HAL
open science

Codes de Gabidulin en caractéristique nulle : application au codage espace-temps

Gwezheneg Robert

▶ To cite this version:

Gwezheneg Robert. Codes de Gabidulin en caractéristique nulle : application au codage espace-temps. Mathématiques générales [math.GM]. Université Rennes 1, 2015. Français. NNT : 2015REN1S083 . tel-01308661v1

HAL Id: tel-01308661
https://theses.hal.science/tel-01308661v1

Submitted on 28 Apr 2016 (v1), last revised 1 Jul 2016 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ANNÉE 2015

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Mathématique et Applications

Ecole doctorale Matisse

présentée par

Gwezheneg Robert

préparée à l'unité de recherche UMR 6625 CNRS–IRMAR

Institut de Recherche Mathématique de Rennes
U.F.R. de Mathématique

Codes de Gabidulin en caractéristique nulle. Application au codage espace-temps.

Thèse soutenue à Rennes le 4 décembre 2015

devant le jury composé de :

Christine BACHOC
Professeur, Université de Bordeaux / rapporteur

Vladimir SIDORENKO
Professeur, Université technique de Munich / rapporteur

Xavier CARUSO
CR CNRS, Université de Rennes 1 / examinateur

Hugues RANDRIAMBOLOLONA
Assimilé MCF, Télécom ParisTech / examinateur

Felix ULMER
Professeur, Université de Rennes 1 / examinateur

Jacques-Arthur WEIL
Professeur, Université de Limoges / examinateur

Pierre LOIDREAU
Assimilé MCF, DGA et Université de Rennes 1 / directeur de thèse

Daniel AUGOT
DR INRIA, INRIA Saclay Île-de-France / co-directeur de thèse

0.1 Historique

Les codes correcteurs ont pour but de résoudre le problème suivant :

Comment transmettre un message de façon fiable,
à travers un canal qui ne l'est pas ?

Il s'agit d'un domaine récent dans l'histoire mathématique. Le premier résultat est publié par C. Shannon en 1948 [Sha48]. Il y modélise la transmission d'information à travers un canal bruité et définit la capacité d'un canal, c'est-à-dire la quantité maximale d'information qu'il peut transmettre. Le théorème de Shannon affirme que, tant que la quantité d'information envoyée dans le canal ne dépasse pas sa capacité, alors il est possible d'atteindre une probabilité d'erreur aussi faible que l'on veut. Le résultat est existentiel, et ne permet pas la construction d'un tel code. Le code de Hamming [Ham50] fera son apparition peu après. Il s'agit du premier exemple non trivial.

Une dizaine d'années plus tard, R. Singleton présente une borne sur les performances des codes [Sin64]. Ces derniers ne peuvent pas permettre à la fois de transmettre beaucoup d'information et de corriger beaucoup d'erreurs. Les codes qui atteignent cette borne sont donc d'un intérêt particulier.

Les codes de Reed-Solomon [RS60] font partie de ces codes. Disposant en plus d'algorithmes de décodage rapides (par exemple [WB86], [Gao03]), ils sont utilisés dans de nombreux domaines (lecteurs CD, ADSL, ...)

En 1978, P. Delsarte [Del78] propose une nouvelle métrique, la métrique rang, reprise par E. M. Gabidulin [Gab85] en 1985. Ce dernier propose une famille de codes dans cette métrique, ainsi qu'un algorithme de décodage. Ces codes, les codes de Gabidulin, sont l'équivalent des codes de Reed-Solomon en métrique rang. Les polynômes sont remplacés par des q-polynômes [Ore33a], un cas particulier de polynômes tordus [Ore33b]. Ainsi, de nombreux algorithmes valables pour les codes de Reed-Solomon ont pu être adaptés pour le décodage des codes de Gabidulin. Par exemple, l'algorithme de L. Welch et E. Berlekamp [BW86] a été adapté par P. Loidreau [Loi06].

Les codes en métrique rang ont bénéficié d'un regain d'intérêt avec l'arrivée du *network coding* ([KM03], [HL08]). Il s'agit d'un protocole décrivant les transmissions de paquets à travers un réseau informatique. Dans ce genre de transmission, la métrique utilisée est la métrique sous-espace [KK08]. Le *lifting* de codes [SKK08] permet de construire de tels codes à partir de codes en métrique rang.

Dans un tout autre registre, près d'un siècle après le premier télégraphe sans fil, apparait l'idée d'utiliser plusieurs antennes pour émettre et recevoir. Il s'agit du système MIMO, utilisé par exemple dans les communications Wi-Fi. En 1999, E. Telatar intègre ces systèmes à la théorie de l'information [T$^+$99].

Un peu avant sont apparus les premiers codes correcteurs adaptés à ce type de transmission [Ala98], [TSC98a]. Il sont appelés codes espace-temps, car la transmission est diversifiée dans l'espace (les antennes sont décalées dans l'espace) et dans le temps (les transmissions sont étalées dans le temps).

Une nouvelle construction est proposée [LK05], permettant la construction de codes espace-temps à partir de codes en métrique rang, les codes de Gabidulin étant particulièrement bien adaptés.

0.2 Description des travaux

Le but de cette thèse est d'étudier la généralisation de codes de Gabidulin, et de concevoir une nouvelle famille de codes espace-temps. Ce document est structuré de la façon suivante.

Le chapitre d'introduction a pour put de présenter les problématiques de la théorie des codes. Nous commencerons par présenter le théorème de Shannon, que nous illustrerons à travers le canal binaire symétrique et le code de répétition. Nous présenterons ensuite les codes en blocs, notamment à travers l'exemple du code de Hamming. Nous donnerons également les résultats généraux sur les codes en métrique de Hamming. Enfin, nous présenterons la façon dont les données numériques sont transmises par ondes, ainsi qu'une modélisation du canal MIMO, pour lequel sont conçus les codes espace-temps.

Le chapitre 2 a pour but de présenter les notions préliminaires à la définition des codes de Gabidulin. Nous nous plaçons dans le cadre d'une extension de corps $K \hookrightarrow L$ de degré m, munie d'un automorphisme $\theta \in \mathrm{Aut}_K(L)$. Au cours de ce chapitre, nous allons voir quelles conditions doivent être vérifiées.

Les codes de Gabidulin, définis sur les corps finis, sont basés sur l'évaluation des q-polynômes. Les q-polynômes étant propres aux corps finis, nous devons également les généraliser : nous obtenons ainsi les θ-polynômes. Après avoir défini les racines de ces polynômes, nous verrons qu'elles forment un espace vectoriel. Afin de garantir les propriétés des codes de Gabidulin, l'espace des racines d'un θ-polynôme doit être de dimension inférieure à son degré. Ce résultat était connu pour les q-polynômes, mais la preuve ne s'adaptait pas à des corps infinis. Nous montrerons que pour cela, le polynôme caractéristique de θ doit être sans facteur carré. Dans ce cas, nous disposons de θ-polynômes annulateurs et interpolateurs ayant les mêmes propriétés que les polynômes classiques, notamment en termes de degré.

Les codes de Gabidulin sont des codes en métrique rang. Nous donnerons la définition de cette métrique, naturellement valable sur des corps infinis. Nous définirons également une nouvelle métrique, basée sur les θ-polynômes. Nous montrerons qu'elles sont équivalentes pourvu que le sous-corps de L stable par θ soit exactement le corps de base K. Pour les besoins de la preuve, nous introduirons également deux autres métriques, qui seront elles aussi équivalentes à la métrique rang.

Nous disposons maintenant de tous les ingrédients pour construire les codes de Gabidulin généralisés. Il ne reste plus qu'à prendre en compte les deux restrictions que nous avons vues afin de définir notre cadre de travail. Nous montrerons que les deux restrictions sont en fait équivalentes, et qu'elles peuvent être formulées de façon plus simple. L'extension $K \hookrightarrow L$ doit être cyclique, et l'automorphisme θ doit être un générateur de $\mathrm{Aut}_K(L)$.

Dans le chapitre 3, nous généraliserons les codes de Gabidulin à des extensions cycliques de degré fini. Après avoir donné leur définition, nous montrerons leurs principales propriétés. Ainsi, ce sont des codes de distance minimale $n-k+1$, ce qui en fait des codes MRD. Il en est de même pour leur dual. Nous verrons également comment modifier le support du code. Enfin, nous définirons comment passer de la version vectorielle à la version matricielle du code à l'aide de la bijection $\mathrm{ext}_{\mathcal{B}}$. Les résultats sont les mêmes que pour les codes de Gabidulin originaux, mais nous fournissons des preuves nouvelles, ne se basant pas sur la finitude du corps L.

Nous aborderons ensuite le premier modèle de canal de transmission. Dans celui-ci, seules des erreurs peuvent se produire. Nous verrons qu'il est alors possible de retrouver les symboles d'information si le poids t de l'erreur vérifie $t \leqslant \lfloor \frac{n-k}{2} \rfloor$. Nous verrons également que le décodage

d'un code de Gabidulin généralisé est équivalent au problème de reconstruction non linéaire, celui-ci pouvant être ramené au problème de reconstruction linéaire. Ces résultats sont la généralisation de [Loi06] aux corps infinis.

Nous dépasserons le cadre de cet article en abordant deux autres modèles de canaux de transmission. En plus des erreurs vues dans le premier modèle, des effacements peuvent apparaitre. Les deux modèles diffèrent par la définition de ces effacements, mais l'un étant une généralisation de l'autre, le même terme *effacement* est utilisé pour les deux modèles. Nous montrerons que dans les deux cas, il est possible de se ramener à un code de Gabidulin de paramètres différents et donc de retrouver les symboles d'information si le nombre t d'erreurs et le nombre s d'effacements [1] vérifient $s + 2t \leqslant n - k$.

Le chapitre 4 est consacré à un algorithme permettant la résolution du problème de Reconstruction. En effet, nous avons vu dans le chapitre précédent que le décodage, dans les trois modèles considérés, se ramenait finalement à ce problème. Après une présentation sommaire, nous prouverons en détail que la sortie de l'algorithme est bien une solution du problème de reconstruction.

Nous ferons ensuite une analyse de la complexité de l'algorithme, afin de montrer qu'elle est quadratique en la longueur du code, à l'exception de quelques éventuels calculs supplémentaires, que nous traiterons à part. Nous donnerons également la complexité des algorithmes de décodage pour les trois modèles étudiés.

Enfin, nous étudierons plusieurs variantes permettant l'amélioration de la complexité de l'algorithme. La première a pour but d'éviter toutes les divisions au cours de l'algorithme. Cela implique d'augmenter le nombre de multiplications. La seconde amélioration consiste à calculer avec des polynômes de plus bas degré, en remarquant une factorisation des polynômes utilisés dans la version de base. Cela permet de diminuer la complexité de l'algorithme. Enfin, la dernière modification concerne le calcul des défauts. Cette variante augmente légèrement la complexité, mais permet de se débarrasser des calculs supplémentaires évoqués dans le paragraphe précédent.

Le chapitre 5 est consacré à un phénomène ne se produisant pas dans les corps finis. En travaillant dans un corps infini, nous somme confrontés à la taille des éléments manipulés. En effet, la taille d'un entier (son nombre de chiffres) ou la taille d'un polynôme (son degré) n'est pas bornée. De plus, le temps de calcul d'une opération élémentaire sur de tels éléments dépend de leur taille. Nous illustrerons ces propos par un exemple montrant que l'algorithme de reconstruction est inutilisable en raison de sa durée d'exécution. En effet, la taille des nombres manipulés augmente exponentiellement avec le nombre d'étape dans l'algorithme. Nous utiliserons les outils habituels, à savoir la réduction modulo un idéal maximal, pour résoudre ce problème.

Après avoir présenté les idéaux maximaux des corps de nombres et des corps de fonctions que nous sommes amenés à utiliser, nous définirons la réduction d'un code correcteur, notion propre aux codes sur des corps infinis. Nous montrerons ensuite que, sous des conditions raisonnables, la réduction d'un code de Gabidulin généralisé est un code de Gabidulin sur un corps fini.

Enfin, le chapitre 6 est consacré aux codes espace-temps. Nous commencerons par décrire les codes espace-temps et donner quelques définitions et propriétés, que nous illustrerons avec le code orthogonal ([HLL08]). Nous aborderons ensuite le problème du décodage. La majoration de la probabilité d'erreur nous permettra de dégager deux critères garantissant de bonnes propriétés au code : le critère du rang et le critère du déterminant. C'est ce premier critère qui explique l'utilisation de codes de Gabidulin pour la conception de codes espace-temps.

1. Le nombre s d'effacements est la somme du nombre s_c d'effacements de colonnes et du nombre s_ℓ d'effacements de lignes.

Nous présenterons ensuite une première construction basée sur les codes de Gabidulin : la construction unifiée ([LK05]). Puis nous définirons une nouvelle construction, basée sur les codes de Gabidulin généralisés. Dans notre construction comme dans la construction unifiée, les deux étapes importantes sont la conception d'une famille de matrices éloignées les unes des autres en métrique rang (ce qui explique l'utilisation de codes en métrique rang) et la conception d'une bijection permettant de définir un nombre complexe à partir de valeurs binaires (l'étiquetage). Les codes de Gabidulin sur des corps de nombres sont donc adaptés à une telle construction. Nous préciserons les paramètres libres de notre construction, et leur influence sur les paramètres du code espace-temps.

Enfin, nous comparerons notre construction aux constructions existantes. Nous verrons que pour des codes de diversité maximale, nos codes présentent les mêmes paramètres que les codes existants.

Les résultats du chapitre 2 et des deux premières sections du chapitre 3 ont été publiés en juillet 2013, lors de la conférence ISIT (*International Symphosium on Information Theory*). Les chapitres 2 à 5 font l'objet d'un article en cours de rédaction. Enfin, les résultats concernant les codes espace-temps (chapitre 6) ont été présentés en mai 2015 à la conférence WCC (*Workshop on Codes and Cryptography*).

Table des matières

0.1 Historique . . . 3
0.2 Description des travaux . . . 4

1 Introduction : les codes correcteurs **9**
1.1 Approche probabiliste des codes correcteurs . . . 10
1.2 Approche algébrique des codes correcteurs . . . 11
1.3 Quelques autres aspects des codes correcteurs . . . 14
1.4 Transmissions MIMO . . . 15
1.4.1 Modulation . . . 15
1.4.2 Le canal SISO . . . 16
1.4.3 Le canal MIMO . . . 17

2 θ-polynômes et métriques basées sur le rang **19**
2.1 θ-polynômes . . . 19
2.1.1 Définition . . . 21
2.1.2 Racines et espace des racines . . . 23
2.1.3 Polynômes annulateurs et interpolateurs . . . 24
2.2 Métriques basées sur le rang . . . 26
2.2.1 Métriques sur des matrices . . . 27
2.2.2 Métrique rang sur L^n . . . 29
2.3 Cadre de travail . . . 33
2.3.1 Exemples . . . 33
2.3.2 Équivalence et extensions cycliques . . . 34
2.3.3 Quelques extensions cycliques . . . 35

3 Codes de Gabidulin généralisés **37**
3.1 Définition . . . 38
3.2 Décodage et reconstruction (modèle d'erreurs seules) . . . 43
3.3 Premier modèle d'effacement . . . 46
3.4 Second modèle d'effacement . . . 49

4 Un algorithme de reconstruction de type Welch-Berlekamp **53**
4.1 Algorithme de type Welch-Berlekamp . . . 53
4.2 Preuve . . . 56
4.3 Complexité . . . 59
4.3.1 Complexité des opérations sur les θ-polynômes . . . 61

4.3.2 Complexité de l'algorithme de Reconstruction 62
4.3.3 Complexité des algorithmes de décodage 65
4.4 Améliorations 66
4.4.1 Une version sans division 66
4.4.2 Polynômes de plus bas degré 67
4.4.3 Mise à jour des défauts 68

5 Codes de Gabidulin généralisés et finis 71
5.1 Problématique : explosion des coefficients 72
5.2 Réduction d'un code de Gabidulin 73
5.2.1 Quotient d'un code de Gabidulin généralisé 74
5.2.2 Décodage d'un code de Gabidulin généralisé via une réduction 75
5.3 Idéaux maximaux 76
5.3.1 Cas des corps de nombres 76
5.3.2 Cas des corps de fonctions 78

6 Application au codage espace-temps 81
6.1 Présentation du codage espace-temps 82
6.2 Codes espace-temps basés sur des codes de Gabidulin 85
6.2.1 Construction de Lu et Kumar 85
6.2.2 Construction à base de codes de Gabidulin généralisés 86
6.2.3 Une nouvelle famille de constellations 88
6.3 Comparaison 90

7 Conclusion / Perspectives 93

Chapitre 1

Introduction : les codes correcteurs

Le domaine des codes correcteurs est relativement récent dans l'univers mathématique. Le problème au cœur de ce domaine est le suivant :

Comment transmettre un message de façon fiable,
à travers un canal qui ne l'est pas ?

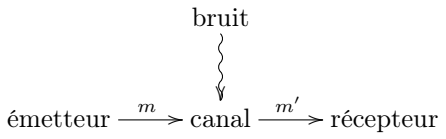

FIGURE 1.1 – Canal de transmission bruité.

Pendant la seconde guerre mondiale, C. Shannon s'intéresse à la localisation d'éléments significatifs dans les messages chiffrés. C'est ainsi que sont nées la théorie de l'information et la modélisation d'un canal de communication. La théorie de l'information est un vaste domaine, comprenant entre autres le codage de l'information. Ce codage peut prendre deux formes : le codage de source et le codage de canal. Le codage de source consiste à transmettre efficacement de l'information à travers un canal fiable (sans bruit). Cela s'applique par exemple dans la compression sans perte. Le codage de canal concerne les canaux bruités, c'est-à-dire que l'information peut être modifiée aléatoirement en traversant le canal. La mise en place de codes correcteurs permet de contrer cet aléa.

Cette approche probabiliste fait l'objet de la première section de l'introduction, à travers le canal binaire symétrique et le code de répétition. Nous énoncerons le théorème de Shannon [Sha48], qui annonce l'existence de codes, et qui motivera la recherche d'exemples explicites.

Quelques années plus tard, les premiers exemples de codes correcteurs font leur apparition, avec notamment le code de Hamming [Ham50] dès 1950. Suivront les codes de Reed-Solomon ([RS60]), les codes BCH ([BRC60] et [Hoc59]) et bien d'autres. Des algorithmes de décodage spécifiques à une famille de codes font également leur apparition, comme par exemple [WB86].

La deuxième section de cette introduction a pour but de présenter une approche plus algébrique des codes correcteurs. Nous présenterons les problèmes classiques au travers du code de Hamming.

Les communications sans fil remontent aux années 1890 avec les télégraphes sans fil. Près d'un siècle plus tard, apparait l'idée d'utiliser plusieurs antennes en même temps (systèmes MIMO). En 1999, E. Telatar étend la théorie de l'information à ces systèmes [T+99]. En parallèle, les premiers codes correcteurs dédiés aux transmissions MIMO font leur apparition. Ils sont appelés codes espace-temps.

La quatrième section de cette introduction a pour but de présenter le canal MIMO. Après avoir décrit les transmissions par ondes, nous aborderons le cas des transmissions SISOn où émetteur et récepteur disposent chacun d'une antenne. Enfin, nous présenterons le canal MIMO, où émetteur et récepteur disposent chacun de plusieurs antennes. Une étude plus détaillée sera décrite dans le chapitre 6.

Enfin, la dernière section a pour but de décrire les travaux présentés dans ce document.

1.1 Approche probabiliste des codes correcteurs

Considérons un message écrit dans l'alphabet binaire $\{0, 1\}$. Nous voulons transmettre ce message à travers un canal binaire symétrique. Dans un tel canal, chaque caractère du message est transmis indépendamment des autres [1]. Il a une probabilité $1 - p$ d'être transmis correctement, et p d'être faux. Nous allons utiliser un code correcteur pour protéger notre transmission des erreurs. Le code le plus simple est le code de répétition, comme illustré dans la figure 1.2.

$$\text{émetteur} \longrightarrow \text{canal} \longrightarrow \text{récepteur}$$

$$\text{émetteur} \longrightarrow \text{codeur} \longrightarrow \text{canal} \longrightarrow \text{décodeur} \longrightarrow \text{récepteur}$$

$$\text{émetteur} \xrightarrow[1]{} \text{codeur} \xrightarrow[111]{} \text{canal} \xrightarrow[110]{} \text{décodeur} \xrightarrow[1]{} \text{récepteur}$$

FIGURE 1.2 – Canal de transmission sans, puis avec codage. Exemple du code de répétition.

Le codage consiste simplement à répéter n fois chaque symbole. Ainsi, le message 1 devient $1 \cdots 1$. Le décodage quant à lui consiste à extraire le symbole majoritaire. S'il y a autant de 0 que de 1, le décodage échoue.

n	probabilité de décodage correct	probabilité de décodage incorrect	échec
1	$1-p$	p	
2	$(1-p)^2$	p^2	$2p(1-p)$
3	$(1-p)^3 + 3(1-p)^2 p$	$p^3 + 3p^2(1-p)$	
4	$(1-p)^4 + 4(1-p)^3 p$	$p^4 + 4p^3(1-p)$	$6p^2(1-p)^2$
5	$(1-p)^5 + 5(1-p)^4 p + 10(1-p)^3 p^2$	$p^5 + 5p^4(1-p) + 10p^3(1-p)^2$	

Le théorème de Shannon s'exprime ainsi :

Théorème 1. *Tant que le taux de transmission* [2] *est inférieur à la capacité* [3] *du canal, alors il existe un schéma de codage/décodage tel que la probabilité de décodage incorrecte soit aussi faible*

1. Aussi le message que nous voulons envoyer sera de longueur 1.
2. Voir [PiR89] pour plus de détails.
3. Voir [PiR89] pour plus de détails.

que l'on veut.

Exemple 1. *Prenons* $p = 0.1$. *Dans ce cas, la capacité du canal binaire symétrique est :*

$$C = 1 + p \log p + (1-p)\log(1-p) \simeq 0,53.$$

Sans code correcteur, le taux de transmission est $\tau = 1$, *et est donc supérieur à la capacité du canal. Il est alors impossible d'avoir une transmission fiable. En utilisant un code de répétition, le taux de transmission* τ *diminue. Il est donc possible de transmettre avec une probabilité d'erreur aussi faible que l'on veut.*

n	τ	*correct*	*incorrect*	*indécis*
1	1	$0,9$	$0,1$	
2	$1/2$	$0,81$	$0,01$	$0,18$
3	$1/3$	$0,972$	$0,028$	
4	$1/4$	$0,9477$	$0,0037$	$0,0486$
5	$1/5$	$0,99144$	$0,00856$	

Cet exemple montre qu'il est possible d'avoir une transmission aussi fiable que voulue, mais le prix à payer est un taux de transmission faible.

L'objectif est donc de trouver un code :

- dont le taux de transmission est aussi proche que possible de la capacité du canal,
- qui permet une probabilité de décodage incorrect aussi faible que voulue.

Le théorème de Shannon affirme qu'un tel code existe, mais ne donne aucune construction.

En pratique, les codes utilisés sont des codes en bloc. Ils permettent d'améliorer la transmission en protégeant les symboles d'information d'un certain nombre d'erreurs. Dans le meilleur cas, cela permet de passer de "un message de longueur k est transmis correctement si tous les k symboles qui le constituent sont transmis correctement" à "un message de longueur k est transmis correctement si au moins k symboles parmi les n symboles effectivement transmis sont transmis correctement".

1.2 Approche algébrique des codes correcteurs

Codes correcteurs en bloc Considérons le canal de communication représenté sur la figure 1.1. Pour lutter contre le bruit lié au canal, nous allons utiliser des blocs de n symboles pour transmettre k symboles du message. Cet ajout de $n-k$ symboles se fait selon une règle, le codage, connue par l'émetteur et le récepteur. Cette règle est une bijection entre les mots d'information (toutes les suites de k symboles du message que l'on veut transmettre) et les mots du code (certaines suites de n symboles que l'on transmet effectivement).

$$\text{codage : } \mathbb{F}_2^k \longrightarrow \mathcal{C} \subsetneq \mathbb{F}_2^n$$

Le code $\mathcal{C}$ est donc un sous ensemble de $\mathbb{F}_2^n$, de cardinal 2^k. Il est dit linéaire s'il s'agit d'un espace vectoriel. Dans ce cas, le codage se doit d'être également linéaire. Le code est de dimension k et de longueur n, et on dit qu'il est de paramètres $[n, k]$.

Exemple 2. *Le code de Hamming est un code de dimension* 4 *et de longueur* 7. *À un mot d'information* $(x_1, x_2, x_3, x_4) \in (\mathbb{F}_2)^4$ *est associé le mot du code* $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \in (\mathbb{F}_2)^7$, *dans*

lequel x_5, x_6 et x_7 sont définis par :

$$x_5 = x_1 + x_2 + x_4,$$
$$x_6 = x_1 + x_3 + x_4,$$
$$x_7 = x_2 + x_3 + x_4.$$

Une matrice génératrice d'un code linéaire est une matrice $G \in \mathcal{M}_{k,n}(\mathbb{F}_2)$ dont les lignes forment une base de $\mathcal{C}$. Une telle matrice n'est pas unique et chacune fournit une règle de codage. Le choix d'une règle de codage se fait donc par le choix d'une de ces matrices, qui sera appelée *la* matrice génératrice du code. Le mot du code c s'obtient à partir du mot d'information m par la relation $c = m \cdot G$.

Exemple 3. *Le code de Hamming admet pour matrice génératrice :*

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

Un code correcteur est dit systématique si les k premiers coefficients des mots du code sont les k symboles du mot d'information correspondant. Dans ce cas, les k premières colonnes de la matrice génératrice forment la matrice identité.

Exemple 4. *Le code de Hamming ainsi défini est systématique.*

Distance Le poids de Hamming $w_{\mathcal{H}}(x)$ d'un mot $x \in (\mathbb{F}_q)^n$ est le nombre de coordonnées x_i non nulles.

$$w_{\mathcal{H}}(x) = \#\{i : x_i \neq 0\}$$

Il permet de définir la distance entre des mots comme le poids de leur différence. En d'autres termes, la distance de Hamming $d_{\mathcal{H}}(x, y)$ entre deux mots $x \in (\mathbb{F}_q)^n$ et $y \in (\mathbb{F}_q)^n$ est le nombre de coordonnées distinctes de x et y.

$$d_{\mathcal{H}}(x, y) = \#\{i : x_i \neq y_i\}$$

Enfin, la distance (de Hamming) minimale d d'un code est la plus petite distance entre deux mots distincts du code.

$$d = \min\{d_{\mathcal{H}}(x, y) : x \in \mathcal{C}, y \in \mathcal{C}, x \neq y\}$$

Exemple 5. *Le code de Hamming que nous avons défini a pour distance minimale* 3.

Si la distance minimale d du code est connue, ses paramètres sont notés $[n, k, d]$ au lieu de $[n, k]$. Les performances d'un code dépendent directement de cette distance minimale.

Canal à erreur Dans ce type de canal, certains coefficients du mot du code $c = m \cdot G$ peuvent être changés durant la transmission. Cela est modélisé par l'ajout d'une erreur $e \in \mathbb{F}_2^n$. Le récepteur obtient alors un mot de la forme $y = c + e = m \cdot G + e$.

Il est ensuite confronté aux problèmes suivants.

- Détection d'erreur : déterminer si le mot y reçu est conforme au mot du code c envoyé ou s'il a subi une erreur.

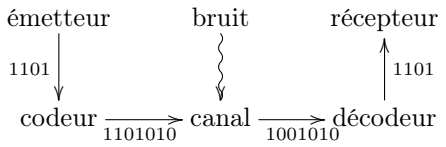

FIGURE 1.3 – Utilisation d'un code pour corriger les erreurs.

- Correction d'erreur : à partir du mot reçu y, retrouver le mot du code c envoyé.
- Décodage : à partir du mot reçu y, retrouver le mot d'information m.

La correction d'erreur et le décodage sont théoriquement équivalents, les mots d'information étant en bijection avec les mots du code. Nous pouvons toujours passer du mot décodé au mot corrigé en le re-codant. Le passage en sens inverse dépend du code. Il est immédiat pour les codes systématiques, et requiert la résolution d'un système linéaire dans les autres cas. Dans la plupart des utilisations, les codes correcteurs ont pour but de protéger les symboles d'information. C'est donc le décodage qui sera étudié, plutôt que la correction.

Proposition 2. *Un code de distance minimale d permet de détecter une erreur si son poids est inférieur ou égal à $d-1$. Un code de distance minimale d permet de décoder (de corriger) une erreur si son poids est inférieur ou égal à $\frac{d-1}{2}$.*

Remarque 1. *Il n'est pas possible de profiter à la fois de la capacité de détection et de la capacité de correction. Si nous utilisons un code pour détecter des erreurs, il n'est pas possible de savoir s'il sera possible de corriger celles-ci, en d'autre termes, nous ne savons pas si son poids est entre* 0 *et* $\frac{d-1}{2}$ *ou entre* $\frac{d-1}{2}$ *et* $d-1$*. Tenter de corriger une erreur dont le poids dépasse la capacité de correction aboutira à un mot du code, mais pas nécessairement le bon.*

Canal à effacements Dans ce type de canal, certains coefficients du mot du code $c = m \cdot G$ peuvent être effacés durant la transmission. La position des effacements est connue, et les symboles en question sont remplacés par le symbole ?. Le récepteur reçoit donc un mot $y \in (\mathbb{F}_2 \cup \{?\})^n$.

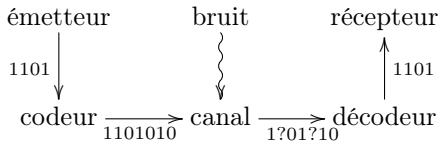

FIGURE 1.4 – Utilisation d'un code pour corriger les effacements.

Il est ensuite confronté aux problèmes suivants.

- Correction d'effacement : à partir du mot reçu y, retrouver le mot du code c envoyé.
- Décodage : à partir du mot reçu y, retrouver le mot d'information m.

Comme dans le cas du canal à erreur, les deux problèmes sont équivalents.

Proposition 3. *Un code de distance minimale d permet de décoder (resp. de corriger) un effacement si son poids est inférieur ou égal à $d-1$.*

Exemple 6. *Le code de Hamming de paramètres* $[7, 4, 3]$ *permet :*

- *de détecter jusqu'à* 2 *symboles erronés,*
- *de corriger jusqu'à* 1 *symbole érroné,*
- *de corriger jusqu'à* 2 *symboles effacés.*

Codes MRD Des propositions 2 et 3, il découle que plus la distance minimale d'un code est élevée, plus le code est résistant. Inversement, plus la dimension du code est élevée, plus il transmet d'information. Les codes les plus intéressants sont donc ceux ayant (pour une longueur fixée) la plus grande distance minimale et la plus grande dimension. R. C. Singleton [Sin64] a montré que ces quantités sont reliées entre elles par la relation :

$$d + k \leqslant n + 1$$

appelée borne de Singleton. Il n'est donc pas possible d'avoir à la fois une grande capacité de correction et un taux de transmission élevé. Les codes pour lesquels la borne précédente est atteinte sont appelés codes MDS (*Maximum Distance Separable*).

Décodage efficace Les problèmes abordés (détection, corrections, décodages) peuvent se résoudre par recherche exhaustive. De tels algorithmes consistent à parcourir tous les mots du code.

- Détection : rendre "correct" si le mot reçu est dans la liste des mots du code, "erroné" sinon.
- Correction d'effacements : pour chaque mot du code, supprimer les coefficients adéquats. Si le poids de l'effacement est inférieur ou égal à $d - 1$, un seul correspondra au mot reçu.
- Correction d'erreur : pour chaque mot du code, calculer sa distance au mot reçu. Si le poids de l'erreur est inférieur ou égal à $\frac{d-1}{2}$, un seul mot du code minimise cette distance.
- Décodage (erreur ou effacement) : comme pour les algorithmes de correction, mais dans ce cas, nous parcourons l'ensemble des mots d'information et nous calculons le mot du code correspondant.

Ces algorithmes sont valables pour tous les codes correcteurs, mais ne sont utilisables que sur des codes ayant peu de mots. En effet, le décodage demande en moyenne $\frac{q^k}{2}$ fois plus de calculs que le codage, pour une complexité en $O(q^k n^\omega)$, ω étant le coût de l'algèbre linéaire. Il faut donc trouver des algorithmes plus efficaces.

Le problème de détection est résolu efficacement par le calcul du syndrome. Celui-ci est nul si et seulement si le mot reçu est un mot du code, et son calcul requiert $O(n^2(n-k))$ opérations.

Les positions effacées étant connues du récepteur, le décodage des effacements se ramène à la résolution d'un système linéaire.

Le problème le plus délicat est le décodage d'erreur. Il existe des algorithmes de décodage efficaces, mais ces derniers ne concernent qu'une famille restreinte de codes. Ainsi, il existe une méthode de décodage propre aux codes de Hamming, plusieurs algorithmes pouvant décoder les codes de Reed-Solomon, ...

1.3 Quelques autres aspects des codes correcteurs

Matrice de contrôle La matrice de contrôle d'un code $\mathcal{C}$ est une matrice H telle que $c \cdot H^t$ est nul pour tout mot du code. Elle comporte $n-k$ lignes et n colonnes. Cette matrice peut également être vue comme la matrice génératrice d'une autre code. Ce code est appelé le code dual de $\mathcal{C}$ et noté $\mathcal{C}^\perp$.

Exemple 7. *Dans le cas du code de Hamming, une matrice de contrôle est :*

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

Codes de Reed-Solomon Les codes de Reed-Solomon sont une famille de codes beaucoup utilisés (lecteurs CD, transmissions ADSL, ...). Ils font partie des codes linéaires, mais sont généralement décrits d'une autre manière. Les symboles d'information sont les coefficients d'un polynôme f de degré strictement inférieur à k. Le codage consiste à évaluer ce polynôme f en n valeurs distinctes $g_1, \ldots, g_n$:

$$\mathcal{C} = \{(f(g_1), \ldots, f(g_n) : f \in \mathbb{F}_q[X], \deg(f) < k\}.$$

Leur utilisation s'explique par les deux propriétés suivantes :

- ces codes sont MDS : pour une longueur n et une dimension k (resp. distance minimale d) fixées, la distance minimale (resp. la longueur) et donc la capacité de correction (resp. le taux de transmission) est maximale.
- il existe des algorithmes de décodage polynomiaux (voir par exemple l'algorithme de Welch-Berlekamp [BW86]).

1.4 Transmissions MIMO

À chaque type de transmission est associé une modélisation, et des codes correcteur spécifiques doivent être conçu pour chacune d'elles. Les codes espaces-temps, que nous présenterons plus en détail dans le chapitre 6, sont les codes dédiés aux transmissions sans-fil à plusieurs antennes. Cette section a pour but de présenter ce canal de communication.

1.4.1 Modulation

Une information numérique ne peut pas être transmis directement par ondes radio. Il doit d'abord être modulé. Cela se fait en faisant varier certains paramètres d'une onde porteuse, comme sa phase ou son amplitude. Ces variations, qui prennent un nombre fini de valeurs, peuvent être traduites en un nombre complexe, la variation de la phase (resp. de l'amplitude) étant interprétée comme un argument (resp. un module).

À l'ensemble des variations possibles correspond donc une famille finie de nombres complexes, appellée constellation. Les constellations les plus connues sont les PAM (variations d'amplitude), PSK (variation de phase), QAM et HEX.

Chaque élément de la constellation correspond à un signal binaire via un étiquetage. Cet étiquetage est la phase de l'encodage qui transforme l'information numérique (sous forme de bits) en un signal analogique (un nombre complexe).

Exemple 8. *Nous voulons transmettre le message* 010110001*, en utilisant la constellation* 8*-PSK, qui n'est autre que l'ensemble des racines huitièmes de l'unité. Les bits sont regroupés par trois, puis un nombres complexe est associé à chaque triplet de la façon suivante.*

information	000	001	010	011	100	101	110	111
nombre complexe	1	ζ	ζ^2	ζ^3	ζ^4	ζ^5	ζ^6	ζ^7

où $\zeta = \frac{1}{\sqrt{2}}$ *est une racine primitive de l'unité. Le message est alors interprété comme la suite de complexes* i, $-i$, $\frac{1}{\sqrt{2}}$.

1.4.2 Le canal SISO

Une transmission SISO (Single Input Single Output) est un système de transmission sans fil dans lequel l'émetteur et le récepteur disposent chacun d'une antenne (sur la même fréquence). Le signal émis va suivre plusieurs chemins avant d'arriver à l'antenne réceptrice. En effet, le signal se propage dans toutes les directions, et va rebondir sur des obstacles.

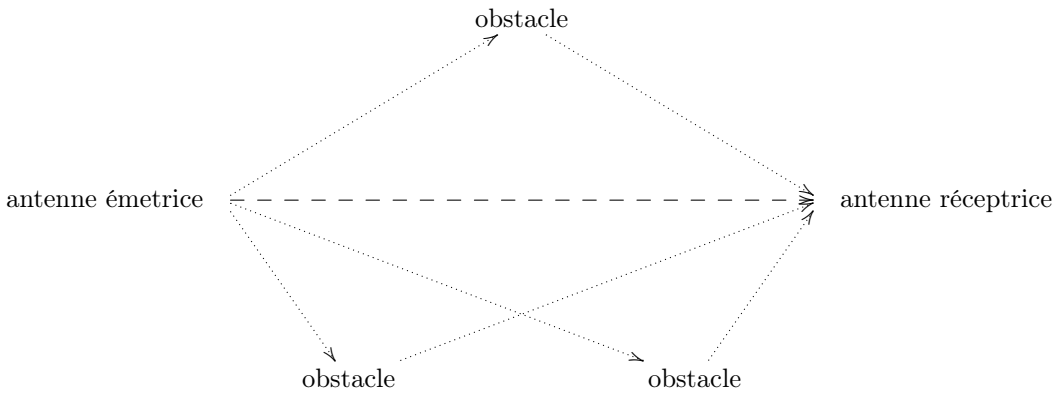

FIGURE 1.5 – Chemin multiple suivi par le signal lors d'une transmission SISO. Le signal peut être plus fiable sur un chemin que les autres.

Le récepteur reçoit alors la somme des signaux correspondant à chaque chemin. Selon le chemin suivi, le signal aura subi une atténuation différente, modélisée par la multiplication par $\alpha_i \in \mathbb{R}_+$, et un déphasage différent ϕ_i, modélisé par la multiplication du signal par un nombre complexe δ_i de module 1. Au signal $x = A\cos(\omega t + \phi)$ se substitue

$$\begin{aligned}\sum_{i=1}^{n} \alpha_i A\cos(\omega t + \phi_i) &= \sum_{i=1}^{n} \alpha_i \delta_i A\cos(\omega t)\\ &= (\sum_{i=1}^{n} \alpha_i \delta_i) A\cos(\omega t)\\ &= hA\cos(\omega t).\end{aligned}$$

Au final, la distorsion du signal est modélisée par le nombre complexe $h = \sum_{i=1}^{n} \alpha_i \delta_i$. Ce nombre h est modélisé par une variable aléatoire de loi :

- de Rice s'il y a une direction particulière selon laquelle le signal est plus fort,
- de Rayleigh sinon.

Enfin, du bruit (gaussien) est ajouté, de sorte que le récepteur obtient le signal $y = hx + e$. Nous supposons le canal connu (i.e. h connu) en réception. Il suffit alors de calculer $h^{-1}y = x + h^{-1}e$, et nous pouvons retrouver le signal x si l'erreur $h^{-1}e$ n'est pas trop importante [4].

4. En comparaison avec l'éloignement des points de la constellation.

1.4.3 Le canal MIMO

Un système de transmission MIMO (Multiple Input Multiple Output) est un système sans fil dans lequel l'émetteur et le récepteur disposent chacun de plusieurs antennes (toutes sur la même fréquence).

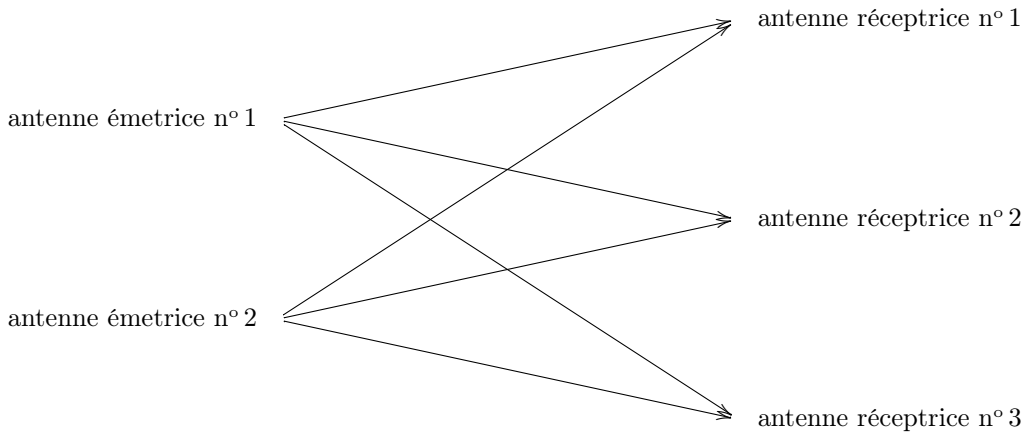

FIGURE 1.6 – Transmission MIMO. Chaque antenne émettrice est reliée à chaque antenne réceptrice par une transmission SISO (voir figure 1.5).

Le nombre d'antennes en émission (resp. en réception) est noté n_t (reps. n_r). À un instant donné, chaque antenne émettrice envoie un signal, représenté par un nombre complexe [5] $x_i \in \mathbb{C}$. Chaque antenne réceptrice reçoit alors une superposition de ces signaux, déformés [6], à laquelle s'ajoute du bruit :

$$\forall j \in [[1, n_r]],\ y_j = \sum_{i=1}^{n_t} h_{i,j} x_i + e_j.$$

Cela s'écrit de façon condensée de la manière suivante :

$$\begin{pmatrix} y_1 \\ \vdots \\ y_{n_r} \end{pmatrix} = \begin{pmatrix} h_{1,1} & \cdots & h_{n_t,1} \\ \vdots & \ddots & \vdots \\ h_{1,n_r} & \cdots & h_{n_t,n_r} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{n_t} \end{pmatrix} + \begin{pmatrix} e_1 \\ \vdots \\ e_{n_r} \end{pmatrix}.$$

Ce procédé est répété T fois, les coefficients $h_{i,j}$ étant supposés ne pas varier pendant cette durée. Les signaux émis et reçus sont reliés par la formule suivante :

$$\begin{pmatrix} y_{1,1} & \cdots & y_{1,T} \\ \vdots & \ddots & \vdots \\ y_{n_r,1} & \cdots & y_{n_r,T} \end{pmatrix} = \begin{pmatrix} h_{1,1} & \cdots & h_{n_t,1} \\ \vdots & \ddots & \vdots \\ h_{1,n_r} & \cdots & h_{n_t,n_r} \end{pmatrix} \begin{pmatrix} x_{1,1} & \cdots & x_{1,T} \\ \vdots & \ddots & \vdots \\ x_{n_t,1} & \cdots & x_{n_t,T} \end{pmatrix} + \begin{pmatrix} e_{1,1} & \cdots & e_{1,T} \\ \vdots & \ddots & \vdots \\ e_{n_r,1} & \cdots & e_{n_r,T} \end{pmatrix}.$$

Un code correcteur pour ce type de transmission est un sous-ensemble $\mathcal{S}$ de l'ensemble des matrices à coefficients dans une constellation $\mathcal{Q}$:

$$\mathcal{S} \subsetneq \mathcal{M}_{n_t,T}(\mathcal{Q}).$$

5. En fait, x est dans une certaine constellation.

6. Pour chaque paire d'antennes, la déformation est celle d'une transmission suivant plusieurs chemins, comme dans le cas d'une transmission SISO : $x \rightsquigarrow hx$ présentée figure 1.5.

Ces codes sont appelés codes espace-temps, car ils possèdent une redondance dans l'espace (via les antennes décalées dans l'espace) et une redondance dans le temps (la transmission d'un mot du code s'étale sur plusieurs instants).

Chapitre 2

θ-polynômes et métriques basées sur le rang

Les codes de Gabidulin sont, à l'instar des codes de Reed-Solomon, des codes d'évaluation. il y a toutefois deux différences essentielles entre ces deux familles de codes.

Pour commencer, les objets qui sont évalués ne sont pas des polynômes mais des θ-polynômes. La première partie de ce chapitre a pour but de présenter ces polynômes particuliers et leurs propriétés.

Ensuite, l'erreur n'est plus mesurée en métrique de Hamming, mais en métrique rang. La deuxième partie de ce chapitre a pour but de présenter les métriques en général, et celles que nous utiliserons en particulier. Nous verrons donc la métrique *term-rank* et la métrique rang, pour laquelle nous donnerons une définition alternative.

Les notions que nous allons définir concernent une extension de corps $K \hookrightarrow L$ de degré fini $[L : K] = m$. Nous considérerons également un automorphisme d'extension $\theta \in \mathrm{Aut}_K(L)$, c'est-à-dire un automorphisme de L qui fixe tous les éléments de K. Ce groupe d'automorphismes est également appelé le groupe de Galois de l'extension et noté $\mathrm{Gal}(K \hookrightarrow L)$.

Les notions de θ-polynômes et de métriques rang existent déjà sur des corps finis, et nous allons dans un premier temps les généraliser à toute extension de degré fini. Certaines des propriétés que nous allons voir au cours des deux premières parties de ce chapitre ne seront valables qu'à certaines conditions. Ces propriétés nous seront nécessaires pour la conception de codes de Gabidulin généralisés. La troisième partie de ce chapitre a pour but de préciser ces conditions, et de décrire une cadre de travail (conditions sur l'extension et l'automorphisme) propice à la généralisation des codes de Gabidulin. Nous terminerons en présentant quelques exemples de telles extensions et automorphismes.

Ces résultats ont été publiés et présentés en juillet 2013, lors de la conférence ISIT (*International Symphosium on Information Theory*).

2.1 θ-polynômes

Considérons un anneau A, et son anneau de polynômes

$$A[X] = \left\{ \sum_{i=0}^{n} a_i X^i, a_i \in A, n \in \mathbb{N} \right\}.$$

Le produit de deux polynômes peut se définir par la formule

$$X \cdot a = a \cdot X, a \in A$$

que nous étendons ensuite aux polynômes par distributivité. Cette définition a deux conséquences pour les polynômes. Tout d'abord, nous obtenons la règle des degrés :

$$\forall P \in A[X], \forall Q \in A[X], \deg(P \cdot Q) \leqslant \deg(P) + \deg(Q). \tag{2.1}$$

Si en plus l'anneau A est intègre, il s'agit d'une égalité. Ensuite, le produit est commutatif si A est lui-même commutatif.

Il est possible de définir un produit ne vérifiant pas cette dernière condition. Nous obtenons ainsi les polynômes tordus décrits dans [Ore33b]. Afin de satisfaire la règle des degrés, ce produit doit être défini par

$$X \cdot a = \theta(a) \cdot X + \delta(a).$$

Du calcul de $X \cdot (a + b)$ et $X \cdot (ab)$ se dégagent les propriétés de θ et δ :

$$\begin{array}{cc} \theta(a+b) = \theta(a) + \theta(b) & \delta(a+b) = \delta(a) + \delta(b) \\ \theta(ab) = \theta(a)\theta(b) & \delta(ab) = \theta(a)\delta(b) + \delta(a)b \end{array}$$

Il s'ensuit que θ doit être un endomorphisme de A et que δ est appelée une θ-dérivation. Ces deux applications font alors partie de la définition de l'anneau de polynômes, noté

$$A[X; \theta, \delta].$$

Dans ce cas, l'intégrité de A ne suffit plus à garantir le cas d'égalité dans (2.1), il faut en plus que θ soit un automorphisme.

Le même auteur donne un exemple de tels anneaux dans [Ore33a]. Il s'agit de l'ensemble des q-polynômes

$$\left\{ \sum a_i X^{q^i}, a_i \in \mathbb{F}_{q^m} \right\}$$

muni de l'addition et de la composition des polynômes. Cet ensemble peut être écrit comme un ensemble de θ-polynômes, en considérant le $(\mathbb{F}_q)$-automorphisme de Frobenius $\theta : x \mapsto x^q$ et la dérivation nulle $\delta : x \mapsto 0$. Nous pouvons alors écrire une correspondance entre les deux écritures.

$$\begin{array}{ccc} \mathbb{F}_{q^m}[X;\theta] & \longrightarrow & \mathbb{F}_{q^m}[X] \\ \sum a_i X^i & \longmapsto & \sum a_i X^{q^i} \\ \text{addition} & & \text{addition} \\ \text{produit} & & \text{composition} \\ X \cdot a = a^q X & & X^q \cdot a = a^q X^q \end{array}$$

Cette correspondance permet de voir les q-polynômes comme des polynômes classiques, et ainsi d'utiliser certaines de leurs propriétés. Considérons par exemple le corps $\mathbb{F}_{32} = \mathbb{F}_2[\alpha]$. Le 2-polynôme $X^3 + \alpha^3 X^2 + (\alpha+1)X + \alpha$ a pour polynôme classique associé $X^8 + \alpha^3 X^4 + (\alpha+1)X^2 + \alpha X$.

Dans cette section, nous définissons et étudions les θ-polynômes, qui sont situés entre les deux constructions précédentes. En effet, ils sont à la fois une généralisation des p-polynômes et un cas particulier des polynômes tordus.

Le but de cette section est d'établir quelles propriétés des p-polynômes restent valables sur des corps infinis, et ainsi de s'affranchir de l'hypothèse de finitude dans les preuves.

q-polynomes	θ-polynomes	polynomes tordus
$X \cdot a = a^q X$	$X \cdot a = \theta(a)X$	$X \cdot a = \theta(a)X + \delta(a)$
$\mathbb{F}_q \hookrightarrow \mathbb{F}_{q^m}$	$K \hookrightarrow L$ cyclique	$K \hookrightarrow L$
$\theta : x \mapsto x^q$	$\theta :$ générateur de $\mathrm{Aut}_K(L)$	$\theta :$ endormorphisme de L
$\delta : x \mapsto 0$	$\delta :$ dérivation triviale	$\delta : \theta$-dérivation de L

FIGURE 2.1 – q-polynômes, θ-polynômes et polynômes tordus

2.1.1 Définition

Définition 1 (θ-polynômes et anneaux de θ-polynômes)**.** *Soient K un corps, L une extension de K de degré fini $[L : K] = m$, et $\theta \in \mathrm{Aut}_K(L)$ un automorphisme. Un θ-polynôme à coefficients dans L est un élément de la forme*

$$\sum_{i \geqslant 0} a_i X^i, a_i \in L$$

avec un nombre fini de a_i non nuls. On note $L[X;\theta]$ l'ensemble des θ-polynômes à coefficients dans L, sur lequel on définit les opérations suivantes. Soient $A = \sum_{i \geqslant 0} a_i X^i \in L[X;\theta]$, $B = \sum_{i \geqslant 0} b_i X^i \in L[X;\theta]$ et $c \in L$.

- *L'addition est définie terme à terme :*

$$A + B = \sum_{i \geqslant 0} (a_i + b_i) X^i.$$

- *Le produit est défini par :*

$$X \cdot c = \theta(c) \cdot X,$$

  ce qui s'étend aux θ-polynômes de la façon suivante :

$$A \cdot B = \sum_{i,j \geqslant 0} a_i \cdot \theta^i(b_j) \cdot X^{i+j}.$$

Définition 2 (degré)**.** *Soit $A = \sum_{i \geqslant 0} a_i X^i \in L[X;\theta]$ un θ-polynôme. Son degré est défini par*

$$\deg(A) = \max\{i : a_i \neq 0\}.$$

Par convention, le degré du θ-polynôme nul est $-\infty$.

Remarque 2. *Le polynôme $A = \sum_{i=0}^{d} a_i X^i \in \mathbb{F}_{q^m}[X; Fr_q]$ a pour polynôme classique associé $\sum_{i=0}^{d} a_i X^{q^i}$. Ainsi, le polynôme classique associé à un q-polynôme de degré d est un polynôme (classique) de degré q^d.*

Hormis la non-commutativité, la structure de cet anneau de polynômes est assez proche de celle des anneaux de polynômes classiques. Seul les anneaux commutatifs peuvent être qualifiés d'euclidiens, nous devrons donc nous contenter de la formulation suivante.

Proposition 4. *$L[X;\theta]$ est un anneau non commutatif, sans diviseurs de zéro. De plus, nous disposons d'une division euclidienne à gauche et à droite : pour tous $A, B \in L[X;\theta]$, il existe $Q_1, Q_2, R_1, R_2 \in L[X;\theta]$ uniques tels que*

$$\begin{aligned} A &= B \cdot Q_1 + R_1, \deg(R_1) \leqslant \deg(B) && \textit{(division à gauche)}, \\ &= Q_2 \cdot B + R_2, \deg(R_2) \leqslant \deg(B) && \textit{(division à droite)}. \end{aligned}$$

Nous pouvons également définir les notions de ppcm et de pgcd. La non-commutativité fait que ces notions doivent être définies *à gauche* et *à droite*. Ils se calculent via l'algorithme d'Euclide étendu comme dans $L[X]$.

Néanmoins, nous trouvons quelques différences importantes entre $L[X]$ et $L[X;\theta]$. Par exemple, la factorisation d'un θ-polynôme n'est pas unique.

Exemple 9. *Considérons l'extension de corps finis $\mathbb{F}_2 \hookrightarrow \mathbb{F}_2[\alpha] = \mathbb{F}_2[X]/(X^2+X+1)$. nous avons alors*

$$X^2 + 1 = (X+1)(X+1) = (X+\alpha)(X+\alpha^2).$$

La différence la plus significative à laquelle nous serons confrontés entre $L[X]$ et $L[X;\theta]$ est la définition de l'évaluation. Il n'est plus possible d'évaluer un polynôme $P \in L[X;\theta]$ en $b \in L$ en "remplaçant X par b". En effet, en évaluant les polynômes de l'exemple précédent en α de cette façon, nous aurions $\alpha^2 = \alpha \times \alpha = 1 \times 0$. Nous devons donc revenir à la définition formelle de l'évaluation : l'évaluation de $P \in L[X;\theta]$ en $b \in L$ est le reste de la division de P par $X - b$ à droite.

Une des particularités des anneaux de polynômes tordus est l'existence d'une autre évaluation. Pour les θ-polynômes, où la dérivation est triviale, cette évaluation se définit ainsi :

Définition 3 (évaluation par opérateurs)**.** *Soit $K \hookrightarrow L$ une extension de corps et $\theta \in \mathrm{Aut}_K(L)$. Soit $A = \sum a_i X^i \in L[X;\theta]$ un θ-polynôme et $b \in L$. L'évaluation par opérateur de A en b est*

$$\mathcal{L}_A(b) = \sum_i a_i \theta^i(b).$$

C'est cette évaluation que nous utiliserons dans la suite. Ce choix est motivé par la remarque suivante.

Remarque 3. *L'évaluation par opérateur d'un θ-polynôme de $\mathbb{F}_q[X;\theta]$ correspond à l'évaluation de son polynôme classique associé.*

Ce choix d'évaluation permet de voir les codes de Gabidulin (dans le cas fini) comme l'évaluation (par opérateur) des q-polynômes et non comme l'évaluation de leurs polynômes classiques associés.

Enfin, nous terminons la présentation des θ-polynômes par les propriétés élémentaires de l'évaluation.

Proposition 5. *Soient $P, R \in L[X;\theta]$ deux θ-polynômes, $a, b \in L$ et $\lambda \in K$. On a les propriétés suivantes :*

$$\begin{aligned} \mathcal{L}_P(\lambda a + b) &= \lambda \mathcal{L}_P(a) + \mathcal{L}_P(b), \\ \mathcal{L}_{PR}(a) &= \mathcal{L}_P(\mathcal{L}_R(a)). \end{aligned}$$

Suite à cette propriété, les θ-polynômes sont parfois appelés polynômes linéaires ou linéarisés.

2.1.2 Racines et espace des racines

Il est bien connu qu'un polynôme classique ne peut pas avoir plus de racines que son degré. Le but de cette section est de fournir une propriété analogue pour les θ-polynômes.

Définition 4 (racines, espace des racines)**.** *Soit $K \hookrightarrow L$ une extension de corps, munie d'un automorphisme $\theta \in \mathrm{Aut}_K(L)$, et soit $A \in L[X;\theta]$ un θ-polynôme. Un élément $b \in L$ est appelé racine de A si*

$$\mathcal{L}_A(b) = 0.$$

L'espace des racines de A est :

$$\mathrm{Roots}(A) = \{b \in L : \mathcal{L}_A(b) = 0\}.$$

Puisque θ est un K-automorphisme, il est clair que tous les éléments de K sont racines du θ-polynôme $X - 1$. Ainsi, il n'est pas envisageable de simplement compter ces racines. Néanmoins, nous avons vu dans la proposition 5 que pour un θ-polynôme $A \in L[X;\theta]$, l'application $b \in L \mapsto \mathcal{L}_A(b) \in L$ est K-linéaire (en voyant L comme un K-espace vectoriel). L'espace des racines de A n'est autre que son noyau, dont nous pouvons calculer la dimension.

Théorème 6. *Soient $K \hookrightarrow L$ une extension de corps et $\theta \in \mathrm{Aut}_K(L)$. Soit $A \in L[X;\theta]$ un θ-polynôme non nul. Si le polynôme caractéristique de θ, vu comme une application K-linéaire, est sans facteur carré, alors la K-dimension de l'espace des racines de A vérifie l'inégalité suivante :*

$$\dim_K(\mathrm{Roots}(A)) \leqslant \deg(A).$$

Dans le cas des q-polynômes, on utilise le fait que l'évaluation correspond à l'évaluation du polynôme classique associé. Ainsi, les racines d'un q-polynôme de degré n sont celles d'un polynôme classique de degré q^n, soit au plus q^n. Celles-ci devant former un $\mathbb{F}_q$ espace vectoriel, celui-ci est de dimension au plus n. Cet argument n'a plus aucun sens dès que le corps est infini.

Démonstration. Soit $A(X) = \sum_{i=0}^{n} a_i X^i$ un θ-polynôme non nul de degré n, et notons $[L : K] = m$ le degré de l'extension. Notons $M \in \mathcal{M}_{m\times m}(K)$ la matrice de θ, vu comme application K-linéaire de L, dans une K-base de L. Par hypothèse, le polynôme caractéristique de cette matrice est sans facteur carré, nous avons donc m valeurs propres distinctes $d_1, \ldots, d_m$ (éventuellement dans une extension L' de L). Ainsi, il existe une matrice de passage $P \in \mathcal{M}_{m,m}(L')$ et une matrice diagonale $D = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_m \end{pmatrix}$ telles que

$$M = P \cdot D \cdot P^{-1}.$$

L'application $b \in L \mapsto \mathcal{L}_A(b) = \sum_i a_i \theta^b(i) \in L$ est également une application K-linéaire de L, nous pouvons donc calculer sa matrice :

$$\begin{aligned} \sum a_i M^i &= P \cdot (\sum a_i D^i) \cdot P^{-1} \\ &= P \cdot \begin{pmatrix} \sum a_i d_1^i & & 0 \\ & \ddots & \\ 0 & & \sum a_i d_m^i \end{pmatrix} \cdot P^{-1} \end{aligned}$$

L'espace des racines de A étant le noyau de cette application, sa dimension est le nombre de 0 sur la diagonale $\sum a_i d_1^i, \ldots, \sum a_i d_m^i$. Les valeurs propres d_i étant distinctes, le polynôme (classique) $\sum_{i=0}^{n} a_i X^i$ ne peut s'annuler que n fois. La dimension de l'espace des racines de A est donc majorée par son degré. □

Remarque 4. *Si le polynôme minimal de θ possède un facteur multiple, alors la dimension de l'espace des racines d'un θ-polynôme A est majorée par $k \deg(A)$, où k désigne le plus grand exposant présent dans la décomposition du polynôme caractéristique.*

Le théorème 6 peut être reformulé de la façon suivante, que nous rencontrerons à de nombreuses reprises.

Corollaire 7. *Soient $v_1, \ldots, v_s \in L^s$ s éléments K-linéairement indépendants de L et soit $f \in L[X;\theta]$ un θ-polynôme tel que*

$$\forall i \in [[1;s]],\ \mathcal{L}_F(v_i) = 0.$$

Si le polynôme caractéristique de θ, vu comme application linéaire, est sans facteur carré, alors soit le degré de f est supérieur ou égal à s, soit f est le θ-polynôme nul.

2.1.3 Polynômes annulateurs et interpolateurs

Intéressons-nous maintenant au problème contraire. Étant donné un sous-K-espace vectoriel de L, existe-t-il un θ-polynôme qui s'annule précisément sur cet espace, et si oui, quel est son degré ?

Théorème 8. *Soit $V \subset L$ un sous-K-espace vectoriel de dimension s de L. Si le polynôme caractéristique de θ, vu comme application linéaire, est sans facteur carré, alors il existe un unique θ-polynôme unitaire $\mathcal{A} \in L[X;\theta]$, de degré s, tel que*

$$\forall v \in V, \mathcal{L}_{\mathcal{A}}(v) = 0,$$

et aucun de degré inférieur.

De plus, si nous disposons d'une base de cet espace, nous pouvons calculer son polynôme annulateur via l'algorithme 1 en $O(s^2)$ opérations dans L.

Définition 5 (polynôme annulateur)**.** *Ce θ-polynôme est appelé polynôme annulateur de V et noté $\mathcal{A}_V$.*

Pour une famille $(v_1, \ldots, v_s) \in L^s$, le polynôme annulateur du K-espace vectoriel engendré par les v_i est noté $\mathcal{A}_{\langle v_1,\ldots,v_s\rangle}$.

Algorithm 1 Calcul du polynôme annulateur

Input: $v_1, \ldots, v_s \in L$, K-linéairement indépendants

Output: $\mathcal{A}(X)$ unitaire tel que $\mathcal{L}_{\mathcal{A}}(v_i) = 0$

0: $\mathcal{A}(X) \leftarrow 1$

0: **for** $1 \leqslant i \leqslant s$ **do**

0: $\quad \mathcal{A}(X) \leftarrow \left(X - \frac{\theta(\mathcal{L}_{\mathcal{A}}(v_i))}{\mathcal{L}_{\mathcal{A}}(v_i)}\right) \cdot \mathcal{A}(X)$

0: **end for**

return $\mathcal{A}(X)$ =0

Démonstration. Soit $(v_1, \cdots, v_s)$ une K-base de V. Pour montrer que le polynôme calculé par l'algorithme 1 est nul sur V, il suffit de vérifier, par récurrence, qu'à la fin de l'étape i, nous avons

$$\forall j \in [[1; i]], \mathcal{L}_{\mathcal{A}}(v_j) = 0.$$

Nous vérifions aisément que le polynôme est unitaire et de degré s. (voir la preuve de la proposition 10 pour plus de détails.) Pour montrer que c'est le seul polynôme vérifiant ces critères, considérons un deuxième tel polynôme. Leur différence est un θ-polynôme de degré $s-1$, qui s'annule également sur V, de dimension s. D'après le corollaire 7, il s'agit du polynôme nul ce qui montre l'unicité. □

Les θ-polynômes annulateurs permettent l'existence de θ-polynômes interpolateurs.

Théorème 9. *Soient $x_1, \ldots, x_s \in L^s$ des éléments K-linéairement indépendants et $y_1, \ldots, y_s \in L^s$ des éléments quelconques de L. Si le polynôme caractéristique de θ, vu comme application linéaire, est sans facteur carré, alors il existe un unique θ-polynôme de degré $s-1$, tel que*

$$\forall i \in [[1; s]], \mathcal{L}_{\mathcal{I}}(x_i) = y_i.$$

Définition 6 (polynôme interpolateur)**.** *Ce θ-polynôme est appelé θ-polynôme interpolateur, et est noté $\mathcal{I}_{[x_1,\ldots,x_s],[y_1,\ldots,y_s]}$.*

Démonstration. Notons $\widehat{x}_i = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_s)$, et considérons le θ-polynôme

$$\mathcal{I}(X) = \sum_{i=1}^{s} y_i \frac{\mathcal{A}_{\langle \widehat{x}_i \rangle}(X)}{\mathcal{L}_{\mathcal{A}_{\langle \widehat{x}_i \rangle}}(x_i)}.$$

Nous vérifions facilement qu'il vérifie les conditions du théorème 9. □

Proposition 10. *Supposons que le polynôme caractéristique de θ est sans facteur carré. Soient $x_1, \ldots, x_s \in L^s$ des éléments K-linéairement indépendants de L et $y_1, \ldots, y_s \in L^s$ des éléments quelconques de L. Alors nous pouvons calculer à la fois les polynômes $\mathcal{A}_{\langle x_1,\ldots,x_s \rangle}$ et $\mathcal{I}_{[x_1,\ldots,x_s],[y_1,\ldots,y_s]}$ en $O(s^2)$ opérations dans L via l'algorithme 2.*

Démonstration. Notons $\mathcal{H}$ l'hypothèse de récurrence suivante :

$$\begin{aligned} \mathcal{H}(r) : \forall i \in [[1; r]], \mathcal{L}_{\mathcal{I}^{(r)}}(x_i) &= y_i, \\ \mathcal{L}_{\mathcal{A}^{(r)}}(x_i) &= 0. \end{aligned}$$

Commençons par montrer que $\mathcal{H}(1)$ est vraie. Pour cela, calculons $\mathcal{A}^{(1)}$ et $\mathcal{I}^{(1)}$.

$$\begin{aligned} \mathcal{A}^{(1)} &= \left(X - \frac{\theta(\mathcal{L}_{\mathcal{A}^{(0)}}(x_1)}{\mathcal{L}_{\mathcal{A}^{(0)}}(x_1)} \right) \mathcal{A}^{(0)} \\ &= \left(X - \frac{\theta(x_1)}{x_1} \right), \\ \mathcal{I}^{(1)} &= 0 + \frac{y_1 - (\mathcal{L}_{\mathcal{I}^{(0)}}(x_1)}{\mathcal{L}_{\mathcal{A}^{(0)}}(x_1)} \mathcal{A}^{(0)} \\ &= \frac{y_1}{x_1}. \end{aligned}$$

Il suffit alors de vérifier les égalités suivantes :

$$\begin{aligned}
\mathcal{L}_{\mathcal{A}^{(1)}}(x_1) &= \mathcal{L}_{\left(X-\frac{\theta(x_1)}{x_1}\right)}(x_1) \\
&= 0, \\
\mathcal{L}_{\mathcal{I}^{(1)}}(x_1) &= \mathcal{L}_{\frac{y_1}{x_1}}(x_1) \\
&= y_1.
\end{aligned}$$

Montrons maintenant que $\mathcal{H}$ est héréditaire. Soit $r \in [[1; s-1]]$ et soient $\mathcal{I}^{(r)}$ et $\mathcal{A}^{(r)}$ des θ-polynômes vérifiant $\mathcal{H}(r)$. Alors nous avons les égalités suivantes :

$$\begin{aligned}
\forall i \in [[1; r]],\ \mathcal{L}_{\mathcal{I}^{(r+1)}}(x_i) &= \mathcal{L}_{\mathcal{I}^{(r)}}(x_i) + \frac{y_{r+1} - \mathcal{L}_{\mathcal{I}^{(r)}}(x_{r+1})}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})}\mathcal{L}_{\mathcal{A}^{(r)}}(x_i) \\
&= y_i + \frac{y_{r+1} - \mathcal{L}_{\mathcal{I}^{(r)}}(x_{r+1})}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})} \cdot 0 \\
&= y_i, \\
\forall i \in [[1; r]],\ \mathcal{L}_{\mathcal{A}^{(r+1)}}(x_i) &= \mathcal{L}_{\left(X-\frac{\theta(\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1}))}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})}\right)\mathcal{A}^{(r)}}(x_i) \\
&= \mathcal{L}_{\left(X-\frac{\theta(\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1}))}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})}\right)}(\mathcal{L}_{\mathcal{A}^{(r)}}(x_i)) \\
&= \mathcal{L}_{\left(X-\frac{\theta(\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1}))}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})}\right)}(0) \\
&= 0,
\end{aligned}$$

$$\begin{aligned}
\mathcal{L}_{\mathcal{I}^{(r+1)}}(x_{r+1}) &= \mathcal{L}_{\mathcal{I}^{(r)}}(x_{r+1}) + \frac{y_{r+1} - \mathcal{L}_{\mathcal{I}^{(r)}}(x_{r+1})}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})}\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1}) \\
&= \mathcal{L}_{\mathcal{I}^{(r)}}(x_{r+1}) + y_{r+1} - \mathcal{L}_{\mathcal{I}^{(r)}}(x_{r+1}) \\
&= y_{r+1}, \\
\mathcal{L}_{\mathcal{A}^{(r+1)}}(x_{r+1}) &= \mathcal{L}_{\left(X-\frac{\theta(\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1}))}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})}\right)\mathcal{A}^{(r)}}(x_{r+1}) \\
&= \theta(\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})) - \frac{\theta(\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1}))}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})} \cdot \mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1}) \\
&= 0,
\end{aligned}$$

qui montrent que $\mathcal{I}^{(r+1)}$ et $\mathcal{A}^{(r+1)}$ vérifient $\mathcal{H}(r+1)$. Ainsi, les polynômes $\mathcal{I}^{(s)}$ et $\mathcal{A}^{(s)}$ en sortie d'algorithme vérifient les conditions requises. □

2.2 Métriques basées sur le rang

La métrique rang est apparue en 1978 dans [Del78], en tant que distance sur l'espace des formes bilinéaires $(V \times V') \to \mathbb{F}_q$, où V et V' désignent des $\mathbb{F}_q$-espaces vectoriels. Elle est reprise en 1985 par E. M. Gabidulin qui l'étudie dans [Gab85], en se concentrant particulièrement sur les codes optimaux (définition 16). Cette métrique est définie sur des vecteurs de L^n, à partir de l'écriture matricielle donnée par $\mathrm{ext}_{\mathcal{B}}$. On peut interpréter cette métrique sur L^n : c'est le nombre maximal de coordonnées K-linéairement indépendantes.

Algorithm 2 Polynômes annulateur et interpolateur

Input: $x_1, \ldots, x_s \in L$ K-linéairement indépendants
$y_1, \ldots, y_s \in L$

Output: $\mathcal{A}(X)$ (unitaire) tel que $\mathcal{L}_{\mathcal{A}}(x_i) = 0$
$\mathcal{I}(X)$ tel que $\mathcal{L}_{\mathcal{I}}(x_i) = y_i$

0: $\mathcal{A}^{(0)} := 1$
0: $\mathcal{I}^{(0)} := 0$
0: **for** $r \in [[0; s-1]]$ **do**
0: $\quad \mathcal{I}^{(r+1)} := \mathcal{I}^{(r)} + \frac{y_{r+1} - \mathcal{L}_{\mathcal{I}^{(r)}}(x_{r+1})}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})} \cdot \mathcal{A}^{(r)}$
0: $\quad \mathcal{A}^{(r+1)} := (X - \frac{\theta(\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1}))}{\mathcal{L}_{\mathcal{A}^{(r)}}(x_{r+1})}) \cdot \mathcal{A}^{(r)}$
0: **end for**
1: **return** $\mathcal{A}^{(s)}, \mathcal{I}^{(s)}$ =0

En 1991, dans [Rot91], R. Roth décrit une autre métrique, qu'il nomme *term-rank metric*. Cette métrique se définit sur un espace de matrices. Elle peut être définie sur des vecteurs de L^n via $\mathrm{ext}_{\mathcal{B}}{}^{-1}$, mais il n'y a pas d'interprétation dans ce cas.

Ces deux métriques ont été définies pour certains canaux de transmission, pour lesquels la métrique de Hamming n'était pas adaptée. Ainsi, la métrique *term-rank* est plus adaptée à du stockage de données réparties dans des tableaux, modèle dans lequel les pannes (effacements) se produisent par colonnes. Quant à la métrique rang (dans sa version vectorielle), elle est adaptée à des erreurs dont la valeur est répétée sur plusieurs coefficients.

Supposons par exemple que l'on veuille stocker des bits dans un tableau, et que les seules erreurs qui puissent se produire soient des inversions de colonnes : les 0 deviennent des 1 et inversement, pour tous les éléments de la colonne. Il est alors possible de corriger n'importe quel nombre de colonnes inversées en utilisant un code en métrique rang de taux très élevé. En effet, l'erreur est dans ce cas de poids 1 en métrique rang, et un code de paramètres $[n, n-2, 3]$ convient. (En fait, il s'agit plutôt d'un effacement (voir section 3.3), et on pourrait augmenter le rendement en utilisant un code de paramètres $[n, n-1, 2]$.)

Enfin, l'apparition du *network coding* a apporté un regain d'intérêt pour la métrique rang. En effet, dans ce modèle où les mots sont des sous-espaces vectoriels, nous construisons des codes à partir de codes en métrique rang, via une construction appelée *lifting* de codes.

Le but de cette section est de présenter de façon générale la notion de distance, et particulièrement celles qui peuvent être définies sur des espaces de matrices. Nous nous attarderons sur la métrique rang, dont nous donnerons une définition alternative, basée sur les θ-polynômes. Nous montrerons à quelle condition ces deux définitions sont équivalentes.

2.2.1 Métriques sur des matrices

Définition 7 (distance)**.** *Une distance sur un espace X est une application*

$$\begin{array}{rccc} d: & X \times X & \to & \mathbb{R} \\ & (x, y) & \mapsto & d(x, y) \end{array}$$

vérifiant, pour tous $a, b, c \in X^3$:

- $d(a, b) = 0 \Leftrightarrow a = b$,

- $d(a,b) = d(b,a)$,
- $d(a,c) \leqslant d(a,b) + d(b,c)$.

Les métriques sont généralement induites via la relation $d(x,y) = w(x-y)$, où w désigne un poids.

Définition 8 (poids)**.** *Un poids sur X est une application*

$$\begin{aligned} d : X &\to \mathbb{R} \\ x &\mapsto w(x) \end{aligned}$$

vérifiant, pour tous $a, b \in X^2$:
- $w(a) = 0 \Leftrightarrow a = 0$,
- $w(a) = w(-a)$,
- $w(a+b) \leqslant w(a) + w(b)$.

Le poids de Hamming est un exemple, mais il ne convient pas à un espace de matrices. En effet, ce poids ne tient pas compte de la répartition des coefficients en lignes et colonnes. La métrique obtenue n'est donc pas adaptée à des codes se basant sur les propriétés des matrices.

La métrique *term-rank* a été définie par R. Roth [Rot91] pour répondre à une situation précise. Les effacements qu'il voulait corriger seront présentés dans la section 3.3.

Définition 9 (poids *term-rank*)**.** *Un recouvrement d'une matrice $M = (m_{i,j}) \in \mathcal{M}_{m\times n}(K)$ est une paire d'ensembles (L, C), où $L \subset \{1, \ldots, m\}$ et $C \subset \{1, \ldots, n\}$, telle que*

$$m_{i,j} \neq 0 \Rightarrow (i \in L) \text{ ou } (j \in C).$$

Le poids term-rank $w_{tr}(M)$ de cette matrice est le plus petit nombre de lignes et colonnes nécessaires pour recouvrir les coefficients non nuls de M :

$$w_{tr}(M) = \min_{(L,C)} \left(\mathrm{card}(L) + \mathrm{card}(C)\right).$$

Définie par des considérations pratiques, cette distance n'est pas aisée à manipuler. La métrique rang lui est généralement préférée.

Définition 10 (poids rang)**.** *Le poids rang $w_r(M)$ d'une matrice M est défini par*

$$w_r(M) = \mathrm{rang}(M).$$

Ces deux métriques sont liées par la relation suivante.

Proposition 11. *Pour toute matrice $M \in \mathcal{M}_{m\times n}(K)$, nous avons :*

$$w_r(M) \leq w_{tr}(M)$$

Ainsi, un code permettant de corriger t erreurs en métrique rang permet de corriger toutes les erreurs de poids inférieur ou égal à t en métrique *term-rank*, mais aussi des erreurs de poids *term-rank* supérieur. La métrique *term-rank* nous servira uniquement à mesurer des matrices pour lesquelles le rang n'est pas défini, comme les matrices d'effacement que nous verrons dans la section 3.3.

Remarque 5. *Toute norme, quelle que soit la valeur absolue sous-jacente, est un poids. Néanmoins, le comportement d'une distance définie à partir d'une norme dépend de cette valeur absolue. Ainsi, la distance de Frobenius, basée sur la valeur absolue usuelle de Q, est une distance non bornée dont les boules sont des boules "sphériques", alors que les distances de Hamming, rang et term-rank, provenant de la valeur absolue triviale ($\forall x \neq 0$, $|x| = 1$), sont des distances bornées, dont les boules sont des sous-espaces non bornés.*

2.2.2 Métrique rang sur L^n

Soit $K \hookrightarrow L$ une extension de corps de degré $[L : K] = m$ et soit $\mathcal{B} = (b_1, \ldots, b_m)$ une K-base de L en tant que K-espace vectoriel. Tout élément $x \in L$ se décompose sous la forme

$$x = \sum_{i=1}^{m} x_i b_i. \tag{2.2}$$

Ainsi, un élément de L peut être vu comme un vecteur de K^m et un vecteur ligne de L^n peut être vu comme une matrice à m lignes et n colonnes sur K.

Définition 11. *Soit $K \hookrightarrow L$ une extension de corps de degré $[L : K] = m$ et soit $\mathcal{B} = (b_1, \ldots, b_m)$ une K-base de L en tant que K-espace vectoriel. Par abus de langage, nous noterons $\mathrm{ext}_{\mathcal{B}}$ les deux bijections suivantes.*

$$\begin{array}{cccc}
\mathrm{ext}_{\mathcal{B}} : & L & \longrightarrow & K^m \\
 & x & \longmapsto & \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \\
\mathrm{ext}_{\mathcal{B}} : & L^n & \longrightarrow & \mathcal{M}_{m\times n}(K) \\
 & (x_1, \ldots, x_n) & \longmapsto & \begin{pmatrix} x_{1,1} & \cdots & x_{n,1} \\ \vdots & \ddots & \vdots \\ x_{1,m} & \cdots & x_{n,m} \end{pmatrix}
\end{array}$$

Cette bijection est à la base de la métrique rang.

Définition 12 (poids basé sur le rang)**.** *Soient $K \hookrightarrow L$ une extension de corps de degré $[L : K] = m$, $\mathcal{B}$ une K-base de L, et $x = (x_1, \ldots, x_n) \in L^n$. Le poids $\mathrm{w}_{\mathcal{B}}$ de x est défini par*

$$\begin{aligned}
\mathrm{w}_{\mathcal{B}}(x) &= \mathrm{rang}_K\left(\mathrm{ext}_{\mathcal{B}}(x_1, \ldots, x_n)\right) \\
&= \mathrm{rang}_K \begin{pmatrix} x_{1,1} & \cdots & x_{n,1} \\ \vdots & \ddots & \vdots \\ x_{1,m} & \cdots & x_{n,m} \end{pmatrix}
\end{aligned}$$

Ce poids ne dépend pas du choix de la base $\mathcal{B}$. En effet, le poids dans une autre base se déduit par multiplication par une matrice de changement de base. Il s'agit d'une bijection, ce qui préserve le rang.

Définition 13 (poids basé sur les θ-polynômes)**.** *Soient $K \hookrightarrow L$ une extension de corps de degré $[L : K] = m$, $\theta \in \mathrm{Aut}_K(L)$ et $x = (x_1, \ldots, x_n) \in L^n$. Le poids $\mathrm{w}_{\mathcal{A}}$ de x est défini par*

$$\mathrm{w}_{\mathcal{A}}(x) = \deg(\mathcal{A}_{\langle x_1, \ldots, x_n \rangle}).$$

Théorème 12. *Nous avons la relation suivante entre les poids définis précédemment :*

$$\forall x \in L^n, \mathrm{w}_{\mathcal{A}}(x) \leqslant \mathrm{w}_{\mathcal{B}}(x).$$

De plus, si $L^\theta = K$, où L^θ désigne le sous corps de L laissé stable par θ, cette relation devient :

$$\forall x \in L^n, \mathrm{w}_{\mathcal{A}}(x) = \mathrm{w}_{\mathcal{B}}(x).$$

Pour démontrer ce théorème, nous allons introduire deux poids intermédiaires.

Définition 14. *Soient $K \hookrightarrow L$ une extension de corps de degré $[L : K] = m$, $\theta \in \mathrm{Aut}_K(L)$ d'ordre s et $x = (x_1, \ldots, x_n) \in L^n$. Nous définissons les poids*

$$\mathrm{w}_{\theta,\mathrm{K}}(x) = \mathrm{rang}_K \begin{pmatrix} \theta^0(x_1) & \cdots & \theta^0(x_n) \\ \vdots & \ddots & \vdots \\ \theta^{s-1}(x_1) & \cdots & \theta^{s-1}(x_n) \end{pmatrix}$$

$$\mathrm{w}_{\theta,\mathrm{L}}(x) = \mathrm{rang}_L \begin{pmatrix} \theta^0(x_1) & \cdots & \theta^0(x_n) \\ \vdots & \ddots & \vdots \\ \theta^{s-1}(x_1) & \cdots & \theta^{s-1}(x_n) \end{pmatrix}$$

où rang_K et rang_L désignent le nombre de colonnes linéairement indépendantes d'une matrice, sur K et L respectivement.

Remarque 6. *Bien que ces quantités soient appelées rangs, seul rang_L est égal au rang (au sens usuel).*

Notation 1. *Dans un soucis de concision, nous utiliserons les notations suivantes :*

$$X_\theta = \begin{pmatrix} \theta^0(x_1) & \cdots & \theta^0(x_n) \\ \vdots & \ddots & \vdots \\ \theta^s(x_1) & \cdots & \theta^s(x_n) \end{pmatrix}$$

et

$$X_{\mathcal{B}} = \mathrm{ext}_{\mathcal{B}}(x_1, \ldots, x_n).$$

Nous noterons $C_{\theta,i}$ et $C_{\mathcal{B},i}$ $(1 \leqslant i \leqslant n)$ leurs colonnes, et $L_{\theta,i}$ $(0 \leqslant i \leqslant s-1)$ la $i+1$-ème ligne de X_θ.

Lemme 13. *Les poids $\mathrm{w}_{\mathcal{A}}$ et $\mathrm{w}_{\theta,\mathrm{L}}$ sont égaux.*

$$\forall x \in L^n, \mathrm{w}_{\mathcal{A}}(x) = \mathrm{w}_{\theta,\mathrm{L}}(x).$$

Démonstration. Commençons par montrer que $\mathrm{w}_{\theta,\mathrm{L}} \leqslant \mathrm{w}_{\mathcal{A}}$.
Soit $x \in L^n$ un élément de poids $w = \mathrm{w}_{\mathcal{A}}(x)$ et notons $\mathcal{A} = \sum_{i=0}^{w} a_i X^i$ son θ-polynôme annulateur (unitaire). Nous avons alors $\theta^w(x_j) = -\sum_{i=0}^{w} a_i \theta^i(x_j)$, ce qui nous permet d'écrire

$$L_{\theta,w} = -\sum_{i=0}^{w} a_i L_{\theta,i}.$$

La $(w+1)$-ème ligne de X_θ est donc une combinaison linéaire des précédentes. En appliquant θ à cette relation, nous obtenons une combinaison linéaire exprimant chacune des lignes situées en dessous de $L_{\theta,w}$ en fonction des w premières. La matrice X_θ est donc de rang au plus w.
Montrons maintenant que $\mathrm{w}_{\mathcal{A}} \leqslant \mathrm{w}_{\theta,\mathrm{L}}$.
Soit $x \in L^n$ un élément de poids $w = \mathrm{w}_{\theta,\mathrm{L}}(x)$. Si les w premières lignes sont linéairement indépendantes, nous pouvons exprimer la $(w+1)$-ème ligne comme combinaison linéaire des précédentes. Si ce n'est pas le cas, alors il y a une combinaison linéaire parmi ces w lignes. En appliquant θ, nous pouvons faire descendre cette relation dans les lignes de la matrice. Dans les deux cas, nous exprimons $(w+1)$-ème ligne en fonction des précédentes. Cette relation fournit un polynôme annulateur des x_i, de degré w. Leur polynôme annulateur de degré minimal est de degré au plus w. □

Lemme 14. *Les poids* $\mathrm{w}_{\theta,\mathrm{K}}$ *et* $\mathrm{w}_{\mathcal{B}}$ *sont égaux.*

$$\forall x \in L^n, \mathrm{w}_{\theta,\mathrm{K}}(x) = \mathrm{w}_{\mathcal{B}}(x).$$

Ce lemme et le suivant, qui consistent à comparer le nombre de colonnes indépendantes dans deux matrices, se prouvent en utilisant un même raisonnement. Considérons une combinaison linéaire entre des colonnes d'une des matrices. Nous montrons ensuite que cette relation en induit une dans les colonnes de l'autre matrice. Le nombre de colonnes libres dans la seconde matrice est donc au plus le nombre de colonnes libres dans la première.

Remarque 7. *L'ordre des* x_i *n'a pas d'importance, aussi, afin de simplifier les notations, et sans perte de généralité, nous supposerons que les* r *premières colonnes d'une matrice sont libres,* r *étant le rang de la matrice. De plus, quand nous considérerons une combinaison linéaire, nous écrirons, toujours sans perte de généralité, que la dernière colonne est combinaison linéaire des* r *premières.*

Démonstration. Commençons par montrer que $\mathrm{w}_{\theta,\mathrm{K}} \leqslant \mathrm{w}_{\mathcal{B}}$.
Soit $x \in L^n$ un élément de poids $w = \mathrm{w}_{\mathcal{B}}(x)$, et considérons une combinaison linéaire entre les colonnes de $X_{\mathcal{B}}$. Sans perte de généralité, nous pouvons supposer qu'il s'agit de la suivante :

$$\begin{pmatrix} x_{n,1} \\ \vdots \\ x_{n,m} \end{pmatrix} = \lambda_1 \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{1,m} \end{pmatrix} + \cdots + \lambda_w \begin{pmatrix} x_{w,1} \\ \vdots \\ x_{w,m} \end{pmatrix}, \lambda_i \in K.$$

Cela mène à la relation entre les x_i :

$$x_n = \lambda_1 x_1 + \cdots + \lambda_w x_w,$$

et les λ_i étant des éléments de K, cette relation est préservée par l'action de θ^k :

$$\theta^k(x_n) = \lambda_1 \theta^k(x_1) + \cdots + \lambda_w \theta^k(x_w).$$

Nous obtenons alors

$$\begin{pmatrix} x_n \\ \vdots \\ \theta^{s-1}(x_n) \end{pmatrix} = \lambda_1 \begin{pmatrix} x_1 \\ \vdots \\ \theta^{s-1}(x_1) \end{pmatrix} + \cdots + \lambda_w \begin{pmatrix} x_w \\ \vdots \\ \theta^{s-1}(x_w) \end{pmatrix}.$$

Ainsi, toute combinaison linéaire entre les colonnes de $X_{\mathcal{B}}$ entraine une combinaison linéaire entre celles de X_θ, et donc X_θ ne peut avoir plus de colonnes libres que $X_{\mathcal{B}}$. Nous avons donc bien $\mathrm{w}_{\theta,\mathrm{K}}(x) \leqslant \mathrm{w}_{\mathcal{B}}(x)$.
Montrons maintenant que $\mathrm{w}_{\mathcal{B}} \leqslant \mathrm{w}_{\theta,\mathrm{K}}$.
Soit $x \in L^n$ un élément de poids $w = \mathrm{w}_{\theta,\mathrm{K}}(x)$. Considérons une combinaison linéaire, par exemple

$$\begin{pmatrix} x_n \\ \vdots \\ \theta^{s-1}(x_n) \end{pmatrix} = \lambda_1 \begin{pmatrix} x_1 \\ \vdots \\ \theta^{s-1}(x_1) \end{pmatrix} + \cdots + \lambda_w \begin{pmatrix} x_w \\ \vdots \\ \theta^{s-1}(x_w) \end{pmatrix}, \lambda_i \in K.$$

En particulier, cette relation est valable pour le premier coefficient de chaque colonne :

$$x_n = \sum_{i=1}^{w} \lambda_i x_i.$$

En décomposant les x_i dans la base $\mathcal{B}$, on obtient

$$\begin{pmatrix} x_{n,1} \\ \vdots \\ x_{n,s} \end{pmatrix} = \lambda_1 \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{1,s} \end{pmatrix} + \cdots + \lambda_w \begin{pmatrix} x_{w,1} \\ \vdots \\ x_{w,s} \end{pmatrix},$$

ce qui est précisément une combinaison linéaire entre les colonnes de $X_{\mathcal{B}}$. Ainsi, $X_{\mathcal{B}}$ ne peut avoir plus de colonnes libres que X_θ, Nous avons donc bien $\mathrm{w}_{\mathcal{B}}(x) = \mathrm{rang}_K(X_{\mathcal{B}}) \leqslant w$. □

Lemme 15. *Les poids* $\mathrm{w}_{\theta,\mathrm{L}}$ *et* $\mathrm{w}_{\theta,\mathrm{K}}$ *sont reliés par la formule suivante.*

$$\forall x \in L^n, \mathrm{w}_{\theta,\mathrm{L}}(x) \leqslant \mathrm{w}_{\theta,\mathrm{K}}(x),$$

avec égalité si $L^\theta = K$.

Démonstration. Commençons par montrer que $\mathrm{w}_{\theta,\mathrm{L}}(x) \leqslant \mathrm{w}_{\theta,\mathrm{K}}(x)$.
Il est clair qu'une combinaison linéaire à coefficients dans K est une combinaison linéaire à coefficients dans L, donc X_θ ne peut pas avoir plus de colonnes libres sur L que sur K, d'où $\mathrm{w}_{\theta,\mathrm{L}}(x) \leqslant \mathrm{w}_{\theta,\mathrm{K}}(x)$.
Montrons maintenant que $\mathrm{w}_{\theta,\mathrm{K}}(x) \leqslant \mathrm{w}_{\theta,\mathrm{L}}(x)$ si $\theta = K$.
Soit $x \in L^n$, notons $w = \mathrm{w}_{\theta,\mathrm{L}}(x) = \mathrm{rang}_L(X_\theta)$ son poids et considérons une combinaison linéaire à coefficients dans L entre les colonnes de X_θ. Cette relation nous donne les équations suivantes :

$$\left\{ \begin{array}{ccccccc} x_n & = & \lambda_1 x_1 & + & \cdots & + & \lambda_r x_r \\ \vdots & & & & & & \vdots \\ \theta^{s-1}(x_n) & = & \lambda_1 \theta^{s-1}(x_1) & + & \cdots & + & \lambda_r \theta^{s-1}(x_r) \end{array} \right. , \lambda_i \in L.$$

En appliquant θ sur chacune des lignes, nous obtenons :

$$\left\{ \begin{array}{ccccccc} \theta^1(x_n) & = & \theta^1(\lambda_1)\theta^1(x_1) & + & \cdots & + & \theta^1(\lambda_1)\theta^1(x_r) \\ \vdots & & & & & & \vdots \\ \theta^{s-1}(x_n) & = & \theta^1(\lambda_1)\theta^s(x_1) & + & \cdots & + & \theta^1(\lambda_r)\theta^s(x_r) \end{array} \right. .$$

Puisque s est l'ordre de θ, nous ramenons la dernière équation en première place et nous obtenons :

$$\left\{ \begin{array}{ccccccc} x_n & = & \theta^1(\lambda_1)x_1 & + & \cdots & + & \theta^1(\lambda_1)x_r \\ \vdots & & & & & & \vdots \\ \theta^{s-1}(x_n) & = & \theta^1(\lambda_1)\theta^{s-1}(x_1) & + & \cdots & + & \theta^1(\lambda_r)\theta^{s-1}(x_r) \end{array} \right. .$$

Les colonnes $C_{\theta,1}, \ldots, C_{\theta,w}$ étant libres, elles forment une base de l'espace qu'elles engendrent, et $C_{\theta,n}$ s'exprime de façon unique dans cette base. Nous obtenons alors que $\lambda_i = \theta(\lambda_i)$. Les λ_i sont donc des éléments de $L^\theta = \{x \in L : \theta(x) = x\}$. Si ce corps fixe est exactement K, alors les λ_i sont dans K et nous avons bien $\mathrm{w}_{\theta,\mathrm{K}}(x) \leqslant \mathrm{w}_{\theta,\mathrm{L}}(x)$. □

La distance rang entre deux éléments x et y de L^n est définie par $d_r(x,y) = \mathrm{w}_{\mathcal{B}}(x-y)$. Sa définition alternative $\mathrm{w}_{\mathcal{A}}(x-y)$, lorsqu'elle est équivalente, permet néanmoins de fournir des preuves plus naturelles aux propriétés des codes de Gabidulin (voir chapitre 3).

2.3 Cadre de travail

Nous avons vu précédemment deux théorèmes, (6 et 12), dont nous aurons besoin pour prouver les propriétés des codes de Gabidulin généralisés. Nous commencerons par regarder un exemple où ces propriétés ne sont pas valables, et remarquerons qu'elles sont liées l'une à l'autre. Nous verrons ensuite que ces deux conditions sont équivalentes, et préciserons dans quels types d'extensions elles sont vérifiées. Nous fixerons ainsi un cadre de travail pour la suite de ce document. Enfin, nous donnerons plusieurs exemples de telles extensions. Nous présenterons aussi bien des corps de nombres que des corps de fractions [Aug].

2.3.1 Exemples

Considérons l'extension

$$K = \mathbb{Q} \hookrightarrow \mathbb{Q}[Y]/(Y^8 + 1) = \mathbb{Q}[\alpha] = L.$$

Un des intérêts de cette extension est que toutes les racines de $Y^8 + 1$ sont explicites et peuvent s'écrire en fonction de α. L'automorphisme θ se définit en envoyant α sur une racine de $Y^8 + 1$. Puisque toutes les racines sont explicites, nous obtenons explicitement 8 K-automorphismes. Choisissons par exemple celui défini par $\theta : \alpha \mapsto \alpha^3$.

Le polynôme caractéristique de θ, vu comme application linéaire, est $(X^4 - 1)^2$. Ce polynôme a un facteur carré, donc on peut trouver des θ-polynômes dont le degré ne majore pas la dimension de l'espace des racines. Par exemple, les racines du θ-polynôme $X - 1$ forment un sous-K-espace vectoriel de dimension 2, engendré par 1 et $\alpha^2 + \alpha^6$.

Les racines de ce polynôme sont précisément les éléments de L^θ, ainsi, ce corps intermédiaire n'est pas égal à K, ce qui implique que les métriques $\mathrm{w}_{\mathcal{A}}$ et $\mathrm{w}_{\mathcal{B}}$ ne sont pas équivalentes.

Considérons maintenant le corps

$$K = \mathbb{Q}[j] = \mathbb{Q}[X]/(X^2 + X + 1),$$

qui contient toutes les racines sixièmes de l'unité $\{1, j, j^2, -1, -j, -j^2\}$, et l'extension

$$L = K[\alpha] = K[Y]/(Y^6 - 2),$$

obtenue en ajoutant une racine sixième de 2 au corps K. (Cela forme une extension de Kummer, voir sous-section 2.3.2) Dans ce cas encore, toutes les racines de $Y^6 - 2$ sont des éléments explicites de K, ce qui permet de connaitre tous les K-automorphismes.

Considérons l'automorphisme défini par : $\theta_1 : \alpha \mapsto j\alpha$. Comme précédemment, nous calculons le polynôme caractéristique de θ, qui vaut $(X^3 - 1)^2$, et qui admet un facteur carré. Ainsi, nous calculons l'espace des racines de $X - 1$, qui est le sous-K-espace vectoriel engendré par 1 et α^3. Les deux métriques sont donc distinctes, comme nous pouvons le voir sur l'exemple suivant :

$$\mathrm{w}_{\mathcal{A}}(1, \alpha, \alpha^3, \alpha^4) = 2 \neq \mathrm{w}_{\mathcal{B}}(1, \alpha, \alpha^3, \alpha^4) = 4.$$

En revanche, pour cette extension, il est possible de choisir un autre automorphisme. Par exemple, θ_2 défini par $\alpha \mapsto -j^2\alpha$ vérifie les conditions des deux théorèmes.

Nous allons voir que ces deux conditions sont en fait équivalentes, et que l'existence d'un automorphisme approprié dépend de l'extension $K \hookrightarrow L$.

2.3.2 Équivalence et extensions cycliques

Commençons par établir l'équivalence des deux conditions.

Théorème 16. *Soit $K \hookrightarrow L$ une extension de corps, et soit $\theta \in \mathrm{Aut}_K(L)$ un K-automorphisme. Alors les deux assertions suivantes sont équivalentes :*

1. *Le polynôme caractéristique de θ, vu comme application K-linéaire, est sans facteur carré.*
2. *Le sous corps de L stable par θ, noté L^θ, est exactement K.*

Démonstration. La première assertion revient à dire que toutes les valeurs propres de θ sont de multiplicité au plus 1. La seconde, quant à elle, revient à dire que 1 doit être de multiplicité 1 comme valeur propre. En effet, les éléments de L^θ sont solution de l'équation $\theta(x) - x = 0$. Tous les éléments de K étant solution, 1 est valeur propre de θ. Dire que cet ensemble L^θ est exactement K revient à dire qu'il est de K-dimension 1, c'est-à-dire que la valeur propre 1 doit être simple.

Il est alors clair que si toutes les valeurs propres sont de multiplicité 1, alors la valeur propre 1 est de multiplicité 1.

Montrons maintenant par contraposée que si 1 est de multiplicité 1, alors toutes les valeurs propres le sont. Pour cela, supposons qu'une valeur propre λ est multiple. Il existe alors deux éléments $u, v \in L$ associés à λ et non colinéaires. Leur quotient $\frac{u}{v} \notin K$ est associé à la valeur propre 1, qui est alors valeur propre multiple. □

Regardons maintenant quelles extensions permettent de trouver un automorphisme vérifiant ces assertions.

Théorème 17. *Soit $K \hookrightarrow L$ une extension de corps, et soit $\theta \in \mathrm{Aut}_K(L)$ un K-automorphisme. Alors les trois assertions suivantes sont équivalentes :*

1. *Le polynôme caractéristique de θ, vu comme application K-linéaire, est sans facteur carré.*
2. *Le sous corps de L stable par θ, noté L^θ, est exactement K.*
3. *L est une extension galoisienne cyclique de K, et θ est un générateur du groupe de Galois.*

Démonstration. Commençons par montrer que $(1, 2)$ impliquent que l'extension est galoisienne. Supposons le polynôme caractéristique de θ sans facteur carré. La matrice de θ, dans une K-base de L, est alors $\mathcal{M}_\theta = P\,\mathrm{Diag}(\lambda_1, \ldots, \lambda_m)P^{-1}$, et celle de θ^m est $\mathcal{M}_{\theta^m} = P\,\mathrm{Diag}(\lambda_1^m, \ldots, \lambda_m^m)P^{-1}$. Puisque θ est d'ordre fini m, les λ_i sont des racines m-ièmes de l'unité, de plus, elles sont distinctes. Ce sont donc toutes les racines m-ièmes de l'unité, et donc les $\theta^i, 0 \leqslant i \leqslant m-1$ sont m automorphismes distincts. Puisqu'on a exactement $[L : K] = m$ automorphismes, l'extension est galoisienne et θ engendre le groupe de Galois.

Réciproquement, supposons l'extension galoisienne. Par la correspondance de Galois (voir [Goz97], [Ste15] ou [Cox11]), nous obtenons

$$\begin{array}{ccc} L & \longleftrightarrow & \{Id\} \\ L^\theta & \longleftrightarrow & \langle\theta\rangle \\ K & \longleftrightarrow & \mathrm{Gal}(K \hookrightarrow L) \end{array}$$

ce qui montre que les assertions $(1, 2)$ sont vérifiées dès que θ est générateur du groupe de Galois. □

Ainsi, le bon cadre pour définir un code de Gabidulin généralisé est constitué d'une extension $K \hookrightarrow L$ galoisienne cyclique et d'un générateur θ de son groupe de Galois.

Voyons maintenant quelques exemples de telles extensions.

2.3.3 Quelques extensions cycliques

Corps finis L'exemple le plus naturel d'extension cyclique, tout au moins pour la construction de codes correcteurs, est le cas des corps finis.

Toute extension de corps finis $\mathbb{F}_q \hookrightarrow \mathbb{F}_{q^m}$ est galoisienne cyclique, et l'automorphisme considéré est généralement le Frobenius.

Extensions cyclotomiques (corps de nombres) Soit m un entier positif non nul. On appelle extension cyclotomique (see [Neu99, §I.10]) de $\mathbb{Q}$ le corps de nombres $\mathbb{Q}(\zeta)$ où ζ désigne une racine m-ième de l'unité. C'est une extension de degré $\varphi(m)$, où φ désigne la fonction d'Euler, qui contient toutes les racines m-ièmes de l'unité. Le groupe de Galois d'une telle extension est $(\mathbb{Z}/m\mathbb{Z})^*$. En particulier, l'extension est cyclique si m est premier. Les automorphismes sont de la forme $\theta_i : \alpha \mapsto \alpha^i$, et les générateurs sont les θ_i pour lesquels i est premier avec m.

Extensions de Kummer (corps de nombres) Les extensions de Kummer (see [Neu99, §IV.3]) forment une autre famille d'extensions cycliques. Soit m un entier naturel positif, K un corps contenant toutes les racines m-ièmes de l'unité (par exemple, K est un corps cyclotomique) et notons ζ une racine primitive de l'unité. Une extension de Kummer est de la forme :

$$K[\alpha] = K[Y]/(Y^m - a), a \in K.$$

Si a n'a aucune racine d-ième dans K, d étant un diviseur de m, alors l'extension est galoisienne cyclique. Les automorphismes sont de la forme $\theta_i : \alpha \mapsto \zeta^i\alpha$, et les générateurs sont les θ_i pour lesquels ζ^i est une racine primitive.

Extensions de Kummer (corps de fonctions) Les extensions de Kummer ne sont pas spécifiques aux corps de nombres. La famille suivante, sur des corps de fractions, est également une extension de Kummer (see [Sti09, §III.7]). Considérons le corps $K = \mathbb{F}_q(x)$ et l'entier $m = q - 1$. On ajoute à K une racine m-ième de x de la façon suivante :

$$K = \mathbb{F}_q(x) \hookrightarrow K[\alpha] = K[Y]/(Y^m - x).$$

Les automorphismes sont de la forme $\theta_i : \alpha \mapsto \zeta_i\alpha$, où les $\zeta_i \in \mathbb{F}_q$ sont des racines m-ièmes de l'unité. Les générateurs sont les θ_i pour lesquels ζ_i est une racine primitive.

Extensions d'Artin-Schreier (corps de fonctions) Les extensions d'Artin-Schreier (see [Sti09, §III.7]) sont une autre famille d'extensions cycliques sur des corps de fonctions. Soit p un nombre premier. Les extensions d'Artin-Schreier sont de la forme :

$$\mathbb{F}_p(x) \hookrightarrow \mathbb{F}_p(x)[\alpha] = \mathbb{F}_p(x)[Y]/(Y^p - Y - x).$$

L'automorphisme $\theta : \alpha \mapsto \alpha + 1$ est alors générateur du groupe de Galois de l'extension.

Chapitre 3

Codes de Gabidulin généralisés

Les codes de Gabidulin [Gab85] ont été découverts par E. M. Gabidulin en 1985. Dans son article, il en décrit la construction sur des corps finis ainsi que leurs propriétés. Par construction, ces codes sont proches des codes de Reed-Solomon [RS60], puisqu'ils consistent en l'évaluation d'un θ-polynôme sur un support bien choisi. Ils héritent ainsi de certaines propriétés des codes de Reed-Solomon, telles que l'optimalité (au sens de Singleton [Sin64]) et l'existence d'algorithmes de décodage efficaces ([BW86], [Gao03]). Ces deux propriétés sont à l'origine de l'intérêt qui leur a été porté ces dernières années, notamment dans le domaine du codage de réseau. Cela demande des codes spécifiques, basés sur une autre métrique, la métrique sous-espace (les mots de code sont des sous-espaces vectoriels), et de tels codes peuvent être construits à partir de codes de Gabidulin (codes liftés [SKK08]).

Le but de ce chapitre est de généraliser la définition des codes de Gabidulin à des corps infinis. Afin de garantir des propriétés similaires à leurs homologues sur les corps finis, nous serons contraints de nous restreindre à des extensions galoisiennes cycliques (voir théorème 17). En effet, nous aurons besoin de maitriser la dimension de l'espace des racines d'un θ-polynôme et nous utiliserons la définition alternative $\mathrm{w}_{\mathcal{A}}$ de la métrique rang basée sur les θ-polynômes (voir définition 13).

Dans la première section, nous définirons les codes de Gabidulin généralisés et présenterons leurs propriétés. Les preuves sont généralement nouvelles, s'affranchissant ainsi des hypothèses de finitude. Nous verrons également que les mots du code peuvent être écrits sous forme de vecteurs ou sous forme de matrices. Dans la suite, nous utiliserons les deux écritures. Nous aborderons ensuite trois modèles de transmission.

La deuxième section est consacrée au cas où seulement des erreurs se produisent. Après avoir présenté le modèle d'erreurs et le problème de Décodage, nous introduirons les problèmes de Reconstruction et verrons comment le décodage s'y ramène. L'algorithme de Décodage reposera sur un algorithme de Reconstruction, utilisé comme une boite noire jusqu'au prochain chapitre.

Nous aborderons dans les troisième et quatrième sections deux modèles dans lesquels se produisent des effacements en plus des erreurs. Nous verrons alors qu'il est possible de gérer ces effacements pour se ramener à un code de Gabidulin avec seulement des erreurs. Les deux modèles diffèrent par la définition des effacements.

La généralisation des codes de Gabidulin ainsi que la méthode de décodage (pour le modèle sans effacements) ont été publiés et présentés en juillet 2013, lors de la conférence ISIT (*International Symphosium on Information Theory*), puis aux journées C2 (Codage et Cryptographie) en mars

2014. Les résultats concernant les effacements ont été évoqués à la conférence ACN (*Algebra, Codes and Network*) en juin 2014 ainsi qu'aux journées C2 en octobre 2015.

3.1 Définition

Définition 15 (Code de Gabidulin généralisé)**.** *Soient $K \hookrightarrow L$ une extension de corps de degré fini $[L : K] = m$ et $\theta \in \mathrm{Aut}_K(L)$ un générateur du groupe de Galois. Soient $k \leqslant n \leqslant m$ des entiers*[1] *et soient $g_1, \cdots, g_n$ des éléments K-linéairement indépendants de L. Ces éléments forment le support du code, noté $g = (g_1, \ldots, g_n)$. Le code de Gabidulin généralisé de support g est l'ensemble :*

$$\mathrm{Gab}_{\theta,k}(g) = \{(\mathcal{L}_f(g_1), \ldots, \mathcal{L}_f(g_n)) : f \in L[X;\theta], \deg(f) < k\}.$$

Remarque 8. *Dans la suite,* code de Gabidulin *fera référence à un code de Gabidulin généralisé, ce qui inclut le cas des codes de Gabidulin originaux. Ces derniers seront désignés spécifiquement par* codes de Gabidulin finis.

Nous obtenons un code de dimension k et de longueur n. Ce code est linéaire, et admet pour matrice génératrice

$$G = \begin{pmatrix} g_1 & \cdots & g_n \\ \theta(g_1) & \cdots & \theta(g_n) \\ \vdots & \ddots & \vdots \\ \theta^{k-1}(g_1) & \cdots & \theta^{k-1}(g_n) \end{pmatrix}.$$

En effet, le produit d'un vecteur $(f_0, \ldots, f_{k-1})$ par la i-ème colonne de cette matrice correspond à l'évaluation du θ-polynôme $f = f_0 + \cdots + f_{k-1}X^{k-1}$ en g_i.

Exemple 10. *Nous allons construire un code pour illustrer les propos tout au long de ce chapitre. Ce code sera de longueur $n = 4$ et de dimension $k = 2$. Nous allons le construire sur une extension de Kummer de degré $m = 4$.*

L'extension est $\mathbb{Q}[i] \hookrightarrow \mathbb{Q}[i][\alpha] = \mathbb{Q}[i][Y]/(Y^4 - i)$ et l'automorphisme est défini par $\theta : \alpha \mapsto i\alpha$. Cette extension dispose d'une base canonique $\mathcal{B} = (1, \alpha, \alpha^2, \alpha^3)$, que nous pouvons reprendre pour définir le support $g = (1, \alpha, \alpha^2, \alpha^3)$.

Un mot d'information est un θ-polynôme de degré strictement inférieur à 2, comme par exemple

$$f = (1 + i\alpha^2 - \alpha^3) + (\alpha^2 - i\alpha^3)X.$$

Nous avons alors

$$\begin{aligned} \mathcal{L}_f(1) &= 1 + (1+i)\alpha^2 - (1+i)\alpha^3, \\ \mathcal{L}_f(\alpha) &= \alpha^2 + 2i\alpha^3, \\ \mathcal{L}_f(\alpha^2) &= -(1+i) - (1+i)\alpha + \alpha^2, \\ \mathcal{L}_f(\alpha^3) &= -2i\alpha^2 + \alpha^3. \end{aligned}$$

Le mot du code correspondant est donc

$$(1 + (1+i)\alpha^2 - (1+i)\alpha^3,\ \alpha^2 + 2i\alpha^3,\ -(1+i) - (1+i)\alpha + \alpha^2,\ -2i\alpha^2 + \alpha^3).$$

1. Le cas particulier $n = k$ correspond à l'absence de codage. Nous rencontrerons ce cas lors de la gestion des effacements.

Distance minimale

Théorème 18 (borne de Singleton)**.** *Soit $\mathcal{C}$ un code de longueur n et de dimension k en métrique rang. Alors sa distance minimale d vérifie la majoration suivante, appelée borne de Singleton.*

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

Définition 16 (codes MRD)**.** *Un code $\mathcal{C}$ de paramètres $[n, k, d]$ vérifiant l'inégalité de Singleton est appelé code MRD (Maximum Rank Distance code).*

Proposition 19. *La distance minimale d d'un code de gabidulin généralisé de longueur n et de dimension k est*

$$d = n - k + 1.$$

Corollaire 20. *Les codes de gabidulin généralisés sont des codes MRD.*

Démonstration. Le code étant linéaire, il suffit de calculer le poids minimal pour connaitre la distance minimale. Soit $c = (c_1, \ldots, c_n)$ un mot non nul du code, et soit $w = \mathrm{w}_{\mathcal{A}}(g)$ son poids. Alors, il existe un θ-polynôme non nul U, de degré w (théorème 8), qui s'annule sur les c_i. Or, c étant un mot du code, les c_i sont de la forme $c_i = \mathcal{L}_f(g_i)$, où f est un θ-polynôme de degré au plus $k - 1$. Nous avons alors

$$\forall i \in [[1; n]],\ \mathcal{L}_{U \cdot f}(g_i) = \mathcal{L}_U(c_i) = 0.$$

Le θ-polynôme non nul $U \cdot f$ s'annule en n valeurs, il est donc de degré au moins n (corollaire 7). Par ailleurs, il est de degré au plus $w + k - 1$ par construction. Nous avons alors

$$n \leqslant w + k - 1.$$

Tout mot du code est de poids au moins $n - k + 1$, il en est de même pour la distance minimale. (Nous utilisons le théorème 12 pour passer de la métrique $\mathrm{w}_{\mathcal{A}}$ à la métrique rang usuelle $\mathrm{w}_{\mathcal{B}}$.) Or, celle-ci est également minorée par $n - k + 1$ (borne de Singleton). Nous avons donc $d = n - k + 1$, et les codes de Gabidulin généralisés sont des codes MRD. □

Matrice de contrôle et code dual

Théorème 21. *Soit $\mathrm{Gab}_{\theta,k}(g)$ un code de Gabidulin généralisé de paramètres $[n, k, d]$ et de support $g = (g_1, \ldots, g_n)$. Le code admet une matrice de contrôle de la forme*

$$H = \begin{pmatrix} h_1 & \cdots & h_n \\ \theta(h_1) & \cdots & \theta(h_n) \\ \vdots & \ddots & \vdots \\ \theta^{n-k-1}(h_1) & \cdots & \theta^{n-k-1}(h_n) \end{pmatrix}$$

où les h_i sont des éléments linéairement indépendants de L, déterminés par les g_i.

Démonstration. Soit $\mathrm{Gab}_{\theta,k}(g)$ un code de Gabidulin généralisé, soit

$$G = \begin{pmatrix} g_1 & \cdots & g_n \\ \theta(g_1) & \cdots & \theta(g_n) \\ \vdots & \ddots & \vdots \\ \theta^{k-1}(g_1) & \cdots & \theta^{k-1}(g_n) \end{pmatrix}$$

sa matrice génératrice, et soit H une matrice de la forme

$$H = \begin{pmatrix} h_1 & \cdots & h_n \\ \theta(h_1) & \cdots & \theta(h_n) \\ \vdots & \ddots & \vdots \\ \theta^{n-k-1}(h_1) & \cdots & \theta^{n-k-1}(h_n) \end{pmatrix}.$$

H est une matrice de contrôle du code si et seulement si $G \cdot H^t = 0$. Cette relation correspond aux $k(n-k)$ équations

$$\begin{array}{llll} \sum_{i=1}^n g_i h_i = 0 & \sum_{i=1}^n g_i \theta(h_i) = 0 & \cdots & \sum_{i=1}^n g_i \theta^{n-k-1}(h_i) = 0 \\ \sum_{i=1}^n \theta(g_i) h_i = 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \sum_{i=1}^n \theta^{k-1}(g_i) h_i = 0 & \cdots & \cdots & \sum_{i=1}^n \theta^{k-1}(g_i)\theta^{n-k-1}(h_i) = 0. \end{array}$$

Une grande partie de ces équations sont redondantes, en effet, θ étant un automorphisme, nous avons l'équivalence suivante :

$$\sum_{i=1}^n \theta^j(g_i)\theta^\ell(h_i) = 0 \Leftrightarrow \sum_{i=1}^n \theta^{j+1}(g_i)\theta^{\ell+1}(h_i) = 0.$$

Ainsi, seules les équations de la première ligne et de la première colonne sont utiles. De la même façon, nous pouvons nous ramener à un système d'équations dans lesquelles θ ne porte que sur les g_i :

$$\sum_{i=1}^n \theta^j(g_i) h_i = 0,\ j \in [[1+k-n; k-1]].$$

Enfin, en notant $\widetilde{g}_i = \theta^{1+k-n}(g_i)$, nous obtenons le système

$$\sum_{i=1}^n \theta^j(\widetilde{g}_i) h_i = 0,\ j \in [[0; n-2]],$$

ou, sous forme matricielle :

$$\begin{pmatrix} \widetilde{g}_1 & \cdots & \widetilde{g}_n \\ \vdots & \ddots & \vdots \\ \theta^{n-2}(\widetilde{g}_1) & \cdots & \theta^{n-2}(\widetilde{g}_n) \end{pmatrix} \begin{pmatrix} h_1 \\ \vdots \\ h_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{3.1}$$

Les g_i étant linéairement indépendants sur L, les $\widetilde{g}_i$ le sont aussi. Ainsi, la matrice

$$\begin{pmatrix} \widetilde{g}_1 & \cdots & \widetilde{g}_n \\ \vdots & \ddots & \vdots \\ \theta^{m-1}(\widetilde{g}_1) & \cdots & \theta^{m-1}(\widetilde{g}_n) \end{pmatrix}$$

est de rang n et

$$\begin{pmatrix} \widetilde{g}_1 & \cdots & \widetilde{g}_n \\ \vdots & \ddots & \vdots \\ \theta^{n-2}(\widetilde{g}_1) & \cdots & \theta^{n-2}(\widetilde{g}_n) \end{pmatrix}$$

est alors de rang $n-1$. Il existe donc une solution $(h_1, \ldots, h_n)$ de ce système. De plus, cette solution est unique à un facteur multiplicatif près (dans L).

Montrons maintenant que les h_i sont linéairement indépendants sur K. Pour cela, ajoutons l'équation

$$\sum_i \lambda_i h_i = 0, \lambda_i \in K \tag{3.2}$$

au système (3.1), et montrons que les λ_i sont tous nuls.

$$\begin{pmatrix} \widetilde{g}_1 & \cdots & \widetilde{g}_n \\ \vdots & \ddots & \vdots \\ \theta^{n-2}(\widetilde{g}_1) & \cdots & \theta^{n-2}(\widetilde{g}_n) \\ \lambda_1 & \cdots & \lambda_n \end{pmatrix} \begin{pmatrix} h_1 \\ \vdots \\ h_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{3.3}$$

Si cette matrice était de rang n, les h_i seraient tous nuls, ce qui est impossible pour une matrice de contrôle. Ainsi, la nouvelle équation 3.2 est une combinaison linéaire des précédentes :

$$\forall i \in [[1;n]],\ \lambda_i = \mu_0\theta^0(\tilde{g}_i) + \cdots + \mu_{n-2}\theta^{n-2}(\tilde{g}_i), \mu_j \in L. \tag{3.4}$$

Notons M le polynôme $\sum \mu_i X^i$. Les λ_i étant des éléments de K, ils sont racines de $X-1$, nous avons donc :

$$\mathcal{L}_{(X-1)\cdot M}(\tilde{g}_i) = 0.$$

$(X-1)\cdot M$ est un θ-polynôme de degré $n-1$, qui s'annule en n valeurs K-linéairement indépendantes. M est le polynôme nul (corollaire 7), donc les λ_i sont nuls. Cela prouve l'indépendance K-linéaire des h_i. □

Corollaire 22. *Le dual d'un code de Gabidulin généralisé de paramètres $[n,k,d]$ est un code de Gabidulin généralisé de paramètres $[n, n-k, k+1]$. En particulier, c'est un code MRD.*

Démonstration. La proposition précédente donne explicitement la matrice génératrice de ce dual. En remarquant qu'il s'agit de la matrice génératrice d'un code de Gabidulin généralisé, nous en déduisons les paramètres du code dual. □

Changement de support La matrice génératrice d'un code de Gabidulin peut "absorber" une matrice à coefficients dans K. Dans certains cas, cela permet d'obtenir un nouveau code de Gabidulin.

Lemme 23. *Soit G la matrice génératrice d'un code de Gabidulin généralisé de longueur n et soit $A \in \mathcal{M}_{n,m}(K)$ une matrice à coefficients dans K. Alors le produit $\widetilde{G} = GA$ vaut :*

$$\begin{pmatrix} g_1 & \cdots & g_n \\ \vdots & \ddots & \vdots \\ \theta^{k-1}(g_1) & \cdots & \theta^{k-1}(g_n) \end{pmatrix} \cdot \begin{pmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{pmatrix} = \begin{pmatrix} \widetilde{g}_1 & \cdots & \widetilde{g}_m \\ \vdots & \ddots & \vdots \\ \theta^{k-1}(\widetilde{g}_1) & \cdots & \theta^{k-1}(\widetilde{g}_m) \end{pmatrix},$$

où $\widetilde{g}_i = \sum_{j=1}^n a_{j,i} g_j$.

Démonstration. Il suffit de calculer le coefficient de la i-ème ligne et de la j-ème colonne de $\widetilde{G}$, qui vaut :

$$\sum_{\ell=1}^n \theta^{i-1}(g_\ell) h_{\ell,j} = \theta^{i-1}(\sum_{\ell=1}^n g_\ell h_{\ell,j}).$$

□

Proposition 24. *Soit G la matrice génératrice d'un code de Gabidulin généralisé et soit $A \in \mathcal{M}_{n,m}(K)$ une matrice à coefficients dans K. Alors la matrice $\widetilde{G} = GA$ est la matrice génératrice d'un code de Gabidulin généralisé si et seulement si la matrice A est injective. Ce nouveau code a pour support la famille $(\widetilde{g}_1, \ldots, \widetilde{g}_m)$ définie dans le lemme 23.*

Démonstration. Le lemme précédent donne la forme de $\widetilde{G}$. Pour que ce soit la matrice génératrice d'un code de Gabidulin généralisé, la famille $\widetilde{g} = (\widetilde{g}_1, \ldots, \widetilde{g}_m)$ doit être libre. Pour cela, il faut et il suffit que la matrice A soit injective de $\mathrm{Vect}(g_1, \cdots, g_n)$ dans l'image de A. □

Versions vectorielle et matricielle Si nous suivons scrupuleusement la définition 15, les mots d'information d'un code de Gabidulin sont des θ-polynômes de degré inférieur à k, et les mots du code sont des vecteurs de longueur n à coefficients dans L.

Nous avons déjà fait une entorse à cette définition en donnant la matrice génératrice du code. En effet, en parlant de matrice génératrice, nous voyons implicitement les mots d'information comme des vecteurs de longueur k. Il ne s'agit là que d'une convention d'écriture, puisqu'un θ-polynôme de $L[\theta; X]_{<k}$ et un vecteur de L^k contiennent la même information. Nous disposons en effet d'une bijection naturelle entre ces deux ensembles :

$$\begin{array}{ccc} L[X;\theta]_{<k} & \longrightarrow & L^k \\ a_0 + a_1 X + \cdots + a_{k-1} X^{k-1} & \longmapsto & (a_0, \ldots, a_{k-1}). \end{array}$$

Nous disposons d'une bijection $\mathrm{ext}_{\mathcal{B}}$ entre L et K^m, qui induit une bijection, également notée $\mathrm{ext}_{\mathcal{B}}$, entre L^n et $\mathcal{M}_{m\times n}(K)$ (voir définition 11). Ainsi, les codes de Gabidulin peuvent être vus comme des ensembles de matrices à m lignes et n colonnes. De même, les mots d'information peuvent être vus comme des matrices à m lignes et k colonnes. La métrique rang est alors celle vue dans la définition 10 plutôt que dans la définition 12 (qui se base sur la bijection $\mathrm{ext}_{\mathcal{B}}$ pour se ramener au cas précédent).

Dans le premier (resp. second) cas, où les mots du codes sont des vecteurs (resp. des matrices), le code sera dit sous forme vectorielle (resp. matricielle). Les mots d'information seront alors vus comme des polynômes ou comme des vecteurs (resp. comme des matrices).

Exemple 11. *Reprenons l'exemple 10 défini précédemment. Nous avions choisi le mot d'information*

$$f = (1 + i\alpha^2 - \alpha^3) + (\alpha^2 - i\alpha^3)X \in L[X;\theta]_{<2}$$

et calculé le mot du code correspondant

$$c(f) = (1 + (1+i)\alpha^2 - (1+i)\alpha^3,\ \alpha + 2i\alpha^3,\ -(1+i) - (1+i)\alpha + \alpha^2,\ -2i\alpha^2 + \alpha^3) \in L^4.$$

Ce mot du code s'écrit sous forme d'une matrice de la façon suivante

$$C(f) = \begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ -1-i & 2i & 0 & 1 \end{pmatrix} \in \mathcal{M}_{4\times 4}(K).$$

Il en est de même pour le mot d'information, qui peut être vu comme un vecteur $f = (1 + i\alpha^2 -$

$\alpha^3, \alpha^2 - i\alpha^3) \in L^2$ *ou comme une matrice*

$$f = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ i & 1 \\ -1 & -i \end{pmatrix} \in \mathcal{M}_{4\times 2}(K).$$

Remarque 9. *Le mot de code est noté $c(f)$ lorsqu'il est sous forme vectorielle et $C(f)$ sous forme matricielle afin de distinguer les deux écritures. En revanche, le mot d'information est noté f quelle que soit la forme sous laquelle on le représente.*

Dans la suite, nous verrons les mots du code selon la forme la plus adaptée au contexte. Par exemple, dans toute cette section, nous avons utilisé la forme vectorielle car les preuves sont plus simples à exprimer avec cette écriture.

3.2 Décodage et reconstruction (modèle d'erreurs seules)

Considérons un code de Gabidulin généralisé Gab de longueur n, dimension k et de support $g = (g_1, \ldots, g_n)$. Supposons que nous voulions transmettre $f \in L[X;\theta]_{<k}$. Pour cela, nous allons encoder f, et obtenir le mot du code $C(f) \in$ Gab. C'est ce mot du code qui sera envoyé à travers un canal. Lors de la transmission, le mot est modifié, ce qui est modélisé par l'ajout d'une erreur e. Le récepteur recevra une version bruitée $Y = C(f) + E$. Le but est alors de retrouver f.

Nous allons définir les problèmes de Reconstruction (linéaire et non linéaire) et étudier les liens entre ces problèmes et le problème de Décodage.

Exemple 12. *En reprenant le mot du code calculé dans les exemples 10 et 11, une erreur s'écrit de la façon suivante (sous forme matricielle) :*

$$\begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+2i & 0 & 0 & 0 \\ -2-i & 2i & -i & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ -1-i & 2i & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ i & 0 & -1 & 2i \\ -1 & 0 & -i & -2 \end{pmatrix}.$$

L'erreur (la matrice de droite) est de poids "petit", ce qui permet de retrouver le mot d'information. Sous forme vectorielle, nous obtenons l'écriture suivante, dans laquelle nous faisons apparaitre le poids de l'erreur :

$$y = c(f) + (i \cdot e, 0, -e, 2i \cdot e), \text{ où } e = \alpha^2 + i\alpha^3.$$

Dans le cadre que nous avons fixé (extensions galoisiennes cycliques), les codes de Gabidulin généralisés ont les mêmes propriétés que les codes de Gabidulin finis. De façon générale, tout algorithme de décodage valable pour les codes de Gabidulin originaux devrait être valable pour les codes de Gabidulin généralisés. La méthode présentée ici se base sur la notion de Reconstruction.

Définition 17 (Décodage $\mathrm{Dec}(n, k, g, y)$)**.**

Soit Gab *un code de Gabidulin de longueur n, dimension k et support g, et soit y un mot reçu dont la distance à un mot du code est inférieure à la capacité de correction $t = \frac{n-k}{2}$ du code. Résoudre le problème de décodage, noté* $\mathrm{Dec}(n, k, g, y)$ *consiste à trouver, s'ils existent, $f \in L[X;\theta]$ et $e \in L^n$ tels que :*

1. $\deg(f) < k$,
2. $w(e) \leqslant t$,
3. $\forall i \in [[1;n]],\ y_i = \mathcal{L}_f(g_i) + e_i$.

Les problèmes de Reconstruction Définissons maintenant les problèmes de Reconstruction, qui seront à la base de notre méthode de décodage.

Définition 18 (Reconstruction Non Linéaire $\mathrm{RNL}(n,k,t,g,y)$)**.**

Soient $k \leqslant n$ des entiers positifs et $t < n$. Soient $g_1, \ldots, g_n$ des éléments de L linéairement indépendants sur K. Enfin, soient $y_1, \ldots, y_n$ des éléments quelconques de L. Résoudre le problème de Reconstruction $\mathrm{RNL}(n,k,t,g,y)$ *consiste à trouver, s'ils existent, deux θ-polynômes f et V dans $L[X;\theta]$ tels que :*

1. $\deg(f) < k$,
2. $\deg(V) \leqslant t$,
3. $V \neq 0$
4. $\forall i \in [[1;n]],\ \mathcal{L}_V(y_i) = \mathcal{L}_{V\cdot f}(g_i)$.

Proposition 25. *Soient n, k des entiers et g, y des vecteurs de L^n comme dans la définition de Décodage et Reconstruction Non Linéaire. Alors les problèmes* $\mathrm{Dec}(n,k,g,y)$ *et* $\mathrm{RNL}(n,k,\lfloor\frac{n-k}{2}\rfloor,g,y)$ *sont équivalents.*

Démonstration. C'est une conséquence des théorèmes 8 et 6 qui stipulent qu'un vecteur de rang t est annulé par un θ-polynôme de rang t et inversement. Plus précisément, si f et e forment une solution de $\mathrm{Dec}(n,k,g,y)$, alors f et $V = \mathcal{A}_{\langle e_1,\ldots,e_n\rangle}$ forment une solution de $\mathrm{RNL}(n,k,\lfloor\frac{n-k}{2}\rfloor,g,y)$. Inversement, si f et V forment une solution de $\mathrm{RNL}(n,k,\lfloor\frac{n-k}{2}\rfloor,g,y)$, alors f et $(e_i) = (y_i - \mathcal{L}_f(g_i))$ forme une solution de $\mathrm{Dec}(n,k,g,y)$. □

Nous pouvons calculer une solution du problème de Reconstruction en résolvant un système d'équations dont les inconnues sont les coefficients de f et V. Cela met en jeu des produits de la forme f_iV_j, ce qui fait que les équations sont quadratiques, d'où le nom de Reconstruction Non Linéaire. Nous pouvons nous ramener à un problème proche, qui ne comporte que des équations linéaires.

Définition 19 (Reconstruction Linéaire $\mathrm{RL}(n,k,t,g,y)$)**.**

Soient $k \leqslant n$ des entiers positifs et $t < n$. Soient $g_1, \ldots, g_n$ des éléments de L linéairement indépendants sur K. Enfin, soient $y_1, \ldots, y_n$ des éléments quelconques de L. Résoudre le problème de Reconstruction Linéaire, noté $\mathrm{RL}(n,k,t,g,y)$ *consiste à trouver, s'ils existent, deux θ-polynômes N et$W$ dans $L[X;\theta]$ tels que :*

1. $\deg(N) < k + t$,
2. $\deg(W) \leqslant t$,
3. $W \neq 0$
4. $\forall i \in [[1;n]],\ \mathcal{L}_W(y_i) = \mathcal{L}_N(g_i)$.

Il est clair que la Reconstruction Linéaire est une généralisation de la Reconstruction Non Linéaire. En effet, une solution (V, f) de $\mathrm{RNL}(n,k,t,g,y)$ fournit une solution $(V, V\cdot f)$ de $\mathrm{RL}(n,k,t,g,y)$.

Remarque 10. *Les coefficients de W et N, solutions du problème de Reconstruction Linéaire, vérifient les équations suivantes :*

$$\begin{pmatrix} g_1 & \cdots & \theta^{k+t-1}(g_1) & y_1 & \cdots & \theta^t(y_1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ g_n & \cdots & \theta^{k+t-1}(g_n) & y_n & \cdots & \theta^t(y_n) \end{pmatrix} \cdot \begin{pmatrix} n_0 \\ \vdots \\ n_{k+t-1} \\ -w_0 \\ \vdots \\ -w_t \end{pmatrix} = 0.$$

Il est donc possible de résoudre le problème de Reconstruction Linéaire par élimination de Gauss. Nous verrons cependant un algorithme plus efficace dans le prochain chapitre.

Théorème 26. *Si $t \leqslant \lfloor \frac{n-k}{2} \rfloor$ et s'il existe une solution de Reconstruction Non Linéaire, alors toute solution (N, W) de reconstruction Linéaire fournit une solution (f, V) de Reconstruction Non Linéaire. Le polynôme f est le quotient de N par W à gauche, noté $W \backslash N$.*

Démonstration. Supposons que le problème $\mathrm{RNL}(n, k, t, g, y)$ admette une solution (V, f) et soit (W, N) une solution du problème $\mathrm{RL}(n, k, t, g, y)$. Les polynômes W et N sont connus, contrairement à V et f. Posons $e_i := y_i - \mathcal{L}_f(g_i)$, $i = 1, \ldots, n$. Nous avons alors :

$$\mathcal{L}_V(e_i) = \mathcal{L}_V(y_i - \mathcal{L}_f(g_i)) = 0.$$

Donc e est de rang au plus t. D'autre part, Nous avons :

$$\begin{aligned} \mathcal{L}_W(e_i) &= \mathcal{L}_W(y_i) - \mathcal{L}_W(\mathcal{L}_f(g_i)) \\ &= \mathcal{L}_N(g_i) - \mathcal{L}_W(\mathcal{L}_f(g_i)). \end{aligned}$$

Puisqu'un θ-polynôme peut être vu comme une application K-linéaire, le rang de $W(e)$ est au plus le rang de e, c'est-à-dire au plus t. Il existe donc un polynôme U de degré au plus t (théorème 8) tel que :

$$\mathcal{L}_U(\mathcal{L}_W(e_i)) = \mathcal{L}_U(\mathcal{L}_N(g_i) - \mathcal{L}_W(\mathcal{L}_f(g_i))) = 0.$$

Nous obtenons donc :

$$\mathcal{L}_{[U \cdot (N - W \cdot f)]}(g_i) = 0.$$

C'est un θ-polynôme de degré au plus $n-1$, nul sur n valeurs linéairement indépendantes. Il s'agit donc du polynôme nul (corollaire 7). Puisqu'il n'y a pas de diviseurs de zéro dans $L[X; \theta]$, nous en déduisons :

$$N - W \cdot f = 0.$$

Ainsi, il suffit d'une division euclidienne pour trouver f. □

Ainsi, dans un contexte de décodage unique jusqu'à la capacité de correction $\lfloor \frac{n-k}{2} \rfloor$, les deux problèmes de Reconstruction sont équivalents.

Remarque 11. *Bien qu'ils ne soient utilisés que dans ce contexte, les deux problèmes de Reconstructions sont définis indépendamment de la notion de code. La relation entre ces deux problèmes reste valable, faisant apparaitre la quantité $\lfloor \frac{n-k}{2} \rfloor$, qui correspond à la capacité de décodage dès lors que le problème est instancié dans un contexte de décodage.*

Théorème 27 (Conclusion)**.** *Dans ce modèle d'erreur, nous pouvons retrouver le polynôme d'information f à partir du mot reçu y si l'erreur e est de poids $w(e) \leqslant t = \lfloor \frac{n-k}{2} \rfloor$. L'algorithme de décodage repose sur un algorithme de Reconstruction.*

Algorithm 3 Décodage (erreur seule)

Input: $k < n$ des paramètres entiers
$g = (g_1, \ldots, g_n) \in L^n$, des éléments K-linéairement indépendants de L
$y = (y_1, \ldots, y_n) \in L^n$,

Output: $f \in L[X;\theta]$

0: $t \longleftarrow \lfloor \frac{n-k}{2} \rfloor$
0: $(N, W) \longleftarrow \mathrm{RNL}(n, k, t, g, y)$
0: (voir 10 ou 4)
0: $f \longleftarrow W \backslash N$
1: **return** f =0

3.3 Premier modèle d'effacement

Considérons un code de Gabidulin généralisé sous forme matricielle. Dans ce modèle, en plus des erreurs vues dans la section précédente, des coefficients de la matrice transmise peuvent être effacés. Un coefficient effacé est noté par la valeur ?.

Le mot reçu se présente de la façon suivante :

$$Y = C(f) + E + \mathcal{E},$$

où $C(f)$ désigne un mot du code, correspondant au mot d'information f, E désigne une matrice d'erreur et $\mathcal{E}$ une matrice d'effacement. La matrice d'effacement est constituée uniquement de 0 et de ?. (L'addition de ? est définie par $\forall x$, $x+? =?$.) Son rang n'est pas défini, aussi nous la mesurerons avec la métrique *term-rank* (voir définition 9).

Dans ce modèle, nous supposons que le récepteur connait l'emplacement des effacements, via un ensemble minimal de lignes et colonnes recouvrant tous les coefficients effacés. Nous noterons S_ℓ (resp. S_c) l'ensemble des indices de ces lignes (resp. colonnes), et s_ℓ (resp. s_c) leur nombre. Nous allons voir qu'il est possible de gérer ces effacements de façon à obtenir un autre code de Gabidulin.

Exemple 13. *En reprenant le mot du code calculé dans les exemples 10 et 11, un mot de code (sous forme matricielle) ayant subi des effacements ressemble à :*

$$\begin{pmatrix} 1 & 0 & ? & 0 \\ 0 & 1 & ? & 0 \\ 1+i & 0 & 1 & -2i \\ ? & ? & ? & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ -1-i & 2i & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & ? & 0 \\ 0 & 0 & ? & 0 \\ 0 & 0 & 0 & 0 \\ ? & ? & ? & 0 \end{pmatrix}.$$

Dans cet exemple, la dernière ligne et la troisième colonne ont été effacées.

Proposition 28. *Un code de Gabidulin généralisé de paramètres $[n, k]$ dont s_c colonnes ont été effacées se ramène à un code de Gabidulin de paramètres $[n - s_c, k]$.*

Démonstration. En effet, chaque colonne d'une matrice du code correspond à une évaluation. En supprimant les colonnes effacées et les coefficients correspondants du support, nous conservons $n - s_c$ évaluations indépendantes d'un θ-polynôme de degré inférieur à k. □

Exemple 14. *En considérant uniquement l'effacement de la troisième colonne,*

$$\begin{pmatrix} 1 & 0 & ? & 0 \\ 0 & 1 & ? & 0 \\ 1+i & 0 & 1 & -2i \\ -1-i & 2i & ? & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ -1-i & 2i & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & ? & 0 \\ 0 & 0 & ? & 0 \\ 0 & 0 & 0 & 0 \\ ? & ? & ? & 0 \end{pmatrix}.$$

nous obtenons, sous forme vectorielle :

$$(\mathcal{L}_f(g_1), \mathcal{L}_f(g_2), \mathcal{L}_f(g_3), \mathcal{L}_f(g_4)) \rightsquigarrow (\mathcal{L}_f(g_1), \mathcal{L}_f(g_2), ?, \mathcal{L}_f(g_4)) \rightsquigarrow (\mathcal{L}_f(g_1), \mathcal{L}_f(g_2), \mathcal{L}_f(g_4)).$$

Nous ne disposons plus que de trois évaluations, mais cela suffit pour interpoler un θ-polynôme de degré au plus 1.

Proposition 29. *Un code de Gabidulin généralisé de paramètres $[n, k]$ dont s_ℓ lignes ont été effacées se ramène à un code de Gabidulin de paramètres $[n, k + s_\ell]$.*

Démonstration. Un effacement de ligne se traduit par la perte, pour toutes les évaluations, du coefficient de y_i selon la composante b_j. Nous connaissons donc l'évaluation en g_i, avec une coordonnée incertaine :

$$\mathcal{L}_f(g_i) = y_i + ?b_j.$$

Il est possible de se débarrasser de cette incertitude en faisant agir le θ-polynôme annulateur $\mathcal{A}_{\langle b_j \rangle}$ de cet élément b_j :

$$\mathcal{L}_{\mathcal{A}_{\langle b_j \rangle}}(\mathcal{L}_f(g_i)) = \mathcal{L}_{\mathcal{A}_{\langle b_j \rangle}}(y_i + ?b_j) = \mathcal{L}_{\mathcal{A}_{\langle b_j \rangle}}(y_i).$$

En posant $z_i = \mathcal{L}_{\mathcal{A}_{\langle b_j \rangle}}(y_i)$ et $F = \mathcal{A}_{\langle b_j \rangle} \cdot f$, nous devons alors résoudre l'équation

$$\mathcal{L}_F(g_i) = z_i,$$

ce qui correspond à un code de Gabidulin de dimension supérieure.
Lorsqu'il y a plusieurs effacements de lignes, nous calculons $\mathcal{A}_{\langle b_j, j \in S_\ell \rangle}$, qui annule toutes les composantes effacées. Il est de degré s_ℓ. □

Exemple 15. *En considérant uniquement l'effacement de la dernière ligne,*

$$\begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ ? & ? & ? & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ -1-i & 2i & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & ? & 0 \\ 0 & 0 & ? & 0 \\ 0 & 0 & 0 & 0 \\ ? & ? & ? & 0 \end{pmatrix},$$

nous obtenons, sous forme vectorielle :

$$\begin{aligned} \mathcal{L}_f(1) &= 1 + (1+i)\alpha^2 + \epsilon_1\alpha^3, \\ \mathcal{L}_f(\alpha) &= \alpha^2 + \epsilon_2\alpha^3, \\ \mathcal{L}_f(\alpha^2) &= -(1+i) - (1+i)\alpha + \alpha^2 + \epsilon_3\alpha^3, \\ \mathcal{L}_f(\alpha^3) &= -2i\alpha^2 + \epsilon_4\alpha^3, \end{aligned}$$

où les $\epsilon_i \in \mathbb{Q}[i] = K$ *sont inconnus. En faisant agir* $A = \mathcal{A}_{\langle \alpha^3 \rangle}$*, nous obtenons :*

$$\begin{aligned} z_1 &= \mathcal{L}_A(y_1) = \mathcal{L}_{A\cdot f}(g_1) = (1+i) - 2\alpha^2, \\ z_2 &= \mathcal{L}_A(y_2) = \mathcal{L}_{A\cdot f}(g_2) = 2i\alpha, \\ z_3 &= \mathcal{L}_A(y_3) = \mathcal{L}_{A\cdot f}(g_3) = -2i + (2-2i)\alpha + (1-i)\alpha^2, \\ z_4 &= \mathcal{L}_A(y_4) = \mathcal{L}_{A\cdot f}(g_4) = (2i+2)\alpha^2. \end{aligned}$$

Ces équations correspondent bien à l'évaluation d'un θ*-polynôme* $A \cdot f$ *de degré* 3*. Il suffit ensuite de le diviser par* A *pour retrouver* f*.*

Théorème 30 (Conclusion)**.** *Soit* Gab *un code de Gabidulin de longueur* n *et de dimension* k*. Soit* $Y = C(f) + E + \mathcal{E}$ *un mot reçu. Si*

$$2 \cdot \mathrm{w_r}(E) + \mathrm{w_{tr}}(\mathcal{E}) \leqslant n - k,$$

alors nous pouvons retrouver le mot d'information f*.*

Démonstration. La gestion des effacements, qu'ils portent sur les lignes ou sur les colonnes, revient à considérer un autre code de Gabidulin généralisé. Avec s_ℓ effacements de ligne et s_c effacements de colonne, nous nous ramenons à un code de longueur $n - s_c$ et de dimension $k + s_\ell$. nous pouvons donc corriger une erreur si son poids est inférieur à $\lfloor \frac{(n-s_c)-(k+s_\ell)}{2} \rfloor$. □

Nous pouvons généraliser l'énoncé du problème de Reconstruction Non-Linéaire au modèle avec erreurs et effacements. RNL_$E(n, k, t, g, y, S_\ell, S_c)$ consiste à trouver un θ-polynôme f de degré strictement inférieur à k et un θ-polynôme V de degré au plus t tels que :

$$\mathcal{L}_{V\cdot\mathcal{V}}(y_i) = \mathcal{L}_{V\cdot\mathcal{V}\cdot f}(g_i), \forall i \notin S_c$$

où $\mathcal{V} = \mathcal{A}_{\langle b_j, j \in S_\ell \rangle}$.

Dans ce cas, l'algorithme de décodage est le suivant.

Algorithm 4 Décodage avec erreurs et effacements (premier modèle)

Input: $k < n$ des paramètres entiers,
$(g_1, \ldots, g_n) \in L^n$, des éléments K-linéairement indépendants,
$(y_1, \ldots, y_n) \in L^n$, des éléments quelconques,
S_c et S_ℓ, les indices des effacements de colonnes et de lignes.

Output: le mot d'information f.

0: $\widetilde{g} \longleftarrow (g_i, i \notin S_c)$
0: $\widetilde{y} \longleftarrow (y_i, i \notin S_c)$
0: $\mathcal{V} \longleftarrow \mathcal{A}_{\langle b_i, i \in S_\ell \rangle}$
0: $z_i \longleftarrow \mathcal{L}_{\mathcal{V}}(y_i)$
0: $(N, W) \longleftarrow \mathrm{RNL}(n - s_c, k + s_\ell, \lfloor \frac{n-s_c-k-s_\ell}{2} \rfloor, \widetilde{g}, \widetilde{y})$
0: $F \longleftarrow W \backslash N$
0: $f \longleftarrow \mathcal{V} \backslash F$
1: **return** f =0

Remarque 12. *Remarquons que cet algorithme contient l'algorithme de décodage vu dans la section précédente (instructions 4 à 4. En effet, les effacements de colonne sont gérés par les instructions 4*

et 4. Les effacements de ligne sont quant à eux gérés autour de l'algorithme de décodage, par les instructions 4, 4 et 4.

Remarquons également que les étapes concernant les effacements de colonne ont pour effet de modifier le support du code. Ainsi, contrairement à la section précédente, tous les algorithmes de décodage ne conviennent pas. En effet, un algorithme se basant sur une propriété du support ne peut être utilisé en cas d'effacements.

3.4 Second modèle d'effacement

Ce modèle, décrit dans [LSS14], trouve son origine dans le domaine du *network coding*. Il se base sur la décomposition de la matrice d'erreur en deux facteurs, l'un ou l'autre pouvant être connu.

Une matrice M de taille $m \times n$ de rang r peut s'écrire comme un produit $A \cdot B$ de deux matrices de rang r, et de tailles respectives $m \times r$ et $r \times n$. Dans le cas où M est une erreur lors de la transmission d'un mot de code (nous considérons que le code est sous forme matricielle), cette décomposition admet une interprétation (pour le code sous forme vectorielle). Les colonnes de A correspondent aux valeurs de l'erreur, et la matrice B à la répartition de ces valeurs sur les coefficients du mot du code. Une telle décomposition n'est pas unique, mais la connaissance d'un facteur d'une décomposition permet d'améliorer le décodage.

Exemple 16. *L'erreur que nous avons vue précédemment peut s'écrire de la façon suivante :*

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ i & 0 & -1 & 2i \\ -1 & 0 & -i & -2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ i \end{pmatrix} \cdot \begin{pmatrix} i & 0 & -1 & 2i \end{pmatrix}.$$

Il s'agit, pour le code sous forme vectorielle, d'une erreur de valeur $\alpha^2 + i\alpha^3$, répartie sur le premier, le troisième et le dernier coefficient du mot du code :

$$(y_1,\, y_2,\, y_3,\, y_4) = (c_1 + i \cdot e,\, c_2,\, c_3 - e,\, c_4 + 2i \cdot e).$$

Dans ce modèle, nous supposons que le récepteur connait A ou B pour une partie des erreurs. Ces erreurs partiellement connues sont appelées effacements de ligne (resp. de colonne) si A (resp. B) est connue. Le mot reçu s'écrit alors de la façon suivante :

$$Y = C(f) + A_e \cdot B_e + A_\ell \cdot B_\ell + A_c \cdot B_c,$$

où A_ℓ et B_c sont connues. En notant s_c (resp. s_ℓ) le nombre d'effacements de colonnes (resp. de lignes), les matrices du membre de droite ont pour tailles respectives $m \times n$, $m \times r$, $r \times n$, $m \times s_r$, $s_r \times n$, $m \times s_c$ et $s_c \times n$.

Le premier modèle d'effacements est un cas particulier de celui-ci, comme le montre l'exemple suivant. Cela explique que les deux notions soient regroupées sous le même terme "effacement".

Exemple 17. *Considérons un effacement de ligne dans le premier modèle. Nous avons l'égalité*

suivante :

$$\begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ ? & ? & ? & ? \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ -1-i & 2i & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ ? & ? & ? & ? \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ -1-i & 2i & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} ? & ? & ? & ? \end{pmatrix}.$$

La principale différence est que dans ce deuxième modèle d'effacements, les ? sont de vraies valeurs (dans L), connues dans la matrice de gauche et inconnues dans celle de droite, contrairement au premier modèle où les ? agissent comme un masque.

Proposition 31. *Un code de Gabidulin généralisé de paramètres $[n, k]$ dont s_c colonnes ont été effacées se ramène à un code de Gabidulin de paramètres $[n - s_c, k]$.*

Démonstration. Des effacements de colonnes sont des erreurs dont la répartition est connue. Nous allons modifier cette répartition afin de regrouper ces erreurs sur un minimum de colonnes. Pour cela, nous allons mettre la matrice B_c sous forme échelonnée réduite par colonnes. Notons M la matrice (inversible) ayant agi sur les colonnes de B_c. Nous obtenons alors l'équation suivante :

$$Y \cdot M = C(f) \cdot M + A_e \cdot B_e \cdot M + A_\ell \cdot B_\ell \cdot M + A_c \cdot B_c \cdot M.$$

$C(f) \cdot M$ est un mot d'un code de Gabidulin, avec un support différent (voir 24). Ce nouveau support s'obtient en appliquant sur l'ancien support chacune des étapes de l'élimination gaussienne ayant permis la mise sous forme échelonnée par colonnes de B_c. La matrice M étant inversible, $A_e \cdot B_e \cdot M$ est une erreur de même poids que $A_e \cdot B_e$ et $A_\ell \cdot B_\ell \cdot M$ est un effacement de ligne de même poids que $A_\ell \cdot B_\ell$. Il suffit alors de supprimer les s_c premières colonnes pour se débarrasser des effacements de colonnes. Nous obtenons donc un code de Gabidulin de paramètres $[n - s_c, k]$. □

Exemple 18. *Considérons un mot de code sous forme matricielle ayant subi un effacement de colonnes de poids 2. Il se présente de la façon suivante :*

$$\begin{pmatrix} c_1 & c_2 & c_3 & c_4 \end{pmatrix} + \begin{pmatrix} \epsilon_1 & \epsilon_2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Après un échange de colonnes ($c_2 \leftrightarrow c_3$) et une transvection ($c_4 \leftarrow c_4 - c_1 - c_2$), nous obtenons :

$$\begin{pmatrix} c_1 & c_3 & c_2 & c_4 - c_1 - c_2 \end{pmatrix} + \begin{pmatrix} \epsilon_1 & \epsilon_2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Il suffit alors de supprimer les deux premières colonnes pour se débarrasser des effacements de colonnes. Les actions sur les colonnes de B_c agissent directement sur les composantes de $c(f)$ et du support g, comme nous pouvons le voir dans le cas de la transvection suivante : si $c_1 = \mathcal{L}_f(g_1) + e_1$ et $c_2 = \mathcal{L}_f(g_2) + e_2$, alors nous obtenons $c_1 - c_2 = \mathcal{L}_f(g_1 - g_2) + e_1 - e_2$.

Proposition 32. *Un code de Gabidulin généralisé de paramètres $[n, k]$ ayant subi un effacement de lignes de poids s_ℓ se ramène à un code de Gabidulin de paramètres $[n, k + s_\ell]$.*

Démonstration. Supposons que les effacements de colonnes aient déjà été traités. Sous forme vectorielle, les effacements de lignes s'écrivent de la façon suivante :

$$y_i = \mathcal{L}_f(g_i) + \sum_{j=1} s_\ell b_{i,j} \epsilon_j + e_i,$$

où les $b_{i,j} \in K$ sont les coefficients de B_ℓ, et où les colonnes de A_ℓ sont les $\mathrm{ext}_{\mathcal{B}}(\epsilon_j)$. En évaluant le polynôme annulateur $\mathcal{A}$ des ϵ_j, nous obtenons :

$$\mathcal{L}_{\mathcal{A}}(y_i) = \mathcal{L}_{\mathcal{A}}(\mathcal{L}_f(g_i)) + \mathcal{L}_{\mathcal{A}}(e_i),$$

ce qui correspond à l'évaluation d'un θ-polynôme de degré au plus $k+s_\ell$. Le poids de $(\mathcal{L}_{\mathcal{A}}(e_1), \ldots, \mathcal{L}_{\mathcal{A}}(e_n))$ est de poids au plus celui de $(e_1, \ldots, e_n)$. Nous obtenons donc un code de Gabidulin de paramètres $[n, k + s_\ell]$. □

Exemple 19. *Considérons un mot du code ayant subi un effacement de lignes de poids 2. Il se présente de la façon suivante :*

$$\begin{pmatrix} 1 & 0 & -1-i & 0 \\ 0 & 1 & -1-i & 0 \\ 1+i & 0 & 1 & -2i \\ -1-i & 2i & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & i \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} a_{1,1} & \cdots & a_{1,4} \\ a_{2,1} & \cdots & a_{2,4} \end{pmatrix},$$

ce qui s'écrit sous forme vectorielle :

$$y_i = \mathcal{L}_f(g_i) + a_{1,i}(1+\alpha) + a_{2,i}(i\alpha^2 - \alpha^3).$$

En faisant agir $A = \mathcal{A}_{\langle 1+\alpha, i\alpha^2-\alpha^3 \rangle}$, nous obtenons :

$$\mathcal{L}_A(y_i) = \mathcal{L}_{A \cdot f}(g_i),$$

ce qui correspond à l'évaluation d'un θ-polynôme de degré 4 en 4 valeurs. Il suffit alors d'interpoler pour obtenir $A \cdot f$, et de faire une division euclidienne pour retrouver le mot d'information f.

Théorème 33 (Conclusion)**.** *Soit* Gab *un code de Gabidulin de longueur n et de dimension k. Soit $Y = C(f) + E + \mathcal{E}$ un mot reçu. Si*

$$2 \cdot \mathrm{w_r}(E) + \mathrm{w_{tr}}(\mathcal{E}) \leqslant n - k,$$

où $\mathrm{w_r}$ et $\mathrm{w_{tr}}$ désignent respectivement les métriques rang et term-rank*, alors nous pouvons retrouver le mot d'information f.*

Démonstration. La gestion des effacements, qu'ils portent sur les lignes ou sur les colonnes, revient à considérer un autre code de Gabidulin généralisé. Avec s_ℓ effacements de ligne et s_c effacements de colonne, nous nous ramenons à un code de longueur $n - s_c$ et de dimension $k + s_\ell$. nous pouvons donc corriger une erreur si son poids est inférieur à $\lfloor \frac{(n-s_c)-(k+s_\ell)}{2} \rfloor$. En effet, le poids de l'erreur n'augmente pas lors de ces modifications. □

Nous pouvons généraliser l'énoncé du problème de Reconstruction Non-Linéaire au modèle avec erreur et effacements. RNL_$E(n, k, t, g, y, S_\ell, S_c)$ consiste à trouver un θ-polynôme f de degré strictement inférieur à k et un θ-polynôme V de degré au plus t tels que :

$$\mathcal{L}_{V\cdot\mathcal{V}}(\widetilde{y}_i) = \mathcal{L}_{V\cdot\mathcal{V}\cdot f}(\widetilde{g}_i), \forall i \in [[s_c + 1; n]]$$

où $\mathcal{V}$ est l'annulateur des colonnes de A_ℓ et $\widetilde{g}$ (resp. $\widetilde{y}$) le support (resp. les évaluations) obtenu(es) après l'élimination gaussienne sur les colonnes de B_c.

Dans ce cas, l'algorithme de décodage est le suivant.

Algorithm 5 Décodage avec erreurs et effacements (second modèle)

Input: $k < n$ des paramètres entiers,
$(g_1, \ldots, g_n) \in L^n$, des éléments K-linéairement indépendants,
$(y_1, \ldots, y_n) \in L^n$, des éléments quelconques,
B_c et A_ℓ.

Output: le mot d'information f.

0: $M \longleftarrow$ matrice permettant d'échelonner B_c selon les colonnes
0: $\widetilde{g} \longleftarrow g \cdot M$
0: $\widetilde{y} \longleftarrow y \cdot M$
0: $\mathcal{V} \longleftarrow \mathcal{A}_{\langle \epsilon_j, j \in [[1;s_\ell]] \rangle}$ où les ϵ_j désignent les $\mathrm{ext}_{\mathcal{B}}{}^{-1}$ des colonnes de A_ℓ.
0: $z_i \longleftarrow \mathcal{L}_{\mathcal{V}}(y_i)$
0: $(N, W) \longleftarrow \mathrm{RNL}(n - s_c, k + s_\ell, \lfloor \frac{n - s_c - k - s_\ell}{2} \rfloor, \widetilde{g}, \widetilde{y})$
0: $F \longleftarrow W \backslash N$
0: $f \longleftarrow \mathcal{V} \backslash F$
1: **return** f =0

Chapitre 4

Un algorithme de reconstruction de type Welch-Berlekamp

Dans le chapitre précédent, nous avons vu que le décodage d'un code de Gabidulin pouvait être ramené à résoudre le problème de reconstruction linéaire $\mathrm{RL}(n, k, t, g, y)$. Comme nous l'avons remarqué précédemment, les coefficients des polynômes solutions vérifient un système d'équations linéaires. Il est donc possible de résoudre ce problème en $O(n^3)$ opérations, par une simple élimination gaussienne.

Une telle approche ne tient pas compte de la structure particulière du système à résoudre. En effet, remarquons que le calcul d'un θ-polynôme interpolateur est un cas particulier du problème de Reconstruction (poser $k = n$ et $t = 0$). Dans le chapitre 2, nous avons vu que nous pouvions le calculer en $O(n^2)$ opérations, ce qui est plus efficace que les $O(n^3)$ opérations que nous donnerait le calcul par élimination Gaussienne. Il est donc raisonnable d'espérer résoudre ce problème avec une complexité plus faible.

Le but de ce chapitre est de présenter un algorithme permettant de résoudre le problème de reconstruction linéaire en temps quadratique. Il est inspiré de l'algorithme de Welch-Berlekamp [WB86], utilisé pour le décodage des codes de Reed-Solomon [RS60]. Une première version incomplète de l'algorithme a été donnée dans [Loi06] pour les codes de Gabidulin sur des corps finis.

La première partie est consacrée à une présentation générale de l'algorithme. [1] Nous en décrirons les principales étapes de façon informelle. Nous n'entrerons dans les détails que dans la deuxième partie, dont le but est de montrer que l'algorithme fournit toujours une solution au problème de reconstruction. La troisième partie sera consacrée à une étude détaillée de la complexité. Enfin, nous présenterons quelques améliorations dans la quatrième partie. Nous préciserons pour chacune l'impact sur la complexité.

4.1 Algorithme de type Welch-Berlekamp

Le but de cet algorithme est de fournir un couple (N, W) de θ-polynômes solution du problème de Reconstruction Linéaire, pour des paramètres vérifiant $t \leqslant \lfloor \frac{n-k}{2} \rfloor$. Rappelons qu'une solution du problème de Reconstruction Linéaire $\mathrm{RL}(n, k, t, g, y)$ (voir définition 19) doit vérifier :

1. Il existe plusieurs variantes de cet algorithme. J'ai choisi de présenter celle qui me parait la plus simple à décrire. Les variantes seront étudiées en fin de chapitre.

Algorithm 6 Algorithme de reconstruction

Input: la dimension k et la longueur n du code
$(g_1, \ldots, g_n)$, le support
$(y_1, \ldots, y_n)$, le mot reçu (bruité)

Output: N et W solutions du problème de reconstruction

```
# Initialisation
N_0(X) ⟵ A_<g_1,...,g_k>
W_0(X) ⟵ 0
N_1(X) ⟵ I_[g_1,...,g_k],[y_1,...,y_k]
W_1(X) ⟵ 1

# Boucle principale
for i from k to n − 1 do
  # Calcul des défauts
  u_0 ⟵ L_{N_0}(g_i) − L_{W_0}(y_i)
  u_1 ⟵ L_{N_1}(g_i) − L_{W_1}(y_i)

  # Boucle secondaire
  if u_0 = 0 et u_1 ≠ 0 then
    j ⟵ i + 2
    while u_0 = 0 et u_1 ≠ 0 et j ⩽ n do
      u_0 ⟵ L_{N_0}(g_j) − L_{W_0}(y_j)
      u_1 ⟵ L_{N_1}(g_j) − L_{W_1}(y_j)
      j ⟵ j + 1
    end while
    if j=n+1 then
      return (N_1, W_1)
    else
      g_{i+1} ⟷ g_j
      y_{i+1} ⟷ y_j
    end if
  end if =0
```

1. $\deg(N) < k + t$,
2. $0 < \deg(W) \leqslant t$,
3. $\forall i \in [[1; n]],\ \mathcal{L}_W(y_i) = \mathcal{L}_N(g_i)$.

L'algorithme construit une solution itérativement. La principale difficulté réside dans le contrôle des degrés : ceux-ci ne doivent pas augmenter trop vite. C'est pour cela que nous allons construire deux paires de θ-polynômes.

Lors de l'initialisation, nous calculons deux couples $(N_0^{(k)}, W_0^{(k)})$ et $(N_1^{(k)}, W_1^{(k)})$. Ces deux couples vérifient la condition d'interpolation pour les k premiers points (g_i, y_i). L'un de ces couples a pour degrés $k - 1$ et 0.

Algorithm 7 Algorithme de reconstruction (suite)

0: # Mise à jour des θ-polynômes, selon les défauts
0: **if** $u_0 \neq 0$ et $u_1 \neq 0$ **then**
0: $N_0 \longleftarrow (X - \frac{\theta(u_1)}{u_1}) \cdot N_1$
0: $W_0 \longleftarrow (X - \frac{\theta(u_1)}{u_1}) \cdot W_1$
0: $N_1 \longleftarrow N_0 - \frac{u_0}{u_1} N_1$
0: $W_1 \longleftarrow W_0 - \frac{u_0}{u_1} W_1$
0: **end if**
0: **if** $u_0 \neq 0$ et $u_1 = 0$ **then**
0: $N_0 \longleftarrow (X - \frac{\theta(u_1)}{u_1}) \cdot N_1$
0: $W_0 \longleftarrow (X - \frac{\theta(u_1)}{u_1}) \cdot W_1$
0: $N_1 \longleftarrow N_0$
0: $W_1 \longleftarrow W_0$
0: **end if**
0: **if** $u_0 = 0$ et $u_1 = 0$ **then**
0: $N_0 \longleftarrow X \cdot N_1$
0: $W_0 \longleftarrow X \cdot W_1$
0: $N_1 \longleftarrow N_0$
0: $W_1 \longleftarrow W_0$
0: **end if**
0: **end for**
0:
0: **if** $n - k$ is even **then**
0: return N_1, W_1
0: **else**
0: return N_0, W_0
0: **end if**=0

Commence alors la boucle principale de l'algorithme, censée se répéter $n - k$ fois (sauf en cas de sortie prématurée, dont nous parlerons plus tard). Au début de la i-ème itération [2], les couples $(N_0^{(i)}, W_0^{(i)})$ et $(N_1^{(i)}, W_1^{(i)})$ vérifient la condition d'interpolation pour i valeurs. Le but est de vérifier cette condition pour $i + 1$ valeurs à la fin de l'itération.

Pour cela, nous commençons par calculer les défauts $u_{0,i+1}^{(i)}$ et $u_{1,i+1}^{(i)}$. Ces quantités permettent de savoir si le couple correspondant interpole déjà le point (g_{i+1}, y_{i+1}) en début d'itération : c'est le cas si et seulement si le $u_{j,i+1}^{(i)}$ correspondant est nul. Quatre cas sont alors possibles, trois d'entre eux menant à une mise à jour (voir paragraphe suivant) des deux couples de sorte qu'ils interpolent le point supplémentaire (g_{i+1}, y_{i+1}). C'est à ce moment que nous gérons l'augmentation des degrés. En effet, l'une des formules de mise à jour augmente de 1 le degré des polynômes concernés, tandis que l'autre laisse le degré invariant [3]. Au cours de cette mise à jour, les couples changent de rôles, afin que l'augmentation de degré porte sur l'autre couple au tour suivant. Ainsi, en $n-k$ itérations, le degré de chaque polynôme a augmenté de $\approx \frac{n-k}{2} \approx t$.

Concernant ces mises à jour, selon la nullité des u_i, quatre cas sont possibles.

2. $i \in [[k; n-1]]$
3. Nous verrons l'évolution exacte des degrés dans la prochaine section.

$u_{0,i+1}^{(i)}$	$u_{1,i+1}^{(i)}$	$N_0^{(i+1)}$	$N_1^{(i+1)}$	type
$\neq 0$	$\neq 0$	$(X - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}}) \cdot N_1^{(i)}$	$N_0^{(i)} - \frac{u_{0,i+1}^{(i)}}{u_{1,i+1}^{(i)}} N_1^{(i)}$	1
$= 0$	$\neq 0$	$(X - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}}) \cdot N_1^{(i)}$	$N_0^{(i)}$	2
$= 0$	$= 0$	$X \cdot N_1^{(i)}$	$N_0^{(i)}$	3
$\neq 0$	$= 0$	pas de mise à jour dans ce cas.		4

TABLE 4.1 – Mises à jour de $N_0^{(i)}$ et $N_1^{(i)}$. Les formules sont les mêmes pour $W_0^{(i)}$ et $W_1^{(i)}$.

Le cas le plus élémentaire est celui où $u_{0,i+1}^{(i)}$ et $u_{1,i+1}^{(i)}$ sont tous deux non nuls. C'est également le cas qui se produit le plus souvent. Cette mise à jour sera dite basique ou de type 1. La seconde, dite de type 2, est le cas où $u_{0,i+1}^{(i)} = 0$ et $u_{1,i+1}^{(i)} \neq 0$. Dans la version de l'algorithme que nous avons donnée, il s'agit d'un cas particulier de la mise à jour basique. Néanmoins, il est préférable de les distinguer, car elles seront traitées différemment dans une des variantes que nous verrons en section 4.4. La mise à jour de type 3 a lieu lorsque $u_{0,i+1}^{(i)}$ et $u_{1,i+1}^{(i)}$ sont tous deux nuls. Dans ce cas, les deux couples sont déjà solutions dès le début de la boucle. Il n'y a pas besoin de les modifier, il suffit de les échanger. Néanmoins, l'un des couples sera multiplié à gauche par X, afin d'uniformiser l'augmentation de degré avec les deux premières mises à jour. Cette astuce permet de simplifier la preuve donnée dans la prochaine section. L'impact sur la complexité peut être négligé. [4]

Le quatrième cas (quand $u_{0,i+1}^{(i)} \neq 0$ et $u_{1,i+1}^{(i)} = 0$) est particulier. En effet, dans ce cas, nous n'avons pas de formule de mise à jour satisfaisante. Aussi, quand nous sommes confrontés à ce cas, nous cherchons à remplacer le point (g_{i+1}, y_{i+1}) que nous devons traiter par un autre point (g_j, y_j) que nous n'avons pas encore traité et pour lequel une autre mise à jour est possible (c'est-à-dire un point pour lequel $u_{0,j}^{(i)} = 0$ ou $u_{1,j}^{(i)} \neq 0$). Deux issues sont alors possibles. Soit nous trouvons un tel point (g_j, y_j), auquel cas nous l'échangeons avec (g_{i+1}, y_{i+1}) et poursuivons l'algorithme. Soit nous sommes dans le cas où $u_{1,i+1}^{(i)}$ est nul et $u_{0,i+1}^{(i)}$ non nul pour tous les (g_j, y_j) encore non traités. Dans ce cas, le couple $(N_1^{(i)}, W_1^{(i)})$ est une solution du problème. Il suffit alors de retourner ce couple. C'est ce que nous appellerons une sortie prématurée. Cela se produit lorsque le poids de l'erreur est inférieur à la capacité de correction du code.

Enfin, terminons cette présentation de l'algorithme en précisant le déroulement de celui-ci. Si le mot y en entrée est de la forme $y = C(f) + e$ avec $\mathrm{w_r}(e) \leqslant \lfloor \frac{n-k}{2} \rfloor$, alors nous aurons en premier lieu $2\,\mathrm{w_r}(e)$ mises à jour de type 1 (parfois de type 2 ou 3), suivies, si $2\,\mathrm{w_r}(e) < n - k$, d'une sortie prématurée. En particulier, si $n - k$ est impair, ou si le poids de l'erreur est inférieur à la capacité de correction du code, alors l'algorithme se termine nécessairement par une sortie prématurée.

4.2 Preuve

Nous allons montrer que pour $t \leqslant \lfloor \frac{n-k}{2} \rfloor$, l'algorithme présenté dans la première section fournit bien une solution du problème de Reconstruction Linéaire $\mathrm{RL}(n, k, t, g, y)$.

4. Multiplier par X revient à décaler les coefficients du polynôme en appliquant θ à chacun d'eux. Ainsi, le degré augmente de 1 mais pas le nombre de coefficients du polynôme, le coefficient unitaire étant nul. Enfin, cette mise à jour se produit très rarement.

Théorème 34. *Soient n, k, t des paramètres entiers vérifiant $k < n$ et $t \leqslant \lfloor \frac{n-k}{2} \rfloor$. Soit $g \in L^n$ une famille de n éléments K-linéairement indépendants de L et $y \in L^n$ des éléments de L. Alors les θ-polynômes N et W en sortie de l'algorithme 6 forment une solution du problème de Reconstruction Linéaire* $\mathrm{RL}(n, k, t, g, y)$.

La preuve va être découpée en trois parties, chacune étant présentée comme un lemme.

Lemme 35. *Soient n, k, t, g et y comme dans l'énoncé du théorème 34. Alors le couple de θ-polynômes (N, W) en sortie de l'algorithme 6 vérifie les conditions d'interpolation du problème de Reconstruction Linéaire* $\mathrm{RL}(n, k, t, g, y)$*, c'est-à-dire :*

$$\forall i \in [[1; n]],\ \mathcal{L}_N(g_i) = \mathcal{L}_W(y_i).$$

Démonstration. La preuve se décompose en deux parties. Nous allons d'abord montrer que les couples $(N_0^{(r)}, W_0^{(r)})$ et $(N_1^{(r)}, W_1^{(r)})$, s'ils ont été calculés, vérifient l'hypothèse de récurrence $\mathcal{H}(r)$ suivante :

$$\mathcal{H}(r) : \forall i \in [[1; r]],\quad \begin{array}{l} \mathcal{L}_{N_0^{(r)}}(g_i) = \mathcal{L}_{W_0^{(r)}}(y_i), \\ \mathcal{L}_{N_1^{(r)}}(g_i) = \mathcal{L}_{W_1^{(r)}}(y_i). \end{array}$$

Nous montrerons ensuite que le couple (N, W) en sortie d'algorithme vérifie la condition d'interpolation, qui n'est autre que $\mathcal{H}(n)$.

Nous pouvons, sans perte de généralité, supposer qu'il n'y a pas d'échange de points (g_i, y_i) lors de l'algorithme. En effet, les évaluations n'étant pas ordonnées, nous pouvons supposer qu'elles se présentent dans un ordre ne nécessitant pas de tels échanges.

Initialisons notre récurrence au rang k. Lors de l'initialisation, les θ-polynômes valent $N_0^{(k)} = \mathcal{A}_{\langle g_1, \cdots, g_k \rangle}$, $W_0^{(k)} = 0$, $N_1^{(k)} = \mathcal{I}_{[g_1, \cdots, g_k]; [y_1, \cdots, y_k]}$ et $W_1^{(k)} = 1$. Il suffit de constater que

$$\begin{aligned} \forall i \in [[1; k]],\ \mathcal{L}_{\mathcal{A}_{\langle g_1, \cdots, g_k \rangle}}(g_i) &= 0 = \mathcal{L}_0(y_i) \\ \mathcal{L}_{\mathcal{I}_{[g_1, \cdots, g_k]; [y_1, \cdots, y_k]}}(g_i) &= y_i = \mathcal{L}_1(y_i) \end{aligned}$$

pour montrer que $\mathcal{H}(k)$ est vraie.

Montrons l'hérédité de $\mathcal{H}$ lors d'une mise à jour de type 1. Soit $k \leqslant r < n$. Par hypothèse de récurrence, nous disposons de θ-polynômes $N_0^{(r)}$, $W_0^{(r)}$, $N_1^{(r)}$, $W_1^{(r)}$ vérifiant $\mathcal{H}(r)$. Les polynômes obtenus par la mise à jour interpolent toujours les r premiers points (g_i, y_i) :

$$\begin{aligned} \forall i \in [[1; r]],\ \mathcal{L}_{N_0^{(r+1)}}(g_i) - \mathcal{L}_{W_0^{(r+1)}}(y_i) & \\ &= \mathcal{L}_{\left(X - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}}\right) N_1^{(r)}}(g_i) - \mathcal{L}_{\left(X - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}}\right) W_1^{(r)}}(y_i) \\ &= \mathcal{L}_{\left(X - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}}\right)}\left(\left(\mathcal{L}_{N_1^{(r)}}(g_i) - \mathcal{L}_{W_1^{(r)}}(y_i)\right)\right) \\ &= 0, \forall i \in [[1; r]],\ \mathcal{L}_{N_1^{(r+1)}}(g_i) - \mathcal{L}_{W_1^{(r+1)}}(y_i) \\ &= \left(\mathcal{L}_{N_0^{(r)}}(g_i) - \frac{u_{0,i+1}^{(i)}}{u_{1,i+1}^{(i)}} \mathcal{L}_{N_1^{(r)}}(g_i)\right) - \left(\mathcal{L}_{W_0^{(r)}}(y_i) - \frac{u_{0,i+1}^{(i)}}{u_{1,i+1}^{(i)}} \mathcal{L}_{W_1^{(r)}}(y_i)\right) \\ &= \left(\mathcal{L}_{N_0^{(r)}}(g_i) - \mathcal{L}_{W_0^{(r)}}(y_i)\right) - \frac{u_{0,i+1}^{(i)}}{u_{1,i+1}^{(i)}} \left(\mathcal{L}_{N_1^{(r)}}(g_i) - \mathcal{L}_{W_1^{(r)}}(y_i)\right) \\ &= 0. \end{aligned}$$

Ces polynômes interpolent également le point (g_{i+1}, y_{i+1}) :

$$\begin{aligned}
&\mathcal{L}_{N_0^{(r+1)}}(g_{i+1}) - \mathcal{L}_{W_0^{(r+1)}}(y_{i+1}) \\
&\qquad = \mathcal{L}_{\left(X - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}}\right)}\left(\left(\mathcal{L}_{N_1^{(r)}}(g_i) - \mathcal{L}_{W_1^{(r)}}(y_i)\right)\right) \\
&\qquad = \mathcal{L}_{\left(X - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}}\right)}(u_{1,i+1}^{(i)}) \\
&\qquad = \theta(u_{1,i+1}^{(i)}) - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}} \cdot u_{1,i+1}^{(i)} \\
&\qquad = 0, \mathcal{L}_{N_1^{(r+1)}}(g_{i+1}) - \mathcal{L}_{W_1^{(r+1)}}(y_{i+1}) \\
&\qquad = \left(\mathcal{L}_{N_0^{(r)}}(g_{i+1}) - \mathcal{L}_{W_0^{(r)}}(y_{i+1})\right) - \frac{u_{0,i+1}^{(i)}}{u_{1,i+1}^{(i)}}\left(\mathcal{L}_{N_1^{(r)}}(g_{i+1}) - \mathcal{L}_{W_1^{(r)}}(y_{i+1})\right) \\
&\qquad = u_{0,i+1}^{(i)} - \frac{u_{0,i+1}^{(i)}}{u_{1,i+1}^{(i)}} \cdot u_{1,i+1}^{(i)} \\
&\qquad = 0.
\end{aligned}$$

Cela montre également l'hérédité lors d'une mise à jour de type 2, qui est un cas particulier de celle-ci. Montrons maintenant l'hérédité de $\mathcal{H}$ lors d'une mise à jour de type 3. Dans ce cas, nous avons $u_{0,i+1}^{(i)} = u_{1,i+1}^{(i)} = 0$, ce qui signifie que les deux couples sont déjà solutions de $\mathcal{H}(r+1)$. (Ils vérifient $\mathcal{H}(r)$ par construction, et $\mathcal{H}(r+1)$ car $u_{0,i+1}^{(i)} = u_{1,i+1}^{(i)} = 0$.) En multipliant $N_1^{(r)}$ et $W_1^{(r)}$ par X, le couple reste solution.

Montrons maintenant que le couple (N, W) retourné par l'algorithme vérifie la condition d'interpolation. Si l'algorithme se termine après avoir traité tous les points (g_i, y_i), alors nous avons vu que les deux couples $(N_0^{(n)}, W_0^{(n)})$ et $(N_1^{(n)}, W_1^{(n)})$ vérifient tous deux la condition d'interpolation. Sinon, il y a eu une sortie prématurée. Notons r le rang de cette étape, l'algorithme retourne alors le couple $(N_1^{(r)}, W_1^{(r)})$. Il vérifie la condition d'interpolation pour les points $(g_i, y_i), i \leqslant r$ par construction. Pour tous les autres points $(g_i, y_i), i > r$, le défaut $u_{1,i+1}^{(i)}$ correspondant est nul (c'est ce qui provoque la sortie prématurée). Nous avons alors

$$\forall i \in [[r+1; n]],\ \mathcal{L}_{N_1^{(r)}}(g_i) - \mathcal{L}_{W_1^{(r)}}(y_i) = 0$$

ce qui montre que le couple $(N_1^{(r)}, W_1^{(r)})$ vérifie bien la condition d'interpolation pour tous les points. □

Lemme 36. *Soient n, k, t, g et y comme dans l'énoncé du théorème 34. Alors les degrés des θ-polynômes N et W en sortie de l'algorithme 6 vérifient les bornes supérieures du problème de Reconstruction Linéaire* $\mathrm{RL}(n, k, t, g, y)$*, c'est-à-dire :*

$$\deg(N) < k + t \text{ et } \deg(W) \leqslant t.$$

Démonstration. Lors des mises à jour, les degrés des polynômes évoluent de la façon suivante :

	mise à jour de type 1	de type 2	de type 3
$\deg(N_0^{(r+1)})$	$\leqslant \max(\deg(N_0^{(r)}), \deg(N_1^{(r)}))$	$= \deg(N_1^{(r)})$	$= \deg(N_1^{(r)})$
$\deg(W_0^{(r+1)})$	$\leqslant \max(\deg(W_0^{(r)}), \deg(W_1^{(r)}))$	$= \deg(W_1^{(r)})$	$= \deg(W_1^{(r)})$
$\deg(N_1^{(r+1)})$	$= \deg(N_0^{(r)}) + 1$	$= \deg(N_0^{(r)}) + 1$	$= \deg(N_0^{(r)}) + 1$
$\deg(W_1^{(r+1)})$	$= \deg(W_0^{(r)}) + 1$	$= \deg(W_0^{(r)}) + 1$	$= \deg(W_0^{(r)}) + 1$

Il est alors possible de calculer une borne supérieure des degrés des polynômes tout au long de l'algorithme. Ces bornes sont données dans le tableau suivant.

	$\deg(N_0^{(r)})$	$\deg(W_0^{(r)})$	$\deg(N_1^{(r)})$	$\deg(W_1^{(r)})$
$r-k=0$	k	$-\infty$ (0)	$k-1$	0
$r-k=1$	k	1	k	0
$r-k=2$	$k+1$	1	k	1
$r-k=3$	$k+1$	2	$k+1$	1
$r-k=4$	$k+2$	2	$k+1$	2
...				
$r-k$	$k+\lfloor\frac{r-k}{2}\rfloor$	$\lfloor\frac{r-k+1}{2}\rfloor$	$k-1+\lfloor\frac{r-k+1}{2}\rfloor$	$\lfloor\frac{r-k}{2}\rfloor$
...				
$r-k=n-k$ pair	$k+t$	t	$k-1+t$	t
$r-k=n-k$ impair	$k+t$	$t+1$	$k+t$	t

Lorsque le dernier tour de l'algorithme est terminé, l'un des deux couples vérifie les conditions de degré. C'est ce couple qui est renvoyé en sortie.

Remarquons également que ces bornes sont strictement croissantes au cours de l'algorithme. Ainsi, dans le cas d'une sortie prématurée, le couple retourné correspond à un des couples intermédiaires $(N_1^{(r)}, W_1^{(r)})$, les polynômes sont donc de degrés inférieurs à ceux des polynômes retournés lors du dernier tour, et vérifient donc également la condition de degré. □

Lemme 37. *Soient n, k, t, g et y comme dans l'énoncé du théorème 34. Alors le θ-polynôme W en sortie de l'algorithme 6 est non nul.*

Démonstration. Le couple (N, W) retourné par l'algorithme vérifiant les conditions d'interpolation, si W était nul, alors nous aurions

$$\forall i \in [[1; n]],\ \mathcal{L}_N(g_i) = \mathcal{L}_W(y_i) = 0.$$

Les g_i étant linéairement indépendants, N est soit nul, soit de degré n. Or, la borne supérieure que nous avons vue précédemment interdit d'avoir $\deg(N) = n$ (Sauf dans le cas particulier où $k = n$ et $t = 0$. Dans ce cas, il s'agit juste de calculer un polynôme interpolateur. L'algorithme se réduisant dans ce cas à l'initialisation, les degrés sont connus et nous avons bien $W \neq 0$.) Enfin, nous concluons en montrant que N et W ne peuvent être nuls simultanément. Pour cela, nous dressons la liste exhaustive des degrés possibles pour les quatre polynômes au cours de l'algorithme dans le schéma 4.1. Nous pouvons voir sur ce schéma que le degré de n'importe lequel des polynômes peut chuter (le polynôme est nul si le degré chute à $-\infty$), mais que les degrés des deux polynômes d'un même couple ne peuvent pas chuter simultanément. □

4.3 Complexité

Le but de cette section est de déterminer le nombre d'opérations élémentaires que nous devons effectuer au cours de l'algorithme. Rappelons que cet algorithme est le constituant principal des algorithmes de décodage que nous avons vus. En effet, les trois modèles que nous avons considérés s'y ramènent, comme indiqué sur la figure 4.2.

Ces opérations élémentaires sont au nombre de quatre :

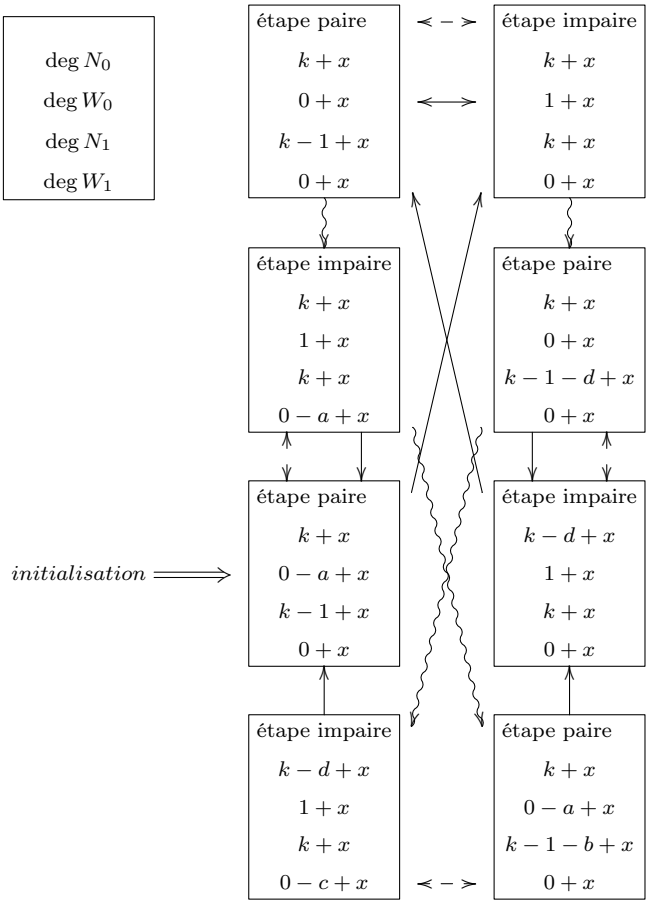

FIGURE 4.1 – Degrés possibles des polynômes au cours de l'algorithme 6. Cette liste exhaustive de tous les degrés possibles montre que les deux polynômes d'un même couple ne peuvent être simultanément nuls. La quantité x permet de regrouper les étapes similaires. Elle représente l'avancement de l'algorithme et augmente tous les deux tours. Les flèches continues indiquent les mises à jours de type 1 sans perte de degré, les flèches ondulées les mises à jour de type 1 dans lesquelles le degré du nouveau polynôme diminue et enfin, les flèches en pointillés indiquent les mises à jour de type 2 ou 3.

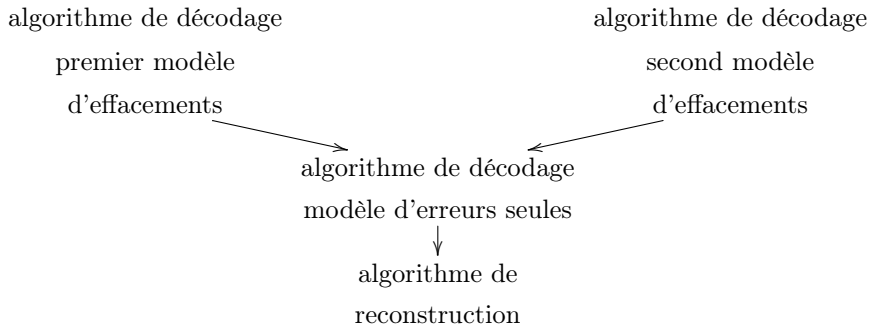

FIGURE 4.2 – Les algorithmes de décodage pour les modèles d'effacements se basent sur l'algorithme de décodage pour le modèle d'erreurs seules, lui-même basé sur l'algorithme de reconstruction.

- addition dans L,
- multiplication dans L,
- division dans L,
- utilisation de θ.

Dans un premier temps, nous allons déterminer la complexité des opérations usuelles sur les θ-polynômes. Nous calculerons ensuite la complexité de l'algorithme de reconstruction seul. Enfin, nous le replacerons dans son contexte en calculant la complexité du décodage et de la gestion des effacements.

4.3.1 Complexité des opérations sur les θ-polynômes

L'algorithme de Reconstruction se compose essentiellement d'opérations sur les θ-polynômes, qu'elles soient usuelles (évaluations, additions et multiplications) ou plus complexes (calcul d'annulateur, d'interpolateur). Nous allons décrire le nombre d'opérations dans L requises par ces différentes opérations dans $L[X;\theta]$.

Soient $A, B \in L[X;\theta]$ des θ-polynômes de degrés respectifs a et b, et soit $c \in L$. Les coûts des opérations usuelles sont résumés dans le tableau suivant.

opération	additions	multiplications	utilisations de θ	divisions
$A + B$	$1 + \min(a, b)$	0	0	0
$A \cdot B$ (si unitaires)	ab	$(1 + a)(1 + b)$ $(1 + a)(b)$	$a(1 + b)$	0
$\mathcal{L}_A(c)$ (si unitaires)	a	$a + 1$ a	a	0
division euclidienne $A = B \cdot Q + R$ (si unitaires)	$(a - b)b$	$(a - b)(b + 1)$ $(a - b)(b)$	$(a - b)(2b)$	$(a - b)$ 0

Remarque 13. *Lors des multiplications, un des facteurs est de degré* 1*, aussi nous pouvons nous contenter de l'algorithme naïf.*

4.3.2 Complexité de l'algorithme de Reconstruction

Nous allons maintenant calculer une borne supérieure du nombre d'opérations requises par l'algorithme de Reconstruction. Pour cela, nous allons compter le nombre d'opérations effectuées lors de l'initialisation, lors de la boucle principale, et lors de la boucle secondaire. Nous en déduirons le nombre d'opérations requises lorsque l'algorithme traite tous les points (g_i, y_i) et lors d'une sortie prématurée.

Initialisation L'initialisation revient à l'utilisation de l'algorithme 2 vu dans le chapitre 2.

Proposition 38. *Soient $x_1, \ldots, x_s$ des éléments K-linéairement indépendants de L et $y_1, \ldots, y_s$ des éléments de L. Alors nous pouvons calculer $\mathcal{A}_{\langle x_1,\ldots,x_s\rangle}$ et $\mathcal{I}_{[x_1,\ldots,x_s];[y_1,\ldots,y_s]}$ avec l'algorithme 2 avec la complexité suivante :*

algorithme 2	*additions*	*multiplications*	*utilisations de θ*	*divisions*
calcul de $\mathcal{A}$ et $\mathcal{I}$	$2s^2 - 2s$	$2s^2 - 3s$	$1.5s^2 - 1.5s$	$2s$
calcul de $\mathcal{A}$ seul	$s^2 - s$	$1.5s^2 + 1.5s$	$s^2 s$	s

Démonstration. Nous pouvons facilement calculer le degré des polynômes, puisqu'il augmente de 1 à chaque itération. Ainsi, $\deg(\mathcal{A}^{(r)}) = r$ et $\deg(\mathcal{I}^{(r)}) = r - 1$. Ainsi, le calcul de $\mathcal{A}^{(r+1)}$ et $\mathcal{I}^{(r+1)}$ nécessite $4r$ additions, $4r - 1$ multiplications, $3r$ utilisations de θ et 2 divisions. ($\mathcal{A}^{(r)}$ est unitaire, ce qui fait gagner 2 produits par itération.) En sommant ces quantités pour $0 \leqslant r \leqslant s - 1$, nous obtenons la complexité annoncée. □

Remarque 14. *$\mathcal{A}^{(r+1)}$ et $\mathcal{I}^{(r+1)}$ sont tous deux évalués en x_{r+i}. Aussi, nous calculons deux fois les itérés $\theta(x_{r+i}), \theta^2(x_{r+i}), \ldots$ Il serait donc possible de gagner $r - 1$ utilisations de θ par itération en mémorisant ces valeurs. Dans ce cas, il serait nécessaire de stocker s coefficients.*

Boucle principale Une itération est essentiellement constituée de deux étapes. La première est le calcul des défauts u_0 et u_1. La seconde est la mise à jour des polynômes. Entre ces deux étapes, il peut y avoir un appel à la boucle secondaire, que nous traiterons dans le prochain paragraphe.

Nous allons compter le nombre maximum d'opérations lors d'une itération. Aussi, nous supposons que la mise à jour est de type 1 (la plus couteuse) et que les polynômes en début d'itération sont de degré maximal.

Au début du tour $i \in [[k; n-1]]$, nous disposons des polynômes $N_0^{(i)}$, $W_0^{(i)}$, $N_1^{(i)}$ et $W_1^{(i)}$, de degrés respectifs $k + \lfloor \frac{i-k}{2} \rfloor$, $\lfloor \frac{i+1-k}{2} \rfloor$, $k - 1 + \lfloor \frac{i+1-k}{2} \rfloor$ et $\lfloor \frac{i-k}{2} \rfloor$.

Nous commençons par calculer les défauts :

$$u_{0,i+1}^{(i)} = \mathcal{L}_{N_0^{(i)}}(g_r) - \mathcal{L}_{W_0^{(i)}}(y_r), \tag{4.1}$$

$$u_{1,i+1}^{(i)} = \mathcal{L}_{N_1^{(i)}}(g_r) - \mathcal{L}_{W_1^{(i)}}(y_r). \tag{4.2}$$

Cela nécessite l'évaluation des quatre polynômes ainsi que deux additions.

Nous appliquons ensuite les formules de mise à jour :

$$N_0^{(i+1)} = (X - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}}1) \cdot N_1^{(i)}, \tag{4.3}$$

$$W_0^{(i+1)} = (X - \frac{\theta(u_{1,i+1}^{(i)})}{u_{1,i+1}^{(i)}}1) \cdot W_1^{(i)}, \tag{4.4}$$

$$N_1^{(i+1)} = N_0^{(i)} - \frac{u_{0,i+1}^{(i)}}{u_{1,i+1}^{(i)}} \cdot N_1^{(i)}, \tag{4.5}$$

$$W_1^{(i+1)} = W_0^{(i)} - \frac{u_{0,i+1}^{(i)}}{u_{1,i+1}^{(i)}} \cdot W_1^{(i)}. \tag{4.6}$$

Cela nécessite deux produits de polynômes (produits po dans le tableau), deux multiplications d'un polynôme par un scalaire (pdt scal-po), deux sommes de polynômes (somme po), ainsi qu'une utilisation de θ et deux divisions.

La contribution de toutes ces opérations est résumée dans le tableau suivant, ainsi que le nombre total d'opérations lors du tour i.

opérations	additions	multiplications	utilisations de θ	divisions
évaluations	$\deg(N_0)$ $+\deg(W_0)$ $+\deg(N_1)$ $+\deg(W_1)$	$\deg(N_0)$ $+\deg(W_0)$ $+\deg(N_1)$ $+\deg(W_1)+4$	$\deg(N_0)$ $+\deg(W_0)$ $+\deg(N_1)$ $+\deg(W_1)$	
produit po	$\deg(N_1)$ $+\deg(W_1)$	$\deg(N_1)$ $+\deg(W_1)+2$	$\deg(N_1)$ $\deg(W_1)+2$	
pdt scal-po		$\deg(N_1)$ $+\deg(W_1)+2$		
sommes po	$\max(\deg(N_0),\deg(N_1))+1$ $+\max(\deg(W_0),\deg(W_1))+1$			
divers	2		1	2
total	$4i-2$	$4i+1$	$3i-1$	2

Boucle secondaire Si nous devons parcourir cette boucle lors du tour r, nous disposons des polynômes $N_0^{(i)}$, $W_0^{(i)}$, $N_1^{(i)}$ et $W_1^{(i)}$, de degrés respectifs $k + \lfloor \frac{i-k}{2} \rfloor$, $\lfloor \frac{i+1-k}{2} \rfloor$, $k - 1 + \lfloor \frac{i+1-k}{2} \rfloor$ et $\lfloor \frac{i-k}{2} \rfloor$.

Cette boucle consiste à calculer $u_{0,j}^{(i)}$ et $u_{1,j}^{(i)}$ pour $j > i+1$ jusqu'à trouver une valeur $j \in [[i+2; n]]$ telle que $u_{0,j}^{(i)} = 0$ et $u_{1,j}^{(i)} \neq 0$. Comme lors de la boucle précédente, le calcul de chaque paire $(u_{0,j}^{(i)}, u_{1,j}^{(i)})$ nécessite l'évaluation des quatre polynômes ainsi que deux additions, soit :

additions	multiplications	utilisations de θ
$\deg(N_0)+\deg(W_0)$ $+\deg(N_1)+\deg(W_1)+2$	$\deg(N_0)+\deg(W_0)$ $+\deg(N_1)+\deg(W_1)+4$	$\deg(N_0)+\deg(W_0)$ $+\deg(N_1)+\deg(W_1)$
$2i+1$	$2i+3$	$2i-1$

Il y a au plus $n - i - 1$ telles évaluations lors de la boucle i.

Complexité totale Nous pouvons maintenant exprimer la complexité totale de l'algorithme de Reconstruction. Considérons dans un premier temps qu'il n'y a pas de sortie prématurée. Il faut alors prendre en compte l'initialisation, ainsi que la boucle principale pour $i \in [[k; n-1]]$, d'où les nombres suivants d'opérations :

- $(2k^2 - 2k) + (2k + 2n - 8)(n - k)$ additions dans L,
- $(2k^2 - 3k) + (2k + 2n - 9)(n - k)$ multiplications dans L,
- $(1.5k^2 - 1.5k) + (1.5k + 1.5n - 5.5)(n - k)$ utilisations de θ,
- $2k + 2(n - k) = 2n$ divisions dans L.

Théorème 39. *En ne conservant que le terme dominant pour chaque opération, l'algorithme de Reconstruction nécessite*

- $2n^2 + O(n)$ *additions dans* L,
- $2n^2 + O(n)$ *multiplications dans* L,
- $1.5n^2 + O(n)$ *utilisations de* θ,
- $2n$ *divisions dans* L.

A cela peuvent s'ajouter des évaluations supplémentaires, si nous devons entrer dans la boucle secondaire au cours de l'algorithme. Le pire cas serait de devoir calculer *tous* les défauts possibles lors de *chaque* itération de la boucle principale. Le nombre d'opérations supplémentaires serait alors de :

$$\sum_{i=k-1}^{n-2} (n - i - 1)C_i$$

où C_i est le coût d'une évaluation, et vaut respectivement $2i + 1$, $2i + 3$ et $2i - 1$ pour l'addition, la multiplication et l'utilisation de θ. En ne conservant que le terme principal, nous obtenons

$$(n - k)\left(\frac{n^2 + 5nk + 2k^2}{3}\right) + O(n^2)$$

additions, multiplications et utilisations de θ supplémentaires.

Remarque 15. *L'algorithme est donc quadratique dans le meilleur cas, et cubique dans le pire cas. Néanmoins, considérer qu'il faut calculer* tous *les défauts possibles à* chaque *itération est une estimation très grossière. En effet, il est rare, en pratique, de devoir calculer une évaluation supplémentaire (sauf sortie prématurée). Il est donc raisonnable de négliger ce surcout et d'affirmer que l'algorithme est quadratique. De plus, nous verrons une variante dans la section 4.4.2 dans laquelle les défauts sont gérés autrement. Cette version est affranchie des évaluations supplémentaires, ce qui la rend quadratique dans tous les cas, mais sa complexité est supérieure.*

Dans un second temps, regardons la complexité de l'algorithme dans le cas d'une sortie prématurée. En notant $2t$ le nombre d'itérations réalisées, nous devons prendre en compte l'initialisation ainsi que les $2t$ premières itérations de la boucle principale ($i \in [[k-1; k+2t-2]]$), ainsi que les $n - k - 2t$ évaluations supplémentaires qui permettent de détecter la sortie prématurée. Le nombre d'opérations est alors :

- $2k^2 + 2(2t)(2k + 2t) + (n - k - 2t)(2k + 4t) + O(n)$ additions dans L,
- $2k^2 + 2(2t)(2k + 2t) + (n - k - 2t)(2k + 4t) + O(n)$ multiplications dans L,
- $1.5k^2 + 1.5(2t)(2k + 2t) + (n - k - 2t)(2k + 4t) + O(n)$ utilisations de θ,
- $2k + 2(2t)$ divisions dans L.

Remarque 16. *Le nombre de calculs est toujours inférieur en cas de sortie prématurée. En effet, les inégalités suivantes sont équivalentes.*

$$\begin{aligned}
2k^{+}2(2t)(2k+2t)+(n-k-2t)(2k+4t) &\leqslant 2n^2,\\
k^{+}(2t)(2k+2t)+(n-k-2t)(k+2t) &\leqslant k^2+(n-k)(n+k),\\
2tk+(n-k)(k+2t) &\leqslant (n-k)(n+k),\\
2tk+(n-k)(2t) &\leqslant (n-k)n,\\
2tn &\leqslant (n-k)n,\\
2t &\leqslant n-k.
\end{aligned}$$

Elles sont donc vraies dès que le poids de l'erreur est inférieur à la capacité de correction du code.

Comme précédemment, des évaluations supplémentaires peuvent avoir lieu, et dans le pire cas, la complexité devient cubique.

Remarque 17. *Nous pouvons remarquer que parmi les quatre polynômes $N_0^{(i)}$, $W_0^{(i)}$, $N_1^{(i)}$ et $W_1^{(i)}$, deux sont unitaires. Ainsi, le nombre d'opérations peut être diminué de 2 multiplications par itération, soit d'un total de $2(n-k)$ multiplications.*

4.3.3 Complexité des algorithmes de décodage

Nous allons maintenant donner la complexité des algorithmes de décodage, pour les trois modèles d'erreurs que nous avons vus. Dans chaque cas, nous considèrerons uniquement le nombre de multiplications et de divisions. (Les nombres d'additions et d'utilisations de θ sont majorés par le nombre de multiplications.) Nous supposerons que tous les paramètres du code (k, t, s_ℓ, s_c) augmentent avec sa longueur n.

Proposition 40. *Le nombre d'opérations lors de l'algorithme 3 de décodage d'un code $[n,k,d]$ dans le modèle d'erreur seule est de :*
- $2n^2+k\lfloor\frac{n-k}{2}\rfloor+O(n)$ *multiplications dans L,*
- $2n-k-1$ *divisions dans L.*

Démonstration. L'algorithme de décodage se compose de l'algorithme de Reconstruction puis de la division de deux θ-polynômes (de degrés $k+t-1$ et t). La complexité de ces deux étapes est donnée dans le tableau suivant.

étape	multiplications	divisions
Reconstruction	$2n^2$	$2(n-k)$
Division	$(k-1)(t+1)$	$k-1$

□

Proposition 41. *Le nombre d'opérations lors de l'algorithme 4 de décodage d'un code $[n,k,d]$ dans le premier modèle d'effacements est de :*
- $2n'^2+\frac{k'(n'-k')}{2}+s_\ell(n'+k'+\frac{s_\ell}{2})+O(n)$ *multiplications dans L,*
- $2n'$ *divisions dans L,*

où s_ℓ et s_c désignent respectivement le nombre d'effacements de lignes et de colonnes, et où $n'=n-s_c$ et $k'=k+s_\ell$ désignent les paramètres du code obtenu après avoir géré les effacements.

Démonstration. L'algorithme de décodage se compose des étapes suivantes : calcul de $\mathcal{V}$, calcul des z_i, algorithme de décodage (modèle d'erreurs seules), division du polynôme obtenu par $\mathcal{V}$, dont la complexité est résumée dans le tableau suivant.

étape	multiplications	divisions
calcul de $\mathcal{V}$	$\frac{3}{2}(s_\ell^2 + s_\ell)$	s_ℓ
calcul des z_i	$n'(s_\ell + 1)$	0
reconstruction	$2n'^2$	$2(n' - k')$
division$W \backslash N$	$(k' - 1)\lfloor \frac{n'-k'}{2} \rfloor)$	$k' - 1$
division $F \backslash \mathcal{V}$	$k(s_\ell + 1)$	k

□

Proposition 42. *Le nombre d'opérations lors de l'algorithme 5 de décodage d'un code $[n, k, d]$ dans le second modèle d'effacements est de :*

- $2n'^2 + \frac{k'(n'-k')}{2} + s_\ell(n' + k' + \frac{s_\ell}{2}) + O(n)$ *multiplications dans L,*
- $2n'$ *divisions dans L,*
- $s_c^2 n$ *produits dans K,*
- $2s_c n$ *produits d'un élément de L par un élément de K,*

où s_ℓ et s_c désignent respectivement le nombre d'effacements de lignes et de colonnes, et où $n' = n - s_c$ et $k' = k + s_\ell$ désignent les paramètres du code obtenu après avoir géré les effacements.

Démonstration. Seule la façon de gérer les effacements de colonnes diffère de l'algorithme précédent. La gestion des effacements consiste à échelonner la matrice B_c par colonnes, en répercutant ces opérations sur les vecteurs g et y. L'échelonnement requiert s_c étapes, comprenant chacune n transvections. Lors de chaque transvection, il faut modifier une colonne de B_c (s_c coefficients dans K), et répercuter l'opération sur g et y (2 coefficients dans L). □

4.4 Améliorations

Le but de cette section est de présenter des variantes de l'algorithme que nous avions étudié dans le début de ce chapitre. Les trois variantes que nous allons voir portent sur des différents éléments de l'algorithme. Elles sont donc indépendantes les unes des autres. La première est une version sans division, adaptée à des codes définis sur des anneaux. La seconde permet de calculer avec des polynôme de plus bas degré, faisant ainsi diminuer le nombre de calculs. Dans la troisième version, nous initialisons et mettons à jour les défauts, au lieu de les calculer quand nous en avons besoin. Le nombre de calculs est plus élevé que dans la version de base, mais le nombre d'opérations est toujours quadratique. Pour chaque variante, nous préciserons l'impact sur la complexité.

4.4.1 Une version sans division

L'intérêt de cette variante est qu'aucune division n'est nécessaire durant l'algorithme de Reconstruction. Elle est donc particulièrement adaptée à des codes dont les coefficient sont dans un anneau, comme par exemple les anneaux d'entiers.

Pour obtenir une version sans divisions, il faut modifier l'initialisation et les mises à jour.

Lors de l'initialisation, nous devons calculer un polynôme annulateur et un polynôme interpolateur. Il est possible de calculer un polynôme annulateur sans faire de divisions (il n'est alors plus

unitaire), mais pour calculer ainsi un polynôme interpolateur, il faut assouplir la définition. Pour des éléments K-linéairement indépendants $x_1, \ldots, x_s$ et des éléments quelconques $y_1, \ldots, y_s$ dans L, nous cherchons un polynôme I vérifiant :

$$\mathcal{L}_I(x_i) = \lambda y_i, 1 \leqslant i \leqslant s.$$

Nous pouvons calculer les polynômes annulateur et interpolateur, ainsi que le coefficient λ, avec l'algorithme 8.

Algorithm 8 Polynômes annulateur et interpolateur (sans division)

Input: $x_1, \ldots, x_s \in L$ K-linéairement indépendants
$y_1, \ldots, y_s \in L$

Output: $\mathcal{A}(X)$ tel que $\mathcal{L}_{\mathcal{A}}(x_i) = 0$
$\mathcal{I}(X)$ et λ tels que $\mathcal{L}_{\mathcal{I}}(x_i) = \lambda y_i$

0: $\mathcal{A} := 1$
0: $\mathcal{I} := 0$
0: $\lambda := 1$
0: **for** $1 \leqslant i \leqslant s$ **do**
0: $\quad \mathcal{I} := \mathcal{L}_{\mathcal{A}}(x_i) \cdot \mathcal{I} + (\lambda y_i - \mathcal{L}_{\mathcal{I}}(x_i)) \cdot \mathcal{A}$
0: $\quad \lambda := \mathcal{L}_{\mathcal{A}}(x_i) \cdot \lambda$
0: $\quad \mathcal{A} := (\mathcal{L}_{\mathcal{A}}(x_i)X - \theta(\mathcal{L}_{\mathcal{A}}(x_i))) \cdot \mathcal{A}$
0: **end for**
1: **return** $\mathcal{A}, \mathcal{I}, \lambda =$0

La seconde modification à apporter concerne les formules de mise à jour, celles-ci sont sont résumées dans le suivant.

$u_{0,i+1}^{(i)}$	$u_{1,i+1}^{(i)}$	$N_0^{(i+1)}$	$N_1^{(i+1)}$	type
$\neq 0$	$\neq 0$	$(u_{0,i+1}^{(i)}X - \theta(u_{0,i+1}^{(i)})) \cdot N_0^{(i)}$	$u_{0,i+1}^{(i)}N_1^{(i)} - u_{1,i+1}^{(i)}N_0^{(i)}$	1
$\neq 0$	$= 0$	$(u_{0,i+1}^{(i)}X - \theta(u_{0,i+1}^{(i)})) \cdot N_0^{(i)}$	$N_1^{(i)}$	2
$= 0$	$= 0$	$X \cdot N_0^{(i)}$	$N_1^{(i)}$	3
$= 0$	$\neq 0$	no update		4

Ainsi, il n'y a aucune division lors de l'algorithme de reconstruction. Les seules divisions se produirons lors de la division de N par W. Ce dernier n'étant plus unitaire, des divisions (dans L) seront nécessaires. Néanmoins, il s'agira de divisions exactes, et elles ne feront pas apparaitre de fractions.

Théorème 43. *Il est possible de modifier l'algorithme afin de ne pas avoir de division lors de la Reconstruction. Seule la division (dans $L[X;\theta]$) de N par W demandera des divisions (dans L). Cette modification apporte $O(n^2)$ multiplications supplémentaires. (le nombre de multiplications est globalement multiplié par $1,5$.)*

4.4.2 Polynômes de plus bas degré

Cette variante a pour but de diminuer le degré des polynômes manipulés, permettant ainsi une diminution de la complexité.

Nous pouvons remarquer que $N_0^{(i)}$ et $N_1^{(i)}$ sont mis à jour en utilisant uniquement des additions et des multiplications à gauche. Aussi, à n'importe quelle étape de l'algorithme, il est possible de les exprimer comme combinaisons polynomiales de leurs valeurs à l'initialisation (à savoir $\mathcal{A}$ et $\mathcal{I}$). Nous avons donc :

$$\forall \ell \in \{0;1\},\ N_\ell^{(i)} = P_\ell^{(i)} \cdot \mathcal{A} + Q_\ell^{(i)} \cdot \mathcal{I}$$

où $P_\ell^{(i)}, Q_\ell^{(i)} \in L[X;\theta]$.

De plus, $P_\ell^{(i)}$ et $Q_\ell^{(i)}$ sont mis à jour avec la même formule que $N_\ell^{(i)}$ et $W_\ell^{(i)}$. Ces polynômes sont initialisés par $P_0^{(k)} = Q_1^{(k)} = W_1^{(k)} = 1$ et $P_1^{(k)} = Q_0^{(k)} = W_0^{(k)} = 0$. Par conséquent, nous avons $Q_\ell^{(i)} = W_\ell^{(i)}$ puisqu'ils ont même initialisation et même mises à jour. Cela ne nous dispense pas de la mise à jour, car nous avons besoin de $\mathcal{A}$ et $\mathcal{I}$ pour le calcul des $u_{\ell,i+1}^{(i)}$ et pour la division de N par W dans l'algorithme de décodage.

Nous pouvons donc remplacer $N_\ell^{(i)}$ par $P_\ell^{(i)}$, dont le degré est $\deg P_\ell^{(i)} = \deg N_\ell^{(i)} - k$. Cela implique de devoir changer le calcul des $u_{\ell,i+1}^{(i)}$:

$$u_{\ell,i+1}^{(i)} = \mathcal{L}_{P_\ell^{(i)}}(\mathcal{L}_{\mathcal{A}}(g_{i+1})) + \mathcal{L}_{W_\ell^{(i)}}(\mathcal{L}_{\mathcal{I}}(g_{i+1}) - y_{i+1})$$

Cette modification est quasiment sans effet sur la complexité (le changement est linéaire pour tout l'algorithme).

La division finale de l'algorithme de décodage doit également être modifiée et devient :

$$W \backslash N = W \backslash (P\,\mathcal{A}) + \mathcal{I}$$

Lors de la division de $P\,\mathcal{A}$ par W, les polynômes sont de même degrés que dans la division de la version basique. Nous devons juste compter $(k+1)t$ opérations supplémentaires lors de ce calcul. (Cela fait quand même moins d'opération que si nous recalculions N et W pour ensuite les diviser.)

Le fait de travailler avec un polynôme de plus bas degré permet de diminuer le nombre de calcul lors des itérations. Le gain est de $2k(n-k)$ additions, $k(n-k)$ utilisations de θ et $2k(n-k)$ multiplications ($4k(n-k)$ dans la versions sans division).

Théorème 44. *Cette version est toujours avantageuse comparée à la version de base. Le nombre de multiplication est diminué de* $1.5k(n-k)$ *pour l'algorithme de base et de* $3.5k(n-k)$ *pour la version sans divisions.*

4.4.3 Mise à jour des défauts

Au lieu de calculer $u_{\ell,j}^{(i)}$ au tour i, nous calculons tous les $u_{\ell,j}^{(k)}$, $\ell \in \{0;1\}$, $j \in [[k+1;n]]$ à l'initialisation, puis nous les mettons à jour à chaque tour, en même temps que $N_\ell^{(i)}$ (ou $P_\ell^{(i)}$) et $W_\ell^{(i)}$ (la formule de mise à jour est la même).

Le nombre d'opérations supplémentaires est négligeable pour l'algorithme de base (de l'ordre de $O(n-k)$), mais pour la version sans division, il est de l'ordre de $O(n^2-k^2)$ opérations supplémentaires.

Le principal avantage de cette version est d'éviter les calculs supplémentaires lors des mises à jour de type 4. En effet, dans ce cas, il suffit de regarder les valeurs des défauts pour constater s'ils sont égaux ou non à 0. Une fois qu'une valeur acceptable est trouvée, il suffit de les échanger. Ainsi, cette variante est toujours quadratique.

Théorème 45. *Cette variante est toujours quadratique. La version avec divisions n'est pas affectée en termes de complexité, mais dans la version sans division, il y a* $O(n^2-k^2)$ *opérations supplémentaires.*

Ce surcout de $O(n^2 - k^2)$ opérations est à mettre en balance avec le nombre d'évaluations supplémentaires requises dans l'algorithme de base.

Chapitre 5

Codes de Gabidulin généralisés et finis

La raison pour laquelle nous avons généralisé les codes de Gabidulin à des corps infinis n'est pas seulement théorique. Nous souhaitons les utiliser pour concevoir des codes utilisables en pratique, en particulier pour des transmissions de type MIMO, où les éléments transmis sont des nombres complexes. Le corps des complexes n'est pas adapté à la définition d'un code de Gabidulin, aussi avons du considérer des extensions $K \hookrightarrow L$ de corps de nombres.

Dans ce chapitre, nous allons en fait nous restreindre aux anneaux d'entiers $\mathcal{O}_{\mathrm{K}}$ et $\mathcal{O}_{\mathrm{L}}$ de ces corps. Cette structure est adaptée à la construction de codes pour les raisons suivantes.

- Le codage de l'information dans les mots du code fait que seul un nombre fini de mots seront utilisés. En particulier, les coefficients de ces mots seront dans une famille finie (appelée constellation).
- Les anneaux d'entiers sont des structures adaptées à l'arithmétique, puisqu'ils sont stables par somme et produit. Ainsi, en choisissant des symboles d'information et un code à support dans $\mathcal{O}_{\mathrm{L}}$, les coefficients des mots du code seront également dans $\mathcal{O}_{\mathrm{L}}$.
- Les entiers sont plus simples à manipuler et à représenter que des éléments quelconques du corps.
- Si nous supposons que l'erreur est à coefficients dans $\mathcal{O}_{\mathrm{L}}$, le mot reçu est lui aussi à coefficients dans $\mathcal{O}_{\mathrm{L}}$, et donc le décodage peut se faire dans cet anneau. La seule opération problématique est alors la division, mais nous disposons d'un algorithme de Reconstruction sans divisions (voir le premier paragraphe de la section 4.4).
- Enfin, un des critères utilisés pour comparer les performances des codes espace-temps (voir chapitre 6) est basé sur le déterminant. Ainsi, si les mots du code sont dans un anneau d'entiers, ces déterminants aussi, et nous pouvons alors minorer tout déterminant non nul (par 1). Cette propriété des anneaux d'entiers est utilisée dans diverses constructions ([], []) pour garantir les performances des codes.

Bien que le code ne comporte qu'un nombre fini de mots, leurs coefficients sont des éléments de $\mathcal{O}_{\mathrm{L}}$. À ce titre, nous serons confrontés au problème de la longueur croissante des coefficients. Au fur et à mesure des calculs, les nombres que nous manipulons seront de plus en plus grands, rendant les calculs de plus en plus long. Aussi, nous allons réduire le code, modulo un idéal bien choisi, afin de calculer dans un corps fini.

Dans une première partie, nous observerons la croissance exponentielle des éléments manipulés lors du décodage d'un code de Gabidulin généralisé. Nous verrons alors que le temps de calcul est tellement important qu'il rend les codes de Gabidulin généralisés inutilisables en pratique.

Nous introduirons la réduction des codes dans la deuxième section. Après avoir décrit la réduction d'un code de Gabidulin, nous déterminerons dans quels cas cette réduction est un code de Gabidulin fini.

La troisième partie sera consacrée à la recherche d'idéaux appropriés. Nous fournirons une méthode permettant de trouver ces idéaux, pour les corps de nombres et pour les corps de fractions sur $\mathbb{F}_q$.

Ces résultats ont été évoqués à la conférence ACN (*Algebra, Codes and Network*) en juin 2014.

5.1 Problématique : explosion des coefficients

Considérons un code de Gabidulin généralisé sur une extension

$$\mathbb{Q} \hookrightarrow K \hookrightarrow L.$$

Dans ce chapitre, nous supposerons que le support du code, les symboles d'information et l'erreur (et donc mots du code et mots reçus) sont à coefficient dans $\mathcal{O}_{\mathrm{L}}$, l'anneau des entiers de L. Nous considérerons une $\mathbb{Q}$-base intégrale $\mathcal{B}$ de $\mathcal{O}_{\mathrm{L}}$.

L'algorithme que nous avons vu dans le chapitre 4 est quadratique. Néanmoins, son exécution sur des codes de petits paramètres prend un temps conséquent. Les temps de calcul sont présentés dans l'exemple 20. La raison n'est pas la complexité de l'algorithme, mais l'augmentation de la longueur des éléments manipulés au cours de son exécution, ce qui cause des temps de calcul conséquents.

Définition 20. *Soit L un corps, $\mathcal{O}_{\mathrm{L}}$ son anneau d'entiers et $\mathcal{B}$ une $\mathbb{Q}$-base intégrale de $\mathcal{O}_{\mathrm{L}}$.*

- *La longueur d'un nombre $x \in \mathbb{Z}$ est le nombre de chiffre de x écrit en base 2.*

$$\ell(x) = 1 + \log_2(x)$$

- *Un entier $x \in \mathcal{O}_{\mathrm{L}}$ se décompose dans la base B en $x = \sum_{b_i \in B} x_i B_i$. Sa longueur est la longueur de son plus grand coefficient.*

$$\ell(x) = \sup_i(\ell(x_i))$$

- *La longueur d'un polynôme $P = \sum_i p_i X^i \in \mathcal{O}_{\mathrm{L}}[X]$ est la longueur de son plus grand coefficient.*

$$\ell(P) = \sup_i(\ell(p_i))$$

Exemple 20. *Le tableau suivant donne le temps de calcul requis pour le décodage de codes de Gabidulin généralisés (sur des corps cyclotomiques) pour de petits paramètres (longueur et dimension du code sont inférieures à 16). Les mots d'information, le support du code ainsi que les erreurs sont petits (longueur inférieure à 2). Le temps retenu, en secondes, est la plus grande valeur parmi*

5 *réalisations.*

$n\backslash k$	2	4	6	8	10	12	14
4	0						
6	0.010	0.020					
8	0.030	0.030	0.060				
10	0.060	0.080	0.140	0.260			
12	0.440	0.610	1.240	2.310	4.260		
14	8.310	10.720	18.110	34.560	54.650	87.090	
16	55.12	69.5	115	208	321	465	698

Malgré des entrées (mots reçus) petites et des sorties petites (mots d'information), les polynômes intermédiaires vont augmenter exponentiellement au cours de l'algorithme. Nous pouvons voir dans l'exemple 21 que la taille des éléments est doublée à chaque itération.

Exemple 21. *Reprenons le code de paramètres* $[8, 4, 5]$ *de l'exemple 20. Si nous considérons un mot* f *de longueur* 1 *et une erreur de longueur* 2*, le mot reçu est de longueur* 2*. En revanche, les polynômes* N *et* W *à la sortie de l'algorithme sont de longueur* 191*.*

En effet, supposons pour simplifier que les 4 polynômes du début de l'étape r de l'algorithme 6 sont de même longueur

$$\ell(N_0^{(r)}) = \ell(W_0^{(r)}) = \ell(N_1^{(r)}) = \ell(W_1^{(r)}).$$

Alors les défauts sont de longueur

$$\ell(u_{i,j}^{(r)}) = \max\left(\ell(N_i^{(r)}) + \ell(g_j), \ell(W_i^{(r)}) + \ell(y_j)\right).$$

Les polynômes étant tous de même longueur, et les symboles du mot reçu y et du support g étant supposés petits, nous pouvons écrire

$$\ell(u_{i,j}^{(r)}) \lesssim \ell(N_i^{(r)}).$$

Les polynômes du tour suivant sont obtenus, lors de la mise à jour, comme somme de produits d'un polynôme avec un défaut, d'où les longueurs suivantes :

$$\ell(N_0^{(r+1)}) \lesssim \ell(N_0^{(r)}) + \ell(u_{0,j}^{(r)}) \lesssim 2\ell(N_0^{(r)}).$$

Ainsi, bien que le nombre total d'opérations au cours de l'algorithme soit quadratique, les opérations portant sur des nombres dont la longueur augmente exponentiellement, nous obtenons un temps de calcul particulièrement élevé.

L'outil habituel pour éviter de calculer avec d'aussi grands nombres est de calculer modulo des entiers m_i, puis d'utiliser le théorème chinois pour reconstruire une solution modulo $M = \prod_i m_i$. Les m_i doivent être choisis étrangers deux à deux, et leur produit M doit majorer le résultat attendu. Dans le cas qui nous intéresse, le résultat étant lui-même petit, un unique idéal m_i bien choisi nous permettra de retrouver le polynôme d'information sans recourir au théorème chinois.

Les résultats présentés dans la prochaine section dépasse le cadre pratique de l'amélioration du temps de calcul, puisque nous ferons le lien entre les codes de Gabidulin généralisés et leurs homologues finis.

5.2 Réduction d'un code de Gabidulin

Le but de cette section est de voir la notion de code quotient et leur utilisation pour le décodage.

5.2.1 Quotient d'un code de Gabidulin généralisé

Définition 21. *Soit L un corps et $\mathcal{O}_{\mathrm{L}}$ son anneau d'entiers. Soit I un idéal de $\mathcal{O}_{\mathrm{L}}$ et notons $\overline{x} = x \pmod I$ le résidu de $x \in \mathcal{O}_{\mathrm{L}}$ modulo I. Enfin, soit $\mathcal{C}$ un code correcteur de dimension k et de longueur n défini sur $\mathcal{O}_{\mathrm{L}}$. Le code quotient $\overline{\mathcal{C}}$ est défini par :*

$$\overline{\mathcal{C}} = \{(\overline{c_1}, \ldots, \overline{c_n}) : (c_1, \ldots, c_n) \in \mathcal{C}\}.$$

Exemple 22. *Soit $K \hookrightarrow L$ une extension cyclique de degré m. Notons $\mathcal{O}_{\mathrm{K}}$ et $\mathcal{O}_{\mathrm{L}}$ leurs anneaux d'entiers respectifs et θ un générateur du groupe $\mathrm{Gal}(K \hookrightarrow L)$ restreint à $\mathcal{O}_{\mathrm{L}}$. Soit $\mathcal{P}$ un idéal de $\mathcal{O}_{\mathrm{L}}$. Enfin, soit $\mathcal{C} = \mathrm{Gab}_{\theta,k}(g)$ un code de Gabidulin défini sur l'extension $K \hookrightarrow L$ et restreint à $\mathcal{O}_{\mathrm{L}}$. Alors le code $\overline{\mathcal{C}}$ est :*

$$\overline{\mathcal{C}} = \left\{(\mathcal{L}_{\overline{f}}(\overline{g_1}), \ldots, \overline{f}\,\overline{g_n}) : \overline{f} \in (\mathcal{O}_{\mathrm{L}}/\mathcal{P})[X; \overline{\theta}], \deg(\overline{f}) < k\right\},$$

où $\overline{\theta}$ est défini par :

$$\forall \overline{x} \in \mathcal{O}_{\mathrm{L}}/\mathcal{P}, \overline{\theta}(\overline{x}) = \overline{\theta(x)}.$$

Nous allons maintenant déterminer à quelles conditions le code obtenu dans l'exemple ci-dessus est un code de Gabidulin généralisé.

Pour cela, nous devons entrer dans le cadre dégagé dans le chapitre 4 :

1. le code doit être défini sur une extension de corps de degré m,
2. le support $\overline{g}$ doit être linéairement indépendant sur le corps de base,
3. $\overline{\theta}$ doit être générateur du groupe de Galois de l'extension.

La restriction de $\mathcal{P}$ à $\mathcal{O}_{\mathrm{K}}$ fournit naturellement un idéal $\sqrt{\ }$ de ce dernier. Le premier point implique que $\mathcal{O}_{\mathrm{L}}/\mathcal{P}$ et $\mathcal{O}_{\mathrm{K}}/\sqrt{\ }$ doivent être des corps, et donc que les deux idéaux doivent être maximaux, mais aussi qu'ils forment une extension de degré m. Nous verrons dans la prochaine section que l'idéal $\sqrt{\ }$ doit être un idéal inerte de $\mathcal{O}_{\mathrm{K}}$ pour avoir une extension $\mathcal{O}_{\mathrm{K}}/\sqrt{\ } \hookrightarrow \mathcal{O}_{\mathrm{L}}/\mathcal{P}$ de degré m. Nous décrirons dans cette prochaine section une méthode permettant de déterminer un tel idéal.

Le second point est l'indépendance linéaire du support. la réduction d'un support $g = (g_1, \ldots, g_n) \in L^n$ (K-linéairement indépendant) quelconque n'a aucune raison d'être $(\mathcal{O}_{\mathrm{K}}/\mathfrak{p})$-linéairement indépendant, comme le montre l'exemple suivant.

Exemple 23. *Considérons la K-base $1, \ldots, \alpha^{m-1}$, et soit $\rho \in \mathfrak{p}$. Alors la famille $\left(1, 1 + \rho\alpha, \ldots, \alpha^{m-1}\right)$ est K-libre, mais sa réduction $\left(\overline{1}, \overline{1}, \overline{\alpha}^3, \ldots, \overline{\alpha}^{m-1}\right)$ est liée sur $\mathcal{O}_{\mathrm{K}}/\mathfrak{p}$.*

Si le support est quelconque, il faut vérifier sa compatibilité avec l'idéal choisi. Néanmoins, il existe un support libre compatible avec tout choix d'idéal.

Proposition 46. *Soit $K \hookrightarrow L = K[\alpha]$ une extension de corps, et notons $\mathcal{O}_{\mathrm{K}}$ et $\mathcal{O}_{\mathrm{L}}$ leurs anneaux d'entiers respectifs. Soit $\mathfrak{P}$ un idéal de $\mathcal{O}_{\mathrm{L}}$ et notons $\mathfrak{p}$ son intersection avec $\mathcal{O}_{\mathrm{K}}$. Considérons le support $g = (1, \ldots, \alpha^{m-1})$. Si $\alpha \notin \mathcal{P}$, la réduction modulo $\mathfrak{P}$ de g est $\mathcal{O}_{\mathrm{K}}/\mathfrak{p}$-libre.*

Démonstration. Pour montrer l'indépendance des éléments, considérons une relation

$$\overline{\mu}_0\overline{1} + \overline{\mu}_1\overline{\alpha} + \cdots + \overline{\mu}_{m-1}\overline{\alpha^{m-1}} = \overline{0}$$

et montrons que les $\overline{\mu}_i$ sont tous nuls. Pour cela, choisissons un représentant $\mu_i \in \mathcal{O}_\mathrm{K}$ de chacun des $\overline{\mu}_i$. Nous obtenons la relation

$$\mu_0 1 + \mu_1 \alpha + \cdots + \mu_{m-1}\alpha^{m-1} \in \mathfrak{P},$$

et nous devons montrer que les μ_i sont dans $\mathfrak{p}$. En projetant cette relation dans $\mathcal{O}_\mathrm{K}$, nous obtenons

$$\mu_0 \in \mathfrak{P} \cap \mathcal{O}_\mathrm{K} = \mathfrak{p}.$$

Nous en déduisons que

$$\mu_1 \alpha + \cdots + \mu_{m-1}\alpha^{m-1} \in \mathfrak{P},$$

et, en factorisant par α, que

$$\alpha(\mu_1 + \cdots + \mu_{m-1}\alpha^{m-2}) \in \mathfrak{P}.$$

Puisque $\alpha \notin \mathcal{P}$, nous en déduisons que

$$\mu_1 + \cdots + \mu_{m-1}\alpha^{m-2} \in \mathfrak{P},$$

puis, par récurrence, que tous les μ_i sont dans $\mathfrak{p}$. Nous avons donc bien établi que les $\overline{\mu_i}$ sont tous nuls, et donc que la famille des $\overline{\alpha^i}$ est libre sur $\mathcal{O}_\mathrm{K}/\mathfrak{p}$. □

Le troisième point concerne l'automorphisme $\overline{\theta}$. Il doit être générateur du groupe $\mathrm{Gal}(\mathcal{O}_\mathrm{K}/\mathfrak{p} \hookrightarrow \mathcal{O}_\mathrm{L}/\mathfrak{P})$.

Proposition 47. *Si les conjugués de α sont distincts modulo $\mathfrak{P}$, alors* $\mathrm{Gal}(\mathcal{O}_\mathrm{K}/\mathfrak{p} \hookrightarrow \mathcal{O}_\mathrm{L}/\mathfrak{P}) = \langle \overline{\theta} \rangle$.

Démonstration. Par contraposée, supposons que $\overline{\theta}$ n'est pas générateur de $\mathrm{Gal}(\mathcal{O}_\mathrm{K}/\mathfrak{p} \hookrightarrow \mathcal{O}_\mathrm{L}/\mathfrak{P})$. Alors il existe $m' \in [[0; m]]$ tel que pour tout $\overline{x} \in \mathcal{O}_\mathrm{L}/\mathfrak{P}$, $\overline{\theta}^{m'}(\overline{x}) = \overline{x}$. Cela veut dire que pour ce m' et pour tout $x \in \mathcal{O}_\mathrm{L}$, nous avons $\theta^{m'}(x) \equiv x \pmod{\mathfrak{P}}$. en particulier $\theta^{m'}(\alpha) \equiv \alpha$, et donc les conjugués de α ne sont pas tous distincts modulo $\mathfrak{P}$. □

5.2.2 Décodage d'un code de Gabidulin généralisé via une réduction

Théorème 48. *Soient* $\mathrm{Gab}_{\theta,k}(g)$ *un code de Gabidulin généralisé,* $\mathfrak{p}$ *un idéal de* $\mathcal{O}_\mathrm{K}$ *et* $\mathfrak{P}$ *l'idéal qu'il engendre dans* $\mathcal{O}_\mathrm{L}$*, de sorte que le code quotient* $\overline{\mathrm{Gab}}_{\overline{\theta},k}(\overline{g})$ *est un code de Gabidulin. Ayant reçu le mot* $y_i = \mathcal{L}_f(g_i) + e_i$*, nous pouvons résoudre le décodage (voir définition 17) pour le code* $\overline{\mathrm{Gab}}_{\overline{\theta},k}(\overline{g})$ *et le mot reçu* $(\overline{y_1}, \ldots, \overline{y_n})$*. La solution de ce problème de décodage est le polynôme d'information* f *modulo* $\mathfrak{p}$*.*

Le décodage fonctionne car le poids rang est compatible avec la réduction.

Lemme 49. *Soit* $e = (y_i - \mathcal{L}_f(g_i))_i$ *le vecteur d'erreur et soit* $\overline{e} = e \pmod{\mathfrak{p}} = (\overline{y_i} - \mathcal{L}_{\overline{f}}(\overline{g_i}))_i$*. Alors le poids de* $\overline{e}$ *est inférieur au poids de* e *:*

$$\mathrm{w_r}(\overline{e}) \leqslant \mathrm{w_r}(e).$$

Démonstration du lemme 49. En effet, soit τ le poids rang de e. Alors il existe un θ-polynôme E de degré τ qui s'annule sur les e_i. Nous avons alors $\mathcal{L}_{\overline{E}}(\overline{e_i}) = \overline{\mathcal{L}_E(e_i)} = 0$, ainsi, l'erreur $\overline{e}$ est de poids au plus τ. □

$$\begin{tikzcd}
(g,y) \arrow[r, "algorithme\ 6"] \arrow[dd, "mod"'] & (N,W) \arrow[r, "div"] \arrow[d, "mod"] & f \arrow[d, "mod"] \\
 & (\overline{N},\overline{W}) \arrow[r, "div"] & \overline{f} \\
 & & \boxed{=} \arrow[u] \arrow[d] \\
(\overline{g},\overline{y}) \arrow[r, "algorithme\ 6"] & (\widetilde{N},\widetilde{W}) \arrow[r, "div"] & \widetilde{f}
\end{tikzcd}$$

FIGURE 5.1 – Décodage modulo l'idéal $\mathfrak{p}$

Démonstration du théorème 48. Considérons un code de Gabidulin généralisé défini sur une extension $K \hookrightarrow L$ et un idéal premier $\mathfrak{p}$ de $\mathcal{O}_{\mathrm{K}}$, qui engendre un idéal $\mathfrak{P}$ dans $\mathcal{O}_{\mathrm{L}}$. Soit y le mot reçu. Supposons que le poids de l'erreur e est au plus la capacité du code $\mathrm{Gab}_{\theta,k}(g)$. (En d'autres termes, le problème de décodage admet un solution.) Notons (N, W) la solution du problème de $\mathrm{RL}(n, k, t, g, y)$ et f la solution de $\mathrm{Dec}(n, k, g, y)$. (Voir Figure 5.1 pour les notations.) Notons $(\overline{N}, \overline{W})$ et $\overline{f}$ leur réduction modulo $\mathfrak{P}$. Nous calculons alors une solution $(\widetilde{N}, \widetilde{W})$ de $\mathrm{RL}(n, k, t, \overline{g}, \overline{y}$, et notons $\widetilde{f}$ le quotient de $\widetilde{N}$ par $\widetilde{W}$. Une telle solution existe, puisque le poids de l'erreur $\overline{e}$ dans le code réduit est inférieur au poids de l'erreur e dans le code original $\mathrm{Gab}_{\theta,k}(g)$, ce poids étant lui-même inférieur à la capacité de correction. Puisque $(\widetilde{N}, \widetilde{W})$ et $(\overline{N}, \overline{W})$ sont deux solutions du même problème de Reconstruction Linéaire (sur l'extension $\mathcal{O}_{\mathrm{K}}/\mathfrak{p} \hookrightarrow \mathcal{O}_{\mathrm{L}}/\mathfrak{P}$), elles aboutissent au même quotient $\overline{f} = \widetilde{f}$, qui est la solution f modulo $\mathfrak{P}$. □

5.3 Idéaux maximaux

Nous avons vu dans la section précédente les conditions pour que la réduction d'un code de Gabidulin généralisé soit un code de Gabidulin. En particulier, nous devons trouver un idéal maximal $\mathfrak{p}$ de $\mathcal{O}_{\mathrm{K}}$ et un idéal maximal $\mathfrak{P}$ de $\mathcal{O}_{\mathrm{L}}$ de sorte que l'extension $\mathcal{O}_{\mathrm{K}}/\mathfrak{p} \hookrightarrow \mathcal{O}_{\mathrm{L}}/\mathfrak{P}$ soit de degré m.

Le but de cette section est de fournir une description des idéaux des anneaux d'entiers, pour les corps de nombres et les corps de fonctions que nous utilisons.

5.3.1 Cas des corps de nombres

Nous allons commencer par décrire l'anneau d'entiers $\mathcal{O}_{\mathrm{F}}$ d'un corps de nombres F. L'étude de la décomposition des idéaux de $\mathbb{Z}$ dans l'anneau $\mathcal{O}_{\mathrm{F}}$ va nous permettre de déterminer si un idéal maximal de $\mathbb{Z}$ fournit un idéal maximal de l'extension. Nous renvoyons les lecteurs à [Coh93, §4.8.1 et 4.8.2] pour plus de détails.

Soit F un corps de nombres, c'est-à-dire une extension algébrique de degré fini m de $\mathbb{Q}$. Un élément $x \in F$, racine d'un polynôme unitaire à coefficients dans $\mathbb{Z}$ est appelé *entier algébrique*. Ces éléments forment un anneau appelé *anneau des entiers* de F, ou encore *ordre maximal* de F, et noté $\mathcal{O}_{\mathrm{F}}$. Les anneaux d'entiers d'un corps de nombres sont des anneaux de Dedekind :

Lemme 50. *Soit F un corps de nombres et $\mathcal{O}_{\mathrm{F}}$ son anneau d'entiers. Alors*

1. *un idéal non nul est premier si et seulement si il est maximal,*
2. *tout idéal est engendré par au plus deux éléments.*

Il suffit donc de savoir quels sont les idéaux premiers de $\mathcal{O}_{\mathrm{F}}$ pour connaitre ses idéaux maximaux et ainsi pouvoir le réduire. Or, la relation entre les idéaux premiers de $\mathbb{Z}$ et ceux d'une extension est connue.

Si nous restreignons un idéal premier de $\mathcal{O}_{\mathrm{F}}$, nous obtenons un idéal premier de $\mathbb{Z}$. Inversement, soit $p\mathbb{Z}$ un idéal premier de $\mathbb{Z}$, et notons $\mathfrak{p}$ l'idéal qu'il engendre dans $\mathcal{O}_{\mathrm{F}}$. Cet idéal n'est généralement pas un idéal premier, mais il admet la décomposition suivante :

$$\mathfrak{p} = \prod_{i=1}^{g} \mathfrak{p}_i^{e_i},$$

où les $\mathfrak{p}_i$ sont les idéaux premiers de $\mathcal{O}_{\mathrm{F}}$ dont la restriction à $\mathbb{Z}$ est $p\mathbb{Z}$. Les exposants e_i sont appelés *indices de ramification*. Les quotients $\mathcal{O}_{\mathrm{F}}/\mathfrak{p}_i$ sont des extensions de $\mathbb{Z}/p\mathbb{Z}$. Leur degré, noté f_i, est appelé *degré résiduel*.

Ces quantités sont reliées entre elles par la relation

$$\sum_{i=1}^{g} f_i e_i = m.$$

Si de plus l'extension $\mathbb{Q} \hookrightarrow F$ est galoisienne, tous les indices de ramification sont égaux ($e_1 = \cdots = e_q = e$) et tous les degrés résiduels sont égaux ($f_1 = \cdots = f_q = f$). La relation devient alors

$$gfe = m.$$

L'idéal $p\mathbb{Z}$ (ou de façon équivalente le nombre premier p) reçoit des noms différents selon ces paramètres.

Définition 22. *Avec les notations précédentes, l'idéal $p\mathbb{Z}$ (ou le nombre premier p) est dit*

- ramifié *s'il existe un indice i tel que $e_i > 1$,*
- non-ramifié *si tous les e_i valent 1,*
- inerte *s'il est non ramifié et si $g = 1$,*
- totalement décomposé *s'il est inerte et si $g = m$ (auquel cas $e_i = f_i = 1$).*

Exemple 24. *Dans $\mathbb{Z}[i]$, anneau d'entiers de $\mathbb{Q}[i]$,*

- *2 est ramifié. En effet, $2 = (1+i)(1-i)$ mais $i+1$ et $i-1$ engendrent le même idéal.*
- *3 est inerte, car c'est un nombre premier dans $\mathbb{Z}[i]$. Ainsi, $\mathbb{Z}[i]/(3) \simeq \mathbb{F}_9$.*
- *5 se décompose totalement. En effet, $5 = (2+i)(2-i)$ et ces facteurs engendrent deux idéaux distincts. Ainsi, $\mathbb{Z}[i]/(5) \simeq \mathbb{F}_5 \times \mathbb{F}_5$.*

Voyons maintenant comment obtenir la décomposition de $\mathfrak{p}$ de façon effective.

Lemme 51. *Soit F un corps de nombres de la forme $F = \mathbb{Q}[z]$, où z est un entier algébrique de polynôme minimal $T(X)$. Supposons que p ne divise pas l'indice $[\mathcal{O}_{\mathrm{F}} : \mathbb{Z}[z]]$. La décomposition de $T(X)$ modulo p est de la forme*

$$T(X) = \prod_{i=1}^{g} T_i(X)^{e_i} \pmod{p}$$

et nous donne la décomposition de $\mathfrak{p}$:

$$\mathfrak{p} = \prod_{i=1}^{g} \mathfrak{p}_i^{e_i},$$

où $\mathfrak{p}_i = (p, T_i(z))$. Les degrés résiduels f_i sont alors les degrés des $T_i(X)$.

Nous utiliserons donc des idéaux et quotients de la forme suivante.

$$\begin{array}{ccccc} \mathbb{Q} & \hookrightarrow & K & \hookrightarrow & L \\ \mathbb{Z} & \hookrightarrow & \mathcal{O}_{\mathrm{K}} & \hookrightarrow & \mathcal{O}_{\mathrm{L}} \\ (p) & & \mathfrak{p} & & \mathfrak{P} \\ \mathbb{Z}/p\mathbb{Z} \simeq \mathbb{F}_p & \hookrightarrow & \mathcal{O}_{\mathrm{K}}/\mathfrak{p} \simeq \mathbb{F}_q & \hookrightarrow & \mathcal{O}_{\mathrm{L}}/\mathfrak{P} \simeq \mathbb{F}_{q^m} \end{array}$$

Remarque 18. *Le cardinal du corps $\mathcal{O}_{\mathrm{K}}/\mathfrak{p}$ est $q = p^f$ où f est le degré résiduel de l'idéal (p) dans $\mathcal{O}_{\mathrm{K}}$. Le but étant de diminuer la durée des calculs, il est plus intéressant de travailler dans des corps de petit cardinal. Pour cela, il est intéressant de considérer un idéal premier $(p) \subset \mathbb{Z}$ qui se décompose totalement dans $\mathcal{O}_{\mathrm{K}}$.*

L'algorithme 9 permet de trouver un idéal adapté à la réduction des codes de Gabidulin. La condition "est non ramifié" (ligne 9) peut être remplacée par "se décompose totalement" pour que le corps de base soit un corps premier.

Algorithm 9 Recherche d'un idéal inerte (corps de nombres)

Input: une extension de corps $K \hookrightarrow L$

Output: un idéal $\surd$ de $\mathcal{O}_{\mathrm{K}}$ inerte dans $\mathcal{O}_{\mathrm{L}}$
l'idéal $\mathcal{P}$ $\surd$ qu'il engendre dans $\mathcal{O}_{\mathrm{L}}$

0: $p \leftarrow 2$
0: décomposer l'idéal qu'il engendre dans $\mathcal{O}_{\mathrm{K}}$ en $\prod_{i=1}^{g} \surd^i$
0: **if** l'idéal est non-ramifié **then**
0: **for** $iin[[1; g]]$ **do**
0: décomposer $\surd\rangle$ dans $\mathcal{O}_{\mathrm{L}}$
0: **if** l'idéal est inerte **then**
1: **return** $\surd$ et l'idéal $\mathcal{P}$ qu'il engendre
1: **end if**
1: **end for**
1: **end if**
1: $p \leftrightarrow$ le prochain nombre premier de $\mathbb{Z}$ =0

5.3.2 Cas des corps de fonctions

Les corps de fonctions que nous utilisons sont de la forme

$$\mathbb{F}_q(X)[Y]/R(Y)$$

où R est un polynôme irréductible de degré m. L'anneau des entiers de $\mathbb{F}_q(X)[Y]$ étant $\mathbb{F}_q[X][Y]$, nous devons déterminer les idéaux de $\mathbb{F}_q[X][Y]/(R(Y))$. Pour cela, nous utiliserons des résultats classiques, présentés dans les lemmes suivants.

Lemme 52. *Soit A un anneau et I un idéal de A. Alors l'application*

$$\begin{array}{ccc} \{\textit{idéaux de } A \textit{ contenant } I\ \} & \to & \{\textit{idéaux de } A/I\ \} \\ J & \mapsto & J/I \end{array}$$

est une bijection. De plus, elle induit une bijection entre les idéaux premiers (resp. maximaux) de A contenant I et les idéaux premiers (resp. maximaux) de A/I.

Lemme 53. *Soit A un anneau principal. Alors les idéaux de $A[T]$ sont d'une des formes suivantes :*
- (0),
- (p), *où $p \in A$ est premier,*
- (P), *où $P \in A[T]$ est irréductible,*
- (p, P).

Seuls ces derniers sont maximaux.

Nous devons trouver un idéal de $\mathbb{F}_q[X][Y]$ contenant (R), c'est-à-dire de la forme $(p, P) \supset (R)$. Nous en déduisons que P divise R. Or ce dernier est irréductible, ce qui implique $P = R$.

Nous utiliserons donc des idéaux et quotients de la forme suivante.

$$\begin{array}{ccccc}
\mathbb{F}_q(X) & \hookrightarrow & \mathbb{F}_q(X)[Y] & \to & \mathbb{F}_q(X)[Y]/(R(Y)) \\
\mathbb{F}_q[X] & \hookrightarrow & \mathbb{F}_q[X][Y] & \to & \mathbb{F}_q[X][Y]/(R(Y)) \\
(p) & & (p, P) & & (p, R) \\
\mathbb{F}_q[X]/(p) & & \hookrightarrow & & (\mathbb{F}_q[X][Y]/(R(Y)))/(p, R) \\
\simeq \mathbb{F}_{q^d} & & & & \simeq (\mathbb{F}_q[X]/(p))[Y]/(R) \simeq \mathbb{F}_{(q^d)^m}
\end{array}$$

où d désigne le degré de $p \in \mathbb{F}_q[X]$ et m le degré de $R \in \mathbb{F}_q[X][Y]$.

Il est plus facile de construire des quotients dans les corps de fractions, puisqu'il suffit de choisir un polynôme irréductible de $\mathbb{F}_q[X]$. Son degré détermine alors les cardinaux des quotients.

Chapitre 6

Application au codage espace-temps

Les codes espace-temps sont des codes correcteurs conçus spécifiquement pour les canaux MIMO. Ces canaux sont intégrés à la théorie de l'information dans [T+99]. A peu près en même temps apparaissent plusieurs constructions, aussi bien de codes convolutifs [TSC98a] que de codes en bloc [Ala98]. De nombreuses autres constructions apparaitront par la suite, comme le *golden code* [BRV05], ou les codes orthogonaux [HLL08].

Si de nombreux codes existent, ce n'est pas le cas des algorithmes de décodage. En effet, ceux-ci restent proches de la recherche exhaustive, bien que l'algorithme *sphere decoding* [HV05] permette de restreindre cette recherche à une partie du code pour les codes linéaires.

Nous allons voir qu'un code espace-temps a de bonnes performances s'il a une distance minimale élevée, tant en métrique euclidienne qu'en métrique rang. La construction unifiée [LK05] se base sur des codes de Gabidulin binaires. L'étape délicate est celle qui consiste à obtenir un code complexe à partir de codes MRD binaires. Cette opération casse la structure de code de Gabidulin et est difficile à inverser. Les codes de Gabidulin généralisés permettent d'obtenir directement un code complexe.

La première section a pour but de présenter les codes espace-temps. Après avoir donné les définitions et propriétés qui nous servirons dans ce chapitre, que nous illustrerons avec l'exemple du code orthogonal, nous verrons une majoration de la probabilité d'erreur associée au décodage à maximum de vraisemblance. Nous en déduirons l'intérêt des codes en métrique rang pour le codage espace-temps.

Dans la deuxième section, nous décrirons deux constructions de codes espace-temps basées sur des codes de Gabidulin. La première construction est la *construction unifiée* de Lu et Kumar, basée sur des codes binaires. Nous définirons ensuite notre construction, basée sur les codes de Gabidulin généralisés. Nous discuterons le choix des paramètres de notre construction en présentant les constellations obtenues.

Enfin, nous présenterons les performances de nos codes dans la troisième section. Après avoir décrit les quantités dont dépend la probabilité d'erreur, nous comparerons notre construction à des codes existants.

Ces résultats ont été présentés en mai 2015 à la conférence WCC (*Workshop on Codes and Cryptography*).

6.1 Présentation du codage espace-temps

Considérons une transmission de type MIMO comme présentée dans le chapitre 1, avec n_t antennes en émission et n_r antennes en réception, s'étendant sur T instants. À l'instant $t \in [[1;T]]$, l'antenne émettrice $i \in [[1;n_t]]$ transmet le symbole $x_{i,t}$ et l'antenne réceptrice $j \in [[1;n_r]]$ reçoit le symbole $y_{j,t}$. Comme nous l'avons vu dans le paragraphe 1.4.3, ces quantités sont reliées par l'équation :

$$\begin{pmatrix} y_{1,1} & \cdots & y_{1,T} \\ \vdots & \ddots & \vdots \\ y_{n_r,1} & \cdots & y_{n_r,T} \end{pmatrix} = \begin{pmatrix} h_{1,1} & \cdots & h_{n_t,1} \\ \vdots & \ddots & \vdots \\ h_{1,n_r} & \cdots & h_{n_t,n_r} \end{pmatrix} \begin{pmatrix} x_{1,1} & \cdots & x_{1,T} \\ \vdots & \ddots & \vdots \\ x_{n_t,1} & \cdots & x_{n_t,T} \end{pmatrix} + \begin{pmatrix} e_{1,1} & \cdots & e_{1,T} \\ \vdots & \ddots & \vdots \\ e_{n_r,1} & \cdots & e_{n_r,T} \end{pmatrix},$$

ou, de façon condensée,

$$Y = HX + E,$$

où $X \in \mathcal{M}_{n_t \times T}(\mathbb{C})$ contient les symboles transmis et $Y \in \mathcal{M}_{n_r \times T}(\mathbb{C})$ les symboles reçus. La matrice $H \in \mathcal{M}_{n_r \times n_t}(\mathbb{C})$, appelée matrice du canal, est modélisée par une matrice aléatoire dont les coefficients sont indépendants et suivent une loi normale $h_{i,j} \sim \mathcal{N}(0,1)$. La matrice d'erreur $E \in \mathcal{M}_{n_r \times T}(\mathbb{C})$ est modélisée par une matrice aléatoire dont les coefficients sont indépendants et suivent une loi normale $h_{i,j} \sim \mathcal{N}(0, N_0)$.

De ces paramètres dépendent la qualité de la transmission, quantifiée par le rapport signal sur bruit (*Signal to Noise Ratio*)

$$\mathrm{SNR} = \frac{\mathbb{E}[\|HX\|_F^2]}{\mathbb{E}[\|E\|_F^2]} = \frac{n_t \mathcal{E}_s}{N_0},$$

où $\mathcal{E}_s = \mathbb{E}[|x_{i,j}|^2]$ représente l'énergie utilisée par antenne.

Un code espace-temps est une famille $\mathcal{S}$ de matrices complexes

$$\mathcal{S} \subset \mathcal{M}_{n_t \times T}(\mathbb{C}).$$

En pratique, il s'agit d'une famille finie, en bijection avec une certaine quantité d'information par une opération de codage. Il est tout de même possible de définir des codes espace-temps infinis. D'ailleurs, un code espace-temps $\mathcal{S}$ est généralement inclus dans un code espace-temps infini $\mathcal{S}_\infty$.

De même, les coefficients d'une matrice ne sont pas n'importe quels nombres complexes, mais font partie d'une famille finie $\mathcal{Q}$ appelée constellation. Il s'agit de l'ensemble des nombres complexes qui seront effectivement transmis à travers le canal.

$$\mathcal{Q} = \{X_{i,j}, X \in \mathcal{S}, i \in [1;n_t]], j \in [[1;T]]\}$$

Exemple 25. *Le code orthogonal $\mathcal{S}_\infty$ (voir[]) est l'ensemble*

$$\mathcal{S}_\infty = \left\{ \begin{pmatrix} x_1 & ix_2 & -x_3^* & -x_4^* \\ x_2 & x_1 & ix_4^* & -x_3^* \\ x_3 & ix_4 & x_1^* & x_2^* \\ x_4 & x_3 & -ix_2^* & x_1^* \end{pmatrix}, x_i \in \mathbb{Q}[i] \right\}.$$

Il doit son nom à l'orthogonalité de ses colonnes pour le produit scalaire hermitien. Nous obtenons un code fini en prenant les coefficients dans une famille finie.

$$\mathcal{S}_{4-PSK} = \left\{ \begin{pmatrix} x_1 & ix_2 & -x_3^* & -x_4^* \\ x_2 & x_1 & ix_4^* & -x_3^* \\ x_3 & ix_4 & x_1^* & x_2^* \\ x_4 & x_3 & -ix_2^* & x_1^* \end{pmatrix}, x_i \in 4-PSK = \{1, -1, i, -i\} \right\}.$$

Le codage s'écrit alors de la façon suivante. Il a été séparé en deux étapes pour plus de clarté.

$$(\mathbb{F}_2)^8 \to (4-PSK)^4 \to \mathcal{S}_{4-PSK},$$

la première étape étant constitué de 4 *fois l'application*

$$\begin{array}{rccc} \mathcal{F}: & (\mathbb{F}_2)^2 & \to & 4-PSK \\ & (a,b) & \mapsto & (-1)^a \cdot i^b. \end{array}$$

La constellation du code est alors

$$\mathcal{Q} = 4-PSK = \{1; -1; i; -i\}$$

et est identique à l'ensemble où sont choisis les x_i.

L'information contenue dans un mot du code peut être quantifiée de plusieurs façons.

Définition 23 (dimension et taux de transmission)**.**

- *La dimension k du code est le nombre de symboles dans chaque matrice. Cette définition suit la terminologie classique des codes correcteurs, en voyant une matrice comme un mot de longueur $n_t \cdot T$.*
- *Le taux de transmission R est le nombre de symboles transmis par utilisation du canal.*[1]
- *Le taux de transmission binaire R_2 est le nombre de bits transmis par utilisation du canal.*

$$R_2 = \frac{1}{T}\log_2(\mathcal{S})$$

Proposition 54. *Si tous les symboles de la constellation $\mathcal{Q}$ transportent la même quantité d'information, en d'autre termes si tous les symboles de $\mathcal{Q}$ sont présents dans les mots du code avec la même fréquence, alors*

-
$$k = \log_{|\mathcal{Q}|}(\mathcal{S})$$

-
$$R = \frac{1}{T}\log_{|\mathcal{Q}|}(\mathcal{S})$$

Exemple 26. *Le code $\mathcal{S}_{4-PSK}$ présenté dans l'exemple 25 permet la transmission de $k = 4$ symboles PSK par mot du code, soit $R = 1$ symbole par utilisation du canal (*spuc*). La constellation 4-PSK permet de coder 2 bits par symbole, d'où un taux de transmission binaire de $R_2 = 2$ bits par utilisation du canal (*bpuc*).*

Le code $\mathcal{S}_\infty$ présenté permet également la transmission de $k = 4$ symboles (éléments de $\mathbb{Q}[i]$) par mot du code, soit $R = 1$ spuc. Son taux de transmission binaire est en revanche infini.

Regardons maintenant la résistance aux erreurs des codes espace-temps. Pour cela, nous aurons besoin des quantités suivantes.

Notation 2. *Soient A et B deux matrices.*

- *$r(A,B)$ désigne le rang de $(A-B)(A-B)^*$,*
- *$\Lambda(A,B)$ désigne le produit des valeurs propres non nulles de $(A-B)(A-B)^*$.*

1. Une utilisation du canal correspond à la transmission d'un colonne d'un mot du code.

Définition 24 (diversité et gain)**.** *Soit $\mathcal{S}$ un code espace-temps.*
La diversité du code est

$$d = \min\left(r(X_i, X_j) : X_i \in \mathcal{S}, X_j \in \mathcal{S}, X_i \neq X_j\right).$$

Le gain du code est

$$\min\left(\Lambda(X_i, X_j) : X_i \in \mathcal{S}, X_j \in \mathcal{S}, X_i \neq X_j\right).$$

Remarque 19. *Un code est dit de diversité pleine si $d = min(n_t, T)$. Dans ce cas, $\Lambda(A, B)$ est le déterminant de la matrice $(A - B)(A - B)^*$.*

Connaissant le mot reçu Y et la matrice du canal H, le décodage à maximum de vraisemblance consiste à trouver le mot du code $X \in \mathcal{S}$ qui minimise la quantité $\|HX - Y\|_F$. La probabilité d'erreur est alors

$$\mathbb{P}(e) = \frac{1}{|\mathcal{S}|} \sum_{X \in \mathcal{S}} \sum_{\widehat{X} \in \mathcal{S}, \widehat{X} \neq X} \mathbb{P}(X \to \widehat{X}),$$

où $\mathbb{P}(X \to \widehat{X})$ désigne la probabilité d'estimer que $\widehat{X}$ a été transmis, alors que le mot vraiment transmis était $X \neq \widehat{X}$, et est appelée probabilité d'erreur par paire. Cette probabilité d'erreur par paire peut être majorée [2] par :

$$\mathbb{P}(X \to \widehat{X}) \leqslant \frac{1}{\left(\Lambda(X, \widehat{X}) c(SRN)^{r(X, \widehat{X})}\right)^{n_r}},$$

où la quantité $c(SRN)$ ne dépend que du SNR. Nous pouvons alors en déduire deux critères.

Théorème 55 (critères du rang et du déterminant)**.**

- *Critère du rang : plus la diversité d'un code est élevée, plus il résiste aux erreurs.*
- *Critère du déterminant : plus le gain d'un code est élevé, plus il résiste aux erreurs.*

Remarque 20. *La diversité a un effet asymptotique sur la probabilité d'erreur, tandis que le gain a un effet plus local. Ainsi, il convient de concevoir des codes de diversité d élevée, puis, parmi ces codes de diversité d, de choisir ceux ayant le gain le plus élevé.*

Comme pour les codes correcteurs que nous avons vus dans les premiers chapitres, il n'est pas possible d'avoir à la fois un taux de transfert élevé et une résistance aux erreurs élevée. Cela s'explique par la majoration suivante.

Théorème 56 (compromis taux – diversité)**.** *Soit $\mathcal{S}$ un code espace-temps de taux de transmission R et de diversité d. Alors*

$$R + d \leqslant n_t + 1.$$

Définition 25 (codes optimaux)**.** *Les codes pour lesquels la relation précédente est une égalité sont appelés codes optimaux. codes carrés requis ?*

Exemple 27. *Le code $\mathcal{S}_{4-PSK}$ défini dans l'exemple 25 est un code de taux de transmission $R = 1$ et de diversité $d = 4$. C'est donc un code optimal.*

Les codes en métrique rang permettent de construire facilement une famille de matrices de diversité élevée, puisque celle-ci n'est autre que la distance minimale. Parmi ces codes, les codes MRD, tels que les codes de Gabidulin, aboutissent à des codes optimaux.

2. Cette majoration est valable pour le canal de Rayleigh. Pour le canal de Rice, voir [TSC98a].

6.2 Codes espace-temps basés sur des codes de Gabidulin

La première construction utilisant des codes de Gabidulin est la construction unifiée [LK05] de Lu et Kumar. La première étape de leur construction est l'utilisation de codes de Gabidulin binaires, permettant la création de plusieurs mots, chacun dans un code MRD binaire. La deuxième étape est une application permettant de passer de plusieurs mots binaires à un unique mot complexe. Cette application doit répondre à des conditions contraignantes afin que la distance minimale du code complexe obtenu soit garantie.

La construction que nous proposons à l'aide de codes de Gabidulin généralisés consiste à échanger ces deux étapes. Ainsi, la mise en place du code en métrique rang intervenant en dernier, il n'y a plus besoin de précautions sur l'application permettant de passer de plusieurs nombres binaires à un nombre complexe.

6.2.1 Construction de Lu et Kumar

L'élément au cœur de la construction de Lu et Kumar est l'application suivante.

Définition 26. *Soient U et V deux entiers. Soit ζ une racine 2^V-ième de l'unité. Notons $\mathbb{Z}(\zeta)$ l'anneau des entiers du corps cyclotomique $\mathbb{Q}(\zeta)$. Soient η un élément non nul de l'idéal $2\,\mathbb{Z}(\zeta)$ et κ un élément non nul de $\mathbb{C}$. L'application μ est définie par :*

$$\begin{array}{rccc} \mu : & (\mathbb{F}_2)^{UV} & \to & \mathbb{C} \\ & (a_{1,1}, \ldots, a_{U,V}) & \mapsto & \kappa \sum_{u=1}^{U} \eta^u \zeta^{\sum_{v=1}^{V} 2^v a_{u,v}}. \end{array}$$

L'image de cette application est l'ensemble

$$\mathcal{Q} = \left\{ \kappa \sum_{u=1}^{U} \eta^u \zeta^{a_u} : a_u \in \mathbb{Z}_{2^V} . \right\}$$

Il est alors possible d'obtenir plusieurs constellations classiques par le choix de certains paramètres, ce qui explique le nom de *construction unifiée.*

Constellation	U, V	η	ζ	κ	size
PAM	$V = 1$	2	-1	1	2^U
QAM	$V = 2$	2	i	$1 + i$	2^{2U}
PSK	$U = 1$	2	RPU [3]	1	2^V

L'application μ peut être facilement adaptée à des matrices binaires, en l'appliquant position par position :

$$\begin{array}{rccc} \mathcal{F} = \mu_{m\times n} : & (\mathcal{M}_{m\times n}(\mathbb{F}_2))^{UV} & \to & \mathcal{M}_{m\times n}(\mathbb{C}) \\ & (A_{1,1}, \ldots, A_{U,V}) & \mapsto & A. \end{array}$$

Les coefficients de A sont alors définis par :

$$A_{i,j} = \kappa \sum_{u=1}^{U} \eta^u \zeta^{\sum_{v=1}^{V} 2^v (A_{u,v})_{i,j}}$$

La construction unifiée permet de coder $UVkm$ symboles d'information binaires dans une matrice complexe de taille $m \times n$ de la façon suivante.

– Les symboles d'information (sur $\mathbb{F}_2$) sont répartis dans UV matrices binaires de taille $m \times k$.

– (première étape) Nous utilisons un code de Gabidulin binaire (sous sa version matricielle)

$$\mathcal{G} : \mathcal{M}_{k,m}(F) \to \mathcal{M}_{n,m}(F)$$

afin de coder UV matrices binaires. Le codage simultané et indépendant de ces matrices est également noté $\mathcal{G}$.

$$\mathcal{G} : (\mathcal{M}_{k,m}(F))^{UV} \to (\mathcal{M}_{n,m}(F))^{UV}$$

– (deuxième étape) Ces UV matrices binaires sont "fusionnées" par $\mathcal{F}$ pour obtenir une unique matrice complexe de taille $m \times n$.

$$\mathcal{F} : (\mathcal{M}_{n,m}(\mathbb{F}_2))^u \to \mathcal{M}_{n,m}(\mathbb{C})$$

$$\boxed{\begin{array}{c} u \text{ matrices} \\ \mathcal{M}_{k,m}(\mathbb{F}_2) \end{array}} \xrightarrow{\mathcal{G}} \boxed{\begin{array}{c} u \text{ matrices} \\ \mathcal{M}_{n,m}(\mathbb{F}_2) \end{array}} \xrightarrow{\mathcal{F}} \boxed{\begin{array}{c} 1 \text{ matrix} \\ \mathcal{M}_{n,m}(\mathcal{Q}) \end{array}}$$

FIGURE 6.1 – Les étapes de la construction de Lu et Kumar

Proposition 57. *Soient U et V des entiers et $\mathcal{C}_{u,v}, 1 \leqslant u \leqslant U, 1 \leqslant v \leqslant V$ des codes correcteurs de distance rang minimale d. Alors leur image par $\mathcal{F}$ est un ensemble de matrices de distance rang minimale d.*

Théorème 58. *Le code obtenu par la construction unifiée est un code espace-temps de diversité d, de taux de transmission $\frac{km}{n}$ et de taux de transmission binaire $\frac{UVkm}{n}$.*

Démonstration. Voir [TSC98a]. □

Remarque 21. *Dans le cas particulier où $n_t = n_r = T$ et $n = m$, la diversité (resp. le taux de transmission) est la distance minimale (resp. la dimension) du code de Gabidulin sous-jacent.*

La longueur du code et la dimension de l'extension $\mathbb{F}_2 \hookrightarrow \mathbb{F}_{2^m}$ correspondent respectivement au nombre d'antennes en réception et à la durée de la transmission. Cette construction n'a de sens que s'il y a plus d'antennes en émission. Dans le cas contraire, nous obtenons une construction analogue en transposant le code. La dimension et la distance minimale du code déterminent quant à elles le taux de transmission et la diversité du code espace-temps obtenu. Ces paramètres sont donc fixés par les conditions d'utilisation. Les autres paramètres de la construction sont l'application $\mathcal{F}$ ainsi que les entiers U et V. Les conditions que doit satisfaire $\mathcal{F}$ sont contraignantes, et $\mu_{m\times n}$ est le seul exemple connu. Les entiers U et V permettent alors de choisir l'ensemble auquel appartiennent les coefficients de la matrice et influent donc sur le taux de transmission binaire. En plus de préserver la métrique rang, l'application $\mathcal{F}$ permet d'obtenir la plupart des constellations usuelles.

6.2.2 Construction à base de codes de Gabidulin généralisés

Le point délicat de la construction précédente réside dans le passage de $(\mathcal{M}_{m\times n}(\mathbb{F}_2))^{UV}$ à $\mathcal{M}_{m\times n}(\mathbb{C})$. L'application $\mathcal{F}$ est difficile à inverser, ainsi, connaissant une matrice complexe $C = \mathcal{F}(C_{1,1}, \ldots, C_{U,V})$, il est difficile de retrouver ses constituants $C_{u,v}$. L'application $\mathcal{F}$ étant appliquée position par position, cela vient de μ, elle-même difficile à inverser.

Enfin, bien que préservant la distance rang minimale, l'application $\mathcal{F}$ fait perdre une partie de la structure. Le code obtenu n'est pas un code d'évaluation, et n'est même pas linéaire. [4]

L'idée à la base de notre construction est d'échanger les deux opérations $\mathcal{G}$ et $\mathcal{F}$ de la construction précédente (voir figure 6.1). Ce changement d'ordre a pour effet de changer les contraintes s'appliquant à $\mathcal{F}$. Notre construction permet de coder Wkm symboles d'information binaires dans une matrice complexe de la façon suivante.

- Les symboles d'information (sur $\mathbb{F}_2$) sont répartis dans W matrices binaires de taille $m \times k$.
- (première étape) Ces U matrices binaires sont "fusionnées" par une application notée $\mathcal{F}$ pour obtenir une unique matrice complexe de taille $m \times k$.

$$\mathcal{F} : (\mathcal{M}_{k,m}(\mathbb{F}_2))^W \to \mathcal{M}_{k,m}(\mathbb{C})$$

- (deuxième étape) Cette matrice est encodée par un code de Gabidulin généralisé (sous sa version matricielle), pour obtenir une matrice complexe de taille $m \times n$.

$$\mathcal{G} : \mathcal{M}_{k,m}(K \subset \mathbb{C}) \to \mathcal{M}_{n,m}(K \subset \mathbb{C}).$$

$$\boxed{\begin{array}{c} u \text{ matrices} \\ \mathcal{M}_{k,m}(\mathbb{F}_2) \end{array}} \overset{\mathcal{F}}{\longrightarrow} \boxed{\begin{array}{c} 1 \text{ matrix} \\ \mathcal{M}_{k,m}(\mathcal{Q}_0) \end{array}} \overset{\mathcal{G}}{\longrightarrow} \boxed{\begin{array}{c} 1 \text{ matrix} \\ \mathcal{M}_{n,m}(\mathcal{Q}) \end{array}}$$

FIGURE 6.2 – Les étapes de notre construction

L'utilisation d'un code de Gabidulin généralisé de paramètres $[n, k, d]$ permet d'obtenir un code espace-temps disposant de bonnes propriétés. Ainsi, la diversité du code espace-temps obtenu est la distance minimale d, et le taux de transmission du code obtenu est $\frac{km}{n}$ spuc. Dans le cas où $m = n$, les codes obtenus sont optimaux, le compromis taux – diversité étant conséquence de la borne de Singleton qui s'applique aux codes de Gabidulin généralisés. Enfin, le code de Gabidulin étant utilisé lors de la dernière étape, le code obtenu ne se contente pas d'avoir une bonne distance minimale, il s'agit aussi d'un code d'évaluation. En particulier, le code est inclus dans un code infini linéaire.

Pour mettre en place un code de Gabidulin généralisé, il faut disposer d'une extension cyclique de degré fini m. Le corps des complexes ne dispose pas de telles extensions, il faut donc utiliser des corps inclus dans $\mathbb{C}$. Ainsi, nous devrons choisir une extension $\mathbb{Q} \hookrightarrow K \hookrightarrow L \subset \mathbb{C}$. Les matrices intermédiaires devront donc être à coefficients dans K.

L'application $\mathcal{F}$ n'aura pas besoin de préserver la métrique rang, il lui suffira d'être injective (bijective avec son image). L'image de $\mathcal{F}$ est constituée d'un nombre fini de matrices, aussi leurs coefficients forment un famille finie $\mathcal{Q}_0$ de nombres complexes, que nous qualifierons de constellation intermédiaire. Cette constellation doit être inclue dans le corps K, afin de permettre l'utilisation d'un code de Gabidulin généralisé.

Comme dans la première construction, il s'agira d'une application

$$\mu : (\mathbb{F}_2)^W \to \mathcal{Q}_0 \subset K$$

appliquée position par position pour obtenir

$$\mathcal{F} = \mu_{m\times k} : (\mathcal{M}_{m\times k}(\mathbb{F}_2))^W \to \mathcal{M}_{m\times k}(\mathcal{Q}_0).$$

4. Les symboles d'information étant dans $\mathbb{F}_2$ et les mots du codes à coefficients dans $\mathbb{C}$, un code espace-temps ne peut pas être linéaire. Un code espace-temps est dit linéaire s'il est inclus dans un code infini linéaire. La définir dans la partie précédente

La totalité des contraintes de $\mathcal{F}$ se transposent à μ, qui doit donc être bijective, et avoir un inverse facile à calculer. En ce sens, μ rempli le rôle d'un étiquetage.

Nous allons maintenant voir les paramètres de la construction et leur influence.

Comme dans la *construction unifiée*, la longueur du code et la dimension de l'extension $K \hookrightarrow L$ correspondent respectivement au nombre d'antennes en réception et à la durée de la transmission. La dimension et la distance minimale du code déterminent quant à elles le taux de transmission et la diversité du code espace-temps obtenu. Ces paramètres sont donc fixés par les conditions d'utilisation.

Les autres paramètres de la construction, à savoir le type d'extension de corps utilisée, la K-base de L, le support du code et la constellation intermédiaire $\mathcal{Q}_0$, qui dépend elle-même de W et μ, vont influencer la constellation du code espace-temps. L'entier W influe également sur le taux de transmission binaire.

6.2.3 Une nouvelle famille de constellations

Description Dans ce paragraphe, nous allons décrire la constellation obtenue avec notre construction. Nous tâcherons ensuite, par un choix approprié de paramètres, de dégager une constellation ayant de bonnes propriétés.

Proposition 59. *Soit $K \hookrightarrow L$ une extension galoisienne cyclique, et θ un générateur du groupe des automorphismes. Soit $\mathcal{B}$ une K-base de L et g le support d'un code de Gabidulin de paramètres $[n, k, d]$. Enfin, soit $\mathcal{Q}_0$ la constellation intermédiaire. Alors la constellation du code espace-temps $\mathcal{S}$ obtenu avec notre construction est*

$$\mathcal{Q} = \left\{ \sum_{i=0}^{k-1} \sum_{j=1}^{m} f_{i,j} b_j \theta^i(g_l) \Big|_{b_h} : f_{i,j} \in \mathcal{Q}_0, l \in [[1; n]], h \in [[1; m]] \right\}.$$

Démonstration. La constellation $\mathcal{Q}$ est l'ensemble des coefficients des matrices obtenues par l'utilisation du code de Gabidulin. Les matrices codées par ce code de Gabidulin sont celles obtenues après l'utilisation de $\mathcal{F}$. Ainsi, les colonnes des mots du code espace-temps sont de la forme

$$\sum_{i=0}^{k-1} f_i \theta^i(g_l),$$

les coefficients f_i correspondant aux colonnes des matrices obtenues à l'issue de la première étape. Ces éléments f_i sont donc de la forme

$$f_i = \sum_{j=1}^{m} f_{i,j} b_j.$$

Les évaluations sont ensuite décomposées dans la base $\mathcal{B}$ pour obtenir une matrice, dont les coefficients sont dans l'ensemble $\mathcal{Q}$ annoncé. □

mettre des eq numérotées dans l'explication des étapes et y faire référence.

Choix des paramètres Pour qu'une constellation soit intéressante dans un contexte de transmission, ses points ne doivent pas être trop proches les uns des autres. En effet, plus les points sont

proches, et plus la probabilité d'erreur augmente. Nous allons donc, par un choix de paramètres appropriés, chercher des constellations avec peu d'éléments.

Remarquons tout d'abord qu'une famille finie de $\mathbb{C}$ ne peut pas être stable par somme. C'est donc en faisant en sorte d'avoir des termes nuls dans la somme 59 que nous pourrons avoir une constellation avec peu d'éléments.

Dans la somme $\sum_{j=1}^{m} f_{i,j} b_j \theta^i(g_l)\Big|_{b_h}$, les éléments $f_{i,j} b_j \theta^i(g_l)$ sont décomposés dans la base $\mathcal{B}$ avant d'être ajoutés. Le nombre de termes dans cette somme dépend donc fortement de la décomposition des $\theta^i(g_l)$ dans la base $\mathcal{B}$, ainsi que du produit des éléments de $\mathcal{B}$ deux à deux. En notant $\theta^i(g_l) = \sum_{a=1}^{m} g_{i,l,a} b_a$, nous obtenons :

$$\begin{aligned}\sum_{j=1}^{m} f_{i,j} b_j \theta^i(g_l)\Big|_{b_h} &= \sum_{j=1}^{m} f_{i,j} b_j \sum_{a=1}^{m} g_{i,l,a} b_a\Big|_{b_h} \\ &= \sum_{j=1}^{m}\sum_{a=1}^{m} f_{i,j} g_{i,l,a} b_j b_a\Big|_{b_h}.\end{aligned}$$

Ainsi, il est préférable de choisir la base $\mathcal{B}$ stable par produit, afin que les $b_j b_a$ ne contribuent à la somme que selon une composante. De même, il est préférable d'avoir un support tel que les $\theta^i(g_l)$ aient le plus possible de composantes nulles dans la base $\mathcal{B}$, afin de n'apparaitre que dans peu de composantes.

Ces contraintes nous mènent aux choix suivants.

- Les racines de l'unité forment un ensemble stable par produit. Les extensions cyclotomiques admettent une base formée de racines de l'unité. Les extensions de Kummer de la forme $K[Y]/(Y^m - y)$, où $y \in K$ désigne une racine primitive de l'unité, admettent également une telle base. C'est donc cette base $\mathcal{B} = (1, \ldots, \alpha^{m-1})$ que nous utiliserons.
- Dans le cadre des extensions de Kummer, l'image par θ de ces éléments est également une racine de l'unité. Aussi il est possible de choisir comme éléments du support des éléments de la base. Dans les extensions cyclotomiques, l'image par θ des α^i est également une racine de l'unité. Néanmoins, dans ces extensions les racines sont reliées entre elles par $\alpha^m = -1 - \cdots - \alpha^{m-1}$. Aussi, l'un des α^i a pour image $-1 - \cdots - \alpha^{m-1}$, et est donc réparti sur toutes les coordonnées. Nous utiliserons donc des extensions de Kummer, avec pour support des éléments de la base.
- Enfin, puisque les éléments $\sum_{a=1}^{m} g_{i,l,a} b_j b_a\Big|_{b_h}$ sont des racines de l'unité, il est pertinent de choisir pour $f_{i,j}$ des racines de l'unité, afin que cette assertion reste valable. L'entier W est donc limité à $\log_2(m)$.

Nous obtenons donc le résultat suivant.

Théorème 60. *Soit $K \hookrightarrow L$ une extension de Kummer de la forme $L = K[\alpha] = K[y]/(Y^m - y)$ où $y \in K$ est une racine m-ième de l'unité et θ un générateur du groupe d'automorphismes. Choisissons $\mathcal{B} = (1, \ldots, \alpha^{m-1})$ comme K-base de L. Soit μ une application dont l'image $\mathcal{Q}_0$ est constituée de racines m-ièmes de l'unité. Enfin, soit* Gab *un code de Gabidulin généralisé de paramètres $[n, k, d]$ et de support $g = (1, \ldots, \alpha^{n-1})$. Alors le code espace-temps obtenu avec notre construction a pour constellation l'ensemble des sommes de k racines m-ièmes de l'unité.*

Propriétés des constellations obtenues Les constellations obtenues par la construction unifiée étaient l'image d'une bijection. Ce n'est pas le cas de nos constellations, qui sont le résultat

de calculs. N'étant pas directement l'image de $\mathcal{F}$, ou μ, mais la transformation par le code de Gabidulin généralisé de celle-ci, leurs propriétés sont sensiblement différentes.

- Le nombre d'éléments dans la constellation augmente avec la dimension du code de Gabidulin utilisé.
- Le module de l'élément le plus éloigné de l'origine vaut k.
- Les éléments proches de l'origine sont utilisés plus souvent.
- 0 est un élément de la constellation si et seulement si on peut écrire $k = d_1 + \cdots + d_s$, où les d_i sont des diviseurs de m autres que 1.

La troisième propriété est intéressante en termes d'énergie. En effet, l'énergie requise pour transmettre un nombre complexe est proportionnelle au carré de son module. Aussi, les éléments demandant le plus d'énergie sont les moins utilisés. Toutefois, la transmission du signal correspondant à 0 est considérée comme problématique. En effet, le signal correspondant est le signal nul, l'onde porteuse disparait alors totalement, ce qui peut aboutir à une perte de synchronisation, surtout s'il faut transmettre beaucoup de 0 à la suite. Il est donc intéressant de disposer de constellations ne contenant pas cette valeur, d'autant plus que si cette valeur 0 est présente, il s'agit d'une valeur souvent utilisée.

Enfin, remarquons que pour $k = 1$ (ce qui correspond aux codes espace-temps de diversité pleine), la constellation est l'ensemble des racines m-ième de l'unité. Ainsi, si m est une puissance de 2, on retrouve une constellation classique.

6.3 Comparaison

Les constructions basées sur les codes de Gabidulin (finis ou généralisés) permettent la conception de codes espace-temps pourvu que la durée de transmission d'un mot du code soit inférieure au nombre d'antennes en émission. (Les matrices que nous obtenons ont plus de lignes que de colonnes.) Bien que ces constructions assurent une certaine diversité au code obtenu, elles ne garantissent rien pour les autres critères de comparaison. Nous comparerons nos codes aux codes existants à travers les quantités suivantes.

- La diversité du code. Nous avons vu que la probabilité d'erreur diminue (asymptotiquement) quand cette quantité augmente. C'est le principal critère pour comparer des codes espace-temps, mais il est garanti par construction, aussi tous les codes que nous comparerons auront même diversité.
- Le gain du code. Comme pour la diversité, nous avons vu dans [TSC98b] que la probabilité d'erreur diminue quand ce paramètre augmente.
- La distance minimale entre deux points de la constellation $\mathcal{Q}$. Dans les transmissions SISO, la première étape du récepteur est de remplacer le signal reçu (un nombre complexe) par le point de la constellation le plus proche. Plus ces points sont éloignés, et plus cette étape est fiable. Bien que les symboles reçus ne soient plus des points de la constellation bruités, mais perturbés par la matrice du canal, nous conserverons ce critère de comparaison.
- La distance euclidienne minimale entre deux mots du code. Pour les codes correcteurs classiques, la distance minimale (dans la métrique adaptée) est la quantité qui permet de déterminer la capacité de correction. L'algorithme de décodage que nous avons évoqué calculant le mot du code (perturbé par la matrice de canal H) le plus proche (en métrique euclidienne) du mot reçu, plus cette quantité est élevée, et plus le code sera résistant aux erreurs.

code	$\min \mathrm{w_r}(X-Y)$	$\min \lambda(X-Y)$	$\min \lVert X-Y \rVert$	$\lVert X \rVert$	constellation
LK	3	1	$\leqslant \sqrt{6}$	4	$\{\pm 1; \pm i\}$
CGG	3	1	$\leqslant 2.875$	$\sqrt{32}$	$\{0; \pm 2; \pm 2i; \pm 1 \pm i\}$

TABLE 6.1 – Comparaison des deux constructions.

code	$\min \mathrm{w_r}(X-Y)$	$\min \lambda(X-Y)$	$\min \lVert X-Y \rVert$	$\lVert X \rVert$
LK	3	$\frac{1}{64}$	$\leqslant \frac{\sqrt{6}}{4}$	1
CGG	3	$\frac{1}{1024}$	$\leqslant 0.51$	1

TABLE 6.2 – Comparaison des deux constructions après normalisation.

– La moyenne des carrés des modules des points de la constellation, pondérés par leur fréquence d'utilisation. Cette quantité représente la quantité d'énergie requise par la transmission.

Cette dernière quantité est un peu particulière, car elle n'influence pas directement les performances du code. Néanmoins, il serait possible d'améliorer à la fois le gain et les deux distances euclidiennes minimales (pour le code et pour la constellation), simplement en multipliant tous les mots du code par une même constante. (Cela n'affecterait pas la diversité du code, et donc le compromis taux – diversité non plus.) La contrepartie est que cela entrainerait une consommation d'énergie supérieure. Aussi, pour pouvoir comparer des codes, nous devrons les normaliser afin de comparer les codes à consommation d'énergie égale.

Première comparaison Dans cet exemple, nous voulons concevoir un code espace-temps constitué de matrices de taille 4×4. Notre objectif est de transmettre 16 bits par matrices, soit un taux de transmission binaire de 2 bpuc. Nous allons utiliser pour cela des codes de Gabidulin (finis ou généralisés) de paramètres $[n, k, d]$. Les paramètres obtenus avec la construction de Lu-Kumar (LK) et avec les codes de Gabidulin généralisés (CGG) sont présentés dans le tableau suivant. Dans les deux cas, le code contient 65536 mots, est de diversité 3 et de taux de transmission 2. Les deux codes sont donc optimaux.

Les paramètres obtenus pour les différentes constructions sont présentées dans la tableau 6.3. (Certaines quantités ne sont pas les valeurs minimales, un parcours exhaustif de toutes les paires de mots du code comprenant plus de deux milliards de cas. La valeur donnée n'est donc qu'un majorant du minimum.) Le tableau 6.3 présente la comparaison des mêmes codes, après normalisation selon la norme des mots du code, qui représente l'énergie utilisée.

Nous pouvons constater sur cet exemple que le code issu de la construction de Lu et Kumar a un gain plus élevé. La probabilité d'erreur par paire est donc plus faible avec cette construction. En revanche, le code issu de notre construction est inclus dans un code linéaire de diversité 3. Ce n'est pas le cas de la construction de Lu et Kumar, puisque leur code contient un mot de rang 1. En effet, si ce code était inclus dans un code linéaire, ce dernier serait de diversité au plus 1.

Deuxième comparaison Concentrons-nous maintenant sur des codes de diversité pleine. Nous allons concevoir un code espace-temps constitué de matrices de taille 4×4. Pour que le code soit de diversité 4, il doit être de taux de transmission 1. Notre objectif est donc de transmettre 8 bits par matrices, soit un taux de transmission binaire de 1 bpuc. Nous allons utiliser pour cela des codes de Gabidulin (finis ou généralisés) de paramètres $[4, 1, 4]$. Les codes obtenus ne comportant que 256 mots, nous allons pouvoir varier les constructions. Ainsi, nous allons étudier les codes obtenus

code	$\min \mathrm{w_r}(X-Y)$	$\min \lambda(X-Y)$	$\min \|X-Y\|$	$\|X\|$
LK (PAM)	4	16	4	8.82
LK (QAM)	4	16	4	$\sqrt{32}$
LK (PSK)	4	4	$\sqrt{8}$	4
CGG	4	4	$\sqrt{8}$	4
ortho	4	4	$\sqrt{8}$	4

TABLE 6.3 – Comparaison de codes optimaux de diversité pleine.

code	$\min \mathrm{w_r}(X-Y)$	$\min \lambda(X-Y)$	$\min \|X-Y\|$	$\|X\|$
LK (PAM)	4	2.610^{-3}	0.45	1
LK (QAM)	4	$\frac{1}{64}$	$\frac{1}{\sqrt{2}}$	1
LK (PSK)	4	$\frac{1}{64}$	$\frac{1}{\sqrt{2}}$	1
CGG	4	$\frac{1}{64}$	$\frac{1}{\sqrt{2}}$	1
Ortho	4	$\frac{1}{64}$	$\frac{1}{\sqrt{2}}$	1

TABLE 6.4 – Comparaison de codes optimaux de diversité pleine après normalisation.

par la construction de Lu et Kumar pour les constellations 4-PSK, 4-PAM et 4-QAM. Nous les comparerons au code obtenu à partir de notre construction ainsi qu'au code orthogonal. Dans tous les cas, le code contient 256 mots, est de diversité 4 et de taux de transmission 1. Les codes sont donc optimaux.

Nous pouvons remarquer à partir de cet exemple que les valeurs minimales dépendent en grande partie de la constellation. En effet, les trois dernières constructions sont basées sur la constellation 4-PSK, et la seconde sur la constellation 4-QAM, qui n'est autre que la 4-PSK à un facteur multiplicatif près. Cela s'explique par les faits suivants. Tous les coefficients sont de norme 1, ce qui explique que tous les mots soient de norme 4. Il en va de même pour les différences de normes, les coefficients de la différence entre deux mots étant de norme 0 ou 2. Les différentes valeurs possibles pour $\|X-Y\|$ correspondent alors au nombre de 0 dans ces différences. Enfin, le gain est le déterminant d'une matrice à coefficients dans $\{\pm 1; \pm i\} \subset \mathbb{Q}[i]$. Puisqu'il s'agit d'un anneau, le gain est un élément de $\mathbb{Q}[i]$. Nous obtenons 4, qui est la plus petite valeur possible, les coefficients des différences étant tous multiples de 2.

Enfin, bien que ces paramètres soient identiques pour les trois derniers codes (même avant normalisation), les trois codes sont distincts deux à deux. Une étude plus poussée de ces codes révèle que le gain minimal est atteint 5236 fois pour la construction LK, 3136 fois pour la construction CGG et 1536 fois pour la construction Ortho. Inversement, la distance euclidienne minimale entre deux mots du code est atteinte 1024 fois pour les constructions CGG et Ortho, contre seulement 256 fois pour la construction LK. Les trois codes sont donc trois codes distincts. Les critères de comparaison évoqués ne permettent pas une comparaison précise de ces trois constructions. Une étude détaillée de la probabilité d'erreur permettrait de les départager. La non-linéarité jouant en défaveur de la construction LK, et la construction Ortho n'étant possible que pour quelques valeurs de n et m, notre construction CGG permettrait de construire des codes aussi intéressants que les codes existant dans le cas particulier des codes optimaux de diversité pleine.

Chapitre 7

Conclusion / Perspectives

θ-polynômes et définitions des codes Les codes de Gabidulin généralisés consistent en l'évaluation, sur un support bien choisi, d'un θ-polynôme. Ces θ-polynômes sont eux-mêmes la généralisation des q-polynômes utilisés dans les codes de Gabidulin finis, et sont toujours un cas particulier des polynômes tordus. Aussi, nous pourrions définir plusieurs variantes des codes de Gabidulin généralisés :

- en ajoutant une dérivation dans la définition du produit (sans effet dans les corps finis),
- en considérant l'évaluation par les restes plutôt que par opérateurs,
- en ajoutant une dérivation et en considérant l'évaluation par les restes.

Nous avons vu qu'avec les θ-polynômes tels que définis dans le chapitre 2, il était important de pouvoir borner la dimension de l'espace des racines d'un polynôme par son degré. Dans ce cas, il était possible de définir la métrique rang à partir des θ-polynômes.

Qu'en est-il dans les trois cas que nous venons d'évoquer ? Quelle est la structure de l'espace des racines des polynômes ainsi définis ? Quelle est la "dimension" de celui-ci ? Est-il possible d'utiliser cet espace de racines pour définir une métrique, et si oui, quelle est cette métrique ?

Codes de Gabidulin généralisés Revenons maintenant aux codes tels qu'ils ont étés définis dans ce document. Plusieurs questions peuvent être soulevées à leur sujet. La première question est propre à la généralisation des codes, mais les suivantes sont également valables pour les codes de Gabidulin finis.

Nous avons vu que les codes de Gabidulin généralisés héritaient des propriétés des codes originaux. Ainsi, tous les algorithmes de décodage devraient pouvoir s'adapter. Ainsi, la méthode de gestion des effacements que nous avons vus devrait pouvoir fonctionner avec tout algorithme qui ne requiert aucune hypothèse sur le support du code. Néanmoins, la rigueur impose une vérification.

Restons dans le décodage avec une remarque à propos de l'algorithme de Reconstruction. Si nous calculons la famille $(u_{1,1}^{(i)}, \ldots, u_{1,n}^{(i)})$ à chaque étape i de l'algorithme, nous pouvons constater que son rang décroit au cours de l'algorithme. Ce rang serait-il lié au rang de l'erreur, et si oui, cela pourrait-il permettre de connaitre le poids de l'erreur à l'avance ?

Enfin, terminons ce paragraphe par une application possible. Que se passerait-il si le support $g = (g_1, \ldots, g_n)$ du code n'était pas libre ? Il est clair que ce ne serait pas un code de longueur n. Néanmoins, les évaluations liées ne sont pas complètement inutiles, puisqu'elles apportent une autre forme de redondance. Un code de support $(\alpha_1 + \alpha_2 + \alpha_3, \alpha_1, \alpha_2, \alpha_3, \alpha_7, \alpha_8, \alpha_4, \alpha_5, \alpha_6, \alpha_4 + \alpha_5 + \alpha_6)$, où la famille $(\alpha_1, \ldots, \alpha_8)$ est libre, permettrait deux types de décodage :

- décodage global : il s'agit de corriger des erreurs habituelles, en ignorant les coefficients aux extrémités du mot,
- décodage local : il s'agit de corriger une erreur ne portant que sur les premiers coefficients, sans avoir besoin d'utiliser les autres coefficients.

De tels codes pourraient être utilisés dans le stockage distribué.

Codes espace-temps Enfin, les dernières perspectives concernent l'application au codage espace-temps. L'idée était à l'origine d'utiliser les codes de Gabidulin généralisés pour disposer d'un algorithme de décodage. Il se trouve que, dans les transmissions MIMO, l'erreur est une matrice dont tous les coefficients sont aléatoires et indépendants. Elle est donc de rang plein (avec une probabilité proche de 1). Il n'est donc pas possible de décoder ce genre d'erreur avec un code en métrique rang, qui ne peut pas corriger plus de $\approx n/2$ erreurs dans le cas d'un code de dimension très faible $k = 1$.

Les codes que nous avons conçu ne contiennent que des matrices à coefficients entiers. Il est donc possible, après avoir inversé la matrice de canal H, d'approcher les valeurs complexes reçues par l'entier le plus proche. Ainsi, les coefficients ayant subi une erreur faible (module inférieur à 0,5) sont débarrassés de leur erreur, et seules les erreurs importantes sont conservées. Le code de Gabidulin n'aurait donc que ces erreurs à corriger. Une simulation montre que cette méthode n'est valable que pour des transmissions relativement peu bruitées.

Nous disposons donc d'une structure supplémentaire par rapport aux codes existants, mais nous ne savons pas encore comment l'utiliser. Une piste serait d'améliorer l'initialisation du *sphere decoding*. En effet, celui-ci commence par la décomposition LU d'une grande matrice. [1] Cette matrice est le produit de la matrice génératrice du code (sous forme matricielle) et de la matrice du canal. Ce produit ressemble à la matrice d'un code de Gabidulin. L'expression de la factorisation LU d'une telle matrice en fonction des coefficients de H et du support du code de Gabidulin pourrait éviter le calcul de la factorisation LU à chaque changement de matrice de canal.

Enfin, il est possible de décrire un modèle dans lequel les codes de Gabidulin généralisés seraient efficaces. Il s'agirait d'un système de type MIMO, avec peu de bruit additif en réception, mais confronté à un bruit fort provenant d'une seule antenne lointaine, sur la même fréquence. Ce bruit serait alors modélisé par une erreur de poids faible, que nous pourrions corriger avec un algorithme de décodage de codes de Gabidulin.

1. de taille $nk \times mn$.

Bibliographie

[AB] Carina Alves and Jean-Claude Belfiore. Space-time codes based on quaternion algebras of small volume.

[Ala98] Siavash Alamouti. A simple transmit diversity technique for wireless communications. *Selected Areas in Communications, IEEE Journal on*, 16(8) :1451–1458, 1998.

[Aug] Daniel Augot. Generalization of gabidulin codes over fields of rational functions. *reference incomplete*.

[BGU07] Delphine Boucher, Willi Geiselmann, and Félix Ulmer. Skew-cyclic codes. *Applicable Algebra in Engineering, Communication and Computing*, 18(4) :379–389, 2007.

[BRC60] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1) :68–79, 1960.

[BRV05] Jean-Claude Belfiore, Ghaya Rekaya, and Emanuele Viterbo. The golden code : A 2 x 2 full-rate space-time code with nonvanishing determinants. *IEEE Transactions on information theory*, 51(4) :1432–1436, 2005.

[BU14a] Delphine Boucher and Felix Ulmer. Linear codes using skew polynomials with automorphisms and derivations. *Designs, codes and cryptography*, 70(3) :405–431, 2014.

[BU14b] Delphine Boucher and Felix Ulmer. Self-dual skew codes and factorization of skew polynomials. *Journal of Symbolic Computation*, 60 :47–61, 2014.

[BW86] Elwyn R Berlekamp and Lloyd R Welch. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[CB12] Xavier Caruso and Jérémy Le Borgne. Some algorithms for skew polynomials over finite fields. *arXiv preprint arXiv :1212.3582*, 2012.

[Cha08] Lionel Chaussade. Codes correcteurs sur des anneaux de öre multivariés. 2008.

[Cha10] Lionel Chaussade. *Codes correcteurs avec les polynômes tordus*. PhD thesis, Université Rennes 1, 2010.

[CLU09] Lionel Chaussade, Pierre Loidreau, and Felix Ulmer. Skew codes of prescribed distance or rank. *Designs, Codes and Cryptography*, 50(3) :267–284, 2009.

[Coh93] Henri Cohen. *A course in computational algebraic number theory*, volume 138. Springer, 1993.

[Cox11] David A Cox. *Galois theory*, volume 61. John Wiley & Sons, 2011.

[Del78] Ph Delsarte. Bilinear forms over a finite field, with applications to coding theory. *Journal of Combinatorial Theory, Series A*, 25(3) :226–241, 1978.

[Gab85] Ernest Mukhamedovich Gabidulin. Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1) :3–16, 1985.

[Gao03] Shuhong Gao. A new algorithm for decoding reed-solomon codes. In *Communications, Information and Network Security*, pages 55–68. Springer, 2003.

[Goz97] Yvan Gozard. *Théorie de galois*. 1997.

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information processing letters*, 43(4) :169–174, 1992.

[Ham50] Richard W Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2) :147–160, 1950.

[HL08] Tracey Ho and Desmond Sui Lun. *Network coding : an introduction*, volume 6. Cambridge University Press Cambridge, 2008.

[HLL08] Camilla Hollanti, Jyrki Lahtonen, and Hsiao-feng Lu. Maximal orders in the design of dense space-time lattice codes. *Information Theory, IEEE Transactions on*, 54(10) :4493–4510, 2008.

[Hoc59] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres (paris)*, 2(147-156) :116, 1959.

[HV05] Babak Hassibi and Haris Vikalo. On the sphere-decoding algorithm i. expected complexity. *Signal Processing, IEEE Transactions on*, 53(8) :2806–2818, 2005.

[JP15] Relinde Jurrius and Ruud Pellikaan. On defining generalized rank weights. *arXiv preprint arXiv :1506.02865*, 2015.

[KK08] Ralf Koetter and Frank R Kschischang. Coding for errors and erasures in random network coding. *Information Theory, IEEE Transactions on*, 54(8) :3579–3591, 2008.

[KM03] Ralf Koetter and Muriel Médard. An algebraic approach to network coding. *Networking, IEEE/ACM Transactions on*, 11(5) :782–795, 2003.

[KT14] Margreta Kuijper and A-L Trautmann. List-decoding gabidulin codes via interpolation and the euclidean algorithm. In *Information Theory and its Applications (ISITA), 2014 International Symposium on*, pages 343–347. IEEE, 2014.

[LK05] H-F Lu and PY Kumar. A unified construction of space-time codes with optimal rate-diversity tradeoff. *Information Theory, IEEE Transactions on*, 51(5) :1709–1730, 2005.

[Loi06] Pierre Loidreau. A welch–berlekamp like algorithm for decoding gabidulin codes. In *Coding and Cryptography*, pages 36–45. Springer, 2006.

[LSS14] Wenhui Li, Vladimir Sidorenko, and Danilo Silva. On transform-domain error and erasure correction by gabidulin codes. *Designs, Codes and Cryptography*, 73(2) :571–586, 2014.

[Neu99] Jürgen Neukirch. *Algebraic number theory*. Springer, 1999.

[OBV07] Frdrique Oggier, Jean-Claude Belfiore, and Emanuele Viterbo. *Cyclic division algebras : A tool for space-time coding*. Now Publishers Inc, 2007.

[OD11] Frederique Oggier and Anwitaman Datta. Self-repairing codes for distributed storage—a projective geometric construction. In *Information Theory Workshop (ITW), 2011 IEEE*, pages 30–34. IEEE, 2011.

[ORBV06] Frédérique Oggier, Ghaya Rekaya, Jean-Claude Belfiore, and Emanuele Viterbo. Perfect space–time block codes. *IEEE Transactions on Information Theory*, 52(9) :3885–3902, 2006.

[Ore33a] Oystein Ore. On a special class of polynomials. *Transactions of the American Mathematical Society*, 35(3) :559–584, 1933.

[Ore33b] Oystein Ore. Theory of non-commutative polynomials. *Annals of mathematics*, pages 480–508, 1933.

[OS12] Frederique Oggier and Adnen Sboui. On the existence of generalized rank weights. 2012.

[PiR89] Alain Poli and LLorenç Huguet i Rotger. *Codes correcteurs : théorie et applications.* Masson, 1989.

[Rot91] Ron M. Roth. Maximum-rank array codes and their application to crisscross error correction. *Information Theory, IEEE Transactions on*, 37(2) :328–336, 1991.

[RS60] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2) :300–304, 1960.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 :379–423 and 623–656, 1948.

[Sin64] Richard C Singleton. Maximum distance-nary codes. *Information Theory, IEEE Transactions on*, 10(2) :116–118, 1964.

[SKK08] Danilo Silva, Frank R Kschischang, and Ralf Koetter. A rank-metric approach to error control in random network coding. *Information Theory, IEEE Transactions on*, 54(9) :3951–3967, 2008.

[SRS03] BA Sethuraman, B Sundar Rajan, and Vummintala Shashidhar. Full-diversity, high-rate space-time block codes from division algebras. *Information Theory, IEEE Transactions on*, 49(10) :2596–2616, 2003.

[Ste15] Ian Nicholas Stewart. *Galois theory.* CRC Press, 2015.

[Sti09] Henning Stichtenoth. *Algebraic function fields and codes*, volume 254. Springer Science & Business Media, 2009.

[T+99] I Emre Telatar et al. Capacity of multi-antenna gaussian channels. *European transactions on telecommunications*, 10(6) :585–595, 1999.

[TSC98a] Vahid Tarokh, Nambi Seshadri, and A Robert Calderbank. Space-time codes for high data rate wireless communication : Performance criterion and code construction. *Information Theory, IEEE Transactions on*, 44(2) :744–765, 1998.

[TSC98b] Vahid Tarokh, Nambi Seshadri, and A Robert Calderbank. Space-time codes for high data rate wireless communication : Performance criterion and code construction. *Information Theory, IEEE Transactions on*, 44(2) :744–765, 1998.

[WAS+11] Antonia Wachter, Valentin Afanassiev, Vladimir Sidorenko, et al. Fast decoding of gabidulin codes. In *WCC 2011-Workshop on coding and cryptography*, pages 433–442, 2011.

[WB86] Lloyd R Welch and Elwyn R Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[WZ13a] Antonia Wachter-Zeh. Bounds on list decoding of rank-metric codes. *IEEE Transactions on Information Theory*, 59(11) :7268–7277, 2013.

[WZ13b] Antonia Wachter-Zeh. *Decoding of Block and Convolutional Codes in Rank Metric.* PhD thesis, Technical University of Denmark, 2013.

Résumé

Les codes espace-temps sont des codes correcteurs dédiés aux transmissions MIMO. Mathématiquement, un code espace-temps est un ensemble fini de matrices complexes. Ses performances dépendent de plusieurs critères, dont la distance minimale en métrique rang. Les codes de Gabidulin sont des codes dans cette métrique, connus pour leur optimalité et pour l'existence d'algorithmes de décodage efficaces. C'est pourquoi ils sont utilisés pour concevoir des codes espace-temps. La principale difficulté est alors de construire des matrices complexes à partir de matrices binaires.

Les travaux présentés dans ce documents consistent à généraliser les codes de Gabidulin à des corps de nombres, en particulier des extensions cyclique. Nous verrons qu'ils ont les mêmes propriétés que leurs analogues sur les corps finis. Nous étudierons plusieurs modèles d'erreurs et d'effacements et présenterons un algorithme qui permettra de retrouver l'information transmise avec une complexité quadratique.

En calculant dans des corps infinis, nous serons confrontés au problème de la taille des éléments, qui augmente exponentiellement au gré des calculs. Pour éviter ce désagrément, nous verrons qu'il est possible de réduire le code afin de calculer dans un corps fini. Enfin, nous proposerons une famille de codes espace-temps dont la construction est basée sur les codes de Gabidulin généralisés. Nous verrons que leurs performances sont similaires à celles des codes existants, et qu'ils disposent d'une structure supplémentaire.

Mots clefs

Codes correcteurs, codes de Gabidulin, métrique rang, polynômes tordus, codes espace-temps, transmission MIMO.

Abstract

Space-time codes are error correcting codes dedicated to MIMO transmissions. Mathematically, a space-time code is a finite family of complex matrices. Its preformances rely on several parameters, including its minimal rank distance. Gabidulin codes are codes in this metric,, famous for their optimality and thanks to efficient decoding algorithms. That's why they are used to design space-time codes. The main difficulty is to design complex matrices from binary matrices.

The aim of the works collected here is to generalize Gabidulin codes to number fields, especially cyclique extesnions. We see that they have the same properties than Gabidulin codes over finite fields. We study several errors and erasures models and introduce a quadratic algorithm to recover transmitted information.

When computing in finite fields, we are faced with the growing size problem. Indeed, the size of the coefficients grows exponentielly along the algorithm. To avoid this problem, it is possible to reduce the code, in order to compute in a finite field. Finally, we design a family of space-time codes, based on generalised Gabidulin codes. We see that our codes have performances similar to those of existing codes, and that they have additional structure.

Mots clefs

Coding theory, Gabidulin codes, rank metric, skew polynomials, space-time codes, MIMO transmission