

Une question intéressante est de savoir quand les deux égalités sont vraies :

$$Ann(\Omega(F)) = F \text{ et } \Omega(Ann(C)) = C.$$

Quelques réponses à cette question se trouvent dans [Nech95]. Ici, nous nous intéressons au cas où le sous-ensemble F est réduit à un seul élément. Dans cette situation, nous avons de bons résultats. Par exemple, comme dans le cas des suites récurrentes linéaires, on a la proposition suivante.

Proposition 22 *Soient $P \in \mathscr{B}$ et C un $\mathbb{F}_q$-espace vectoriel de $\mathbb{A}^m$. Alors,*
• *L'ensemble $\Omega(P)$ est un $\mathbb{F}_q$-espace vectoriel de dimension $\ell \deg(P)$.*
• *L'ensemble $Ann(C)$ est un idéal à gauche de $\mathscr{B}$.*

La preuve est similaire à celle de la Proposition 19.

Remarque 21
• *Un code cyclique C de longueur m sur $\mathbb{A}$ peut être vu comme un sous-espace vectoriel particulier de $S(\mathbb{A})$:*
À $c = (c_0, \ldots, c_{m-1}) \in C$, on associe $u \in S(\mathbb{A})$ tel que :

$$\forall i \in \mathbb{N}, \quad u_i = c_{i \bmod m}.$$

• *L'idéal $Ann(C)$ n'est pas forcément principal.*

Bien que $Ann(C)$ ne soit pas généralement principal, on a le résultat suivant :

Proposition 23 *Soit P un polynôme réversible de $\mathscr{B}$. Supposons qu'il existe Q un polynôme réversible de $\mathscr{B}$ tel que $X^m - 1 = PQ$. Alors, $Ann(\Omega(P))$ est un idéal principal de $\mathscr{B}$ et est exactement l'idéal engendré par P :*

$$Ann(\Omega(P)) = \langle P \rangle.$$

Preuve :
La preuve est basée sur le fait que $Ann(\Omega(P))$ ne peut contenir de polynômes de degré inférieur au degré de P. ⊡

6.2.2 Matrice génératrice de $\Omega(P)$-codes

Dans cette section, nous ne considérons que des factorisations de $X^m - 1$ de la forme $X^m - 1 = PQ$ avec P et Q des polynômes réversibles.

Le premier résultat est analogue à un résultat du cas cyclique. En effet, dans le cas cyclique, si $X^n - 1 = PQ$, cela implique que $\langle P \rangle$ est l'idéal annulé par Q. Cela signifie que pour tout R, $\quad R \in \langle P \rangle \Longleftrightarrow QR = 0$.

Proposition 24 *Soient P et Q deux polynômes réversibles de $\mathscr{B}$ tels que $X^m - 1 = PQ$. Alors,*

$$\Omega(P) = Q\mathbb{A}^m = \{Qx \quad | \quad x \in \mathbb{A}^m\}.$$

Preuve :
• Soit $y \in Q\mathbb{A}^m$. Alors, il existe $y_0 \in \mathbb{A}^m$ tel que $y = Qy_0$ et ainsi $Py = PQy_0 = (X^m - 1)y_0$. D'où $y \in \Omega(P)$. Finalement,

$$Q\mathbb{A}^m \subseteq \Omega(P).$$

• Considérons l'application :

$$\begin{array}{rccc} \Phi_Q : & \mathbb{A}^m & \longrightarrow & \mathbb{A}^m \\ & y & \longmapsto & Qy \end{array}$$

Φ_Q est un morphisme de $\mathbb{F}_q$-espaces vectoriels.

$\ker(\Phi_Q) = \Omega(Q)$ et $im(\Phi_Q) = Q\mathbb{A}^m$, ainsi

$$\begin{array}{rcl} \dim(Q\mathbb{A}^m) & = & \ell m - \dim(\Omega(Q)) \\ & = & \ell m - \ell \deg(Q) \\ & = & \ell m - \ell(m - \deg(P)) \\ & = & \ell \deg(P) \\ & = & \dim(\Omega(P)). \end{array}$$

Puisque $Q\mathbb{A}^m \subseteq \Omega(P)$ et $\dim(Q\mathbb{A}^m) = \dim(\Omega(P))$, on en déduit que :

$$\Omega(P) = Q\mathbb{A}^m.$$

⊡

Avec les mêmes notations, on a le corollaire suivant.

Corollaire 4 *Une matrice génératrice de $\Omega(P)$ est*

$$G_{\Omega(P)} = \begin{pmatrix} {}^tq_0 & {}^tq_1 & {}^tq_2 & \cdots & {}^tq_{\deg(Q)} & 0 & 0 & \cdots & 0 \\ 0 & {}^tq_0 & {}^tq_1 & \cdots & {}^tq_{\deg(Q)-1} & {}^tq_{\deg(Q)} & 0 & \cdots & 0 \\ & & \ddots & & & & \ddots & & \end{pmatrix}.$$

Preuve :
Si $X^m - 1 = PQ$ avec P et Q deux polynômes réversibles, d'après la Proposition 24, on a $\Omega(P) = Q\mathbb{A}^m$.

Soient $c_{i,j} \in \mathbb{A}^m, \quad i \in \{0, \ldots, \deg(P)-1\}, \quad j \in \{0, \ldots, \ell-1\}$ les vecteurs $(0, \ldots, 0, 1, 0, \ldots, 0)$ où les '1' sont en position $i\ell + j$.

Ainsi, tous les $Qc_{i,j}$ sont des mots de code de $\Omega(P)$. Or $Qc_{i,j}$ est le mot de code

$$(\underbrace{0, \ldots, 0}_{i\ell \text{ zéros}}, col_j(q_0), col_j(q_1), \ldots, col_j(q_{\deg(Q)}), 0, \ldots, 0)$$

où $col_j(A)$ désigne la $j^{\text{ème}}$ colonne de la matrice A.

Les mots de code $Qc_{0,0}, \ldots, Qc_{0,\ell-1}$ sont linéairement indépendants puisque q_0 est inversible. Ainsi, tous les $Qc_{i,j}$ sont linéairement indépendants.

Finalement, puisque $\dim(\Omega(P)) = \ell \deg(P), \quad \Omega(P)$ est engendré par tous ces $Qc_{i,j}$. ⊡

6.2.3 Construction de $\Omega(P)$-codes généraux

La partie la plus difficile dans la construction de tels codes est la factorisation de $X^m - 1$ en polynômes à coefficients matriciels. Pour la construction de codes généraux, nous n'avons de méthode plus efficaces que des méthodes naïves pour effectuer la factorisation.

Avec longueur prescrite

Dans ce cas, on utilise la méthode la plus naïve pour factoriser $X^m - 1$, la recherche exhaustive.

Ici, on souhaite prescrire la longueur, on donne ainsi en paramètres de l'algorithme ℓ et m (puisque $n = \ell m$). Le paramètre $\deg_{max}$ permet de borner la recherche exhaustive : on ne cherche que des facteurs de $X^m - 1$ de degré inférieur à $\deg max$. Rappelons qu'on $\Omega(P)$-code a pour dimension $\ell \deg(P)$.

Entrées :
$\ell, m, \deg_{max}$.

Algorithme :
$list_{poly} \longleftarrow [\ ]$;
Pour d allant de 1 à $\deg_{max}$
Pour $P \in \mathbb{M}_\ell(\mathbb{F}_q)[X]$ unitaire de degré d
Si P divise $X^m - 1$ alors
Ajouter P dans $list_{poly}$;
Fin Pour
Fin Pour

Sortie :
$list_{poly}$.

Fig. 6.1 – Algorithme de construction de $\Omega(P)$-codes avec longueur prescrite.

Pour un degré d donné, il y a $q^{d\ell^2}$ polynômes unitaires de degré d. Il est donc facile de comprendre que cette méthode atteint vite ses limites quand $n = m\ell$ grandit.

Avec dimension prescrite

Ici, on souhaite prescrire la dimension du code, on donne ainsi en paramètres ℓ et le degré deg des polynômes P que nous considérons. En effet, la dimension d'un tel code est $k = \ell \deg(P)$.

Pour déterminer la longueur de tels codes, on calcule l'exposant m du polynôme P que l'on considère. Rappelons que si m est l'exposant de P, alors P divise $X^m - 1$. On peut ainsi construire un code de longueur ℓm.

Le paramètre nb détermine le nombre de polynômes P (et donc le nombre de codes) que l'on tire.

Entrées :
$\ell, \deg, nb$.
Algorithme :
list $\longleftarrow$ [ ];
Pour i allant de 1 à nb
Choisir au hasard un polynôme réversible $P \in \mathbb{M}_\ell(\mathbb{F}_q)[X]$ de degré deg ;
Calculer m l'exposant de P ;
Ajouter $[P, m]$ à la liste ;
Fin Pour ;
Sortie :
list.

FIG. 6.2 – Algorithme de construction de $\Omega(P)$-codes avec dimension prescrite

À la fin de cet algorithme, chaque élément $[P, m]$ de *list* correspond à un code ℓ-quasi-cyclique de longueur $n = m\ell$ et de dimension $k = \deg \ell$ annulé par P.

Avec cette méthode, on trouve un code à chaque étape. Cependant, on ne peut pas contrôler la longueur du code. De plus, dans la plupart des cas, l'exposant d'un polynôme pris au hasard est très grand ; ce qui donne un code de très grande longueur avec une dimension petite.

Exemple 26 *Soient* $q = 4$ *et* $\ell = 2$. *Soit* $\mathbb{F}_4 = \mathbb{F}_2[\omega]$ *où* $\omega^2 + \omega + 1 = 0$.
• *Soit*
$P(X) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega & \omega^2 \\ 0 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} 0 & \omega \\ 0 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 0 \\ 1 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & \omega^2 \\ \omega & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$.
L'exposant de P *vaut* $m = 255$. *Ainsi* P *divise* $X^{255} - 1$ *et on obtient un code* 2*-quasi-cyclique de paramètres* $[510, 10, 204]$.

• *Soit*
$Q(X) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 0 \\ 1 & \omega \end{pmatrix} X^4 + \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & \omega \\ 1 & 0 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ 1 & 0 \end{pmatrix} X + \begin{pmatrix} \omega & \omega \\ 0 & 1 \end{pmatrix}$.
L'exposant de Q *vaut* $m = 1020$. *Ainsi* P *divise* $X^{1020} - 1$ *et on obtient un code* 2*-quasi-cyclique de paramètres* $[2040, 10, 1020]$.

6.3 Construction de $\Omega(P)$-codes auto-duaux

Dans cette section, nous construisons des codes auto-duaux pour les produits scalaires Euclidiens et Hermitiens.

Définition 43 *Soit $\langle\ ,\ \rangle$ un produit scalaire sur $\mathbb{F}_q^n$ (ici Euclidien ou Hermitien). Soit C un code sur $\mathbb{F}_q$ de longueur n.*
• *Le code dual $C^\perp$ de C est défini par*

$$C^\perp := \{d \in \mathbb{F}_q^n \quad | \quad \text{Pour tout } c \in C, \quad \langle c, d\rangle = 0\}.$$

• *Le code C est dit* auto-dual *s'il est égal à son code dual $C^\perp$.*

$$C = C^\perp.$$

6.3.1 Construction de $\Omega(P)$-codes auto-duaux Euclidiens

Définition 44 *Soient $\mathscr{R}$ un anneau commutatif et $n \in \mathbb{N}^*$ un entier naturel. Le* produit scalaire Euclidien *dans $\mathscr{R}^n$ est défini par :*

$$\text{pour tout } a = (a_1, \ldots, a_n) \text{ et } b = (b_1, \ldots, b_n) \text{ dans } \mathscr{R}^n,$$

$$\langle a, b\rangle_e = \sum_{i=1}^{n} a_i b_i.$$

On note $C^{\perp_e}$ le code dual de C pour la produit scalaire Euclidien.

Théorème 5 *Si P et Q sont deux polynômes réversibles de $\mathbb{M}_\ell(\mathbb{F}_q)[X]$ tels que $X^m - 1 = PQ$, alors*

$$\Omega(P)^{\perp_e} = \Omega({}^tQ^*).$$

Preuve :
(du Théorème 5)

Pour démontrer ce théorème, nous avons tout d'abord besoin de ces deux résultats.

Lemme 1 *Soit $d \in \mathbb{A}^m$. Alors*

$$d \in \Omega(P)^{\perp_e} \Longleftrightarrow \text{Pour tout } y \in \mathbb{A}^m, \quad \sum_{i=0}^{m-1}\sum_{j=0}^{m-1}\sum_{a=0}^{\ell-1}\sum_{b=0}^{\ell-1} q_{a,b}^j y_{i+j,b} d_{i,a} = 0.$$

Preuve :
(du Lemme 1)

Par souci de clarté dans cette démonstration, on note $\langle\ ;\ \rangle_{\mathbb{A}^m}$ (resp. $\langle\ ;\ \rangle_{\mathbb{A}}$) le produit scalaire Euclidien entre deux éléments de $\mathbb{A}^m$ (resp. $\mathbb{A}$) et $*$ le produit entre un polynôme

de $\mathbb{M}_\ell(\mathbb{F}_q)[X]$ et un vecteur de $\mathbb{A}^m$.

$$
\begin{aligned}
d \in \Omega(P)^{\perp_e} &\iff \text{Pour tout } c \in \Omega(P), \quad \langle c; d\rangle_{\mathbb{A}^m} = 0 \\
&\iff \text{Pour tout } y \in \mathbb{A}^m, \quad \langle Q * y; d\rangle_{\mathbb{A}^m} = 0 \\
&\iff \text{Pour tout } y \in \mathbb{A}^m, \quad \sum_{i=0}^{m-1} \langle (Q*y)_i; d_i\rangle_{\mathbb{A}} = 0 \\
&\iff \text{Pour tout } y \in \mathbb{A}^m, \quad \sum_{i=0}^{m-1} \langle \sum_{j=0}^{m-1} q_j.y_{i+j}; d_i\rangle_{\mathbb{A}} = 0 \\
&\iff \text{Pour tout } y \in \mathbb{A}^m, \quad \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} \langle q_j.y_{i+j}; d_i\rangle_{\mathbb{A}} = 0 \\
&\iff \text{Pour tout } y \in \mathbb{A}^m, \quad \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} \langle (\sum_{b=0}^{\ell-1} q_{a,b}^j.y_{i+j,b})_{a=0,\ldots,\ell-1}; (d_{i,a})_{a=0,\ldots,\ell-1}\rangle_{\mathbb{A}} = 0 \\
&\iff \text{Pour tout } y \in \mathbb{A}^m, \quad \sum_{i=0}^{m-1}\sum_{j=0}^{m-1}\sum_{a=0}^{\ell-1}\sum_{b=0}^{\ell-1} q_{a,b}^j y_{i+j,b} d_{i,a} = 0
\end{aligned}
$$

⊡

Lemme 2 *Soit $d \in \mathbb{A}^m$. Alors*

$$d \in \Omega({}^tQ^*) \Leftrightarrow \text{ Pour tout } k = 0,\ldots,m-1, j = 0,\ldots,\ell-1, \quad \sum_{i=0}^{m-1}\sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+k,a} = 0.$$

Preuve :
(du Lemme 2)

Soient $R \in \mathbb{M}_\ell(\mathbb{F}_q)[X]$ et $d \in \mathbb{A}^m$.

$$\begin{aligned} R * d &= \sum_{i=0}^{m-1} r_i X^i * d \\ &= \sum_{i=0}^{m-1} r_i X^i * (d_0, \ldots, d_{m-1}) \\ &= (\sum_{i=0}^{m-1} r_i d_{i+k})_{k=0,\ldots,m-1} \\ &= (\sum_{i=0}^{m-1} (\sum_{a=0}^{\ell-1} r_{j,a}^i d_{i+k,a})_{j=0,\ldots,m-1})_{k=0,\ldots,m-1} \end{aligned}$$

$$\text{Ainsi } R * d = 0 \quad \Leftrightarrow \quad \text{Pour tout } j,k = 0,\ldots,m-1, \quad \sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} r_{j,a}^i d_{i+k,a} = 0$$

Calculons ${}^tQ^* * d$. On a

$$Q(X) = \sum_{i=0}^{m-1} q_i X^i, \quad q_i = (q_{j,a}^i)_{j=0,\ldots,m-1;a=0,\ldots,m-1}$$

$${}^tQ(X) = \sum_{i=0}^{m-1} {}^tq_i X^i, \text{ avec } {}^tq_i = (q_{a,j}^i)_{j=0,\ldots,m-1;a=0,\ldots,m-1}$$

$${}^tQ^*(X) = \sum_{i=0}^{m-1} {}^tq_{\deg(Q)-i} X^i, \text{ avec } {}^tq_{\deg(Q)-i} = (q_{a,j}^{\deg(Q)-i})_{j=0,\ldots,m-1;a=0,\ldots,m-1}$$

d'où,

$$\begin{aligned} {}^tQ^* * d \quad &\Leftrightarrow \quad \text{Pour tout } k = 0,\ldots,m-1, j = 0,\ldots,\ell-1, \quad \sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} q_{a,j}^{\deg(Q)-i} d_{i+k,a} = 0 \\ &\Leftrightarrow \quad \text{Pour tout } k = 0,\ldots,m-1, j = 0,\ldots,\ell-1, \quad \sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+\deg(Q)+k,a} = 0 \\ &\Leftrightarrow \quad \text{Pour tout } k = 0,\ldots,m-1, j = 0,\ldots,\ell-1, \quad \sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+k,a} = 0 \end{aligned}$$

$\boxdot$

Nous avons maintenant tous les outils pour démontrer le Théorème 5

Montrons tout d'abord que $\Omega(P)^{\perp_e} \subset \Omega({}^tQ^*)$.

Soit $d \in \Omega(P)^{\perp_e}$, $\quad \Omega(P)^{\perp_e}$ étant ℓ-quasi-cyclique,

$$\text{Pour tout } k = 0, \ldots, m-1, \quad X^k.d \in \Omega(P)^{\perp_e}.$$

Ainsi, d'après le Lemme 1,

$$\text{Pour tout } y \in \mathbb{A}^m, \ \forall k = 0, \ldots, m-1, \quad \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \sum_{a=0}^{\ell-1} \sum_{b=0}^{\ell-1} q_{a,b}^j y_{i+j,b} d_{i+k,a} = 0. \tag{6.4}$$

Soit $j \in \{0, \ldots, m-1\}$. Pour $y = (e_j; 0; \ldots; 0)$, $(e_j : j^{\text{ème}}$ vecteur de la base canonique de $\mathbb{A} = \mathbb{F}_q^\ell)$.
On a

$$y_{i+j,b} \quad = \quad \delta_b^j \text{ si } i + j = 0 \mod m \text{ et } 0 \text{ sinon}$$

Ainsi, pour ce y, l'équation 6.4 nous donne

$$\text{Pour tout } k = 0, \ldots, m-1, \quad \sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+k,a} = 0$$

Mais ceci est vrai pour un j arbitraire, ainsi

$$\text{Pour tout } k = 0, \ldots, m-1, \text{ et } j = 0, \ldots, \ell-1, \quad \sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+k,a} = 0$$

D'où, d'après le Lemme 2, $d \in \Omega({}^tQ^*)$. Ainsi

$$\Omega(P)^{\perp_e} \subset \Omega({}^tQ^*).$$

Donc

$$\begin{array}{rcll} \dim(\Omega(P)^{\perp_e}) & = & \ell m - \dim(\Omega(P)) & \\ & = & \ell m - \ell \deg(P) & \text{via la Proposition 19} \\ & = & \ell(m - \deg(P)) & \\ & = & \ell \deg(Q) & \end{array}$$

et

$$\begin{array}{rcll} \dim(\Omega({}^tQ^*)) & = & \ell \deg({}^tQ^*) & \text{via la Proposition 19} \\ & = & \ell \deg(Q) & \text{car } Q \text{ est réversible} \\ & = & \dim(\Omega(P)^{\perp_e}) & \end{array}$$

D'où l'égalité.

⊡

Grâce à ce théorème, il va nous être relativement facile de déterminer les $\Omega(P)$-codes auto-duaux Euclidiens.

Dorénavant, m doit être pair : $m = 2m'$. Ainsi, afin de trouver des codes auto-duaux Euclidiens, il faut trouver des polynômes à coefficients matriciels P de degré m' tels que

$$X^m - 1 = P\ {}^tP^*$$

.

• Première méthode :

Mettons en variables les coefficients des matrices-coefficients du polynôme P. Il y en a $\ell^2 m'$. L'équation polynomiale $X^m - 1 = P\ {}^tP^*$ donne $\ell^2 m$ équations scalaires de degré 2. Nous avons donc à résoudre un système de $\ell^2 m$ équations de degré 2 à $\ell^2 m'$ inconnues. Nous utilisons donc les outils classiques de calcul de bases de Groebner.

• Deuxième méthode :

Imposons de plus que $P = {}^tP^*$. Par la même technique, on se retrouve à devoir résoudre un système de $\ell^2 m$ équations de degré 2 à $\ell^2 m'/2$ inconnues. Le nombre d'inconnues a été divisé par 2.

Finalement, cette contrainte ($P = {}^tP^*$) n'est pas gênante, puisque nous avons remarqué expérimentalement que les bons codes (ceux avec une grande distance minimale) se trouvent dans cette famille.

Pour construire la table suivante, nous avons utilisé la seconde méthode. De plus, les bons codes que nous avons trouvé étaient tous dans $\mathbb{F}_4$.

Longueur n	Meilleure borne connue	Meilleure distance minimale trouvée	Nombre de codes	Remarques
8	4	**4**	2	Remarque 22
12	6	4	2	
16	6	**6**	3	
20	8	7	2	
24	8-10	**8**	9	
28	9-11	**9**	2	Remarque 23
32	10-12	**10**	8	
36	10-14	**10**	22	
40	12-16	**12**	8	

FIG. 6.3 – Table des $\Omega(P)$-codes auto-duaux Euclidiens sur $\mathbb{F}_4$ obtenus par notre méthode.

Dans la plupart des cas (distances minimales en gras), la meilleure borne connue est atteinte par au moins un de ces codes (sauf pour $n = 12$ et 20). Cependant, ceci n'est pas surprenant puisque la famille des codes quasi-cyliques est connue pour contenir de bons codes [GZ06].

Remarque 22 *Un de ces* $[8, 4, 4]$*-code a un polynôme annulateur binaire :*

$$P(X) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Ainsi, une matrice génératrice est :

$$G_{\Omega(P)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Remarque 23 *Dans [BU09],* D. Boucher *et* F. Ulmer *ont trouvé 4 codes auto-duaux Euclidiens sur* $\mathbb{F}_4$ *de paramètres* $[28, 14, 9]$*. Ces codes ont été construits à partir d'anneaux de polynômes tordus. Les 2 codes que nous avons trouvés ici sont conjugués mais non équivalents (à permutations près) aux leurs. Ils ne sont pas non plus équivalents au code résidu quadratique étendu.*

Soit $\mathbb{F}_4 = \mathbb{F}_2[\omega]$ *où* $\omega^2 + \omega + 1 = 0$*. Nos codes sont respectivement annulés par :*

$$P_1(X) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^6 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^5 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^4$$
$$+ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ \omega^2 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

et

$$P_2(X) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} 1 & \omega \\ 1 & 1 \end{pmatrix} X^5 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^4 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

La liste complète des codes se trouvent en Annexe.

6.3.2 Construction de $\Omega(P)$-codes auto-duaux Hermitiens

Dans cette section, on ne considère que des codes sur le corps $\mathbb{F}_4$. On note θ l'application de Froebenius sur $\mathbb{F}_4$: $\theta(x) = x^2$.

Définition 45 *Soient $\mathscr{R}$ un anneau commutatif, $n \in \mathbb{N}^*$ un entier naturel et θ un automorphisme de $\mathscr{R}$ d'ordre* 2. *Le* produit scalaire Hermitien *dans $\mathscr{R}^n$ est défini par :*

$$\textit{pour tout } a = (a_1, \ldots, a_n) \textit{ et } b = (b_1, \ldots, b_n) \textit{ dans } \mathscr{R}^n,$$

$$\langle a, b \rangle_h = \sum_{i=1}^{n} a_i \theta(b_i).$$

$$\textit{Si } \mathscr{R} = \mathbb{F}_4, \quad \theta(x) = x^2 \textit{ et } \langle a, b \rangle_h = \sum_{i=1}^{n} a_i b_i^2.$$

Nous avons un résultat similaire à celui du cas Euclidien :

Théorème 6 *Si P et Q sont deux polynômes réversibles de $\mathbb{M}_\ell(\mathbb{F}_q)[X]$ tels que $X^m - 1 = PQ$, alors*

$$\Omega(P)^{\perp_h} = \Omega(\theta({}^tQ^*)).$$

Preuve :
(du Théorème 6)

Pour démontrer ce théorème, nous avons tout d'abord besoin de ces deux résultats.

Lemme 3 *Soit $d \in \mathbb{A}^m$. Alors*

$$d \in \Omega(P)^{\perp_h} \Longleftrightarrow \textit{Pour tout } y \in \mathbb{A}^m, \quad \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \sum_{a=0}^{\ell-1} \sum_{b=0}^{\ell-1} q_{a,b}^j y_{i+j,b} \theta(d_{i,a}) = 0.$$

Preuve :
(du Lemme 3)
La preuve de ce lemme est la même que celle du Lemme 1. Le θ provient du produit scalaire Hermitien. ⊡

Lemme 4 *Soit $d \in \mathbb{A}^m$. Alors*

$$d \in \Omega(\theta({}^tQ^*)) \Leftrightarrow \text{ Pour tout } k = 0, \ldots, m-1, j = 0, \ldots, \ell - 1, \quad \sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} \theta(q_{a,j}^{-i}) d_{i+k,a} = 0.$$

Preuve :
(du Lemme 4)

Dans le Lemme 2, nous avons

$${}^tQ^*(X) = \sum_{i=0}^{m-1} {}^tq_{\deg(Q)-i} X^i, \text{ avec } {}^tq_{\deg(Q)-i} = (q_{a,j}^{\deg(Q)-i})_{j=0,\ldots,\ell-1;a=0,\ldots,m-1}.$$

Ainsi

$$\theta({}^tQ^*)(X) = \sum_{i=0}^{m-1} \theta({}^tq_{\deg(Q)-i}) X^i, \text{ avec } \theta({}^tq_{\deg(Q)-i}) = (\theta(q_{a,j}^{\deg(Q)-i}))_{j=0,\ldots,\ell-1;a=0,\ldots,m-1}.$$

D'où le résultat. ⊡

Nous avons maintenant tous les outils pour démontrer le Théorème 5

Montrons tout d'abord que $\Omega(P)^{\perp_h} \subset \Omega(\theta({}^tQ^*))$.

Soit $d \in \Omega(P)^{\perp_h}$, $\quad \Omega(P)^{\perp_h}$ étant ℓ-quasi-cyclique,

$$\text{Pour tout } k = 0, \ldots, m-1, \quad X^k.d \in \Omega(f)^{\perp_h}.$$

Ainsi, d'après le Lemme 3,

$$\text{Pour tout } y \in \mathbb{A}^m, \ \forall k = 0, \ldots, m-1, \quad \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \sum_{a=0}^{\ell-1} \sum_{b=0}^{\ell-1} q_{a,b}^j y_{i+j,b} \theta(d_{i+k,a}) = 0. \qquad (6.5)$$

Soit $j \in \{0, \ldots, m-1\}$. Pour $y = (e_j; 0; \ldots; 0)$, $(e_j : j^{\text{ème}}$ vecteur de la base canonique de $\mathbb{A} = \mathbb{F}_q^\ell)$.
On a

$$y_{i+j,b} \quad = \quad \delta_b^j \text{ si } i+j=0 \mod m \text{ et } 0 \text{ sinon}$$

Ainsi, pour ce y, l'équation 6.5 nous donne

$$\text{Pour tout } k=0,\ldots,m-1, \quad \sum_{i=0}^{m-1}\sum_{a=0}^{\ell-1} q_{a,j}^{-i}\theta(d_{i+k,a}) = 0$$

Mais ceci est vrai pour un j arbitraire, ainsi

$$\text{Pour tout } k=0,\ldots,m-1, \text{ et } j=0,\ldots,\ell-1, \quad \sum_{i=0}^{m-1}\sum_{a=0}^{\ell-1} q_{a,j}^{-i}\theta(d_{i+k,a}) = 0$$

$$\text{Pour tout } k=0,\ldots,m-1, \text{ et } j=0,\ldots,\ell-1, \quad \theta(\sum_{i=0}^{m-1}\sum_{a=0}^{\ell-1} q_{a,j}^{-i}\theta(d_{i+k,a})) = 0$$

$$\text{Pour tout } k=0,\ldots,m-1, \text{ et } j=0,\ldots,\ell-1, \quad \sum_{i=0}^{m-1}\sum_{a=0}^{\ell-1} \theta(q_{a,j}^{-i})\theta^2(d_{i+k,a}) = 0$$

$$\text{Pour tout } k=0,\ldots,m-1, \text{ et } j=0,\ldots,\ell-1, \quad \sum_{i=0}^{m-1}\sum_{a=0}^{\ell-1} \theta(q_{a,j}^{-i})d_{i+k,a} = 0 \quad \text{car } \theta^2 = Id.$$

D'où, d'après le Lemme 4, $d \in \Omega(\theta({}^tQ^*))$. Ainsi

$$\Omega(P)^{\perp_h} \subset \Omega(\theta({}^tQ^*)).$$

Donc

$$\begin{array}{rcll} \dim(\Omega(P)^{\perp_h}) & = & \ell m - \dim(\Omega(P)) & \\ & = & \ell m - \ell \deg(P) & \text{via la Proposition 19} \\ & = & \ell(m-\deg(P)) & \\ & = & \ell \deg(Q) & \end{array}$$

et

$$\begin{array}{rcll} \dim(\Omega(\theta({}^tQ^*))) & = & \ell \deg(\theta({}^tQ^*)) & \text{via la Proposition 19} \\ & = & \ell \deg(Q) & \text{car } Q \text{ est réversible} \\ & = & \dim(\Omega(P)^{\perp_h}) & \end{array}$$

D'où l'égalité.

⊡

Comme dans le cas Euclidien, m doit être pair : $m = 2m'$. Et ici, afin de trouver des codes auto-duaux Hermitiens, il faut trouver des polynômes à coefficients matriciels P de degré m' tels que

$$X^m - 1 = P\theta({}^tP^*).$$

Utilisons la même méthode que dans le cas Euclidien. Mettons en variables les coefficients des matrices-coefficients du polynôme P. Il y en a $\ell^2 m'$. L'équation polynomiale $X^m - 1 = P\theta({}^tP^*)$ donne $\ell^2 m$ équations scalaires de degré 3. Dans le cas Euclidien, les équations obtenues étaient de degré 2, elles sont ici de degré 3 à cause du l'automorphisme $\theta(x) = x^2$. Nous avons donc à résoudre un système de $\ell^2 m$ équations de degré 3 à $\ell^2 m'$ inconnues. Nous utilisons toujours les outils classiques de calcul de bases de Groebner.

Remarque 24 *Cependant, l'astuce du cas Euclidien ($P = {}^tP^*$), qui donnerait ici $P = \theta({}^tP^*)$, n'est pas utilisable. En effet cette contrainte est trop forte et trop peu de codes se trouvent dans cette sous-famille.*

Nous ne pouvons donc pas réduire le nombre de variables de moitié. C'est pour cela que nous ne pouvons atteindre de longueurs de codes aussi grandes que dans le cas Euclidien.

Longueur n	Meilleure borne connue	Meilleure distance minimale trouvée	Nombre de codes
8	4	**4**	1
12	4	**4**	1
16	6	**6**	1
20	8	**8**	1
24	8	**8**	7
28	10	**10**	2

FIG. 6.4 – Table des $\Omega(P)$-codes auto-duaux Hermitiens sur $\mathbb{F}_4$ obtenus par notre méthode.

Pour le cas Hermitien aussi, la meilleure borne connue est atteinte par au moins un de nos codes. Cependant, très peu de codes sont trouvés par cette technique.

La liste complète des codes se trouvent en Annexe.

Chapitre 7

Équivalence des codes quasi-cycliques

Dans cette partie, nous abordons les codes d'un point de vue structurel. En effet, nous nous intéressons à l'équivalence par permutations de codes. D'où la définition :

Définition 46 *Soient C un code linéaire en bloc de longueur n et $\sigma \in \mathscr{S}_n$ une permutation, on définit le code $\sigma(C)$ par*

$$\sigma(C) = \{\sigma(c) = (c_{\sigma^{-1}(1)}, \ldots, c_{\sigma^{-1}(n)}) \quad | \quad c = (c_1, \ldots, c_n) \in C\}.$$

Ainsi, on dit que deux codes C et D de longueur n sont équivalents par permutation s'il existe une permutation $\sigma \in \mathscr{S}_n$ telle que

$$D = \sigma(C).$$

Plus précisément, nous nous intéressons à l'équivalence par permutations de codes d'une même famille. Tout d'abord nous rappellerons les travaux effectués sur l'équivalence des codes cycliques (on parlera de permutations conservant la cyclicité). Ensuite nous détaillerons notre apport sur l'équivalence des codes quasi-cycliques (on parlera ici de permutations conservant la quasi-cyclicité).

7.1 Équivalence des codes cycliques

Définition 47 *On dit que $\sigma \in \mathscr{S}_n$ est une permutation conservant la cyclicité si pour tout code cyclique C de longueur n,*

$$\sigma(C) \text{ est un code cyclique.}$$

Ou encore, si pour tout code C de longueur n, si on note T le décalage circulaire,

$$(T \in Aut(C)) \Longrightarrow (T \in Aut(\sigma(C))).$$

Un sous-groupe des permutations de longueur n, le sous-groupe des multiplicateurs, joue un rôle central dans cette théorie.

Définition 48 *Soient n et a deux entiers premiers entre-eux. On définit le multiplicateur par a dans $\mathscr{S}_n$ par*

$$\begin{array}{rcl} M_a : \mathbb{Z}/n\mathbb{Z} & \longrightarrow & \mathbb{Z}/n\mathbb{Z} \\ i & \longmapsto & ai \end{array}$$

Il est évident que les multiplicateurs sont des permutations conservant la cyclicité, et dans certains cas, ils sont les seuls.

Tout d'abord, *B. Alspach* et *T.D. Parsons* montrent en 1979 dans [Als79] le résultat suivant dans le cas où la longueur est le produit de deux nombres premiers distincts.

Théorème 7 *Soient C et D deux codes cycliques de longueur n. Si de plus*

$$n = pr, \quad p > r \text{ sont premiers}$$
$$\text{et le } p\text{-sous-groupe de Sylow du groupe d'automorphismes de } C \text{ est d'ordre } p,$$

alors, C et D sont équivalents par permutations si et seulement si C et D sont équivalents par un multiplicateur.

Ensuite, en 1987, *P.P. Pálfy* montre dans [Pal87] le résultat suivant.

Théorème 8 *Soient C et D deux codes cycliques de longueur n. Si de plus*

$$pgcd(n, \phi(n)) = 1 \text{ ou } n = 4,$$

alors, C et D sont équivalents par permutations si et seulement si C et D sont équivalents par un multiplicateur.

Dans [HJP93], *W.C. Huffman*, *V. Job* et *V. Pless* regroupent ces résultats existants et tentent une généralisation à des longueurs puissances d'un nombre premier.

Se basant sur les travaux de *N. Brand* [Bra91], ils prouvent le résultat suivant dans le cas $n = p^2$.

Théorème 9 *Soient C et D deux codes cycliques de longueur $n = p^2$ avec p un premier impair. Si C et D sont équivalents par permutations, alors ils sont équivalents par*

– *un multiplicateur*

ou

– *un multiplicateur généralisé fois un multiplicateur.*

Avec la définition de multiplicateur généralisé :

Définition 49 *Soient $n = p^2$ et $d \in (\mathbb{Z}/p\mathbb{Z})^*$. On définit le multiplicateur généralisé $\mu_d \in \mathscr{S}_{p^2}$ par*

$$\begin{array}{rcl} \mu_d : \mathbb{Z}/p^2\mathbb{Z} & \longrightarrow & \mathbb{Z}/p^2\mathbb{Z} \\ k = i + pj & \longmapsto & \mu_d(k) = ((id) \ mod \ p) + pj \end{array}$$

où $0 \leq i, j < p$.

Finalement, ils donnent des propriétés sur les multiplicateurs généralisés dans le cas $n = p^r$ sans donner de résultats similaires sur l'équivalence des codes. Remarquons que cette étude demande de s'y intéresser au cas par cas, et la plupart des démonstrations reposent sur des résultats de théorie des groupes. Il est donc compréhensible que des résultats ont été trouvés seulement dans des cas simples.

Notons aussi que les multiplicateurs (simples ou généralisés) jouent un rôle important dans cette étude et que dans les démonstrations apparaît souvent le sous-groupe $N_{\mathscr{S}_n}(\langle T \rangle)$, le *normalisateur* de T dans $\mathscr{S}_n$.

Dans le but de généraliser ces résultats, nous déterminons dans ce chapitre $N_{\mathscr{S}_n}(\langle \tau \rangle)$ le normalisateur de τ dans $\mathscr{S}_n$ où τ est le quasi-shift.

7.2 Équivalence des codes quasi-cycliques

Dans cette section, nous utilisons la même représentation que pour la construction de $\Omega(P)$-codes. On note toujours T le décalage circulaire.

Définition 50 *Un code C de longueur $n = m\ell$ est dit ℓ-quasi-cyclique si :*

$$Pour \ tout \ c \in C, \quad T^\ell(c) \in C.$$

En d'autres mots, C est laissé stable par la permutation sur les colonnes T^ℓ.

Définition 51 *On dira que $\sigma \in \mathscr{S}_n$ est une permutation conservant la ℓ-quasi-cyclicité si pour tout code quasi-cyclique C de longueur n et d'indice ℓ,*

$$\sigma(C) \ est \ un \ code \ quasi\text{-}cyclique \ d'indice \ \ell.$$

Ou encore, si pour tout code C de longueur n, si on note $\tau = T^\ell$ le décalage circulaire de ℓ positions (quasi-shift),

$$(\tau \in Aut(C)) \Longrightarrow (\tau \in Aut(\sigma(C))).$$

Dans la suite, on ne considérera que des codes quasi-cycliques d'indice ℓ et on notera $\tau = T^\ell$ le décalage circulaire de ℓ positions.

Le but de cette partie est de déterminer le Normalisateur de $\langle\tau\rangle$ dans $\mathscr{S}_n$.

$$\begin{array}{rcl} N_{\mathscr{S}_n}(\langle\tau\rangle) & = \{\varphi \in \mathscr{S}_n & \mid \quad \langle\tau\rangle \circ \varphi = \varphi \circ \langle\tau\rangle\} \\ & = \{\varphi \in \mathscr{S}_n & \mid \quad \exists r, \quad \tau \circ \varphi = \varphi \circ \tau^r\}. \end{array}$$

Notations 2 *Chaque élément i de $\{0, \ldots, n-1\}$ s'écrit de manière unique sous la forme*

$$i = a\ell + b \text{ avec } a \in \{0, \ldots, m-1\} \text{ et } b \in \{0, \ldots, \ell - 1\}.$$

D'où l'isomorphisme

$$\begin{array}{ccc} \mathbb{Z}/n\mathbb{Z} & \xrightarrow{\sim} & (\mathbb{Z}/m\mathbb{Z}) \times (\mathbb{Z}/\ell\mathbb{Z}) \\ a\ell + b & \longmapsto & (a, b) \end{array}$$

Dorénavant nous noterons (a, b) la valeur $a\ell + b$.

Ainsi, tout permutation $\varphi \in \mathscr{S}_n$ peut être notée $\varphi = (\theta, \lambda)$ où

$$\varphi(a, b) = (\theta(a, b), \lambda(a, b))$$

$$(\varphi(a\ell + b) = \theta(a, b)\ell + \lambda(a, b)).$$

7.3 Calcul du normalisateur $N_{\mathscr{S}_n}(\langle\tau\rangle)$.

Proposition 25 *Soit $\varphi \in \mathscr{S}_n$. Alors*

$$(\exists r, \quad \tau \circ \varphi = \varphi \circ \tau^r) \Longleftrightarrow \exists r, \quad \forall a, b, \quad \begin{cases} \theta(a + r, b) = \theta(a, b) + 1 \\ \lambda(a + r, b) = \lambda(a, b). \end{cases}$$

Preuve :
• Supposons qu'il existe r tel que $\tau \circ \varphi = \varphi \circ \tau^r$. Ainsi, pour tout a, b,

$$\begin{array}{rcl} \tau \circ \varphi(a, b) & = & \varphi \circ \tau^r(a, b) \\ \tau(\theta(a, b), \lambda(a, b) & = & \varphi(a + r, b) \\ (\theta(a, b) + 1, \lambda(a, b))) & = & (\theta(a + r, b), \lambda(a + r, b)) \end{array}$$

D'où, pour tout a, b,

$$\begin{cases} \theta(a + r, b) = \theta(a, b) + 1 \\ \lambda(a + r, b) = \lambda(a, b). \end{cases}$$

• Supposons maintenant que $\varphi \in \mathscr{S}_n$ soit telle que pour tout a, b,

$$\begin{cases} \theta(a+r,b) = \theta(a,b) + 1 \\ \lambda(a+r,b) = \lambda(a,b). \end{cases}$$

Soient $a \in \{0,\ldots,m-1\}$ et $b \in \{0,\ldots,\ell-1\}$. Alors

$$\begin{aligned} \varphi \circ \tau^r(a,b) &= \varphi(a+r,b) \\ &= (\theta(a+r,b), \lambda(a+r,b)) \\ &= (\theta(a,b)+1, \lambda(a,b)) \\ &= \tau(\theta(a,b), \lambda(a,b)) \\ &= \tau \circ \varphi(a,b). \end{aligned}$$

D'où l'égalité. ⊡

Proposition 26 *Si $\varphi \in \mathscr{S}_n$ est telle qu'il existe r, $\tau \circ \varphi = \varphi \circ \tau^r$, alors*

φ conserve la ℓ-quasi-cyclicité.

Preuve :
Soient $\varphi \in \mathscr{S}_n$ et r tels que $\tau \circ \varphi = \varphi \circ \tau^r$.
Soit C un code de longueur n tel que $\tau \in Aut(C)$.

Montrons que $\tau \in Aut(\varphi(C))$, c'est-à-dire que pour tout $d \in \varphi(C)$, $\tau(d) \in \varphi(C)$.

Soit $d \in \varphi(C)$. Alors il existe $c \in C$ tel que $d = \varphi(c)$.

Ainsi, $\tau(d) = \tau \circ \varphi(c)$. Or par hypothèse, $\tau \circ \varphi = \varphi \circ \tau^r$, alors $\tau(d) = \varphi \circ \tau^r(c)$.

Mais $\tau \in Aut(C)$, donc $\tau^r(c)$ est un élément de C et donc $\tau(d) = \varphi \circ \tau^r(c)$ est un élément de $\varphi(C)$. ⊡

Une question que l'on peut se poser est la suivante : Est-ce qu'il y a équivalence dans l'énoncé de la Proposition ?
Dans le cas cyclique, il n'y a pas équivalence, et déterminer les éléments n'appartenant pas au normalisateur $N_{\mathscr{S}_n}(\langle\tau\rangle)$ est très difficile. Cependant, dans [HJP93], *W. C. Huffman, V. Job* et *V. Pless* les déterminent dans des cas particuliers ($pgcd(n,\varphi(n)) = 1$, $n = 4$, $n = p$, $n = p^2$, $n = pr$).

Ici nous ne nous intéressons qu'au calcul du normalisateur $N_{\mathscr{S}_n}(\langle\tau\rangle)$.

Proposition 27 *Si $\varphi \in \mathscr{S}_n$ est telle qu'il existe r, $\tau \circ \varphi = \varphi \circ \tau^r$, alors*

ce r est unique (modulo m)
et $pgcd(r, m) = 1$.

Preuve :
• Soient r, s tels que $\tau \circ \varphi = \varphi \circ \tau^r$ et $\tau \circ \varphi = \varphi \circ \tau^s$. Alors

$$\begin{array}{ll} & \varphi \circ \tau^r = \varphi \circ \tau^s \\ \Rightarrow & \varphi = \varphi \circ \tau^{s-r} \\ \Rightarrow & Id = \tau^{s-r} \\ \Rightarrow & s = r \bmod m \quad \text{car } \tau \text{ est d'ordre } m. \end{array}$$

• Supposons que $pgcd(r, m) = t \neq 1$. On a

$$\begin{array}{ll} & \tau \circ \varphi = \varphi \circ \tau^r \\ \Rightarrow & \tau^2 \circ \varphi = \tau \circ (\tau \circ \varphi) = (\tau \circ \varphi) \circ \tau^r = \varphi \circ \tau^r \circ \tau^r = \varphi \circ \tau^{2r} \\ & \vdots \\ \Rightarrow & \tau^{m/t} \circ \varphi = \varphi \circ \tau^{rm/t} = \varphi \circ \tau^{ppcm(r,m)} \\ \Rightarrow & \tau^{m/t} \circ \varphi = \varphi \quad \text{car } m \text{ divise } ppcm(r, m) \\ \Rightarrow & \tau^{m/t} = Id \\ \Rightarrow & m/t = 0 \bmod m. \end{array}$$

Or ceci est impossible car $t \neq 1$. Et ainsi

$$pgcd(r, m) = 1.$$

⊡

On a ainsi la Définition/Proposition suivante :

Proposition 28

$$\begin{array}{ccc} o : N_{\mathscr{S}_n}(\langle\tau\rangle) & \longrightarrow & (\mathbb{Z}/m\mathbb{Z})^* \\ \varphi & \longmapsto & o(\varphi) = r \quad \text{où } \tau \circ \varphi = \varphi \circ \tau^r \end{array}$$

est un morphisme de groupes.

Preuve :
D'après la Proposition 27, $o(\varphi)$ est bien défini puisque qu'il est unique et appartient bien à $(\mathbb{Z}/m\mathbb{Z})^*$.

Soient $\varphi_1, \varphi_2 \in \mathbb{N}_{\mathscr{S}_n}(\langle\tau\rangle)$ tels que $o(\varphi_1) = r_1$ et $o(\varphi_2) = r_2$.

$$\begin{array}{ll} \tau \circ \varphi_1 \circ \varphi_2 & = \varphi_1 \circ \tau^{r_1} \circ \varphi_2 \\ & = \varphi_1 \circ \tau^{r_1 - 1} \circ \varphi_2 \circ \tau^{r_2} \\ & \quad \vdots \\ & = \varphi_1 \circ \varphi_2 \circ \tau^{r_2 r_1 \bmod m}. \end{array}$$

Et ainsi $o(\varphi_1 \circ \varphi_2) = r_1 r_2 = o(\varphi_1) o(\varphi_2)$.

De plus, $o(Id) = 1$.

Donc o est bien un morphisme de groupes. ⊡

Considérons maintenant les différents cas pour r.

7.4 Cas où $r = 1$

Dans ce cas, cela revient à considérer $ker(o) = \{\varphi \in \mathscr{S}_n \quad | \quad \tau \circ \varphi = \varphi \circ \tau\}$.

Dans la suite, par souci de clarté, nous noterons $\mathscr{F}_{\ell,m} = \mathscr{F}(\mathbb{Z}/\ell\mathbb{Z}, \mathbb{Z}/m\mathbb{Z})$ l'ensemble des fonction de $\mathbb{Z}/\ell\mathbb{Z}$ dans $\mathbb{Z}/m\mathbb{Z}$.

Définition 52 *Soient A et B deux ensembles finis. Soient $\mathscr{S}_A$ et $\mathscr{S}_B$ les deux groupes de permutations respectifs de A et B. On considère le produit cartésien $C = A \times B$.*
Le produit en couronne *$\mathscr{S}_B \wr \mathscr{S}_B$ est le groupe de permutations sur C défini par : $\chi \in \mathscr{S}_A \wr \mathscr{S}_B$ si et seulement si il existe une permutation $\beta \in \mathscr{S}_B$ et $\#B$ permutations $\alpha_u \in \mathscr{S}_A$ indexées par les éléments $u \in B$ telles que*

$$\text{pour tout } (a,b) \in C, \quad \chi((a,b)) = (\alpha_b(a), \beta(b)).$$

Théorème 10 *Soit $\varphi \in \mathscr{S}_n$. Alors*

$$\varphi \in ker(o) \iff \begin{cases} \text{Il existe } \lambda' \in \mathscr{S}_\ell \text{ et } \theta' \in \mathscr{F}_{\ell,m} \text{ telles que} \\ \text{pour tout } a, b, \quad \varphi(a,b) = (\theta'(b) + a, \lambda'(b)). \end{cases}$$

Preuve :
D'après la Proposition 25,

$$\begin{aligned} \varphi \in ker(o) &\iff \forall a, b, \quad \begin{cases} \theta(a+1,b) = \theta(a,b) + 1 \\ \lambda(a+1,b) = \lambda(a,b) \end{cases} \\ &\iff \forall a, b, \quad \begin{cases} \theta(a,b) = \theta(0,b) + a = \theta'(b) + a \\ \lambda(a,b) = \lambda'(b) \end{cases} \\ &\iff \forall a, b, \quad \varphi(a,b) = (\theta'(b) + a, \lambda'(b)). \end{aligned}$$

Remarquons que θ' et λ' ne dépendent que de b.

Tout d'abord, puisque φ est bijective, λ' l'est aussi.

Ensuite, montrons qu'une telle permutation φ reste bijective même si θ' est quelconque. Soient $c \in \mathbb{Z}/m\mathbb{Z}$ et $d \in \mathbb{Z}/\ell\mathbb{Z}$. On cherche a et b tels que $\varphi(a,b) = (c,d)$.

On pose $b = \lambda'^{-1}(d)$ (λ' est bijective).

On pose $a = c - \theta'(b)$.

Et on a bien que

$$\begin{aligned} \varphi(a,b) &= \varphi(c - \theta'(\lambda'^{-1}(d)), \lambda'^{-1}(d)) \\ &= (\theta'(\lambda'^{-1}(d)) + c - \theta'(\lambda'^{-1}(d)), \lambda'(\lambda'^{-1}(d))) \\ &= (c,d). \end{aligned}$$

On n'a donc aucune condition sur θ'. ⊡

Corollaire 5 *Si $\varphi \in ker(o)$, alors*

$$\varphi \in \mathscr{S}_m \wr \mathscr{S}_\ell.$$

Exemple 27

• *Le décalage circulaire de $i\ell$ positions : $\varphi = \tau^i$.*

$$\varphi(a,b) = (a+i, b).$$

$$\begin{cases} \theta'(b) = i \\ \lambda'(b) = b. \end{cases}$$

• *Le décalage circulaire de i positions : $\varphi = T^i$.*

$$\begin{cases} \theta'(b) = (b+i) \ div \ \ell \\ \lambda'(b) = (b+i) \ mod \ \ell \end{cases}$$

• *La concaténation d'une permutation π sur les blocs de taille ℓ : $\varphi = |\pi|\pi| \cdots |\pi|$.*

$$\varphi(a,b) = (a, \pi(b))$$

$$\begin{cases} \theta'(b) = 0 \\ \lambda'(b) = \pi(b) \end{cases}$$

7.5 Cas général $(pgcd(r,m)=1)$

Théorème 11 *Soit $\varphi \in \mathscr{S}_n$. Alors*

$$\varphi \in N_{\mathscr{S}_n}(\langle\tau\rangle) \Longleftrightarrow \left\{ \begin{array}{l} \textit{Il existe } \lambda' \in \mathscr{S}_\ell \textit{ et } \theta' \in \mathscr{F}_{\ell,m} \textit{ telles que} \\ \textit{pour tout } a,b, \quad \varphi(a,b) = (\theta'(b) + ar^{-1}, \lambda'(b)). \end{array} \right.$$

Preuve :
Dans ce cas, $\tau \circ \varphi = \varphi \circ \tau^r$ et d'après la Proposition 25,

$$\begin{aligned} \varphi \in N_{\mathscr{S}_n}(\langle\tau\rangle) \text{ et } o(\varphi) = r \quad &\Longleftrightarrow \forall a,b, \quad \left\{ \begin{array}{l} \theta(a+r,b) = \theta(a,b) + 1 \\ \lambda(a+r,b) = \lambda(a,b) \end{array} \right. \\ &\Longleftrightarrow \forall a,b, \quad \left\{ \begin{array}{l} \theta(a,b) = \theta(0,b) + ar^{-1} = \theta'(b) + ar^{-1} \\ \lambda(a,b) = \lambda'(b) \end{array} \right. \\ &\Longleftrightarrow \forall a,b, \quad \varphi(a,b) = (\theta'(b) + ar^{-1}, \lambda'(b)). \end{aligned}$$

Ici aussi, remarquons que θ' et λ' ne dépendent que de b.

Tout d'abord, puisque φ est bijective, λ' l'est aussi.

Ensuite, montrons qu'une telle permutation φ reste bijective même si θ' est quelconque. Soient $c \in \mathbb{Z}/m\mathbb{Z}$ et $d \in \mathbb{Z}/\ell\mathbb{Z}$. On cherche a et b tels que $\varphi(a,b) = (c,d)$.

On pose $b = \lambda'^{-1}(d)$ (λ' est bijective).

On pose $a = (c - \theta'(b))r$.

Et on a bien que

$$\begin{aligned} \varphi(a,b) &= \varphi((c - \theta'(\lambda'^{-1}(d)))r, \lambda'^{-1}(d)) \\ &= (\theta'(\lambda'^{-1}(d)) + (c - \theta'(\lambda'^{-1}(d)))rr^{-1}, \lambda'(\lambda'^{-1}(d))) \\ &= (c,d). \end{aligned}$$

On n'a donc aucune condition sur θ'. ⊡

Remarque 25 *Remarquons que si l'on remplace r par 1, on retombe exactement sur le Théorème 10. Le Théorème 11 représente dons le cas général.*

Corollaire 6 *Si $\varphi \in N_{\mathscr{S}_n}(\langle\tau\rangle)$, alors*

$$\varphi \in \mathscr{S}_m \wr \mathscr{S}_\ell.$$

Exemple 28
• $\varphi = M_{r^{-1}}$.

$$\varphi(a,b) = (r^{-1}a + (r^{-1}b \ div \ \ell), r^{-1}b \ mod \ \ell).$$

$$\begin{cases} \theta'(b) = r^{-1}b \ div \ \ell \\ \lambda'(b) = r^{-1}b \ mod \ \ell \end{cases}$$

On remarque qu'on retombe bien sur le multiplicateur classique

$$\varphi(a\ell + b) = r^{-1}(a\ell + b).$$

7.6 Sur le cardinal de $N_{\mathscr{S}_n}(\langle\tau\rangle)$

Théorème 12 *Soit $n = m\ell$. Alors*

$$\#N_{\mathscr{S}_n}(\langle\tau\rangle) = \phi(m) \times \ell! \times m^{\ell}.$$

Preuve :
D'après le Théorème 11, pour un r (premier avec m) donné, un élément φ de $N_{\mathscr{S}_n}(\langle\tau\rangle)$ avec $o(\varphi) = r$ est entièrement déterminé par
• θ' un élément quelconque de $\mathscr{F}(\mathbb{Z}/\ell\mathbb{Z}, \mathbb{Z}/m\mathbb{Z})$
• et λ' un permutation quelconque de $\mathscr{S}_\ell$.

Ainsi il y a $\ell! \times m^{\ell}$ permutations φ dans $N_{\mathscr{S}_n}(\langle\tau\rangle)$ avec $o(\varphi) = r$.

Or, il y a $\phi(m)$ éléments inversibles dans $\mathbb{Z}/m\mathbb{Z}$.

Finalement, il y a donc $\phi(m) \times \ell! \times m^{\ell}$ éléments dans $N_{\mathscr{S}_n}(\langle\tau\rangle)$.

⊡

Conclusion et perspectives

Dans cette thèse, nous avons abordé deux thèmes bien distincts. Dans une première partie, nous avons étudié des algorithmes de détection et de reconstruction de codes. Et dans la deuxième, nous avons étudié la structure des codes quasi-cycliques. Ces deux thèmes semblent bien distincts, cependant l'amélioration de la connaissance de la structure de certaines familles de codes peut aboutir à des algorithmes dédiés de reconstruction. Nous en avons donné un exemple pour la reconstruction des codes cycliques. Ainsi l'étude théorique des codes semble être une piste intéressante pour développer de nouveaux algorithmes de reconstruction plus efficaces.

Reconnaissance de codes

La première partie de cette thèse est consacrée au problème de reconnaissance de codes. Nous avons décidé de ne pas étudier le problème général de reconstruction, mais plutôt l'élaboration d'algorithmes dédiés pour des problèmes plus faciles tels que la reconnaissance d'un code ou encore la reconstruction d'un code appartenant à une famille. Cette hypothèse faite sur le code utilisé permet d'obtenir des résultats bien meilleurs. Néanmoins, ces hypothèses sont réalistes. En effet, en pratique, on n'utilise pas de codes aléatoires puisqu'il n'existe pas d'algorithme efficace pour les décoder. On utilise donc des familles de codes possédant un algorithme de décodage efficace tels que les codes de *Reed-Solomon* et les codes LDPC. Nous avons donc maintenant en notre possession des algorithmes permettant de reconstruire de tels codes. Une première perspective serait d'étudier plus profondément ces algorithmes sur des extensions de $\mathbb{F}_2$. En effet, certains codes utilisés en pratique comme les codes de *Reed-Solomon* sont définis sur des extensions. Ensuite, il semble intéressant de continuer l'étude théorique de la structure des codes en espérant pouvoir en adapter un algorithme dédié de reconstruction. Finalement, il semble envisageable d'adapter ces algorithmes à de l'information souple. Cependant ce genre d'adaptation casserait tous les principes de linéarité mais permettrait d'insérer des probabilités qu'il est possible de mettre à jour à chaque nouveau calcul comme dans le décodage des turbo-codes.

Ensuite, nous nous sommes intéressés à l'élaboration d'un algorithme de détection des

paramètres d'un code convolutif. En effet, les algorithmes de reconstruction de codes convolutifs existants font tous une recherche exhaustive sur les paramètres. Cette étude semblait donc intéressante. L'algorithme que nous avons élaboré permet de retrouver le nombre d'entrées, de sorties, et de cases mémoire du codeur ainsi que la probabilité d'erreur du canal. Nous avons montré que pour un taux d'erreur supérieur à 2.10^{-2} seul l'algorithme de détection aboutit alors que pour des taux d'erreur inférieurs, l'algorithme de détection est utile quand la probabilité d'erreur du canal n'est pas connu. Cependant, comparativement aux algorithmes de reconstruction, l'algorithme de détection demande une grande longueur de séquence. Une perspective importante serait de chercher un meilleur test statistique permettant d'utiliser une séquence plus courte. Toutefois, un tel algorithme ne permettrait peut-être pas de retrouver autant de paramètres.

Structure des codes quasi-cycliques

Dans la deuxième partie de cette thèse nous avons abordé la structure des codes quasi-cycliques. Dans un premier temps, nous avons exhibé une nouvelle famille de codes quasi-cycliques annulés par des polynômes à coefficients matriciels. Le but de ce travail était de généraliser les résultats existants pour les codes cycliques aux codes quasi-cycliques. Il a été possible de définir une action des polynômes à coefficients matriciels sur les mots de code. Ainsi, nous avons défini des codes quasi-cycliques annulés par des polynômes à coefficients matriciels. Nous avons finalement montré que le dual d'un code annulé par un polynôme est encore un code annulé par un polynôme qu'il est facile de déterminer. Nous avons ainsi pu construire des codes auto-duaux Euclidiens et Hermitiens, dont certains atteignent les meilleurs distances minimales connues. Une première perspective serait d'élaborer un algorithme de décodage de telles familles. Il semble abordable d'adapter l'algorithme de décodage des codes BCH pour certains codes particuliers. Cependant il faudrait définir un analogue de racine d'un polynôme alors que l'anneau de polynômes que nous considérons n'est ni intègre ni commutatif. Ensuite, afin de construire de tels codes, on est amené à factoriser $X^m - 1$ en polynômes à coefficients matriciels. A l'heure actuelle, nous n'avons pas d'algorithmes efficaces pour effectuer une telle factorisation. En effet, nous utilisons des outils classiques de calcul de bases de Groebner, ce qui ne permet d'atteindre que des petites longueurs de codes. Il semble donc intéressant d'essayer d'améliorer cette factorisation.

Ensuite, toujours dans l'optique de généraliser les résultats existants pour les codes cycliques, nous nous sommes intéressés aux permutations laissant invariant la quasi-cyclicité. Nous avons calculé le normalisateur de T^ℓ, le ℓ-quasi-shift, dans $\mathscr{S}_n$. L'ensemble de ses éléments sont des permutations laissant invariant la quasi-cyclicité dans le cas général. Cependant, il se peut qu'elles ne soient pas les seules. Il reste donc à montrer cela dans les cas où longueur vaut p, pr, p^2, ... de façon analogue aux codes cycliques.

Annexe

Algorithme de reconstruction d'un code cyclique

```
// longueur des codes
n:=1011;

// probabilité d'erreur du canal
p:=0.001;

P<x> := PolynomialRing(GF(2));

FILE:=Open("sequence.txt","r");

// alpha=beta=10^-6
// a=phi^-1(alpha), b=phi^-1(1-beta)
a:=-4.753424310;
b:=4.753424310;

// puisqu'aucun algorithme de recherche de mots de poids faible n'est appliqué,
// w est arbitrairement fixé à n/2
w:=n/2;

// calcul du nombre de mots requis
M:=Round(((b*Sqrt(1-(1-2*p)^(2*w))-a)/((1-2*p)^w))^2);
T:=1/2*(M+a*Sqrt(M));

// chargement de la sequence
C:=ZeroCode(GF(2),n);
mot := [C!0 : k in [1..M]];
for i:=1 to M do
  for j:=1 to n do
    c:=Getc(FILE);
    if c eq "1" then
```

```
      mot[i][j]:=1;
    end if;
  end for;
end for;

// calcul des générateurs des codes maximaux
F:=Factorization(x^n-1);

L:=[F[k][1] : k in [1..#F]];
L1:=[ReciprocalPolynomial((x^n-1) div L[k]) : k in [1..#F]];
N:=[Coefficients(L1[k]) : k in [1..#F]];



h:=[C!0 : k in [1..#F]];
for j:=1 to #F do
  for i:=1 to #N[j] do
    h[j][i]:=N[j][i];
  end for;
end for;

Q:=1;

// boucle sur chaque code maximal
for k:=1 to #F do

  // calcul du nombre de produits scalaires faux
  count:=0;

  for i:=1 to M do
    if InnerProduct(h[k],mot[i]) eq 1 then
      count := count+1;
    end if;
  end for;

  // si le nombre de produits scalaires faux est en-dessous du seuil
  if count le T then Q := Q * L[k]; end if;

end for;

if Q ne 1 then
  print "n = ",n;
```

```
  print "P = ";Q;
end if;
```

Liste des $\Omega(P)$-codes auto-duaux Euclidiens

Voici la liste des polynômes annulant des $\Omega(P)$-codes auto-duaux Euclidiens atteignant la meilleure distance minimale trouvée.

- $[8, 4, 4]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & \omega \\ \omega & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[12, 6, 4]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega & \omega \\ \omega & \omega \end{pmatrix} X^2 + \begin{pmatrix} \omega & \omega \\ \omega & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[16, 8, 6]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^3 + \begin{pmatrix} \omega & 0 \\ 0 & \omega \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[20, 10, 7]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^4 + \begin{pmatrix} 0 & \omega^2 \\ 1 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 0 & 1 \\ \omega^2 & 0 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} 0 & \omega \\ 1 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 0 & 1 \\ \omega & 0 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[24, 12, 8]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega & 0 \\ 0 & \omega \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega^2 \\ \omega^2 & \omega \end{pmatrix} X^3 + \begin{pmatrix} \omega & 0 \\ 0 & \omega \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega^2 \\ \omega & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & \omega \\ \omega & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} 1 & \omega^2 \\ \omega^2 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^5 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & \omega \\ \omega & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} 1 & \omega \\ 1 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & 1 \\ 1 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & \omega \\ \omega^2 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} 1 & \omega \\ \omega & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & \omega^2 \\ \omega & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} 1 & \omega \\ \omega & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} 1 & 1 \\ \omega^2 & 1 \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega^2 \\ \omega^2 & \omega \end{pmatrix} X^3 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega & 0 \\ 0 & \omega \end{pmatrix} X^4 + \begin{pmatrix} \omega & 1 \\ 1 & \omega \end{pmatrix} X^3 + \begin{pmatrix} \omega & 0 \\ 0 & \omega \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[28, 14, 9]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^6 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^5 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^4 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ \omega^2 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} 1 & \omega \\ 1 & 1 \end{pmatrix} X^5 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^4 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

- $[32, 16, 10]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^6 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} 1 & \omega^2 \\ \omega^2 & 1 \end{pmatrix} X^4 + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} X^3 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 1 \\ 1 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^3 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} 1 & \omega \\ \omega & 1 \end{pmatrix} X^4 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^7 + \begin{pmatrix} \omega & 0 \\ 0 & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega & 1 \\ 1 & \omega \end{pmatrix} X^4 + \begin{pmatrix} \omega & 0 \\ 0 & \omega \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^7 + \begin{pmatrix} 0 & \omega \\ 1 & 0 \end{pmatrix} X^6 + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} X^5 + \begin{pmatrix} \omega & 1 \\ 1 & \omega \end{pmatrix} X^4 + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 0 & 1 \\ \omega & 0 \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^7 + \begin{pmatrix} 0 & 1 \\ \omega^2 & 0 \end{pmatrix} X^6 + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 1 \\ 1 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 0 & \omega^2 \\ 1 & 0 \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

- $[36, 18, 10]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega & \omega^2 \\ \omega & \omega \end{pmatrix} X^6 + \begin{pmatrix} 0 & \omega \\ \omega & 0 \end{pmatrix} X^5 + \begin{pmatrix} 0 & \omega \\ \omega & 0 \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega \\ \omega^2 & \omega \end{pmatrix} X^3 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & \omega^2 \\ \omega & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} 0 & \omega^2 \\ \omega^2 & 0 \end{pmatrix} X^5 + \begin{pmatrix} 0 & \omega^2 \\ \omega^2 & 0 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & \omega \\ \omega^2 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^7 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^5 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^4 + \begin{pmatrix} 1 & 1 \\ \omega^2 & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^7 + \begin{pmatrix} 1 & \omega \\ 1 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^5 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^4 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^8 + \begin{pmatrix} 0 & \omega \\ 1 & 0 \end{pmatrix} X^7 + \begin{pmatrix} \omega & 0 \\ \omega^2 & \omega \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega \\ \omega^2 & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega & \omega^2 \\ \omega & \omega \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega^2 \\ 0 & \omega \end{pmatrix} X^3 + \begin{pmatrix} 0 & 1 \\ \omega & 0 \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^8 + \begin{pmatrix} 0 & 1 \\ \omega^2 & 0 \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & \omega \\ 0 & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & \omega \\ \omega^2 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & \omega^2 \\ \omega & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & 0 \\ \omega & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} 0 & \omega^2 \\ 1 & 0 \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega \\ 1 & 1 \end{pmatrix} X^7 + \begin{pmatrix} 0 & 1 \\ \omega & 0 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & 1 \\ 0 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 0 \\ 1 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} 0 & \omega \\ 1 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^8 + \begin{pmatrix} 1 & 1 \\ \omega^2 & 1 \end{pmatrix} X^7 + \begin{pmatrix} 0 & \omega^2 \\ 1 & 0 \end{pmatrix} X^6 + \begin{pmatrix} \omega & 0 \\ 1 & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega & 1 \\ 0 & \omega \end{pmatrix} X^4 + \begin{pmatrix} 0 & 1 \\ \omega^2 & 0 \end{pmatrix} X^3 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^8 + \begin{pmatrix} 0 & 1 \\ \omega & 0 \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ \omega^2 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 1 & 1 \\ \omega^2 & 1 \end{pmatrix} X^5 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^4 + \begin{pmatrix} 1 & \omega^2 \\ 0 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 0 & \omega \\ 1 & 0 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ \omega^2 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & 0 \\ \omega^2 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & \omega^2 \\ 0 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} 1 & \omega^2 \\ 0 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ \omega^2 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^8 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^7 + \begin{pmatrix} \omega & 1 \\ \omega & \omega \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega \\ 1 & \omega \end{pmatrix} X^3 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^8 + \begin{pmatrix} \omega^2 & \omega^2 \\ 1 & \omega^2 \end{pmatrix} X^7 + \begin{pmatrix} \omega & \omega \\ 1 & \omega \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega \\ 1 & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega & 1 \\ \omega & \omega \end{pmatrix} X^4 + \begin{pmatrix} \omega & 1 \\ \omega & \omega \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 1 \\ \omega^2 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^8 + \begin{pmatrix} \omega^2 & \omega^2 \\ 1 & \omega^2 \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & 1 \\ 0 & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & 0 \\ 1 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 1 \\ \omega^2 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega & \omega \\ \omega & \omega \end{pmatrix} X^8 + \begin{pmatrix} 0 & \omega \\ \omega & 0 \end{pmatrix} X^7 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^4 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^3 + \begin{pmatrix} 0 & \omega \\ \omega & 0 \end{pmatrix} X^2 + \begin{pmatrix} \omega & \omega \\ \omega & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega & \omega \\ \omega & \omega \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega \\ \omega & 1 \end{pmatrix} X^7 + \begin{pmatrix} 1 & \omega^2 \\ \omega & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & \omega^2 \\ \omega & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & \omega \\ \omega^2 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} 1 & \omega \\ \omega^2 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & \omega \\ \omega & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & \omega \\ \omega & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^8 + \begin{pmatrix} 0 & 1 \\ \omega^2 & 0 \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ \omega & 1 \end{pmatrix} X^6 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X^5 + \begin{pmatrix} 1 & \omega \\ 1 & 1 \end{pmatrix} X^4 + \begin{pmatrix} 1 & \omega \\ 0 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 0 & \omega^2 \\ 1 & 0 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega \\ 1 & 1 \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ \omega & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega & 0 \\ \omega & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega & \omega \\ 0 & \omega \end{pmatrix} X^4 + \begin{pmatrix} 1 & \omega \\ 0 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^8 + \begin{pmatrix} \omega & \omega \\ 1 & \omega \end{pmatrix} X^7 + \begin{pmatrix} \omega & 1 \\ 0 & \omega \end{pmatrix} X^6 + \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} X^4 + \begin{pmatrix} \omega & 0 \\ 1 & \omega \end{pmatrix} X^3 + \begin{pmatrix} \omega & 1 \\ \omega & \omega \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^8 + \begin{pmatrix} \omega & \omega \\ 1 & \omega \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & \omega^2 \\ 1 & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & \omega^2 \\ 1 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 1 \\ \omega^2 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & 1 \\ \omega^2 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega & 1 \\ \omega & \omega \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^8 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & 1 \\ \omega^2 & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & \omega^2 \\ 1 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^9 + \begin{pmatrix} \omega^2 & \omega^2 \\ \omega^2 & \omega^2 \end{pmatrix} X^8 + \begin{pmatrix} 0 & \omega^2 \\ \omega^2 & 0 \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} 0 & \omega^2 \\ \omega^2 & 0 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & \omega^2 \\ \omega^2 & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

- $[40, 20, 12]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^{10} + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega & \omega \\ 0 & \omega \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & 1 \\ 1 & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} \omega & 0 \\ \omega & \omega \end{pmatrix} X^4 + \begin{pmatrix} 1 & 1 \\ \omega^2 & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^{10} + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega \\ 1 & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & \omega^2 \\ 0 & \omega^2 \end{pmatrix} X^6 + \begin{pmatrix} \omega & 1 \\ 1 & \omega \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 0 \\ \omega^2 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega^1 & 1 \\ \omega & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^{10} + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^9 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega \\ 1 & 1 \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega^2 & \omega \\ \omega & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^{10} + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^9 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^{10} + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^9 + \begin{pmatrix} \omega & 0 \\ 0 & \omega \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega^2 \\ 0 & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega & \omega^2 \\ \omega^2 & \omega \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ \omega^2 & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega & 0 \\ 0 & \omega \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^{10} + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^9 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega & \omega^2 \\ \omega^2 & \omega \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} 1 & 1 \\ \omega^2 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^{10} + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^9 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^8 + \begin{pmatrix} \omega & \omega^2 \\ 1 & \omega \end{pmatrix} X^7 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega & 1 \\ \omega^2 & \omega \end{pmatrix} X^3 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^{10} + \begin{pmatrix} \omega^2 & \omega \\ 1 & \omega^2 \end{pmatrix} X^9 + \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} X^8 + \begin{pmatrix} 1 & \omega \\ 0 & 1 \end{pmatrix} X^7 + \begin{pmatrix} \omega^2 & \omega \\ \omega & \omega^2 \end{pmatrix} X^5 + \begin{pmatrix} 1 & 0 \\ \omega & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} \omega^2 & 1 \\ \omega & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Liste des $\Omega(P)$-codes auto-duaux Hermitiens

Voici la liste des polynômes annulant des $\Omega(P)$-codes auto-duaux Hermitiens atteignant la meilleure distance minimale trouvée.

- $[8, 4, 4]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[12, 6, 4]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[16, 8, 6]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^4 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^3 + \begin{pmatrix} \omega & 1 \\ 1 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[20, 10, 8]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 1 \\ 1 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega \\ \omega & 1 \end{pmatrix} X^3 + \begin{pmatrix} 1 & \omega^2 \\ \omega^2 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 1 \\ 1 & \omega \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[24, 12, 8]$:

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

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X^5 + \begin{pmatrix} \omega & \omega \\ \omega^2 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega^2 \\ \omega & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega & \omega \\ \omega^2 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega & 1 \\ 1 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega \\ \omega^2 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} \omega & 1 \\ 1 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega & \omega^2 \\ 0 & 0 \end{pmatrix} X^4 + \begin{pmatrix} \omega & 1 \\ 1 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} 0 & 0 \\ \omega & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega & \omega^2 \\ 0 & 0 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & \omega^2 \\ \omega & \omega \end{pmatrix} X^3 + \begin{pmatrix} 0 & 0 \\ \omega & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega & 1 \\ 1 & 0 \end{pmatrix} X^4 + \begin{pmatrix} 0 & \omega^2 \\ \omega & 0 \end{pmatrix} X^3 + \begin{pmatrix} 0 & 1 \\ 1 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^6 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega^2 & 1 \\ \omega^2 & 1 \end{pmatrix} X^4 + \begin{pmatrix} \omega^2 & \omega^2 \\ \omega & \omega \end{pmatrix} X^3 + \begin{pmatrix} 1 & \omega \\ 1 & \omega \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- $[28, 14, 10]$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^7 + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} X^6 + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} X^5 + \begin{pmatrix} \omega & 0 \\ 1 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega & 1 \\ 0 & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} X^2 + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X^7 + \begin{pmatrix} 1 & \omega^2 \\ 1 & 1 \end{pmatrix} X^6 + \begin{pmatrix} \omega & 1 \\ \omega^2 & 0 \end{pmatrix} X^5 + \begin{pmatrix} \omega & \omega^2 \\ 0 & \omega^2 \end{pmatrix} X^4 + \begin{pmatrix} \omega & 0 \\ \omega & \omega^2 \end{pmatrix} X^3 + \begin{pmatrix} 0 & \omega \\ 1 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 1 \\ \omega & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Table des figures

Index

Bibliographie

[Als79] B. Alspach et T.D. Parsons, "Isomorphism of circulant graphs and digraphs", dans *Discrete Mathematics*, 25, pages 97-108, 1979.

[BGH06] J. Barbier, G. Sicot et S. Houcke, "Algebraic approach for the reconstruction of linear and convolutional error correcting codes", dans *Proceedings of the 3rd International Conference on Computer Science and Engineering CISE 2006*, volume 16, pages 66-71, Venice, Italy, novembre 2006.

[Bar07] J. Barbier, "Analyse de canaux de communication dans un contexte non coopératif", Thèse de doctorat, École polytechnique, 2007.

[BCGO09] T. Berger, P.L. Cayrel, P. Gaborit, et A. Otmani, "Reducing Key Length of the McEliece Cryptosystem", dans *Proceedings of Second International Conference on Cryptology - AfricaCrypt 2009*, Gammarth, Tunisie, Juin 21-25 2009.

[BMV78] E.R Berlekamp, R.J. McEliece et H.C.A Van Tilborg, "On the inherent intractability of certain coding problems", *IEEE Transactions on Information Theory*, 24(3) :384-386, 1978.

[BLP08] D.J. Bernstein, T. Lange et C. Peters, "Attacking and defending the McEliece cryptosystem", à paraître dans *Proceedings of PQCrypto 2008*, Springer.

[BU09] D. Boucher et F. Ulmer, "Coding with skew polynomial rings", A paraître dans *Journal of Symbolic Computation*.

[Bra91] N. Brand, "Polynomial isomorphisms of combinatorial objects", dans *Graphs and Combinatorics*, 7, 1991.

[Cam04] M. Camus, "Suites récurrentes linéaires à coefficients matriciels", mémoire de DEA, Université de Limoges, 2004.

[Can96] A. Canteaut, "Attaques de cryptosystèmes à mots de poids faible et construction de fonctions t-résilientes", Thèse de doctorat, Université Paris-VI, 1996.

[CC98] A. Canteaut et F. Chabaud, "A new algorithm for finding minimum-weight words in a linear code : application to primitive narrow-sense BCH codes of length 511", *IEEE Transaction on Information Theory*, 44(1) :367-378, Janvier 1998.

[CCN09] P-L. Cayrel, C. Chabot et A. Necer, "Quasi-cyclic codes over ring of matrices", à paraître dans le journal Finite Fields and Applications.

[Cha06] C. Chabot, "Reconnaissance de codes", Mémoire de Master, Université de Limoges, 2006.

[Cha07] C. Chabot, "Recognition of a code in a noisy environment", dans *Proceedings of the 2007 IEEE International Symposium on Information Theory*, 2211-2215, Nice, 2007.

[Cha09] C. Chabot, "Reconstruction of families of codes. Application to cyclic codes.", dans *Proceedings du International Workshop on Coding and Cryptography*, Ullensvang, NORVEGE, Mai 2009.

[CB09] C. CHABOT ET J.BARBIER, "Detection of convolutional codes parameters. Application to reconstruction", soumis.

[Clu04] M. CLUZEAU, "Reconstruction of a linear scrambler", in *Proceedings of the 2004 IEEE International Symposium on Information Theory*, Chicago, USA, pp 230.

[Clu06a] M. CLUZEAU, "Block code reconstruction using iterative decoding techniques", in *Proceedings of the 2006 IEEE International Symposium on Information Theory*, ISIT06, Seattle, USA, Juillet 2006.

[Clu06b] M. CLUZEAU, "Reconnaissance d'un schéma de codage", Thèse de doctorat, École polytechnique, 2006.

[Clu07] M. CLUZEAU, "Reconstruction of a linear scrambler", in *IEEE Transactions on Computers*, vol 56, num 9, Septembre 2007.

[Clu08] M. CLUZEAU ET J.P. TILLICH, "On the code reverse engineering problem", dans *Proceedings of the 2008 IEEE International Symposium on Information Theory, ISIT08*, Toronto, Canada, Juillet 2008.

[Clu09] M. CLUZEAU ET M. FINIASZ, "Recovering a Code's Length and Synchronization from a Noisy Intercepted Bitstream", dans *Proceedings of ISIT 2009*,IEEE, 2009.

[CS93] J. CONAN ET G. SÉGUIN, "Structural properties and enumeration of quasi-cyclic codes", dans *Applicable Algebra in Engineering, Communication and Computing 4*, 1993.

[DH07] J. DINGEL ET J. HAGENAUER, "Parameter Estimation of a convolutional encoder from noisy observations", dans *Proceedings of the 2007 IEEE International Symposium on Information Theory, ISIT 07*, Nice, Juin 2007.

[Eli55] P. ELIAS, "Coding for noisy channels", dans *IRE International Convention Record (part 4)*, pages 37-46, 1955.

[Fil97] É. FILIOL, "Reconstruction of convolutional encoders over $GF(q)$", dans *IMA Conference on Cryptography and Coding*, volume 1355 de *Lecture Notes in Computer Science*, pages 100-110, Springer-Verlag, 1997.

[Fil00] É. FILIOL, "Reconstruction of punctured convolutional encoders", dans *International Symposium on Information Theory and Applications*, pages 4-7, SITA and IEICE, 2000.

[Fil01] É. FILIOL, "Techniques de reconstruction en cryptologie et théorie des codes", Thèse de doctorat, École Polytechnique, 2001.

[FL01] P. FITZPATRICK ET K. LALLY, "Algebraic structure of quasicyclic codes", dans *Discrete Applied Mathematics*, 111(2001), pages 157-175.

[For73] JR. G.D. FORNEY, "The Viterbi algorithm", in *Proc. IEEE*, volume 61, 268-276, mars 1973.

[Gab05] P. GABORIT, "Shorter keys for code-based cryptography", dans *Proceedings of Workshop on Codes and Cryptography*, Bergen, pages 81-90, 2005.

[GZ06] P. GABORIT ET G. ZÉMOR, "Asymptotic improvement of the Gilbert-Varshamov bound for binary linear codes", *2006 IEEE International Symposium on Information Theory*, July 2006, pp287-291.

[HJP93] W. C. HUFFMAN, V. JOB ET V. PLESS, "Multipliers and generalized multipliers of cyclic objects and cyclic codes", *Journal of combinatorial theory*, Series A 62, pp 183-215, 1993.

[HP03] W.C HUFFMAN ET V. PLESS, "Fundamentals of error-correcting codes", Cambridge, University Press, 2003.

[JJ02] T. JOHANSSON ET F. JÖNSSON, "On the complexity of some cryptographic problems based on the general decoding problem", dans *IEEE Transactions on Information Theory*, volume 48, No 10, Octobre 2002.

[LB88] P.J. LEE ET E.F. BRICKELL, "An observation on the security of McEliece's public-key cryptosystem", dans *Advances in Cryptology - EUROCRYPT'88*, volume 330 de *Lecture Notes in Computer Science*, pages 275-280, Springer-Verlag, 1988.

[Leo88] J.S. LEON, "A probabilistic algorithm for computing minimum weights of large error-correcting codes", *IEEE Transaction on Information Theory*, 34(5) :1354-1359, 1988.

[LS01a] S. LING ET P. SOLÉ, "Decomposing quasi-cyclic codes", dans *Proceedings of International Workshop on Coding cryptography*, pages 507-517, Paris 2001.

[LS01b] S. LING ET P. SOLÉ, "On the algebraic structure of quasi-cyclic codes I : finite fields", dans *IEEE Transactions on Information Theory*, 47, pages 2751-2760, 2001.

[MS77] F.J. MACWILLIAMS AND N.J. SLOANE, *The theory of error-correcting codes*, Amsterdam, The Netherlands : North-Holland, 1977.

[McE78] R.J. MCELIECE, "A public-key cryptosystem based on algebraic coding theory", *JPL DSN Progress Report*, pages 114-116, 1978.

[McE98] R.J. MCELIECE, *Handbook of coding theory*, volume 2, chapitre 12, "The algebraic theory of convolutional codes", 1065-1138, Elsevier Science, 1998.

[NB07] N. NARDEAU ET W. BENMOUSSA, "Transformation de la matrice génératrice d'un code quasi-cyclique", Rapport de TER, Université de Limoges, 2007.

[Nec99] A. NECER, "Systèmes récursifs et algèbre de Hadamard de suites récurrentes linéaires sur des anneaux commutatifs", *Communications in Algabra*, 27 (12), pp 6175,6189, 1999.

[Nech95] A. A. NECHAEV, "Finite quasi-frobenius modules, applications to codes and linear recurrences", *Fundamentalnaya i prikladnaya matematika*, 1 : pp 229-254, 1995.

[Nie95] H. NIEDERREITER, "The Multiple-Recursive Matrix Method for Pseudorandom Number Generation", *Finite fields and their applications*, pp 3-30, 1995.

[Omu69] J.K. OMURA, "On the Viterbi decoding algorithm", *IEEE Transactions on Information Theory*, IT-15 :177-179, janvier 1969.

[OTD09] A. OTMANI, J.P. TILLICH ET L. DALLOT, "Cryptanalysis of Two McEliece Cryptosystems Based on Quasi-Cyclic Codes", à paraître dans *Special issues of Mathematics in Computer Science*.

[Pal87] P.P. PÁLFY, "Isomorphism problem for relational structures with a cyclic automorphism", dans *European Journal of Combinatorics*, 8, pages 35-43, 1987.

[PH88] A. POLI AND LI. HUGUET, *Codes correcteurs-Théorie et applications*, Paris, France : Masson, 1988.

[Ric95] B. RICE, "Determining the parameters of a rate $1/n$ convolutional encoder over $GF(q)$", dans *Third International Conference on Finite Fields and Applications*, Glasgow, 1995.

[Nist01] ANDREW RUKHIN, JUAN SOTO, JAMES NECHVATAL, MILES SMID, ELAINE BARKER, STEFAN LEIGH, MARK LEVENSON, MARK VANGEL, DAVID BANKS, ALAN HECKERT, JAMES DRAY ET SAN VO, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", 2001.

[Sha48] C.E. SHANNON, "A mathematical theory of communication", *Bell System Technical Journal*, 27 :379-423, 623-656, 1948.

[Ske97] G. SKERSYS, "Etudes de codes quasi-cycliques comme codes concaténés", Preprint, Université de Limoges, 1997.

[Ske99] G. SKERSYS, "Calcul du groupe d'automorphismes des codes. Détermination de l'équivalence des codes", Thèse de doctorat, Université de Limoges, 1999.

[Ste89] J. STERN, "A method for finding codewords of small weight", dans *Coding Theory and Applications*, volume 388 de *Lecture Notes in computer Science*, pages 106-113, Springer-Verlag, 1989.

[Val00] A. VALEMBOIS, "Décodage, Détection et Reconnaissance de codes", Thèse de Doctorat, Université de Limoges, 2000.

[Val01] A. VALEMBOIS, "Detection and recognition of a binary linear code", *Discrete Applied Mathematics*, 111 :199-218, 2001.

[Var97] A. VARDY, "The intractability of computing the minimum distance of a code", *IEEE Transactions on Information Theory*, 43 :1757-1766, Novembre 1997.

[Vit67] A.J. VITERBI, "Error bounds for convolutional codes and asymptotically optimum decoding algorithm", *IEEE Transactions of Information Theory*, IT-13 :260-269, avril 1967.

Résumé :
Dans cette thèse, nous abordons tout d'abord le problème de reconnaissance de codes. Il consiste à retrouver la structure d'un code correcteur d'erreurs utilisé lors d'une transmission de données seulement à partir de la séquence bruitée interceptée. Nous donnons ici des méthodes efficaces pour la reconnaissance d'un code connu, pour la reconstruction de codes appartenant à une famille tels que les codes cycliques et pour la détection des paramètres de codes convolutifs. Ensuite, nous étudions la structure des codes quasi-cycliques parallèlement aux résultats connus pour les codes cycliques. Nous donnons une construction d'une sous-famille de codes quasi-cycliques annulés par un polynôme à coefficients matriciels. Cette construction permet de trouver des codes ayant de bonnes distances minimales. Finalement, nous nous intéressons aux permutations laissant invariante la quasi-cyclicité d'un code.

Mots-clés : Codes correcteurs d'erreurs, Reconnaissance de codes, Codes quasi-cycliques, Codes convolutifs, Codes cycliques, Analyse du train binaire, Structure de codes.

Abstract :
In this thesis, we first deal with the problem of recognition of codes. It consists in recovering the structure of an error-correcting code used during a data transmission only from the noisy intercepted sequence. We give efficient methods for the recognition of a known code, for the reconstruction of codes belonging to a family like cyclic codes and for the detection of parameters of convolutional codes. Then, we study the structure of quasi-cyclic codes in parallel of the results known for cyclic codes. We give a construction of a sub-family of quasi-cyclic codes cancelled by a polynomial with matricial coefficients. Some of these codes reach large minimum distances. Finally, we deal with permutations keeping the quasi-cyclicity of a code.

Keywords : Error correcting codes, Recognition of codes, Quasi-cyclic codes, Convolutional codes, Cyclic codes, Analysis of binary sequences, Structure of codes.