Influence du mapping sur la reconnaissance d'un système de communication

Marion Bellard

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Marion Bellard

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Influence du mapping sur la reconnaissance d'un système de communication

soutenue le 30 janvier 2014

devant le jury composé de :

Thierry Berger	Université de Limoges	Rapporteur
Pierre Loidreau	DGA-MI	Rapporteur
Gilles Burel	Université de Bretagne Occidentale	Examinateur
Jean-Pierre Tillich	INRIA Paris-Rocquencourt	Examinateur
Annick Valibouze	Université Pierre et Marie Curie	Examinateur
Nicolas Sendrier	INRIA Paris-Rocquencourt	Directeur de thèse

Marion Bellard

Influence du mapping sur la reconnaissance d'un système de communication

INRIA-équipe-projet SECRET
Domaine de Voluceau
78153 Le Chesnay

Remerciements

Je tiens à remercier ici toutes les personnes qui m'ont permis d'effectuer cette thèse dans d'excellentes conditions, aussi bien pour leurs qualités techniques que humaines.

Je tiens d'abord à remercier Nicolas Sendrier mon directeur de thèse pour ses conseils et ses idées qui ont été précieuses et nécessaires au bon déroulement de ces trois années de thèse.

Je tiens également à remercier Jean-Pierre Tillich, avec qui j'ai eu la chance de travailler, pour son exigence et sa patience.

Merci ensuite à mes deux rapporteurs Pierre Loidreau et Thierry Berger qui ont donné de leur temps pour la relecture de ce manuscrit. Je devrais d'ailleurs remercier doublement Thierry Berger, je lui dois en effet ma découverte des codes correcteurs au cours de mes deux années de Master. Merci également à Gilles Burel et Annick Valibouze d'avoir accepté de faire partie de mon jury.

Je ne saurais remercier les gens qui m'ont aidée et soutenue durant cette thèse sans remercier Audrey, avec qui j'ai partagé de multiples interrogations et rebondissements, pour sa disponibilité. Je lui souhaite bonne chance pour la suite.

Mes pensées vont ensuite vers Maxime et Jérôme sans qui cette thèse n'aurait pas eu lieu et bien sûr Florine, Denis, Marc, Pierre, Angélique et François pour leur accueil et leur sympathie. Je ne doute pas que j'aurai le plaisir de les revoir.

Un grand merci à tous les membres et anciens du projet SECRET que j'ai eu l'occasion de côtoyer Anne, Pascale, María, Christelle, André, Anthony, Ayoub, Baudoin, Céline, Christina, Denise, Gaëtan, Grégory, Joëlle, Mamdouh, Matthieu, Rafael, Stéphane, Valentin, Vincent, Virginie et Yann. Je garderai d'excellents souvenirs des trois années que j'ai passées au projet.

Je dois une dédicace particulière à Christelle qui est d'un grand secours et qui nous aide toujours avec le sourire, à Ayoub, Baudoin, Céline, Christina et Valentin avec qui j'ai partagé mes séances de footing et à Christina de nouveau en tant que pâtissière en chef du projet et ancienne collègue de bureau, merci à elle pour ses excellents gâteaux et sa bonne humeur. Merci également à tous ceux avec qui j'ai appris l'art des mots croisés.

Mes prochains remerciements vont à mes collègues Grégory, Valentin et Virginie qui font du bureau 1 un endroit où il fait bon travailler. Mille mercis à Grégory pour tous ses conseils et son aide précieuse, à Valentin pour sa disponibilité sans faille et à Virginie pour sa gentillesse et sa présence.

Merci ensuite à mes proches pour leur soutien et leur patience, mes amies Amanda et Adeline, ma famille Fanny, Michael, Fredo, Ingrid, Christine, mes parents bien sûr et enfin Nicolas.

Notations

Notations mathématiques

$\boldsymbol{H}^\top$	la transposée de la matrice H
$\boldsymbol{d_H}(\ .\ ,\ .\ )$	la distance de Hamming
$\boldsymbol{d_e}(\ .\ ,\ .\ )$	la distance euclidienne
$\boldsymbol{W_H}(\ .\ )$	le poids de Hamming
$\mathbb{F}_{\boldsymbol{q}}$	le corps à q éléments, q puissance d'un nombre premier
$\boldsymbol{GL(a, \mathbb{F}_2)}$	l'ensemble des matrices inversibles de taille $a \times a$ à coefficients dans $\mathbb{F}_2$
$\boldsymbol{pgcd}$	le plus grand diviseur commun
$\boldsymbol{ppcm}$	le plus petit multiple commun
$\boldsymbol{degg(x)}$	le degré du polynôme $g(x)$

Notations relatives aux mappings

$\boldsymbol{a}$	le nombre de bits par symboles binaires de la modulation
$\boldsymbol{\mathcal{C}}$	une constellation
$\boldsymbol{C_L(f)}$	la classe linéaire de f
$\boldsymbol{C_A(f)}$	la classe affine de f
$\boldsymbol{f}$	un mapping
$\boldsymbol{G_p}$	la pénalité de Gray moyenne
$\boldsymbol{G_{p^k}}$	la pénalité de Gray maximum

Notations relatives aux codes correcteurs

$\mathscr{C}$	un code
$\boldsymbol{n}$	la longueur du code
$\boldsymbol{k}$	la dimension du code
$\boldsymbol{G}$	une matrice génératrice
$\boldsymbol{H}$	une matrice duale

Chapitre 1

$\boldsymbol{A_0}$	l'amplitude de l'onde porteuse
$\boldsymbol{\mathcal{A}}$ et $\boldsymbol{\mathcal{B}}$	des alphabets de modulation
$\boldsymbol{b(t)}$	un bruit blanc gaussien
$\boldsymbol{f_0}$	la fréquence de l'onde porteuse
$\boldsymbol{g(t)}$	une impulsion élémentaire
$\boldsymbol{M}$	le nombre de points de la constellation
$\boldsymbol{M-ASK}$,$\boldsymbol{M-PSK}$, $\boldsymbol{M-QAM}$	une modulation à M points
$\boldsymbol{p(t)}$	une onde porteuse
$\boldsymbol{r(t)}$	le signal reçu
$\boldsymbol{s_I(t)}$	la porteuse en phase
$\boldsymbol{s_Q(t)}$	la porteuse en quadrature
$\boldsymbol{s(t)}$	le signal émis
$\boldsymbol{\varphi_0}$	la phase de l'onde porteuse

Chapitre 4

$\boldsymbol{C_{\setminus I}}$	code poinçonné aux positions indéxées par I
$\boldsymbol{H_I}$	matrice duale restreinte aux positions indixées par I
$\boldsymbol{p}$	la probabilité d'erreur du canal binaire symétrique
$\boldsymbol{P_C}$	la probabilité de collisions pour des données codées
$\boldsymbol{P_{aléa}}$	la probabilité de collisions pour des données aléatoires
$\boldsymbol{R = \frac{P_C}{P_{aléa}}}$	le rapport des probabilités de collisions
$\boldsymbol{t}$	la taille des blocs de lecture d'une séquence binaire
$\boldsymbol{X_{obs}}$	le nombre de collisions observées sur une séquence binaire
$\boldsymbol{x_{obs}}$	la probabilité de collisions observée sur une séquence biniare

Résumé

Ce document présente les travaux effectués durant ma thèse au sein de l'Equipe-projet SECRET à INRIA Paris-Rocquencourt.

Nous nous intéressons à la reconstruction d'un système de communication dans un contexte non coopératif. Nous cherchons d'une part à reconstruire l'association réalisée entre symboles binaires et symboles physiques lors de la conversion de données binaires en un signal modulé (éventuellement bruité). On appelle cette opération le *mapping*. Dans un canal hertzien, une modulation M-aire peut transmettre M symboles distincts formant une *constellation* dans un espace euclidien multidimensionnel (par exemple bidimensionnel, représentant la phase et l'amplitude des symboles transmis). Nous regardons essentiellement les modulations de 4 à $256 - QAM$ (*Quadrature Amplitude Modulation*). En l'absence de redondance il est impossible de décider quel mapping a été utilisé, aussi nous supposons ici que les données binaires sont codées. Nous nous intéressons particulièrement aux mappings respectant le critère de Gray (deux symboles de la constellation, voisins pour la distance euclidienne, correspondront à des symboles binaires à distance de Hamming de 1). Ce type de mapping est en effet couramment utilisé car il permet de limiter l'impact des erreurs d'interprétations lors de la démodulation. La recherche exhaustive du mapping pour ce type de constellations devient impossible lorsque le nombre de points de celles-ci augmente, alors que les grandes constellations possèdent l'avantage d'augmenter les débits de transmission. Nous définissons alors deux types de classes d'équivalences de mappings : linéaires et affines. Nous définissons de plus un test basé sur la reconnaissance de codes convolutifs et le taux d'erreur du canal. Nous construisons un algorithme de reconstruction, en appliquant ce test, de manière appropriée, à un ensemble de représentants des classes d'équivalences définies. Nous verrons que le résultat d'un tel parcours n'est pas unique mais qu'il permet de réduire significativement le nombre de mappings possibles tout en réduisant le coût de la recherche.

D'autre part nous définissons une méthode de détection des paramètres d'un code convolutif ainsi qu'une méthode de reconstruction du dual d'un tel code. Nous regardons une séquence binaire codée et bruitée par blocs de taille t. Deux blocs identiques définissent ce que l'on appelle une *collision*. La cardinalité de l'ensemble des blocs observés pour des données codées est inférieure à celle observée pour des données aléatoires. Donc la donnée de la probabilité de collisions pour différentes tailles de blocs permet de distinguer des données codées de données aléatoires. Des blocs glissants sont utilisés pour caractériser la présence d'un code convolutif. Cela permet d'obtenir la longueur, la dimension du code et le degré maximum des polynômes générateurs du dual ainsi que la taille de chacune des équations de parité. De plus en poinçonnant des bits dans chacun des blocs observés nous déduisons de la probabilité de collisions des équations de parité du code et une matrice génératrice du dual. La probabilité de collision est de plus invariante pour le changement de mapping lorsque la taille de bloc est un multiple de la taille des symboles du mapping. Nous sommes alors en mesure de détecter la longueur du code sans connaître le mapping qui a été utilisé lors de la modulation.

Table des matières

Introduction

La transmission d'une information numérique nécessite notamment un codage, à l'aide de codes correcteurs d'erreurs, puis une adaptation au canal de transmission que l'on appelle modulation. La modulation est composée de deux parties. La première appelée le mapping permet d'associer des symboles binaires à des paramètres physiques comme une amplitude et une phase. La seconde partie de la modulation permet elle d'utiliser ces valeurs pour modifier ponctuellement un signal porteur. Lors de la réception d'un signal il faut effectuer les opérations inverses de celles appliquées en émission afin de retrouver l'information émise. Un signal reçu doit donc avant toute chose être démodulé. On parle de contexte non coopératif lorsqu'un observateur doit recouvrer la séquence binaire émise à partir du seul signal transmis. Dans ce contexte il faut donc démoduler le signal observé pour associer aux symboles physiques une information binaire et enfin reconstruire le codage canal utilisé. Nous nous intéressons ici à ce problème de reconstruction d'un système de communication. Retrouver la modulation correspond au problème de la démodulation aveugle pour lequel il existe de nombreux travaux académiques, cependant cette démodulation n'inclut pas la reconstruction du mapping. Nous étudions alors le problème de la reconstruction du mapping. Retrouver le codage utilisé correspond au problème de la reconstruction de codes correcteurs d'erreurs pour lequel il existe également de nombreux travaux de recherche. On pourra citer notammant le travail de A. Valembois sur la reconstruction du dual d'un code correcteur d'erreur. Son algorithme se base sur la méthode de Canteaut-Chabaud de recherche de mots de petits poids dans un code. Des travaux spécifiques sur la reconstruction des codes convolutifs existent comme les thèses de E. Filiol, J. Barbier et M. Côte. Nous sommes intéressés à résoudre le problème de la reconnaissance de mapping lorsque les données sont codées par un codeur convolutif et nous utilisons le travail de M. Côte pour notre étude. Cela nous permet d'obtenir une matrice génératrice du code correcteur utilisé à partir d'une séquence codée bruitée. Cependant, si l'on ne connaît pas l'association réalisée entre symboles binaires et symboles physiques, le mapping, on ne connaît pas le comportement de l'algorithme de reconnaissance de code convolutif. Nous nous intéressons à ce problème et étudions l'interaction entre la reconnaissance de codes convolutifs et la recherche de mapping.

Dès lors que la transmission d'un signal s'effectue grâce à un protocole de communication, la séquence composée des mots de code transmis est encapsulée. De ce fait, il est possible de s'appuyer sur des balises pour connaître la synchronisation des mots de codes, c'est-à-dire que l'on connait le début des mots. Nous considérons alors dans ces travaux que la synchronisation est connue et qu'on n'avons pas à la rechercher.

Nous avons abordé nos travaux selon deux axes. Le premier objectif est de décrire une méthode de reconstruction du mapping à partir de données démodulées avec un mapping par défaut. Le second axe consiste à obtenir des informations sur le code de manière invariante pour le mapping.

Le second axe nous conduit à l'utilisation d'un test statistique basé sur le comptage de collisions dans une séquence binaire observée par blocs. Ce test permet d'estimer le cardinal de l'ensemble observé et donne une information à la fois quantitative et qualitative sur l'espace des blocs observés. Nous définissons alors une méthode de détection de la longueur d'un code convolutif invariante pour le changement de mapping. C'est-à-dire que nous sommes en mesure de dire si un code convolutif à été utilisé pour coder les données observées et le cas écheant d'en donner la longueur et ce sans connaître le mapping utilisé. Nous définissons de plus une méthode de détection des paramètres d'un code convolutif et de reconstruction du dual connaissant le mapping. C'est l'objet du Chapitre 4. Ce travail a été soumis à publication.

Le premier axe concerne la reconstruction du mapping lorsque les données observées sont codées. Nous nous sommes principalement intéressés au cas des codes convolutifs. Lorsque l'on parcourt les divers mappings possibles on peut observer une multitude de séquences codées correspondant à divers codes correcteurs. Nous explicitons alors les liens qui relient les différents codes observés. Des classes d'équivalences de mapping peuvent être définies. Elles seront présentées dans le Chapitre 5. Ceci nous conduit à l'élaboration d'une méthode de reconstruction de mappings permettant de diminuer le nombre de mappings à parcourir pour les petites tailles de constellations. Pour des constellations à 16 points et plus, nous considérons uniquement les mappings de type Gray et quasi-Gray. La méthode s'étend naturellement à ces mappings. Celle-ci est applicable dans le cas où le signal transmis est un signal bruité. Nous regardons également le comportement des codes de Reed-Solomon vis-à-vis du changement de mapping. Ceci est vu au Chapitre 6. Cette partie du travail à fait l'objet d'une publication à ISITA'2012 [7].

Chapitre 1

Introduction à la Transmission du signal Numérique

On peut représenter un système de communication (FIGURE 1.1) par la suite des traitements appliqués aux données binaires, en émission et en réception, afin d'assurer leur transmission.

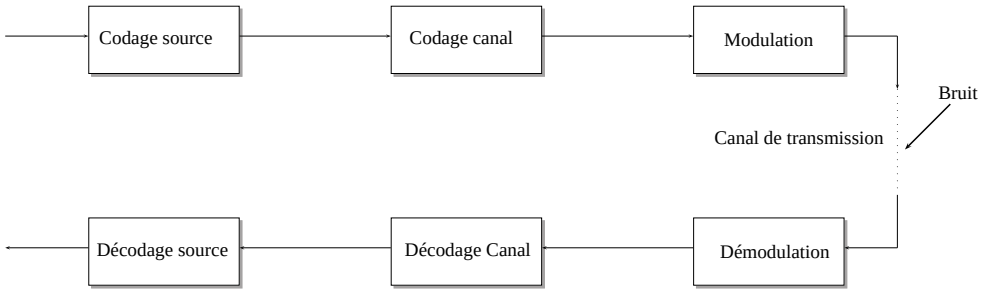


FIGURE 1.1 – Synoptique d'une transmission numérique

Codage source : Le codage de source consiste à éliminer la redondance de manière à réduire la quantité de données à transmettre.

Codage canal : Quel que soit le type de canal de transmission utilisé, le signal transmis est soumis à des perturbations, que l'on appelle bruit, pouvant entraîner une altération des données. Afin d'y remédier, il est d'usage d'ajouter de la redondance à l'information dans le but de détecter puis corriger les erreurs de transmission. Nous utilisons pour cela des codes correcteurs d'erreurs.

Modulation : La modulation consiste à transformer un message en un signal adapté à la transmission sur un support physique.

Une introduction aux codes correcteurs d'erreurs est donnée au Chapitre 3. Nous nous intéressons à présent à différents types de modulations. Nous définissons ici ce qu'est une constellation et introduisons la notion de mapping, avant d'aborder les modèles de canaux de transmission. Ce chapitre est basé sur les ouvrages [19], [39] et [22].

1.1 Modulation

Nous nous intéressons ici à la transmission de données numériques. L'information à véhiculer est donc une séquence binaire, tandis qu'un support physique permet de transmettre une information sous forme d'un signal continu. La modulation consiste alors à transformer l'information binaire en un signal continu adapté au support de transmission.

Nous verrons qu'il existe deux possibilités pour transmettre un signal numérique : la transmission en bande de base ou la modulation d'une onde porteuse. La transmission en bande de base consiste à associer à la séquence binaire d'entrée un signal physique qui est transmis dans une plage de fréquence contenant la fréquence nulle. Elle s'utilise dans le cas d'une transmission filaire. La modulation par onde porteuse consiste à modifier un ou plusieurs paramètres d'une onde porteuse, en fonction du symbole binaire à transmettre (un bloc de bits de longueur donnée), et à l'émettre dans une plage de fréquence donnée qui peut être différente pour différents utilisateurs.

Une démarche commune aux deux méthodes de transmission est l'association d'une information physique et d'un symbole binaire. C'est ce que nous appelons *le mapping*.

1.1.1 Transmission du signal

Transmission en bande de base

La notion de transmission en bande de base ne sera pas utile en tant que telle mais sera utile pour la compréhension du mécanisme de modulation présenté dans le paragraphe suivant.

L'information à transmettre étant une séquence binaire, la première approche est d'associer au bit 0 un état significatif (par exemple une tension positive) et d'associer au bit 1 un autre état significatif (par exemple une tension négative). On appelle cette opération le codage, même si ce terme peut porter à confusion.

Un exemple de codage : le codage NRZ (No Return to Zero) Le codage NRZ associe au bit 0 une valeur α et associe au bit 1 la valeur $-\alpha$. La transmission de la séquence 0110101 se traduira alors par la transmission du signal suivant :

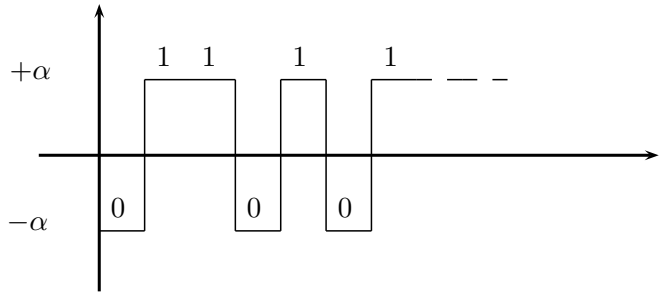


FIGURE 1.2 – Exemple de transmission suivant un codage NRZ

Cet exemple ne constitue pas une règle générale, le bit 0 pouvant être codé par une valeur α positive ou négative et inversement pour le bit 1.

On dit alors que l'ensemble $\mathcal{A} = \{\alpha, -\alpha\}$ est un *alphabet* pour le codage NRZ.

De manière générale, la suite de bits constituant le message est mise sous forme de symboles binaires, en créant des paquets de a bits. Il existe donc 2^a symboles binaires distincts auquels sont associés des valeurs α_i appartenant à un alphabet $\mathcal{A}$ de cardinal 2^a. Le signal émis est de la forme $s(t) = \sum_j \alpha_j g(t - jT)$ où α_j est la $j^{\text{ème}}$ valeur à transmettre et appartient à l'alphabet $\mathcal{A}$ choisi et $g(t)$ est une impulsion élémentaire de durée T.

Dans l'exemple du codage NRZ, l'impulsion utilisée est telle que $g(t) = \begin{cases} 1 & \text{si } 0 \leq t < T \\ 0 & \text{sinon} \end{cases}$. On détaille le calcul de $s(t)$ pour la transmission de la séquence binaire 0110101 dans le tableau ci-dessous :

j	bit à transmettre	α_j	$s(t)$
0	0	$-\alpha$	$-\alpha g(t) = -\alpha, \forall\ 0 \leq t < T$
1	1	α	$\alpha g(t-T) = \alpha, \forall\ T \leq t < 2T$
2	1	α	$\alpha g(t-2T) = \alpha, \forall\ 2T \leq t < 3T$
3	0	$-\alpha$	$-\alpha g(t-3T) = -\alpha, \forall\ 3T \leq t < 4T$
4	1	α	$\alpha g(t-4T) = \alpha, \forall\ 4T \leq t < 5T$
5	0	$-\alpha$	$-\alpha g(t-5T) = -\alpha, \forall\ 5T \leq t < 6T$
6	1	$-\alpha$	$-\alpha g(t-6T) = -\alpha, \forall\ 6T \leq t < 7T$

Modulation par onde porteuse

On définit une onde porteuse par l'expression $p(t) = A_0 cos(2\pi f_0 t + \varphi_0)$ où A_0 est l'amplitude du signal, f_0 sa fréquence et φ_0 sa phase.

La modulation par onde porteuse consiste à transmettre de l'information en modifiant un ou plusieurs paramètres de cette onde porteuse. Le signal en bande de base est alors transposé en fréquence afin d'assurer la transmission dans une bande de fréquence adaptée. C'est-à-dire que le signal en bande de base devient le signal modulant. De plus le signal est transmis à une fréquence donnée qui peut être différente pour différents utilisateurs. Il existe différents types de modulation utilisant des paramètres distincts de la porteuse comme :

- la modulation *ASK (Amplitude Shift Keying)*, une modulation d'amplitude
- la modulation *PSK (Phase Shift Keying)*, une modulation de phase
- la modulation *FSK (Frequency Shift Keying)*, une modulation de fréquence
- la modulation *QAM (Quadrature Amplitude Modulation)*, une modulation de phase et d'amplitude

Nous ne nous intéressons ici qu'aux modulations d'amplitude et/ou de phase. On note a le nombre de bits par symbole binaire. On a donc $M = 2^a$ symboles possibles. Nous supposons ici, pour simplifier

l'écriture des signaux modulés, que l'amplitude A_0 des porteuses est égale à 1.

Modulation ASK Le signal modulé s'écrit

$$s(t) = \sum_j \alpha_j g(t - jT) cos(2\pi f_0 t)$$

En notant

$$s_I(t) = \sum_j \alpha_j g(t - jT)$$

on a

$$s(t) = cos(2\pi f_0 t) s_I(t)$$

Le signal $s_I(t)$ modifie alors l'amplitude de la porteuse $p(t) = cos(2\pi f_0 t)$.
On prendra a priori comme alphabet $\{\pm\alpha, \pm 3\alpha, \ldots, \pm(M-1)\alpha\}$.

Modulation PSK Le signal modulé s'écrit

$$s(t) = \sum_j cos(2\pi f_0 t + \alpha_j) g(t - jT)$$

En notant

$$s_I(t) = \sum_j cos(\alpha_j) g(t - jT)$$

et

$$s_Q(t) = \sum_j sin(\alpha_j) g(t - jT)$$

on a

$$s(t) = cos(2\pi f_0 t) s_I(t) - sin(2\pi f_0 t) s_Q(t)$$

Le signal $s_I(t)$ modifie l'amplitude de la porteuse $p(t) = cos(2\pi f_0 t)$ tandis que le signal $s_Q(t)$ modifie l'amplitude de la porteuse $sin(2\pi f_0 t) = cos(2\pi f_0 t + \frac{\pi}{2})$. On appelle la première porteuse la *porteuse en phase*, la seconde la *porteuse en quadrature*.
On pourra prendre, pour $M = 4$, l'un des deux alphabets suivants $\{ \frac{i\pi}{4} \, | i = 1,\ 3,\ 5,\ 7\}$ ou $\{ \frac{i\pi}{2} \, | k = 0,\ 1,\ 2,\ 3\}$. On prendra pour $M = 8$ l'alphabet $\{ \frac{i\pi}{4} \, | i = 0,\ 1,\ 2, \ldots,\ 7\}$.

Modulation QAM La modulation d'amplitude et de phase simultanée nécessite l'utilisation de deux alphabets $\mathcal{A}$ et $\mathcal{B}$ de cardinal respectif 2^b et $2^{b'}$. À un symbole binaire de $b + b'$ bits on associe un couple $(\alpha_i, \beta_{i'}) \in \mathcal{A} \times \mathcal{B}$. Et le signal modulé s'écrit

$$s(t) = \sum_j \alpha_j cos(2\pi f_0 t + \beta_j) g(t - jT)$$

$(\alpha_j, \beta_j) \in \mathcal{A} \times \mathcal{B}$ étant la $j^{\text{ème}}$ valeur à transmettre.
En écrivant

$$s_I(t) = \sum_j \alpha_j cos(\beta_j) g(t - jT)$$

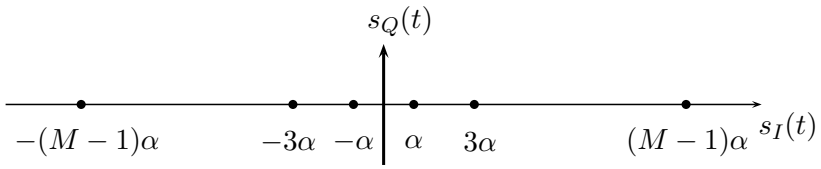


FIGURE 1.3 – Constellation d'une modulation M-ASK ayant pour alphabet $\{\pm\alpha, \pm 3\alpha, \ldots, \pm(M-1)\alpha\}$

et

$$s_Q(t) = \sum_j \alpha_j sin(\beta_j) g(t - jT)$$

on a

$$s(t) = cos(2\pi f_0 t) s_I(t) - sin(2\pi f_0 t) s_Q(t)$$

Comme pour la modulation de phase, $s_I(t)$ et $s_Q(t)$ modifient respectivement l'amplitude de la porteuse en phase et de la porteuse en quadrature.

Les trois modulations présentées ici s'écrivent sous la forme $s(t) = cos(2\pi f_0 t) s_I(t) - sin(2\pi f_0 t) s_Q(t)$ avec des expressions différentes pour $s_I(t)$ et $s_Q(t)$ (avec $s_Q(t) = 0$ pour une modulation ASK). À ce titre on peut classer chacune de ces modulations comme des cas, particuliers ou non, de modulation de type QAM.

Nous étudierons pour ces modulations les divers mappings possibles. Nous allons voir à présent la notion de constellation et leur forme pour ces modulations.

On note $M - ASK$, $M - PSK$ et $M - QAM$ des modulations à $M = 2^a$ points, c'est-à-dire admettant $log_2 M$ bits par symbole binaire.

1.1.2 Constellations

Pour un temps t_j fixé, il existe (α_j, β_j) tel que :

$\begin{cases} s_I(t_j) & = \alpha_j \\ s_Q(t_j) & = 0 \end{cases}$ pour une modulation ASK

$\begin{cases} s_I(t_j) & = \cos(\alpha_j) \\ s_Q(t_j) & = \sin(\alpha_j) \end{cases}$ pour une modulation PSK

$\begin{cases} s_I(t_j) & = \alpha_j \cos(\beta_j) \\ s_Q(t_j) & = \alpha_j \sin(\beta_j) \end{cases}$ pour une modulation QAM

On représente alors l'ensemble des $s_I(t_j)$ et $s_Q(t_j)$ dans le plan, formant ainsi une *constellation*, représentée par la FIGURE 1.3 dans le cas d'une modulation M-ASK.

Définition 1.1.1 (Constellation) *Une constellation est la représentation des différents états de la modulation*

On peut faire de même pour les modulations de type PSK et QAM. On obtient pour une modulation PSK un ensemble de points répartis sur un cercle. (Le rayon du cercle étant déterminé par l'amplitude A_0 de la porteuse, que nous avons supposé normalisée à 1 afin de simplifier l'écriture des signaux modulés). Un exemple de constellation PSK pour un nombre de bits par symbole égal à 3 est donné par la FIGURE 1.5.

FIGURE 1.4 – Constellation d'une modulation 4-PSK

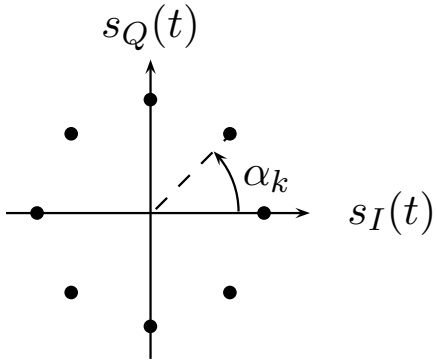


FIGURE 1.5 – Constellation 8-PSK

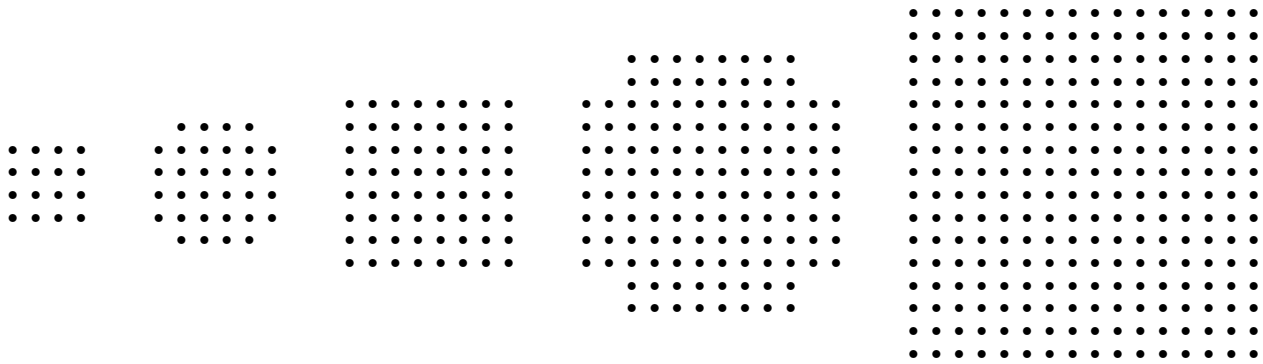


FIGURE 1.6 – Constellations correspondant à des modulations (de gauche à droite) : 16-QAM, 32-QAM, 64-QAM, 128-QAM et 256-QAM

Le choix de l'alphabet (ou des alphabets) détermine la forme de la constellation. Les modulations 32 et $128-QAM$ sont d'ailleurs ici cruciformes contrairement aux modulations à 2^a points avec a pair.

Nous étudions ici les mappings des constellations données par la FIGURE 1.6 pour les modulations QAM de 16 à 256 points, par la FIGURE 1.5 pour une constellation à 8 points et enfin par la FIGURE 1.4 pour une constellation à 4 points, soit de 2 à 8 bits par symbole binaire. La représentation sous forme de constellation sera à présent l'unique représentation pour une modulation donnée.

1.2 Mapping

La modulation est l'opération qui permet de convertir une séquence d'information binaire en un signal adapté au support de transmission. Le nombre de bits que l'on souhaite transmettre à chaque intervalle de temps détermine la taille de l'alphabet à utiliser et donc le nombre de points de la constellation pour une modulation donnée.

Définition 1.2.1 (Mapping 1) *On appelle mapping l'association réalisée entre un symbole binaire et un élément de l'alphabet correspondant à la modulation choisie.*

Un exemple de mapping

Prenons une modulation $2-ASK$ d'alphabet $\mathcal{A} = \{-\alpha, \alpha\}$, on peut définir un mapping en associant au bit 0 la valeur $-\alpha$, et au bit 1 la valeur α. On peut représenter cette association directement sur la constellation. Mais un second mapping est possible en faisant l'association inverse, c'est-à-dire en associant au bit 0 la valeur α et au bit 1 la valeur $-\alpha$.

On peut alors considérer l'opération de mapping comme l'association d'un point de la constellation et d'un symbole binaire.

Définition 1.2.2 (Mapping 2) *Soit $\mathcal{C}$ une constellation à $M = 2^a$ points. On appelle mapping toute bijection, f, de $\mathcal{C}$ dans $\{0,1\}^a$, qui à un point P de $\mathcal{C}$ associe un symbole binaire de longueur a.*

Naturellement d'après la définition 1.2.2, le nombre de mappings possible pour une constellation donnée est de $(2^a)!$. Ce nombre est décrit pour différentes tailles de constellations par la Table 1.1.

Table 1.1 – Nombre de mappings pour $a = 2, 3, 4$

a	Nombre de mappings
2	24
3	40 320
4	$> 2 * 10^{13}$

Nous utiliserons à présent cette définition du mapping. Notons qu'il existe cependant certains mappings ayant des propriétés permettant de limiter les erreurs de transmission, comme les mappings de type Gray [30] que nous utiliserons par la suite.

1.3 Canal de transmission

Lors de la transmission d'un signal, celui-ci est soumis à des atténuations et déformations propres au canal (par exemple dû à la distance entre deux antennes pour une communication sans-fil). Il est également soumis à des perturbations externes que l'on appelle couramment du bruit.

Différents supports de transmission permettent de véhiculer une information (câble électrique, fibre optique, liaison sans-fil ...). Chaque type de transmission possède un modèle de propagation qui lui est propre. Une liaison sans-fil est en particulier sensible aux obstacles entraînant des réflexions, diffractions du signal. À ce titre, ces liaisons sont les plus difficiles à modéliser et il existe différents modèles de canaux de transmission.

Les deux modèles les plus couramment utilisés sont le canal à bruit blanc additif gaussien et le canal binaire symétrique. Ils sont tous deux des canaux dits sans mémoire. Il existe cependant d'autres modèles simulant notamment les canaux multi-trajets tels que les canaux de Rayleigh et de Rice, utilisés pour modéliser les radiocommunications, auxquels nous ne nous intéressons pas ici.

Le canal à bruit blanc additif gaussien : le signal reçu $r(t)$ s'écrit comme l'addition du signal émis $s(t)$ et du bruit blanc gaussien $b(t)$. $b(t)$ est modélisé, dans une bande de fréquence limitée, par un processus aléatoire centré de moyenne nulle, de variance σ^2, dont la densité de probabilité est :

$$p(b) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-b^2}{2\sigma^2}}$$

La qualité d'une transmission analogique se mesure à l'aide du rapport signal à bruit, noté *SNR* (*Signal to Noise Ratio*) et exprimé généralement en décibels. Il mesure le rapport de puissance entre le signal et le bruit. Plus le bruit est élevé plus le rapport signal à bruit diminue. La qualité d'une transmission numérique se mesure quant à elle à l'aide de deux outils distincts qui sont le *rapport signal à bruit par bit*, Eb/N_0, et le *Taux d'Erreur Binaire* (*TEB* ou *BER* pour *Bit Error Rate*). Le rapport signal à bruit par bit mesure le rapport entre l'énergie véhiculée par un bit, Eb, et la densité spectrale de puissance du bruit, N_0.

Le taux d'erreur binaire définit la probabilité pour un bit d'être faux :

$$BER = \frac{nombre\ de\ bit\ faux}{nombre\ de\ bits\ transmis}$$

Il est possible de calculer le taux d'erreur binaire en fonction du rapport signal à bruit par bit. Ce calcul dépend fortement de la modulation choisie et du mapping utilisé. On trouve ce calcul dans la littérature pour différents types de modulation ([22, 39, 40]).

Le canal binaire symétrique : les entrées et sorties du canal binaire symétrique sont des valeurs binaires. Il est caractérisé par la probabilité d'erreur p qu'un bit (0 ou 1) soit modifié (en son opposé) lors de la transmission. La probabilité qu'un bit ne soit pas modifié lors de la transmission est alors $1 - p$. Il est représenté par la FIGURE 1.7.

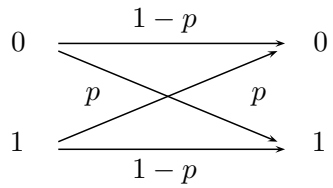


FIGURE 1.7 – Canal binaire symétrique de probabilité d'erreur p

On considère que le canal binaire symétrique regroupe alors la modulation, le canal et la démodulation dans le schéma d'un système de communication.

Il est courant de considérer deux modèles de canaux de transmission :
- Un modèle dit *interne* AWGN (*Additive White Gaussian Noise*), pour lequel est calculé le taux d'erreur binaire (en fonction de la modulation et du mapping).

- Un modèle dit *externe* BSC (*Binary Symetric Channel*) ayant pour probabilité d'erreur p la probabilité définie par le BER du canal interne.

Ce qui rend transparent la modulation et la démodulation pour le codage canal.

1.4 Démodulation

On appelle *démodulation* l'extraction des paramètres α_k et β_k contenus dans les composantes $s_I(t)$ et $s_Q(t)$ d'un signal émis $s(t)$. Le signal reçu $s'(t)$ correspond au signal $s(t)$ transformé par le bruit du

canal de transmission. Les composantes $s'_I(t)$ et $s'_Q(t)$ reçues correspondent alors à des modifications des composantes $s_I(t)$ et $s_Q(t)$ et la donnée d'un couple $(s'_I(t), s'_Q(t))$ donnera lieu à interprétation, dans le sens où l'on évaluera la distance entre le point de coordonnées $(s'_I(t), s'_Q(t))$ et les points de la constellation.

En effet, l'objectif est de retourner les symboles les plus probablement émis. Un détecteur a *maximum de vraisemblance* (*ML* pour *Maximum Likelihood*) fait l'hypothèse que les symboles émis sont équiprobables. Il calcule alors la distance euclidienne entre les symboles reçus et les symboles possibles. La décision se fait, pour un symbole reçu donné, en faveur du symbole de l'alphabet le plus proche au sens de la distance euclidienne.

De cette manière une erreur de décision peut être commise si le point de coordonnées $(s'_I(t), s'_Q(t))$ est plus proche géométriquement d'un point de la constellation différent de celui correspondant au signal émis à l'instant t. Lorsqu'une telle erreur se produit, le plus souvent, un point de la constellation est interprété comme étant un de ses plus proches voisins.

Chapitre 2

Mappings

Le mapping est l'association réalisée entre les points d'une constellation et les différents symboles binaires possibles pour une taille de constellation donnée. La taille d'une constellation en désigne le nombre de points, elle détermine alors le débit de transmission. À une constellation de $M = 2^a$ points, est associé un ensemble de 2^a symboles binaires de longueur a. Nous utilisons tout au long de ce manuscrit la notation a pour désigner le nombre de bits par symbole binaire du mapping. Nous rappelons la définition de mapping comme étant une bijection de la constellation $\mathcal{C}$ dans $\{0,1\}^a$.

Définition 2.0.1 (Mapping) *Soit $\mathcal{C}$ une constellation à $M = 2^a$ points. On appelle mapping toute bijection f, de $\mathcal{C}$ dans $\{0,1\}^a$, qui à un point P de $\mathcal{C}$ associe un symbole binaire de longueur a.*

Nous rappelons que le nombre de mappings, pour une constellation donnée, est le nombre de permutations de $\{0,1\}^a$, c'est-à-dire $(2^a)!$. Nous pouvons alors décrire l'ensemble des mappings en décrivant l'ensemble de ces permutations, pour de petites tailles de constellations (*ie* $a = 2$ ou $a = 3$).

Il existe cependant des mappings permettant de limiter le *Taux d'Erreur Binaire* (TEB ou BER pour *Bit Error Rate*) en réception. Il définit le nombre d'erreurs de transmission ou d'écriture pour une information numérique. Un taux d'erreur de 10^{-2} signifie qu'un bit sur 100 est erroné. Les mappings Gray [30] permettent de limiter les erreurs de transmission et donc de diminuer le BER.

Définition 2.0.2 (Mapping Gray) *Soit $\mathcal{C}$ une constellation à $M = 2^a$ points. Soit f un mapping de $\mathcal{C}$. On dit que f est un mapping Gray si pour tous points P_i, P_j de $\mathcal{C}$, voisins pour la distance euclidienne, alors $f(P_i)$ et $f(P_j)$ sont à distance de Hamming de 1.*

Le terme *mapping* est essentiellement utilisé pour désigner l'association des symboles binaires et des points d'une constellation en dimension 2. Il est alors courant d'utiliser le terme *codage* dans le cas d'une constellation en dimension 1.

La décision réalisée lors de la démodulation consiste à associer le point reçu, à l'instant t, au point de la constellation le plus proche, au sens de la distance euclidienne. Si le symbole binaire, correspondant au point de la constellation le plus proche, est à distance de Hamming de 1, du symbole binaire émis à l'instant t, alors un seul bit est erroné à cet instant.

F. Gray a introduit en 1953 ce type de codage. Il a pour but de limiter les erreurs de transmission. Le code Gray impose des éléments successifs ayant un seul bit différent. Nous pouvons par exemple coder les entiers de 0 à 7 en binaire naturel : 000, 001 , 010 , 011, 100, 101, 110, 111 ou selon le codage

Gray : 000, 001, 011, 010, 110, 111, 101, 100. Le codage Gray assure de plus un minimum d'opérations lors du passage d'un élément au suivant. Ce codage, également appelé binaire réfléchi, est le codage initialement introduit par F. Gray.

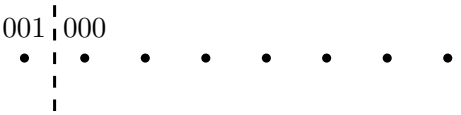
Il existe d'autres codages que nous pouvons qualifier de codage Gray puisqu'ils respectent la distance de Hamming de 1 entre deux symbole successifs. Nous voyons en effet dans ce chapitre différents types de codage Gray, réfléchis ou non réfléchis et de quelle façon les générer d'après les travaux de E.N. Gilbert [29].

Ces codages sont utilisés pour construire des mappings Gray pour des constellations PSK, QAM carrées. Les travaux de R.D. Wesel, X. Liu, J.M Cioffi et C. Komminakis [56] ont permis de décrire la méthode de construction de mappings Gray à partir du produit direct de deux codages Gray. Ainsi il nous est possible d'énumérer tous les mappings Gray. Ils ont également mis en avant l'impossibilité de construire un mapping Gray pour une constellation en croix. Nous décrivons alors des méthodes issues essentiellement de ces travaux [32, 56] d'une part et des travaux de J.G. Smith [45] d'autre part, permettant de définir des mappings approchant le critère de Gray. Ils sont communément appelés des mappings *quasi-Gray*. On pourra également citer les travaux de P.K. Vitthaladevuni, M.S. Alouini et J.C. Kieffer [51, 52] dans ce domaine. Cependant il n'existe pas de définition formelle des mappings quasi-Gray au sens où tous les mappings peuvent être considérés comme quasi-Gray avec des caractéristiques se rapprochant plus ou moins du critère de Gray.

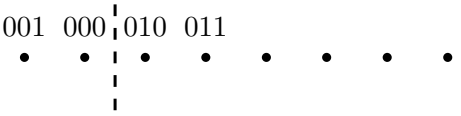
2.1 Codages Gray réfléchis

Nous définissons ici un codage Gray réfléchi et sa méthode de construction. Un tel codage est construit comme suit : à l'étape 0 on choisit un symbole parmi les 2^a possibles. Ce symbole sert de labellisation pour le point le plus à gauche de la constellation. À l'étape i on choisit un bit à complémenter (non utilisé pendant les étapes précédentes). À cette étape, 2^i points sont déjà labellisés. Nous labellisons alors les 2^i suivants par symétrie en complémentant le bit précédemment choisi. L'exemple d'un codage Gray réfléchi pour une constellation unidimensionnelle à 8 points est donné ci-dessous, il permet d'illustrer cette construction.
Le symbole d'origine est choisi :

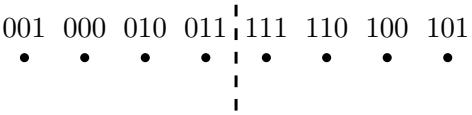
Le 3ème bit est complémenté pour obtenir le symbole suivant :



Les deux symboles suivants sont obtenus par symétrie et en complémentant le 2ème bit de chaque symbole.



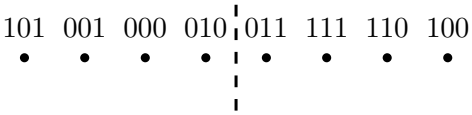
Les derniers symboles s'obtiennent alors en complémentant le 1er bit de chaque symbole, en utilisant également une symétrie :



Cette construction permet de décrire $2^3.3!$ codages Gray réfléchis différents : 2^3 pour le choix du symbole d'origine et 3! pour le choix de l'ordre des bits à complémenter. Plus généralement, pour une constellation à 2^a points il existe $2^a.a!$ codages Gray réfléchis.

2.2 Codages Gray non réfléchis

F. Gray a introduit les codes Gray réfléchis. Il existe cependant des codages également appelés codages Gray qui sont de type non réfléchis. Le codage donné ci-dessous en est un exemple. Nous pouvons cependant remarquer que les symboles situés aux extrémités sont à distance de Hamming de 1. Nous appelons de tels codages des codages Gray cycliques. Les codages Gray réfléchis sont d'ailleurs des codages cycliques.



Il existe donc trois familles de codages Gray : les codages Gray réfléchis, les codages Gray cycliques et enfin les codages qui ne sont ni réfléchis ni cycliques.

2.3 Codages Gray

E.N. Gilbert [29] a décrit en 1957 une méthode de génération de l'ensemble des codages Gray basée sur la description de chemins dans un graphe. On définit Q_a comme étant un graphe à 2^a sommets où deux sommets sont reliés par une arête si leurs labels respectifs ne diffèrent que d'un bit. Une représentation de ces graphes dans le cas $a = 2, 3, 4$ est donnée par la FIGURE 2.1, où chaque sommet est labellisé par l'écriture binaire de son numéro.

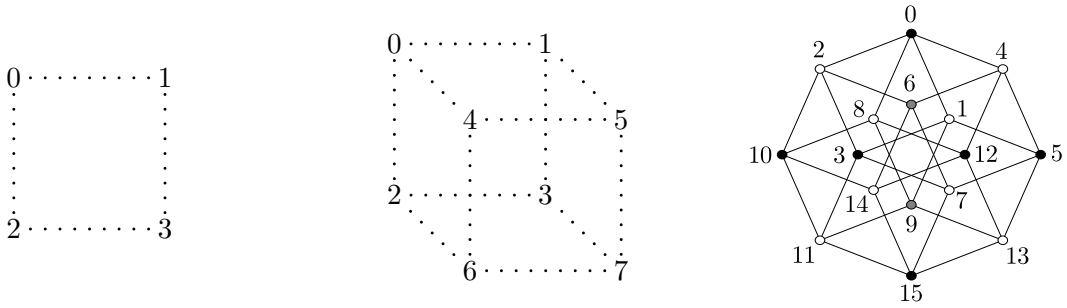


FIGURE 2.1 – Représentation des graphes Q_a, de gauche à droite $a = 2, 3, 4$

Les cubes ainsi construits permettent d'obtenir d'une part le nombre de codages Gray pour $a \in [\![1, 4]\!]$, et d'autre part les codages Gray en eux-mêmes. En effet, la description d'un codage Gray est équivalente à la description d'un chemin, de longueur 2^a de Q_a, ne parcourant qu'une seule fois chacun des sommets.

Le chemin défini par les sommets 0, 1, 3, 2, dans Q_2, permet d'obtenir, par le biais de ses labels, le codage Gray suivant : 00, 01, 11, 10. Le chemin 0, 2, 3, 1 définit également un codage Gray. Il existe deux chemins de longueur 4 parcourant chacun des sommets de Q_2 et ayant pour origine le sommet 0, ainsi que pour chacune des origines possibles. Il y a donc 8 codages Gray à 2 bits. Nous pouvons déterminer le nombre de codages Gray cycliques pour $a \in [\![1, 4]\!]$ en posant la contrainte suivante : les chemins retenus sont ceux ayant pour extrémités des sommets voisins.

Le nombre de codage Gray obtenu par simulation est :
- pour $a = 2$: 8 dont 8 cycliques
- pour $a = 3$: 144 dont 96 cycliques
- pour $a = 4$: 91392 dont 43008 cycliques

Les codages Gray réfléchis sont utilisés pour la construction de mappings quasi-Gray, à savoir des mappings respectant "au maximum" le critère de Gray, dans le cas de constellation comme les $32-QAM$ et $128-QAM$. Les codages Gray cycliques sont utilisés pour la construction de mappings Gray valides pour les constellations PSK. L'ensemble des codages Gray est également utilisé dans le cas des constellations QAM carrées ou rectangulaires.

2.4 Mappings Gray PSK

Une constellation PSK est un ensemble de points dans le plan répartis sur un cercle de rayon A (l'amplitude du signal). Un codage Gray cyclique permet donc de définir un mapping Gray pour ce type de constellation en "enroulant" ce code sur lui-même. Ainsi il existe 96 mappings Gray PSK, dont 48 sont des mappings issus de codages Gray réfléchis.

Nous remarquons que la notion de points voisins dans une constellation diffère selon sa configuration. Un point a uniquement deux voisins dans une constellation PSK tandis qu'il peut en avoir jusqu'à quatre dans une constellation QAM.

2.5 Mappings Gray QAM

Nous décrivons ici la méthode de génération d'un mapping Gray, pour une constellation QAM carrée ou rectangulaire, issue des travaux Wesel, Liu, Cioffi, Komminakis. Cette méthode utilise le produit direct de deux codages Gray. Elle permet notamment la génération de l'ensemble des mappings Gray. Soit une constellation QAM à $2^a \times 2^b$ points. Soient deux codages Gray à 2^a et 2^b points respectivement. L'un est utilisé verticalement, l'autre horizontalement. Nous pouvons alors labelliser un point en adjoignant ses labels verticaux et horizontaux correspondants. En vue de générer l'ensemble des mappings Gray possibles sans parcourir deux fois le même mapping, il faut fixer la contrainte de ne pas inverser l'ordre des bits issus d'un même codage. Il y a alors $\binom{a+b}{a}$ mappings possibles pour deux codages horizontaux et verticaux fixés. Deux exemples de tels mappings sont donnés par la FIGURE 2.2.

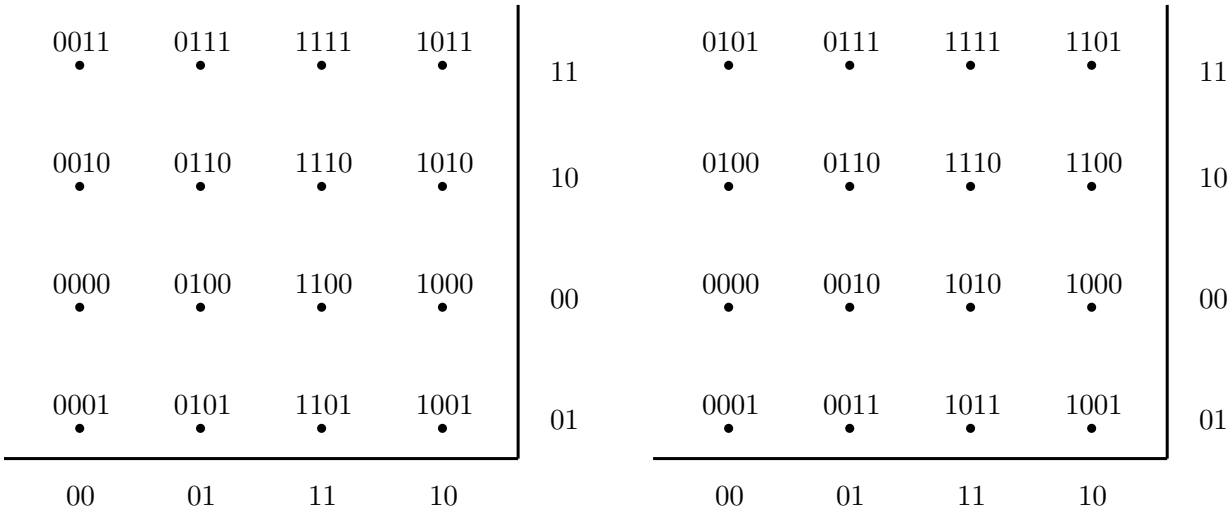


FIGURE 2.2 – Deux mappings Gray construits par produit direct de codes Gray

On note n_l (respectivement n_c) le nombre de codages Gray lignes (respectivement colonnes). Le nombre de mappings Gray pour une constellation QAM rectangulaire de taille $2^a \times 2^b$ est de

$$n_l \,.\, n_c \,.\, \binom{a+b}{a}$$

Le nombre de mappings Gray pour les constellations $4-PSK$, $8-PSK$, $16-QAM$, $64-QAM$ et enfin $256-QAM$ est donné par la TABLE 2.1.

Constellation	Nombre de mappings Gray
$4-PSK$	8
$8-PSK$	96
$16-QAM$	384
$64-QAM$	414 720
$256-QAM$	584 674 836 480

TABLE 2.1 – Nombre de mappings Gray

La génération de mappings Gray réfléchis pour les constellations QAM carrées ou rectangulaires peut se faire par une méthode similaire à celle définie pour les codages Gray. Elle peut également se faire par produit direct de deux codages Gray réfléchis. Les mappings Gray minimisent les erreurs de réception. Ils sont potentiellement très utilisés et sont à ce titre des cibles privilégiées pour la suite de notre travail. Il est impossible d'utiliser des mappings Gray pour les constellations en croix, $32-QAM$ et $128-QAM$, nous nous intéressons donc aux mappings dits quasi-Gray.

2.6 Mappings quasi-Gray

Wesel & al. ont démontré qu'il n'est pas possible de construire de mappings Gray pour les constellations en croix, notamment $32-QAM$ et $128-QAM$. Il est cependant possible d'utiliser des mappings dits quasi-Gray respectant autant que possible le critère de Gray. Il existe dans la littérature différents

mappings quasi-Gray. Le but de cette section est d'en présenter quelques uns. La qualité d'un mapping quasi-Gray peut être jugée grâce aux deux critères suivants : la *pénalité de Gray moyenne*, la *pénalité de Gray maximum*. Comme il n'existe pas de définition formelle des mappings quasi-Gray, en pratique, on recherche des mappings qui minimisent les pénalités de Gray moyenne et maximum.

Définition 2.6.1 (Pénalité de Gray moyenne) *On définit la pénalité de Gray moyenne, notée G_p, comme le nombre moyen de bits différents entre deux symboles binaires voisins.*

Définition 2.6.2 (Pénalité de Gray maximum) *On définit la pénalité de Gray maximum, notée G_{p^k}, comme le nombre maximum de bits différents entre deux symboles binaires voisins.*

Mappings $32-QAM$

Méthode par déplacement de labels

La première approche se base sur la définition de mappings quasi-Gray à partir de mappings rectangles. Deux méthodes sont schématisées ici, la première issue de [56] est vue dans la FIGURE 2.3, la seconde issue de [45] est vue dans la FIGURE 2.4.

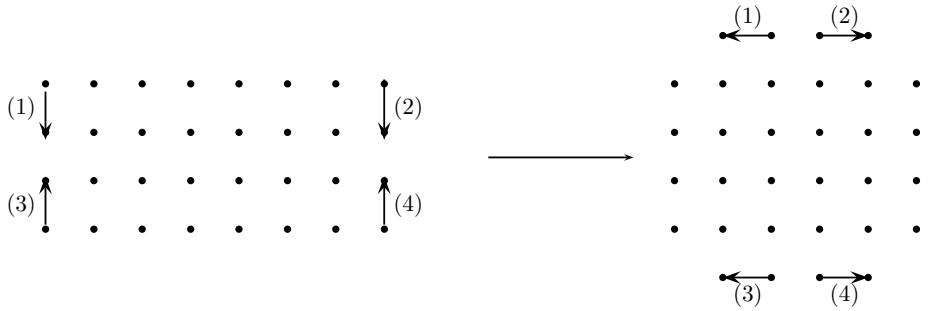


FIGURE 2.3 – Construction d'un mapping $32-QAM$ par la méthode de *Wesel & al.*

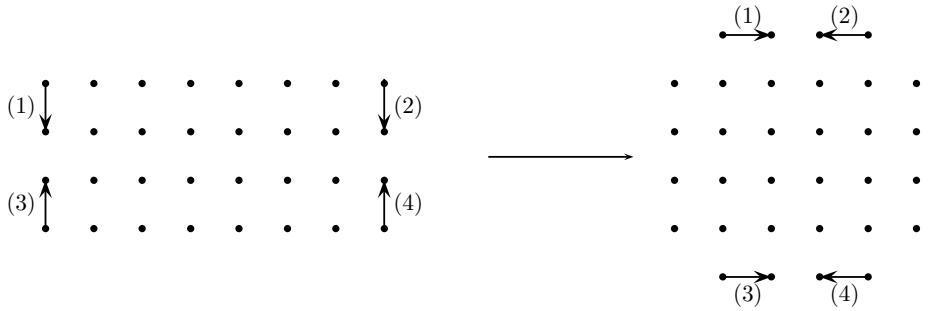


FIGURE 2.4 – Construction d'un mapping $32-QAM$ par la méthode de *J.G. Smith*

Méthode par définition de transitions

La seconde approche se base sur la notion de transition. On appelle transition entre deux symboles binaires s_1 et s_2, un vecteur v, de taille a tel que $s_2 = s_1 + v$. Les transitions pour les constellations $32-QAM$ sont des vecteurs de taille 5. *Wesel & al.* ont étudié ces transitions pour en déterminer le nombre nécessaire à l'obtention d'un mapping $32-QAM$, ainsi que les agencements décrivant des mappings quasi-Gray raisonnables. Il en résulte qu'il est possible de labelliser entièrement une

constellation, pour a impair, avec $a+1$ transitions : les transitions unitaires (de poids de Hamming 1) et une transition combinaison linéaire des précédentes.

Pour les mappings $32-QAM$, ils proposent ainsi six schémas (FIGURE 2.5). La construction d'un mapping avec cette méthode consiste à choisir le symbole binaire correspondant à l'un des points de la constellation et à appliquer les transitions successivement afin d'obtenir la labellisation de l'ensemble de la constellation.

Nous avons matérialisé les transitions non unitaires en les soulignant de gris dans ces schémas de construction. Cela permet de remarquer les positions où les contraintes de Gray ne sont pas respectées. Le nombre de ces positions détermine en partie la qualité du mapping quasi-Gray ainsi défini. Le nombre de mappings quasi-Gray obtenus grâce à l'un des six schémas correspond à 5!.32 (5! pour le choix des transitions e_1 à e_5, e_6 étant fixe, 32 pour le choix du symbole d'origine).

Correspondance des méthodes

Le premier schéma de la FIGURE 2.5 (en haut à gauche) correspond à un mapping Gray réfléchi pour lequel les points situés aux extrémités droite et gauche ont été déplacés. Ce schéma correspond à la construction de la FIGURE 2.3. On peut également représenter la construction de la FIGURE 2.4 sous forme de schéma de transition (FIGURE 2.6) à partir du schéma de transition d'un mappings rectangulaire réfléchi (FIGURE 2.7).

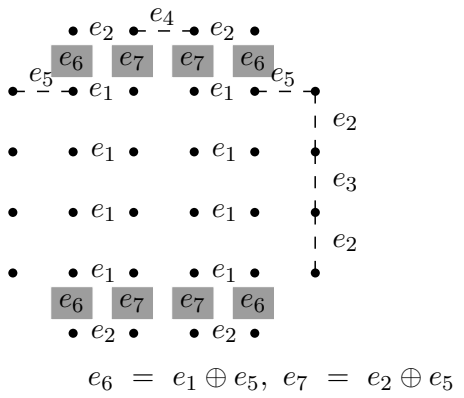


FIGURE 2.6 – Schéma de transition des mappings quasi-Gray de *J.G. Smith*

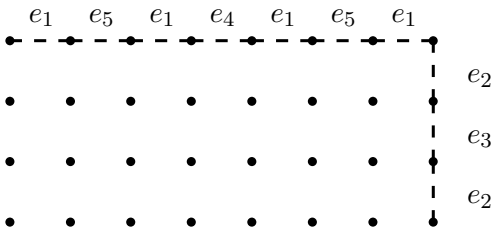
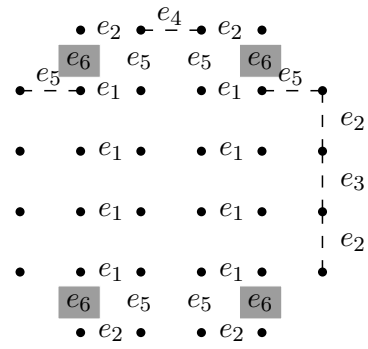
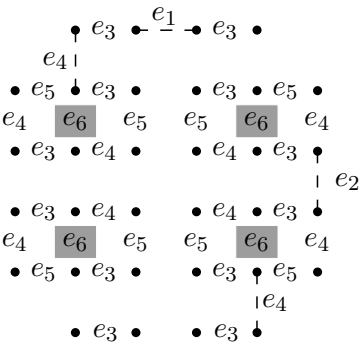


FIGURE 2.7 – Schéma de transition d'un mappings Gray réfléchi rectangulaire $32-QAM$



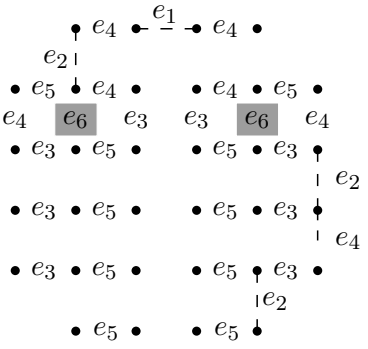
$e_6 = e_1 \oplus e_2 \oplus e_5$

(a) Construction 1



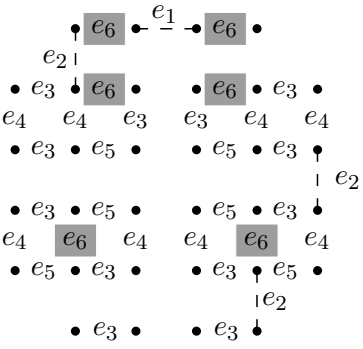
$e_6 = e_3 \oplus e_4 \oplus e_5$

(b) Construction 2



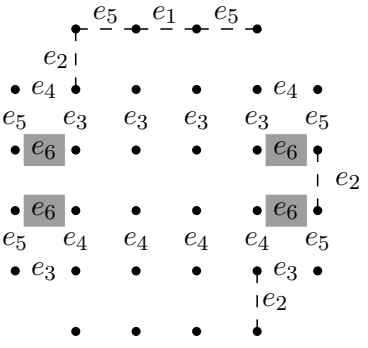
$e_6 = e_3 \oplus e_4 \oplus e_5$

(c) Construction 3



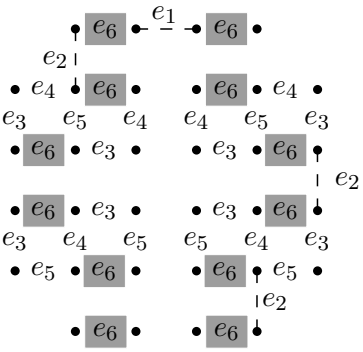
$e_6 = e_3 \oplus e_4 \oplus e_5$

(d) Construction 4



$e_6 = e_3 \oplus e_4 \oplus e_5$

(e) Construction 5



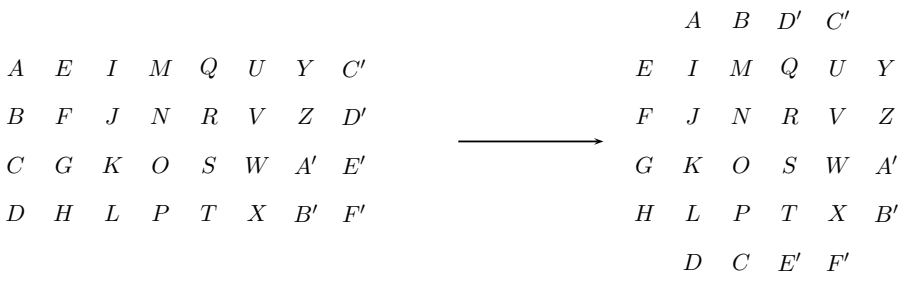
$e_6 = e_3 \oplus e_4 \oplus e_5$

(f) Construction 6

FIGURE 2.5 – Six schémas de transitions pour générer des 32-QAM quasi-Gray. Les traits en pointillés signifient que toute la ligne, ou toute la colonne, est construite avec la même transition

Pénalité de Gray

Nous pouvons calculer pour chacun des schémas présentés ici la pénalité de Gray moyenne. Ainsi pour la méthode de la FIGURE 2.4 la pénalité moyenne est de $\frac{7}{6}$:



Avec les notations précédentes, les points J, N, R, V, K, O, S, W puis E, F, G, H, Y, Z, A', B' ont une pénalité moyenne de 1. En effet, ils respectent le critère de Gray. Les points A, C', D et F'ont une pénalité de $\frac{3}{2}$, les points B, D', C, E' ont une pénalité de $\frac{4}{3}$ et enfin les points restants ont une pénalité de $\frac{5}{4}$. La pénalité moyenne est donc de $\frac{7}{6}$.

De la même manière, il est possible de calculer les pénalités de Gray pour les autres méthodes de construction, elles sont données par la TABLE 2.2

FIGURE	2.5a	2.5b	2.5c	2.5d	2.5e	2.5f	2.4
G_p	$\frac{19}{16}$	$\frac{9}{8}$	$\frac{17}{16}$	$\frac{118}{96}$	$\frac{110}{96}$	$\frac{71}{48}$	$\frac{7}{6}$
G_p^k	3	3	3	3	3	3	2

TABLE 2.2 – Pénalité de Gray de mappings 32 – QAM

Mappings 128 – QAM

Nous présentons ici deux types de mappings quasi-Gray pour les constellations 128 – QAM cruciformes. Ces deux méthodes se basent sur la modification de mappings Gray rectangulaires. La seconde méthode est issue des travaux de P.K. Vitthaladevuni, M.S. Alouini et J.C. Kieffer [52]. Elles sont données par la FIGURE 2.8 et la FIGURE 2.9.

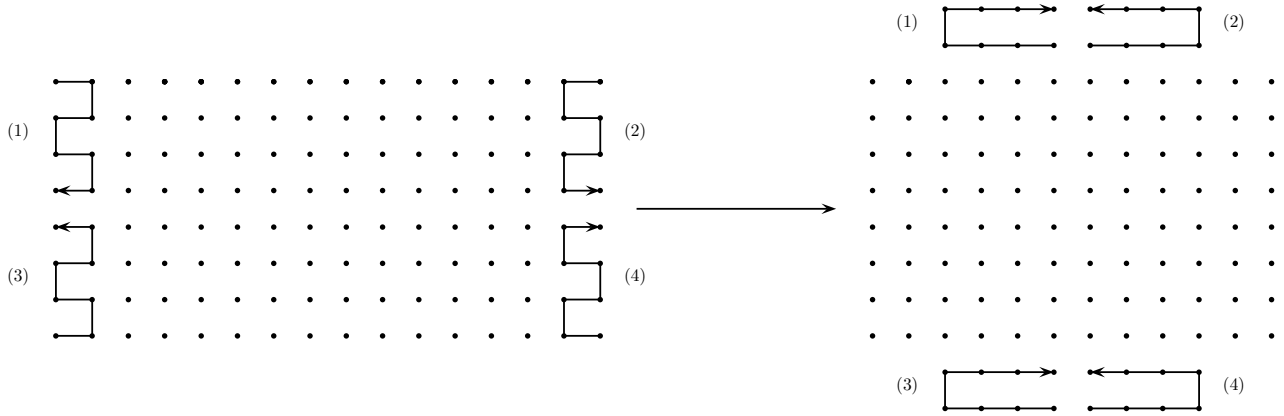


FIGURE 2.8 – Construction d'un mapping $128 - QAM$ par la méthode de *Wesel & al.*

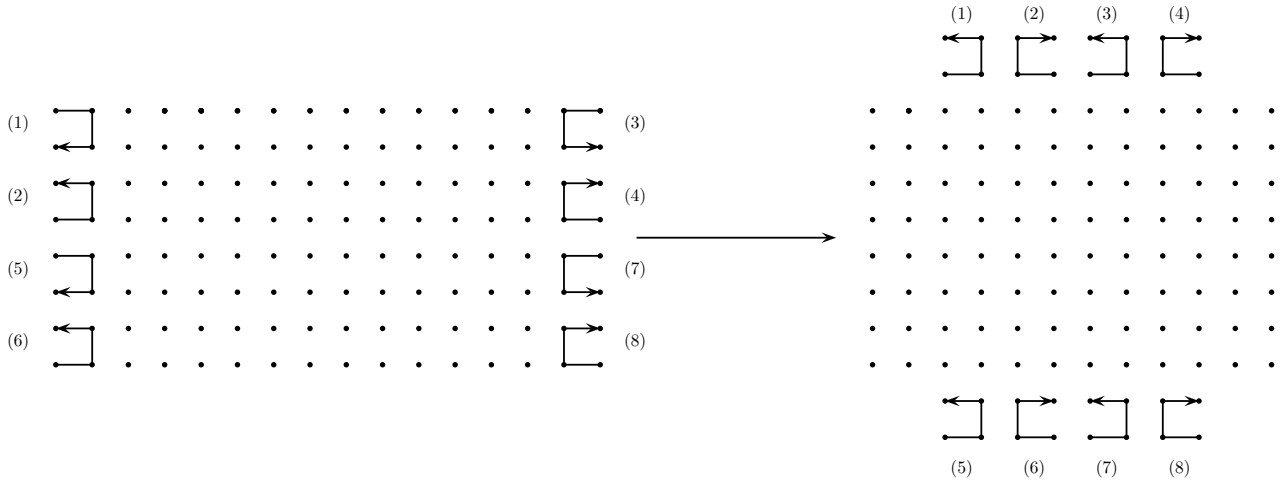


FIGURE 2.9 – Construction d'un mapping $128 - QAM$ par la méthode de *Vitthaladevuni & al.*

La pénalité de Gray du mapping présenté FIGURE 2.9 est donnée dans [52] est vaut

$$1 + \frac{1}{\sqrt{2 * 128}} + \frac{1}{3 * 128} \simeq 1.0651$$

Ces méthodes s'étendent au cas des constellations $512 - QAM$. Dans la suite de ce travail nous utilisons les mappings quasi-Gray définis à partir de mappings Gray rectangles. Elles sont efficaces dans le sens où elles nécessitent peu de transformations alors que nous sommes déjà en mesure de générer les mappings Gray rectangulaires.

Chapitre 3

Codes Convolutifs et Codes de Reed-Solomon

Ce chapitre introductif a pour objectif de donner des notions de théorie des codes. Notamment nous présentons ce que sont les codes convolutifs et les codes de Reed-Solomon. Ces deux types de codes sont ceux utilisés dans le Chapitre 6 pour la reconnaissance du mapping bien que nous nous soyons essentiellement concentrés sur le cas des codes convolutifs. Nous présentons en effet de plus une méthode de détection des codes convolutifs dans le Chapitre 4.

3.1 Introduction aux codes correcteurs d'erreurs

Les codes correcteurs d'erreurs sont des systèmes visant à réduire les erreurs de transmission. Le codage de l'information binaire à transmettre, par ce type de systèmes, permet d'ajouter de la redondance ; c'est-à-dire que l'on transmet plus de données que la quantité d'informations initiale. La redondance introduite est utilisée lors de la réception des données afin de détecter et corriger les erreurs de transmission.

Il existe deux grandes familles de codes : les codes en blocs et les codes convolutifs. Pour chacune de ces deux familles, l'information est séparée en blocs de taille constante k appelés *mots d'information*. Les blocs obtenus par codage de ces mots d'informations sont appelés des *mots de code*.

On note $u = (u_1, \dots, u_k)$, $u_i \in \mathbb{F}_q$, un mot d'information et $y = (y_1, \dots, y_n)$, $y_i \in \mathbb{F}_q$, le mot de code associé où $n > k$. On appelle n la *longueur du code*.

Codes en blocs Nous allons à présent voir des exemples de codes en blocs et les premières définitions de codes, notamment de codes dits linéaires.

Un exemple de code : le code à répétition

Le principe du code à répétition est simplement de répéter un certain nombre de fois n chacun des bits du message à transmettre. Pour $n = 3$ le message 0110 devient 000111111000. Si une erreur se produit lors de la transmission alors on retrouvera le message initial par vote majoritaire. Cette méthode permet de corriger toutes les erreurs de poids 1 (un seul bit sur les trois répétés est faux).

Ce code a l'avantage de pouvoir corriger autant d'erreurs que l'on souhaite en augmentant le nombre de répétition, mais le nombre de bits à transmettre est alors trop important. Pour $n = 3$ si le message à transmettre est de longueur m alors le message codé sera de longueur $3m$.

On définit le *rendement* d'un code par le ratio k/n où k est la longueur des mots d'informations et n celle des mots de code. Dans l'exemple du code à répétition $k = 1$ et $n = 3$.

On définit la *distance minimale* d'un code, noté d, comme la plus petite distance séparant deux mots de code (pour la distance de Hamming). Le code à répétition a pour distance minimale $d = n$.

La question du décodage est liée à la notion de capacité de correction, c'est-à-dire pour un code donné, combien d'erreurs est-on en mesure de corriger. Soit $\mathscr{C}$ un code de longueur n, dont la longueur des mots d'informations est k et de distance minimale d. Soit $y = (y_1, \ldots, y_n),\ y_i \in \mathbb{F}_q$ un mot de code. Lors de la transmission du signal, ce mot de code peut être modifié. Le mot reçu s'écrit alors $y' = y + e$ où e correspond à l'erreur ajoutée.

On appelle *décodage par maximum de vraisemblance* le décodage qui consiste à associer à y' le mot de code y le plus proche (au sens de la distance de Hamming). Alors on peut décoder de manière unique tous mots reçus ayant au plus $t = \lfloor \frac{d-1}{2} \rfloor$ erreurs. t est appelé la *capacité de correction.*

L'objectif de la théorie des codes est de définir des codes assurant à la fois de bon taux de transmission et de bonnes capacités de correction.

Un exemple de code : le code de Hamming

Soit $G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$. Le code défini par l'ensemble des $y = uG$, avec $u = (u_1, u_2, u_3, u_4)$, $u_i \in \mathbb{F}_2$, a pour longueur 7, des mots d'informations de longueur 4 et une distance minimale de 3. Il a donc pour capacité de correction $t = 1$, pour un rendement de $\frac{4}{7}$. Il est ainsi un code de même distance minimale que le code à répétition mais a un meilleur rendement.

Le code de Hamming présenté ici est défini par une matrice G de taille $k \times n$, à coefficients dans $\mathbb{F}_2$, que l'on appelle une *matrice génératrice.* Il peut également être défini par une *matrice* dite *de parité*, notée H, de taille $(n-k) \times n$, à coefficients dans $\mathbb{F}_2$, telle que $GH^\top = 0_{k\times(n-k)}$. Alors, pour tout mot de code y de $\mathscr{C}$ on a $yH^\top = 0$. On peut ainsi déterminer l'appartenance d'un mot à un code en connaissant une matrice de parité.

Nous nous attachons maintenant à définir les notions de code, matrice génératrice et matrice de parité pour les codes dit linéaires.

Définition 3.1.1 (Code) *Un code $\mathscr{C}$ sur $\mathbb{F}_q$ de longueur n est un sous-ensemble de $\mathbb{F}_q^n$*

Définition 3.1.2 (Code linéaire) *Un code linéaire $\mathscr{C}$ sur $\mathbb{F}_q$ de longueur n et de dimension k est un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k.*

On écrit $[n, k, d]_q$ les paramètres d'un code linéaire $\mathscr{C}$ de longueur n, dimension k, et distance minimale d défini sur $\mathbb{F}_q$ et $[n, k, d]$ lorsque qu'il n'y a pas d'ambiguïté sur le corps de base.

Définition 3.1.3 (Matrice génératrice) *Une matrice génératrice d'un code linéaire $\mathscr{C}$ est une matrice dont les vecteurs lignes forment une base de $\mathscr{C}$.*

Une matrice génératrice n'est donc pas unique et un code possède autant de matrices génératrices que de bases.

Définition 3.1.4 (Code dual) *Soit $\mathscr{C}$ une code linéaire de longueur n et dimension k. On note $\mathscr{C}^\perp$ le code dual de $\mathscr{C}$ tel que $\mathscr{C}^\perp = \{y \in \mathbb{F}_q^n \mid uy = 0, \forall u \in \mathscr{C}\}$ où $uy = \sum_{i=1}^n u_i y_i$, $u_i, y_i \in \mathbb{F}_q$*

Lors de l'utilisation d'un code en bloc, chaque mot d'information est codé indépendamment des autres. Le mot de code y dépend uniquement du mot d'information u. On définit la *fonction de codage* pour les codes en blocs comme l'application linéaire injective $\phi : \mathbb{F}_q^k \to \mathbb{F}_q^n$ qui à un mot d'information u associe un mot de code $y = \phi(u)$. On peut d'ailleurs définir un code en bloc comme l'ensemble des mots de code $y = uG$ où G est une matrice de l'application ϕ dans une base donnée. D'autre part, lorsque l'on dispose d'une séquence binaire à coder, cette dernière est décomposée en blocs de taille k qui seront codés pour former des blocs de taille n, constituant ainsi la séquence codée. On note alors respectivement u_j et y_j les mots d'informations et de codes obtenus au temps j. On obtient donc la relation suivante : $y_j = u_j G$. Lorsque les séquences d'informations et les séquences codées sont vues comme des séquences infinies, on écrit $U(D) = \sum_{j\geq 0} u_j D^j$ où D est une variable indéterminée et $u_j = (u_0, \ldots, u_k)_j = (u_{0,j}, \ldots, u_{k,j})$. On a alors $Y(D) = U(D)G$ avec $Y(D) = \sum_{j\geq 0} y_j D^j$.

3.2 Codes cycliques

Soient $c = (c_0, c_1, \ldots, c_{n-1})$ appartenant à $\mathbb{F}_q^n$ et $c' = (c_{n-1}, c_0, \ldots, c_{n-2})$ le décalé de c. Un code de longueur n est dit cyclique si pour tout mot de code $c = (c_0, c_1, \ldots, c_{n-1})$ son décalé $c' = (c_{n-1}, c_0, \ldots, c_{n-2})$ est aussi un mot de code. Ce qui implique que $(c_{n-2}, c_{n-1}, c_0, \ldots, c_{n-3})$, $(c_{n-3}, c_{n-2}, c_{n-1}, \ldots, c_{n-4})$ et ainsi de suite sont aussi des mots de code.

Pour les mots de codes d'un code cyclique on adopte usuellement l'écriture suivante

$$c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

D'où

$$c'(x) = c_{n-1} + c_0 x + \cdots + c_{n-2} x^{n-1}$$

D'autre part

$$\begin{aligned} xc(x) &= c_0 x + c_1 x^2 + \cdots + c_{n-1} x^n \\ &= c_{n-1}(x^n - 1) + (c_0 x + c_1 x^2 + \cdots + c_{n-2} x^{n-1} + c_{n-1}) \end{aligned}$$

et

$$xc(x) \ mod \ (x^n - 1) = c_{n-1} + c_0 x + \cdots + c_{n-2} x^{n-1}$$

donc

$$c'(x) = xc(x) \ mod \ (x^n - 1)$$

De même, si c est décalé de deux rangs vers la droite, on obtient $(c_{n-2}, c_{n-1}, c_0, \ldots, c_{n-3})$, ce qui correspond à $x^2 c(x) \ mod \ (x^n - 1)$. C'est également vrai pour les décalés suivants.

L'ensemble des opérations sur $c(x)$ se fait alors dans l'anneau quotient $\mathbb{F}_q[x]/(x^n-1)$ et on identifie $\mathbb{F}_q^n$ à $\mathbb{F}_q[x]/(x^n-1)$.

En montrant que $\mathscr{C}$ est un idéal de cet anneau, on obtient alors l'existence d'un unique polynôme unitaire $g(x)$ de degré minimal et tel que $g(x)$ est un générateur de $\mathscr{C}$. De plus $g(x)$ divise x^n-1 dans $\mathbb{F}_q[x]$.

Notons r le degré de $g(x) = g_0 + g_1x + \cdots + g_rx^r$. Ce polynôme $g(x)$ est un générateur du code alors tout mot de code $c(x)$ est un multiple de $g(x)$, c'est-à-dire $c(x) = m(x)g(x)$ avec $m(x) = m_0 + m_1x + \cdots + m_{k-1}x^{k-1}$. Le code est de dimension k et $\{g(x), xg(x), x^2g(x), \ldots, x^{k-1}g(x)\}$ est une base de $\mathscr{C}$. Le polynôme $c(x) = m(x)g(x)$ de degré inférieur ou égal à $n-1$ s'écrit :

$$c(x) = m_0g(x) + m_1xg(x) + \cdots + m_{k-1}x^{k-1}g(x)$$

ce qui se représente sous la forme matricielle suivante :

$$\begin{pmatrix} m_0 & m_1 & \ldots & m_{k-1} \end{pmatrix} \begin{pmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{pmatrix}$$

ou encore

$$\begin{pmatrix} m_0 & m_1 & \ldots & m_{k-1} \end{pmatrix} \begin{pmatrix} g_0 & g_1 & \ldots & g_r & & \\ & g_0 & g_1 & \ldots & g_r & \\ & & \ddots & \ddots & & \ddots \\ & & & g_0 & g_1 & \ldots & g_r \end{pmatrix}$$

Cette matrice de taille $k \times n$ est une matrice génératrice de $\mathscr{C}$. D'autre part soit $h(x)$ tel que $g(x)h(x) = x^n - 1$ alors

$$H_{(n-k)\times n} = \begin{pmatrix} h_k & h_{k-1} & \ldots & h_0 & & \\ & h_k & k_{k-1} & \ldots & g_0 & \\ & & \ddots & \ddots & & \ddots \\ & & & h_k & h_{k-1} & \ldots & h_0 \end{pmatrix}$$

est une matrice génératrice du dual où $h(x) = h_0 + h_1x + \cdots + h_{k-1}x^{k-1}$

Codes cycliques raccourcis Soit $\mathscr{C}'$ le sous-ensemble de $\mathscr{C}$ composé de l'ensemble des mots de code $c'(x)$ tels que $c'(x) = m(x)g(x)$ avec

$$m(x) = m_0 + m_1x + \cdots + m_{k-i-1}x^{k-i-1}$$

et i un entier positif. C'est-à-dire

$$m = \begin{pmatrix} m_0 & m_1 & \ldots & m_{k-i-1} & 0 & \ldots & 0 \end{pmatrix}.$$

Alors $\mathscr{C}'$ est un code de dimension $k-i$ et de longueur $n-i$. Un tel code est dit *raccourci*.

Codes Reed-Solomon Les codes Reed-Solomon font partie des codes cycliques les plus utilisés, ils permettent de corriger les erreurs se produisant par paquets. Il ont été inventés en 1960 par Reed et Solomon.

Nous supposons à présent que n est premier avec q. Il existe deux façons de décrire un code de Reed-Solomon. La première que nous donnons est la première définie historiquement et permet de donner des résultats théoriques, la seconde est cependant utilisée pour décrire des codeurs et décodeurs de ces codes.

Définition 3.2.1 *Soient α un élément primitif de $\mathbb{F}_{q^m}$, $n = q^m - 1$ et $m(x) = m_0 + m_1 x + \cdots + m_{k-1}x^{k-1} \in \mathbb{F}_{q^m}[x]$ un mot d'information sous forme polynomiale.*
Soit $\phi : m(x) \to c$ tel que

$$c = (c_0, c_1, \ldots, c_{n-1}) = \Big(m(1), m(\alpha), m(\alpha^2), \ldots, m(\alpha^{n-1})\Big)$$

Le code de Reed-Solomon de longueur $n = q^m - 1$ et de dimension k sur $\mathbb{F}_{q^m}$ est l'ensemble des images, par ϕ, des polynômes de degré inférieur ou égal à $k-1$ à coefficients dans $\mathbb{F}_{q^m}$.

La seconde construction se rapproche de celle des codes BCH (ils doivent leur nom à Bose, Chaudury et Hocquenghem) que nous allons évoquer à présent.

Rappelons que le polynôme minimal d'un élément de $\mathbb{F}_{q^m}$ est le polynôme (non nul) de plus faible degré dont il est racine. De plus les éléments admettant le même polynôme minimal sont dits conjugués. Un polynôme primitif est un polynôme minimal dont au moins une racine est un élément primitif de $\mathbb{F}_{q^m}$.

Un code BCH de longueur n capable de corriger au moins t erreurs se construit selon la procédure suivante : Prendre m le plus petit entier tel que $\mathbb{F}_{q^m}$ a une racine n-ième primitive α. Choisir un entier b positif (souvent $b = 1$). Déterminer ensuite les polynômes minimaux de $\alpha^b,\ \alpha^{b+1},\ \ldots,\ \alpha^{b+2t-1}$ à coefficients dans $\mathbb{F}_q$. Le polynôme générateur $g(x)$ du code est le plus petit multiple commun de ces polynômes minimaux.

Ainsi, les coefficients de $g(x)$ sont dans $\mathbb{F}_q$ ainsi que les mots du code tandis que les racines de $g(x)$ sont elles dans $\mathbb{F}_{q^m}$. De plus, certains des α^i sont potentiellement conjugués donc $g(x)$ est de degré au plus $2t$.

Le code ainsi construit est un code de longueur n, de dimension $n - \deg g(x)$ et de distance minimale supérieure ou égale à $\delta = 2t + 1$.

Les $\alpha^b,\ \alpha^{b+1},\ \ldots,\ \alpha^{b+2t-1}$ sont par définition racines de leur polynômes minimaux respectifs et sont alors racines de $g(x)$ sur $\mathbb{F}_{q^m}$. Donc pour tout α^i, $(i = b,\ b+1,\ \ldots,\ b+2t-1\ )$, $g(\alpha^i) = 0$ et par conséquent $c(\alpha^i) = m(\alpha^i)g(\alpha^i) = 0$. Ceci se traduit par

$$c_0 + c_1\alpha^i + c_2(\alpha^i)^2 + \cdots + c_{n-1}(\alpha^i)^{n-1} = 0,\ \forall i = b,\ b+1,\ \ldots,\ b+2t-1$$

Donc

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

et

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

est également une matrice génératrice du dual.

Passons à présent aux codes de Reed-Solomon. Dans $\mathbb{F}_{q^m}$ le polynôme minimal d'un élément α est égal à $(x - \alpha)$. Un tel code prend comme polynôme générateur

$$g(x) = (x - \alpha^b)(x - \alpha^{b+1}) \cdots (x - \alpha^{b+2t-1}).$$

$g(x)$ est alors de degré exactement $2t$. Donc $n - k = 2t$. Et le code est en mesure de corriger t erreurs.

Un exemple de code Reed-Solomon : Nous décrivons ici un code de longueur 15 et de dimension 9 sur $\mathbb{F}_{2^4}$. Ce code est en mesure de corriger 3 erreurs. De plus $\mathbb{F}_{2^4}$ est identifié à $\mathbb{F}_2[x]/(x^4+x+1)$. Soit α dans $\mathbb{F}_{2^4}$ une racine de x^4+x+1. Nous avons $\alpha^{15} = 1$. On peut d'autre part décrire l'ensemble des puissances successives de α. En effet $\alpha^4 = \alpha + 1$ puisque α est racine de $x^4 + x + 1$. Ensuite $\alpha^5 = \alpha^2 + \alpha$. Les puissances suivantes se calculent de la même manière et dépendent uniquement de 1, α, α^2 et α^3. On peut donc en donner une représentation vectorielle dans cette base. Les 7 premières puissances sont données dans le tableau ci-dessous :

Puissance de α	Représentation dans la base	Représentation vectorielle
1	1	$(1,0,0,0)$
α	α	$(0,1,0,0)$
α^2	α^2	$(0,0,1,0)$
α^3	α^3	$(0,0,0,1)$
α^4	$\alpha + 1$	$(1,1,0,0)$
α^5	$\alpha^2 + \alpha$	$(0,1,1,0)$
α^6	$\alpha^3 + \alpha^2$	$(0,0,1,1)$

et α est bien un élément primitif de $\mathbb{F}_{2^4}$. Prenons $b = 1$, on a $n = 15$ et $k = 9$ donc $2t = 6$ et $g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)(x - \alpha^5)(x - \alpha^6)$.

L'avantage de tels codes est que l'on maîtrise très bien la capacité de correction. De plus, les codes définis sur $\mathbb{F}_{2^8}$ sont souvent utilisés du fait de leur praticité. En effet, chaque élément de $\mathbb{F}_{2^8}$ se représente dans la base $\{1, \alpha, \dots \alpha^7\}$ sur un octet.

Par ailleurs, il est possible de donner des matrices systématiques pour les matrices génératrices de $\mathscr{C}$ ainsi que pour $\mathscr{C}^\perp$. En effet, la division euclidienne de x^{n-k+i} par $g(x)$ pour $i = 0, 1, \dots, k-1$

donne

$$x^{n-k+i} = q_i(x)g(x) + r_i(x)$$

avec $r_i(x) = r_{i,0} + r_{i,1}x + \cdots + r_{i,n-k-1}x^{n-k-1}$. C'est-à-dire

$$x^{n-k+i} - r_i(x) = q_i(x)g(x)$$

donc $x^{n-k+i} - r_i(x)$ est un multiple de $g(x)$ et est aussi un mot de code. Le polynôme $r_i(x)$ s'écrit sous forme matricielle

$$\begin{pmatrix} r_{i,0} & r_{i,1} & \dots & r_{i,n-k-1} & 0 & \dots & 0 \end{pmatrix}$$

donc $x^{n-k} - r_0(x)$ correspond à

$$\begin{pmatrix} -r_{0,0} & -r_{0,1} & \dots & -r_{0,n-k-1} & 1 & 0 & \dots & 0 \end{pmatrix}$$

et on obtient la matrice génératrice suivante :

$$G = \begin{pmatrix} -r_{0,0} & -r_{0,1} & \dots & -r_{0,n-k-1} & 1 & 0 & 0 & \dots & 0 \\ -r_{1,0} & -r_{1,1} & \dots & -r_{1,n-k-1} & 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & & & \ddots & & \\ -r_{k-1,0} & -r_{k-1,1} & \dots & -r_{k-1,n-k-1} & 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

et

$$H = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & r_{0,0} & r_{1,0} & \dots & r_{k-1,0} \\ 0 & 1 & 0 & \dots & 0 & r_{0,1} & r_{1,1} & \dots & r_{k-1,1} \\ & & \ddots & & & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 & r_{0,n-k-1} & r_{1,n-k-1} & \dots & r_{k-1,n-k-1} \end{pmatrix}$$

Les codes en blocs ont l'avantage d'être largement étudiés et permettent à ce titre des constructions variées. En revanche la complexité de codage et décodage peut varier selon le type de code. Les codes convolutifs sont eux implémentés sous forme de registres à décalage et présentent ainsi l'avantage d'offrir un codage simple et un décodage rapide au fur et à mesure de la réception de l'information. Cependant, ils fournissent généralement de plus faibles capacités de correction.

On définit pour les codes convolutifs linéaires la notion de *matrice génératrice* de manière similaire à la notion de matrice génératrice pour les codes en blocs. Les définitions présentées dans le paragraphe suivant sont essentiellement issues de [38]. Nous parlerons ici uniquement de codes convolutifs linéaires, nous utiliserons donc le terme code convolutif pour signifier code convolutif linéaire.

3.3 Codes convolutifs

Lors de l'utilisation d'un code convolutif, le mot de code y_j dépend du mot d'information u_j mais également des mots d'informations reçus aux instants précédents. y_j dépend donc de $u_j, u_{j-1}, \dots, u_{j-M}$ où M correspond à la *mémoire* du codeur. On définit donc la fonction de codage d'un code convolutif comme l'application linéaire injective $\phi : \mathbb{F}_2^k \to \mathbb{F}_2^n$ telle que $y_j = \phi(u_j, u_{j-1}, \dots, u_{j-M})$. Ainsi,

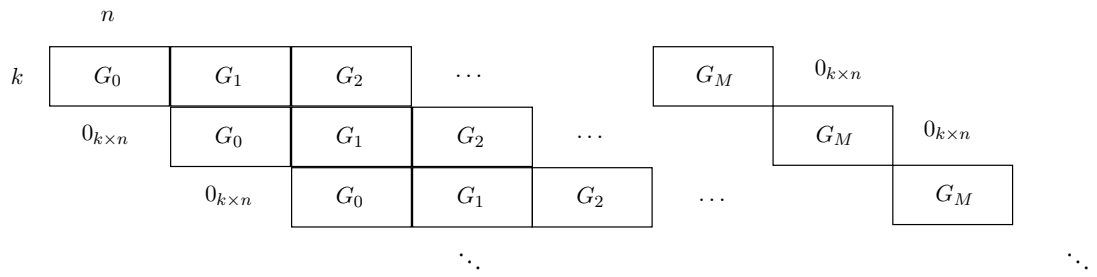


FIGURE 3.1 – Matrice génératrice binaire d'un code convolutif

la matrice G de cette application dans une base donnée prendra la forme d'une matrice shiftée infinie comme définie par la FIGURE 3.1 , où chacune des matrices G_i est une matrice binaire de taille $k \times n$.

On pourra également décrire la matrice G comme une matrice à coefficients dans $\mathbb{F}_2(D)$. On note $G(D) = G_0 + G_1 D + G_2 D^2 + \dots G_M D^M$, $G(D)$ est alors une matrice de taille $k \times n$ à coefficients dans $\mathbb{F}_2(D)$.

On peut d'ailleurs voir les codes en blocs comme des codes convolutifs de mémoire nulle.

On distinguera ici la notion de code et de codeur. Le code est l'ensemble des mots de code tandis que le codeur (ou encodeur) est la réalisation physique de la fonction de codage. D'autre part, la réalisation physique d'une fonction de codage correspond à l'implémentation de registres à décalage. Mais au même titre qu'un code dispose de plusieurs matrices génératrices, il existe différentes réalisations pour un même code généré. Les registres nécessitant le moins de mémoire sont à priori les plus utilisés et donc les plus intéressants du point de vue de la reconnaissance. Nous verrons alors que l'algorithme de reconnaissance de codes convolutifs que nous utilisons dans la suite se propose de retourner une des matrices génératrices correspondant à ces registres. On note alors (n, k) les paramètres d'un codeur convolutif ayant k entrées et n sorties.

Nous ne décrirons pas précisément le fonctionnement de cet algorithme mais en donnerons les principales composantes. Cet algorithme sera ensuite pour nous un moyen de discriminer les mappings lors de la reconnaissance de mapping en présence de données codées par un codeur convolutif.

Définition 3.3.1 (Code convolutif) *Un (n, k) code convolutif $\mathscr{C}$ est un sous-espace vectoriel de dimension k de $\mathbb{F}_2(D)^n$.*

Définition 3.3.2 (Matrice génératrice) *Une matrice génératrice $G(D)$ du code $\mathscr{C}$ de paramètres (n, k) est une matrice de taille $k \times n$ dont les lignes forment une base de $\mathscr{C}$.*

Une matrice génératrice d'un code convolutif est donc une matrice à coefficients dans $\mathbb{F}_2(D)$. On parle de *matrice génératrice polynomiale* lorsque les coefficients de $G(D)$ appartiennent à $\mathbb{F}_2[D]$.

On peut écrire $U(D) = \sum_{j\geq 0} u_j D^j$ avec $u_j = (u_0, \dots, u_k)_j = (u_{0,j}, \dots, u_{k,j})$ et $Y(D) = \sum_{j\geq 0} y_j D^j$. On a alors $Y(D) = U(D)G(D)$.

$$(u_0, u_1, \dots, u_M, u_{M+1}, \dots) \times \begin{pmatrix} G_0 & G_1 & \dots & G_M & 0 & \dots & & & \\ & G_0 & G_1 & \dots & G_M & 0 & \dots & & \\ & & \ddots & & & & & & \\ & & & & G_0 & G_1 & \dots & G_M & 0 \\ & & & & & G_0 & G_1 & \dots & G_M \\ & & & & & \ddots & \ddots & & \ddots \end{pmatrix}$$

$$\underbrace{}_{y_0} \quad \underbrace{}_{y_1} \qquad \underbrace{}_{y_M} \quad \underbrace{}_{y_{M+1}} \quad \dots$$

FIGURE 3.2 – Codage d'une séquence par un code convolutif

3.3.1 Représentation binaire

Rappelons que du point de vue binaire la séquence codée (composée des mots de code concaténés), notée $y_0 y_1 \dots y_M \dots$ est telle que :

$$\begin{aligned} y_0 &= u_0 G_0 \\ y_1 &= u_0 G_1 + u_1 G_0 \\ &\dots \\ y_M &= u_0 G_M + u_1 G_{M-1} + \dots + u_M G_0 \\ y_{M+1} &= u_1 G_M + u_2 G_{M-1} + \dots + u_{M+1} G_0 \end{aligned}$$

où u_j est le mot d'information à l'instant j et les G_i les blocs de la matrice génératrice binaire infinie de $\mathscr{C}$ présentée précédemment.

Remarquons que dès lors que $j \geq M+1$, y_j ne dépend plus de u_0. En effet la mémoire du codeur est de M et $y_j = \phi(u_j, u_{j-1}, \dots, u_{j-M})$.

Lorsque $j = 0, \dots, M-1$, y_j ne dépend pas de tous les G_i ($i = 0, \dots, M$). Cela constitue une phase d'initialisation.

Dès lors que $j \geq M$, y_j dépend de tous les G_i et la sortie y_j est obtenue en effectuant le produit de la séquence d'entrée par la matrice binaire infinie. Nous avons alors :

$$y_j = \sum_{i=0}^{M} u_{j-i} G_i.$$

Ce produit est représenté par la FIGURE 3.2

3.3.2 Équations de parité

Soit
$$\mathrm{H} = \begin{pmatrix} H_0 & H_1 & \dots & H_M & 0 & \dots & & & \\ & H_0 & H_1 & \dots & H_M & 0 & \dots & & \\ & & \ddots & & & \ddots & & & \\ & & & H_0 & H_1 & \dots & H_M & 0 & \\ & & & & H_0 & H_1 & \dots & H_M & 0 \\ & & & & & \ddots & & & \ddots \\ & & & & & & & & \end{pmatrix}$$
une matrice duale de $\mathscr{C}$. On a
$$\begin{pmatrix} y_0 & y_1 & \dots & y_M \end{pmatrix} \begin{pmatrix} H_0 & \dots & H_M \end{pmatrix}^\top = 0_{1\times(n-k)}.$$
On dit que $\begin{pmatrix} y_0 & y_1 & \dots & y_M \end{pmatrix}$ vérifie $n-k$ équations de parité. Il en est de même pour $\begin{pmatrix} y_1 & y_2 & \dots & y_{M+1} \end{pmatrix}$, $\begin{pmatrix} y_2 & y_3 & \dots & y_{M+2} \end{pmatrix}$ et ainsi de suite. Les équations de parité s'appliquent à toutes les sous-séquences de longueur $n(M+1)$ par blocs glissants (avec un décalage de n).

3.3.3 Représentation en série

La séquence d'information s'écrit comme étant la série
$$U(D) = \sum_{j\geq 0} u_j D^j.$$
Notons
$$Y(D) = \sum_{j\geq 0} y_j D^j.$$
Nous avons alors la relation $Y(D) = U(D)G(D)$ avec $G(D) = G_0 + G_1 D + \cdots + G_M D^M$. En effet $G(D) = G_0 + G_1 D + \cdots + G_M D^M$ donc
$$\begin{aligned}
\left(\sum_{j\geq 0} u_j D^j\right) G(D) &= \sum_{j\geq 0} u_j G_0 D^j + \sum_{j\geq 0} u_j G_1 D^{j+1} + \cdots + \sum_{j\geq 0} u_j G_M D^{j+M} \\
&= \sum_{j\geq 0} u_j G_0 D^j + \sum_{j\geq 1} u_{j-1} G_1 D^j + \cdots + \sum_{j\geq M} u_{j-M} G_M D^j \\
&= u_0 G_0 + (u_0 G_1 + u_1 G_0) D + \dots \\
&\quad + (u_0 G_M + u_1 G_{M-1} + \cdots + u_M G_0) D^M + \left(\sum_{j>M} \sum_{i=0}^{M} u_{j-i} G_i\right) D^j \\
&= \sum_{j\geq 0} y_j D^j = Y(D)
\end{aligned}$$

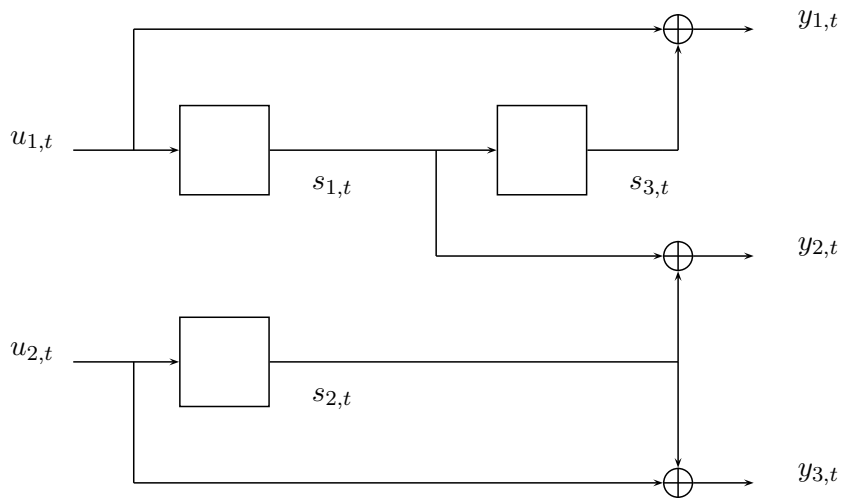


FIGURE 3.3 – Codeur de rendement 2/3

3.3.4 Codeurs

Les sorties d'un codeur convolutif dépendent des entrées à différents instants. En termes d'implémentation, il faut donc stocker différents états de l'entrée dans des registres. Notons m ce nombre de registres. Les m registres représentent alors l'état interne du codeur, noté s_j (s_j est un vecteur binaire de taille m). Notons que dans le cas particulier où $k = 1$ alors $m = M$. L'entier m est également appelé la *longueur de contrainte* du codeur.

Il est possible de déterminer une matrice génératrice du code $\mathscr{C}$ en fonction du vecteur d'entrée u_j, du vecteur d'état s_j et du vecteur de sortie y_j pour un codeur donné. Le vecteur d'état s_j dépend de l'état s_{j-1} à l'instant précédent et de l'entrée u_j à l'instant j. La sortie y_j dépend de l'entrée u_j à l'instant j et de l'état interne s_j. L'opération de codage se décrit alors comme suit :

$$\begin{aligned} s_{j+1} &= s_j A_{m\times m} + u_j B_{k\times m} \\ y_j &= u_j U_{k\times n} + s_j V_{m\times n} \end{aligned}$$

avec $A_{m\times m}$, $B_{k\times m}$, $U_{k\times n}$, $V_{m\times n}$ des matrices binaires et $s_0 = 0_{1\times m}$. En posant

$$E(D) = B.\left(D^{-1}Id_m - A\right)^{-1}$$

nous obtenons

$$Y(D) = U(D)\left[U + E(D)V\right]$$

d'où

$$G(D) = U + E(D)V.$$

Nous proposons à présent un exemple de codeur convolutif de rendement 2/3 donné par la FIGURE 3.3. Notons

$$\begin{aligned} u_j &= (u_1, u_2)_j = (u_{1,j}, u_{2,j}) \\ y_j &= (y_1, y_2, y_3)_j = (y_{1,j}, y_{2,j}, y_{3,j}) \\ s_j &= (s_1, s_2, s_3)_j = (s_{1,j}, s_{2,j}, s_{3,j}). \end{aligned}$$

Le codage est décrit comme suit :

$$\begin{aligned} y_{1,j} &= u_{1,j} + s_{3,j} \\ y_{2,j} &= s_{1,j} + s_{2,j} \\ y_{3,j} &= u_{2,j} + s_{2,j} \end{aligned}$$

Nous obtenons alors

$$\begin{aligned} U_{k\times n} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ V_{m\times n} &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \\ A_{m\times m} &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ B_{k\times m} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \end{aligned}$$

Et finalement

$$E(D) = \begin{pmatrix} D & 0 & D^2 \\ 0 & D & O \end{pmatrix}$$

d'où

$$G(D) = \begin{pmatrix} 1+D^2 & D & O \\ 0 & D & 1+D \end{pmatrix}$$

Nous utilisons également dans ce manuscrit une notation décimale pour représenter les polynômes générateurs dont la représentation en base 2 donne les coefficients des polynômes dans l'ordre décroissant. Dans le cas de l'exemple nous obtenons $\begin{pmatrix} 5 & 2 & 0 \\ 0 & 2 & 3 \end{pmatrix}$.

3.4 Reconstruction de codes

La méthode décrite par A. Valembois dans [47] et [48] est actuellement utilisée pour la reconstruction de codes en blocs et de codes convolutifs. Cette méthode permet la reconstruction d'une matrice

génératrice du dual. Elle s'appuie sur la méthode de Canteaut-Chabaud pour trouver des mots candidats du dual et décide selon un test statistique de leur appartenance au dual.

Il existe de plus des méthodes spécifiques au type de codes recherchés. En effet, connaître une matrice binaire du dual n'est pas suffisant pour le décodage. Il faut d'une part retrouver une matrice génératrice du code puis en extraire la structure algébrique.

B. Rice a initié des travaux sur la reconnaissance des codes convolutifs $(n, 1)$ dans le cas non bruité dans [41]. E. Filiol a ensuite proposé une méthode de reconstruction des codes convolutifs lorsque les données observées sont bruitées dans [28] et [27]. Puis une méthode algébrique a été donnée par J. Barbier, G. Sicot et S. Houcke dans [5], elle est ensuite détaillée dans la thèse de J. Barbier [3]. De nombreuses autres études ont depuis été menées sur la reconstruction des codes convolutifs dont [14, 21, 24, 33, 35–37, 53, 59–61]

3.4.1 Codes Convolutifs

Soit $\mathscr{C}$ un code tel que les polynômes générateurs soient premiers entre eux et de longueur de contrainte m. Le codeur utilisé nécessite m bits pour remplir ses registres. Une séquence d'information de taille $ks + m$ s'envoie sur une séquence codée de taille ns. Soit $\mathscr{C}_s$ cette restriction :

$$\mathscr{C}_s : \mathbb{F}_2^{ks+m} \to \mathbb{F}_2^{ns}$$

Le théorème suivant, issue de [14], donne le rang de l'application $\mathscr{C}_s$ en fonction de s et montre qu'à partir de $s > \frac{km}{n-k}$ toutes les séquences de taille ns ne correspondent pas à une séquence codée. Le critère du rang est à la base de la plupart des résultats sur la reconnaissance de codes convolutifs.

Théorème 3.4.1 *Soit $s \in \mathbb{N}^*$. Soit $\mathscr{C}_s : \mathbb{F}_2^{ks+m} \to \mathbb{F}_2^{ns}$ l'application linéaire obtenue par restriction. Si tous les polynômes générateurs sont premiers entre eux alors*

$$rang(\mathscr{C}_s) = \begin{cases} ns & si\ s \leq \frac{km}{n-k} \\ k(s+m) & si\ s \geq \frac{km}{n-k} \end{cases}$$

Il en découle que la matrice dont les lignes sont des séquences codées de taille ns n'est pas de rang plein lorsque $s > \frac{km}{n-k}$. Ce défaut de rang peut alors s'identifier par un pivot de Gauss. Malgré tout, lorsque la séquence observée est bruitée, cette méthode n'est plus envisageable.

Les alternatives sont alors de chercher des plages non bruitées, d'utiliser une méthode appelée Gauss randomisé issue des travaux de G. Sicot et S. Houcke [42, 43] qui est applicable pour de faibles taux d'erreur ou encore d'utiliser la méthode introduite par A. Valembois [47, 48]. M. Cluzeau dans [15] a également repris et fait évoluer l'approche proposée par A. Valembois

L'algorithme de reconnaissance de codes convolutifs linéaires dont nous disposons (issu de [20, 21]) utilise la méthode de A. Valembois et se propose alors de retourner une matrice génératrice correspondant à un codeur minimisant le nombre de registres nécessaires à l'implémentation, lorsque la séquence binaire observée est une séquence codée et bruitée. Cet algorithme retourne alors une matrice génératrice parmi les matrices génératrices polynomiales (à coefficients $\mathbb{F}_2[D]$) appelée une *matrice canonique*

(voir [20, 38]). Nous allons à présent décrire dans les grandes lignes le fonctionnement de cet algorithme.

A partir d'une séquence, codée et bruitée, tronquée, nous cherchons à retrouver une matrice génératrice du code utilisé ainsi que le taux d'erreur associé. Nous ne connaissons ni n (la longueur du code) ni k (la dimension du code), et effectuons alors cette recherche également.

Nous avons vu précédemment que pour un mot observé, nous disposons d'un critère d'appartenance au code $\mathscr{C}$ utilisé ; à savoir $y \in \mathscr{C}$ si et seulement si $yH^\top = 0_{1\times(n-k)}$ avec H une matrice génératrice du dual de $\mathscr{C}$ et si et seulement si pour tout $h \in \mathscr{C}^\perp$ on a $yh = 0$

Cet algorithme procède en 3 étapes principales. Le premier objectif est de trouver une matrice génératrice du dual de $\mathscr{C}$, noté H, autrement appelée *matrice de parité*. Il utilise pour cela l'algorithme de A. Valembois et la méthode de Canteaut-Chabaud. Cette recherche s'effectue à partir d'une matrice formée de mots issus de la séquence observée.

La séquence observée étant bruitée, cette première étape permet de sélectionner des mots du dual non bruités. La seconde étape utilise alors ces mots du dual non bruités pour reconstruire le dual du dual, à savoir une matrice génératrice du code lui-même.

La dernière étape a pour but d'extraire une matrice canonique de la matrice reconstruite à l'étape 2.

A ces étapes se rajoutent (entre la première et la deuxième étape) une étape de recherche de la longueur n du code et en fin d'algorithme une étape de recherche du taux d'erreur. La recherche du taux d'erreur consiste à décoder la séquence observée par l'algorithme de Viterbi avec la matrice génératrice retournée par l'algorithme de reconnaissance de codes.

Première étape : Reconstruction d'une matrice H du dual de $\mathscr{C}$
Nous disposons d'une séquence observée S à partir de laquelle nous formons une matrice $\tilde{R}$ de la manière suivante : la séquence est découpée en blocs de taille $n(m+1)$ (quitte à itérer sur la taille des blocs lorsque ces paramètres sont inconnus), chacun de ces blocs constitue une ligne de $\tilde{R}$.

Si $\tilde{R}$ n'est pas une matrice bruitée : pour $h \in \mathscr{C}^\perp$, alors $\tilde{R}h^\top$ vaut 0. C'est-à-dire h appartient au noyau de $\tilde{R}$. Si $\tilde{R}$ est bruitée : pour $h \in \mathscr{C}^\perp$, alors $\tilde{R}h^\top$ est de poids faible.

L'algorithme de recherche de mots de poids faibles de Canteaut-Chabaud permet alors de retrouver des h candidats et l'algorithme de A. Valembois permet de décider s'ils appartiennent ou non à $\mathscr{C}^\perp$ selon le poids de Hamming de $\tilde{R}h^\top$ et de h.

En sortie de cette première étape, nous disposons alors de mots du dual. Le produit d'une ligne de $\tilde{R}$ par h détermine une *équation de parité*.

M. Côte et N. Sendrier [20] utilisent alors ces mots du dual h pour former une seconde matrice, que l'on notera R et reconstituer une matrice génératrice du dual. Cette reconstruction se base sur une élimination de Gauss. En effet, une application de l'élimination de Gauss sur la matrice R la transforme en une matrice de forme particulière dans laquelle il est possible d'identifier un *bloc minimal* :

c'est-à-dire un ensemble de lignes et de colonnes adjacentes engendrant le sous-espace vectoriel décrit par R. Ce bloc, noté H, engendre $\mathscr{C}^{\perp}$.

Deuxième étape : Reconstruction d'une matrice génératrice de $\mathscr{C}$ à partir du dual

Cette étape consiste à retrouver le dual du dual de $\mathscr{C}$, c'est-à-dire à retrouver le code lui-même. La matrice H précédemment reconstituée est utilisée pour former une nouvelle matrice H' sur laquelle est pratiquée également une élimination de Gauss afin de trouver le bloc minimal, noté G, engendrant le sous-espace vectoriel décrit par H'.

Troisième étape : Reconstruction d'une matrice génératrice polynomiale canonique de $\mathscr{C}$

Après utilisation d'algorithmes de basicité et réduction [20] sur la matrice G reconstruite à l'étape 2, une matrice $G(D)$ polynomiale canonique de $\mathscr{C}$ est fournie.

Les étapes présentées ici reposent sur le fait que la longueur n soit connue. N. Sendrier propose d'appliquer la méthode de Canteaut-Chabaud et l'algorithme de A. Valembois à une matrice $\tilde{R}$ pour quelques longueurs de blocs bien choisies de telle sorte que cette longueur puisse être un multiple de la longueur n du code. D'autre part, le calcul de $w_H(\tilde{R}h^{\top})$ fournit une indication sur le taux d'erreur potentiel du canal. Il est alors possible de distinguer la plus petite longueur de code fournissant pour l'ensemble des mots du dual un faible taux d'erreur. Cette longueur est la longueur n du code $\mathscr{C}$.

Cet algorithme de reconnaissance de codes convolutifs permet d'effectuer une reconnaissance de codes lorsque la séquence observée est bruitée. Des tests de résistance au bruit ont été effectués dans [20]. Il permet de plus d'effectuer cette recherche à paramètres inconnus. Cette méthode fournit également le taux d'erreur du canal en appliquant un décodage de la séquence observée.

3.4.2 Codes Reed-Solomon

La difficulté de la reconstruction des codes de Reed-Solomon réside dans la recherche de la structure algébrique. En effet il existe des méthodes de reconstruction de codes en blocs permettant de reconstruire une base du code dual et ainsi de reconstruire une base du code. Il est possible notamment d'effectuer un pivot de Gauss sur une matrice contenant les données observées, de faire appel à le méthode de Gauss randomisé ou enfin d'utiliser la méthode A. Valembois ou de M. Cluzeau.

Cependant ce n'est pas suffisant pour caractériser un code de Reed-Solomon. Les méthodes énoncées plus haut offrent la possibilité de reconstruire une base du code vu sous forme binaire mais ne permettent pas de donner la structure algébrique du code. Pour cela il existe des techniques spécifiques au type de code utilisé.

L'algorithme de V.M. Sidelnikov et S.O. Shestakov [44] répond à ce problème pour les codes de Reed-Solomon. Il prend en entrée une matrice de taille génératrice $k \times n$, contenant des éléments de $\mathbb{F}_q$, mise sous forme systématique. Nous avons vu que la matrice systématique d'un tel code dispose d'une forme particulière, ceci permet d'écrire un ensemble d'équations qui après résolution fournissent les paramètres du code. Cet algorithme fonctionne de plus lorsque les coordonnées du code ont été permutées. Nous verrons que dans notre cas ce point fort sera à notre désavantage. Aussi nous utiliserons finalement un algorithme moins performant qui sera évoqué ultérieurement.

Chapitre 4

Signature de codes convolutifs et recherche du dual par tests statistiques

Lors de l'étude d'un système de communication et de la partie codage canal qui le compose, nous souhaitons avant toute chose identifier la famille de codes utilisée, voire les paramètres du code. En effet, ces informations permettent respectivement d'utiliser un algorithme de reconnaissance de code approprié et d'accélérer la recherche. Nous nous intéressons alors dans ce chapitre à une méthode permettant de distinguer si une séquence observée est aléatoire ou codée par un codeur convolutif. Cette technique possède notamment l'avantage de fonctionner lorsque le mapping utilisé est inconnu. Elle permet alors de distinguer la longueur d'un code convolutif sans connaissance du mapping. De plus nous étendons cette méthode à une méthode de reconstruction du dual d'un code convolutif lorsque le mapping est connu.

Nous appelons *signature* un biais observé via un test statistique, appliqué à une séquence codée et bruitée, permettant d'identifier l'utilisation d'un code correcteur d'erreurs. Nous supposons dans la suite que nous disposons d'une séquence codée éventuellement bruitée. Christophe Chabot [14] s'est intéressé à divers tests statistiques, dédiés aux générateurs pseudo-aléatoires, comme la série de test du *NIST* (National Institute of Standards and Technology) [1] . Ces tests ont été expérimentés dans le cadre de l'identification de séquences codées et bruitées par des codes en blocs, des codes convolutifs et des turbo-codes. Le résultat de ces tests a montré que les codes en blocs ne sont en général pas identifiables de cette façon sauf pour de petites longueurs et dimensions non réalistes. Cependant, le test de compression de *Lempel-Ziv* (méthode de compression par dictionnaire) s'est révélé adapté pour l'identification de codes convolutifs et turbo-codes et ce dans un contexte bruité également. Nous nous intéressons cependant dans la suite de ce chapitre aux codes convolutifs uniquement.

Christophe Chabot s'est alors intéressé à d'autres méthodes de compression comme la compression utilisée dans *bzip2*, faisant appel à la transformée de *Burrows-Wheeler* [11]. Les meilleurs taux de compression étant obtenus avec *bzip2*, il a défini un test statistique basé sur le calcul du nombre de runs (séquences maximales de bits identiques) dans une séquence codée après application de la transformée de *Burrows-Wheeler*. Ce test permet l'identification des paramètres d'un code convolutif.

Il existe également un test dit de profondeur permettant d'obtenir la longueur de contrainte d'un code convolutif de longueur 2.

Il est clair que les codes convolutifs offrent l'avantage d'être de petites longueurs n et dimensions k (pour que le décodage en treillis soit possible), et de degré d compris typiquement entre 2 et 10. Les objets que nous manipulons sont donc de tailles bien inférieures à celles nécessaires pour des codes en blocs. La méthode de détection que nous définissons dans ce chapitre s'applique en théorie pour des codes en blocs mais elle n'est cependant applicable en pratique que pour de petits codes et est donc particulièrement adaptée pour la détection des codes convolutifs. C'est pourquoi nous nous intéressons à ce type de code dans ce chapitre.

Dès lors que nous observons une équation de parité sur une séquence, l'espace ambiant n'atteint pas l'ensemble de toutes les séquences possibles. Pour un code de longueur n et de degré maximum d, les séquences de taille $t = n(d+1)$ sont au nombre de 2^{t-1} au plus contre 2^t pour une séquence aléatoire. En effet il existe au moins un bit de redondance donc au moins un bit est entièrement déterminé par les autres. Les tests qui sont décrits dans ce chapitre utilise ce phénomène. Nous l'utilisons également en définissant un test basé sur le comptage de blocs de taille t dans une séquence codée et bruitée. En itérant sur la taille des blocs, nous pouvons observer une diminution du nombre de blocs possibles sur certaines longueurs. Nous effectuons de plus cette mesure par comptage de collisions. Nous observons alors une probabilité de collisions différente pour des données codées et des données aléatoires. L'utilisation de blocs glissants permet de plus de caractériser la présence d'un code de type convolutif. Cette méthode autorise également la reconnaissance des paramètres du code utilisé dans un contexte bruité. Nous savons par ailleurs grâce au paradoxe des anniversaires que seulement $2^{t/2}$ blocs environ sont nécessaires à l'observation de collisions. Nous définissons ensuite une méthode de reconstruction du dual par comptage de collisions en poinçonnant la séquence observée de manière adéquate.

Nous verrons enfin que la signature d'un code convolutif peut se détecter sans connaissance a priori du mapping utilisé. Ce qui offre un avantage considérable pour la reconnaissance de système de communication. En effet nous sommes en mesure grâce à ce test de déterminer la longueur du code convolutif utilisé sans avoir reconnu le mapping utilisé. Ceci permettra alors d'effectuer la recherche du mapping avec une information non négligeable sur le code à rechercher. La reconnaissance du mapping est l'objet du Chapitre 6. Cependant ce travail sur la détection de codes convolutifs a été effectué ultérieurement au travail sur la reconnaissance du mapping et n'a donc pas pu être utilisé alors.

L'estimation du cardinal d'un observable, tout en minimisant la quantité de mémoire utilisée, est un défi primordial pour l'analyse de trafic internet par exemple. Marianne Durand s'intéresse dans sa thèse [26] à divers algorithmes d'estimation de cardinal. Nous n'avons pas exploré ces possibilités pour estimer la cardinalité d'une séquence codée vue par blocs mais il serait intéressant d'étudier l'efficacité de ces algorithmes dans le contexte de la reconstruction de codes.

4.1 État de l'art

4.1.1 Test de profondeur

Le test de profondeur décrit dans [31] s'applique au cas d'un code convolutif de longueur 2 et donc de dimension 1. Dans ce cas la longueur de contrainte du code coïncide avec $d+1$ où d est le degré maximum du code.

Soit $\mathscr{C}$ tel que P_1 et P_2 soient des polynômes générateurs. Le codeur utilisé nécessite M bits pour remplir ses registres. Une séquence d'information de taille $s+m$ s'envoie sur une séquence codée de taille $2s$. Soit $\mathscr{C}_s$ cette restriction :

$$\mathscr{C}_s : \mathbb{F}_2^{s+m} \to \mathbb{F}_2^{2s}$$

Le théorème suivant est à la base de la plupart des résultats sur la reconnaissance de codes convolutifs. Il donne en effet le rang de l'application $\mathscr{C}_s$ en fonction de s et montre qu'à partir de $s > m$ toutes les séquences de taille $2s$ ne correspondent pas à une séquence codée.

Théorème 4.1.1 ?? *Soit $s \in \mathbb{N}^*$. Soit $\mathscr{C}_s : \mathbb{F}_2^{s+m} \to \mathbb{F}_2^{2s}$ l'application linéaire obtenue par restriction. Si pgcd(P_1,P_2)=1 alors*

$$\mathscr{C}_s \text{ est surjective} \Leftrightarrow s \leq m$$
$$\mathscr{C}_s \text{ est injective} \Leftrightarrow s \geq m$$

et

$$rang(\mathscr{C}_s) = \begin{cases} 2s & si\ s \leq m \\ s+m & si\ s \geq m \end{cases}$$

Dès lors que $s > m$, $\mathscr{C}_s$ est injective et non surjective, on cherche donc le plus petit s tel que $\mathscr{C}_{s+1}$ n'est pas surjective. Il existe des séquences de longueur $2s$ qui ne sont pas des séquences codées dans ce cas. La distribution de ces séquences n'est pas uniforme lorsque $s > m$

Le test de profondeur consiste alors à choisir un nombre significatif de séquences de longueur $2s$ (s peut être initialisé à 2). Pour chaque bloc de taille 2, on regarde la probabilité d'apparition des 2 bits suivants. Si la répartition est uniforme, on incrémente s de 1 sinon s correspond à la longueur de contrainte du code. Ce test est applicable pour un canal binaire symétrique de probabilité d'erreur allant jusqu'à 0,05.

4.1.2 Test de *Burrows-Wheeler* et Runs

La transformée de *Burrows-Wheeler* effectue un arrangement sur les données tel que les motifs récurrents d'une séquence observée entraînent des suites de bits identiques en sortie de la transformée. Cette modification de la séquence facilite ensuite la compression par dictionnaire. Elle est ainsi utilisée dans *bzip2*. Les compressions obtenues en appliquant la transformée de Burrows-Wheeler à une séquence codée indiquent que le taux d'erreur et les paramètres de longueur de contrainte sont identifiables. Christophe Chabot a donc élaboré un test basé sur la transformée de *Burrows-Wheeler* et une P_{value} mesurant la variation entre le nombre de runs observés et le nombre de runs théoriques pour une séquence aléatoire.

Il a de plus remarqué que pour des séquences de longueur donnée, générées par des codes de paramètres n, k et m, les P_{value} observées sont identiques quel que soit le code de même paramètres utilisé (codes non dégénérés) et sont fonction du taux d'erreur du canal. Il obtient ainsi des courbes types pour différents paramètres. Ces courbes sont de plus toutes distinctes. Lorsque que la P_{value} observée correspond à plusieurs courbes, on peut augmenter artificiellement le bruit jusqu'à identifier une seule courbe. Ceci donne alors n, k et m ainsi que le taux d'erreur initial.

Nous pouvons également tester le calcul de l'entropie sur une séquence codée lue par blocs de taille t. Lorsque t atteint la longueur nécessaire pour disposer d'une équation de parité, l'entropie calculée vaut bien $t-1$.

Les tests que l'on vient de voir se basent sur le fait que le cardinal des motifs observés lorsque l'on regarde une séquence codée lue par blocs de taille t ne vaut pas 2^t lorsque l'on dépasse une certaine taille de blocs. Donc, pour mesurer cette diminution du nombre de motifs possibles, à partir d'un certain rang, il faut effectuer un comptage sur ces motifs. Mais l'inconvénient est la taille des données nécessaires pour obtenir ce résultat. C'est pourquoi nous nous sommes intéressés ensuite à élaborer une mesure de ce cardinal par comptage de collisions. Ce test repose alors sur le paradoxe des anniversaires.

4.2 Test de collisions

L'objectif de ce chapitre est de décrire dans un premier temps une méthode de reconnaissance des paramètres d'un code convolutif. Elle s'appuie sur le comptage du nombre de collisions dans une séquence binaire lue par bloc. La probabilité de collisions entre deux symboles binaires de taille t, pour des données aléatoires, est de $\frac{1}{2^t}$. Dès lors que les données sont codées, la probabilité de collisions augmente. Le comptage des collisions permet donc de distinguer des données aléatoires de données codées lorsque t est suffisamment grand pour correspondre à la taille d'une équation de parité.

Nous pouvons lire la séquence par blocs disjoints ou par blocs glissants. Mais les équations de parité d'un code convolutif étant valides par blocs glissants (avec un décalage correspondant à la longueur du code) nous tirons parti de ce phénomène en effectuant le comptage par blocs glissants.

Les paramètres du code sont alors identifiés en itérant sur la taille des blocs de lecture. Ce procédé a l'avantage d'avoir un faible coût et d'être résistant au bruit.

Le test que nous effectuons repose sur le paradoxe des anniversaires. Nous observons une séquence binaire par blocs de taille t. Lorsqu'un symbole, de taille t, apparaît i fois dans la séquence on dit qu'il y a $\frac{i.(i-1)}{2}$ collisions. Nous comptons alors le nombre total de collisions que nous notons X_N pour une séquence composée de N blocs. Dans un contexte non bruité cela correspond à

$$X_N = \sum_{i=1}^{N} \sum_{j>i} x(s_i, s_j)$$

où

$$x(s_i, s_j) = \begin{cases} 1 & \text{si } s_i = s_j \\ 0 & \text{sinon.} \end{cases}$$

Nous pouvons alors comparer la probabilité de collisions observée et l'espérance de ce nombre pour des données aléatoires ou codées. En effet pour des données aléatoires, vues par bloc de taille t on a

$$P(s = s') = \frac{1}{2^t}$$

et par linéarité de l'espérance on a :

$$\begin{aligned}\mathbb{E}\left[X_N\right] &= \sum_{i=1}^{N}\sum_{j>i}\mathbb{E}[x(s_i, s_j)] \\ &= \frac{1}{2^t}\left|\{i, j \in [\![1, N]\!], j > i\}\right| \\ &= \frac{N.(N-1)}{2^{t+1}}\end{aligned}$$

Le test statistique se déroule alors comme suit : Soit une séquence binaire observée, composée de N blocs de taille t. Nous comptons le nombre, noté X_{obs}, de collisions observées sur cette séquence. Après normalisation, la probabilité empirique, x_{obs}, de collision, vaut

$$x_{obs} = \frac{2.X_{obs}}{N.(N-1)}$$

Pour observer de manière sûre une collision, pour des données aléatoires, il faut disposer d'au moins $2^t + 1$ symboles soit $t.(2^t + 1)$ bits. Mais d'après le paradoxe des anniversaires on a une collision avec une probabilité supérieure ou égale à α pour $c_\alpha t\sqrt{2^t}$ bits où $c_\alpha = \sqrt{-2\ln(1-\alpha)}$. Lorsque nous avons i équations de parité, il faut disposer de $c_\alpha t\sqrt{2^{t-i}}$ bits pour obtenir au moins une collision avec une probabilité supérieure ou égale à α. Le tableau suivant donne alors les quantités de données nécessaires pour différentes valeurs de α lorsque il existe une équation de parité avec $t = 20$.

α	Nombre de bits nécessaires
1/2	17 051
2/3	21 467
3/4	24 114

Nous nous plaçons dans le cas du canal binaire symétrique de probabilité d'erreur p et nous calculons dans la suite de ce chapitre la probabilité de collisions pour le code de parité ainsi que lorsque les données sont codées par un codeur convolutif et possèdent une équation de parité. Le cas du code de parité n'est pas utile en soit puisque nous nous intéressons uniquement aux codes convolutifs mais il facilite la compréhension de la probabilité de collisions en présence d'une équation de parité.

Dans le cas non bruité, le modèle de probabilité de collisions s'étend lorsqu'il y a plusieurs équations de parité. Il est en revanche plus difficile d'établir des formules génériques pour les probabilités de collisions lorsqu'il existe plusieurs équations de parité. Nous verrons alors expérimentalement pour le canal binaire symétrique que la distinction entre des données codées possédant plusieurs équations de parité et des données aléatoires est claire.

De plus, nous utilisons comme mesure le rapport entre la probabilité de collisions pour des données aléatoires et la probabilité de collisions empirique.

4.3 Espérance du nombre de collisions

Dans un contexte non bruité, deux symboles reçus sont égaux si et seulement si les symboles émis sont égaux. Cependant, dans un canal bruité, on peut obtenir une collision entre deux symboles reçus pour des symboles émis différents et inversement, deux symboles émis égaux peuvent correspondre à des symboles reçus différents.

Nous pouvons donc observer des collisions sur la séquence reçue là où il n'y en avait pas sur la séquence non bruitée et inversement. Nous calculons alors la probabilité de collisions entre deux mots de code bruités pour le code de parité de longueur t. Ensuite, nous calculons cette même probabilité de collisions entre deux symboles ayant la taille d'une équation de parité.

4.3.1 Code de parité

Nous supposons ici que la séquence émise est constituée de mots de code du code de parité de longueur t. Nous notons s et s' deux mots de code du code de parité choisis uniformément dans l'ensemble des mots de code. Ils sont émis dans un canal binaire symétrique de probabilités d'erreur p, et r et r' sont les symboles reçus correspondants. Nous calculons alors la probabilité que r soit égal à r'.

Proposition 4.3.1 *Soient r et r' deux mots de code du code de parité, de taille t, reçus après transmission à travers un canal binaire symétrique de probabilité d'erreur p alors*

$$P\left(r = r'\right) = \frac{1}{2^{t-1}} \sum_{\substack{i=0 \\ i \ pair}}^{t} \left(p^2 + (1-p)^2\right)^{t-i} (2p(1-p))^i \binom{t}{i}$$

Démonstration : Tout d'abord dans un bloc de taille t, le dernier bit dépend des $t-1$ précédents par définition du code de parité. Donc il existe seulement 2^{t-1} blocs distincts.

Lorsque le canal n'est pas bruité, c'est à dire $p = 0$, $P(r = r') = P(s = s') = \frac{1}{2^{t-1}}$.

Notons

$$s = s_0 s_1 \dots s_{t-i-1} s_{t-i} \dots s_{t-1}$$

et

$$s' = s'_0 s'_1 \dots s'_{t-i-1} s'_{t-i} \dots s'_{t-1}$$

Sans perdre de généralité on peut supposer que s et s' sont différents sur les i dernières positions et identiques sur les $t-i$ premières. Appelons r et r' les versions bruitées de s et s' respectivement. Alors

$$P(r = r') = P(r_0 = r'_0,\ \dots\ ,\ r_{t-i-1} = r'_{t-i-1},\ r_{t-i} = r'_{t-i},\ \dots\ ,\ r_{t-1} = r'_{t-1})$$

Ces évènements sont indépendants dans un canal binaire symétrique donc

$$P(r = r') = \prod_{i=0}^{t-1} P(r_i = r'_i)$$

Alors

$$\begin{aligned} P(r_0 = r'_0) &= P(r_0 = s_0, r'_0 = s'_0) + P(r_0 = \overline{s_0}, r'_0 = \overline{s_0}') \\ &= (1-p)^2 + p^2 \end{aligned}$$

et

$$\begin{aligned} P(r_{t-1} = r'_{t-1}) &= P(r_{t-1} = s_{t-1}, r'_{t-1} = \overline{s_{t-1}}') + P(r_{t-1} = \overline{s_{t-1}}, r'_{t-1} = s'_{t-1}) \\ &= 2p(1-p) \end{aligned}$$

De plus

$$P(r_0 = r'_0) = \ \ldots \ = P(r_{t-i-1} = r'_{t-i-1})$$

et

$$P(r_{t-i} = r'_{t-i}) = \ \ldots \ = P(r_{t-1} = r'_{t-1}).$$

D'où pour s et s' fixé

$$P(r = r') = \left((1-p)^2 + p^2\right)^{t-i} (2p(1-p))^i .$$

Or la différence de s et s' est de poids pair puisque la somme de deux mots de code est un mot de code. Donc il y a $\sum_{\substack{i=0 \\ i \text{ pair}}}^{t} \binom{t}{i}$ possibilités de positionner cette différence. D'où

$$P(r = r') = \frac{1}{2^{t-1}} \sum_{\substack{i=0 \\ i \text{ pair}}}^{t} \left((1-p)^2 + p^2\right)^{t-i} (2p(1-p))^i \binom{t}{i}$$

□

Remarque 4.3.2 *Remarquons que lorsque* $p = 0$

$$\sum_{\substack{i=0 \\ i \ pair}}^{t} \left((1-p)^2 + p^2\right)^{t-i} (2p(1-p))^i \binom{t}{i} = 1$$

donc on a bien $P(r = r') = \frac{1}{2^{t-1}}$ *dans ce cas. Ce qui correspond bien à l'intuition que l'on a.*

Donc, lorsque $p = 0$, la probabilité de collision est deux fois plus élevée pour le code de parité que pour des données aléatoires. Nous pouvons alors distinguer aisément des données codées par le code de parité, de données aléatoires, sur des blocs de taille t, quitte à itérer sur t. Dans ce cas, les blocs sont lus de manière disjointe naturellement.

Notons P_C la probabilité de collision pour le code de parité et $P_{aléa}$ la probabilité de collision, pour des données aléatoires. Le rapport $\frac{P_C}{P_{aléa}}$ est représenté dans la FIGURE 4.1 pour différents niveaux de bruit. Nous remarquons que lorsque le canal est bruité jusqu'à $p = 10^{-3}$, nous distinguons clairement la présence d'un code car le rapport des probabilités de collisions est proche de 2. Cependant, lorsque le bruit augmente, nous voyons qu'il sera toujours aisé de distinguer de très petits codes tandis que pour les codes de plus grandes longueurs (au delà de $n = 10$), cette distinction risque d'être moins claire voire impossible.

Nous nous intéressons dans la suite de ce chapitre à la reconnaissance des paramètres d'un code convolutif. Soit $\mathscr{C}$ un code convolutif possédant une équation de parité de taille t, nous montrons à présent qu'il est possible de distinguer la présence d'une telle équation en observant la probabilité de collision sur des blocs de taille t. Nous établissons pour cela la probabilité de collisions attendue.

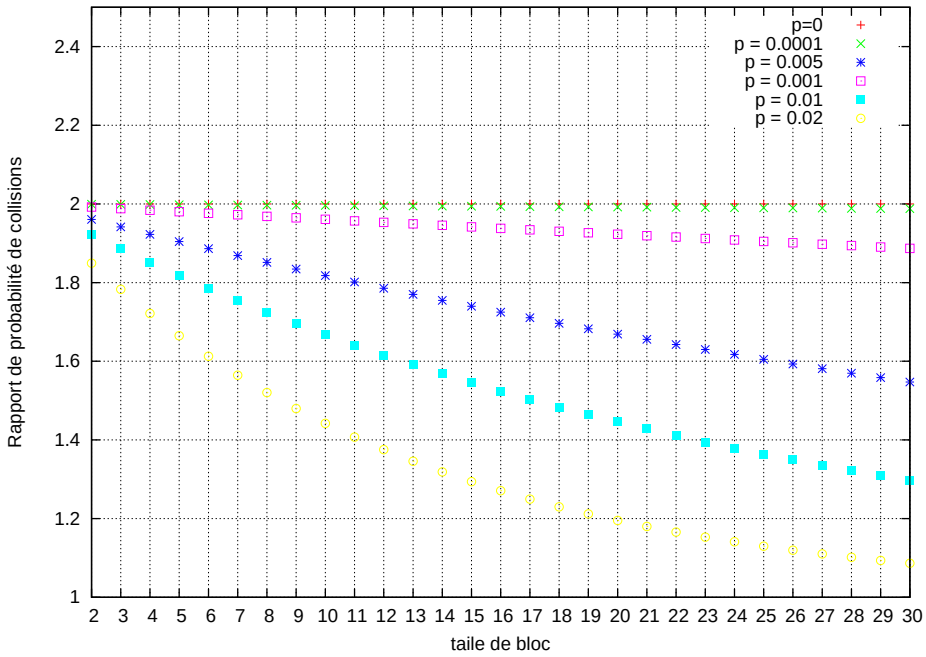


FIGURE 4.1 – Rapports de probabilité de collision pour les codes de parité de taille t

4.3.2 Équation de parité de poids u

Notons u le poids de l'équation de parité. Nous pouvons voir un bloc de taille t comme un mot de code du code de parité de longueur u auquel s'ajoutent $t-u$ bits non contraints. Le nombre de symboles possibles est donc de 2^{t-1} et le nombre de mots vérifiant l'équation de parité est de

$$\sum_{\substack{i=0 \\ i \text{ pair}}}^{u} \binom{u}{i} \cdot \sum_{j=0}^{t-u} \binom{t-u}{j}$$

Nous notons à présent s et s' deux mots de longueur t, vérifiant une équation de parité de poids u, choisis uniformément dans l'ensemble des mots vérifiant cette équation. Il sont émis dans un canal binaire symétrique de probabilités d'erreur p, et r et r' sont les symboles reçus correspondants comme précédemment. Nous calculons alors la probabilité que r soit égal à r'.

Proposition 4.3.3 *Soient r et r' deux symboles, de taille t, contenant le support d'une équation de parité, reçus après transmission à travers un canal binaire symétrique de probabilité d'erreur p alors avec $q = 2p(1-p)$*

$$P(r = r') = \frac{1 + (1-2q)^u}{2^t}$$

Démonstration : La démonstration est similaire à la démonstration faite pour le code de parité. Le nombre de mots vérifiant une équation de parité de poids u est

$$\sum_{\substack{i=0 \\ i \text{ pair}}}^{u} \binom{u}{i} \cdot \sum_{j=0}^{ut} \binom{t-u}{j}.$$

Les symboles s et s' diffèrent sur un certain nombre de positions et cette différence se répartit sur les bits intervenants dans l'équation de parité et les bits n'intervenant pas dans l'équation de parité. Donc on peut calculer la probabilité de collisions entre les symboles r et r' correspondants de manière indépendante sur les bits de l'équation de parité et sur les bits restants. On obtient donc

$$P(r = r') = \frac{1}{2^{t-1}} \left(\sum_{\substack{i=0 \\ i \text{ pair}}}^{u} \binom{u}{i} (1-q)^{u-i} q^i \right) . \left(\sum_{j=0}^{t-u} \binom{t-u}{j} (1-q)^{t-u-j} q^j \right) .$$

Or

$$\left(\sum_{j=0}^{t-u} \binom{t-u}{j} (1-q)^{t-u-j} q^j \right) = 1.$$

D'autre part, $(1-2q)^u = ((1-q)-q)^u$ et $1 = ((1-q)+q)$. En développant ces deux terms avec la formule du binôme de Newton, on obtient que

$$\sum_{\substack{i=0 \\ i \ pair}}^{u} \binom{u}{i} (1-q)^{u-i} q^i = \frac{1+(1-2q)^u}{2}.$$

D'où le résultat.

□

Remarque 4.3.4 *Nous avons bien*

$$P(r = r') = \frac{1}{2^{t-1}}$$

lorsque $p = 0$

On peut en déduire qu'en l'absence de bruit, la seule donnée de la probabilité de collision sur un bloc de taille t ne permet pas de différencier le code de parité et la présence d'une unique équation de parité. Cependant, nous avons vu dans le Chapitre 3 que les équations de parité d'un code convolutif sont valides par blocs glissants (avec un décalage correspondant à la longueur du code). Donc, nous appliquons notre méthode de comptage par blocs glissants. De plus, en itérant sur les tailles de blocs de lecture de la séquence codée, nous pourrons observer une multitude d'équations de parité. Nous verrons qu'il est alors aisé de distinguer la longueur n du code ainsi que sa dimension.

Nous utilisons comme mesure le rapport $R = \frac{P_C}{P_{aléa}}$. Lorsqu'il existe une équation de parité de longueur t et de poids u alors

$$R = 2. \sum_{\substack{i=0 \\ i \text{ pair}}}^{u} \binom{u}{i} (1-q))^{u-i} (q)^i$$

Ce rapport de probabilité de collisions ne dépend donc pas de la taille de l'équation mais uniquement de son poids. Il est alors identique au rapport des probabilités de collisions pour le code de parité de longueur u. Remarquons que les probabilités de collisions que nous avons calculées précédemment sont directement reliées à la notion d'énumérateur des poids. En effet, pour un code $\mathscr{C}$ de longueur

n l'énumérateur des poids donne une description du nombre de mots codes de poids 0, 1, ..., n. On note a_i le nombre de mots de code de poids i. Alors, l'énumérateur des poids est défini comme suit :

$$W_{\mathscr{C}}(x) = \sum_{i=0}^{n} a_i x^i$$

Si l'on considère le code de parité de longueur u alors

$$W_{\mathscr{C}}(x) = \sum_{i=0}^{u} a_i x^i.$$

Or le nombre de mots de poids i est de $\binom{u}{i}$ lorsque i est pair et vaut 0 sinon donc

$$W_{\mathscr{C}}(x) = \sum_{\substack{i=0 \\ ipair}}^{u} \binom{u}{i} x^i$$

et enfin

$$(1-q)^u W_{\mathscr{C}}\left(\frac{q}{1-q}\right) = \sum_{\substack{i=0 \\ ipair}}^{u} \binom{u}{i} q^i \left(1-q\right)^{u-i}.$$

Donc lorsqu'il existe une équation de parité de longueur t et de poids u on a

$$P(r = r') = \frac{\left(1-q\right)^u}{2^{t-1}} W_{\mathscr{C}}\left(\frac{q}{1-q}\right)$$

avec $\mathscr{C}$ est le code de parité de longueur u.

Par ailleurs, $(1-q)^u W_{\mathscr{C}}\left(\frac{q}{1-q}\right)$ correspond au nombre de fois où un mot vérifiant une équation de parité de poids u est envoyé à travers un canal binaire symétrique de probabilité d'erreur q vers un mot r vérifiant cette même équation de parité. Cela correspond donc au nombre de fois où l'on ne détectera pas d'erreur en réception pour un canal binaire symétrique de probabilité d'erreur q.

Les valeurs des rapports de probabilités de collisions pour différents u sont données en annexe. Le tableau suivant en donne quelques résultats.

p	u	P_c/P_{alea}	p	u	P_c/P_{alea}	p	u	P_c/P_{alea}
	1	1.999600		1	1.996004		1	1.960400
	2	1.999200		2	1.992024		2	1.922368
	3	1.998801		3	1.988060		3	1.885842
10^{-4}	4	1.998401	10^{-3}	4	1.984112	10^{-2}	4	1.850763
	5	1.998002		5	1.980179		5	1.817073
	6	1.997603		6	1.976262		6	1.784717
	7	1.997204		7	1.972361		7	1.753642
	8	1.996805		8	1.968476		8	1.723798
	9	1.996406		9	1.964606		9	1.695135
	10	1.996008		10	1.960751		10	1.667608
	11	1.995609		11	1.956912		11	1.641171
	12	1.995211		12	1.953088		12	1.615780
	13	1.994813		13	1.949279		13	1.591395
	14	1.994415		14	1.945486		14	1.567976
	15	1.994017		15	1.941708		15	1.545484
	16	1.993620		16	1.937945		16	1.523883
	17	1.993222		17	1.934197		17	1.503137
	18	1.992825		18	1.930464		18	1.483213
	19	1.992428		19	1.926746		19	1.464078
	20	1.992031		20	1.923042		20	1.445700
	21	1.991634		21	1.919354		21	1.428051
	22	1.991238		22	1.915680		22	1.411100
	23	1.990841		23	1.912021		23	1.394820
	24	1.990445		24	1.908377		24	1.379185
	25	1.990049		25	1.904747		25	1.364170
	26	1.989653		26	1.901131		26	1.349749
	27	1.989257		27	1.897531		27	1.335899
	28	1.988861		28	1.893944		28	1.322597
	29	1.988466		29	1.890372		29	1.309822
	30	1.988071		30	1.886814		30	1.297553

Nous remarquons que lorsque le poids de l'équation u augmente, le rapport P_c/P_{alea} diminue pour un taux d'erreur donné. Lorsque le taux d'erreur est de l'ordre de 10^{-3}, le rapport des probabilités de collisions est significatif quelque soit le poids des équations de parité (comprise entre 2 et 30). En revanche, pour $p = 10^{-2}$, ce rapport est moins révélateur pour les grandes équations de parité. En effet, plus le taux d'erreur augmente plus le rapport des probabilités de collisions diminue.

Ces valeurs obtenues pour différents u et p peuvent être utilisées afin de définir des seuils de tolérance sur le rapport des probabilités de collisions en fonction du taux d'erreur. La section suivante donne la méthode qui permet de décider si les données observées sont des données aléatoires ou des données codées par un codeur convolutif.

4.4 Distingueur de code convolutif

Nous avons vus précédemment que lorsque chaque bloc de taille de t contient le support d'une équation de parité de poids u, la probabilité de collisions entre deux blocs est de $\frac{1}{2^{t-1}}$ dans le cas non bruité et de $\frac{1+(1-2q)^u}{2^t}$ dans le cas bruité contre $\frac{1}{2^t}$ pour des données aléatoires. Pour distinguer si des données sont codées, ou non, il suffit alors de distinguer la distribution du nombre de collisions lorsque les données sont aléatoires et celle du nombre de collisions lorsque chaque bloc de taille t contient le support d'une équation de parité. Par ailleurs chaque bloc de taille t contient le support d'une équation de parité s'il existe une équation de cette longueur et si le décalage entre les blocs correspond à la longueur du code.

Notons $Q = \frac{1}{2^t}$, $P = \frac{1+(1-2q)^u}{2^t}$ et N' le nombre de couples de blocs observés. Nous faisons alors l'approximation suivante : le nombre de collisions pour des données aléatoires suit une loi binomiale de paramètres N' et Q, tandis que le nombre de collisions pour des données codées (chaque bloc de taille t contient le support d'une équation de parité de poids u) suit une loi binomiale de paramètres N' et P. Il s'agit en effet d'une approximation puisque nous regardons la séquence observée par blocs glissants, dans ce cas ces blocs ne sont pas indépendants.

Nous devons alors distinguer la distribution du nombre de collisions pour des données aléatoires et celle obtenue pour des données codées. En comptant le nombre de collisions X_N et en définissant un seuil de décision T, nous pouvons alors distinguer ces deux distributions selon le test suivant.

$$\begin{cases} X_N > T & \text{On décide que les données sont codées} \\ X_N \leq T & \text{On décide que les données sont aléatoires} \end{cases}$$

En revanche, il est possible de décider avec ce procédé que des données sont codées alors qu'elles sont aléatoires (c'est que l'on appelle une fausse-alarme) et inversement (c'est que l'on appelle la non-détection). Les probabilités de fausse-alarme P_{fa} et de non détection P_{nd} correspondent aux formulations suivantes :

$P_{fa} = \{P\left(X_N > T\right) \mid \text{Les données sont aléatoires}\}$

$P_{nd} = \{P\left(X_N \leq T\right) \mid \text{Chaque bloc de taille contient le support d'une équation de parité}\}$

Les valeurs choisies pour ces probabilités donnent alors la valeur du seuil de décision. Elles donnent également la valeur de N' et par conséquent le nombre minimal de blocs N nécessaire à la réussite de ce test. Nous faisons alors une seconde approximation en considérant que le nombre de colliions pour des données aléatoires suit une loi normale d'espérance QN' et d'écart-type $\sqrt{N'Q(1-Q)}$, tandis le nombre de collisions suit une loi normale d'espérance PN' et d'écart-type $\sqrt{N'P(1-P)}$ lorsque chaque bloc de taille t contient le support d'une équation de parité. Nous pouvons alors choisir $T = \frac{QN'+PN'}{2}$ dans un premier temps et déterminer la valeur minimale de N' de telle sorte que les probabilités de fausse-alarme et de non-détection soient inférieures à des valeurs choisies. Les valeurs minimales trouvées pour N' sont différentes selon que l'on regarde la probabilité de fausse-alarme ou la probabilité de non détection, on pourra alors ajuster la valeur du seuil de décision. On pourra se référer à l'algorithme 1 de [8] qui présente un procédé de ce type. Cette méthode donne le nombre minimale de bloc à observer pour la réussite du test de distinction de données codées et de données

t	u	$p = 10^{-2}$	$p = 10^{-3}$	$p = 10^{-4}$
12	4	1564	1380	1364
	5	1620	1386	1364
	6	1676	1390	1364
	7	1736	1394	1366
	8	1800	1400	1366
18	5	12 878	11 008	10 840
	6	13 340	11 046	10 844
	7	13 820	11 084	10 848
	8	14 318	11 122	10 852
	9	14 838	11 160	10 856
	10	15 378	11 198	10 858
	11	15 940	11 236	10 862
	12	16 524	11 276	10 866
24	6	106 596	88 248	86 632
	7	110 436	88 552	86 662
	8	114 428	88 856	86 692
	9	118 580	89 160	86 722
	10	122 900	89 466	86 752
	11	127 392	89 774	86 780
	12	132 066	90 082	86 810
	13	136 928	90 392	86 840
	14	141 986	90 704	86 870
	15	147 248	91 016	86 900

TABLE 4.1 – Nombre minimal de bits tel que P_{fa} et P_{nd} soient inférieures ou égales à 10^{-3} avec $n = 2$

aléatoires, par conséquent nous obtenons le nombre minimal de bits consécutifs à observer. En effet, soit n le décalage appliqué entre chaque bloc alors $N' = 1 + \lfloor \frac{N-t}{n} \rfloor$.

La TABLE 4.1 donne le nombre minimal de bits consécutifs à observer pour distinguer les deux distributions de probabilités lorsque P_{fa} et P_{nd} sont inférieures ou égales à 10^{-3} et $n = 2$, pour différentes tailles t de bloc, différents poids u d'équation de parité et enfin différents taux d'erreur. La quantitée de données nécessaire à la distinction de données codées par un code de longueur 2 est donc très faible et très inférieure à celle nécessaire à l'application de la méthode de C. Chabot qui nécessite environ 10^6 bits.

Lorsque nous disposons de t tel que chaque bloc de taille t contient le support d'une équation de parité nous pouvons alors distinguer des données codées de données aléatoires. En pratique nous ne connaissons ni le décalage à appliquer entre les blocs ni la taille des équations de parité. Nous allons alors itérer sur la taille t des blocs et sur le décalage à appliquer entre chaque bloc. Nous verrons alors dans le section suivante que ceci nous permet de détecter la longueur du code ainsi que la longueur des équations de parité du code.

4.5 Détection des paramètres d'un code convolutif par un test statistique

Les figures 4.2, 4.3 et 4.4 permettent d'observer le rapport des probabilités de collisions pour une séquence codée par un code de longueur 3 et de dimension 1 possédant deux équations de parité (l'une de longueur 12, l'autre de longueur 15).

Nous pouvons lire sur ses courbes la longueur du code, sa dimension ainsi que le degré maximum des polynômes générateurs du dual.

4.5.1 Détection de la longueur de code

La détection de la longueur du code peut s'effectuer par blocs disjoints ou par blocs glissants. La FIGURE 4.2 présente le calcul de la probabilité de collision sur des blocs disjoints. Nous pouvons observer qu'il y a une augmentation de R pour des blocs de taille 12, ce qui correspond à la taille de la plus petite équation de parité. A partir de $t = 12$ la séquence n'est plus considérée comme aléatoire par le test de collisions. Il y a ensuite 3, 5 et 7 équations pour $t = 15, 18$ et 21. Cela donne alors une certaine période à la courbe. Cette période correspond à la longueur du code.

La FIGURE 4.3 présente ce calcul pour des blocs glissants avec différents décalages entre les blocs que nous appelons également shifts. On observe la même période (qui donne la longueur du code), mais on voit clairement qu'une erreur sur le shift entre les blocs implique une diminution de R. Le décalage maximisant ici la probabilité de collisions est 3. Pour un shift multiple de 3, on obtiendra une courbe très proche de celle pour 3. Donc, la longueur du code est aussi le plus petit shift qui met à jour une équation de parité. Dans le cas non bruité, c'est le plus petit shift tel que l'on ait un rapport de probabilité de collisions de 2. Pour le cas bruité, il faudra autoriser une marge de tolérance sur ce rapport de probabilité. Il peut être définit en fonction du taux d'erreur.

La FIGURE 4.4 présente le cas de données bruitées. Seules les courbes obtenues avec le bon shift sont données. Nous voyons qu'avec un canal binaire symétrique dont la probabilité d'erreur binaire est 0.01 nous distinguons toujours clairement la taille de la plus petite équation de parité ainsi que des suivantes. Nous en déduirons donc également les paramètres du code.

4.5.2 Détection de la dimension de code

Nous allons à présent étudier la forme du dual d'un code convolutif afin d'obtenir le comportement des équations de parité lorsque l'on augmente la taille des blocs lus. Nous en tirerons une méthode de détection de la dimension du code.

Rappelons qu'un code convolutif de longueur n et de dimension k peut être vu comme un sous-espace vectoriel de $\mathbb{F}_2(D)$. Les travaux concernant la reconnaissance de tels codes sont relativement récents. Nous citerons notamment les thèses de E. Filiol [27] J. Barbier [3] et M. Côte [20]. Bien que E. Filiol ait étudié le problème de reconnaissance d'un code convolutif en terme polynomial, il a montré que ce problème peut s'exprimer comme un système linéaire sur $\mathbb{F}_2$. M. Côte a utilisé cette vision du problème et obtenu des résultats sur la forme d'un code convolutif vu sous forme binaire. Il a ainsi obtenu un algorithme de reconnaissance de codes convolutifs basé sur la résolution de systèmes linéaires sur $\mathbb{F}_2$. Nous utilisons dans ce paragraphe ces notions afin d'expliciter la forme du dual d'un code convolutif vu sous forme binaire.

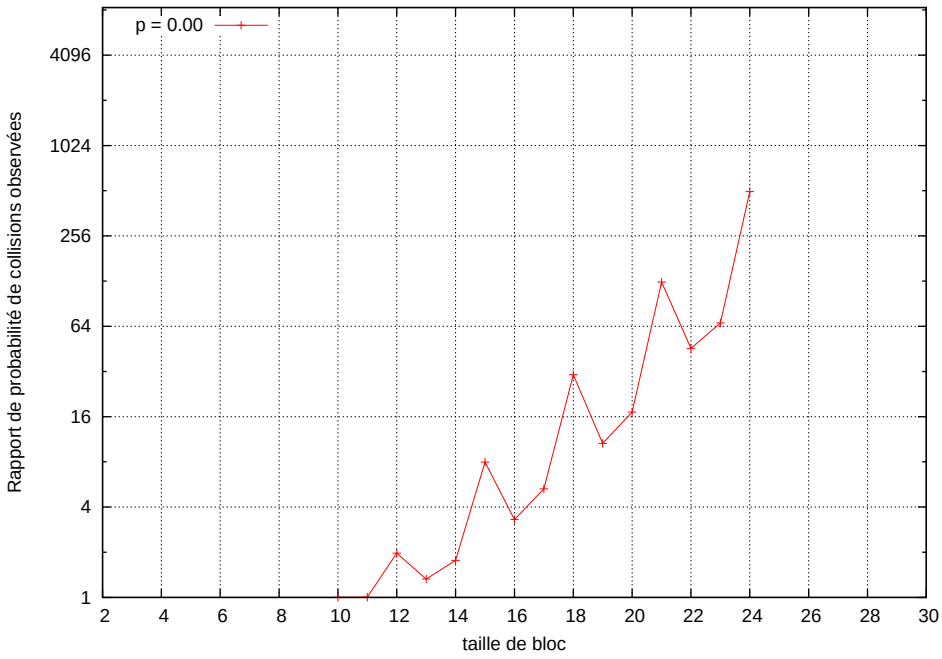


FIGURE 4.2 – Rapport de probabilité de collisions empirique pour code convolutif (3,1) par blocs disjoints

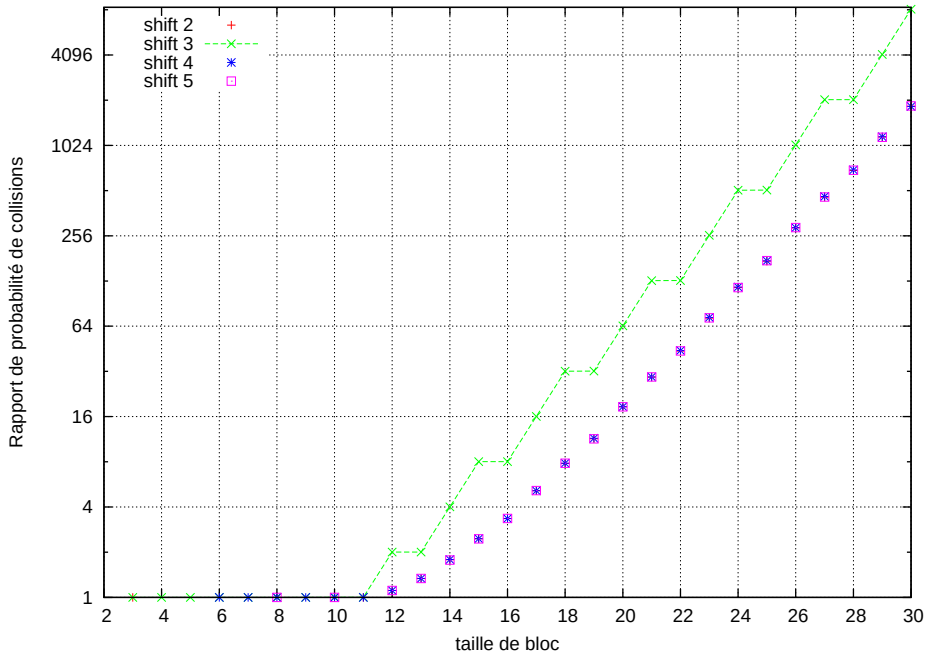


FIGURE 4.3 – Rapport de probabilité de collisions empirique pour un code convolutif (3,1) par blocs glissants

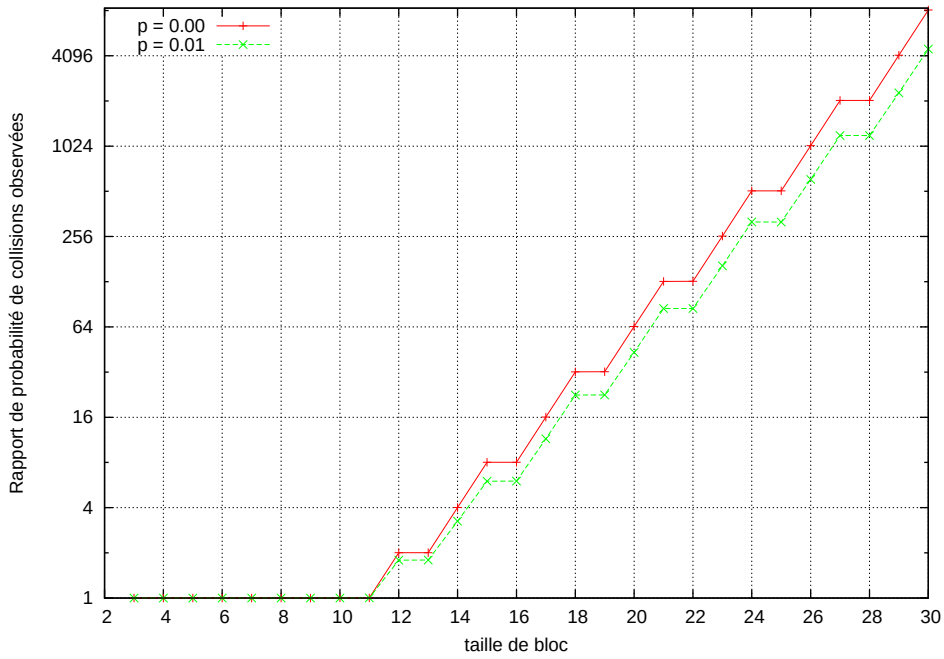


FIGURE 4.4 – Rapport de probabilité de collisions empirique pour un code convolutif (3,1) par bloc glissant sur des données codées bruitées et non bruitées

Code binaire associé à un code convolutif et forme du dual

Soit $\mathscr{C}$ un code convolutif de longueur n et de dimension k. Soit $Y = (Y_1(D), Y_2(D), \ldots, Y_n(D))$ une séquence codée de $\mathscr{C}$.

Pour tout $s \geq 0$, notons $\overline{\mathscr{C}}_s = \{Y \in \mathscr{C} \mid deg\ Y < s\}$. $\overline{\mathscr{C}}_s$ peut être vu comme un sous-espace vectoriel de $\mathbb{F}_2^{sn}$. Soit $\mathscr{C}_s =< \overline{\mathscr{C}}_s >$ le code convolutif engendré par les séquences codées polynomiales de degré strictement inférieur à s.

Proposition 4.5.1 *[21] La suite des codes convolutifs $(\mathscr{C}_s)_{s\geq 0}$ est croissante pour l'inclusion :*

$$\{0\} = \mathscr{C}_v \subsetneq \mathscr{C}_{v+1} \subset \cdots \subset \mathscr{C}_d \subsetneq \mathscr{C}_{d+1} = \mathscr{C}$$

avec $v = e_{min}(\mathscr{C})$ et $d = e_{max}(\mathscr{C})$ ($e_{min}(\mathscr{C})$ et $e_{max}(\mathscr{C})$ sont les indices de Forney minimaux et maximaux de $\mathscr{C}$)
De plus pour tout $s > 0$

$$\dim_{\mathbb{F}_2}(\overline{\mathscr{C}}_s) = \dim_{\mathbb{F}_2}(\overline{\mathscr{C}}_{s-1}) + \dim_{\mathbb{F}_2(D)}(\mathscr{C}_s)$$

où $\dim_{\mathbb{F}_2(D)}(\mathscr{C}_s)$ est la dimension de $\mathscr{C}_s$ en tant que sous-espace vectoriel de $\mathbb{F}_2(D)$

Le dual d'un code convolutif étant lui-même un code convolutif, nous pouvons également écrire

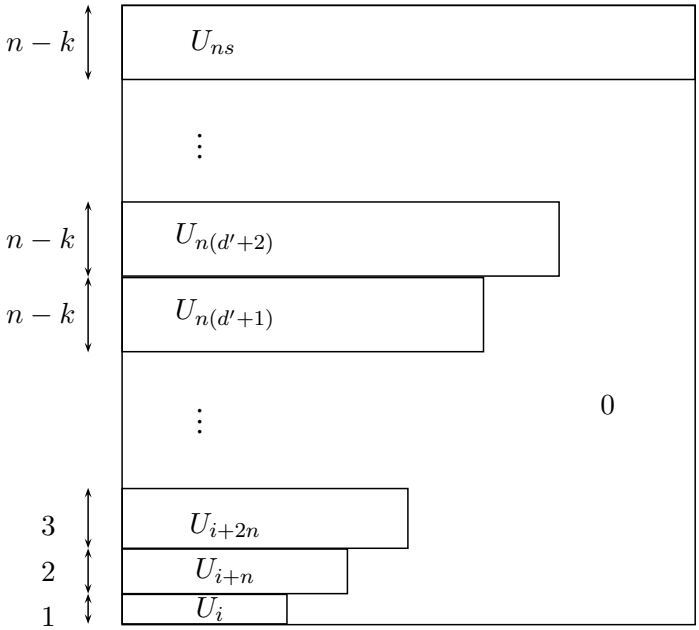
$$\{0\} = \mathscr{C}'^{\perp}_v \subsetneq \mathscr{C}^{\perp}_{v'+1} \subset \cdots \subset \mathscr{C}'^{\perp}_d \subsetneq \mathscr{C}^{\perp}_{d'+1} = \mathscr{C}^{\perp}$$

avec $v' = e_{min}(\mathscr{C}^{\perp})$ et $d' = e_{max}(\mathscr{C}^{\perp})$ où $\mathscr{C}^{\perp}_i$ est le code dual restreint au degré i. Et pour tout $s > 0$

$$\dim_{\mathbb{F}_2}(\overline{\mathscr{C}}^{\perp}_s) = \dim(\overline{\mathscr{C}}^{\perp}_{s-1}) + \dim_{\mathbb{F}_2(D)}(\mathscr{C}^{\perp}_s)$$

où $\dim_{\mathbb{F}_2(D)}(\mathscr{C}_s^{\perp})$ est la dimension de $\mathscr{C}_s^{\perp}$ en tant que sous-espace vectoriel de $\mathbb{F}_2(D)$.

Soit H_{sn} une matrice binaire génératrice de $\mathscr{C}_s^{\perp}$ alors pour tout $s \geq d'+1$ (d' est le degré maximal des polynômes du dual) H_{sn} est composé de sous-matrices de hauteurs croissantes jusqu'à $n-k$:



On peut ensuite extraire une matrice génératrice polynomiale du dual à partir de la plus petite sous-matrice de hauteur $n-k$. Les équations sont lues par bloc de n avec de gauche à droite les coefficients des polynômes dans l'ordre décroissant des degrés. Ces $n-k$ équations forment une base de $\mathscr{C}^{\perp}$ en tant que sous-espace vectoriel de $\mathbb{F}_2(D)$.

Exemple : Soit $\mathscr{C}$ un code de longueur 3 et de dimension 1 tel que

$$U_t = \begin{pmatrix} 101 & 001 & 110 & 101 & 110 \\ 110 & 001 & 010 & 011 & 000 \end{pmatrix}$$

est la plus petite sous-matrice de hauteur $n-k=2$. Les bits des trois premières colonnes sont les coefficients des monômes de degré 4 , les 3 suivants ceux des monômes de degré 3 et ainsi de suite. On obtient alors une matrice génératrice du dual :

$$H = \begin{pmatrix} D^4+D^2+D+1 & D^2+1 & D^4+D^3+D \\ D^4 & D^4+D^2+D & D^3+D \end{pmatrix}$$

Les sous-matrices U_t sont de rang plein, formées de t colonnes et orthogonales à toutes séquences codées $(y_1,\ldots,y_t)$ de taille t. Soit $h=(h_1,\ldots,h_i) \in U_i$ le plus petit bloc de H_{sn} alors $h^{(1)}=(h_1,\ldots,h_i,\underbrace{0\ldots,0}_{n})$ et $h^{(2)}=(\underbrace{0\ldots,0}_{n},h_1,\ldots,h_i)$ sont orthogonaux à toutes séquences codées $(y_1,\ldots,y_i,y_{i+1},\ldots,y_{i+n})$ de longueur $i+n$. Donc, $h^{(1)}$ et $h^{(2)}$ appartiennent à U_{i+n}. Donc, toutes les équations de parité observées sur des blocs de taille i sont observées également sur des blocs de taille

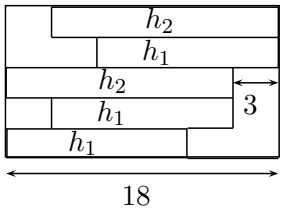
$i+n$. De plus, lorsque $t > n(d'+1)$ alors U_t est composé des éléments de U_{t-n} décalés de n rang vers la droite auxquels on a ajouté des zéros sur la partie de gauche.

Lorsque nous effectuons un test par comptage de collisions sur des tailles de blocs croissantes pour une séquence codée de $\mathscr{C}$, nous regardons également les sous-codes binaires de $\mathscr{C}$. Nous détectons donc les tailles i, $i+n$ jusqu'à $n(d'+1)$ et ainsi de suite. Il est alors possible de déduire la dimension du code. À partir d'un certain rang, le code observé correspond au code $\mathscr{C}$ donc le nombre d'équations supplémentaires obtenu par palier de n est stable. Ce nombre d'équations supplémentaires correspond à $n-k$. La taille de bloc pour laquelle on dispose pour la première fois des équations de parité de $\mathscr{C}$ vaut $n(d'+1)$, on en déduit alors $d'+1$.

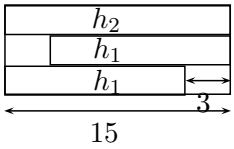
Notons P_{obs} la probabilité de collision pour les données observées, $P_{aléa}$ la probabilité de collisions pour des données aléatoires. Le rapport $R = \frac{P_{obs}}{P_{aléa}}$ sera utilisé dans la suite de ce chapitre pour mesurer le caractère aléatoire ou non d'une séquence lue pour différentes tailles de blocs. Il est également utilisé pour mesurer le nombre d'équations de parité.

Le code $\mathscr{C}$ de longueur 3 et de dimension 1 que l'on a vu précédemment (4.2, 4.3 et 4.4) possède une équation de parité de taille 15 et une équation de taille 12, à savoir un mot du dual de taille 12 noté $h^{(1)}$ et un mot du dual de 15 noté $h^{(2)}$. Comme vu précédemment $\left(h^{(1)}, 0, \ldots, 0\right)$, $\left(0, \ldots, 0, h^{(1)}\right)$ et $h^{(2)}$ sont des mots du dual. Alors les matrices H_{12}, H_{15} et H_{18} ont la forme suivante :

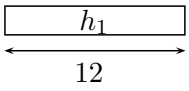
$H_{18} =$



$H_{15} =$



$H_{12} =$



Donc, nous gagnons deux équations de parité à chaque augmentation de la longueur de bloc de 3, c'est-à-dire $n-k$ équations de parité à chaque augmentation de la taille de bloc de n. La plus petite taille de bloc mettant à jour les deux équations de parité nécessaires à engendrer le dual correspond à $15 = n(d'+1)$.

Nous pouvons remarquer que le cas des blocs disjoints met en avant une diminution du rapport des probabilités pour les longueurs 13 et 14. En effet, nous perdons la structure d'équations de parité sur un certain nombre de blocs. En utilisant des blocs glissants de taille 13, nous conservons l'équation observée pour la taille 12 et ainsi de suite.

De plus, l'utilisation de blocs glissants permet d'observer les équations de parité une par une. En effet, si l'on dispose de deux équations de taille t alors l'une est de taille t et l'autre de taille $t-1$ au

plus. C'est ce qu'on allons voir à présent.

Code binaire associé à une équation de parité

Nous regardons ici la forme des sous-matrices U_t vues précédemment. Cela nous permettra d'avoir une vue précise de la forme du dual recherché.

Soit i la plus petite longueur telle qu'il existe une équation de parité de taille i. Soit $(y_1, \dots, y_i)$ un bloc de taille i sur lequel agit une équation de parité alors on peut écrire y_i en fonction des y_1 à y_{i-1}.

Notons $y_i = y_1 g_{i-1} + y_2 g_{i-2} + \cdots + y_{i-1} g_1$ cette équation de parité avec $g_1, \dots, g_{i-1} \in \mathbb{F}_2$, alors

$$(y_1, \dots, y_i) = (y_{i-1}, \dots, y_1) \begin{pmatrix} & & & 1 & g_1 \\ & & 1 & & g_2 \\ & \iddots & & & \vdots \\ 1 & & & & g_{i-1} \end{pmatrix}$$

et

$$(y_1, \dots, y_i) \begin{pmatrix} g_{i-1} \\ g_{i-2} \\ \vdots \\ g_1 \\ 1 \end{pmatrix} = 0$$

Donc, chaque bloc de taille i (par bloc glissant) est un mot de code du code binaire de dimension $i-1$ et de longueur i de matrice génératrice

$$G_i = \begin{pmatrix} & & & 1 & g_1 \\ & & 1 & & g_2 \\ & \iddots & & & \vdots \\ 1 & & & & g_{i-1} \end{pmatrix}$$

et

$$U_i = \begin{pmatrix} g_{i-1} & g_{i-2} & \dots & g_1 & 1 \end{pmatrix}$$

Lorsqu'il existe deux équations de parité (linéairement indépendantes) agissant sur un bloc de taille t, on peut écrire y_t en fonction des y_1 à y_{t-1} de deux manières différentes.

Notons $y_t = y_1 g_{t-1} + y_2 g_{t-2} + \cdots + y_{t-1} g_1$ la première équation de parité avec $g_1, \dots, g_{t-1} \in \mathbb{F}_2$ et alors $y_t = y_1 g'_{t-1} + y_2 g'_{t-2} + \cdots + y_{t-1} g'_1$ avec $g'_1, \dots, g'_{t-1} \in \mathbb{F}_2$. La somme de ces deux équations étant une équation de parité, on peut considérer que $g_1 = 1$ et $g'_1 = 0$ ou $g_1 = g'_1 = 0$ et $g_2 = 1$ et $g'_2 = 0$. Il existe donc deux équations telles que y_{t-1} dépend de y_{t-2} à y_1 et y_t dépend de y_{t-1} à y_1 donc a fortiori de y_{t-2} à y_1.

De manière générale, chaque bloc de taille t admettant j équations de parité linéairement indépendantes est un mot de code du code binaire de dimension $t-j$ et de longueur t de matrice génératrice

$$G_t = \begin{pmatrix} & & 1 & g_{1,1} & \dots & g_{1,i} \\ & 1 & & g_{2,1} & \dots & g_{2,i} \\ \iddots & & & \vdots & & \vdots \\ 1 & & & g_{t-j,1} & \dots & g_{t-j,i} \end{pmatrix}$$

et

$$U_t = \begin{pmatrix} g_{t-j,i} & \cdots & g_{1,i} & & & 1 \\ g_{t-j,i-1} & \cdots & g_{1,i-1} & & 1 & \\ \vdots & & \vdots & \iddots & & \\ g_{t-j,1} & \cdots & g_{1,1} & 1 & & \end{pmatrix}$$

Toutes sommes d'équations de parité étant une équation de parité, la concaténation des sous-matrices $U_{t\leq s}$ forme une matrice H_{sn} de forme systématique et cela sans que le code convolutif soit systématique sous forme polynomiale.

4.5.3 Détection de la longueur des équations de parité

Les itérations sur t permettent de trouver la taille de la plus petite équation de parité et des suivantes. La longueur du code s'obtient alors très facilement en observant une "période" sur la courbe de R pour différentes tailles de bloc ou par l'observation du plus petit shift mettant en avant une équation de parité.

D'autre part, lorsque les données sont codées et non bruitées, le rapport de probabilité R pour des blocs de taille t vaut $2^t/2^{\dim \mathscr{C}_t}$. La dimension de $\mathscr{C}_t$ est de t moins le nombre d'équations de parité sur les blocs de taille t. Le logarithme en base 2 de R nous permet donc de manière générale d'obtenir le nombre d'équations de parité.

De plus, dès lors que le nombre d'équations de parité que l'on gagne à chaque période est stable (suffisamment longtemps) nous disposons de la dimension $n-k$ du code dual et donc de la dimension k du code $\mathscr{C}$. Par ailleurs la plus petite longueur de bloc pour laquelle U_t contient $n-k$ équations de parité est $n(d'+1)$, où d' est le degré maximum des polynômes générateurs du dual. La valeur $d'+1$ est donc connue également.

4.5.4 Résultats de tests

Nous avons effectué des tests de détection de paramètres pour les codes convolutifs suivants en faisant varier t de 4 à 30 pour des séquences codées de longueur $n \times 10^5$. Les résultats sont positifs pour tous les codes testés jusqu'à $p = 10^{-2}$, au-delà nous ne sommes pas en mesure de donner l'ensemble des paramètres des codes. Pour les codes $\begin{pmatrix} 11 & 6 & 11 \\ 6 & 11 & 15 \end{pmatrix}$ et $\begin{pmatrix} 43 & 61 \end{pmatrix}$ on ne discerne aucun paramètre pour $p = 10^{-1}$, pour les deux autres on a seulement une incertitude sur la valeur de $d'+1$.

Code	Taux d'erreur	Détection des paramètres
$\begin{pmatrix} 43 & 61 \end{pmatrix}$	0	✓
	10^{-3}	✓
	10^{-2}	✓
	10^{-1}	✗
$\begin{pmatrix} 11 & 13 & 15 \end{pmatrix}$	0	✓
	10^{-3}	✓
	10^{-2}	✓
	10^{-1}	✗
$\begin{pmatrix} 11 & 6 & 11 \\ 6 & 11 & 15 \end{pmatrix}$	0	✓
	10^{-3}	✓
	10^{-2}	✓
	10^{-1}	✗
$\begin{pmatrix} 43 & 55 & 57 & 61 \end{pmatrix}$	0	✓
	10^{-3}	✓
	10^{-2}	✓
	10^{-1}	✗

Nous avons également effectué ces mêmes tests mais pour des séquences codées de longueurs $n \times 5 \times 10^4$ puis $n \times 10^4$ et des taux d'erreurs de 0, 10^{-2} et 10^{-3}. Les tests pour le code de rendement 2/3 mènent à un échec quel que soit le taux d'erreur tandis que les autres tests permettent de donner les paramètres des codes.

4.6 Application à la reconstruction du dual

Nous nous intéressons à présent à l'application de cette méthode pour la reconstruction du dual d'un code convolutif. Nous avons vu quelle est la forme du dual d'un tel code. Nous pouvons donc reconstruire le dual d'un code convolutif en recherchant la composition de la matrice H_{sn} vue précédemment. Les positions correspondant à la partie systématique de la matrice sont connues, il nous reste alors à déterminer les positions restantes. Nous savons de plus qu'en itérant sur la taille des blocs de lecture, les tailles des différentes équations de parité sont obtenues pas à pas.

Soit t_{min} la taille de la plus petite équation de parité. En poinçonnant un bit de chaque bloc de taille t_{min} il est possible de déterminer si le bit poinçonné intervient ou non dans l'équation de parité. En effet, si ce bit n'intervient pas dans l'équation de parité, il n'est pas déterminé par les autres. En le supprimant, on supprime un degré de liberté et donc le nombre de collisions augmente. A contrario, si le bit supprimé appartient à l'équation de parité, le nombre de collisions ne varie pas. L'observation des variations du rapport de probabilités de collisions donne alors la plus petite équation de parité. Si cela est suffisant pour un code de rendement $k/k+1$, il faut en revanche être en mesure de reconstruire les équations de parité suivantes pour les autres rendements.

La dimension d'un tel code peut être calculée en fonction de la dimension du code non poinçonné et du rang d'un ensemble de colonnes de H_{sn}. La probabilité de collisions observée sur la séquence codée poinçonnée donne alors une indication sur la matrice H_{sn}. C'est cette information que nous utilisons pour reconstruire pas à pas le dual d'un code convolutif.

4.6.1 Reconstruction du dual d'une matrice binaire

Soit $\mathscr{C}$ un code binaire de dimension k et de longueur n de matrice génératrice G et H une matrice du dual sous forme systématique. Soit I un sous-ensemble de $[\![1, n]\!]$. Notons H_I la matrice H restreinte à l'ensemble des colonnes indicées par I. $\mathscr{C}_{\setminus I}$ est le code engendré par $G_{\setminus I}$ la matrice génératrice G poinçonnée sur les colonnes indicées par I.

Le lemme suivant permet de donner la dimension de $\mathscr{C}_{\setminus I}$ en fonction du rang de H_I. Il n'est pas spécifique au cas des codes convolutifs c'est pourquoi nous parlons de reconstruction du dual d'une matrice binaire. Mais nous l'utilisons dans la section suivante pour déterminer le rang des colonnes du dual.

Lemme 4.6.1 $\dim \mathscr{C}_{\setminus I} = \dim \mathscr{C} - |I| + rang\ H_I$.

Démonstration : H est une matrice à $n-k$ lignes et n colonnes s'écrivant sous une forme systématique :

$$H = \begin{pmatrix} g_{k,n-k} & g_{k-1,n-k} & \cdots & g_{1,n-k} & & & 1 \\ g_{k,n-k-1} & g_{k-1,n-k-1} & \cdots & g_{1,n-k-1} & & 1 & \\ \vdots & \vdots & & \vdots & \iddots & & \\ g_{k,1} & g_{k-1,1} & \cdots & g_{1,1} & 1 & & \end{pmatrix}$$

Et G a la forme suivante :

$$G = \begin{pmatrix} & & 1 & g_{1,1} & g_{1,2} & \cdots & g_{1,n-k} \\ & 1 & & g_{2,1} & g_{2,2} & \cdots & g_{2,n-k} \\ \iddots & & & \vdots & \vdots & & \vdots \\ 1 & & & g_{k,1} & g_{k,2} & \cdots & g_{k,n-k} \end{pmatrix}$$

Nous supposons sans perte de généralité que nous poinçonnons G sur les i premières colonnes et sur les j premières colonnes de sa partie non systématique avec $i \leq k$, $j \leq n-k$ et $i+j \leq n-k$. C'est-à-dire $I = \{1, \ldots, i, k+1, k+2, \ldots k+j\}$.

$$\text{Alors } H_I = \left(\begin{array}{ccc|ccc} g_{k,n-k} & \cdots & g_{k-i+1,n-k} & 0 & & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ g_{k,j+1} & \cdots & g_{k-i+1,j+1} & 0 & & 0 \\ \hline g_{k,j} & \cdots & g_{k-i+1,j} & & & 1 \\ \vdots & & \vdots & & \iddots & \\ g_{k,1} & \cdots & g_{k-i+1,1} & 1 & & \end{array}\right)$$

$$\text{donc rang } H_I = j + \text{rang} \begin{pmatrix} g_{k,n-k} & \cdots & g_{k-i+1,n-k} \\ \vdots & & \vdots \\ g_{k,j+1} & \cdots & g_{k-i+1,j+1} \end{pmatrix}$$

$$\text{et } G_{\setminus I} = \left(\begin{array}{ccc|ccc} & & 1 & g_{1,j+1} & \cdots & g_{1,n-k} \\ & 1 & & g_{2,j+1} & \cdots & g_{2,n-k} \\ \iddots & & & \vdots & & \vdots \\ 1 & & & g_{k-i,j+1} & \cdots & g_{k-i,n-k} \\ \hline 0 & \cdots & 0 & g_{k-i+1,j+1} & \cdots & g_{k-i+1,n-k} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & g_{k,j+1} & \cdots & g_{k,n-k} \end{array}\right)$$

d'où

$$\begin{aligned} \text{rang } G_{\setminus I} &= k - i + \text{ rang} \begin{pmatrix} g_{k,n-k} & \cdots & g_{k-i+1,n-k} \\ \vdots & & \vdots \\ g_{k,j+1} & \cdots & g_{k-i+1,j+1} \end{pmatrix} \\ &= k - i - j + \text{ rang} \begin{pmatrix} g_{k,n-k} & \cdots & g_{k-i+1,n-k} \\ \vdots & & \vdots \\ g_{k,j+1} & \cdots & g_{k-i+1,j+1} \end{pmatrix} + j \end{aligned}$$

donc rang $G_{\setminus I}$ = rang $G - |I| +$ rang H_I et dim $\mathscr{C}_{\setminus I}$ = dim $\mathscr{C} - |I| +$ rang H_I □

Ce lemme implique l'identification de colonnes nulles par poinçonnage des colonnes de G une à une, donne un test d'égalité de deux colonnes ainsi qu'un test d'égalité partielle de deux colonnes. Nous détaillons ici ces tests, ils sont ensuite utilisés directement dans la partie 4.6.2 pour la reconstruction du dual d'un code convolutif.

Identification de colonnes nulles En particulier, il est possible de déterminer les colonnes du dual formées de zéros. En poinçonnant la première colonne de G alors

$$H_I = \begin{pmatrix} g_{k,n-k} & g_{k,n-k-1} & \cdots & g_{k,1} \end{pmatrix}^\top$$

Donc dim $\mathscr{C}_{\setminus I}$ = dim $\mathscr{C} - 1$ si et seulement si $H_I = \begin{pmatrix} 0 & 0 & \ldots & 0 \end{pmatrix}^\top$ et vaut dim $\mathscr{C}$ sinon, ceci étant vrai pour les k premières colonnes. Il ne peut y avoir de variation de la dimension en supprimant les colonnes suivantes.

Égalité de deux colonnes du dual Pour $n - k \geq 2$ la suppression des deux premières colonnes de G teste l'égalité des deux premières colonnes de H. En effet dim $\mathscr{C}_{\setminus I}$ = dim $\mathscr{C} - 2$ lorsque ces deux colonnes sont nulles, dim $\mathscr{C}_{\setminus I}$ = dim $\mathscr{C} - 1$ lorsque les deux premières colonnes de H sont identiques ou que l'une des deux est nulle, dim $\mathscr{C}_{\setminus I}$ = dim $\mathscr{C}$ sinon.

Ensuite la suppression de la première colonne de G et sa première colonne non systématique permet de donner le rang de

$$H_I = \begin{pmatrix} g_{k,n-k} & 0 \\ \vdots & \vdots \\ g_{k,2} & 0 \\ g_{k,1} & 1 \end{pmatrix}$$

Et dim $\mathscr{C}_{\setminus I} = \dim \mathscr{C} - 1$ si et seulement si H_I possède deux colonnes identiques ou une colonne nulle. Or les colonnes nulles sont connues, nous pouvons donc les éliminer des colonnes à tester.

Le premier test est valide pour toutes paires de colonnes parmi les k premières. Le second pour toutes paires de colonnes dont l'une est systématique et l'autre non systématique.

Égalité partielle de colonnes du dual Pour $n - k \geq 3$ on peut supprimer les colonnes de G trois par trois. Comme précédemment, les colonnes nulles sont connues donc nous pouvons considérer qu'elles ne sont pas utilisées ici. On peut soit tester la présence de zéros à partir d'un certain rang i dans une colonne en supprimant une colonne systématique de G et $i-1$ colonnes non systématiques, soit tester l'égalité de deux colonnes à partir du rang i en supprimant deux colonnes de G systématiques et $i-1$ colonnes non systématiques.

4.6.2 Reconstruction du dual d'un code convolutif

Lorsque les données sont non bruitées et non poinçonnées, le rapport de probabilité R pour des blocs de taille t vaut $2^t/2^{\dim \mathscr{C}_t}$. Notons $R_{\setminus I} = 2^t/2^{\dim \mathscr{C}_{t\setminus I}}$ le rapport de probabilité obtenu avec poinçonnage, alors $R_{\setminus I}/R$ mesure la variation du rang de H_I. Nous déduisons donc de chaque poinçonnage des informations sur H_I. De plus, nous avons vu précédemment que par bloc glissant, nous observons les équations de parité une à une. Nous pouvons donc les déterminer de proche en proche en augmentant la taille des blocs. Nous avons vu que l'identification des paramètres d'un code convolutif est possible dans le cas où les données observées sont bruitées. Cette méthode est également valide dans ce cas.

Première équation La première équation est déterminée par poinçonnage des bits un à un sur les blocs de taille t. Tous les poinçonnages entraînant une variation de R impliquent que les bits correspondant sont à 0 dans l'équation de parité en question. Nécessairement, les poinçonnages n'entraînant pas de variation du rapport des probabilités de collisions correspondent aux bits à 1 de l'équation de parité.

Seconde équation La seconde équation est déterminée de la manière suivante : On identifie les colonnes $\begin{pmatrix}0 & 0\end{pmatrix}^\top$ et $\begin{pmatrix}1 & 0\end{pmatrix}^\top$ en poinçonnant de nouveau un à un les bits de chaque bloc. D'autre part, il existe une colonne $\begin{pmatrix}0 & 1\end{pmatrix}^\top$ puisque le dual a une forme systématique. Les colonnes identiques sont donc identifiées en poinçonnant la position correspondant à $\begin{pmatrix}0 & 1\end{pmatrix}^\top$ et la position correspondant à la colonne à déterminer. Les colonnes valant $\begin{pmatrix}1 & 1\end{pmatrix}^\top$ sont celles qui n'entraînent pas de variation de R lors de ce test d'égalité de colonnes.

Équations supplémentaires Lorsque 3 équations sont présentes : On connaît les deux dernières lignes du dual, il reste à déterminer uniquement la première ligne. Les colonnes $\begin{pmatrix}0 & 0 & 0\end{pmatrix}^\top$ et $\begin{pmatrix}1 & 0 & 0\end{pmatrix}^\top$ sont identifiées par poinçonnage des bits un à un. Les colonnes $\begin{pmatrix}0 & 0 & 1\end{pmatrix}^\top$ et $\begin{pmatrix}0 & 1 & 0\end{pmatrix}^\top$

sont obtenues par identification de colonnes, ce qui donne également les colonnes $\begin{pmatrix}1 & 1 & 0\end{pmatrix}^\top$ (pas de variation de R par identification avec $\begin{pmatrix}0 & 1 & 0\end{pmatrix}^\top$) et $\begin{pmatrix}1 & 0 & 1\end{pmatrix}^\top$ (pas de variation de R par identification avec $\begin{pmatrix}0 & 0 & 1\end{pmatrix}^\top$). Les colonnes restantes ${}^\top\begin{pmatrix}0 & 1 & 1\end{pmatrix}$ et $\begin{pmatrix}1 & 1 & 1\end{pmatrix}^\top$ sont identifiées par la donnée du rang de $\begin{pmatrix}0 & 0 & x\\ 0 & 1 & 1\\ 1 & 0 & 1\end{pmatrix}$

Cette méthode est particulièrement efficace pour les codes de rendement $k/k+1$ puisqu'elle ne nécessite qu'une identification soit $n(d'+1)$ comptages de collisions, connaissant les paramètres du code. Elle est de plus résistante au bruit. La FIGURE 4.5 donne le rapport des probabilités de collisions obtenues en poinçonnant tour à tour les bits des blocs de taille 21 pour le code $\begin{pmatrix}11 & 6 & 11\\ 6 & 11 & 15\end{pmatrix}$. Les positions donnant un rapport de probabilité proche de la valeur de référence (2 ici) sont les positions correspondant à des bits à 1 dans l'équation de parité, les autres correspondent aux bits à 0. L'équation obtenue est donc

$$111100011000100110111$$

ce qui correspond à $\begin{pmatrix}D^6+D^5+D^2+D+1 & D^6+D^4+D+1 & D^6+D^4+1\end{pmatrix}$.

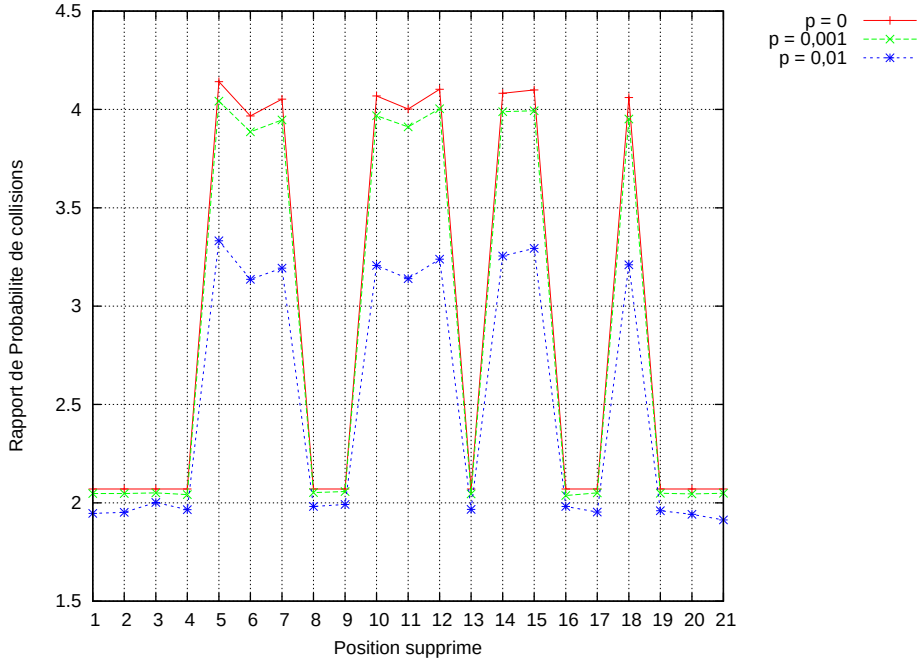


FIGURE 4.5 – Rapport de probabilité de collisions par poinçonnage

4.7 Signature de codes convolutifs à mapping inconnu

Le chapitre suivant aborde la question de la reconnaissance de mappings en présence de données codées (par un codeur convolutif) et bruitées. Un algorithme de reconstruction de codes convolutifs est utilisé comme testeur afin d'identifier les différents mappings possibles. Nous y définissons un algorithme de reconnaissance de mappings mais il est alors fondamental de s'assurer que la séquence d'origine est bien une séquence codée par un codeur convolutif. La connaissance a priori de la longueur du code utilisé serait de plus intéressante pour accélérer les tests de reconnaissances de codes effectués.

Il s'avère que pour certaines longueurs de blocs de lecture, la probabilité de collisions est invariante quel que soit le mapping. Cette propriété conduit à l'observation de la signature d'un code convolutif à mapping inconnu, elle permet de plus d'en déduire la longueur du code. Bien que ce travail ne soit pas utilisé dans le Chapitre 6 il est un atout considérable dans la recherche du mapping.

Prenons t tel que chaque bloc lu soit composé de μ symboles de taille a. Prenons r et r' deux blocs identiques de taille t. Lors d'un changement de mapping (application d'une bijection de $\mathbb{F}_2^a$), chaque symbole de taille a est envoyé vers un nouveau symbole binaire, mais r et r' étant composés des mêmes symboles ils sont toujours identiques après changement de mapping. Le nombre de collisions et donc la probabilité de collisions sont maintenues pour tout changement de mapping lorsque t est un multiple de a.

En revanche, nous pouvons perdre des collisions lorsque a ne divise pas t. Le test de collisions sera alors appliqué pour les tailles de blocs multiples de la taille des symboles du mapping. Ceci permet de caractériser la présence d'un code convolutif sans connaissance à priori du mapping.

D'autre part, nous avons vu que la longueur du code correspond au plus petit shift mettant à jour une équation de parité. On note t_{min} la longueur de cette équation. Le nombre de collisions, sur une séquence obtenue avec le mauvais mapping, est maintenu notamment pour $t = ppcm(a, t_{min})$. Un tel t est alors la plus petite taille de bloc permettant de voir une équation de parité, nous distinguerons donc n sur cette longueur.

Chapitre 5

Classes de Mappings

Nous nous intéressons dans cette thèse à la reconstruction de systèmes de communications dans un contexte non coopératif. Nous étudions l'influence du mapping sur cette reconnaissance et plus particulièrement la reconnaissance de celui-ci.

Rappelons qu'une modulation M-aire peut transmettre M symboles binaires distincts formant ainsi une constellation dans un espace euclidien bidimensionnel. Nous regardons ici les modulations 4 et $8-PSK$ ainsi que les modulations de 16 à $256-QAM$ vues au Chapitre 1 (Figure 1.4, 1.5 et 1.6). On note $a = log_2\ M$.

En l'absence de redondance il est impossible de décider quel mapping a été utilisé. C'est pourquoi nous supposons que les données binaires sont codées. Un algorithme de reconnaissance de code, spécifique au type de code recherché, peut être utilisé en tant que testeur. L'un des critères permettant de valider un mapping est donc la détection d'un code.

La première approche consiste à parcourir l'ensemble des mappings possibles, pour une modulation donnée, et à effectuer pour chacun une recherche de code. Le problème de cette approche est le nombre de mappings à parcourir, décrit par la Table 5.1

Pour $a = 4$, le nombre de mappings est supérieure à 2^{44}. Pour chacun d'eux, il faut effectuer une recherche de code, et en estimant à 1 seconde cette recherche il faudrait plus de 500 000 années de calcul. Nous ne pouvons donc pas envisager de recherche exhaustive au delà de $a = 3$. Nous avons alors déterminé des classes d'équivalences de ces mappings, de sorte que nous puissions effectuer une seule recherche de code par classe. Ces classes permettent de réduire considérablement le coût de la recherche du mapping. Cette méthode, présentée dans le Chapitre 6, est malgré tout insuffisante lorsque la taille de la constellation dépasse $a = 4$. Nous avons en effet plus de 2^{89} classes à parcourir pour $a = 5$. Dans ce cas nous restreignons notre recherche à des mappings Gray ou quasi-Gray. La méthode s'adapte alors très bien à cette contrainte et permet la reconstruction de mappings Gray

Table 5.1 – Nombre de mappings pour $a = 2, 3, 4, 6, 8$

a	Nombre de mappings
2	24
3	40 320
4	$> 2^{44}$
6	$> 2^{295}$
8	$> 2^{1683}$

pour des constellations ayant jusqu'à 256 points. Nous distinguerons les cas de constellations 32 et $128 - QAM$.

Nous décrivons ici deux relations d'équivalences. Elles définissent des classes d'équivalences dites linéaires et affines. Les classes linéaires forment une sous-partition des classes affines et nous disposons ainsi d'une classification à deux niveaux. Nous décrivons ensuite la répartition des mappings Gray dans ces classes et décrivons la répartition de certains mappings quasi-Gray également.

Rappelons que l'on note $\mathcal{C}$ une constellation, a le nombre de bits par symbole binaire de celle-ci. $\mathbb{F}_2$ est le corps à deux éléments et que $GL(a, \mathbb{F}_2)$ est l'ensemble des matrices inversibles $a \times a$ à coefficients dans $\mathbb{F}_2$. Enfin I_a représente la matrice identité de taille $a \times a$.

5.1 Relations d'équivalences

Nous définissons ici deux relations d'équivalences pour un nombre a de bits par symbole donné.

Relation linéaire : On dit que deux mappings f et f' sont *linéairement* équivalents, noté par $f' \in C_L(f)$, si et seulement si il existe une matrice inversible $\mathcal{L}$ de taille $a \times a$ telle que pour tout $P \in \mathcal{C}$, $f'(P) = f(P).\mathcal{L}$

Relation affine : On dit que deux mappings f et f' sont *affinement* équivalents, noté par $f' \in C_A(f)$, si et seulement si il existe une matrice inversible $\mathcal{L}$ de taille $a \times a$ et un vecteur binaire v de longueur a tels que pour tout $P \in \mathcal{C}$, $f'(P) = f(P).\mathcal{L} + v$

Ces relations sont des relations d'équivalences. Ainsi, elles définissent des partitions de l'ensemble des mappings pour une constellation donnée. Le nombre de matrices inversibles de taille $a \times a$, dans un corps fini, nous donne directement le nombre d'éléments dans une classe linéaire :

$$|C_L(f)| = \prod_{i=0}^{a-1} \left(2^a - 2^i\right)$$

Le nombre de mappings dans une classe affine s'obtient en comptant le nombre de vecteur v de $\mathbb{F}_2^a$:

$$|C_A(f)| = 2^a . \prod_{i=0}^{a-1} \left(2^a - 2^i\right)$$

Ces nombres sont donnés pour quelques valeurs de a dans les TABLE 5.2 et 5.3. Rappelons que le nombre de mappings possibles pour une constellation à 2^a points est de $2^a!$

TABLE 5.2 – Répartition linéaire pour $a = 2 \ldots 8$

a	2	3	4	5	6	7	8
Nombre de mappings	24	40320	$> 2^{44}$	$> 2^{117}$	$> 2^{295}$	$> 2^{715}$	$> 2^{1683}$
Nombre de classes linéaires	4	240	1 037 836 800	$> 2^{94}$	$> 2^{261}$	$> 2^{668}$	$> 2^{1621}$
Nombre de mappings par classe linéaire	6	168	20 160	$> 2^{23}$	$> 2^{34}$	$> 2^{47}$	$> 2^{62}$

TABLE 5.3 – Répartition affine pour $a = 2 \dots 8$

a	2	3	4	5	6	7	8
Nombre de mappings	24	40320	$> 2^{44}$	$> 2^{117}$	$> 2^{295}$	$> 2^{715}$	$> 2^{1683}$
Nombre de classes affines	1	30	64 864 800	$> 2^{89}$	$> 2^{255}$	$> 2^{661}$	$> 2^{1613}$
Nombre de mappings par classe affine	24	1344	322 560	319 979 520	$> 2^{40}$	$> 2^{54}$	$> 2^{70}$

Soit f un mapping de $\mathcal{C}$ dans $\mathbb{F}_2^a$, $\mathcal{L}$ une matrice inversible de $GL(a, \mathbb{F}_2^a)$, v un élément de $\mathbb{F}_2^a$. On note respectivement $f + v$ et $f.\mathcal{L}$ les mappings tels que :

$$\begin{aligned} f + v : \mathcal{C} &\to \mathbb{F}_2^a \\ P &\mapsto f(P) + v \end{aligned}$$

$$\begin{aligned} f \,.\, \mathcal{L} : \mathcal{C} &\to \mathbb{F}_2^a \\ P &\mapsto f(P) \,.\, \mathcal{L} \end{aligned}$$

Le mapping $f \,.\, \mathcal{L} + v$ est également définit par $(f \,.\, \mathcal{L} + v)(P) = f(P)\mathcal{L} + v$.

5.2 Partitionnement

Les classes d'équivalences linéaires forment une sous-partition des classes affines. En effet lorsque deux mappings sont linéairement équivalents, ils le sont affinement. La propriété suivante nous permet de donner une description de la répartition des classes linéaires dans les classes affines et une manière de générer les classes linéaires qui est utilisée dans le Chapitre 6.

Proposition 5.2.1

$$\bigcup_{v \in \mathbb{F}_2^a} (C_L(f + v)) = C_A(f)$$

et

$$C_L(f + v) \cap C_L(f + v') = \emptyset$$

pour tout $v \neq v' \in \mathbb{F}_2^a$

Démonstration :

Montrons que $C_L(f + v) \cap C_L(f + v') = \emptyset$ pour tout $v \neq v' \in \mathbb{F}_2^a$.
Soit $g \in C_L(f + v)$, $v \in \mathbb{F}_2^a$, il existe $\mathcal{L} \in GL(a, \mathbb{F}_2)$ telle que $\forall P \in \mathcal{C}$, $g(P) = f(P)\mathcal{L} + v\mathcal{L}$.
Alors $g \in C_L(f + v')$, $v' \in \mathbb{F}_2^a$ si et seulement si il existe $\mathcal{L}' \in GL(a, \mathbb{F}_2)$ telle que $\forall P \in \mathcal{C}$, $g(P) = f(P)\mathcal{L}' + v'\mathcal{L}'$.

Donc $g \in C_L(f + v')$ si et seulement si $f(P)\mathcal{L} + f(P)\mathcal{L}' = v\mathcal{L} + v'\mathcal{L}'$.

Soit P_z tel que $f(P_z) = (0, ..., 0)$, on obtient en remplaçant P par P_z dans la précédente équation $v\mathcal{L} = v'\mathcal{L}'$ et $\mathcal{L} = \mathcal{L}'$ d'où $v = v'$ également.

Donc

$$C_L(f + v) \cap C_L(f + v') = \emptyset$$

pour tout $v \neq v' \in \mathbb{F}_2^a$

Par ailleurs

$$\bigcup_{v \in \mathbb{F}_2^a} (C_L(f + v)) \subset C_A(f)$$

et les deux ensembles ont même cardinal, ce qui entraîne que

$$\bigcup_{v \in \mathbb{F}_2^a} (C_L(f + v)) = C_A(f)$$

□

Il y a 2^a classes linéaires par classes affines. La répartition globale des mappings dans les classes linéaires et affines est donnée dans la TABLE 5.4

a	2	3	4	5	6	7	8
Nombre de mappings	24	40320	$> 2^{44}$	$> 2^{117}$	$> 2^{295}$	$> 2^{715}$	$> 2^{1683}$
Nombre de classes affines	1	30	64 864 800	$> 2^{89}$	$> 2^{255}$	$> 2^{661}$	$> 2^{1613}$
Nombre de classes linéaires par classe affine	4	8	16	32	64	128	256

TABLE 5.4 – Répartition linéaire et affine pour $a = 2 \dots 8$

5.3 Représentants

Nous obtenons un mapping par classe linéaire de la manière suivante : Étant donné un mapping f et $C_A(f)$ sa classe affine, $C_A(f)$ est partitionnée en 2^a classes linéaires pour lesquelles l'ensemble $\{f + v_1, \dots, f + v_{2^a}\}$, $v_i \in \mathbb{F}_2^a$ est un ensemble de représentants.

En effet supposons que $f + v_i \in C_L(f + v_j)$, alors $\forall P \in \mathcal{C}$ il existe $\mathcal{L} \in GL(a, \mathbb{F}_2)$ telle que $(f + v_i)(P) = (f + v_j)(P).\mathcal{L}$.

Soit P_z tel que $f(P_z) = (0, \dots, 0)$ alors $(f + v_i)(P_z) = (f + v_j)(P_z).\mathcal{L}$ si et seulement si $v_i = v_j\mathcal{L}$ donc $f(P) = f(P)\mathcal{L}$ pour tout $P \in \mathcal{C}$.

Et finalement $f + v_i \in C_L(f + v_j)$ si et seulement si $\mathcal{L} = I_a$ la matrice identité et $v_i = v_j$.

Ceci nous donne une manière naturelle de parcourir un mapping par classe linéaire à partir d'un jeu de représentants des classes affines. Il nous reste alors à obtenir un ensemble de ces représentants.

La méthode, qui consiste à choisir aléatoirement (avec remise) un mapping puis un second tant que ce dernier appartient à la classe affine du premier et ainsi de suite jusqu'à obtenir un jeu de représentants, correspond au problème du collecteur de coupons. En effet, soit X le nombre de mappings

tirés tels qu'il y en ait au moins un par classe affine. Soit X_i le nombre de mappings tirés pour passer de $i-1$ classes représentées à i classes représentées.

Nous avons pour une modulation à 8 points, c'est-à-dire 30 classes affines :

$$X_1 \ = \ 1$$

et

$$P(X_i \ = \ x) \ = \ \left(1 - \frac{30-(i-1)}{30}\right)^{x-1} . \frac{30-(i-1)}{30}$$

X_i suit une loi géométrique de paramètre $p_i \ = \ \frac{30-(i-1)}{30}$ et

$$\mathbb{E}[X] \ = \ \sum_{i \ = \ 1}^{30} \mathbb{E}[X_i] \ = \ \sum_{i \ = \ 1}^{30} \frac{1}{p_i} \ = \ 30 \sum_{i \ = \ 1}^{30} \frac{1}{30-(i-1)}$$

Pour une constellation à 8 points $\mathbb{E}[X]$ vaut alors 119.849613928, c'est-à-dire qu'il faudrait tirer en moyenne 120 mappings puis vérifier le nombre de classes obtenues.

Nous avons cependant observé pour les constellations à 8 points qu'en fixant les labels de certains points nous restreignons l'espace de recherche des représentants affines de manière conséquente. En fixant la position des symboles binaires $(0,0,0)$, $(0,0,1)$ et $(0,1,0)$, le nombre de mappings à 8 points décrit est de $(8-3)! = 120$. Ces 120 éléments sont formés de 4 éléments par classes affines, comme le montre la proposition suivante. Cet espace décrit alors 30 classes affines distinctes, ce qui correspond au nombre total de classes affines recherchées dans ce cas.

Proposition 5.3.1 *Soient P_1, P_2 et $P_3 \in \mathcal{C}$. Soient f et f' deux mappings de $\mathcal{C}$ dans $\mathbb{F}_2^3$ tels que*

$$\begin{array}{rcl} f(P_1) \ = \ f'(P_1) \ = \ (0,0,0) \\ f(P_2) \ = \ f'(P_2) \ = \ (0,0,1) \\ f(P_3) \ = \ f'(P_3) \ = \ (0,1,0) \end{array} \tag{5.1}$$

alors $f' \ \in \ C_A(f)$ si et seulement si il existe une matrice inversible $\mathcal{L}$ de la forme $\begin{pmatrix} 1 & l_1 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ avec $l_1, \ l_2 \ \in \mathbb{F}_2$ telle que $f' = f.\mathcal{L}$

Démonstration : $f' \ \in C_A(f)$ si et seulement si

$$\begin{aligned} (0,0,0) &= (0,0,0)\mathcal{L} + v \\ (0,0,1) &= (0,0,1)\mathcal{L} + v \\ (0,1,0) &= (0,1,0)\mathcal{L} + v \end{aligned}$$

La première équation implique $v = 0$, les deux suivantes fixent les deux dernières lignes de $\mathcal{L}$. □

Donc pour f tel que $f(P_1) \ = \ (0,0,0)$, $f(P_2) \ = \ (0,0,1)$ et $f(P_3) \ = \ (0,1,0)$ il existe 3 mappings f' vérifiant ces conditions et tels que $f' \ = \ f.\mathcal{L}$

avec

$$\mathcal{L} \in \left\{ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right\} \tag{5.2}$$

La génération de l'ensemble des mappings vérifiant la condition 5.1 , composé de 120 éléments, permet d'extraire 30 représentants affines par élimination des mappings f' vérifiant la condition 5.2. Ce qui nous donne une méthode déterministe pour la génération des représentants.

Remarque 5.3.2 *Si nous fixons la position des symboles* $(0,0,0)$, $(0,0,1)$, $(0,1,0)$ *et* $(1,0,0)$ *nous obtenons seulement* 24 *mappings différents, représentant* 24 *classes affines distinctes. La démonstration se fait de manière similaire à la démonstration précédente.*

Nous avons étudié cette démarche pour la génération de représentants affines pour une constellation $16-QAM$. Dans ce cas le nombre de classes affines est de 64 864 800. Si l'on fixe la position des trois symboles $(0,0,0,0)$, $(0,0,0,1)$ et $(0,0,1,0)$, le nombre de mappings décrit est de $(16-3)! = 6\,227\,020\,800$. L'espace de recherche des représentants affines est alors trop grand. Dans ce cas nous scindons l'espace de recherche en deux espaces distincts fournissant chacun une partie des représentants. Ils sont explicités par la proposition suivante.

Proposition 5.3.3 *Soient* P_1, P_2, P_3, P_4 *et* $P_5 \in \mathcal{C}$. *Soient* f *et* f' *deux mappings de* $\mathcal{C}$ *dans* $\mathbb{F}_2^4$ *tels que*

$$\begin{array}{rcl} f(P_1) = f'(P_1) & = & (0,0,0,0) \\ f(P_2) = f'(P_2) & = & (0,0,0,1) \\ f(P_3) = f'(P_3) & = & (0,0,1,0) \\ f(P_4) = f'(P_4) & = & (0,1,0,0) \end{array} \tag{5.3}$$

Soient g *et* g' *deux mappings de* $\mathcal{C}$ *dans* $\mathbb{F}_2^4$ *tels que*

$$\begin{array}{rcl} g(P_1) = g'(P_1) & = & (0,0,0,0) \\ g(P_2) = g'(P_2) & = & (0,0,0,1) \\ g(P_3) = g'(P_3) & = & (0,0,1,0) \\ g(P_4) = g'(P_4) & = & (0,0,1,1) \\ g(P_5) = g'(P_5) & = & (0,1,0,0) \end{array} \tag{5.4}$$

alors $f' \in C_A(f)$ *(respectivement* $g' \in C_A(g)$*) si et seulement si il existe une matrice inversible* $\mathcal{L}$ *de la forme* $\begin{pmatrix} 1 & l_1 & l_2 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ *avec* $l_1,\ l_2,\ l_3 \in \mathbb{F}_2$ *telle que* $f' = f.\mathcal{L}$ *(respectivement* $g' = g.\mathcal{L}$*).*

Démonstration : La démonstration est similaire à celle de la proposition 5.3.1. □

Donc pour f tel que $f(P_1) = (0,0,0,0)$, $f(P_2) = (0,0,0,1)$, $f(P_3) = (0,0,1,0)$ et $f(P_4) = (0,1,0,0)$ il existe 7 mappings f' distincts vérifiant la condition 5.3 et tels que $f' = f.\mathcal{L}$ où

$$\mathcal{L} = \begin{pmatrix} 1 & l_1 & l_2 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \; l_1,\ l_2,\ l_3 \ \in \mathbb{F}_2 \text{ et } \mathcal{L} \neq I_4 \tag{5.5}$$

De même pour g tel que $g(P_1) = (0,0,0,0)$, $g(P_2) = (0,0,0,1)$, $g(P_3) = (0,0,1,0)$, $g(P_4) = (0,0,1,1)$ et $g(P_5) = (0,1,0,0)$ il existe 7 mappings g' distincts vérifiant la condition 5.4 et tels que $g' = g.\mathcal{L}$ où $\mathcal{L}$ vérifie la condition 5.5.

Proposition 5.3.4 *Les classes affines décrites par les mappings vérifiant respectivement 5.3 et 5.4 sont distinctes.*

Démonstration : Soient f et g deux mappings de $\mathcal{C}$ dans $\mathbb{F}_2^4$ vérifiant respectivement les conditions 5.3 et 5.4.

f et g sont affinement équivalent si et seulement il existe $\mathcal{L} \in GL(a, \mathbb{F}_2)$ telle que $g(P) = f(P)\mathcal{L}$ de part la position commune du symbole $(0,0,0,0)$. Les lignes 3 et 4 de $\mathcal{L}$ sont fixés par la position commune des symboles $(0,0,0,1)$ et $(0,0,1,0)$.

$g(P_4) = f(P_4)\mathcal{L}$ implique $\mathcal{L} = \begin{pmatrix} l_1 & l_2 & l_3 & l_4 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ et alors $\mathcal{L}$ n'est pas inversible d'où la contradiction. □

La génération de l'ensemble des mappings vérifiant la condition 5.3 composé de $(16-4)! = 479\,001\,600$ mappings peut permettre l'extraction de $(16-4)!/8 = 59\,875\,200$ représentants affines, tandis que l'ensemble des mappings vérifiant la condition 5.4 permet de décrire $(16-5)!/8 = 4\,989\,600$ classes affines distinctes.

Cependant pour $a = 4$ la complexité de l'algorithme de recherche de mappings présenté dans le Chapitre 6 s'avère assez élevée. Nous supposons donc que pour $a \geq 4$ les mappings considérés sont des mappings Gray ou quasi-Gray et étudions dans la suite de ce chapitre la répartition de ces mappings dans les classes linéaires et affines.

5.4 Classification des mappings Gray

Les mappings Gray sont en nombre réduit par rapport au nombre de mappings possibles pour une constellation donnée. Ils sont définit pour les constellations 4 et $8-PSK$, ainsi que pour les constellations QAM rectangulaires. Ils sont décrit dans le Chapitre 2. Leur nombre est rappelé dans la TABLE 5.5.

Constellation	Nombre de mappings Gray
$4-PSK$	8
$8-PSK$	96
$16-QAM$	384
$64-QAM$	414 720
$256-QAM$	584 674 836 480

TABLE 5.5 – Nombre de mappings Gray

Nous avons étudié la répartition des mappings Gray dans les classes linéaires et affines, définies plus tôt, dans le but d'étendre l'algorithme de reconnaissance de mapping, présenté dans le Chapitre 6, au cas des mappings Gray. Nous montrons qu'ils sont contenus dans un nombre relativement restreint de classes affines, qu'ils sont contenus dans l'ensemble des classes linéaires concernées et que le nombre de mappings Gray par classe linéaire est de $a!$.

5.4.1 Partitionnement

Propriété 5.4.1 *Soit f un mapping Gray, soit $f' \in C_A(f)$. Alors f' est un mapping Gray si et seulement si il existe une matrice de permutation $\mathcal{L}$, de taille $a \times a$, et un vecteur $v \in \mathbb{F}_2^a$ tels que $f' = f.\mathcal{L} + v$*

Démonstration : Soit f un mapping Gray, soient $P_i, P_j \in \mathcal{C}$ tels que $d_e(P_i, P_j) = 1$ (distance euclidienne) alors $d_H(f(P_i), f(P_j)) = 1$ (distance de Hamming). Soit $f' \in C_A(f)$, il existe $\mathcal{L} \in GL(a, \mathbb{F}_2)$ et $v \in \mathbb{F}_2^a$ tels que $f' = f.\mathcal{L} + v$.

Si $\mathcal{L}$ est une matrice de permutation, de taille $a \times a$, par définition la multiplication d'un élément de $\mathbb{F}_2^a$ par $\mathcal{L}$ ne modifie pas le poids de cet élément. La distance de Hamming entre deux éléments de $\mathbb{F}_2^a$ est donc conservée :

$$\begin{aligned} d_H(f'(P_i), f'(P_j)) &= w_H(f'(P_i) + f'(P_j)) \\ &= w_H(f(P_i)\mathcal{L} + f(P_j)\mathcal{L}) \\ &= w_H([f(P_i) + f(P_j)]\,\mathcal{L}) \\ &= w_H(f(P_i) + f(P_j)) \\ &= 1 \end{aligned}$$

Donc si $\mathcal{L}$ est une matrice de permutation et $v \in \mathbb{F}_2^a$ alors $f'.\mathcal{L} + v$ est un mapping Gray.

Inversement, supposons que f' est un mapping Gray alors $f'(P_i) + f'(P_j)$ a un poids de Hamming de 1. Il en est de même pour $f(P_i) + f(P_j)$.

Soit k la coordonnée non nulle de $f(P_i) + f(P_j)$. Alors $f'(P_i) + f'(P_j) = (f(P_i) + f(P_j))\mathcal{L}$ est la $k^{\text{ème}}$ ligne de $\mathcal{L}$ et a un poids de Hamming égal à 1.

Or pour tout $k = 1 \,..\, a$, il existe $P_i, P_j \in \mathcal{C}$ tels que $f(P_i)$ et $f(P_j)$ diffèrent à la position k. Sinon une coordonnée serait fixe sur l'ensemble des $f(P)$, $P \in \mathcal{C}$ et f ne serait pas une bijection.

Donc chaque ligne de $\mathcal{L}$ a un poids de 1. De plus $\mathcal{L}$ est une matrice inversible donc $\mathcal{L}$ est une matrice de permutation. □

On en déduit que si une classe affine contient un mapping Gray alors elle en contient $a!2^a$ car $a!$ est le nombre de matrices de permutations de taille $a \times a$. La répartition des mappings Gray est donnée dans la Table 5.6.

Les mappings dit réfléchis sont de plus compris dans une seule et même classe affine. En effet les mappings Gray réfléchis possèdent des symétries qui sont conservées par l'ajout d'un motif constant à l'ensemble des symboles binaires. De plus la multiplication par une matrice de permutation a pour effet de modifier l'ordre d'inversion des bits et éventuellement le symbole d'origine.

Table 5.6 – Distribution des mappings Gray pour $a = 4, 6, 8$

a	4	6	8
Nombre de classes affines contenant des mappings Gray	1	9	56 644
Nombre de classes linéaires par classe affine	16	64	256
Nombre de mappings Gray par classe linéaire	24	720	40 320

5.4.2 Représentants

Étant donné un mapping f de type Gray et $C_A(f)$ sa classe affine, $C_A(f)$ est partitionnée en 2^a classes linéaires pour lesquelles l'ensemble $\{f + v_1, ..., f + v_{2^a}\}$ est un ensemble de représentants tel que chacun des $f + v_i$ soit un mapping Gray. En effet on sait que $\{f + v_1, ..., f + v_{2^a}\}$ est un ensemble de représentants dans le cas général et que l'ajout d'un motif v à l'ensemble des symboles binaires obtenus par f conserve le critère de Gray.

Nous pouvons donc nous poser la question de l'obtention de représentants affines de type Gray. Nous nous appuyons pour cela sur la construction de *Wesel & al.* Rappelons que cette construction définit un mapping Gray comme le produit direct de deux codes Gray (unidimensionnel). Nous énumérons dans un premier temps, à équivalence affine près, les codes Gray (unidimensionnel) de taille 2^b avec $2b = a$. Pour cela nous fixons le symbole d'origine, puis la permutation des bits de chaque symbole.

La Table 5.7 donne le nombre de codes Gray possible pour b bits par symboles ainsi que le nombre d'origines possibles, le nombre de permutations différentes et enfin le nombre de codes représentants.

Codes représentants $b = 3$

Pour obtenir les codes représentants pour $b = 3$, nous fixons le symbole d'origine à $(0, 0, 0)$. Le symbole suivant est à distance de Hamming de 1, c'est donc $(0, 0, 1)$, à permutation près. L'arbre donné par la Figure 5.1 représente les différents codes possibles. On voit que certains chemins explorés sont impossibles. Nous obtenons ainsi trois codes distincts.

b	2	3	4
Nombre de codes Gray	8	144	91392
Nombre d'origines	4	8	16
Nombre de permutations	2!	3!	4!
Nombre de codes Gray représentants	1	3	238

TABLE 5.7 – Classification des codes Gray pour $b = 1, 2, 3$

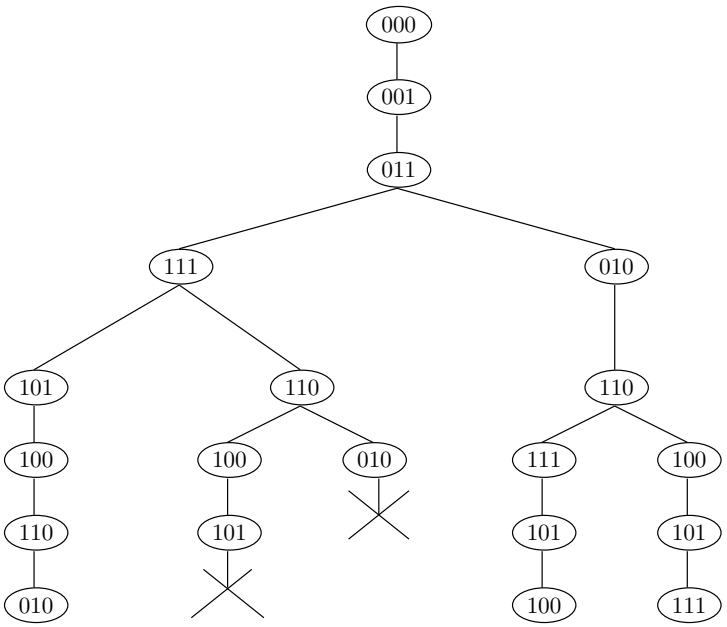


FIGURE 5.1 – Codes Gray représentants à 8 symboles

Propriété 5.4.2 *Ces codes ne sont pas affinements équivalents.*

Démonstration : Soient f et f' deux codes parmi les trois énoncés plus haut. f et f' sont affinement équivalents si et seulement si il existe une matrice de permutation $\mathcal{L}$ de taille 3×3 et un vecteur v de longueur 3 tels que $f' = f.\mathcal{L} + v$. La position du symbole $(0, 0, 0)$ étant fixe alors $v = 0$. La position commune des symboles $(0, 0, 1)$ et $(0, 1, 1)$ fixe les deux dernières lignes de la matrice $\mathcal{L}$ et par conséquent la première également. On en déduit alors que $\mathcal{L}$ est la matrice identité. □

Codes représentants $b = 4$

Pour $b = 4$ nous obtenons un arbre (FIGURE 5.2) décrivant les contraintes fixés aux codes Gray.

En recherchant d'une part les codes commençant par $(0, 0, 0, 0)$, $(0, 0, 0, 1)$, $(0, 0, 1, 1)$ puis $(0, 1, 1, 1)$ et d'autre part ceux commençant par $(0, 0, 0, 0)$, $(0, 0, 0, 1)$, $(0, 0, 1, 1)$ puis $(0, 0, 1, 0)$ et $(0, 1, 1, 0)$ nous obtenons 238 codes distincts.

Propriété 5.4.3 *Ces 238 codes ne sont pas affinements équivalents.*

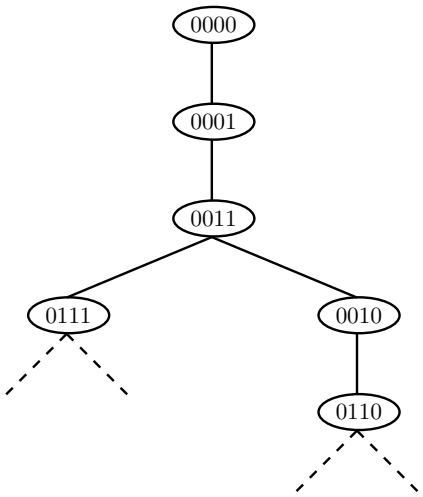


FIGURE 5.2 – Initialisation des codes Gray représentants à 16 symboles

Démonstration :

Soient f et f' deux codes parmi les 238 trouvés. f et f' sont affinement équivalents si et seulement si il existe une matrice de permutation $\mathcal{L}$ de taille 4×4 et un vecteur v de longueur 4 tels que $f' = f.\mathcal{L} + v$. Comme précédemment, on a $v = (0, 0, 0, 0)$.

De plus il n'existe pas de matrice $\mathcal{L}$ telle que $(0, 1, 1, 1) = (0, 0, 1, 0).\mathcal{L}$. Donc deux codes n'appartenant pas à la même branche de l'arbre 5.2 ne sont pas équivalents.

D'autre part, soient deux codes appartenant à la branche gauche de l'arbre alors

$$\begin{cases} (0,0,0,1) & = & (0,0,0,1).\mathcal{L} \\ (0,0,1,1) & = & (0,0,1,1).\mathcal{L} \\ (0,1,1,1) & = & (0,1,1,1).\mathcal{L} \end{cases}$$

d'où $\mathcal{L} = I_4$.

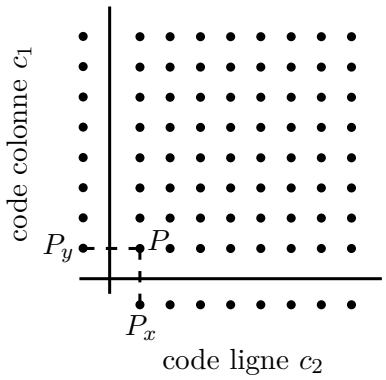
Il en est de même pour deux codes appartenant à la branche de droite avec

$$\begin{cases} (0,0,0,1) & = & (0,0,0,1).\mathcal{L} \\ (0,0,1,1) & = & (0,0,1,1).\mathcal{L} \\ (0,1,1,0) & = & (0,1,1,0).\mathcal{L} \end{cases}$$

d'où $\mathcal{L} = I_4$.

□

Les 238 codes obtenus sont alors des représentants affines. Nous utilisons ensuite la méthode de construction de mappings Gray de *Wesel & al* en concaténant les symboles du code colonne aux symboles du code ligne de la manière suivante :



avec $f(P) = c_1(P_y) \mid c_2(P_x)$

$\mid$ est l'opérateur de concaténation.

La propriété suivante montre que l'utilisation de cette construction pour l'ensemble des couples codes lignes et colonnes représentants permet la génération des mappings Gray représentants affines.

Propriété 5.4.4 *Soient f et f' deux mappings distincts obtenus par produit direct de deux codes représentants affines. Alors $C_A(f) \neq C_A(f')$.*

Démonstration : Comme pour les démonstrations précédentes si f et f' sont affinement équivalents alors $v = 0$ car la position du symbole $(0, \ldots, 0)$ est fixe. De plus ils sont distincts, nous pouvons donc supposer que

$$\begin{aligned} f(P) &= c_1(P_y) \mid c_2(P_x) \\ f'(P) &= c_1(P_y) \mid c_3(P_x) \end{aligned}$$

avec $c_2 \neq c_3$. Donc f et f' sont affinement équivalents si et seulement si c_2 et c_3 le sont. Or nous avons démontré précédemment qu'une telle configuration n'est pas possible. □

Nous sommes donc en mesure de générer un mapping Gray par classes affines contenant des mappings Gray et ce sans avoir à parcourir l'ensemble des mappings Gray.

Nous nous intéressons à présent à la génération de représentants quasi-Gray. Les méthodes présentées dans le Chapitre 2 utilisent des mappings Gray réfléchis rectangulaires. Ces mappings Gray réfléchis sont compris dans une seule et même classe affine.

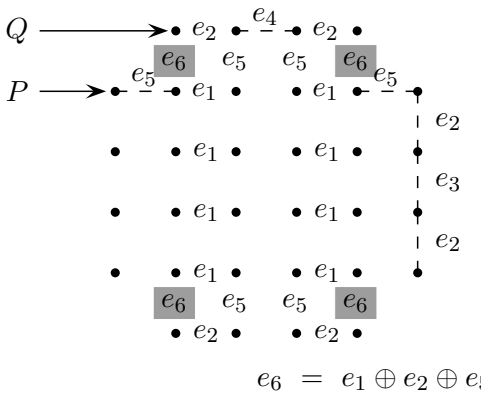
5.5 Classification des mappings quasi-Gray

La multiplication de l'ensemble des symboles d'un mapping quasi-Gray par une matrice de permutation, de même que l'ajout d'un vecteur v, conserve les pénalités de Gray moyenne et maximum. Au même titre que les mappings Gray, deux mappings quasi-Gray f et f' sont affinement équivalents si et seulement pour tout point P de la constellation il existe une matrice de permutation $\mathcal{L}$ et un

vecteur v tels que $f'(P) = f(P)\mathcal{L} + v$. Nous connaissons donc la taille des classes linéaires et affines pour les mappings quasi-Gray. Cependant le nombre de classes affines dépend du nombre de méthode autorisées pour la génération de ces mappings.

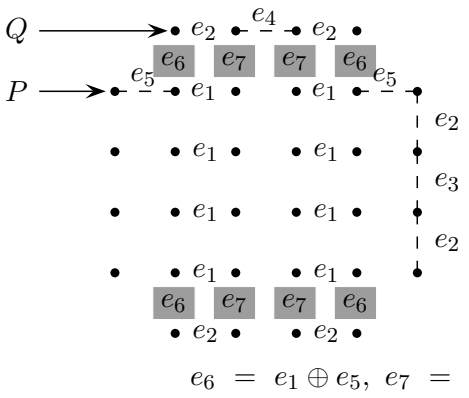
Chaque méthode énoncée dans le Chapitre 2 décrit en effet une classe affine. Le nombre de classes affines correspond alors au nombre de méthodes choisies.

Nous avons présentés précédemment 7 méthodes de construction de mappings quasi-Gray $32-QAM$. Chacune d'entre elles peut se définir par l'utilisation de transition entre les points de la constellation. En fixant les transitions e_1 à e_5 puis un symbole de la constellation, on obtient ainsi 7 mappings distincts. Ces 7 mappings sont alors des représentants affines. En effet ils ne sont pas équivalents, nous pouvons donc décrire 7 classes affines distinctes pour les mappings $32-QAM$. Nous démontrons ici ce fait pour deux méthodes parmi les sept, le raisonnement s'étendant de manière naturelle aux autres classes. De plus nous avons dans la suite de ce manuscrit choisi de nous intéresser aux deux méthodes de génération à partir de mappings rectangles. Nous les rappelons ici :



$$e_6 = e_1 \oplus e_2 \oplus e_5$$

(a) Méthode de *Wesel & al.*



$$e_6 = e_1 \oplus e_5,\ e_7 = e_2 \oplus e_5$$

(b) Méthode *J.G. Smith*

Propriété 5.5.1 *La transformation d'un mapping réfléchi rectangulaire* $32-QAM$ *par ces deux méthodes engendre deux mappings quasi-Gray non équivalent.*

Démonstration :

Soient f le mapping obtenu par la méthode (a), et f' celui obtenu par la méthode (b). Nous avons d'une part :

$$\begin{aligned} f'(Q) &= f'(P) + e_6 + e_5 \\ &= f'(P) + e_1 \end{aligned}$$

et d'autre part :

$$\begin{aligned} f(Q) &= f(P) + e_6 + e_5 \\ &= f'(P) + e_1 + e_2 \end{aligned}$$

d'où $f(Q) = f'(Q) + e_2$.

Or il existe des points fixes communs aux deux méthodes de construction, on en déduit que s'il y a équivalence entre f et f' elle ne peut être que linéaire. Les mappings f et f' ne sont donc pas équivalents. □

Nous disposons de deux méthodes distinctes dans le cas de constellations $128-QAM$. Comme précédemment nous décrivons alors 2 classes affines distinctes et obtenons les représentants en appliquant ces méthodes à un même mapping réfléchi de forme rectangulaire.

La répartition des mappings quasi-Gray dans les classes affines et linéaires et récapitulées dans la Table 6.5.

a	5	7
Nombre de classes affines contenant des mappings quasi-Gray	2	2
Nombre de classes linéaires par classe affine	32	128
Nombre de mappings quasi-Gray par classe linéaire	120	5040

Table 5.8 – Distribution des mappings Gray pour $a = 4, 6, 8$

Nous n'avons pas envisagé la reconstruction de mappings au-delà de constellations à 256 points. Les méthodes de construction de mappings quasi-Gray s'étendent cependant pour les constellations $512-QAM$, mais les classes linéaires de taille $9! = 362880$ serait alors un frein pour le déroulement de notre méthode. Elle est cependant possible pour des constellations de 4 et $8-PSK$, puis pour des mappings Gray ou quasi-Gray de 16 à $256-QAM$.

Chapitre 6

Reconstruction de mappings en présence de données codées et bruitées

Le problème que nous étudions ici est celui de la recherche du mapping utilisé lors de la modulation, dans un contexte non coopératif. Nous considérons pour cela des modulations de 4 à 256 points. En l'absence de redondance il est impossible de décider quel est le mapping de modulation utilisé. Nous supposons alors que les données sont codées et il devient faisable d'utiliser la structure de code pour sélectionner les mappings possibles. Nous nous sommes principalement intéressés au cas de données bruitées, codées à l'aide d'un code convolutif. Nous présentons donc ici un algorithme de reconnaissance de mapping lorsque les données sont codées par un code convolutif et bruitées. Les démonstrations présentées dans ce chapitre sont donc appliquées au cas des codes convolutifs. Cependant cet algorithme est adaptable pour d'autres types de codes. La fin de ce chapitre est donc consacrée à la mise en oeuvre de notre méthode pour la reconnaissance de mapping lorsque les données sont codées par un code de Reed-Solomon et sont non bruitées.

Dans un premier temps nous nous consacrons au cas des codes convolutifs et utilisons l'algorithme de reconnaissance de code convolutif, présenté dans le Chapitre 3, comme un testeur. Lorsque nous disposons du bon mapping, nous sommes en mesure de reconstruire le code convolutif utilisé. La première approche consiste donc à parcourir l'ensemble des étiquetages possibles pour une modulation donnée, à effectuer pour chacun de ces mappings une reconnaissance de code convolutif et utiliser comme critère de décision la présence d'un code. Nous verrons que ce critère n'est pas suffisant pour effectuer une décision et que les paramètres du code trouvé ainsi que le taux d'erreur du canal associé seront également à prendre en compte.

Le problème évident de cette méthode est le nombre d'étiquetage à parcourir. Ce nombre est rappelé dans la Table 6.1 pour quelques valeurs de a. Une approche exhaustive du problème n'est donc pas envisageable pour des modulations à plus de 8 points. D'autre part nous constatons que certains étiquetages provoquent un échec lors de la reconnaissance d'un code convolutif, c'est-à-dire que nous n'en trouvons aucun. D'autres permettent la reconstruction d'un code. Le second problème est alors l'équivalence de certains mappings au regard de la reconnaissance de codes. Notamment une transformation linéaire du mapping confère une certaine stabilité au test de reconnaissance de code convolutif.

TABLE 6.1 – Nombre de mappings pour $a = 2, 3, 4, 6, 8$

a	Nombre de mappings
2	24
3	40 320
4	$> 2^{44}$
6	$> 2^{295}$
8	$> 2^{1683}$

Nous proposons alors un algorithme de reconstruction du mapping, pour des données codées par un code convolutif et bruitées, utilisant les classes d'équivalences linéaires et affines définies dans le Chapitre 5. Ces classes nous permettent de réduire le coût de la recherche en appliquant un test de reconnaissance de code convolutif, de manière appropriée, à un ensemble de représentants des classes d'équivalences. Nous verrons que le résultat d'un tel parcours n'est pas unique mais qu'il permet de réduire significativement le nombre de mappings possibles tout en réduisant le coût de la recherche.

Les classes linéaires forment une sous-partition des classes affines. Nous savons de plus décrire un élément par classe. Nous proposons alors d'effectuer une reconnaissance de code convolutif par classe affine puis une par classe linéaire selon le résultat obtenu à l'étape précédente. Et enfin nous explorons entièrement les classes linéaires sélectionnées.

Cette méthode est malgré tout insuffisante lorsque la taille de la constellation augmente. Dans ce cas nous restreignons notre recherche à des étiquetages de type Gray. La méthode s'adapte alors très bien à cette contrainte et permet la reconstruction pour des constellations ayant jusqu'à 256 points.

Nous l'avons de plus testée pour des données codées par un code de Reed-Solomon et non bruitées. Nous supposons alors que nous connaissons certains paramètres du code, mais la méthode semble s'étendre correctement pour ce type de code.

Rappelons que la notation a représente le nombre de bits par symboles du mapping, ce qui correspond à une modulation ayant 2^a symboles. Nous supposons que le signal observé a été démodulé en utilisant un mapping par défaut et donc que la modulation est connue. Nous disposons alors d'une séquence binaire dont le mapping n'est a priori pas le bon, sur laquelle des modifications sont appliquées, par bloc de taille a bits, pour effectuer des changements de mapping.

Il est à noter que le travail présenté dans le Chapitre 4 concernant la signature des codes convolutifs et notamment la reconnaissance de la longueur d'un code convolutif lorsque le mapping est inconnu n'est pas utilisé ici. En effet l'algorithme présenté dans le présent chapitre a été élaborée précédemment à la méthode de reconnaissance de codes convolutifs dont il est question dans le Chapitre 4. Nous proposons donc un algorithme de reconnaissance de mapping sans aucune connaissance sur le code convolutif utilisé.

6.1 Classes d'équivalences

Notons f_0 le mapping utilisé. Pour tout mapping linéairement équivalent nous obtenons une reconnaissance de code positive (avec des matrices génératrices différentes et éventuellement des paramètres différents). C'est cette propriété qui nous permet d'effectuer une seule recherche de code convolutif par classe linéaire. Pour le mapping f_0, les paramètres obtenus n_0, k_0 et le taux d'erreur p_0 du canal, sont minimaux parmi l'ensemble des paramètres trouvés lors de l'exploration de l'ensemble des mappings. Cependant il existe un certain nombre de mappings linéairement équivalents fournissant ces mêmes paramètres, ils seront alors indistinguables.

6.1.1 Équivalence linéaire

Rappelons que deux mappings f et f' de $\mathcal{C}$ dans $\mathbb{F}_2^a$ sont dits *linéairement équivalents* lorsqu'il existe $\mathcal{L} \in GL(a, \mathbb{F}_2)$ telle que $f' = f.\mathcal{L}$. On note dans ce cas $f' \in C_L(f)$.

Soit S une séquence binaire obtenue par démodulation (à 2^a points). Notons

$$S = s_{*,1} \dots s_{*,a} | s_{*,1} \dots s_{*,a} | \dots\dots | s_{*,1} \dots s_{*,a}.$$

C'est-à-dire que la séquence est la concaténation des symboles issus de la démodulation. L'indice de temps n'étant pas nécessaire à la compréhension nous le noterons $*$ pour plus de simplicité.
Soient n la longueur du code $\mathscr{C}$ utilisé lors de la transmission et $ppcm(a, n) = \lambda n = \mu a$. On pose

$$\mathcal{L}^{[\mu]} = \begin{pmatrix} \mathcal{L} & & 0 \\ & \ddots & \\ 0 & & \mathcal{L} \end{pmatrix}$$

la matrice de taille $\mu a \times \mu a$ formée de la matrice inversible $\mathcal{L}$ répétée μ fois. Lorsque S est une séquence codée par un (n, k) codeur convolutif, elle peut aussi être vue comme étant produite par un $(\lambda n, \lambda k)$ codeur convolutif correspondant au regroupement (λ fois) du codeur (n, k). Nous montrons ici que toute transformation linéaire de S conserve le rendement k/n et qu'il en existe un certain nombre conservant également la longueur de code. Il nous faut pour cela nous intéresser aux codes regroupés.

Une description des codes regroupés est donnée dans [38]. La définition suivante en est extraite, nous avons cependant adapté les notations à nos besoins.

Définition 6.1.1 (Décomposition polyphase) *Soit $g(D) = a_0 + a_1 D + a_2 D^2 + \cdots$ un polynôme. Pour tout $M \geq 1$, la $M^{ème}$ décomposition polyphase de $g(D)$ est donnée par*

$$\left(g^{(1)}(D), g^{(2)}(D), \dots, g^{(M)}(D)\right) = \left(\sum_{k\geq 0} a_{kM} D^k,\ \sum_{k\geq 0} a_{kM+1} D^k, \dots,\ \sum_{k\geq 0} a_{kM+(M-1)} D^k\right)$$

La $j^{ème}$ composante est appelée la $(j, M)^{ème}$ composante polyphase

M. Cluzeau et M. Finiasz dans [17] donnent la forme d'un code regroupé pour un code de longueur 2 et de dimension 1. Nous reprenons ces notations et donnons ici la forme de la matrice génératrice

d'un code regroupé λ fois, pour un code $\mathscr{C}$ de longueur n et de dimension k quelconques. On note $C^{[\lambda]}$ un tel code et $C^{[\lambda]}\mathcal{L}^{[\mu]}$ le code obtenu par multiplication de la matrice génératrice de $C^{[\lambda]}$ par $\mathcal{L}^{[\mu]}$.

Soit $G(D) = \begin{pmatrix} g_{1,1}(D) & g_{1,2}(D) & \dots & g_{1,n}(D) \\ \vdots & \vdots & & \vdots \\ g_{k,1}(D) & g_{k,2}(D) & \dots & g_{k,n}(D) \end{pmatrix}$ une matrice génératrice polynomiale de $\mathscr{C}$.

Notons $G^{(i)} = \begin{pmatrix} g^{(i)}_{1,1}(D) & g^{(i)}_{1,2}(D) & \dots & g^{(i)}_{1,n}(D) \\ \vdots & \vdots & & \vdots \\ g^{(i)}_{k,1}(D) & g^{(i)}_{k,2}(D) & \dots & g^{(i)}_{k,n}(D) \end{pmatrix}$ où $g^{(i)}_{l,c}(D)$ est la $(i,\lambda)^{\text{ème}}$ composante polyphase de $g_{l,c}(D)$.

Alors

$$G^{[\lambda]} = \begin{pmatrix} G^{(1)} & G^{(2)} & G^{(3)} & \dots & G^{(\lambda-1)} & G^{(\lambda)} \\ DG^{(\lambda)} & G^{(1)} & G^{(2)} & \dots & G^{(\lambda-2)} & G^{(\lambda-1)} \\ DG^{(\lambda-1)} & DG^{(\lambda)} & G^{(1)} & \dots & G^{(\lambda-3)} & G^{(\lambda-2)} \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & \ddots & \vdots \\ DG^{(2)} & DG^{(3)} & DG^{(4)} & \dots & DG^{(\lambda)} & G^{(1)} \end{pmatrix}.$$

Notons à présent

$$M = \begin{pmatrix} G^{(\lambda)} \\ G^{(\lambda-1)} \\ G^{(\lambda-2)} \\ \vdots \\ \vdots \\ G^{(1)} \end{pmatrix}$$

et

$$Z = \left(\begin{array}{ccc|ccc} & & & 1 & & \\ & & & & \ddots & \\ & & & & & \ddots \\ \hline D & & & & & 1 \\ & \ddots & & & & \\ & & \underbrace{D}_{k} & & & \end{array}\right).$$

Z est une matrice de taille $\lambda k \times \lambda k$. Alors $G^{[\lambda]}$ correspond à la concaténation des $Z^j M$ avec $j = \lambda - 1, \cdots, 0$, c'est-à-dire

$$G^{[\lambda]} = \left(\begin{array}{c|c|c|c|c} Z^{\lambda-1}M & Z^{\lambda-2}M & \dots & ZM & M \end{array} \right)$$

Lemme 6.1.2 *Soient S une séquence binaire, $\mathcal{L}$ une matrice de taille $a \times a$ inversible et S' la séquence binaire S multiplié par $\mathcal{L}$, par bloc de taille a. Si S est produite par un (n,k) codeur convolutif alors S' est produite par un codeur de même rendement et S' est produite par un (n,k) codeur si et seulement si $\mathcal{L}$ est diagonale par blocs de taille* $\text{pgcd}(a,n) \times \text{pgcd}(a,n)$

Démonstration : La séquence S peut être vue comme étant produite par un $(\lambda n, \lambda k)$ codeur convolutif correspondant au regroupement (λ fois) du codeur (n,k).

Il est alors possible d'écrire $S(D) = U(D)G^{[\lambda]}$ avec $U(D) = \sum_j u_j D^j$, où u_j est un vecteur binaire de taille λk et $G^{[\lambda]}$ une matrice génératrice de taille $\lambda k \times \lambda n$ du code $\mathscr{C}$ regroupé λ fois.

$S(D) = \sum_j s_j D^j$ où $s_j = (s_{j,1}, \ldots, s_{j,\mu a})$ est le j-ème bloc de taille μa de S.

Soit $S' = s'_{*,1} \ldots s'_{*,\mu a} | s'_{*,1} \ldots s'_{*,\mu a} | \ldots\ldots | s'_{*,1} \ldots s'_{*,\mu a}$ telle que $(s'_{*,1}, \ldots, s'_{*,\mu a}) = (s_{*,1}, \ldots, s_{*,\mu a})\mathcal{L}$

S' peut également être vue comme la multiplication de S par $\mathcal{L}^{[\mu]}$ (par blocs de taille μa) alors

$$\begin{aligned} S(D)\mathcal{L}^{[\mu]} &= \left[U(D)G^{[\lambda]}\right]\mathcal{L}^{[\mu]} \\ &= U(D)\left[G^{[\lambda]}\mathcal{L}^{[\mu]}\right] \end{aligned}$$

D'autre part $S(D)\mathcal{L}^{[\mu]}$ correspond à l'écriture polynomiale de S', notons la $S'(D) = \sum_j s'_j D^j$ où s'_j est le j-ème bloc de taille μa de S'. On a donc $S'(D) = U(D)\left[G^{[\lambda]}\mathcal{L}^{[\mu]}\right]$. S' peut alors être vue comme une séquence codée du code de matrice génératrice $G^{[\lambda]}\mathcal{L}^{[\mu]}$ et de rendement $\frac{\lambda k}{\lambda n} = \frac{k}{n}$
De plus, $G^{[\lambda]}\mathcal{L}^{[\mu]}$ est la matrice génératrice d'un code regroupé si et seulement si il existe une matrice M' de taille $\lambda k \times n$ telle que

$$G^{[\lambda]}\mathcal{L}^{[\mu]} = \left(\; Z^{\lambda-1}M' \;\middle|\; Z^{\lambda-2}M' \;\middle|\; \ldots \;\middle|\; ZM' \;\middle|\; M' \; \right)$$

Donc si et seulement si $\mathcal{L}^{[\mu]}$ est de la forme

$$\begin{pmatrix} A_{n\times n} & & & \\ & A_{n\times n} & & \\ & & \ddots & \\ & & & A_{n\times n} \end{pmatrix}$$

De plus $\mathcal{L}^{[\mu]}$ est inversible donc $A_{n\times n}$ est inversible et si $\mathcal{L}^{[\mu]}$ est une matrice de permutation alors $A_{n\times n}$ est une matrice de permutation
Or $\mathcal{L}^{[\mu]}$ est composée de blocs de taille $a \times a$ donc elle est formée de blocs identiques de taille $pgcd(a,n) \times pgcd(a,n)$.

□

Nous supposons à présent que nous disposons d'assez de données pour permettre la reconstruction des codes rencontrés. Notons $R(S_f) = \begin{cases} \emptyset \\ (n,k,G(D),p) \text{ sinon} \end{cases}$ où n, k, $G(D)$, p sont respectivement

la longueur, la dimension, une matrice génératrice du code trouvé et p le taux d'erreur du canal. S_f est la séquence obtenue en utilisant l'étiquetage f.

Propriété 6.1.3 *Soient f, f' deux mappings tels que $f' \in C_L(f)$, alors $R(S_f) = \emptyset$ si et seulement si $R(S_{f'}) = \emptyset$*

Cette propriété découle directement du lemme 6.1.2, c'est elle qui permet le fonctionnement de notre algorithme. Nous utilisons en effet cette relation pour effectuer un seul test par classe linéaire. De plus le nombre de mappings conservant la longueur de code dans une classe linéaire correspond au nombre de matrice inversible de taille $pgcd(a,n) \times pgcd(a,n)$. Donc le nombre de solutions retournées par notre algorithme (défini page 87) sera à minima le cardinal de $GL(pgcd(a,n), \mathbb{F}_2)$ dans le cas non bruité et $pgcd(a,n)!$ dans le cas bruité

Exemple de reconnaissance de mapping pour le code $\begin{pmatrix}\mathbf{43} & \mathbf{61}\end{pmatrix}$ Pour $a = 2$ nous avons 4 classes linéaires et 6 mappings par classes. En effectuant dans un premier temps une reconnaissance de codes par classe linéaire nous obtenons dans le cas non bruité le résultat suivant : deux classes linéaires fournissent un taux d'erreur de 0 tandis que 2 autres fournissent un taux d'erreur proche de 0.15. Nous reviendrons un peu plus tard sur ce phénomène. Nous explorons alors les classes linéaires qui donnent un taux d'erreur de 0. Dans chacune de ces deux classes les résultats obtenus sont identiques. La liste des codes obtenus est la suivante :

$$\begin{pmatrix}22 & 43\end{pmatrix} \quad \begin{pmatrix}43 & 22\end{pmatrix} \quad \begin{pmatrix}43 & 61\end{pmatrix} \quad \begin{pmatrix}22 & 61\end{pmatrix} \quad \begin{pmatrix}61 & 22\end{pmatrix} \quad \begin{pmatrix}61 & 43\end{pmatrix}$$

Ces codes sont des transformations du code $\begin{pmatrix}43 & 61\end{pmatrix}$. Les mappings correspondant (au nombre de 12) sont donc indistinguables si l'on ne connaît pas le code utilisé initialement. Lorsque l'on se place dans le cas bruitée ($p = 5.10^{-3}$) nous sélectionnons de nouveau deux classes linéaires. Ces deux classes donnent comme estimation du taux d'erreur $p = 0.005120$. En explorant ces classes nous obtenons la même liste de codes mais seules les codes $\begin{pmatrix}61 & 43\end{pmatrix}$ et $\begin{pmatrix}43 & 61\end{pmatrix}$ minimisent le taux d'erreur ($p = 0.005120$ pour ces codes contre $p = 0.007840$ et $p = 0.007510$ pour les autres). En effet, la transformation qui relie ces deux codes est une simple permutation des sorties du codeur. Dans ce cas de figure seuls deux mappings sont indistinguables dans une classe linéaire soit 4 au total.

Nous utilisons à présent la propriété de stabilité sur les classes affines également. Nous l'effectuons toutefois sur une séquence binaire modifiée.

6.1.2 Équivalence affine

Cette propriété étant fausse sur les classes affines, nous modifions le test appliqué pour obtenir ce même résultat de stabilité. Les classes affines étant plus grandes que les classes linéaires, cela nous permet d'accélérer la recherche.

Rappelons que deux mappings f et f' de $\mathcal{C}$ dans $\mathbb{F}_2^a$ sont dit *affinement équivalents* si et seulement si il existe $\mathcal{L} \in GL(a, \mathbb{F}_2)$ et $v \in \mathbb{F}_2^a$ tels que $f' = f.\mathcal{L} + v$ et on note $f' \in C_A(f)$.

Pour $\delta > 0$ un entier et S une séquence binaire. Notons $S << \delta$ la séquence S shiftée (avec perte) de δ positions et $D_\delta(S) = S - (S << \delta)$. L'opération $D_\delta(S) = S - (S << \delta)$ nous ramène au cas de l'équivalence linéaire, nous obtenons alors le lemme suivant.

Lemme 6.1.4 *Soient S une séquence binaire, $\mathcal{L}$ une matrice binaire inversible de taille $a \times a$, $v \in \mathbb{F}_2^a$ et S' la séquence binaire obtenue en appliquant à S la transformation $x \mapsto x\mathcal{L}+v$ (par bloc de taille a). Si S est produite par un (n,k) codeur convolutif alors pour tout δ multiple de* $\mathrm{ppcm}(a,n)$*, la séquence $D_\delta(S')$ est produite par un codeur de même rendement.*

Démonstration : $S' = s'_{*,1} \dots s'_{*,a} \dots\dots s'_{*,1} \dots s'_{*,a}$ avec $(s'_{*,1}, \dots, s'_{*,a}) = (s_{*,1}, \dots, s_{*,a})\mathcal{L} + v$

Lorsque δ est un multiple de a la séquence $D_\delta(S') = S' - (S' << \delta)$ est la multiplication de $D_\delta(S)$ par $\mathcal{L}$, par bloc de taille a. La partie affine est en effet supprimée par la différence.

Lorsque δ est un multiple de n la séquence $D_\delta(S) = S - (S << \delta)$ est une séquence codée par le codeur (n,k) (par linéarité du code). D'après le lemme 6.1.2, si δ est un multiple de $ppcm(a,n)$, la séquence $D_\delta(S')$ est produite par un codeur de rendement k/n.

□

La propriété suivante découle alors directement du lemme 6.1.4.

Propriété 6.1.5 *Soient f et f' deux mappings de $\mathcal{C}$ dans $\mathbb{F}_2^a$, tels que $f' \in C_A(f)$.*
Alors $R(D_\delta(S_f)) = \emptyset$ si et seulement si $R(D_\delta(S'_f)) = \emptyset$

L'inconvénient d'une telle méthode est que nous pouvons augmenter le taux d'erreur de la séquence testée. Nous supprimons également une petite partie des données (2δ bits). Mais nous pouvons à présent tester un seul étiquetage par classe affine. Nous disposons alors de toutes les propriétés nécessaires à la mise en oeuvre de notre algorithme.

Nous avons précédemment évoqué le fait que la solution retournée par cet algorithme n'est pas unique. En effet il existe au moins *pgcd(a,n)*! mappings équivalents. Ils ne peuvent être discriminés sans autre apport d'informations. De plus de multiples classes linéaires peuvent être sélectionnées à l'étape 2 de notre algorithme. C'est ce que nous allons voir à présent.

6.1.3 Nombre de solutions par classe affine

Nous avons vu au Chapitre 5 que nous sommes en mesure de décrire un ensemble de représentants des classes affines pour une taille de modulation donnée. Nous noterons f_A un tel représentant. De plus $\{f_A + v_1, \cdots, f_A + v_{2^a}\}$ est l'ensemble de représentants des classes linéaires que nous choisissons (les v_i parcourent $\mathbb{F}_2^a$).

Soient v un vecteur binaire de longueur a et S_v la séquence binaire contenant les bits de v (répétés un certain nombre de fois jusqu'à obtenir une séquence de même longueur que la séquence que la séquence observée). Si S_v est une séquence codée de $\mathscr{C}$ alors $S_{f_0} + S_v = S_{f_0+v}$ est une séquence codée de $\mathscr{C}$. Alors la reconnaissance de code convolutif pour le mapping $f_0 + v$ fournit comme résultat G_0, n_0, k_0 et p_0 comme probabilité d'erreur. Les deux mappings f_0 et $f_0 + v$ sont alors parfaitement

indistinguables.

Il existe dans ce cas une deuxième classe linéaire fournissant une reconnaissance de code optimale. Ceci devrait impliquer l'exploration de deux classes linéaires mais nous montrons ici que l'ensemble des codes obtenus dans $C_L(f_0)$ correspond à l'ensemble des codes obtenus pour $C_L(f_0 + v)$.

Propriété 6.1.6 *Soit v tel que S_v est une séquence codée de $\mathscr{C}$ et $\mathcal{L}$ une matrice inversible de taille $a \times a$ alors $S_{(f_0+v)\mathcal{L}}$ est une séquence codée de $C^{[\lambda]}\mathcal{L}^{[\mu]}$ avec $p \geq p_0$ comme probabilité d'erreur binaire.*

Démonstration : Nous avons vu dans la preuve du lemme 6.1.2 que pour tout mapping f' tel que $f' = f_0.\mathcal{L}$ alors $S_{f'}$ est une séquence codée par $G_0^{[\lambda]}\mathcal{L}^{[\mu]}$.
De plus si S_v est une séquence codée de $\mathscr{C}$ alors $S_{v\mathcal{L}}$ est une séquence codée par $G_0^{[\lambda]}\mathcal{L}^{[\mu]}$.
Donc $S_{f'} + S_{v\mathcal{L}} = S_{(f_0+v)\mathcal{L}}$ est une séquence codée par $G_0^{[\lambda]}\mathcal{L}^{[\mu]}$.

D'autre part, lorsque $\mathcal{L}$ est un matrice de permutation, $S_{(f_0+v)\mathcal{L}}$ est telle que la probabilité d'erreur binaire associée est p_0 (on a simplement permuté les sorties du codeur). Lorsque $\mathcal{L}$ n'est pas une matrice de permutation on peut potentiellement augmenter le taux d'erreur. □

Cette propriété permet donc d'explorer une seule classe linéaire parmi celles qui sont optimales (fournissant p_0 comme probabilité d'erreur) et d'étendre les résultats obtenus.

Lorsque $n|a$ nous pouvons de plus donner le nombre de telles classes linéaires. En effet, soit $\lambda n = a$, toute séquence d'information périodique de période λk (chaque entrée du codeur est de période λ) engendre une séquence codée de période $\lambda n = a$. Or il existe $2^{\lambda k}$ séquences d'informations de période λk donc il existe $2^{\lambda k}$ séquences S_v correspondant à une séquence codée et finalement $2^{\lambda k}$ classes linéaires optimales. En revanche il est difficile d'estimer ce nombre de classes linéaires dans d'autres cas.

Prenons le code $\begin{pmatrix} 43 & 61 \end{pmatrix}$, pour $a = 2$ nous obtenons deux classes linéaires optimales car la séquence $01 \cdots 01$ est une séquence codée, pour $a = 3$ nous avons quatre classes linéaires correspondant à la bonne classe linéaire et à celles des séquences $011 \cdots 011$, $101 \cdots 101$ et $110 \cdots 100$. Le code $\begin{pmatrix} 11 & 13 & 15 \end{pmatrix}$ lui donne une seule classe linéaire pour $a = 3$ et deux pour $a = 2$ (la séquence $110 \cdots 110$ est une séquence codée).

6.1.4 Unicité de la classe affine

Beaucoup de classes affines fournissent un résultat pour la reconnaissance de codes convolutifs, mais nos expériences ont montré que celle fournissant le meilleur rendement de code est la bonne classe affine et qu'elle est unique. Pour $a = 2$ il existe une seule classe affine donc il est inutile de faire une sélection dans ce cas. Lorsque $a = 3$ il existe alors 30 classes affines, la sélection de la bonne classe affine a donc une importance pour limiter le nombre de tests à effectuer. Pour les codes de rendement $k/k + 1$, seule une classe affine fournit une reconnaissance de code positive donc la sélection est faite dans ce cas. Lorsque le code est de rendement différent on observe sur certaines classes affines des codes de dimension plus élevées que celle du code d'origine. En effet il semble qu'en effectuant un test sur une mauvaise classe affine nous perdions des équations de parité. Par exemple le code décimal $\begin{pmatrix} 43 & 61 \end{pmatrix}$ donne comme résultat, pour $a = 3$ et dans le cas non bruité, une classe ayant pour rendement

1/2 (nous avons trouvé un code de longueur 6 et de dimension 3) et sept classes de rendement 5/6 (nous avons trouvé des codes de longueur 6 et de dimension 5).

Dès lors que nous effectuons une reconnaissance de codes dans une classe affine qui n'est pas la bonne nous risquons de perdre des équations de parité. Nous recherchons donc la classe affine qui possède le plus d'équations de parité et donc le plus petit rendement. Il semble de plus que cette classe affine soit unique.

Nous pouvons à présent décrire le fonctionnement de notre algorithme de reconnaissance avec les critères de sélection des classes affines, linéaires puis des mappings en eux-mêmes.

6.2 Algorithme

Cet algorithme procède en 3 étapes :

- La première étape est exécutée pour chaque élément f_A d'un ensemble de représentants F des classes affines. Une reconnaissance de code convolutif est effectué sur S_{f_A}. Nous sélectionnons les classes affines fournissant un rendement k/n minimal. S'il en existe plusieurs on choisira toujours celles ayant le plus petit taux d'erreur.

- La seconde étape effectue une reconnaissance de code pour les mappings $f_A + v$, $v \in \mathbb{F}_2^a$ dans une classe affine $C_A(f_A)$ sélectionnée à l'étape précédente. Nous retenons les classes linéaires fournissant un taux d'erreur p minimal parmi les taux d'erreur trouvés à cette étape.

- La troisième étape permet de tester tous les mappings d'une classe linéaire par classe affine (parmi celle retenus jusqu'ici). Nous retenons les mappings correspondant à la plus petite longueur de code. Parmi ceux là nous conservons uniquement ceux de plus faible taux d'erreur s'il y a lieu. Nous transposons les résultats de la classe linéaire explorée aux autres classes linéaires optimales.

Notons que δ, le décalage effectué afin d'éliminer la partie affine du mapping, doit être un multiple de a et de la longueur n_0 du code initialement utilisé. Or nous ne connaissons pas n_0. Nous choisissons alors δ de manière à ce qu'il soit multiple de tous les n_0 possibles (typiquement n_0 est compris entre 2 et 8 pour un code convolutif). Il est cependant possible de tester le décalage pour plusieurs valeurs δ en prenant garde à ne pas trop augmenter la complexité de l'algorithme.

Nous avons testé notre algorithme avec des codes de différents rendements et différentes longueurs de contraintes pour $a = 2$ et 3. La TABLE 6.2 suivante rappelle le nombre de classes affines et linéaires pour ces tailles de constellations. Le nombre de reconnaissance de codes effectuées est donc de 11 (contre 24 auparavant) pour $a = 2$ et de 206 (contre 40320 de manière exhaustive) si l'unicité de la classe affine de plus petit rendement est avérée.

Lorsque $a \geq 4$ le nombre de classes affines à parcourir, c'est-à-dire le nombre de tests à effectuer devient très important et il est difficile (pour $a = 4$) voire inenvisageable d'appliquer notre algorithme (pour $a > 4$). Nous réduisons alors notre espace de recherche aux mappings Gray.

a	2	3
Nombre de mappings	24	40320
Nombre de classes affines	1	30
Nombre de classes linéaires par classe affine	4	8
Nombre de mappings par classes linéaires	6	168

TABLE 6.2 – Répartition linéaire et affine pour $a = 2$ et 3

La TABLE 6.3 donne le nombre de mappings pour $a = 4$ dans les cas non Gray et Gray ainsi que leur répartition dans les classes affines et linéaires. Il existe une seule classe affine contenant des mappings Gray dans ce cas. Le nombre de classes linéaires reste inchangé et le nombre de mappings Gray par classe linéaire correspond au nombre de matrices de permutation de taille $a \times a$. En effet seules ces matrices conservent le critère de Gray. Ceci a été démontré dans le Chapitre 5.

a	4	a	4
Nombre de mappings	$> 2^{44}$	Nombre de mappings Gray	384
Nombre de classes affines	64 864 800	Nombre de classes affines contenant des mappings Gray	1
Nombre de classes linéaires par classe affine	16	Nombre de classes linéaires par classe affine	16
Nombre de mappings par classes linéaires	20160	Nombre de mappings Gray par classes linéaires	24

TABLE 6.3 – Répartition linéaire et affine pour $a = 4$

La section suivante explicite le fonctionnement de l'algorithme de reconnaissance de mappings appliqués aux mappings Gray.

6.3 Applications aux mappings Gray

L'algorithme de reconstruction de mappings appliqués aux mappings Gray est très semblable au cas général. La seule différence consiste à explorer uniquement les mappings Gray et donc les classes affines contenant des mappings Gray. Nous avons pour cela dénombré les mappings Gray dans le Chapitre 5 pour chaque taille de constellations de $a = 4$ à $a = 8$, ainsi que la répartition de ces mappings dans les classes affines et linéaires. Rappelons que le critère de Gray est stable par addition d'un motif constant à l'ensemble des symboles d'un mapping. Nous savons de plus générer un ensemble de représentants pour les classes affines grâce aux travaux de *Wesel et al.* Donc il nous est possible de générer pour chaque classe affine concernée un ensemble de représentants des classes linéaires par simple addition d'un motif constant. Il est possible de décrire un ensemble de représentants des classes affines pour les mappings quasi-Gray également en fixant des méthodes de construction de ces mappings. Notre algorithme s'étend alors à la reconstruction de mappings pour les grandes tailles de constellation.

La répartition des mappings Gray est rappelée ci-dessous ainsi que celle des mappings quasi-Gray (construits selon deux méthodes distinctes à partir de mappings rectangles Gray).

a	4	6	8
Nombre de classes affines contenant des mappings Gray	1	9	56 644
Nombre de classes linéaires par classe affine	16	64	256
Nombre de mappings Gray par classe linéaire	24	720	40 320

TABLE 6.4 – Distribution des mappings Gray pour $a = 4,\ 6,\ 8$

a	5	7
Nombre de classes affines contenant des mappings quasi-Gray	2	2
Nombre de classes linéaires par classe affine	32	128
Nombre de mappings quasi-Gray par classe linéaire	120	5040

TABLE 6.5 – Distribution des mappings Gray pour $a = 5,\ 7$

Les constellations $4 - PSK$ et $16 - QAM$ sont représentées par une seule classe affine, il est donc inutile de vérifier le rendement dans ce cas, il suffit de constater la présence d'un code convolutif.

Le nombre de reconnaissance de codes est alors de 41 pour $a = 4$ puis 793 pour $a = 6$ et enfin 97220 pour $a = 8$ dans l'hypothèse où l'on a bien unicité de la classe affine. Nous rappelons que dans le cas où deux classes affines auraient même rendement nous choisissons celle donnant le plus petit taux d'erreur. Ceci peut en effet se produire lorsque que l'on se place en présence d'erreur, nous perdons alors des équations de parité également sur la bonne classe affine. Nous réduisons ainsi considérablement le nombre de tests à effectuer. La section suivante donne quelques résultats obtenus.

6.4 Résultats obtenus

Les codes convolutifs ayant servis à effectuer les tests sont données à la page suivante.

Les tests de reconstruction de mappings ont été appliqués pour différentes tailles de constellations, principalement, $a = 2$, 3, 4 et 6. Tous les résultats obtenus dans le cas sans erreur sont concluants et confirment l'unicité de la classe affine de plus petit rendement ainsi que le nombre de mappings solutions dans une classe linéaire. Lorsque nous plaçons dans le cas avec erreur les résultats sont inégaux selon les codes. Nous voyons ici un exemple d'exécution pour $a = 6$ et le code $\begin{pmatrix} 43 & 61 \end{pmatrix}$ vu précédemment. Pour des taux d'erreur allant jusqu'à 3.10^{-3} une classe affine (sur 9), huit classes linéaires (sur 64) et enfin deux mappings dans chaque classe linéaire sont sélectionnées. Donc nous obtenons 16 mappings possibles. A partir de 4.10^{-3} seize classes linéaires sont retenues, nous explorons alors une classe linéaire ne correspondant pas à une séquence codée et les codes retournés par notre algorithme sont des codes de longueur 6 et de dimension 4. Or les transformations préservant la longueur du code sont les matrices de permutation de taille 6×6. Pour $a = 6$ une classe linéaire est justement composée de 720 mappings correspondants à 720 matrices de permutations de taille 6×6. Donc le nombre de mappings possibles est de $720.16 = 11\,520$. Nous perdons donc beaucoup de précision.

Globalement les tests que nous avons effectués montrent que cet algorithme fonctionne bien pour de faibles taux d'erreur jusqu'à 5.10^{-3}. Au delà de ce taux d'erreur nous commençons à observer des résultats ne comportant pas le bon mapping sur certains codes. Les résultats dépendent en effet

Dénomination	Code sous forme décimale
2a	$\begin{pmatrix} 43 & 61 \end{pmatrix}$
2b	$\begin{pmatrix} 91 & 121 \end{pmatrix}$
2c	$\begin{pmatrix} 369 & 491 \end{pmatrix}$
2d	$\begin{pmatrix} 1245 & 1969 \end{pmatrix}$
3a	$\begin{pmatrix} 11 & 13 & 15 \end{pmatrix}$
3b	$\begin{pmatrix} 39 & 43 & 61 \end{pmatrix}$
3c	$\begin{pmatrix} 149 & 217 & 247 \end{pmatrix}$
3d	$\begin{pmatrix} 1259 & 1465 & 1661 \end{pmatrix}$
32a	$\begin{pmatrix} 11 & 6 & 11 \\ 6 & 11 & 15 \end{pmatrix}$
32b	$\begin{pmatrix} 3 & 6 & 13 \\ 28 & 25 & 15 \end{pmatrix}$
32c	$\begin{pmatrix} 21 & 24 & 15 \\ 40 & 7 & 53 \end{pmatrix}$
32d	$\begin{pmatrix} 51 & 44 & 25 \\ 22 & 43 & 35 \end{pmatrix}$

Dénomination	Code sous forme décimale
4a	$\begin{pmatrix} 43 & 55 & 57 & 61 \end{pmatrix}$
4b	$\begin{pmatrix} 93 & 93 & 103 & 115 \end{pmatrix}$
4c	$\begin{pmatrix} 157 & 189 & 203 & 239 \end{pmatrix}$
4d	$\begin{pmatrix} 307 & 349 & 475 & 485 \end{pmatrix}$
43a	$\begin{pmatrix} 7 & 6 & 2 & 1 \\ 12 & 5 & 0 & 13 \\ 8 & 4 & 15 & 1 \end{pmatrix}$
43b	$\begin{pmatrix} 1 & 12 & 14 & 3 \\ 8 & 11 & 2 & 7 \\ 14 & 0 & 3 & 11 \end{pmatrix}$

par définition des résultats obtenus pour la recherche de code et si une erreur se produit lors de la reconstruction d'un code cela provoque des erreurs pour notre méthode également.

D'autre part la sélection des classes linéaires s'effectue en fonction du taux d'erreur observés pour chacune de ces classes. Lorsque le taux d'erreur atteint 5.10^{-3} un certain nombre de tests échouent. En effet on fait la différence entre les classes linéaires à sélectionner sur la base de l'appartenance de la séquence S_v à l'ensemble des séquences codées. Les autres séquences correspondent à des séquences codées mais bruitées. Lorsque le taux d'erreur augmente la distinction entre les deux n'est pas clair.

Le code $3a$ est un exemple de codes pour lequel nous avons eu des difficultés à appliquer la reconstruction du mapping. Lors de la sélection des classes affines pour $a = 3$ un code $(6, 4)$ nous était retournés là où nous aurions du retrouver un code de rendement $1/3$. Le code reconstruit était un code pour lequel toutes les équations de parité n'ont pas été trouvées, provoquant alors une erreur de sélection de la classe affine et ce même dans le cas sans erreur. L'utilisation d'un seuil de détection a permis de résoudre le problème dans le cas sans erreur. Ce seuil définit le nombre de minimum de mots du dual nécessaires à la reconstruction d'un code forçant ainsi le testeur à rechercher plus d'équations. Ce seuil doit ensuite être diminué lorsque le taux d'erreur augmente.

6.5 Application au cas des codes Reed-Solomon

Nous avons vu qu'il existe une méthode efficace de reconstruction des codes de Reed-Solomon, cependant elle fonctionne même lorsque les coordonnées du code ont été permutées. Il s'avère que pour la reconnaissance de mapping ce n'est pas à notre avantage. En effet il faut pouvoir distinguer dans une classe linéaire si certains mappings correspondent à des séquences codées par un code de Reed-Solomon et d'autres non.

La procédure de reconnaissance de code que nous utilisons alors s'applique uniquement dans le cas non bruitée. En effet nous formons une matrice contenant sur chaque ligne un mot de code (sous forme binaire) et effectuons un pivot de Gauss. Cette opération permet alors d'observer un défaut de rang et d'obtenir une base du dual. De plus lorsque la matrice du dual est sous forme systématique elle permet directement de donner une matrice génératrice binaire du code. En itérant sur les différentes longueurs possibles on obtient alors la longueur et la dimension du code vu comme code binaire.

Or la matrice génératrice d'un code de Reed-Solomon dispose d'une forme cyclique sur $\mathbb{F}_q^m$. Elle est formée des coefficients du polynôme générateur :

$$\begin{pmatrix} g_0 & g_1 & \cdots & g_r & & & \\ & g_0 & g_1 & \cdots & g_r & & \\ & & \ddots & \ddots & & \ddots & \\ & & & g_0 & g_1 & \cdots & g_r \end{pmatrix}$$

Chaque coefficients g_i est représenté dans une base de dimension le degré de l'extension. Donc la matrice binaire précédemment obtenue peut être mise sous forme cyclique pour le bon degré d'extension. En itérant sur les degrés d'extension possibles, cette méthode permet de retrouver également les

coefficients du polynôme générateur.

Par ailleurs les racines de ce polynôme sont consécutives par construction. Donc un code de Reed-Solomon est utilisé si et seulement nous pouvons vérifier que $g(x)$ dispose de racines consécutives. Pour cela on recherche les racines de $g(x)$ dans l'ensemble des racines possibles des polynômes primitifs de degré le degré de l'extension.

Cette méthode est applicable uniquement dans le cas non bruité puisque nous utilisons une élimination de Gauss pour la reconstruction du dual. Donc les résultats que nous donnons sont limités mais permettent de valider notre méthode pour de tels codes.

Nous effectuons des tests distincts selon le niveau de précision voulue. Pour la sélection des classes affines nous regardons simplement le rang de la matrice contenant les mots de code après l'élimination de Gauss. Ceci nous donne alors la redondance introduite par le code. Nous effectuons ce test sur de données décalés puis ajoutées à elle-même comme pour l'algorithme défini sur les codes convolutifs. Nous supposons alors que nous connaissons la longueur du code pour effectuer le bon décalage. En effet pour des codes en blocs il nous est impossible, étant données les longueurs de codes, de trouver un décalage qui soit multiple de toutes les longueurs possibles. Cependant l'élimination de Gauss est appliquée pour chaque longueur, il est alors possible d'inclure le décalage à chaque itération. Nous sélectionnons alors la classe affine fournissant la plus grande redondance. Ceci s'apparente au choix de la classe affine de plus faible rendement que nous effectuions pour les codes convolutifs. Nous effectuons de nouveau ce test sur les représentants des classes linéaires. Pour ces deux premières étapes nous ne cherchons pas de structure de corps puisqu'elle est à priori détruire. Nous appliquons enfin une reconstruction complète du code pour chacun des mappings dans les classes linéaires sélectionnées.

Nous avons effectués des tests de reconstruction de mappings pour des données initialement codées par un code de Reed-Solomon, raccourci ou non, et ce dans le cas sans erreur. Ils ont fourni de 1 à 128 mappings au plus pour $a = 2 \dots 7$ et des longueurs de codes, dimensions et degrés d'extensions divers.

Annexe A

Rapport de probabilités de collisions avec une équation de parité

$p = 0.0001$	
u	R
1	1.999600
2	1.999200
3	1.998801
4	1.998401
5	1.998002
6	1.997603
7	1.997204
8	1.996805
9	1.996406
10	1.996008
11	1.995609
12	1.995211
13	1.994813
14	1.994415
15	1.994017
16	1.993620
17	1.993222
18	1.992825
19	1.992428
20	1.992031
21	1.991634
22	1.991238
23	1.990841
24	1.990445
25	1.990049
26	1.989653
27	1.989257
28	1.988861
29	1.988466
30	1.988071

$p = 0.001$	
u	R
1	1.996004
2	1.992024
3	1.988060
4	1.984112
5	1.980179
6	1.976262
7	1.972361
8	1.968476
9	1.964606
10	1.960751
11	1.956912
12	1.953088
13	1.949279
14	1.945486
15	1.941708
16	1.937945
17	1.934197
18	1.930464
19	1.926746
20	1.923042
21	1.919354
22	1.915680
23	1.912021
24	1.908377
25	1.904747
26	1.901131
27	1.897531
28	1.893944
29	1.890372
30	1.886814

$p = 0.005$	
u	R
1	1.980100
2	1.960596
3	1.941480
4	1.922745
5	1.904382
6	1.886385
7	1.868746
8	1.851458
9	1.834514
10	1.817907
11	1.801631
12	1.785678
13	1.770043
14	1.754719
15	1.739700
16	1.724980
17	1.710553
18	1.696413
19	1.682555
20	1.668972
21	1.655659
22	1.642612
23	1.629824
24	1.617290
25	1.605006
26	1.592966
27	1.581166
28	1.569601
29	1.558266
30	1.547157

$p = 0.01$	
u	R
1	1.960400
2	1.922368
3	1.885842
4	1.850763
5	1.817073
6	1.784717
7	1.753642
8	1.723798
9	1.695135
10	1.667608
11	1.641171
12	1.615780
13	1.591395
14	1.567976
15	1.545484
16	1.523883
17	1.503137
18	1.483213
19	1.464078
20	1.445700
21	1.428051
22	1.411100
23	1.394820
24	1.379185
25	1.364170
26	1.349749
27	1.335899
28	1.322597
29	1.309822
30	1.297553

$p = 0.05$	
u	R
1	1.810000
2	1.656100
3	1.531441
4	1.430467
5	1.348678
6	1.282430
7	1.228768
8	1.185302
9	1.150095
10	1.121577
11	1.098477
12	1.079766
13	1.064611
14	1.052335
15	1.042391
16	1.034337
17	1.027813
18	1.022528
19	1.018248
20	1.014781
21	1.011973
22	1.009698
23	1.007855
24	1.006363
25	1.005154
26	1.004175
27	1.003381
28	1.002739
29	1.002219
30	1.001797

$p = 0.1$	
u	R
1	1.640000
2	1.409600
3	1.262144
4	1.167772
5	1.107374
6	1.068719
7	1.043980
8	1.028147
9	1.018014
10	1.011529
11	1.007379
12	1.004722
13	1.003022
14	1.001934
15	1.001238
16	1.000792
17	1.000507
18	1.000325
19	1.000208
20	1.000133
21	1.000085
22	1.000054
23	1.000035
24	1.000022
25	1.000014
26	1.000009
27	1.000006
28	1.000004
29	1.000002
30	1.000002

Table des figures

Liste des tableaux

Bibliographie

[1] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. National Institute of Standards and Technology Special Publication 800-22 Revision 1a, 2010.

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. on Information Theory*, 20(2) :284–287, 1974.

[3] J. Barbier. *Analyse de canaux de communication dans un contexte non coopératif*. Thèse de doctorat, École Polytechnique, November 2007.

[4] J. Barbier and J. Letessier. Forward Error Correcting Codes Characterization Based on Rank Properties. In *International Conference on Wireless Communications and Signal Processing (WCSP)*, Nanjing, China, November 2009. IEEE.

[5] J. Barbier, G. Sicot, and S. Houcke. Algebraic approach of the reconstruction of linear and convolutional error correcting codes. In *World Academy of Science, Engineering and Technology*, volume 16, pages 66–71, November 2006.

[6] G. Batail. *Théorie de l'information*. 1997.

[7] Marion Bellard and Nicolas Sendrier. Reconstruction of constellation labeling with convolutional coded data. In *Proceedings of 2012 IEEE International Symposium on Information Theory and its Applications ISITA"2012*, pages 653–657, Honolulu, Hawaii, USA, October 2012.

[8] C. Blondeau, B. Gérard, and J.P. Tillich. Accurate estimates of the data complexity and success probability for various cryptanalyses. *Designs Codes and Cryprography*, 1-3 :3–34, 2011. Special Issue on Coding and Cryptography.

[9] P.L. Boyd and C. Robertson. Recovery of Unknown Constraint Length and Generator Polynomials for Linear Convolutional Encoders. In *21th Century Military Communications Conference*, volume 2, pages 947–951, Los Angeles, CA, USA, October 2000. IEEE.

[10] P.L. Boyd and C. Robertson. Recovery of Unknown Constraint Length and Generator Polynomials for Linear Convolutional Encoders in Noise. In *21th Century Military Communications Conference*, volume 2, pages 952–956, Los Angeles, CA, USA, October 2000. IEEE.

[11] M. Burrows and D.J. Wheeler. A Block-sorting Losless Data Compression Algorithm. Research report, Digital, Systems Research Center, California, 1994.

[12] A. Canteaut and F. Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code : Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. 44(1) :367–378, January 1998.

[13] C. Chabot. Recognition of a code in a noisy environment. In *Proc. of the IEEE Int. Symp. Information Theory*, pages 2211–2215, Nice, France, June 2007. IEEE.

[14] C. Chabot. *Reconnaissance de codes, structure des codes quasi-cycliques.* Thèse de doctorat, Université de Limoges, September 2009.

[15] M. Cluzeau. *Reconnaissance d'un schéma de codage.* Thèse de doctorat, École Polytechnique, November 2006.

[16] M. Cluzeau and M. Finiasz. Reconstruction of Punctured Convolutional Codes. In *Information Theory Workshop (ITW)*, Taormina, Italy, October 2009. IEEE.

[17] M. Cluzeau and M. Finiasz. Recovering a Code's Length and Synchronization from a Noisy Intercepted Bitstream. In *Proc. of the IEEE Int. Symp. Information Theory*, pages 2737–2741, Seoul, Korea, 2009. IEEE.

[18] M. Cluzeau and J.P. Tillich. On the Code Reverse Engineering Problem. In *Proc. of the IEEE Int. Symp. Information Theory*, pages 634–638, Toronto, Canada, 2008. IEEE.

[19] Collectif d'auteur sous la direction de Geneviève Baudoin. Modulations Numériques. In *Radiocommunications Numériques*, volume 1-Principes, modélisation et simulation, chapter 5. Dunod, 2002.

[20] M. Côte. *Reconnaissance de codes correcteurs d'erreurs.* Thèse de doctorat, École Polytechnique, March 2010.

[21] M. Côte and N. Sendrier. Reconstruction of convolutional codes from noisy observation. In *Proc. of the IEEE Int. Symp. Information Theory*, pages 546–550, Seoul, Korea, 2009. IEEE.

[22] T.M. Cover and J.A. Thomas. Gaussian Channel. In *Elements of Information Theory*, chapter 9, pages 261–299. Wiley InterScience, 2006.

[23] P. Csillag. In *Introduction aux codes correcteurs d'erreurs.* Ellipses, 1990.

[24] J. Dingel and J. Hagenauer. Parameter estimation of a convolutional encoder from noisy observation. In *Proc. of the IEEE Int. Symp. Information Theory*, pages 1776–1780, Nice, France, June 2007. IEEE.

[25] R.D. Doran. The Gray Code. Research report, University of Auckland, Centre for Discrete Mathematics and Theoritical Computer Science, 2007.

[26] Marianne Durand. *Combinatoire analytique et algorithmique des ensembles de données.* Thèse de doctorat, Ecole doctorale de l'Ecole Polytechnique, 2004.

[27] E. Filiol. *Techniques de reconstruction en cryptologie et théorie des codes.* Thèse de doctorat, École Polytechnique, March 2001.

[28] Eric Filiol. Reconstruction of Convolutional Encoders over $GF(q)$. In *Cryptography and Coding : 6th IMA Int. Conf.*, pages 101–109, 1997.

[29] E.N. Gilbert. Gray Codes and Paths on the n-Cube. *The Bell System Technical Journal*, May 1958.

[30] F. Gray. Pulse code communications. *U.S. Patent 2 632 058*, March 1953.

[31] INRIA, ENSTA, LIX, XLIM. Reconnaissance de codes, Codes et Signatures. Technical report, 2007.

[32] X. Liu and R.D. Wesel. Profile Optimal 8-QAM and 32-QAM Constellations. In *36th Annual Allerton Conference On Communication, Control and Computing*, 1998.

[33] P. Lu and Y. Zou. Fast Computations of Gröbner Bases and Blind Recognitions of Convolutional Codes. In C. Carlet and B. Sunar, editors, *Arithmetic of Finite Fields, First International Workshop, WAIFI 2007*, LNCS, pages 303–317, Madrid, Spain, June 2007. Springer.

[34] M. Marazin. *Reconnaissance en aveugle de codeur à base de code convolutif : Contribution à la mise en oeuvre d'un récepteur intelligent.* Thèse de doctorat, Université de Bretagne Occidentale, December 2009.

[35] M. Marazin, R. Gautier, and G. Burel. Dual code method for blind identification of convolutional encoder for cognitive radio receiver design. In *GLOBECOM Workshops*, Honolulu, USA, 2009. IEEE.

[36] M. Marazin, R. Gautier, and G. Burel. Blind recovery of k/n rate convolutional encoders in a noisy environment. *EURASIP Journal on Wireless Communications and Neworking*, 2011.

[37] M. Marazin, R. Gautier, and G. Burel. Some interesting dual-code properties of convolutional encoder for standards self-recognition. *Institution of Engineering and Technology Communications*, 6(8) :931–935, May 2012.

[38] R.J. McEliece. *The Algebraic Theory of Convolutional Codes*, volume I, chapter 12, pages 1065–1138. North-Holland, 1998.

[39] T.K. Moon. A context for Error Correcting Coding. In *Error correction Coding*, pages 2–52. Wiley InterScience, 2005.

[40] J.G. Proakis. Modulation and Demodulation for the Additive Gaussian Noise Channel. In Alar E.Elken and John M. Morriss, editors, *Digital Communications*, chapter 4, pages 220–361. McGraw-Hill Book Company, 1989.

[41] B. Rice. Determining the parameters of a rate $\frac{1}{n}$ convolutional encoder over $GF(q)$. In *Third International Conference on Finite Fields and Applications*, Glasgow, 1995.

[42] G. Sicot and S. Houcke. Blind detection of interleaver parameters. In *International Conference on Acoustics, Speech, and Signal Processing (ICCASP)*, volume 3, pages 829–832, Philadelphia, USA, March 2005. IEEE.

[43] G. Sicot and S. Houcke. Etude Statistique du seuil dans la detection d'entrelaceur. 2005.

[44] V.M. Sidelnikov and S.O. Shestakov. On insecurity of cryptosystems based on generalized reed-solomon codes. *Discrete Mathematics and Applications*, 2(4) :439–444, 1992.

[45] J.G. Smith. Odd-Bit Quadrature Amplitude-Shift Keying. *IEEE Trans. Commun*, 23 :385–389, March 1975.

[46] G. Ungerboeck. Channel coding with multilevel/phase signals. *IEEE Transactions on Information Theory*, 28(1), 1982.

[47] A. Valembois. *Décodage, Détection et Reconnaissance des Codes Linéaires Binaires.* Thèse de doctorat, Université de Limoges, October 2000.

[48] A. Valembois. Detection and recognition of a binary linear code. *Discrete Applied Mathematics*, 111 :199–218, July 2001.

[49] A.J.H. Vinck, P. Dolezal, and Y.G. Kim. Convolutional Encoder State Estimation. *IEEE Trans. on Information Theory*, 44(4) :1604–1608, July 1998.

[50] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Information Theory*, 13(2) :260–269, 1967.

[51] P.K. Vitthaladevuni and M.S. Alouini. Exact BER Computation for Cross 32-QAM Constellation. In *First International Symposium on Control, Communications and Signal Processing*, pages 643–646. IEEE, 2004.

[52] P.K. Vitthaladevuni, M.S. Alouini, and J.C. Kieffer. Exact BER Computation for Cross QAM Constellations. *IEEE Transactions On Wireless Communications*, 4(6), November 2005.

[53] F. Wang, Z. Huang, and Y. Zhou. A Method for Blind Recognition of Convolution Code Based on Euclidean Algorithm. In *International Conference on Wireless Communications and Mobile Computing*, pages 1414–1417, Shanghai, September 2007. IEEE.

[54] L. F. Wei. Trellis-coded modulation with multidimensional constellations. *IEEE Transactions on Information Theory*, 33(4), Jul 1987.

[55] R.D. Wesel and X. Liu. Edge Profile Optimal Constellation Labeling. In *International Conference on Communications*, volume 3, pages 1198–1202. IEEE, Jun 2000.

[56] R.D. Wesel, X. Liu, J.M. Cioffi, and C. Komminakis. Constellation Labeling for Linear Encoders. *IEEE Transactions On Information Theory*, 47(6) :2417–2431, September 2001.

[57] Kyu-Young Whang, Brad T. Vander-Zanden, and Howard M.Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems*, 15(2) :208–229, June 1990.

[58] Y. Wu, Y. Zhao, and H. Li. Constellation Design for Odd-bit Quadrature Amplitude Modulation. In *Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–4, Sydney, 2010. IEEE.

[59] H. Xie, X.M. Chai, F.H. Wang, and Z.T. Huang. A method for blind identification of rate 1/2 convolutional code based on improved Euclidean algorithm. In *11th International Conference on Signal Processing (ICSP)*, volume 2, pages 1307–1310, Beijing, China, October 2012. IEEE.

[60] Y. Zrelli, R. Gautier, M. Marazin, E. Rannou, and E. Radoi. Focus on Theoretical Properties of Blind Convolutional Codes Identification Methods Based on Rank Criterion. In *9th International Conference on Communications*, pages 353–356, Bucharest, Romania, June 2012. IEEE.

[61] Y. Zrelli, M. Marazin, R. Gautier, and E. Rannou. Blind Identification of Convolutional Encoder Parameters over GF(2^m) in the Noiseless Case. In *20th International Conference on Computer Communications and Neteworks (ICCCN)*, Maui, HI, USA, 2011. IEEE.