UNIVERSITÉ DE LIMOGES

ECOLE DOCTORALE Science-Technologie-Santé

Faculté des Sciences

Année : 2005 **Thèse No: 8**

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Mathématiques et Applications

presentée et soutenue par

Carmen-Simona NEDELOAIA

le 28 février 2005

Étude des énumérateurs des poids des codes linéaires utilisant des formes décomposées des matrices génératrices

Thèse dirigée par : Pascale CHARPIN,
Directeur de Recherche au projet Codes, INRIA Rocquencourt

JURY :

Christine BACHOC, Professeur à l'Université Bordeaux I, rapporteur
Antoine LOBSTEIN, Chargé de Recherche au CNRS (ENST Paris), rapporteur
Thierry BERGER, Professeur à l'Université de Limoges, examinateur
Jean-Pierre BOREL, Professeur à l'Université de Limoges, examinateur
Anne CANTEAUT, Chargée de Recherche au projet Codes, INRIA, examinateur
Philippe GABORIT, Maître de Conférences à l'Université de Limoges, examinateur

---

Laboratoire d'Arithmétique, Calcul formel et Optimisation — UMR CNRS 6090,
123 av. Albert Thomas, 87060 Limoges
Projet Codes, INRIA Rocquencourt, Domaine de Voluceau, 78153 Le Chesnay

Mis en page avec la classe thloria.

Remerciements

Je tiens à remercier tout d'abord ma directrice de thèse, Pascale Charpin, pour tout ce qu'elle m'a appris sur le plan professionnel et humain ; pour la confiance qu'elle m'a témoignée en m'orientant vers un sujet si passionnant ; pour la "liberté" qu'elle m'a accordée dans mon activité de recherche, pour les longues heures de travail ensemble et de correction de mes manuscrits, pour sa gentillesse, pour son ouverture d'esprit.

Je remercie de tout mon coeur Anne Canteaut pour l'intêret qu'elle a porté à mon travail et pour son amitié. Je la remercie pour son aide constante en programmation, sans laquelle les pages de "Résultats numériques" seraient bien diminuées.

Je voudrais exprimer toute ma gratitude à Philippe Gaborit qui a encadré mon travail à Limoges. Je lui remercie pour ses cours en DEA ; je lui dois notamment le fait de savoir programmer en Magma et l'orientation de ma recherche vers les codes auto-duaux, qui lui sont tellement familiers. Sans lui, le sixième chapitre de ma thèse ne serait pas écrit. Je remercie aussi notre coauteur, Alfred Wassermann, dont les programmes fournissent toujours des résultats remarquables ; il a su répondre à toutes nos questions avec précision et gentillesse.

J'ai trouvé en Antoine Lobstein un lecteur très attentif de mon manuscrit. Ses nombreuses corrections, remarques et questions ont beaucoup amélioré la qualité de ce document. Je le remercie, ainsi que Christine Bachoc, pour l'honneur qu'il m'ont fait en acceptant d'être rapporteurs de ma thèse.

Je remercie également Thierry Berger pour ses cours et ses conseils, qui ont fait que ma recherche s'est orienté vers les codes correcteurs d'erreurs. Je le remercie aussi pour son aide administrative qui a fait que mon travail à Limoges a pu se dérouler dans de bonnes conditions.

J'adresse mes sincères remerciements à Jean-Pierre Borel pour l'honneur qu'il m'a fait en acceptant d'être président du jury et pour l'intérêt qu'il a porté à mon travail.

À travers Nicolas Sendrier, je remercie tous les membres du projet CODES et tous les chercheurs que j'ai côtoyés durant ces quatre dernières années, pour leurs conseils et pour l'accueil toujours chaleureux qu'ils m'ont réservé.

C'est grâce à l'atmosphère propice au travail de recherche, qui règne au Département de Mathématiques, et à mes collegues de Maîtrise et DEA de Limoges, que j'ai décidé de poursuivre mes études doctorales en France. Merci à Matthieu Finiasz, Ayoub Otmani, Laurent Dubreuil, Cédric Lauradoux, Fabien Galand, Fabien Laguillaumie pour leurs conseils informatiques, à Philippe Ségalat, Samuel Maffre et Nicolas Le Roux qui furent mes collègues de bureau à Limoges. Je remercie tous mes collègues thésards de l'INRIA et de Limoges pour leurs conseils et leur amitié.

Un grand merci à Christelle Guiziou-Cloître, Yolande Vieceli, Sylvie Laval, Patricia Vareille, Martine Guerletin, Nadine Tchefranoff et Chantal Subileau pour m'avoir aidée dans les démarches administratives avec toute leur compétence.

Durant trois années j'ai bénéficié d'une Bourse Régionale ; je remercie le Conseil Régional de la Région Limousin pour avoir soutenu mon projet de recherche. Je remercie aussi les membres du Département de Mathématiques de l'Université Paris VIII, au sein duquel j'ai travaillé la dernière année de ma thèse.

Enfin, une pensée de reconnaissance va vers mes parents et vers mes amis, de Paris ou d'ailleurs, pour tout ce qu'ils m'ont apporté.

Table des matières

Introduction générale

Dans cette thèse nous étudions les paramètres de certaines classes de codes, la plupart d'entre elles étant des classes de codes invariants sous le groupe affine, dits *codes affines-invariants*. Nous utilisons des propriétés diverses, liées surtout à la forme de la matrice génératrice, pour le calcul des distances minimales et des polynômes énumérateurs. Nous voulons obtenir des résultats originaux concernant des codes *longs*. Nous essayons donc de simplifier les matrices génératrices par décomposition et d'établir une relation entre les matrices composantes (ou les codes composants) et le code de départ. Toute approche théorique, dans cette thèse, est finalisée par des résultats numériques ; ainsi, minimiser la complexité de tout calcul devient prioritaire.

Un mot de code est un bloc de symboles ordonnés. Dans la majorité des cas nous devrons modifier l'ordre initial des coordonnées qui est un élément de définition du code. En effet, l'ordre initial ne nous permet pas de décomposer efficacement la matrice génératrice. Il s'agit surtout de l'ordre *cyclique* des bits et, comme nous allons le voir, presque tous nos codes sont cycliques (ou cycliques étendus). Ainsi, les codes de Reed-Muller (RM) illustrent bien cette situation. Ils peuvent être construits sous forme cyclique, mais aussi via l'*ordre standard des bits*. Cette dernière forme permet la construction $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ itérative de ces codes (voir Section 2.2), que nous avons particulièrement étudiée. Nous rappelons la définition des codes RM seulement suivant l'ordre standard (voir Chapitre 1).

Une autre classe de codes très importante est la classe des codes BCH étendus et de leurs duaux. En exploitant le fait qu'ils ont pour sous-codes certains codes RM, et donc qu'ils héritent de leurs propriétés, nous choisissons de passer ces codes en ordre standard. Plus généralement, ce passage est possible pour tout code affine-invariant et permet une construction itérative de ces codes (voir Section 4.2). Les codes BCH, leurs duaux, et les codes RM sont affines-invariants.

Nous étudions aussi certains codes auto-duaux, dont les plus connus sont les codes résidus quadratiques. Pour eux, on ne connaît pas de décompositions efficaces, qui puissent aboutir au calcul de nouveaux polynômes énumérateurs, par exemple. Nous avons résolu ce problème en utilisant un algorithme de calcul du nombre des mots de petit poids (Section 6.4) et la théorie des invariants (Section 6.3).

Le document contient une partie introductive, suivie de deux parties où sont exposés nos résultats.

Introduction.

Le premier chapitre résume les notations et les notions de base : code linéaire et ses para-

mètres, dualité, équivalence, groupe du code, code cyclique, code affine-invariant ... Nous donnons les définitions et les propriétés de base des classes de codes auxquelles nous nous sommes intéressés.

Dans le deuxième chapitre, nous présentons les constructions par décomposition. Un code est ainsi exprimé à l'aide d'autres codes, en général de longueur beaucoup plus petite. Nous insistons sur la *construction carrée*, en montrant comment l'utiliser de façon itérative pour obtenir les codes RM. On retrouvera cette construction, toujours itérative, dans le cas des codes cycliques auto-duaux (Chapitre 5).

Une modification de cette construction nous amène par la suite à introduire la classe des codes LTSC (*construits en forme carrée modifiée*), qui contient les codes symétriques et réversibles (à équivalence près). On montrera plus loin qu'en fait tous les codes affines-invariants peuvent être mis sous forme LTSC. Cette construction s'adressant aux codes de longueur paire, il sera préférable de travailler avec le code étendu et de retrouver par la suite les paramètres du code de départ, par des résultats bien connus. Enfin, les deux dernières sections sont consacrées à des constructions de codes de longueurs composées : des codes auto-duaux et BCH. Comme nous l'avons déjà remarqué pour les codes de longueur paire, tout code de longueur composée n est candidat à une décomposition suivant les diviseurs de n ; ici nous rappelons les constructions *bloc-circulante* et *doublement circulante* (qui ne sont pas exploitées dans cette thèse, faute de cyclicité), ou les constructions qui admettent pour sous-code une somme directe (qui sont généralisées en Section 4.2).

Le troisième chapitre est un peu à part. Il traite des treillis, c'est-à-dire des représentations graphiques des codes, qui ont été au centre de nos préoccupations au début de cette thèse, notamment pour le calcul du polynôme énumérateur du code RM$(4, 9)$. En effet, suite au calcul de l'énumérateur de poids du code RM$(3, 9)$ [97], le code RM (auto-dual) $(4, 9)$ est le seul dont on ne connaît pas la distribution parmi tous les codes RM avec $n \leq 512$ [103]. La démarche de Kasami *et al.* en [26] nous a orienté vers l'étude des treillis [75].

Toute la présentation générale dans la Section 3.1 sert par la suite à l'étude du treillis minimal. Pour un ordre des coordonnées fixé, il s'avère que le treillis minimal correspond à une *matrice orientée dans le sens du treillis* (Section 3.2.1). En étudiant de plus près le cas des codes cycliques, on prouve qu'ils ont la plus mauvaise complexité des états, donc il faut trouver une permutation optimale des bits pour construire un treillis minimal (voir Section 3.3). On retrouve ainsi nos précédents commentaires sur l'ordre optimal des bits.

Rappelons que nous travaillons avec des codes de grande longueur, par conséquent on a intérêt à construire un graphe sectionné, pour réduire la densité des arêtes. Une section entière est consacrée au rappel de la construction des treillis sectionnés. Pour simplifier encore ce treillis, on calcule le nombre des sous-treillis parallèles et isomorphes (Section 3.4.4). Tous ces résultats sont illustrés par les codes RM avec leur construction carrée itérative ; on prolonge ainsi la Section 2.2. On peut en plus calculer le sous-treillis minimal [64], mais cela ne concerne que les codes dont on connaît les mots de poids minimal (par exemple les codes RM). En conclusion, ce chapitre nous permet d'avoir une vision graphique de la construction carrée itérative. Celle-ci est assez simple, contrairement à celle de la construction carrée modifiée, qui complique beaucoup le treillis

associé.

Résultats.

Dans les deux parties qui suivent nous présentons les résultats de notre recherche. Ils ont fait l'objet de deux conférences ([76], [36]) et de deux articles publiés ([77], [37]), ainsi que d'un rapport de recherche [78]. La deuxième partie réunit les aspects liés aux distances minimales. Après un bref rappel des méthodes utilisées, nous présentons l'algorithme de recherche de mots de petits poids, dû à Canteaut et Chabaud [18]. Par la suite, nous appliquons cet algorithme ainsi que la décomposition carrée modifiée aux cas des duaux des codes BCH étendus (EBCH$^\perp$).

La Section 4.2 est dédiée aux rappels. En particulier nous expliquons comment mettre un code affine-invariant en ordre standard des bits. Nous présentons la construction (liée à cet ordre) de la matrice de contrôle des codes EBCH. Nous montrons ainsi que les codes affines-invariants admettent une construction carrée modifiée récursive (Théorème 4.1).

L'intérêt de toute décomposition d'un code donné est le transfert de certaines propriétés des codes composants au *grand* code. Ainsi, une borne supérieure pour la dimension du code est induite par des bornes sur les dimensions des codes composants appelés B et $(A/B, \widetilde{A/B})$ (voir la première *Remarque importante* dans la Section 4.2). Pour cela nous consacrons une section entière à l'étude du code B, en considérant les codes cycliques primitifs (de longueur 2^m) en tant que codes dans une algèbre de corps. Nous donnons de nouvelles bornes supérieures pour la dimension de B (Proposition 4.3). L'étude du code B se prolonge dans la Section 4.4, dont le noyau est la comparaison entre ce code et les codes de Reed-Muller. Ceci est obtenu par la décomposition carrée modifiée de tous les codes EBCH$^\perp$ de longueur $n \leq 512$ ou en utilisant la borne précédemment obtenue.

Une analyse détaillée de tous les résultats numériques (listés en Annexes A.1 et A.2) est réalisée en Section 4.5. Nous remarquons notamment que les bornes supérieures de d_B et $d_{(A/B,\widetilde{A/B})}$ ne sont pas très éloignées l'une de l'autre et que le mot de plus petit poids du code EBCH$(2^m, \delta)$ a été trouvé dans le code $(A/B, \widetilde{A/B})$, à trois exceptions près pour $n \leq 512$ (voir *Remarque importante* en Section 4.5). De plus, les bornes obtenues en utilisant l'algorithme probabiliste et celles obtenues par la décomposition des codes sont identiques pour la plupart des codes. Ainsi, la décomposition du code sert plus à trouver des propriétés intrinsèques des codes, qu'aux résultats purement numériques.

En Section 4.6, nous donnons quelques réflexions liées à un résultat de Kasami *et al.* (voir le Théorème 4.2), en exploitant la construction carrée modifiée des codes concernés.

La troisième partie est consacrée au calcul des polynômes énumérateurs. On s'intéresse à deux classes de codes : les codes cycliques auto-duaux (*c.a.d.*) et les codes duadiques. On commence le Chapitre 5 par l'étude des polynômes générateurs des codes *c.a.d.* . Notamment, nous rappelons la forme générale de ces polynômes (voir la relation (5.7)) qui induit que les codes *c.a.d.* sont des codes cycliques à racines multiples. Par conséquent, ils héritent de la construction itérative $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ donnée par van Lint [65]. Nous généralisons cette construction en Section 5.2.1 et nous donnons une formule pour la matrice génératrice du code (à équivalence près), n'utilisant que les sommes et les produits de Kronecker de codes de longueurs beaucoup plus petites (voir la relation

(5.12)). Cette formule est utilisée dans la sous-section suivante pour obtenir le résultat central de ce chapitre, qui précise que tout code cyclique binaire à racines multiples est équivalent à une construction carrée bien déterminée (voir la preuve du Théorème 5.4). Le paragraphe suivant est consacré à un cas particulier : on prouve que les codes *c.a.d.* dont toutes les racines ont une multiplicité paire sont des sommes directes.

Nous appliquons tous ces résultats à l'énumération des codes *c.a.d.* de longueur $n \leq 120$ (Section 5.3.1) et au calcul de leurs polynômes énumérateurs (voir Théorème 5.8). La complexité du calcul est analysée en Section 5.3.2 ; il s'avère qu'elle dépend de la dimension du code complément A/B, donnée précisément par la relation (5.22) en fonction seulement du polynôme générateur du code de départ.

Pour une partie des codes *c.a.d.* , les polynômes énumérateurs ont été calculés en utilisant des formules simplifiées (voir Section 5.3.3) ; de plus, sachant que tous les codes *c.a.d.* sont de Type I [93], nous avons pu utiliser le code *ombre* (Section 5.3.4) pour le calcul du nombre des mots de petit poids. À la fin du chapitre, nous comparons les différentes méthodes utilisées.

Notre manuscrit se termine par le chapitre consacré aux codes duadiques. Cette classe de codes contient les codes résidus quadratiques (QR). Ces codes, ainsi que les codes quadratiques doublement circulants (QDC), peuvent avoir de très bons paramètres, certains étant même *extrémaux*. Contrairement aux codes BCH, leurs énumérateurs de poids ne sont pas connus même pour des petites longueurs. Nous avons donc entamé une étude de ces codes. La solution proposée ici n'est pas liée à la décomposition de ces codes, mais exploite la théorie des invariants (rappelée en Section 6.3) et le calcul de mots de petit poids dans ces codes via un algorithme dû à Wassermann (voir Section 6.4). On obtient ainsi les énumérateurs des poids de tous les codes duadiques et QDC binaires de longueur $n \leq 152$ (à une exception près) et des meilleurs codes QDC et QR étendus ternaires pour $n \leq 96$.

Tous les résultats numériques sont listés en Annexe.

Notations

$C[n,k,d]$: code linéaire C de longueur n, de dimension k et de distance minimale d ;

G : matrice génératrice ;

$d^{\perp}$: distance duale ;

$d^{\perp}_{\max}$: borne supérieure pour la distance duale ;

k_A, d_B ou G_C : paramètres des codes indexés suivant le code qu'ils caractérisent ;

$\mathbb{F}_q$: corps de Galois à q éléments ;

$\mathbf{c}$: mot de code (toujours en gras) ;

$\overleftarrow{\mathbf{c}} = (c_n, c_{n-1}, \ldots, c_1)$: inverse du mot de code $\mathbf{c} = (c_1, \ldots, c_{n-1}, c_n)$;

$\mathbf{1}$: vecteur dont toutes les coordonnées valent 1 ;

$\mathbf{0}$: vecteur dont toutes les coordonnées valent 0 ;

$\mathrm{wt}(\mathbf{c})$: poids de Hamming du vecteur $\mathbf{c}$;

A_i : nombre des mots de poids i dans un code ;

W_C : polynôme énumérateur des poids du code linéaire C ;

W_e : partie de poids pair de W_C ;

W_o : partie de poids impair de W_C ;

$\mathbf{a} \cdot \mathbf{b}$: produit scalaire des vecteurs $\mathbf{a}$ et $\mathbf{b}$;

$\mathbf{ab}$: produit booléen de $\mathbf{a}$ et $\mathbf{b}$;

$C^{\perp}$: code dual du code C (voir déf. 1.4) ;

G_i : i-ème ligne de la matrice G ;

G^t : *transposée* d'une matrice G ;

H : matrice de contrôle ;

0 : matrice nulle ;

Id_k : matrice identité de rang k ;

(Id_k, M) : matrice en forme systèmatique ;

$\mathcal{S}_n$: le groupe symétrique de n éléments ;

$\pi(i)$: l'image de i sous l'action d'une permutation π ;

$\pi(\mathbf{c}) = \mathbf{b}$: si $c_i = b_{\pi(i)}$ pour tout $1 \leq i \leq n$;

$i \,(\mathrm{mod} j)$: le reste de la division de i par j (i, j deux naturels) ;

fsd : code formellement auto-dual ;

c.a.d. : code cyclique auto-dual ;

$\mathrm{Gr}(C)$: le groupe du code C ;

$Aut(C)$: le groupe d'automorphismes de C ;

$PSL_2(p)$: le groupe de permutations sur l'ensemble $\{\infty, 0, \ldots, p-1\}$, voir déf. 1.19 ;

σ : la permutation shift $(0, 1, \ldots, n-1)$;

μ_a : multiplicateur (permutation), voir définition 1.14 ;

$R_n = \mathbb{F}_q[x]/(x^n - 1)$: l'anneau commutatif des classes des polynômes dans$\mathbb{F}_q[x]$ modulo $x^n - 1$;

$g(x)$: d'habitude, polynôme générateur du code ;

$\deg g$: degré du polynôme g ;

$\overleftarrow{g}(x)$: polynôme réciproque de g ;

$e(x)$: d'habitude, polynôme idempotent du code ;

I ou I_C : ensemble de définition d'un code cyclique C (voir aussi la déf. 5.2) ; ensemble d'information dans la description de l'algorithme Canteaut-Chabaud ;

φ : la fonction d'Euler ;

$|T|$: cardinal de l'ensemble T ; nombre de lignes d'une matrice en Section 5.3.2 ;

$\mathrm{BCH}(2^m - 1, \delta)$: codes Bose-Chaudhuri-Hocquenghem primitifs, de longueur $2^m - 1$ et de distance construite δ ;

EBCH : code BCH étendu ;

δ : distance construite des codes BCH, voir déf. 1.16 ;

Q : le sous-groupe des résidus quadratiques ;

N : l'ensemble des non-résidus quadratiques ;

QR, QR_p : codes résidus quadratiques, voir déf. 1.17 ;

XQR : codes QR étendus ;

XQ_{n-1} : code QR étendu de longueur n en Appendix C ;

∞ : indice de la nouvelle coordonnée dans le code étendu ;

$\mathrm{RM}(r, m)$: code de Reed-Muller de longueur 2^m et d'ordre r ;

$G(r, m)$: matrice génératrice du code $\mathrm{RM}(r, m)$;

$k_{r,m}$: dimension du code $\mathrm{RM}(r, m)$ (voir Prop. 1.5) ;

$d_{r,m}$: distance minimale du code $\mathrm{RM}(r, m)$, égale à 2^{m-r} ;

EC : code étendu du code C (voir Section 2.1) ;

C^T : code poinçonné du code C par rapport à l'ensemble T ;

C_T : code raccourci du code C par rapport à l'ensemble T ;

(C, D) : code de matrice génératrice (G_C, G_D) ;

$C \oplus D$: somme directe des codes C et D (voir déf. 2.4) ;

$C + D$: somme de C et D (voir déf. 2.4) ;

$S(C, D)$: construction (somme) $|\mathbf{u}|\mathbf{u} + \mathbf{v}|$ de C et D (voir déf. 2.4) ;

$|C/D|^2$: construction carrée de premier niveau des codes C et D (voir déf. 2.4) ;

SC : abréviation pour la construction carrée ;

$\|A/B\|^2$: construction carrée modifiée de premier niveau (voir déf. 2.6) ;

LTSC : abréviation pour la construction carrée modifiée ;

$C \otimes D$: produit de Kronecker de deux codes C et D (voir déf. 2.5) ;

C/D : code complément de D par rapport à C (voir déf. 2.2) ;

$[C/D]$: ensemble des représentants des translatés de C/D (voir remarque 2.1) ;

$C_2 \subset C_1$: inclusion des codes, C_2 sous-code de C_1 ;

$G_2 \subset G_1$: si toutes les lignes d'une matrice G_2 sont des lignes d'une autre matrice G_1 ;

$G_C \backslash G_D$ ou $G_{C/D}$: si $G_D \subset G_C$, l'ensemble des $k_C - k_D$ lignes dans G_C qui n'appartiennent pas à G_D ;

$\Delta(r/(r-1), m-1)$: $G(r, m-1) \backslash G(r-1, m-1)$;

$G_C = G_D$: si $C = D$;

$G_C \sim G_D$: si $C \sim D$ (équivalence des codes) ;

$\sim$: de l'ordre de (en Section 6.3) ;

$|C_0/C_1/\ldots/C_m|^{2^m}$: construction carrée itérative de m-ième niveau, pour une séquence des codes $C_0 \supset C_1 \supset \ldots \supset C_m$ (voir Section 2.2) ;

$F_0 = [1, t]$ et $F_1 = [t+1, n]$: une partition du support $[1, 2t]$ du code C de longueur $2t$;

$\widetilde{G_{A/B}}$: matrice obtenue par la permutation des lignes de $G_{A/B}$ (voir Section 2.3) ;

$(\alpha_0, \alpha_1, \ldots, \alpha_{m-1})$: base de l'espace vectoriel $\mathbb{F}_{2^m}$ sur $\mathbb{F}_2$;

$\psi[X]$: le m-uplet binaire $(b_0, b_1, \ldots, b_{m-1})$, tel que $X = \sum_{i=0}^{m-1} b_i \alpha_i$;

ϕ : décomposition en base 2 d'un entier ;

$\mathbf{F}$: corps de décomposition $\mathbb{F}_{2^m}$ de $X^{2^m-1} - 1$ sur $\mathbb{F}_2$;

$\mathcal{A}$: l'algèbre $\mathbb{F}_2[\{\mathbf{F}, +\}]$ (Section 4.3) ;

ϕ_s : l'application $\mathbb{F}_2$-linéaire de $\mathcal{A}$ sur $\mathbf{F}$ définie par la relation (4.6) ;

$\preceq$: l'ordre partiel défini par la relation (4.12) ;

$C_s^{(b)}$ ou (s) : classe cyclotomique de s, modulo b ;

$(s)^i$, $i > 1$: la classe (s) répétée i fois ;

$\equiv$: équivalence modulo ;

α : racine de l'unité ;

$M_s^{(b)}(x)$: polynôme minimal de α^s sur $\mathbb{F}_2$ (voir déf. 5.1) ;

C^i : le code cyclique de polynôme générateur $g_1 \dots g_i$ (spécifique au Chapitre 5) ;

$C^{i,j}$: la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de C^i et C^j ;

$C^{i/j}$: C^i/C^j pour $i<j$; l'ensemble des représentants des translatés de C^j dans C^i ;

$C^{i,\ell}$: la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de $C^{i,j}$ et $C^{k,\ell}$, pour $i<j<k<\ell$;

G^i, $G^{i,j}$, $G^{i/j}$, resp. $G^{i,\ell}$: les notations similaires pour leurs matrices génératrices ;

$G_{(2,2)}$: matrice définie en Section 5.2.1 ;

$G_{(2^a,2^a)}$: matrice définie en Section 5.2.1 ;

ℓ^i : pour $i \in [1,\, 2^{a-1}]$, la i-ème ligne de $G_{(2^{a-1},2^{a-1})}$;

r^i : pour $i \in [1,\, 2^a]$, la i-ème ligne de $G_{(2^a,2^a)}$;

p.p.c.m. : plus petit commun multiple ;

p.g.c.d. : plus grand commun diviseur ;

C_0, C_2 : sous-codes d'un code auto-dual C (déf. 5.4) ;

S : ombre d'un code ;

B_i : nombre de mots de poids i dans l'ombre d'un code ;

QDC : codes quadratiques doublement circulants ;

$\mathcal{B}_p(r,s,t)$: notation du code QDC en Appendix C, construit comme en Section 6.2 ;

D : code duadique en général ;

C_1, C_2 et D_1, D_2 : paires des codes duadiques définis en Section 6.1 ;

S_1, S_2, T_1, T_2 : ensembles de définition des codes duadiques ;

$\epsilon = n \pmod 8$ (en Section 6.1) ;

$W_1, \dots, W_5$, W_2', W_3', W_5' : polynômes invariants définis en Section 6.3 ;

$\lfloor$, $\lceil$: partie entière inférieure, resp. supérieure ;

$\binom{n}{k}$: nombre de combinaisons de k éléments parmi n ;

Notations spécifiques au Chapitre 3

$\mathcal{E}(C)$: encodeur du code C ;

T : treillis d'un code ;

$S_i, 1 \le i \le n$: espace de l'encodeur au temps i ;

s_0, s_f : sommet initial, resp. final, du treillis ;

$(|S_0|, |S_1|, \dots, |S_n|)$: profil de la complexité des espaces des états ;

$\rho_i = \log_2 |S_i|$: dimension de S_i ;

$(\rho_0, \rho_1, \dots, \rho_n)$: profil des dimensions des espaces des états ;

$\rho_{\max}$: maximum des dimensions des états ;

TOGM : matrice adaptée à la structure du treillis (voir déf. 3.2) ;

$\mathbf{g_i}$: i-ème ligne d'une matrice $G = (g_{i,j})$;

span($\mathbf{g}$), aspan($\mathbf{g}$) : étendue, resp. étendue active, de la ligne $\mathbf{g}$ (voir déf. 3.3) ;

a_i : bit d'information ;

G_i^p, G_i^f, G_i^s : sous-ensembles des lignes de G (voir déf. 3.4) ;

A_i^p, A_i^f, A_i^s : sous-ensembles des bits d'information correspondant aux lignes de G_i^p, G_i^f, G_i^s ;

$C_{i,j}$: sous-code de C qui contient tous les mots de C de la forme

$$(0, 0, \ldots, 0, u_{i+1}, u_{i+2}, \ldots, u_j, 0, 0, \ldots, 0) ;$$

$C_{0,i}$, $C_{i,n}$: code préfixe, resp. suffixe de C par rapport à l'instant i ;

$p_{i,j}(C)$: code linéaire de longueur $j-i$ obtenu en enlevant les i premières et les $n-j$ dernières composantes de chaque mot de code de C ;

$C_{i,j}^{\mathrm{tr}}$: le code $p_{i,j}(C_{i,j})$;

$k_{i,j}$: dimension de $C_{i,j}^{\mathrm{tr}}$;

$L(s_i, s_j)$: les chemins du treillis qui relient un sommet s_i au niveau i avec un sommet s_j au niveau j du treillis, $0 \leq i < j \leq n$;

$I_i(\mathbf{g}^*)$: défini par (3.16) ;

$U = \{h_0, h_1, \ldots, h_\ell\}$: bornes des sections dans un treillis ℓ-sectionné ;

$\rho_{\ell,\max}$: $\max\limits_{0 \leq j \leq \ell} \rho_{h_j}$, dimension maximale dans un treillis ℓ-sectionné ;

$\mathbf{h}_i$: i-ème *colonne* de la matrice de contrôle H ;

H_i : $[\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_i]$;

$\ell(s)$: indice de l'état s (voir déf. 3.8).

Première partie

Préliminaires

Table des figures

1

Généralités sur les codes correcteurs

Dans ce chapitre nous présentons les notations et les objets que nous utiliserons tout au long de ce document. Notre principale référence pour la théorie des codes est [41]. Pour des compléments sur les corps finis, voir [68].

Définition 1.1 Un code linéaire $C[n,k]$ *de* longueur n *et de* dimension k *sur le corps de Galois à* q *éléments* $\mathbb{F}_q$ *est un sous-espace linéaire de dimension* k *dans* $\mathbb{F}_q^n$. *Un vecteur de* C *est appelé* mot de code. *Toute matrice* $k \times n$ *dont les lignes sont* k *vecteurs de* C *linéairement indépendants est* une matrice génératrice, *notée habituellement* G.

Remarque 1.1 Désormais, les vecteurs seront indiqués par des lettres **en gras**, pour les distinguer de leurs coordonnées. La notation $\mathbf{0}$ (resp. $\mathbf{1}$) désigne un vecteur dont toutes les coordonnées valent 0 (resp.1). De plus, il existe des matrices particulières :

- la matrice *nulle*, notée par 0 quand on déduit aisément du contexte qu'il s'agit d'une matrice, et non pas d'une entrée nulle ;
- la matrice identité de rang k, notée Id_k, dont les seules k entrées non-nulles valent 1 et sont sur la diagonale principale ;
- les matrices *sous forme systématique* (Id_k, M) ;
- la *transposée* d'une matrice G, notée par G^t.

Définition 1.2 *Un ensemble* $I \subset N$ *de cardinal* k *est un* ensemble d'information *pour le code* C *si et seulement si il existe une matrice génératrice en forme systématique* $(Id_k, Z)_I$ *de* C, *où* Id_k *est la matrice identité* $k \times k$. *L'ensemble* $J = N \backslash I$ *est appelé* ensemble de redondance.

Nous travaillons, sauf précision supplémentaire, avec des codes binaires et la distance sera la distance de Hamming. Comme d'habitude, wt($\mathbf{c}$) désigne le *poids de Hamming* du vecteur $\mathbf{c} = (c_1, ..., c_n)$, c'est-à-dire le nombre de coordonnées non-nulles de $\mathbf{c}$.

Définition 1.3 La distance minimale *de* C *est le plus petit poids des mots de* C :

$$d = \min\{wt(\mathbf{a} - \mathbf{b}) \ : \ \mathbf{a}, \mathbf{b} \in C, \mathbf{a} \neq \mathbf{b}\} = \min\{wt(\mathbf{c}) \ : \ \mathbf{c} \in C, \mathbf{c} \neq \mathbf{0}\}.$$

Le code C *est dit* $(d-1)$*-détecteur d'erreurs et* $\lfloor (d-1)/2 \rfloor$*-correcteur d'erreurs.*

La distance minimale du code est ainsi un paramètre essentiel pour mesurer*la performance* du code.

Définition 1.4 *Le dual $C^{\perp}$ d'un code linéaire binaire $C[n, k]$ est le code linéaire de longueur n et dimension $n-k$, orthogonal de C pour le produit scalaire usuel*

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i \,(\mathrm{mod}\, q).$$

La matrice génératrice du code $C^{\perp}$, noté habituellement H, est appelée matrice de contrôle *de C car :*

$$\mathbf{c} \in C \Longleftrightarrow \mathbf{c}H^t = 0. \tag{1.1}$$

La distance minimale de $C^{\perp}$, appelée ici distance duale, sera notée $d^{\perp}$.

Remarque 1.2 Pour la définition de la distance duale des codes *nonlinéaires* voir [70, Ch.5, p.139].

Dans le chapitre 4, nous donnons des bornes supérieures pour la distance duale ; elles seront notées par $d^{\perp}_{\max}$.

Remarque 1.3 Pour distinguer entre plusieurs codes, les paramètres du code seront indicés par le code qu'ils caractérisent (exemple : k_A, d_B ou G_C).

Définition 1.5 *Un code C est appelé* auto-dual *s'il est égal à son dual (i.e. $C = C^{\perp}$). Un code binaire auto-dual dont tous les poids sont divisibles par 4 est appelé* doublement pair *ou* de Type II *; autrement il est un code auto-dual* de Type I. *Un code* ternaire *auto-dual est dit* de Type III *si tous ses poids sont multiples de* 3.

Notons que la longueur d'un code de Type II est un multiple de huit.

Définition 1.6 *Le* polynôme énumérateur des poids *du code linéaire C est défini par*

$$W_C(x, y) = \sum_{\mathbf{c} \in C} x^{n - wt(\mathbf{c})} y^{wt(\mathbf{c})} = \sum_{i=1}^{n} A_i x^{n-i} y^i,$$

où A_i désigne le nombre des mots de poids i dans C. La suite :

$$A_0,\, A_1,\, \ldots,\, A_i,\, \ldots,\, A_n$$

est appelée distribution des poids *du code C.*

Dans beaucoup de cas, il s'avère qu'un autre ordre des coordonnées d'un code est plus adapté que l'ordre initial. Nous sommes donc souvent amenés à travailler avec des permutations des coordonnées d'un code.

Définition 1.7 *Soit $n \geq 2$ et soit π une permutation dans le groupe symétrique de n éléments $\mathcal{S}_n$, qui agit sur $\{1, 2, \ldots, n\}$; alors $\pi(i)$ désigne* l'image de i sous l'action de π.

On appelle k-cycle de $\{1, 2, \ldots, n\}$ toute permutation π telle qu'il existe $k \in \{2, 3, \ldots, n\}$ et $a_1, a_2, \ldots, a_k \in \{1, \ldots, n\}$, deux à deux distincts, tels que :

$$\begin{cases} \pi(a_1) = a_2, \pi(a_2) = a_3, \ldots, \pi(a_k) = a_1, \\ \forall\, i \in \{1, \ldots, n\} \backslash \{a_1, \ldots, a_k\}, \quad \pi(i) = i. \end{cases} \tag{1.2}$$

L'ensemble $\{a_1, a_2, \ldots, a_k\}$ est appelé le support de π, et on note

$$\pi = (a_1, a_2, \ldots, a_k).$$

Exemple : La permutation $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 5 & 2 & 4 & 3 \end{pmatrix}$ est le 3-cycle (2, 5, 3) et la permutation $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}$ est le 5-cycle (1, 2, 3, 4, 5).

Notation : Si C est un code de longueur n, $\mathbf{c} = (c_1, \ldots, c_n) \in C$ et $\pi \in \mathcal{S}_n$, nous notons

$$\pi(\mathbf{c}) = \mathbf{b} \text{ si } c_i = b_{\pi(i)},\ i \in \{1, 2, \ldots, n\}.$$

Exemple : Si $\pi = (1, 2, 3, 4, 5)$ et $\mathbf{c} = (1, 0, 1, 0, 0)$ alors $\pi(\mathbf{c}) = (0, 1, 0, 1, 0)$.

Définition 1.8 *Une* transformation monomiale *est une permutation qui agit sur n coordonnées, suivie (ou non) par la multiplication de certaines colonnes de la matrice génératrice par des éléments non-nuls dans $\mathbb{F}_q$. Deux codes sur $\mathbb{F}_q$ sont dits* monomialement équivalents *si on peut passer de l'un à l'autre par une transformation monomiale.*

Deux codes monomialement équivalents ont le même polynôme énumérateur, mais deux codes qui ont le même polynôme énumérateur ne sont pas forcément monomialement équivalents.

Un code qui a le même polynôme énumérateur que son dual est appelé*formellement auto-dual (fsd)*. Si C est (monomialement) *équivalent* à son dual, alors C est nommé *isodual*.

Définition 1.9 *L'ensemble de transformations monomiales qui envoie un code C en lui-même forme* le groupe du code, *noté $Gr(C)$.*

Proposition 1.1 *[84, Th.66, p.91]* On a $\mathrm{Gr}(C) = \mathrm{Gr}(C^{\perp})$.

Une classe très importante de codes sont les codes cycliques : ils contiennent notamment les codes BCH[1], qui sont largement utilisés. Nous présentons ici seulement le cas des codes cycliques de longueur n sur $\mathbb{F}_q$ tels que p.g.c.d.$(n, q) = 1$, appelés aussi codes *cycliques à racines simples*. Le cas des codes *à racines multiples* sera détaillé en Chapitre 5.

Définition 1.10 *Soit $\sigma = (0, 1, \ldots, n-1)$ la permutation appelée* shift, *c'est-à-dire :*

$$\sigma(i) = (i + 1)\, (mod\ n).$$

Un code $C[n, k]$ est appelé cyclique *si pour tout $\mathbf{c} = (c_0, \ldots, c_{n-2}, c_{n-1}) \in C$ on a aussi $\sigma(\mathbf{c}) = (c_{n-1}, c_0, \ldots, c_{n-2}) \in C$.*

[1] Bose-Chaudhuri-Hocquenghem, introduits en [13], [14] et [43]

Cela signifie que pour tout $\mathbf{c} \in C$, *tous les shifts* de $\mathbf{c}$ sont dans C.

À tout mot de code $(c_0, c_1, \ldots, c_{n-1}) \in C$ on peut faire correspondre le polynôme de degré au plus $n-1$:

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

Par cette *bijection*, on identifie le code C à un ensemble de polynômes modulo $x^n - 1$.

Proposition 1.2 *Notons $R_n = \mathbb{F}_q[x]/(x^n - 1)$ l'anneau commutatif des classes des polynômes dans $\mathbb{F}_q[x]$ modulo $x^n - 1$. Le code C est cyclique si est seulement si C est un idéal principal de R_n.*

Définition 1.11 *Le polynôme unitaire de plus petit degré de C est appelé* le polynôme générateur de C. *Si on note ce polynôme par $g(x) = g_0 + g_1 x + g_2 x^2 + \ldots + g_{n-k-1} x^{n-k-1} + x^{n-k}$, alors une matrice génératrice de C est de la forme :*

$$\begin{bmatrix} g_0 & g_1 & g_2 & \cdots & \cdots & g_{n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & \cdots & g_{n-k-1} & 1 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & 0 & \ldots & 0 & g_0 & g_1 & g_2 & \cdots & \cdots & g_{n-k-1} & 1 \end{bmatrix}. \tag{1.3}$$

La définition est correcte car ce polynôme est unique et engendre le code (voir [41, Ch. 1, Th. 5.2]). En particulier, $g(x)$ est un diviseur de $x^n - 1$, donc son calcul passe par la factorisation de $x^n - 1$, ce qui n'est pas toujours un problème facile. Pour éviter cela, on cherche d'autres générateurs ; on rappelle ainsi la définition et les propriétés des idempotents générateurs, dans le cas binaire.

Définition 1.12 *Un générateur $e(x)$ d'un idéal de R_n qui vérifie $e^2(x) = e(x)$ est appelé* idempotent générateur.

Par la suite nous allons énumérer quelques propriétés [41, p.53–59] :

- un polynôme *binaire* $e(x)$ est un idempotent dans R_n si et seulement si l'ensemble des puissances des x correspondant à des coefficients non-nuls dans l'expression de$e(x)$ est une réunion de classes cyclotomiques ;
- à chaque code cyclique de longueur n sur $\mathbb{F}_q$ tel que p.g.c.d.$(n, q) = 1$ correspond un unique idempotent qui est en plus générateur ; la matrice qu'il forme avec ses $k-1$ shifts est une matrice génératrice de C ;
- le polynôme générateur $g(x)$ du code d'idempotent $e(x)$ est obtenu par :

$$g(x) = \text{p.g.c.d.}(e(x), x^n - 1).$$

Définition 1.13 *Soit C un code cyclique de longueur n. Les racines du polynôme générateur forment* l'ensemble de définition, *ou* les zéros *du code.* Les nonzéros *du code C forment le complémentaire de cet ensemble par rapport à $\{0, \ldots, n-1\}$.*

Ces deux ensembles sont des réunions de classes cyclotomiques, sans multiplicités si la longueur du code et la caractéristique du corps sont premières entre elles, i.e. si le code est à racines simples.

Nous analysons par la suite $\mathrm{Gr}(C)$, quand C est un code cyclique de longueur n sur $\mathbb{F}_q$, avec p.g.c.d $(n, q) = 1$. Par définition, le shift σ appartient à ce groupe. Une autre permutation qui appartient à $\mathrm{Gr}(C)$ est définie par

$$\tau : i \longrightarrow qi \,(\mathrm{mod}\, n), \quad i \in \{0, 1, \ldots, n-1\}$$

(voir [84, p.89-90]). Ainsi le groupe engendré par σ et τ est un sous-groupe de $\mathrm{Gr}(C)$. En général, il est difficile d'en dire plus sur le groupe entier, sauf pour certaines classes particulières de codes, comme on le verra par la suite.

Dans cette thèse, nous devons souvent énumérer des codes cycliques. Cela sera fait *à équivalence près*, car deux codes équivalents ont les mêmes propriétés de correction et des paramètres identiques : même polynôme énumérateur (donc même capacité de correction, via la distance minimale) ; si l'un est décodable, l'autre l'est aussi etc. Tout d'abord, nous allons rappeler la définition suivante.

Définition 1.14 *Soit a un entier plus petit que n tel que p.g.c.d.$(a, n) = 1$. Alors la permutation*

$$\mu_a : i \longrightarrow ai \,(mod n), \quad i \in \{0, 1, \ldots, n-1\},$$

est appelée multiplicateur.

Un multiplicateur transforme un code cyclique en un code cyclique équivalent, ou il laisse le code inchangé si a est une puissance de q modulo n (voir [84, Th.65, p.90]). Si maintenant nous voulons savoir sous quelles conditions deux codes cycliques sont équivalents seulement par multiplicateur, nous faisons référence aux résultats de Pálfy généralisés par Huffman *et al.* en [47]. Une partie des résultats s'applique à n'importe quelle classe d'objets cycliques et peut s'étendre via des multiplicateurs généralisés. Pour la suite de notre étude, le théorème suivant sera suffisant.

Théorème 1.1 *[47, Th 1.1] Soient C et C' deux codes cycliques de longueur n sur un corps fini. Supposons que n vérifie une des conditions suivantes :*

- *p.g.c.d.$(n, \varphi(n)) = 1$ (φ étant* la fonction d'Euler*) ou $n = 4$;*
- *n est de la forme $n = pr$, avec $p > r$ deux nombres premiers, et de plus le p-sous-groupe de Sylow du groupe d'automorphisme de C est d'ordre p.*

Alors C et C' sont équivalents par permutation si et seulement si C et C' sont équivalents par multiplicateur.

Définition 1.15 *Soit C un code binaire de longueur 2^m. Alors les coordonnées peuvent être indicées par les éléments $X \in \mathbb{F}_{2^m}$. Le code est appelé* affine-invariant *s'il est invariant par les permutations $X \mapsto \alpha X + \beta$, où $\alpha, \beta \in \mathbb{F}_{2^m}$ et $\alpha \neq 0$.*

Remarque 1.4 Les codes BCH primitifs étendus et leurs duaux, les codes de Reed-Solomon étendus, les codes de Reed-Muller sont des codes affine-invariants ([70, 8.5 et 13.9], [80, 8.11]).

Nous allons maintenant définir les principales classes de codes que nous étudierons dans cette thèse.

Définition 1.16 *Soit δ un entier tel que $2 \leq \delta \leq n$. Un* code BCH binaire au sens strict de longueur n et de distance construite δ *est le code cyclique de dimension maximale tel que*

$$\{\alpha^i \ : \ 1 \leq i \leq \delta - 1\}$$

est un sous-ensemble des zéros du code, où α est un élément primitif de $\mathbb{F}_{2^m}$ et m le plus petit entier tel que n divise $2^m - 1$. Les codes BCH$(2^m - 1, \delta)$ sont appelés codes BCH primitifs.

Une autre classe importante de codes cycliques est celle des codes résidus quadratiques.

Définition 1.17 *Considérons le corps $\mathbb{F}_p$, avec p premier impair. Soit Q le sous-groupe des* résidus quadratiques *dans le groupe multiplicatif de $\mathbb{F}_p^*$ et N l'ensemble des* non-résidus quadratiques *:*

$$Q = \{a \ : \ \exists \ \ x \in \mathbb{F}_p^* \text{ tel que } a = x^2 (\mathrm{mod}\, p)\}, \quad N = \mathbb{F}_p^* \backslash Q.$$

Les codes résidus quadratiques sont construits en utilisant ces ensembles ; nous voulons en plus qu'on puisse passer de Q à N via une permutation des coordonnées :

$$\mu_a : i \longrightarrow ai \,(\mathrm{mod}\, n), \quad i \in \{0, 1, \ldots, n-1\} \text{ et p.g.c.d.}(a, n) = 1.$$

Définition 1.18 *Nous supposons que $2 \in Q$, donc $p = \pm 1 \,(\mathrm{mod}\, 8)$. Avec les notations*

$$e_1(x) = \sum_{i \in Q} x^i \quad et \quad e_2(x) = \sum_{i \in N} x^i,$$

un code est appelé code résidu quadratique *de longueur p (noté par QR ou QR_p) s'il a pour idempotent $e_1(x)$, $e_2(x)$, $1 + e_1(x)$ ou $1 + e_2(x)$.*

Par conséquent, il existe quatre codes QR de paramètres $[p, (p+1)/2]$, resp. $[p, (p-1)/2]$, qui sont liés par des relations bien définies selon que $-1 \in Q$ (donc $p = 1 \,(\mathrm{mod}\, 8)$) ou $-1 \in N$ (donc $p = -1 \,(\mathrm{mod}\, 8)$) (voir [82, Th.69, resp. Th.70]).

Les codes QR peuvent être étendus de façon classique en rajoutant un bit de contrôle de parité. On utilisera pour eux les notations classiques XQR ou XQ, et la nouvelle coordonnée sera indicée par ∞.

Proposition 1.3 *Les codes XQR sont auto-duaux de Type II si $p = -1 \,(\mathrm{mod}\, 8)$ et ne sont pas auto-duaux si $p = 1 \,(\mathrm{mod}\, 8)$.*

Définition 1.19 *Pour tout nombre premier p, on définit $PSL_2(p)$ le groupe de permutations sur l'ensemble $\{\infty, 0, \ldots, p-1\}$, engendré par les permutations suivantes ($i \in \mathbb{F}_p$) :*

$$\begin{aligned}
\sigma &: \ i \to i + 1 \,(\mathrm{mod}\, p), \infty \to \infty, \\
\mu_a &: \ i \to ai \,(\mathrm{mod}\, p), \ \textit{pour } a \in Q, \ \infty \to \infty, \\
\rho &: \ i \to -\frac{1}{i \,(\mathrm{mod}\, p)}, \ i \neq 0, \ 0 \to \infty, \ \infty \to 0.
\end{aligned}$$

L'ordre de $PSL_2(p)$ est $(p-1)p(p+1)/2$ (voir [70, p.491]).

Proposition 1.4 (**Gleason & Prange** [41, Th.12.4, p.118]) *$PSL_2(p)$ est contenu dans le groupe d'automorphisme d'un code XQR sur $\mathbb{F}_p$.*

Preuve. Voir [8] pour une démonstration dans le cas binaire. ◇

Plus de résultats, ainsi qu'une généralisation de ces codes, seront présentés en Chapitre 6.

Pour définir les codes de Reed-Muller, nous rappelons d'abord la définition du produit booléen.

Définition 1.20 *Soit $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ et $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ deux n-uplets.* Le produit booléen *de $\mathbf{a}$ et $\mathbf{b}$ est défini par*

$$\mathbf{ab} = (a_1 \cdot b_1, a_2 \cdot b_2, \ldots, a_n \cdot b_n),$$

où '$\cdot$' est le 'et' logique, *i.e. $a_i \cdot b_i = 1$ si et seulement si $a_i = 1$ et $b_i = 1$.*

La définition suivante est équivalente à celle, usuelle, qui utilise les fonctions booléennes ; les vecteurs $\mathbf{v}_i$ sont les lignes d'une matrice dont les colonnes sont formées par les éléments de $\mathbb{F}_{2^m}$, écrits par ordre croissant en tant que m-uplets binaires. On ajoute tous les produits d'ordre $\leq r$ de ces vecteurs et le mot $\mathbf{1}$ pour obtenir une base du code.

Définition 1.21 *Pour tout $1 \leq i \leq m$, soit $\mathbf{v}_i$ le 2^m-uplet binaire :*

$$\mathbf{v}_i = (\underbrace{0, \ldots, 0}_{2^{i-1}}, \underbrace{1, \ldots, 1}_{2^{i-1}}, \underbrace{0, \ldots, 0}_{2^{i-1}}, \ldots, \underbrace{1, \ldots, 1}_{2^{i-1}})$$

obtenu en alternant 2^{m-i+1}-uplets de longueur 2^{i-1} formés uniquement par des 0 ou par des 1. Soit $\mathbf{1}$ le 2^m-uplet $(1, 1, \ldots, 1)$. Le code de Reed-Muller de longueur $n = 2^m$ et d'ordre r, noté par $RM(r, m)$, est engendré par l'ensemble suivant :

$$\begin{aligned} G(r,m) \quad = \quad & \{\mathbf{1}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m, \mathbf{v}_1\mathbf{v}_2, \mathbf{v}_1\mathbf{v}_3, \ldots, \mathbf{v}_{m-1}\mathbf{v}_m, \ldots \\ & \ldots \textit{jusqu'à tous les produits de degré } r\}. \end{aligned} \tag{1.4}$$

On peut montrer que les vecteurs de $G(r, m)$ sont linéairement indépendants. En considérant ces vecteurs comme lignes d'une matrice, on déduit que $G(r, m)$ est une matrice génératrice du code $RM(r, m)$. Utilisant la définition précédente, on obtient le résultat suivant.

Proposition 1.5

1. *Le code de Reed-Muller de longueur $n = 2^m$ et d'ordre r est de dimension :*
$$k_{r,m} = 1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r}, \tag{1.5}$$
et de distance minimale $d_{r,m} = 2^{m-r}$.
2. *Le dual du code $RM(r, m)$ est le code $RM(m - r - 1, m)$. Par conséquent, un code de Reed-Muller est auto-dual si et seulement si m est impair et $r = (m-1)/2$.*

Exemple : Pour $r = 1$ et $m = 3$, la matrice génératrice du code de Reed-Muller construite via la définition précédente est :

$$G(1,3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{1.6}$$

Le code $RM(1,3)$ est donc de paramètres $[8,4,4]$ (en fait il est *équivalent* au code de Hamming étendu $[8,4,4]$) ; il est de plus auto-dual.

Comme on le verra au Chapitre 4, la construction donnée par la définition 1.21 correspond à un ordre des coordonnées appelé *ordre standard des bits*; on peut aussi l'obtenir en employant des fonctions booléennes (voir [85, Ch. 1.13]). Elle est identique à la construction$|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ itérative de ces codes, est adaptée à la décomposition du code (Section 2.2), à la construction des treillis de complexité minimale (voir Section 3.4.4) et, par conséquent, au calcul des mots de petits poids dans un code (exemple : Section 4.2).

Une autre construction des codes de Reed-Muller, cette fois *en ordre cyclique des bits*, est indiquée dans [70, p. 383] et sera utilisée en Section 4.4.

2

Décompositions de la matrice génératrice

Dans ce chapitre nous présentons les décompositions qui concernent les matrices génératrices des codes, et tout particulièrement, les classes des codes EBCH et des codes auto-duaux. L'étude est centrée sur les constructions carrée et carrée modifiée.

Notre but est d'utiliser ces décompositions pour le calcul des distances minimales et des polynômes énumérateurs des poids. Ainsi nous voulons ramener l'étude des codes de grande longueur à l'étude des codes de longueur beaucoup plus petite, en itérant certaines constructions, comme dans la Section 2.2 pour les codes de Reed-Muller.

2.1 Constructions de nouveaux codes

Étant donné un code linéaire binaire C, *le code étendu* EC est obtenu en rajoutant un bit de contrôle de parité :

$$\mathbf{c} = (c_1, c_2, \ldots, c_n) \in C \Rightarrow e\mathbf{c} = (c_1, c_2, \ldots, c_n, c_\infty) \in EC \text{ où } c_\infty = \sum_{i=1}^{n} c_i.$$

L'opération inverse, qui consiste à enlever une coordonnée, produit *le code poinçonné*. Plus généralement, étant donné un ensemble $T \subset [1, n]$, on obtient les codes suivants :

- *le code poinçonné* (punctured code) par rapport à T, noté C^T : le code de longueur $n - |T|$ obtenu en supprimant les coordonnées indicées par les éléments de T.
- *le code raccourci* (shortened code) par rapport à T, noté C_T : le code de longueur $n - |T|$ obtenu en utilisant l'ensemble des mots de C nuls sur T, et en supprimant les coordonnées dans T.

Exemple : Considérons le code binaire $C[6, 3, 2]$ de matrice génératrice :

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

et choisissons $T = \{4, 5\}$. Évidemment, la matrice génératrice du code C^T est :

$$G^T = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Pour déterminer C_T, nous devons isoler les mots du code C dont la 4-ième et la 5-ième coordonnées sont nulles. On exclut aisément les lignes de G et la somme de ces trois lignes. Il reste l'ensemble {(000000), (110000), (101000), (011000)}. En supprimant les coordonnées dans T, nous obtenons le code engendré par :

$$G_T = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Définition 2.1 *On définit la* somme *de deux codes de longueur n, C et D, comme étant le code de longueur n*

$$C + D = \{\mathbf{u} + \mathbf{v} : \ \mathbf{u} \in C \ \text{ et } \ \mathbf{v} \in D\}.$$

Définition 2.2 *Considérons un code $C[n, k_C]$ et un sous-code $D[n, k_D]$ de C. On appelle* code complément *tout* code linéaire C/D $[n, k_C - k_D]$ *tel que $C = D + C/D$ et $C/D \cap D = \{\mathbf{0}\}$.*

Exemple : Soit C le code de Hamming étendu $[8, 4, 4]$ de matrice génératrice

$$G_C = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

et un sous-code de C, noté D, de matrice génératrice

$$G_D = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Comme D est engendré aussi par les lignes 2 et 3 de G_C, on peut construire une matrice génératrice du code C/D en utilisant les lignes 1 et 4 de G_C :

$$G_{C/D} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Notation : Si toutes les lignes d'une matrice G_2 sont des lignes d'une autre matrice G_1, on note

$$G_2 \subset G_1.$$

Exemple : Dans l'exemple précédent, $G_C \supset G_{C/D}$ mais on n'a pas $G_C \supset G_D$.
Notation : Si C est un code de matrice génératrice G_C et D un sous-code de C, nous considérons G_D le sous-ensemble de k_D lignes dans G_C qui engendrent D. Notons par $G_C \backslash G_D$ ou $G_{C/D}$ l'ensemble des $k_C - k_D$ lignes dans G_C qui n'appartiennent pas à G_D.

Remarque 2.1 Dans la définition du code complément, le code C pouvant être partitionné en $2^{k_C-k_D}$ *translatés* disjoints de D, la **notation** $[C/D]$ désignera par la suite *un ensemble complet des représentants* des translatés de cette partition. **En fait, les $2^{k_C-k_D}$ mots engendrés par $G_C \backslash G_D$ peuvent être utilisés en tant que** $[C/D]$.

Définition 2.3 *Deux matrices seront considérées* égales *(resp.* équivalentes*) si elles engendrent le même code (resp. un code équivalent).*

Définition 2.4 *Pour deux codes* arbitraires *C et D de longueur n nous définissons :*

- *La* somme directe *de C et D, soit le code de longueur* $2n$

$$C \oplus D = \{(\mathbf{u}, \mathbf{v}) : \ \mathbf{u} \in C \ \text{ et } \ \mathbf{v} \in D\}.$$

- *La* construction (somme) $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ *de C et D, notée $S(C,D)$, produit le code de longueur* $2n$

$$S(C, D) = \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \ \mathbf{u} \in C \ \text{ et } \ \mathbf{v} \in D\}.$$

- *La* construction carrée (SC) de premier niveau *(*1-level squaring construction*) produit le code de longueur* $2n$

$$|C/D|^2 = \{(\mathbf{u}+\mathbf{x}, \mathbf{v}+\mathbf{x}) : \ \mathbf{u}, \mathbf{v} \in D \ \text{ et } \ \mathbf{x} \in C/D\},$$

où D est un sous-code *de C et C/D est défini par la Définition 2.2.*

Remarque 2.2 Par un raisonnement simple nous obtenons que, *si D est un sous-code de C,* la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de C et D est identique à la construction carrée de ces codes. Il suffit de noter que $C = D + C/D$ et de passer des codes aux matrices génératrices :

$$\begin{bmatrix} G_C & G_C \\ 0 & G_D \end{bmatrix} = \begin{bmatrix} G_D & G_D \\ G_{C/D} & G_{C/D} \\ 0 & G_D \end{bmatrix} = \begin{bmatrix} G_D & 0 \\ 0 & G_D \\ G_{C/D} & G_{C/D} \end{bmatrix}.$$

Définition 2.5 *Le* produit de Kronecker *$C \otimes D$ de deux codes C et D (pas forcément de même longueur) est le code de matrice génératrice $G_C \otimes G_D$, obtenu en remplaçant chaque entrée $g_{i,j}$ dans G_C par la matrice $g_{i,j}G_D$, où G_C et G_D sont les matrices génératrices de C et D, respectivement.*

Exemple : Soit le code $C[4, 2, 1]$ de matrice génératrice

$$G_C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

et soit le code $D[3, 2, 2]$ de matrice génératrice

$$G_D = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

La matrice génératrice du code $C \otimes D$ est

$$G_{C\otimes D} = \begin{bmatrix} G_D & 0 & 0 & 0 \\ 0 & G_D & G_D & G_D \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

2.2 Construction carrée itérative et codes de Reed-Muller

Nous présenterons dans cette section une méthode de construction de longs codes en partant d'une séquence de codes de petite longueur. Les codes ainsi obtenus présentent entre autres l'avantage d'avoir un treillis simple (voir les paragraphes 3.4.4, 3.4.5 et la Section 3.5) et peuvent être décodés par une technique multi-étages. Pour plus de détails voir [64, Ch. 8].

Pour obtenir une construction carrée itérative de niveau m, nous avons besoin d'un code $C_0[n, k_0, d_0]$ et d'une séquence de sous-codes de C_0 comme suit :

$$C_0 \supset C_1 \supset \ldots \supset C_m, \tag{2.1}$$

où C_i est un code $[n, k_i, d_i]$, $1 \leq i \leq m$. Sans perdre en généralité, nous pouvons supposer qu'entre les matrices génératrices de ces codes, il existe une relation analogue à (2.1), c'est-à-dire

$$G_0 \supset G_1 \supset \ldots \supset G_m. \tag{2.2}$$

En construisant la chaîne des partitions

$$C_0/C_1,\ C_0/C_1/C_2, \ldots, C_0/C_1/C_2/\ldots/C_m \tag{2.3}$$

on considère, comme en Remarque 2.1, la matrice génératrice de C_i/C_{i+1} comme étant

$$G_{i/(i+1)} = G_i \backslash G_{i+1}.$$

La matrice génératrice de la construction carrée de **premier niveau** $|C_0/C_1|^2$ est donnée par

$$\begin{bmatrix} G_1 & 0 \\ 0 & G_1 \\ G_{0/1} & G_{0/1} \end{bmatrix} = Id_2 \otimes G_1 + (1,1) \otimes G_{0/1}.$$

Passons maintenant de la construction de premier niveau à celle du **deuxième niveau**. En première étape nous construisons les codes $U := |C_0/C_1|^2$ et $V := |C_1/C_2|^2$ via la construction carrée de premier niveau. Par (2.1) on a $V \subset U$ et on peut construire

$$\begin{aligned} |C_0/C_1/C_2|^4 &= |U/V|^2 = \{(\mathbf{a}+\mathbf{x}, \mathbf{b}+\mathbf{x}) : \ \mathbf{a}, \mathbf{b} \in V \text{ et } \mathbf{x} \in [U/V]\} \\ &= \{(\mathbf{a}+\mathbf{x}, \mathbf{b}+\mathbf{x}) : \ \mathbf{a}, \mathbf{b} \in |C_1/C_2|^2 \text{ et } \mathbf{x} \in [|C_0/C_1|^2/|C_1/C_2|^2]\}. \end{aligned} \tag{2.4}$$

Nous allons montrer comment déduire une matrice génératrice pour cette construction. Préalablement, il faudra calculer $G_{U/V} = G_U \backslash G_V$ comme suit :

$$G_U = \begin{bmatrix} G_1 & 0 \\ 0 & G_1 \\ G_{0/1} & G_{0/1} \end{bmatrix} = \begin{bmatrix} G_2 & 0 \\ G_{1/2} & 0 \\ 0 & G_2 \\ 0 & G_{1/2} \\ G_{0/1} & G_{0/1} \end{bmatrix} = \begin{bmatrix} G_2 & 0 \\ 0 & G_2 \\ G_{1/2} & G_{1/2} \\ 0 & G_{1/2} \\ G_{0/1} & G_{0/1} \end{bmatrix} = \begin{bmatrix} G_V & \\ 0 & G_{1/2} \\ G_{0/1} & G_{0/1} \end{bmatrix},$$

d'où la matrice génératrice de $|C_0/C_1/C_2|^4$:

$$\begin{bmatrix} G_V & 0 \\ 0 & G_V \\ G_{U/V} & G_{U/V} \end{bmatrix} = \begin{bmatrix} G_2 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ G_{1/2} & G_{1/2} & 0 & 0 \\ 0 & 0 & G_2 & 0 \\ 0 & 0 & 0 & G_2 \\ 0 & 0 & G_{1/2} & G_{1/2} \\ 0 & G_{1/2} & 0 & G_{1/2} \\ G_{0/1} & G_{0/1} & G_{0/1} & G_{0/1} \end{bmatrix} = \begin{bmatrix} G_2 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & 0 & G_2 & 0 \\ 0 & 0 & 0 & G_2 \\ G_{0/1} & G_{0/1} & G_{0/1} & G_{0/1} \\ G_{1/2} & G_{1/2} & G_{1/2} & G_{1/2} \\ 0 & 0 & G_{1/2} & G_{1/2} \\ 0 & G_{1/2} & 0 & G_{1/2} \end{bmatrix}. \quad (2.5)$$

Rappellant que $G(r,m)$ désigne la matrice génératrice du code $RM(r,m)$, nous obtenons pour la matrice génératrice de $|C_0/C_1/C_2|^4$ la formule

$$\begin{aligned} & Id_4 \otimes G_2 + (1111) \otimes G_{0/1} + \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \otimes G_{1/2} \\ = \quad & Id_4 \otimes G_2 + G(0,2) \otimes G_{0/1} + G(1,2) \otimes G_{1/2}. \end{aligned} \quad (2.6)$$

Remarque 2.3 La construction carrée étant un cas particulier de construction $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ (Rem. 2.2), la distance minimale du code $|C_0/C_1|^2$ est $\min\{2d_0, d_1\}$. De façon analogue, la distance minimale de $|C_0/C_1/C_2|^4$ est $\min\{4d_0, 2d_1, d_2\}$.

De façon analogue on peut construire un code en forme carrée de**niveau** m en utilisant une séquence de codes de type (2.1) (voir [64, Ch. 8]). Nous allons illustrer cela par la construction des codes de Reed-Muller en utilisant des codes RM de longueur plus petite.

Soit i tel que $0 < i \leq r$ et soit la séquence de sous-codes de $RM(r,m)$:

$$RM(r,m) \supset RM(r-1,m) \supset \ldots \supset RM(r-i,m). \quad (2.7)$$

Nous rappelons que le code $RM(r,m)$ peut être obtenu par la construction $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ des codes $RM(r,m-1)$ et $RM(r-1,m-1)$ et, sachant que $RM(r-1,m-1) \subset RM(r,m-1)$, on retrouve une construction carrée en utilisant la Remarque 2.2. Avec**la notation :**

$$\Delta(r/(r-1), m-1) = G(r,m-1) \backslash G(r-1,m-1),$$

on obtient

$$\begin{aligned} G(r,m) &= \begin{bmatrix} G(r,m-1) & G(r,m-1) \\ 0 & G(r-1,m-1) \end{bmatrix} \\ &= \begin{bmatrix} \Delta(r/(r-1),m-1) & \Delta(r/(r-1),m-1) \\ G(r-1,m-1) & 0 \\ 0 & G(r-1,m-1) \end{bmatrix}. \end{aligned} \tag{2.8}$$

On a donc $RM(r,m) = |RM(r,m-1)/RM(r-1,m-1)|^2$; les relations analogues pour $RM(r,m-1)$ et $RM(r-1,m-1)$ donnent

$$\begin{aligned} RM(r,m) &= |RM(r,m-2)/RM(r-1,m-2)/RM(r-2,m-2)|^4 = \dots \\ \dots &\quad |RM(r,m-i)/RM(r-1,m-i)/\dots/RM(r-i,m-i)|^{2^i}. \end{aligned} \tag{2.9}$$

Pour la matrice génératrice on a la formule suivante :

$$G(r,m) = Id_{2^i} \otimes G(r-i,m-i) + \sum_{0 \le j < i} G(j,i) \otimes \Delta((r-j)/(r-j-1),m-i). \tag{2.10}$$

Un code est dit i-décomposable s'il peut être exprimé par une construction carrée de$i$-ème niveau, à partir d'une chaîne de sous-codes du type (2.1). La relation (2.9) implique que les codes $RM(r,m)$ sont i-décomposables, pour $1 \le i \le r$.

Dans le chapitre 5 nous donnerons une construction itérative des codes cycliques auto-duaux et, en application, quelques méthodes de calcul des polynômes des poids.

2.3 Construction carrée modifiée

Dans cette section nous présentons la construction carrée modifiée, introduite par Forney en [32] et qui s'avère très utile pour le calcul des mots de petits poids (voir Chapitre 4). La référence principale est l'article de Berger & Be'ery [6].

Définition 2.6 *Soit C un code de longueur $n = 2t$ et de dimension k. Soit $F_0 = [1, t]$ et $F_1 = [t+1, n]$ une partition du support $[1, 2t]$ du code C. Supposons que C satisfait :*

- *les codes poinçonnés par rapport à F_0 et à F_1 sont égaux (notons ce code A) ;*
- *les codes raccourcis par rapport à F_0 et à F_1 sont égaux (notons ce code B).*

Le code C est dit en forme linéaire carrée modifiée — ou code LTSC — *(*linear twisted squaring constructed). Il sera noté $C = \|A/B\|^2$.

Nous dirons par la suite qu'un code *est LTSC* s'il satisfait les propriétés décrites par la définition précédente. Ces propriétés étant dépendantes de l'ordre choisi pour les colonnes, tout code équivalent à un code LTSC est dit *mis sous la forme LTSC* (par une permutation des colonnes du code).

Remarque 2.4 La définition précédente implique que A et B sont deux codes de longueur t tels que B est un sous-code de A et $k_A + k_B = k$, où k_A et k_B désignent les dimensions de A, resp. B.

Essayons de donner une écriture pour la matrice génératrice d'un code LTSC. La deuxième condition dans la définition du code LTSC implique que la somme directe $B \oplus B$ est un sous-code de C. Fixons

$$\begin{bmatrix} G_B & 0 \\ 0 & G_B \end{bmatrix}$$

pour matrice génératrice de la somme directe. Maintenant notons par (G_g, G_d) la matrice génératrice du code complément de C par rapport à la somme directe. Via la première condition en définition 2.6, il en résulte que les matrices

$$\begin{bmatrix} G_g \\ G_B \\ 0 \end{bmatrix} \text{ et } \begin{bmatrix} G_d \\ 0 \\ G_B \end{bmatrix}$$

engendrent le code A.

Remarque 2.5 *On souligne que les matrices G_g et G_d engendrent le même code, plus précisément un système de représentants des translatés de B dans A, noté $[A/B]$. Le passage d'une matrice à l'autre se fait via des opérations sur les lignes, ce qui justifie la* **notation**

$$G_g = G_{A/B}, \;\; G_d = \widetilde{G_{A/B}}.$$

On trouve ainsi une matrice génératrice de C de la forme :

$$\begin{bmatrix} G_{A/B} & \widetilde{G_{A/B}} \\ G_B & 0 \\ 0 & G_B \end{bmatrix}. \tag{2.11}$$

Conséquences :

1. Via les notations en Chapitre 1, on a

$$\|A/B\|^2 = (A/B, \widetilde{A/B}) + (B \oplus B),$$

où $(A/B, \widetilde{A/B})$ désigne le code qui a pour matrice génératrice $(G_{A/B}, \widetilde{G_{A/B}})$. La relation entre les dimensions de ces codes est donc :

$$k = k_{(A/B, \widetilde{A/B})} + 2k_B. \tag{2.12}$$

2. Tout mot de code de $C = \|A/B\|^2$ est de la forme :

$$(\mathbf{b}_1 + \mathbf{d},\, \mathbf{b}_2 + \mathbf{d}'), \quad \mathbf{d}, \mathbf{d}' \in [A/B], \quad \mathbf{b}_1, \mathbf{b}_2 \in B.$$

Si $\mathbf{d} = \mathbf{d}'$ pour tous les mots, on retrouve la *construction carrée de premier niveau*

$$|A/B|^2 = \{(\mathbf{b}_1 + \mathbf{d},\, \mathbf{b}_2 + \mathbf{d}) : \quad \mathbf{d} \in [A/B], \quad \mathbf{b}_1, \mathbf{b}_2 \in B\},$$

qui est identique à la construction $|\mathbf{u}|\mathbf{u} + \mathbf{v}|$ (voir les Remarques 2.4 et 2.2).

Exemple : Considérons le code RM(1, 5) de longueur 32 et dimension 6 engendré par la matrice :

$$\left[\begin{array}{cccccccccccccccccccccccccccccccc}
1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&0\\
0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\
0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{array}\right].$$

Le mot **1** de longueur 32 peut être retrouvé en faisant la somme des lignes de cette matrice génératrice. On déduit $B[16, 1, 16]$, qui ne contient que le mot **1** de longueur 16. Le code $A[16, 5, 8]$ a pour matrice génératrice :

$$\left[\begin{array}{cccccccccccccccc}
1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1\\
0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\
0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1
\end{array}\right].$$

Enfin, la matrice génératrice $(G_{A/B}, \widetilde{G_{A/B}})$ est :

$$\left[\begin{array}{cccccccccccccccc|cccccccccccccccc}
0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\
0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1
\end{array}\right].$$

Dans ce cas, on trouve $G_{A/B} = \widetilde{G_{A/B}}$, donc une construction carrée.

Le théorème suivant sera très important par la suite. En fonction des paramètres qu'on voudra caractériser, ce résultat nous permettra de travailler sur un code ou sur son dual (voir Chapitre 4).

Théorème 2.1 *[6] Le dual de tout code LTSC est un code LTSC. Plus précisément, si* $C = \|A/B\|^2$ *alors* $C^\perp = \|B^\perp/A^\perp\|^2$.

Preuve. Pour le code $C^\perp$, considérons les codes raccourcis par rapport à F_0, resp. F_1. Notons ces codes W et Z et leurs matrices génératrices G_W, resp. G_Z. Le code de matrice génératrice

$$\begin{bmatrix} G_W & 0 \\ 0 & G_Z \end{bmatrix}$$

est alors un sous-code de $C^\perp$. Par conséquent, nous pouvons considérer le code complément de ce sous-code par rapport au code $C^\perp$, et nous noterons (G_X, G_Y) sa matrice génératrice. La matrice génératrice de $C^\perp$ peut alors s'écrire sous la forme

$$\begin{bmatrix} G_X & G_Y \\ G_W & 0 \\ 0 & G_Z \end{bmatrix}.$$

Via la définition du code dual nous obtenons $G_{C^\perp} G_C^t = 0$ donc

$$\begin{bmatrix} G_X & G_Y \\ G_W & 0 \\ 0 & G_Z \end{bmatrix} \begin{bmatrix} G_{A/B}^t & G_B^t & 0 \\ \widetilde{G_{A/B}}^t & 0 & G_B^t \end{bmatrix} = 0,$$

où la matrice de C est donnée par (2.11). Par conséquent on a les relations suivantes :

$$\begin{cases} G_X G_{A/B}^t + G_Y \widetilde{G_{A/B}}^t = 0 \\ G_X G_B^t = G_Y G_B^t = 0 \\ G_W G_{A/B}^t = G_W G_B^t = 0 \\ G_Z \widetilde{G_{A/B}}^t = G_Z G_B^t = 0. \end{cases}$$

La dernière relation implique $G_Z G_A^t = 0$ et donc $Z \subset A^\perp$. Supposons qu'il existe un mot $\mathbf{v} \in A^\perp \backslash Z$, alors $(0, \mathbf{v}) \in C^\perp$ donc $\mathbf{v} \in Z$, contradiction. Par conséquent $Z = A^\perp$ et de même pour W.

Par ailleurs $X + W \subset B^\perp$, et en raison de dimensions égales,

$$\begin{aligned} k_X + k_W &= k_{C^\perp} - k_Z = (n - k_C) - k_Z = [n - (k_A + k_B)] - (n/2 - k_A) \\ &= n/2 - k_B = k_{B^\perp}, \end{aligned}$$

on a l'égalité des codes. Le même raisonnement implique $Y + Z = B^\perp$, d'où la conclusion. $\diamond$

Exemple : Soit C le code de Reed-Muller de paramètres $r = 4$, $m = 9$. Sa matrice génératrice se décompose ainsi :

$$G(4,9) = \begin{bmatrix} \Delta(4/3,8) & \Delta(4/3,8) \\ G(3,8) & 0 \\ 0 & G(3,8) \end{bmatrix}.$$

Les codes A et B dans la construction LTSC (ici SC) sont $RM(4,8)$, resp. $RM(3,8)$. Les codes $B^\perp$ et $A^\perp$ correspondant à la décomposition du code dual sont $RM(3,8)^\perp = RM(4,8)$ et $RM(4,8)^\perp = RM(3,8)$, donc le code $RM(4,9)$ et son dual ont la même décomposition LTSC. Cela était prévisible car pour tout $r = (m-1)/2$, le code $RM(r,m)$ est auto-dual (voir Prop. 1.5)

La structure des codes LTSC les relie à d'autres classes de codes. Parmi elles, on s'intéresse aux codes symétriques et réversibles.

Définition 2.7 *Soit C un code de longueur paire $n = 2t$. Tout mot de code $\mathbf{c}$ est une paire $(\mathbf{c}_1, \mathbf{c}_2)$, où $\mathbf{c}_1$ est formé par les t premiers symboles de $\mathbf{c}$ et $\mathbf{c}_2$ par les t derniers symboles de $\mathbf{c}$. Le code C est appelé* symétrique *si :*

$$(\mathbf{c}_1, \mathbf{c}_2) \in C \iff (\mathbf{c}_2, \mathbf{c}_1) \in C.$$

Théorème 2.2 *[6] Un code linéaire symétrique C est un cas particulier de code LTSC. Précisément, $C = \|A/B\|^2$, avec $\widetilde{G_{A/B}} = EG_{A/B}$, où E^2 est égale à la matrice identité.*

Preuve. Soit C un code symétrique. En considérant les codes poinçonnés, la définition précédente implique alors $C^{F_0} = C^{F_1}$. De plus, si $(\mathbf{c}_1, \mathbf{0}) \in C$, alors $(\mathbf{0}, \mathbf{c}_1) \in C$ et, par conséquent,

$C_{F_0} = C_{F_1}$. Le code C est donc un code LTSC, et une matrice génératrice de ce code est donnée par(2.11). Le code étant symétrique, la matrice

$$\begin{bmatrix} \widetilde{G_{A/B}} & G_{A/B} \\ 0 & G_B \\ G_B & 0 \end{bmatrix}$$

est aussi une matrice pour ce code. Le passage entre les deux matrices se fait par multiplication avec une matrice inversible et, particulièrement ici, on peut considérer cette matrice de la forme

$$\begin{bmatrix} E & 0 \\ 0 & Id \end{bmatrix}$$

où Id est la matrice identité et E est une matrice inversible de $k_A - k_B$ lignes. On obtient donc :

$$\begin{bmatrix} E & 0 \\ 0 & Id \end{bmatrix} \begin{bmatrix} \widetilde{G_{A/B}} & G_{A/B} \\ G_B & 0 \\ 0 & G_B \end{bmatrix} = \begin{bmatrix} G_{A/B} & \widetilde{G_{A/B}} \\ G_B & 0 \\ 0 & G_B \end{bmatrix},$$

et finalement $G_{A/B} = E\widetilde{G_{A/B}} = E^2 G_{A/B}$, d'où la conclusion. ◇

Définition 2.8 *À chaque mot de code* $\mathbf{c} = (c_1, \ldots, c_{n-1}, c_n)$ *on associe son* inverse *:*

$$\overleftarrow{\mathbf{c}} = (c_n, c_{n-1}, \ldots, c_1).$$

Un code C est dit réversible *si pour tout* $\mathbf{c} \in C$ *le mot* $\overleftarrow{\mathbf{c}}$ *est dans C.*

Remarque 2.6 Dans la définition du code symétrique (ou réversible) il suffit de considérer pour $\mathbf{c}$ seulement les éléments d'une base du code. Ces générateurs vérifient ces propriétés par l'appartenance au code. Par ailleurs, ces propriétés sont invariantes par des combinaisons linéaires.

Exemple : Considérons le code RM(1, 5) engendré par la matrice $\mathcal{G}$:

$$\begin{bmatrix}
1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&0\\
0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\
0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{bmatrix}$$

Nous voulons montrer que ce code est symétrique et réversible. Pour cela nous travaillons sur la base donnée par les lignes de la matrice génératrice. À partir d'un mot $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$ on construit $(\mathbf{c}_2, \mathbf{c}_1)$, ce qui revient à interchanger les colonnes i et $n/2 + i$, $n = 32$, de la matrice $\mathcal{G}$, pour tout $i \in \{1, 2, \ldots, n/2\}$. Cette opération donne la matrice $\mathcal{G}_{\text{sym}}$ suivante :

$$\begin{bmatrix}
0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1\\
0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\
0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

et il suffit de vérifier qu'elle engendre le même code (l'ordre des coordonnées étant fixé). Par la suite, $\mathcal{G}_{\mathrm{sym},i}$ (resp. $\mathcal{G}_i$) désigne la i-ième ligne de la matrice $\mathcal{G}_{\mathrm{sym}}$ (resp. $\mathcal{G}$). Sachant que le mot $\mathbf{1}$ appartient aux deux codes, on note que $\mathcal{G}_{\mathrm{sym},i} = \mathbf{1} + \mathcal{G}_i$ pour $i \in \{1, 6\}$ et $\mathcal{G}_{\mathrm{sym},i} = \mathcal{G}_i$ pour $i \in \{2, 3, 4, 5\}$. Notre code est donc symétrique.

Maintenant pour vérifier que le code est réversible, il suffit d'interchanger les colonnes$i$ et $n+1-i$ de la matrice $\mathcal{G}$, pour tout $i \in \{1, 2, \ldots, n/2\}$. On obtient ainsi la matrice $\mathcal{G}_{\mathrm{rev}}$:

$$\left[\begin{array}{cccccccccccccccccccccccccccccccc}
0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0\\
1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}\right]$$

qui engendre le même code, car $\mathcal{G}_{\mathrm{rev},i} = \mathbf{1} + \mathcal{G}_i$ pour tout $i \in \{1, \ldots, 6\}$.

Proposition 2.1 *[6] Tout code réversible C de longueur paire est* équivalent *à un code LTSC symétrique et sa matrice génératrice G_C est de la forme :*

$$\begin{bmatrix} \overleftarrow{G_{A/B}} & \widetilde{G_{A/B}} \\ \overleftarrow{G_B} & 0 \\ 0 & G_B \end{bmatrix},$$

avec $\overleftarrow{G_{A/B}}$ et $\overleftarrow{G_B}$ obtenues en inversant l'ordre des colonnes dans $G_{A/B}$ et G_B.

Preuve. Soit C' le code obtenu par l'inversion de l'ordre des $n/2$ premières coordonnées de C ; donc C est équivalent à C'. Montrons maintenant que C' est un code LTSC. Considérons $(\mathbf{c}_1, \mathbf{c}_2) \in C'$, donc $(\overleftarrow{\mathbf{c}}_1, \mathbf{c}_2) \in C$. Mais C est réversible, donc $(\overleftarrow{\mathbf{c}}_2, \mathbf{c}_1) \in C$ et finalement $(\mathbf{c}_2, \mathbf{c}_1) \in C'$. Le code C' est par conséquent symétrique, donc LTSC (voir Théorème 2.2) et sa matrice génératrice est de la forme (2.11). ◇

Remarque 2.7 Chaque code réversible de longueur paire devient symétrique si on inverse l'ordre d'une moitié de ses coordonnées et réciproquement (voir la preuve précédente).

Définition 2.9 *Un code $C = \|A/B\|^2$ est un code LTSC* récursif *s'il est LTSC, les codes A et B sont LTSC, les codes composants de A et B sont LTSC et ainsi de suite. Le degré de récursivité est le nombre de pas dans cette décomposition.*

Comme nous allons voir en Théorème 4.1, tous les codes affines-invariants (de longueur 2^m) peuvent être mis sous forme LTSC (du m-ième degré de décomposition).

2.4 Constructions spécifiques aux codes auto-duaux

Dans cette section nous présentons quelques constructions qui concernent de classes de codes auxquelles nous nous sommes intéressés. Comme on le verra par la suite, ces constructions n'ont pas été effectivement utilisées pour le calcul des polynômes des poids, mais elles sont très utiles pour la construction des treillis *tail-biting* [30] et pour les algorithmes de décodage [31].

Définition 2.10 *Une matrice de la forme :*

$$G = \begin{bmatrix} g_0 & g_1 & \cdots & g_{\ell-2} & g_{\ell-1} \\ g_{\ell-1} & g_0 & \cdots & g_{\ell-3} & g_{\ell-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ g_1 & g_2 & \cdots & g_{\ell-1} & g_0 \end{bmatrix} \tag{2.13}$$

est appelée matrice circulante. *Si chaque élément* g_i, $0 \leq i \leq \ell - 1$, *est une matrice, alors* G *est appelée* matrice bloc-circulante.

La construction des matrices bloc-circulantes est liée à l'ordre des permutations.

Définition 2.11 *Soit* π *une permutation d'un code* C. L'ordre *de* π *est le plus petit entier* $\ell \geq 1$ *tel que* $\pi^\ell = \iota$, *où* ι *est la permutation identité.*

Évidemment, on ne peut pas mettre tous les codes $C[n, k, d]$ sous forme bloc-circulante. L'entier ℓ dans (2.13) doit vérifier

$$\ell \mid k, \quad \text{et} \quad \ell \mid n.$$

Supposons qu'un code contient une permutation d'ordre3 (c'est le cas de tous les codes résidus-quadratiques et de certains codes BCH). La matrice génératrice bloc-circulante est de la forme

$$G = \begin{bmatrix} A & B & 0 \\ 0 & A & B \\ B & 0 & A \end{bmatrix} \tag{2.14}$$

avec A et B deux matrices $k/3 \times n/3$ et 0 la matrice nulle $k/3 \times n/3$. Les paramètres de cette décomposition pour les codes XQR de longueur $n \in \{18, 24, 42, 48, 72\}$ sont donnés en [29]. De plus, la matrice génératrice du code $BCH(35, 15, 8)$ peut être mise en forme *presque bloc-circulante*

$$\begin{bmatrix} A & B & 0 & N \\ 0 & A & B & N \\ B & 0 & A & N \end{bmatrix}, \tag{2.15}$$

où A, B (resp. N) sont les matrices génératrices des deux codes $[10, 5, 3]$ (resp. du code $[5, 5, 1]$).

Nous présentons par la suite une construction doublement circulante dans le cas des codes résidus quadratiques. La principale référence est [48].

Définition 2.12 *Une* matrice doublement circulante *est une matrice* $k \times 2k$ *de la forme* (L, R) *où* L *et* R *sont deux matrices circulantes.*

Avant tout, rappelons que (voir Proposition 1.4)

$$PSL_2(p) \subset Aut(XQR_p)), \quad p \equiv \pm 1 \,(\mathrm{mod}\, 8)$$

et que toute permutation de $PSL_2(p)$ d'ordre $(p+1)/2$ est formée par deux cycles de longueur $(p+1)/2$. L'étude des matrices en forme doublement circulante commence avec l'observation de l'existence, pour $p \in \{7, 23\}$, d'une construction simple pour les codes XQR_p) [70, Ch. 16].

À partir d'un idempotent $\mathbf{c}$ du code XQR et d'une permutation spéciale $g \in PSL_2(p)$ on peut construire une matrice génératrice, notée $M(\mathbf{c}, g)$, ayant les lignes

$$\mathbf{c},\ g(\mathbf{c}),\ g^2(\mathbf{c}),\ \ldots,\ g^{(p-1)/2}(\mathbf{c}).$$

Il s'avère que, pour $p \in \{7,\ 23\}$, la matrice ainsi obtenue est sous la forme $(Id_{(p+1)/2}, A)$, avec A matrice circulante, donc une *matrice doublement circulante en forme systématique* À ce point, un simple raisonnement indique l'existence d'une forme systématique dès que l'une des matrices L ou R est inversible. Par conséquent, le problème à résoudre a la formulation suivante :

Est-il possible, pour n'importe quel code XQR, de trouver un mot de code $\mathbf{c}$ *et une permutation* $g \in PSL_2(p)$ *d'ordre* $(p+1)/2$ *tels qu'au moins une des matrices* L *et* R*, où* $M(\mathbf{c}, g) = (L, R)$*, soit inversible ?*

Dorénavant, les deux cycles d'une permutation $g \in PSL_2(p)$ d'ordre $(p+1)/2$ sont notés E et F. Une formulation équivalente en termes d'ensemble d'information (voir Définition 1.2) est la suivante :

Si $p \equiv \pm 1\,(mod 8)$ *et* $g = (E)(F)$ *est un élément de* $PSL_2(p)$ *d'ordre* $(p+1)/2$*, est-ce qu'un des deux cycles de* g *donne toujours un ensemble d'information pour le code* XQR_{p+1}*,* $(p+1)/2$*] ?*

Remarque 2.8 Dans le cas des codes XQR auto-duaux (donc pour $p \equiv -1 \pmod 8$) si un cycle donne un ensemble d'information pour XQR_p, l'autre cycle donne aussi un ensemble d'information pour le même code (voir [48, Prop. 2.5]).

Les résultats obtenus en [48] ont été prouvés en utilisant le lemme suivant dans une démonstration par l'absurde :

Lemme 2.1 *Supposons que le code* XQR_p *ne possède pas de matrice génératrice en forme doublement circulante. Alors le code* $XQR_p^{\perp}$ *a une matrice génératrice* G *qui satisfait :*

- G *est une matrice* $(p+1)/2 \times (p+1)$ *de la forme :*

$$G = \begin{bmatrix} U & 0 \\ 0 & V \\ S & T \end{bmatrix}, \tag{2.16}$$

  toutes les matrices U, V, S, T *ayant* $(p+1)/2$ *colonnes.*
- *chaque sous-matrice* U *et* V *est matrice génératrice d'un* code cyclique.
- *les matrices* $\begin{pmatrix} U \\ S \end{pmatrix}$ *et* $\begin{pmatrix} V \\ T \end{pmatrix}$ *sont aussi des matrices génératrices de codes cycliques.*

De manière théorique, Jenson a trouvé deux classes infinies de codes XQR qui admettent une construction doublement circulante, les hypothèses étant respectivement :

- $p \equiv 1 \pmod 8$ et $q = (p+1)/2$ sont premiers et 2 est une racine primitive $(\mathrm{mod}\, q)$ [48, Th. 3.3] ;
- $p \equiv 1 \pmod 8$ et $q = (p+1)/2$ sont premiers et $x^q - 1$ a trois facteurs irréductibles sur $\mathbb{F}_2$ [48, Th. 3.6].

Par ailleurs, l'existence de cette construction a été testée sur ordinateur pour tous les XQR_p, $p \leq 200$. Notamment deux exceptions ($p = 89$ et $p = 167$) prouvent que :

Proposition 2.2 *La construction doublement circulante* n'est pas possible pour tous *les codes XQR (auto-duaux ou pas).*

2.5 Constructions liées aux codes de longueur composée

Nous avons déjà présenté quelques constructions pour des codes de longueur composée :
- pour les codes de longueur 2^m la construction carrée itérative (les codes RM— voir Section 2.2) et en général, pour tous les codes affines-invariants, la construction LTSC (voir Section 2.3 et Chapitre 4) ;
- les constructions circulante et doublement circulante, notamment dans le cas des codes résidus quadratiques (voir section précédente) ;
- la construction $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ itérative (dans ce cas égale à la construction carrée itérative) des codes cycliques à racines multiples (voir Section 5.2.1).

Par la suite nous allons détailler la décomposition des codes BCH de longueur composée, proposée par Vardy & Be'ery en [101, Section 2]. Supposons $C[n,k]$ un code BCH et α une racine n-ième de l'unité. Soit T un sous-ensemble de l'ensemble des coordonnées $\{0,\,1,\,\ldots,\,n-1\}$. Tout au long de cette section, $C(T)$ désigne le code $C_{[0,\,n-1]\setminus T}$ raccourci par rapport à $[0,\,n-1]\setminus T$, c.-à-d. les mots du code C qui sont entièrement nuls en dehors de T, dont on enlève ces coordonnées nulles. Le code $C(T)$ est donc de longueur $|T|$.

Proposition 2.3 *[101] Soient T_1 et T_2 deux sous-ensembles de $[0,\,n-1]$. Supposons qu'il existe un $a \in [0,\,n-1]$ tel que :*

$$T_2 = a + T_1 \pmod n. \tag{2.17}$$

Alors $C(T_1) = C(T_2)$.

Supposons maintenant que C est de longueur composée, c'est-à-dire $n = n_1 n_2$.
Soit $I = \{i_1,\,i_2,\,\ldots,\,i_e\}$ l'ensemble de définition du code C. Considérons :

$$T_1 = \{0,\,n_2,\,2n_2,\,\ldots,\,(n_1-1)n_2\} \tag{2.18}$$

et définissons l'ensemble

$$S = \{i \pmod{n_1} : i \in I\}. \tag{2.19}$$

Suivant [101], on a :

Proposition 2.4 *Avec les notations précédentes, l'ensemble de définition du code $C(T_1)$ est $\{\beta^s \,|\, s \in S\}$, où $\beta = \alpha^{n_2}$ est une racine primitive n_1-ième de l'unité. Par conséquent, le code $C(T_1)$ est aussi un code BCH de paramètres $[n_1, n_1 - |S|]$.*

Remarque 2.9 Les preuves de ces propositions montrent qu'elles restent vraies pour tout code C cyclique. Le fait que les racines en I soient *consécutives* entraîne juste que le code $C(T_1)$ est aussi un code BCH.

La suite de la décomposition est facile : on partitionne $[0, n_1n_2 - 1]$ en n_2 ensembles disjoints $T_1, T_2, \ldots, T_{n_2}$ tels que pour tout $j \in \{1, 2, \ldots, n_2\}$ on a :

$$T_j = [(j-1) + T_1] \,(\mathrm{mod}\, n),$$

T_1 étant défini par la relation (2.18). La Proposition 2.3 donne $C(T_1) = C(T_2) = \ldots = C(T_{n_2})$ et via la Proposition 2.4 on conclut que la somme directe $C(T_1) \oplus C(T_2) \oplus \ldots \oplus C(T_{n_2})$ est un sous-code de C de dimension $n_2(n_1 - |S|) = n - n_2|S|$.

Le noyau de la décomposition est donc le calcul de $|S|$, donc le passage de I à S utilisant la relation (2.19). L'ensemble de définition du code C est une réunion de classes cyclotomiques, donc il faudra s'intéresser à la transformation de ces classes. Nous allons illustrer cela par un exemple pratique comme suit.

Exemple : Considérons le code BCH de longueur 75 et de distance construite 7. Son ensemble de définition est $I = C_1^{(75)} \cup C_3^{(75)} \cup C_5^{(75)}$ et sa dimension $75 - |I| = 75 - (20 + 20 + 4) = 31$. Le passage des classes cyclotomiques modulo 75 à des classes cyclotomiques modulo 15 se fait en respectant le schéma suivant :

$$\begin{array}{l} C_0^{(75)}, C_{15}^{(75)} \longrightarrow C_0^{(15)} \\ C_1^{(75)} \longrightarrow C_1^{(15)}, \quad C_3^{(75)} \longrightarrow C_3^{(15)} \\ C_5^{(75)}, C_{25}^{(75)}, C_{35}^{(75)} \longrightarrow C_5^{(15)} \\ C_7^{(75)} \longrightarrow C_7^{(15)}. \end{array}$$

On choisit les ensembles $T_1 = \{0, 5, 10, \ldots, 70\}$, $T_2 = \{1, 6, 11, \ldots, 71\}$, $T_3 = \{2, 7, 12, \ldots, 72\}$, $T_4 = \{3, 8, 13, \ldots, 73\}$, $T_5 = \{4, 9, 14, \ldots, 74\}$. Les codes $C(T_1), \ldots, C(T_5)$ sont tous égaux et d'ensemble de définition $S = C_1^{(15)} \cup C_3^{(15)} \cup C_5^{(15)}$. Ils ont par ailleurs la dimension $15 - |S| = 15 - (4 + 4 + 2) = 5$, leur somme directe étant donc un code $[75, 25]$.

D'autres décompositions sont présentées en [101] dans le cas des codes BCH primitifs étendus de longueur $n + 1 = 2^m$. Il s'agit d'isoler des sommes directes, sous-codes des codes EBCH primitifs, notamment via un résultat analogue à la proposition 2.3 (voir [101, Prop. 3]). Le problème à résoudre équivaut à :

Trouver des sous-ensembles $A_1, A_2, \ldots, A_h$ *de* $\mathbb{F}_{2^m}$ *tels que :*

- $A_1 \cup A_2 \cup \ldots \cup A_h = \mathbb{F}_{2^m}$;
- *pour tout* $i \neq j$ *on a* $A_i \cap A_j = \emptyset$;
- *pour tout* j_1, j_2 *et pour un* $\alpha^a \in \mathbb{F}_{2^m}$ *on a* $A_{j_1} = \alpha^a + A_{j_2}$.

Un façon de résoudre cela est de considérer $\mathbb{F}_{2^m}$ comme espace vectoriel et de le partitionner en un sous-ensemble et ses translatés. Vardy et Be'ery proposent d'énumérer toutes ces partitions à l'aide d'un ordinateur et de choisir celle qui donne la somme directe de dimension maximale. Cette dimension se calcule de la façon suivante : on définit pour chaque $j \in \{1, 2, \ldots, h\}$

$$T_j = \{i \,:\, \alpha^i \in A_j\}.$$

La somme directe $C(T_1) \oplus C(T_2) \oplus \ldots \oplus C(T_h)$ est un sous-code de C de dimension $hk_{C(T_1)}$. Grâce à cette "recherche exhaustive", en [101] on donne les décompositions seulement pour les

EBCH de longueur $n \leq 64$. Nous allons voir au Chapitre 4 *une généralisation de cette méthode* due à Berger et Be'ery [6] qui permet une décomposition facile des codes EBCH même de très grande longueur (voir aussi nos résultats numériques en Section A.2).

3

Treillis associé à un code en bloc

Dans ce chapitre nous continuons l'étude des formes décomposées des matrices génératrices par leurs treillis associés. Notamment on va appliquer l'étude théorique aux codes de Reed-Muller pour montrer que l'ordre standard des bits (favorable à la décomposition des matrices) donne une bonne complexité, contrairement à l'ordre cyclique qui donne la plus mauvaise complexité des états du treillis.

Le treillis d'un code est une représentation graphique d'un code (en bloc ou convolutif) dans laquelle chaque chemin représente un mot de code. Cette représentation permet un décodage à maximum de vraisemblance avec une complexité suffisamment réduite.

Nous présentons dans ce chapitre les concepts fondamentaux et les propriétés structurelles des treillis pour un *code en bloc*. Pour des preuves des résultats ou des compléments, se reporter à [74], [58], [68], [32], [33] ou [75]. Les livres de référence sont [64] et [100].

3.1 Treillis à arêtes indexées pour les codes binaires

Considérons un code linéaire binaire $C[n, k]$ de matrice génératrice G et de matrice de contrôle H. Pendant chaque intervalle d'encodage, un message de k bits d'information est shifté dans la mémoire de l'encodeur et codé par un mot de code de n bits, donc il suffit de n instants pour construire et shifter les n bits.

Si $\Gamma = \{0, 1, \ldots, n\}$ est *l'étendue* de l'encodage, C peut être représenté par un treillis n-sectionné T. Soit $\mathcal{E}(C)$ l'encodeur de C.

Définition 3.1 *Un* treillis n-sectionné *de C de profondeur n, noté T, est un graphe orienté qui consiste en $n+1$* niveaux *pour les sommets (ou* états*) et pour les* arêtes *tel que :*

(i) Pour $0 \leq i \leq n$, les sommets au niveau i représentent les états de l'espace S_i de l'encodeur au temps i. Au temps 0 (ou au niveau 0) il n'existe que le sommet initial *noté s_0, tandis qu'au temps n (ou au niveau n) il n'existe que* le sommet final *noté s_f.*

(ii) Pour $0 < i \leq n$, une arête dans la i-ème section du treillis T relie un état $s_{i-1} \in S_{i-1}$ à un état $s_i \in S_i$ et est indexée *par un bit u_i d'un mot de code qui représente la sortie de l'encodeur $\mathcal{E}(C)$ dans l'intervalle $[i, i+1]$. Une arête représente* une transition *des états.*

(iii) *Sauf s_0, chaque état a* au moins une et au plus deux *arêtes d'arrivée et sauf s_f, chaque état a* au moins une et au plus deux *arêtes de départ. L'état initial n'a pas d'arêtes d'arrivée et l'état final n'a pas d'arêtes de départ.*
Deux arêtes qui partent d'un même sommet ont des indices différents.

(iv) *Il existe* un chemin direct *qui relie s_0 et s_f par une suite $(u_0, u_1, \ldots, u_n)$ si et seulement si $(u_0, u_1, \ldots, u_n)$ est un mot de code de C.*

Deux états du treillis sont *adjacents* s'il existe une arête qui les relie. Pendant l'intervalle d'encodage Γ, l'encodeur part de l'état initial s_0, passe par la suite des états :

$$(s_0, s_1, \ldots, s_i, \ldots, s_f),$$

génère le mot de code :

$$(u_0, u_1, \ldots, u_i, \ldots, u_n)$$

et arrive sur le sommet s_f.

Un treillis 8-sectionné du code de Reed-Muller[8, 4, 4] —voir sa matrice génératrice en (1.6)— est présenté à la figure 3.1.

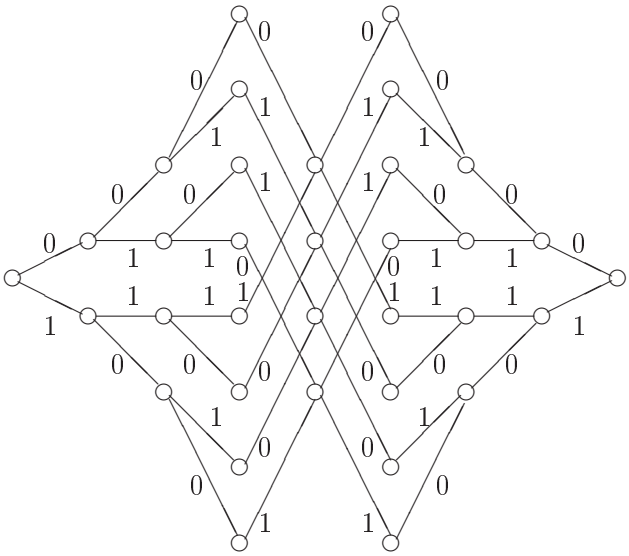

FIG. 3.1 – Un treillis associé au code $RM(1,3)$

La suite :

$$(|S_0|, |S_1|, \ldots, |S_n|)$$

est appelée *le profil de la complexité des espaces* des états et est une mesure de la complexité d'un treillis n-sectionné.

On démontrera plus loin que $|S_i|$ est une puissance de 2, pour tout $0 \leq i \leq n$. Si on note :

$$\rho_i = \log_2 |S_i|$$

la dimension de l'espace des états à l'instant i, la suite :

$$(\rho_0, \rho_1, \ldots, \rho_n)$$

est appelée *le profil des dimensions* des espaces des états.

La figure 3.1 nous donne le profil des états et celui des dimensions pour le code $RM(1, 3)$: (1, 2, 4, 8, 4, 8, 4, 2, 1) et (0, 1, 2, 3, 2, 3, 2, 1, 0), respectivement.

3.1.1 Matrice adaptée à la structure du treillis

Pour faciliter la construction du treillis associé à un code, on donne à sa matrice génératrice une forme spéciale.

Définition 3.2 *Soit* $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ *un n-uplet non-nul. La première composante non-nulle de* $\mathbf{u}$ *est nommée* la tête *et la dernière est nommée* la queue.

Une matrice génératrice G *de* C *est dite* matrice adaptée à la structure du treillis ou TOGM—*"trellis oriented generator matrix"—si :*

1. *les colonnes qui contiennent les têtes des lignes sont en ordre croissant ;*
2. *il n'existe pas de lignes ayant leurs queues sur la même colonne.*

Théorème 3.1 *(Kschischang et Sorokine [58]) Toute matrice génératrice peut être transformée en TOGM en appliquant deux fois* le pivot de Gauss.

Exemple 3.1 Considérons pour le code de RM(1,3) [8, 4, 4] la matrice génératrice suivante :

$$G_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

qui n'est pas TOGM car les têtes des lignes sont sur les colonnes 1, 5, 3, 2 qui ne sont pas en ordre croissant (première condition de la définition) et de plus toutes les lignes finissent sur la même colonne. Pour que la première condition soit satisfaite, il suffit de permuter les lignes2 et 4 pour obtenir :

$$G_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

On obtient une TOGM en additionnant la ligne 4 aux lignes 1, 2 et 3 :

$$G = \begin{bmatrix} \mathbf{g_1} \\ \mathbf{g_2} \\ \mathbf{g_3} \\ \mathbf{g_4} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

car les queues des lignes sont sur des colonnes distinctes.

Définition 3.3 *Soit une TOGM G et soit $\mathbf{g} = (g_1, g_2, \ldots, g_n)$ une ligne de G.* L'étendue *de* $\mathbf{g}$ *est le* plus petit intervalle $\{i, i+1, \ldots, j\}$ *qui contient tous les bits non-nuls de* $\mathbf{g}$. *Elle est notée* $span(\mathbf{g}) = [i, j]$.

Si $\mathbf{g}$ *est d'étendue* $[i, j]$, l'étendue active *de* $\mathbf{g}$ *est* $aspan(\mathbf{g}) = [i, j-1]$ *pour* $i < j$ *et* $aspan(\mathbf{g}) = \emptyset$ *si* $i = j$.

Soient $\mathbf{g_1}, \mathbf{g_2}, \ldots, \mathbf{g_k}$ les lignes de G et $\mathbf{a} = (a_1, a_2, \ldots, a_k)$ un bloc de k bits d'information qui doit être encodé. Le mot de code correspondant sera :

$$\mathbf{u} = (u_1, u_2, \ldots, u_n) = a_1 \cdot \mathbf{g}_1 + a_2 \cdot \mathbf{g}_2 + \ldots + a_k \cdot \mathbf{g}_k.$$

Remarque 3.1 On voit facilement que le ℓ-ième bit d'information a_ℓ affecte la sortie $\mathbf{u}$ de l'encodeur $\mathcal{E}(C)$ sur l'étendue de la ℓ-ième ligne $\mathbf{g}_\ell$ de G.

Par conséquent, à l'instant i, le nombre des bits d'information qui affectent le bit de sortie u_{i+1} est égal au nombre de lignes $\mathbf{g}$ de G telles que $i \in \text{aspan}(\mathbf{g})$. *Ces bits d'information définissent l'état de l'encodeur au temps* i.

3.1.2 Suppléments sur l'espace des états

Dans cette section nous donnerons une formulation mathématique pour l'espace des états d'un treillis n-sectionné d'un code en bloc binaire $C[n, k]$ de matrice génératrice G.

Définition 3.4 *Pour tout instant* i, $0 \leq i \leq n$, *on partitionne les lignes de* G *en trois sous-ensembles* disjoints *:*

1. G_i^p *qui contient les lignes de* G *dont les étendues sont contenues dans l'intervalle* $[1, i]$ *;*
2. G_i^f *qui contient les lignes de* G *dont les étendues sont contenues dans l'intervalle* $[i+1, n]$ *;*
3. G_i^s *qui contient les lignes de* G *dont les étendues* actives *contiennent* i.

On note A_i^p, A_i^f, A_i^s *les sous-ensembles des bits d'information qui correspondent aux lignes de* G_i^p, G_i^f, G_i^s *resp.*

Conséquence importante : Entre les tailles des trois ensembles définis précédemment existe la relation :

$$|G_i^p| + |G_i^s| + |G_i^f| = k. \tag{3.1}$$

Les bits d'information de A_i^p n'affectent pas les sorties de l'encodeur après l'instant i (donc ils constituent "le passé" par rapport à l'instant i) et les bits d'information de A_i^f n'affectent les sorties de l'encodeur qu'après l'instant i (donc ils sont "le futur" par rapport à l'instant i).

Les bits d'information de A_i^s affectent les sorties de l'encodeur avant et après l'instant i (on dit qu'ils *définissent* l'état de l'encodeur $\mathcal{E}(C)$ à l'instant i).

Proposition 3.1 *Pour chaque* i, $0 \leq i \leq n$, $|S_i|$ *est une puissance de* 2.

Preuve. Soit $\rho_i = |A_i^s| = |G_i^s|$. Alors il existe 2^{ρ_i} états possibles, distincts pour l'instant i, chaque état étant défini par une combinaison des ρ_i bits d'information de A_i^s. Ces états forment l'espace des états S_i et le paramètre ρ_i est *la dimension* de cet espace. Dans le treillis de C, ces états sont représentés par 2^{ρ_i} sommets au niveau i du treillis. ◇

Exemple 3.2 Considérons la TOGM G obtenue en Exemple 3.1.

Les étendues des lignes de G sont respectivement : span($\mathbf{g}_1$) = [1, 4], span($\mathbf{g}_2$) = [2, 7], span($\mathbf{g}_3$) = [3, 6] et span($\mathbf{g}_4$) = [5, 8].

Leurs étendues actives sont aspan($\mathbf{g}_1$) = [1, 3], aspan($\mathbf{g}_2$) = [2, 6], aspan($\mathbf{g}_3$) = [3, 5] et aspan($\mathbf{g}_4$) = [5, 7].

Pour chaque $0 \leq i \leq 8$, en comptant le nombre de lignes actives à l'instant i, on obtient le profil des dimensions des espaces des états : (0, 1, 2, 3, 2, 3, 2, 1, 0).

3.1.3 Transition des états et sorties. Construction du treillis

Pour $0 \leq i < n$ supposons que l'encodeur $\mathcal{E}(C)$ est dans l'état $s_i \in S_i$. En passant de l'instant i à l'instant $i+1$, l'encodeur génère un bit u_{i+1} et passe de l'état s_i à l'état $s_{i+1} \in S_{i+1}$. Soit

$$G_i^s = \{\mathbf{g}_1^{(i)}, \mathbf{g}_2^{(i)}, \ldots, \mathbf{g}_{\rho_i}^{(i)}\} \tag{3.2}$$

et

$$A_i^s = \{a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}\}, \tag{3.3}$$

où $\rho_i = |G_i^s|$.

Soit $\mathbf{g}^*$ la ligne de G_i^f dont la tête est au niveau $i+1$ (*si elle existe* elle est unique, à cause de la condition 1 de la définition d'une TOGM). Alors sa $(i+1)$-ème composante est $g_{i+1}^* = 1$. Soit a^* le bit d'information qui correspond à la ligne $\mathbf{g}^*$.

Le bit de sortie pendant l'intervalle de temps $[i, i+1]$ est :

$$u_{i+1} = a^* + \sum_{\ell=1}^{\rho_i} a_\ell^{(i)}\, g_{\ell,\,i+1}^{(i)}, \tag{3.4}$$

où $g_{\ell,\,i+1}^{(i)}$ est la $(i+1)$-ème composante de $\mathbf{g}_\ell^{(i)}$ dans G_i^s. Comme a^* affecte le bit de sortie à l'instant $i+1$, il est appelé *le bit d'information d'entrée courant* Le deuxième terme est la contribution de l'état s_i définie comme une combinaison des bits d'information de A_i^s.

Le bit de sortie peut avoir deux valeurs différentes qui dépendent de celle de a^* et qui correspondent à *deux arêtes* qui partent de s_i, indexées 0 et 1 respectivement.

S'il n'existe pas de ligne $\mathbf{g}^*$ dans G_i^f, alors le bit de sortie est donné par :

$$u_{i+1} = \sum_{\ell=1}^{\rho_i} a_\ell^{(i)}\, g_{\ell,\,i+1}^{(i)}. \tag{3.5}$$

Dans ce cas nous pouvons considérer $a^* = 0$ et le bit de sortie ne peut avoir qu'une seule valeur, donc il existe *une seule arête* qui part de s_i vers un sommet de S_{i+1}.

Exemple 3.3 Pour la même TOGM G obtenue en Exemple 3.1, considérons l'instant $\mathbf{i = 2}$. On trouve $G_2^p = \emptyset$, $G_2^f = \{\mathbf{g_3}, \mathbf{g_4}\}$ et $G_2^s = \{\mathbf{g_1}, \mathbf{g_2}\}$. Alors a_1 et a_2 définissent l'état de l'encodeur à l'instant 2 et il existe 4 états distincts $\{00, 01, 10, 11\}$.

On remarque aussi que $\mathbf{g}^* = \mathbf{g_3}$, donc le bit courant d'entrée est $a^* = a_3$ et le bit de sortie est donné par :

$$u_3 = a_3 + a_1 \cdot g_{1,3} + a_2 \cdot g_{2,3} = a_3 + a_1.$$

Pour chaque état défini par a_1 et a_2 il existe deux valeurs distinctes de u_3, donc deux arêtes qui divergent, comme nous l'avons déjà vu dans la figure 3.1.

Considérons l'instant $\mathbf{i = 3}$. On obtient $G_3^p = \emptyset$, $G_3^f = \{\mathbf{g}_4\}$ et $G_3^s = \{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$. Alors a_1, a_2 et a_3 définissent l'état de l'encodeur à l'instant 3 et il existe 8 états distincts comme dans la figure 3.1.

Il n'y a pas de $\mathbf{g}^* \in G_3^f$ dont la tête soit au niveau 4, donc $a^* = 0$ et :

$$u_4 = a_1 \cdot g_{1,4} + a_2 \cdot g_{2,4} + a_3 \cdot g_{3,4} = a_1 + a_2 + a_3.$$

Ainsi dans le treillis il existe une seule arête qui part de chacun des 8 sommets, comme nous l'avons déjà vu dans la figure 3.1.

Soit $\mathbf{g^0}$ la ligne de G_i^s dont la queue est au niveau $i+1$ (*si elle existe* elle est unique, à cause de la condition 2 de la définition d'une TOGM). Soit a^0 le bit d'information qui correspond à la ligne $\mathbf{g^0}$. À l'instant $i+1$ on a :

$$G_{i+1}^s = (G_i^s \setminus \{\mathbf{g}^0\}) \cup \{\mathbf{g}^*\} \tag{3.6}$$

et

$$A_{i+1}^s = (A_i^s \setminus \{a^0\}) \cup \{a^*\}. \tag{3.7}$$

À partir de A_i^s, A_{i+1}^s, (3.4) et (3.5) on peut construire le treillis n-sectionné T.

Construction du treillis n-sectionné du code—section par section.
On suppose le treillis construit jusqu'à la i-ème section.

Étapes pour construire la $(i+1)$-ème section :

1. déterminer G_{i+1}^s et A_{i+1}^s par (3.6) et (3.7) et construire S_{i+1} ;
2. pour chaque $s_i \in S_i$, déterminer l(es) état(s) de transition en utilisant les règles données précédemment et relier s_i et ses états adjacents de S_{i+1} par des arêtes ;
3. indicer les arêtes par les bits de sortie u_{i+1} correspondants calculés en utilisant (3.4) ou (3.5).

3.1.4 Propriétés structurelles

Dans ce paragraphe nous allons présenter quelques propriétés structurelles du treillis associé à un code. Pour cela plusieurs définitions et notations spécifiques à ce but seront introduites. Pour plus de détails voir [64, Sec. 3.5].

Pour $0 \leq i < j \leq n$, soit :

1. $C_{i,j}$ le sous-code de C qui contient tous les mots de code dont les coordonnés non-nulles sont incluses dans l'ensemble $\{i+1, i+2, \ldots, j\}$. Alors chaque mot de code de $C_{i,j}$ est de la forme :
$$(0, 0, \ldots, 0, u_{i+1}, u_{i+2}, \ldots, u_j, 0, 0, \ldots, 0) ;$$
Pour un i fixé, les codes $C_{0,i}$ et $C_{i,n}$ sont appelés le code *préfixe* et le code *suffixe* de C par rapport à l'instant i.
2. $p_{i,j}(C)$ le code linéaire de longueur $j-i$ obtenu en enlevant les i premières et les $n-j$ dernières composantes de chaque mot de code de$C$ (voir la définition des codes poinçonnés). Cette notation sera employée pour des codes ou tout simplement pour des mots de code ;
3. $C^{\mathrm{tr}}_{i,j}$ le code poinçonné (ou tronçonné) par rapport à $[i,j]$ du sous-code $C_{i,j}$, i.e. :
$$C^{\mathrm{tr}}_{i,j} = p_{i,j}(C_{i,j}). \tag{3.8}$$

Le résultat suivant est très utilisé dans toute la théorie des treillis.

Théorème 3.2 *L'espace S_i des états au niveau i dans un treillis n-sectionné (construit comme dans le paragraphe précédent, via une matrice TOGM) est isomorphe à :*

$$C/(C_{0,i} \oplus C_{i,n}), \quad p_{0,i}(C)/C^{tr}_{0,i}, \quad \text{ou} \quad p_{i,n}(C)/C^{tr}_{i,n}. \tag{3.9}$$

Preuve. Nous allons détailler la preuve juste pour le premier isomorphisme ; pour les autres voir [64, Sec. 3.5].

La définition de $C_{i,j}$ et la structure TOGM de G impliquent que $C_{i,j}$ est engendré par les lignes de G dont les étendues sont contenues dans l'intervalle $[i+1, j]$. Par conséquent, les sous-codes $C_{0,i}$ et $C_{i,n}$ sont engendrés par les lignes de G^p_i et G^f_i, respectivement. Notons $k_{0,i}$ et $k_{i,n}$ leurs dimensions. Via la relation (3.1), la dimension de S_i est :

$$\rho_i = |G^s_i| = k - |G^p_i| - |G^f_i| = k - k_{0,i} - k_{i,n}. \tag{3.10}$$

La somme directe $C_{0,i} \oplus C_{i,n}$, est un sous-code de C de dimension :

$$k_{0,i} + k_{i,n},$$

car $C_{0,i} \cap C_{i,n} = \{\mathbf{0}\}$. La partition $C/(C_{0,i} \oplus C_{i,n})$ contient :

$$|C/(C_{0,i} \oplus C_{i,n})| = 2^{k-k(C_{0,i})-k(C_{i,n})} = 2^{\rho_i} \tag{3.11}$$

translatés de $C_{0,i} \oplus C_{i,n}$. L'équation (3.11) signifie que le nombre des états dans S_i est égal au nombre des translatés dans la partition $C/(C_{0,i} \oplus C_{i,n})$.

Notons Σ_i le sous-espace de C qui est engendré par les lignes de G_i^s. Alors chaque mot de code de Σ_i est donné par :

$$\mathbf{v} = (a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)}) \cdot G_i^s = a_1^{(i)} \cdot \mathbf{g}_1^{(i)} + a_2^{(i)} \cdot \mathbf{g}_2^{(i)} + \ldots + a_{\rho_i}^{(i)} \cdot \mathbf{g}_{\rho_i}^{(i)} \tag{3.12}$$

où $a_\ell^{(i)} \in A_i^s$, $\forall\, \ell \in [1,\, \rho_i]$. Les 2^{ρ_i} mots de code de Σ_i peuvent être utilisés comme *représentants* des éléments de la partition $C/(C_{0,\,i} \oplus C_{i,\,n})$, donc Σ_i est l'espace de ces représentants.

La relation (3.12) implique une *correspondance bijective* entre $\mathbf{v}$ et l'état $s_i \in S_i$ défini par $(a_1^{(i)}, a_2^{(i)}, \ldots, a_{\rho_i}^{(i)})$. Comme il y a une *correspondance bijective* entre $\mathbf{v}$ et un élément de $C/(C_{0,\,i} \oplus C_{i,\,n})$, on en déduit une *correspondance bijective* entre un état de S_i et un élément de $C/(C_{0,\,i} \oplus C_{i,\,n})$. ◇

Nous finirons ce paragraphe en rappelant le résultat suivant :

Proposition 3.2 *[64, Sec. 3.5] Pour $0 \leq i < j \leq n$, $L(s_i,\, s_j)$ désigne les chemins du treillis T qui relient un sommet s_i au niveau i avec un sommet s_j au niveau j du treillis. Alors $L(s_i,\, s_j)$ est un translaté dans la partition $p_{i,\,j}(C)/C_{i,\,j}^{tr}$. Le nombre des chemins qui relient s_i et s_j est alors :*

$$|L(s_i,\, s_j)| = \begin{cases} 2^{k_{i,\,j}} & \textit{si on peut relier } s_i \textit{ et } s_j, \\ 0 & \textit{sinon,} \end{cases} \tag{3.13}$$

car les codes $C_{i,\,j}^{tr}$ et $C_{i,\,j}$ sont tous les deux de dimension $k_{i,\,j}$.

3.2 Complexité du treillis

Dans cette section nous présentons une analyse de la complexité d'un treillis n-sectionné associé à un code en bloc linéaire [64]. On étudiera plutôt la complexité des états et celle des arêtes, qui interviennent dans les algorithmes de décodage, par exemple celui de Viterbi. *La complexité des états* est une mesure du profil des dimensions des espaces des états, tandis que *la complexité des arêtes* est déterminée par le nombre des arêtes dans le treillis.

Premièrement on déduit une majoration simple de la dimension des espaces des états et on rappelle que la complexité des états est la même pour le code et pour son dual. Par la suite, on introduit les concepts de *treillis minimal* et de *permutation optimale des bits* dans le sens de la complexité des états. On rappelle aussi que l'utilisation d'une TOGM conduit à un treillis minimal. Finalement nous étudions la complexité des arêtes.

Complexité des états

Soit $C[n,\, k]$ un code en bloc linéaire binaire et considérons le treillis n-sectionné associé. Comme nous l'avons déjà vu dans le chapitre précédent, la complexité des états est mesurée par le profil des dimensions des états :

$$(\rho_0,\, \rho_1,\, \rho_2, \ldots,\, \rho_n),$$

où, pour $0 \leq i \leq n$, on a (voir (3.10)) :

$$\rho_i = \log_2 |S_i| = k - k_{0,\,i} - k_{i,\,n}.$$

Notons $\rho_{\max}$ le maximum des dimensions des états, c'est-à-dire :

$$\rho_{\max} = \max_{0 \leq i \leq n} \rho_i.$$

Remarque 3.2 En utilisant l'algorithme de Viterbi, le nombre maximal des survivants et le nombre maximal de chemins qu'on doit stocker sont tous les deux égaux à$2^{\rho_{\max}}$. $\rho_{\max}$ est alors une mesure clef pour la complexité du décodage (ou la complexité du treillis).

Théorème 3.3 *On a la borne supérieure suivante :*

$$\rho_{max} \leq \min\{k,\, n-k\}. \tag{3.14}$$

Remarque 3.3 La borne (3.14) a été prouvée par Wolf [104], mais elle n'est pas très précise. Pourtant pour les codes cycliques cette borne donne la complexité exacte des états (voir Section 3.3).

Une borne plus précise est la suivante :

Théorème 3.4 *On a la borne supérieure suivante pour* ρ_i *(*$0 \leq i \leq n$*) :*

$$\rho_i \leq \min\{i,\, k,\, n-k,\, n-i\}. \tag{3.15}$$

Le résultat suivant est très important, car il permet de choisir l'étude d'un code ou de son dual, tout en obtenant la même complexité.

Théorème 3.5 *[64, Section 5.1] Le code* C *et son dual* $C^{\perp}$ *ont* la même complexité des états.

3.2.1 Treillis minimal

Un treillis n-sectionné est dit *minimal* si le nombre total des états est minimal. Un treillis minimal est *unique* à un isomorphisme près [74]. Même si cette définition est la définition usuelle, une autre, plus précise et plus utile est basée sur le profil des dimensions des espaces des états.

Un treillis n-sectionné est *minimal par rapport aux dimensions* des espaces des états si les dimensions sont minimales à chaque niveau. Une définition plus précise est la suivante :

Définition 3.5 *Soit* T *un treillis* n*-sectionné pour un code* $C[n,\, k]$ *et soit* $(\rho_0,\, \rho_1,\, \rho_2, \ldots,\, \rho_n)$ *son profil des dimensions.* T *est dit* minimal *si pour tout autre treillis* n*-sectionné* T' *qui a le profil des dimensions* $(\rho'_0,\, \rho'_1,\, \rho'_2, \ldots,\, \rho'_n)$*, on a :*

$$\forall\, i,\;\; 0 \leq i \leq n \;\; : \quad \rho_i \leq \rho'_i.$$

Supposons qu'il existe un treillis minimal dans le sens de la définition 3.5. Il est évident qu'il est également minimal par rapport au nombre total des états. *L'existence* d'un tel treillis pour chaque code en bloc linéaire est garantie par le théorème suivant :

Théorème 3.6 *Soit $C[n, k]$ un code en bloc linéaire et G sa TOGM. Le treillis n-sectionné T construit en utilisant G est minimal par rapport au profil des dimensions.*

L'unicité d'un treillis minimal par rapport au nombre total des états (à un isomorphisme près) implique *l'unicité* du treillis minimal par rapport au profil des dimensions. Cela donne *l'équivalence* entre les deux définitions. Le théorème 3.6 nous assure que l'équation (3.10) donne la dimension minimale ρ_i, $0 \leq i \leq n$, pour un treillis n-sectionné associé à un code. On remarque aussi que cette dimension dépend des dimensions des codes préfixe et suffixe et celles-là sont fixées pour un code.

Par contre, bien qu'une permutation sur les positions des bits ne change pas la distribution de poids, elle change les dimensions des codes préfixe et suffixe à l'instant i. Par conséquent *les valeurs de ρ_i ne sont pas invariantes*.

Une permutation qui induit des dimensions minimales à chaque instant est dite *optimale*. Il est clair qu'une telle permutation réduit la complexité des états, mais en général en trouver une est *un problème difficile* (cependant on les connaît pour les codes RM [56]).

3.2.2 Complexité des arêtes

La complexité des arêtes pour un treillis n-sectionné est définie comme étant le nombre total des arêtes du treillis. Cette complexité détermine le nombre des additions demandées par un algorithme de décodage.

En utilisant les résultats vus dans le paragraphe précédent, à chaque instant i, $0 \leq i < n$, il existe deux arêtes qui partent d'un sommet de S_i s'il existe une ligne $\mathbf{g}^*$ dans G_i^f et sinon il n'y a qu'une seule arête. Posons :

$$I_i(\mathbf{g}^*) = \begin{cases} 1 & \text{si } \mathbf{g}^* \notin G_i^f, \\ 2 & \text{si } \mathbf{g}^* \in G_i^f. \end{cases} \tag{3.16}$$

Soit E le nombre total des arêtes dans T. Alors :

$$E = \sum_{i=0}^{n-1} |S_i| \cdot I_i(\mathbf{g}^*) = \sum_{i=0}^{n-1} 2^{\rho_i} \cdot I_i(\mathbf{g}^*). \tag{3.17}$$

Exemple 3.4 Si on considère encore le code $RM(1, 3)$ vu en Exemple 3.1, on trouve :

$$I_0(\mathbf{g}^*) = I_1(\mathbf{g}^*) = I_2(\mathbf{g}^*) = I_4(\mathbf{g}^*) = 2$$

et

$$I_3(\mathbf{g}^*) = I_5(\mathbf{g}^*) = I_6(\mathbf{g}^*) = I_7(\mathbf{g}^*) = 1.$$

Le profil des dimensions étant (0, 1, 2, 3, 2, 3, 2, 1, 0), on utilise (3.17) et on obtient :

$$\begin{aligned} E &= 2^0 \cdot 2 + 2^1 \cdot 2 + 2^2 \cdot 2 + 2^3 \cdot 1 + 2^2 \cdot 2 + 2^3 \cdot 1 + 2^2 \cdot 1 + 2^1 \cdot 1 \\ &= 2 + 4 + 8 + 8 + 8 + 8 + 4 + 2 = 44. \end{aligned}$$

Un treillis n-sectionné est dit *minimal par rapport au nombre des arêtes* s'il a une complexité des arêtes minimale, ce qui est le cas pour un treillis minimal [68]. On peut chercher aussi une permutation sur l'ordre des bits qui minimise cette complexité. Il suffit donc de minimiser chaque produit dans (3.17), par exemple si on a $I_i(\mathbf{g}^*) = 1$ pour tout $0 \leq i < n$ tel que ρ_i est grand.

3.3 Première application : treillis associé à un code cyclique

Considérons un code cyclique binaire $C[n, k]$ de polynôme générateur :

$$g(X) = 1 + g_1 X + g_2 X^2 + \ldots + g_{n-k-1} X^{n-k-1} + X^{n-k},$$

où pour $1 \leq i \leq n-k$, $g_i \in \mathbb{F}_2$. Une matrice génératrice de ce code est :

$$G = \begin{bmatrix} 1 & g_1 & g_2 & \ldots & \ldots & g_{n-k-1} & 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & g_1 & g_2 & \ldots & \ldots & g_{n-k-1} & 1 & 0 & \ldots & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & 0 & \ldots & 0 & 1 & g_1 & g_2 & \ldots & \ldots & g_{n-k-1} & 1 \end{bmatrix}.$$

Cette matrice génératrice a les propriétés suivantes :

1. elle est TOGM (les têtes des lignes sont sur les colonnes $1, 2, \ldots, k$ qui sont en ordre croissant et les queues sont placées sur des colonnes distinctes) ;
2. pour $1 \leq i \leq k$, l'étendue de la ligne $\mathbf{g}_i$ est donnée par :

$$\text{span}(\mathbf{g}_i) = [i, n-k+i] \ ;$$

3. la taille de l'étendue active de chaque ligne est constante : $n-k$.

Considérons le treillis à arêtes indexées de C et considérons premièrement *le cas* $\mathbf{k > n-k}$. Pour $0 \leq i \leq n-k$, le nombre des lignes telles que $i \in \text{aspan}(\mathbf{g})$ est i et ces lignes sont exactement les i premières. Pour $n-k < i \leq k$, ce nombre est $n-k$, et pour $k < i \leq n$, ce nombre est $n-i$. Quand $i > k$, on a $n-i < n-k$, donc $\rho_{\max} = n-k$ et le profil des dimensions est :

$$(0, 1, \ldots, n-k-1, n-k, \ldots, n-k, n-k-1, \ldots, 1, 0).$$

Considérons *le deuxième cas* $\mathbf{k \leq n-k}$. Pour $0 \leq i \leq k$, le nombre des lignes telles que $i \in \text{aspan}(\mathbf{g})$ est i (les i premières lignes). Pour $k \leq i \leq n-k$, ce nombre est k, et pour $n-k < i \leq n$, ce nombre est $n-i$. Quand $i > n-k$, on a $n-i < k$, donc $\rho_{\max} = k$ et le profil des dimensions est :

$$(0, 1, \ldots, k-1, k, \ldots, k, k-1, \ldots, 1, 0).$$

Les deux cas analysés ci-dessus impliquent :

$$\rho_{\max} = \min\{k, n-k\}.$$

Cela veut dire qu'un code cyclique a *la plus mauvaise complexité des états* (elle atteint la borne supérieure (3.14) vue dans la Section 3.2).

Le treillis associé à un code cyclique est *symétrique*, c'est-à-dire que la partie gauche et celle de droite (par rapport au centre) sont identiques du point de vue structurel. *Pour réduire la complexité des états, il est nécessaire d'effectuer une permutation optimale des bits des positions [56], [57].*

Comme G est une TOGM, on déduit facilement que :

$$I_i(\mathbf{g}^*) = \begin{cases} 2 & \text{si } 0 \leq i < k, \\ 1 & \text{sinon.} \end{cases}$$

Via la relation (3.17), la complexité des arêtes d'un code cylique est :

$$E = \begin{cases} (n - 2k + 4) \cdot 2^k - 4 & \text{si } k < n - k, \\ (4k - 2n + 4) \cdot 2^{n-k} - 4 & \text{si } k \geq n - k. \end{cases}$$

3.4 Treillis sectionné

Nous introduisons dans cette section la notion de treillis sectionné, qui sera très utile pour simplifier la structure du treillis et obtenir des propriétés supplémentaires. Nous allons comprendre cela dès que, après avoir défini le treillis sectionné dans le cas général, nous passerons au code $RM(1, 3)$, qui nous suit à travers ce chapitre. Par la suite, nous présenterons une méthode de construction à partir d'une TOGM ; les états seront *indexés* en utilisant *une matrice de contrôle*. En application, nous détaillons la construction du treillis sectionné à états indexés du même $RM(1, 3)$.

3.4.1 Sections dans un treillis

Soit $C[n, k]$ un code en bloc linéaire et Γ son intervalle d'encodage. Pour un entier $0 < \ell \leq n$, considérons :

$$U = \{h_0, h_1, \ldots, h_\ell\} \subset \Gamma, \tag{3.18}$$

où $0 = h_0 < h_1 < h_2 < \ldots < h_\ell = n$.

Définition 3.6 *Un* treillis ℓ-sectionné *associé à C avec les bornes de la section dans U, noté $T(U)$, est obtenu à partir d'un treillis n-sectionné T de la manière suivante :*

1. *on enlève tous les sommets dans $S_h, h \notin U$, et toutes les arêtes qui partent ou qui arrivent sur ces sommets ;*
2. *pour $1 \leq j \leq \ell$, on relie un état $s \in S_{h_{j-1}}$ avec un état $s' \in S_{h_j}$ par une arête d'indice α si et seulement s'il existe un chemin d'indice α (obtenu en concatenant les indices des arêtes du chemin) qui relie s et s' dans T.*

Remarque 3.4 Dans $T(U)$, une arête qui relie $s \in S_{h_{j-1}}$ avec $s' \in S_{h_j}$ représente $h_j - h_{j-1}$ bits d'un mot de code.

Un *sous-graphe* du treillis est appelé *sous-treillis*. Un sous-treillis de $T(U)$ qui contient $S_{h_{j-1}}$, S_{h_j} et toutes les arêtes qui relient des sommets de $S_{h_{j-1}}$ avec des sommets de S_{h_j} est appelé *j-section* de $T(U)$. La longueur d'une j-section est $h_j - h_{j-1}$.

Si les longueurs de toutes les sections d'un treillis ℓ-sectionné $T(U)$ sont égales, alors $T(U)$ est dit *uniformément sectionné*. Si $\ell < n$, deux états adjacents peuvent être reliés par des arêtes multiples (nommées *arêtes parallèles*) ayant des indices différents.

Comme en Section 3.1 (voir relation (3.10)), nous définissons :

$$\rho_{h_j} = \log_2 |S_{h_j}| = k - k_{0,\,h_j} - k_{h_j,\,n} \tag{3.19}$$

la dimension de S_{h_j}, et

$$(\rho_0,\, \rho_{h_1}, \ldots,\, \rho_{h_{\ell-1}},\, \rho_n)$$

le profil des dimensions d'un treillis ℓ-sectionné $T(U)$, $U = \{0,\, h_1, \ldots,\, h_{\ell-1},\, n\}$.

Si on choisit les instants de U pour lesquels les dimensions des espaces des états sont petites, alors $T(U)$ aura un profil des dimensions petit. La dimension maximale est définie de manière évidente :

$$\rho_{\ell,\,\max} = \max_{0 \le j \le \ell} \rho_{h_j}. \tag{3.20}$$

Exemple 3.5 Considérons encore le RM-code [8, 4, 4] et son treillis 8-sectionné comme dans la figure 3.1. Supposons qu'on choisit $\ell = 4$ et $U = \{0,\, 2,\, 4,\, 6,\, 8\}$. En utilisant la définition 3.6, nous trouvons le treillis uniformément 4-sectionné présenté dans la figure 3.2, dans lequel chaque arête est indexée par deux bits.

Le profil des dimensions est (0, 2, 2, 2, 0) et $\rho_{4,\,\max} = 2$. C'est un treillis 4-sectionné à 14-états. Il est facile de remarquer que la partie droite du treillis (les sections 3 et 4) est *l'image par un miroir* de la partie gauche (les sections 1 et 2), ce qui permet un décodage *bidirectionnel*. On remarque aussi deux sous-treillis *parallèles* et *structurellement* identiques (*isomorphes*), sans connexions entre eux. La structure parallèle nous permet d'utiliser deux décodeurs de Viterbi identiques à 2 états qui travaillent en parallèle.

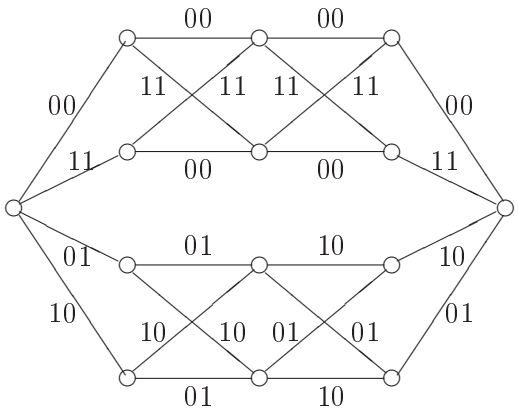

FIG. 3.2 – Un treillis 4-sectionné du code $RM(1, 3)$

3.4.2 Complexité des arêtes et connexion des états

Considérons la j-ème section du treillis minimal ℓ-sectionné $T(U)$, $U = \{h_0,\, h_1, \ldots,\, h_\ell\}$, associé à un code linéaire $C[n,\, k]$. Ce treillis minimal est obtenu à partir d'une TOGM du code. Chaque arête de cette section est indexée par $h_j - h_{j-1}$ bits.

Définition 3.7 *Soit $s \in S_{j-1}$ et $s' \in S_j$ deux états adjacents et $L(s, s')$ l'ensemble des arêtes parallèles qui les relient. Parfois il est plus pratique de regarder ces arêtes parallèles comme une seule, qui sera appelée* arête composée*, tandis que $L(s, s')$ sera appelé* indice d'une arête composée*.*

La complexité des arêtes pour une section dans un treillis est mesurée par :

1. la complexité d'une arête composée ;
2. le nombre des arêtes composées *distinctes* dans la section ;
3. le nombre total des arêtes composées dans la section.

Ces trois paramètres peuvent être exprimés en fonction des dimensions deC_{h_{j-1},h_j}, $C_{0,h_{j-1}}$, $C_{h_j,n}$ et $p_{h_{j-1},h_j}(C)$ (voir définitions Section 3.1.4) ; on peut donc les obtenir directement à partir d'une matrice adaptée à la structure du treillis du code.

La complexité des arêtes du treillis est la somme des complexités des arêtes de toutes les sections. Pour une analyse plus détaillée voir [64, Sec. 6.2].

3.4.3 Méthode de construction

Un treillis minimal ℓ-sectionné pour un code en bloc $C[n, k]$ peut être construit directement à partir d'une TOGM G. Soit $U = \{h_0, h_1, \ldots, h_\ell\}$ l'ensemble des bornes de la section, où $0 = h_0 < h_1 < \ldots < h_{\ell-1} < h_\ell = n$.

Supposons que $T(U)$ est construit jusqu'à la j-ème section, $1 \leq j < \ell$. On se propose de construire la $(j+1)$-ème section. On divise les lignes de G en trois ensembles disjoints $G^p_{h_j}$, $G^f_{h_j}$ et $G^s_{h_j}$:

1. $G^p_{h_j}$ contient les lignes de G dont les étendues sont incluses dans l'intervalle $[1, h_j]$;
2. $G^f_{h_j}$ contient les lignes de G dont les étendues sont incluses dans l'intervalle $[h_j + 1, n]$;
3. $G^s_{h_j}$ contient les lignes de G dont les étendues *actives* contiennent h_j.

Il est clair que $G^p_{h_j}$ et $G^f_{h_j}$ engendrent les codes préfixe et suffixe C_{0,h_j} et $C_{h_j,n}$, respectivement. Soit $A^s_{h_j}$ l'ensemble des bits d'information qui correspondent aux lignes de$G^s_{h_j}$. Alors les bits de $A^s_{h_j}$ définissent S_{h_j}, c.-à-d. pour chaque ρ_{h_j}-uplet qui représente des valeurs des bits dans $A^s_{h_j}$, il existe un état correspondant dans S_{h_j} (car $\rho_{h_j} = |A^s_{h_j}|$).

Pour déterminer les arêtes composées entre les états deS_{h_j} et les états de $S_{h_{j+1}}$ ainsi que les arêtes parallèles entre deux états adjacents, on divise$G^f_{h_j}$ en trois sous-ensembles distincts :

1. $G^{f,p}_{h_j,h_{j+1}}$ contient les lignes de $G^f_{h_j}$ dont les étendues sont contenues dans l'intervalle $[h_j + 1, h_{j+1}]$;
2. $G^{f,s}_{h_j,h_{j+1}}$ contient les lignes de $G^f_{h_j}$ dont les étendues *actives* contiennent h_{j+1} ;
3. les lignes de $G^f_{h_j}$ qui restent forment $G^f_{h_{j+1}}$.

Soient $p_{h_j,h_{j+1}}(G^{f,p}_{h_j,h_{j+1}})$, $p_{h_j,h_{j+1}}(G^{f,s}_{h_j,h_{j+1}})$ et $p_{h_j,h_{j+1}}(G^s_{h_j})$ les matrices provenant de$G^{f,p}_{h_j,h_{j+1}}$, $G^{f,s}_{h_j,h_{j+1}}$ et $G^s_{h_j}$ en ne gardant que les colonnes d'indices compris entreh_j et h_{j+1}.

Remarque 3.5 Chaque arête composée entre deux états $s_j \in S_j$ et $s_{j+1} \in S_{j+1}$ est un coset qui appartient à la partition $p_{h_j,h_{j+1}}(C)/C^{\mathrm{tr}}_{h_j,h_{j+1}}$, donc le nombre des arêtes composées entre deux états adjacents est $|C^{\mathrm{tr}}_{h_j,h_{j+1}}|$. Les lignes de $p_{h_j,h_{j+1}}(G^{f,p}_{h_j,h_{j+1}})$ engendrent le code $C^{\mathrm{tr}}_{h_j,h_{j+1}}$ et les lignes de $p_{h_j,h_{j+1}}(G^{f,p}_{h_j,h_{j+1}})$, $p_{h_j,h_{j+1}}(G^{f,s}_{h_j,h_{j+1}})$ et $p_{h_j,h_{j+1}}(G^{s}_{h_j})$ engendrent $p_{h_j,h_{j+1}}(C)$.

Soit $s_j \in S_j$ défini par le ρ_{h_j}-uplet formé par les bits de $A^s_{h_j}$:

$$(a^{(j)}_1,\, a^{(j)}_2, \ldots,\, a^{(j)}_{\rho_{h_j}}), \tag{3.21}$$

où $\rho_{h_j} = \log_2 |S_{h_j}| = |G^s_{h_j}|$. Soient $\{\mathbf{g}^{(j)}_1,\, \mathbf{g}^{(j)}_2, \ldots,\, \mathbf{g}^{(j)}_{\rho_{h_j}}\}$ les lignes de $G^s_{h_j}$. Alors :

$$\mathbf{u} = a^{(j)}_1 \cdot \mathbf{g}^{(j)}_1 + a^{(j)}_2 \cdot \mathbf{g}^{(j)}_2 + \ldots + a^{(j)}_{\rho_{h_j}} \cdot \mathbf{g}^{(j)}_{\rho_{h_j}} \tag{3.22}$$

est un mot de code (ou chemin) qui passe par s_{h_j} au niveau h_j.

Soit $B_{h_j,h_{j+1}}$ le code de longueur $h_{j+1} - h_j$ engendré par $p_{h_j,h_{j+1}}(G^{f,s}_{h_j,h_{j+1}})$. Alors pour chaque $\mathbf{b} \in B_{h_j,h_{j+1}}$, il existe une arête composée qui part de s_{h_j}, et qui est constituée des arêtes suivantes :

$$\{p_{h_j,h_{j+1}}(\mathbf{u}) + \mathbf{b} + \mathbf{x} \;:\; \mathbf{x} \in C^{\mathrm{tr}}_{h_j,h_{j+1}}\}. \tag{3.23}$$

Remarque 3.6 Le nombre des arêtes composées qui partent de s_{h_j} est par conséquent $|B_{h_j,h_{j+1}}|$.

Pour la construction de $T(U)$, les états seront *indexés* en utilisant la matrice de contrôle H (pour une méthode basée sur G voir [64, Ch. 4]). Considérons premièrement le cas plus général d'un treillis n-sectionné. Soit

$$H = [\mathbf{h}_1,\, \mathbf{h}_2, \ldots,\, \mathbf{h}_j, \ldots,\, \mathbf{h}_n] \tag{3.24}$$

la matrice de contrôle, où $\mathbf{h}_j$ est la j-ème colonne de H $(1 \le j \le n)$.
Soit H_i la sous-matrice qui est formée par les i premières colonnes de H $(1 \le i \le n)$:

$$H_i = [\mathbf{h}_1,\, \mathbf{h}_2, \ldots,\, \mathbf{h}_i]. \tag{3.25}$$

Il est évident que le rang de H_i est au plus $n - k$:

$$\mathrm{Rang}(H_i) \le n - k, \tag{3.26}$$

et que, via (1.1), pour chaque $\mathbf{c} \in C^{\mathrm{tr}}_{0,i}$ on a :

$$\mathbf{c} \cdot H^t_i = \mathbf{0}_{n-k}. \tag{3.27}$$

Considérons la partition $p_{0,i}(C)/C^{\mathrm{tr}}_{0,i}$ et D un élément de cette partition différent de $C^{\mathrm{tr}}_{0,i}$. Pour chaque $\mathbf{a} \in D$,

$$\mathbf{a} \cdot H^t_i = (s_1,\, s_2, \ldots,\, s_{n-k}) \neq \mathbf{0}_{n-k} \tag{3.28}$$

et cette valeur est *la même* pour tous les vecteurs de D, c'est-à-dire, pour $\mathbf{a}_1,\, \mathbf{a}_2 \in D$ et $\mathbf{a}_1 \neq \mathbf{a}_2$,

$$\mathbf{a}_1 \cdot H^t_i = \mathbf{a}_2 \cdot H^t_i = (s_1,\, s_2, \ldots,\, s_{n-k}). \tag{3.29}$$

Le $(n-k)$-uplet $(s_1, s_2, \ldots, s_{n-k})$ est appelé *l'indice* du translaté D. Soient D_1 et D_2 deux translatés différents appartenant à $p_{0,i}(C)/C_{0,i}^{\text{tr}}$ et $\mathbf{a}_1 \in D_1$, $\mathbf{a}_2 \in D_2$. Cela implique que $\mathbf{a}_1 \neq \mathbf{a}_2$ et

$$\mathbf{a}_1 \cdot H_i^t \neq \mathbf{a}_2 \cdot H_i^t,$$

autrement dit *des translatés différents ont des indices différents* Ces remarques sont à la base de la définition suivante.

Définition 3.8 *Soit $L(s_0, s)$ l'ensemble des chemins dans le treillis associé à C qui relient s_0 avec un $s \in S_i$.* L'indice $\ell(s)$ d'un état $s \in S_i\,(0 \leq i \leq n)$, *calculé à partir d'une* matrice de contrôle H *de C, est défini par le $(n-k)$-uplet :*

$$\ell(s) = \mathbf{a} \cdot H_i^t = (s_1, s_2, \ldots, s_{n-k}), \tag{3.30}$$

pour tout $\mathbf{a} \in L(s_0, s)$. Pour $i = 0$, $H_i = \emptyset$ et l'état initial s_0 est indexé par le $(n-k)$-uplet $\mathbf{0}_{n-k}$. Pour $i = n$, $L(s_0, s_f) = C$ et l'état final s_f est aussi indexé par $\mathbf{0}_{n-k}$.

Remarque 3.7 La définition précédente, l'application bijective entre les états deS_i et les cosets appartenant à $p_{0,i}(C)/C_{0,i}^{\text{tr}}\,(0 \leq i \leq n)$, et la relation (3.30) impliquent que chaque état $s \in S_i$ a un *unique indice* et que des états différents ont des indices différents.

Pour $0 \leq i < n$, soient $s_i \in S_i$ et $s_{i+1} \in S_{i+1}$ deux états adjacents et u_{i+1} l'indice de l'arête qui les relie. La définition de l'indice d'un état implique :

$$\begin{aligned} \ell(s_{i+1}) &= (u_1, u_2, \ldots, u_i, u_{i+1}) \cdot H_{i+1}^t = (u_1, u_2, \ldots, u_i) \cdot H_i^t + u_{i+1} \cdot \mathbf{h}_{i+1}^t \\ &= \ell(s_i) + u_{i+1} \cdot \mathbf{h}_{i+1}^t, \end{aligned} \tag{3.31}$$

donc, étant donnés un état indexé $\ell(s_i)$ à l'instant i et la sortie u_{i+1}, l'état d'arrivée indexé par $\ell(s_{i+1})$ est déterminé de manière unique.

Revenons à la construction d'un treillis ℓ-sectionné. Soit

$$H_{h_j, h_{j+1}} = [\mathbf{h}_{h_j\,+\,1}, \mathbf{h}_{h_j+2}, \ldots, \mathbf{h}_{h_{j+1}}] \tag{3.32}$$

la sous-matrice de H qui est formée par les colonnes de $\mathbf{h}_{h_j+1}$ à $\mathbf{h}_{h_{j+1}}$. Soit

$$\ell(s_{h_j}) = p_{0,h_j}(\mathbf{u}) \cdot H_{h_j}^t \tag{3.33}$$

l'indice de l'état s_{h_j}. Alors l'arête composée donnée par la relation (3.23) relie s_{h_j} avec un état $s_{h_{j+1}} \in S_{h_{j+1}}$ d'indice :

$$\ell(s_{h_{j+1}}) = \ell(s_{h_j}) + (p_{h_j, h_{j+1}}(\mathbf{u}) + \mathbf{b}) \cdot H_{h_j, h_{j+1}}^t. \tag{3.34}$$

Pour simplifier la construction de la $(j+1)$-ème section entre les instants h_j et h_{j+1}, on construit à la fin de la construction de la j-ème section une table notée Q_{h_j}. Chaque entrée de Q_{h_j} est un triplet :

$$(\mathbf{f}, \ell(s), \mathbf{c}).$$

La première composante $\mathbf{f}$ est le ρ_{h_j}-uplet formé par une combinaison spécifique de ρ_{h_j} bits d'information de $A^s_{h_j}$ et qui définit un état s de S_{h_j}. La seconde composante $\ell(s)$ est l'indice de s, tandis que la troisième composante est donnée par :

$$\mathbf{c} = p_{h_j,\,h_{j+1}}(\mathbf{u}) = p_{h_j,\,h_{j+1}}(\mathbf{f} \cdot G^s_{h_j}), \tag{3.35}$$

où $\mathbf{u} = \mathbf{f} \cdot G^s_{h_j}$ est donné par (3.22).

La construction de la $(j+1)$-ème section de $T(U)$ se déroule de la manière suivante :

(1) on construit $C^{\mathrm{tr}}_{h_j,\,h_{j+1}}$ et $B_{h_j,\,h_{j+1}}$;

(2) pour chaque entrée $(\mathbf{f},\, \ell(s),\, \mathbf{c}) \in Q_{h_j}$ et chaque $\mathbf{b} \in B_{h_j,\,h_{j+1}}$, on construit l'arête composée :

$$\{\mathbf{b} + \mathbf{c} + \mathbf{x} \ :\ \mathbf{x} \in C^{\mathrm{tr}}_{h_j,\,h_{j+1}}\}\ ; \tag{3.36}$$

(3) pour chaque état de départ $s \in S_{h_j}$ et chaque $\mathbf{b} \in B_{h_j,\,h_{j+1}}$, l'état d'arrivée $s' \in S_{h_{j+1}}$ est indexé par :

$$\ell(s') = \ell(s) + (\mathbf{b} + \mathbf{c}) \cdot H^t_{h_j,\,h_{j+1}}. \tag{3.37}$$

Ces étapes sont répétées jusqu'à la construction de la ℓ-ième section de $T(U)$.

3.4.4 Décomposition parallèle

Nous avons donné dans le paragraphe précédent une méthode de construction d'un treillis ℓ-sectionné minimal pour un code en bloc. Cela permet de diminuer le nombre des états et la complexité des arêtes, mais, pour des codes de grandes longueurs et dimensions, la densité des connexions entre les états peut s'avérer insurmontable pour une implémentation.

La **décomposition parallèle** *d'un treillis minimal est la décomposition en sous-treillis parallèles et structurellement identiques sans connexions entre eux, telle que chaque sous-treillis ait une dimension petite et une complexité raisonnable.*

Ici, nous rappelons seulement le résultat suivant ; une étude approfondie peut être trouvée en [64, Ch. 7].

Théorème 3.7 *Le logarithme en base* 2 *du nombre de sous-treillis parallèles et isomorphes dans un treillis minimal ℓ-sectionné avec les bornes de la section $\{h_0, h_1, \ldots, h_\ell\}$ est donné par le nombre des lignes de la matrice adaptée à la structure du treillis, dont les étendues actives contiennent $\{h_1, h_2, \ldots, h_{\ell-1}\}$.*

Exemple 3.6 Considérons le code $RM(r, m)$ obtenu via la construction carrée itérative (voir Section 2.2) et son treillis minimal 2^i-sectionné, avec $0 \le i \le r$. La matrice génératrice de ce code est donnée par la relation (2.10) :

$$G(r,m) = Id_{2^i} \otimes G(r-i, m-i) + \sum_{0 \le j < i} G(j,i) \otimes \Delta((r-j)/(r-j-1), m-i).$$

Pour construire une matrice TOGM, il suffit de mettre en forme TOGM toutes les matrices dans cette formule. À noter que si $G(r, m-2)$ est en forme TOGM, il en est de même pour tous les $\Delta((r-j)/(r-j-1), m-i)$, ainsi que pour $G(r-i, m-i)$.

Mais cette opération ne change pratiquement rien pour ce qui est des lignes dont les étendues actives contiennent 2^{m-i}, $2 \times 2^{m-i}$, $3 \times 2^{m-i}$, ..., et $(2^i - 1)2^{m-i}$. Il s'agit seulement des lignes de $G(0, i) \otimes \Delta((r)/(r-1), m-i) = (1, 1, \ldots, 1) \otimes \Delta((r)/(r-1), m-i)$ qui sont au nombre de $k_{r,m-i} - k_{r-1,m-i} = \begin{pmatrix} m-i \\ r \end{pmatrix}$—via la relation (1.5).

Le treillis minimal 2^i-sectionné de $RM(r, m)$ consiste donc en $2^{\begin{pmatrix} m-i \\ r \end{pmatrix}}$ sous-treillis parallèles et isomorphes, sans connexions entre eux.

3.4.5 Exemple de construction

Considérons le code de Reed-Muller [8, 4, 4] et sa TOGM (cf. Exemple 3.1) :

$$G = \begin{bmatrix} \mathbf{g_1} \\ \mathbf{g_2} \\ \mathbf{g_3} \\ \mathbf{g_4} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Sachant que ce code est auto-dual, on prendra la matrice$G$ à la place de la matrice de contrôle H. Cela nous permettera aussi d'éviter la confusion entre les bornes de la section (notéesh_j) et les lignes de H (notées $\mathbf{h}_j$).

Supposons qu'on veuille construire un treillis 4-sectionné avec$U = \{0, 2, 4, 6, 8\}$.

1. La première section ($h_0 = 0$, $h_1 = 2$).
   Nous calculons $G_0^p = \emptyset$, $G_0^s = \emptyset$, $G_0^f = \{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4\}$.
   (1) $C_{0,2}^{\mathrm{tr}} = \{\mathbf{0}\}$ et $G_{0,2}^{f,s} = \{\mathbf{g}_1, \mathbf{g}_2\}$, donc le code $B_{0,2}$ a pour matrice génératrice :

   $$p_{0,2}(G_{0,2}^{f,s}) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

   (2) $Q_0 = \{(0, 0000, 0)\}$ et $\mathbf{b} \in \{00, 01, 11, 10\}$. D'après (3.36) les arêtes qui partent de s_0 sont indexées par 00, 01, 11, 10 respectivement.
   (3) On calcule les états dans S_1 en utilisant (3.37) et on trouve 0000, 1100, 0100, 1000.

2. La deuxième section ($h_1 = 2$, $h_2 = 4$).
   On trouve $G_2^p = \emptyset$, $G_2^s = \{\mathbf{g}_1, \mathbf{g}_2\}$, $G_2^f = \{\mathbf{g}_3, \mathbf{g}_4\}$.
   (1) $C_{2,4}^{\mathrm{tr}} = \{\mathbf{0}\}$ et $G_{2,4}^{f,s} = \{\mathbf{g}_3\}$, donc le code $B_{2,4}$ a pour matrice génératrice :

   $$p_{2,4}(G_{2,4}^{f,s}) = \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

   (2) Nous construisons tout d'abord la table Q_2, $\mathbf{f}$ étant un 2-uplet formé par une combinaison de a_1 et a_2, donc $\mathbf{f} \in \{00, 01, 11, 10\}$. Les valeurs successives prises par $\mathbf{u}$ sont $\mathbf{0}$, $\mathbf{g}_2$, $\mathbf{g}_1 + \mathbf{g}_2$, $\mathbf{g}_1$ et donc $\mathbf{c} \in \{00, 01, 10, 11\}$. En tenant compte que

   $$H_2^t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix},$$

on calcule les indices des états (dans S_1) correspondant aux valeurs de $\mathbf{u}$: 0000, 1100, 1000, 0100, respectivement. Nous avons ainsi obtenu :

$$Q_2 = \{(00,\, 0000, 00),\, (01,\, 1100,\, 01),\, (11,\, 1000,\, 10),\, (10,\, 0100,\, 11)\}.$$

En parcourant cette table et les valeurs de $\mathbf{b} \in \{00, 11\}$ on obtient en utilisant (3.36) les indices des arêtes de cette section (00, 11), (01, 10), (10, 01), (11, 00) (les paires correspondent au même sommet dans S_1).
(3) On calcule les états dans S_2 en utilisant (3.37) et on trouve $(0000, 0100)$, (0010, 0110), (0010, 0110), (0000, 0100) (correspondant aux paires des arêtes antérieures).

3. La troisième section ($h_2 = 4$, $h_3 = 6$).
Nous calculons $G_4^p = \{\mathbf{g}_1\}$, $G_4^s = \{\mathbf{g}_2,\, \mathbf{g}_3\}$, $G_4^f = \{\mathbf{g}_4\}$.
(1) $C_{4,6}^{\mathrm{tr}} = \{\mathbf{0}\}$ et $G_{4,6}^{f,\,s} = \{\mathbf{g}_4\}$, donc le code $B_{4,6}$ a pour matrice génératrice :

$$p_{4,6}(G_{4,6}^{f,\,s}) = \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

(2) Nous construisons la table Q_4, $\mathbf{f}$ étant un 2-uplet formé par une combinaison de a_2 et a_3, donc $\mathbf{f} \in \{00,\, 01,\, 11,\, 10\}$. Les valeurs successives prises par $\mathbf{u}$ sont $\mathbf{0}$, $\mathbf{g}_3$, $\mathbf{g}_2 + \mathbf{g}_3$, $\mathbf{g}_2$ et donc $\mathbf{c} \in \{00,\, 11,\, 01,\, 10\}$. Mais

$$H_4^t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix},$$

donc les indices des états (dans S_2) correspondant aux valeurs de $\mathbf{u}$ sont : 0000, 0100, 0110, 0010, respectivement. Par conséquent :

$$Q_4 = \{(00,\, 0000,\, 00),\, (01,\, 0100,\, 11),\, (11,\, 0110,\, 01),\, (10,\, 0010,\, 10)\}.$$

En parcourant cette table et les valeurs de $\mathbf{b} \in \{00, 11\}$ on obtient en utilisant (3.36) les indices des arêtes de cette section (00, 11), (11, 00), (01, 10), (10, 01) (les paires correspondent au même sommet dans S_2).
(3) On calcule les états dans S_3 en utilisant (3.37) et on trouve $(0000, 0100)$, $(0000, 0100)$, $(0101, 0001)$, $(0101, 0001)$ (correspondant aux paires des arêtes antérieures).

4. La dernière section ($h_3 = 6$, $h_4 = 8$).
On trouve $G_6^p = \{\mathbf{g}_1,\, \mathbf{g}_3\}$, $G_6^s = \{\mathbf{g}_2,\, \mathbf{g}_4\}$, $G_6^f = \emptyset$.
(1) $C_{6,8}^{\mathrm{tr}} = \{\mathbf{0}\}$ et $G_{6,8}^{f,\,s} = \emptyset$, donc $B_{6,8} = \emptyset$.
(2) On construit la table Q_6, $\mathbf{f}$ étant un 2-uplet formé par une combinaison de a_2 et a_4, donc $\mathbf{f} \in \{00,\, 01,\, 11,\, 10\}$. Les valeurs successives prises par $\mathbf{u}$ sont $\mathbf{0}$, $\mathbf{g}_4$, $\mathbf{g}_2 + \mathbf{g}_4$, $\mathbf{g}_2$ et

donc $\mathbf{c} \in \{00, 11, 01, 10\}$. En tenant compte que

$$H_6^t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

on calcule les indices des états (dans S_3) correspondant aux valeurs de $\mathbf{u}$: 0000, 0100, 0001, 0101, respectivement. Nous avons ainsi obtenu :

$$Q_6 = \{(00, 0000, 00), (01, 0100, 11), (11, 0001, 01), (10, 0101, 10)\}.$$

En parcourant cette table on obtient en utilisant (3.36) les indices des arêtes de cette section 00, 11, 01, 10.

(3) On calcule les états dans S_4 en utilisant (3.37) et on trouve l'état final 0000.

Tous ces résultats sont résumés par la figure 3.3 (l'ordre dans les combinaisons des 2 bits est changé en $\{00, 11, 01, 10\}$ pour rendre le treillis plus clair). Nous soulignons encore la symétrie par un miroir (sauf pour les indices des états) et les deux sous-treillis parallèles et structurellement identiques déjà vus dans le paragraphe 3.1.

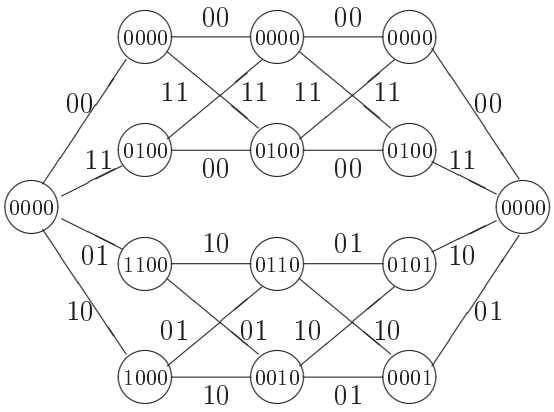

FIG. 3.3 – Un treillis minimal 4-sectionné à états indicés du code $RM(1,3)$

3.5 Deuxième application : treillis des codes de Reed-Muller

Nous détaillons dans cette section les treillis sectionnés des codes obtenus via la construction carrée. Nous allons naturellement nous reporter aux résultats dans la Section 2.2. Pour simplifier cette démarche, nous traitons seulement la construction carrée du deuxième niveau.

Considérons le code $C = |C_0/C_1/C_2|^4$ de longueur $4n$ et sa matrice génératrice donnée par (2.5) ou (2.6). Pour la mettre en forme TOGM, il suffit de le faire pour toutes les matrices qui interviennent. Mais une fois la matrice G_0 en forme adaptée à la structure du treillis, il en est de même pour $G_{0/1}$, $G_{1/2}$ et G_2. De plus, $G(1,2)$ peut être mise sous la forme TOGM :

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

La TOGM qui sera utilisée pour la construction du treillis 4-sectionné de $C = |C_0/C_1/C_2|^4$ est :

$$G = (1111) \otimes G_{0/1} + \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \otimes G_{1/2} + Id_4 \otimes G_2. \tag{3.38}$$

Utilisant les notations et les résultats de la Section 3.4.3 nous obtenons :

$$\begin{aligned} G^s_n &= (1111) \otimes G_{0/1} + (1100) \otimes G_{1/2}, \\ G^s_{2n} &= (1111) \otimes G_{0/1} + (0110) \otimes G_{1/2}, \\ G^s_{3n} &= (1111) \otimes G_{0/1} + (0011) \otimes G_{1/2}. \end{aligned}$$

$$\begin{aligned} G^{f,p}_{0,n} &= (1000) \otimes G_2, \quad G^{f,p}_{n,2n} = (0100) \otimes G_2, \\ G^{f,p}_{2n,3n} &= (0010) \otimes G_2, \quad G^{f,p}_{3n,4n} = (0001) \otimes G_2. \end{aligned}$$

$$\begin{aligned} G^{f,s}_{0,n} &= (1111) \otimes G_{0/1} + (1100) \times G_{1/2}, \\ G^{f,s}_{n,2n} &= (0110) \times G_{1/2}, \\ G^{f,s}_{2n,3n} &= (0011) \times G_{1/2}, \\ G^{f,s}_{3n,4n} &= \emptyset. \end{aligned}$$

Si k_i désigne la dimension du code C_i, la structure du treillis a les propriétés suivantes :

1. Le nombre des états dans S_n, S_{2n} et S_{3n} est le même :
$$|G^s_n| = |G^s_{2n}| = |G^s_{3n}| = 2^{k_0 - k_2}.$$
2. Les arêtes parallèles dans chacune des 4 sections sont des translatés de C_2 dans C_0/C_1. En relation (3.23) $\mathbf{u}$ et $\mathbf{b}$ sont fixés et $\mathbf{x}$ variable
$$\mathbf{x} \in C^{\mathrm{tr}}_{in,\,(i+1)n} = C^{f,p}_{in,\,(i+1)n} = C_2.$$
Chaque arête composée consiste donc en 2^{k_2} arêtes.
3. Le nombre des arêtes parallèles qui partent de chaque état est
$$|B_{in,\,(i+1)n}| = |G^{f,s}_{in,\,(i+1)n}|,$$
car en relation (3.23) $\mathbf{u}$ et $\mathbf{x}$ sont fixés et $\mathbf{b}$ variable, $\mathbf{b} \in B_{in,\,(i+1)n}$.
Cela donne $2^{k_0 - k_2}$ arêtes divergeant de s_0, $2^{k_1 - k_2}$ arêtes divergeant de chaque état en S_n ou S_{2n} ; tous les états en S_{3n} convergent dans s_f.

4. Le nombre total des arêtes composées dans la deuxième et troisième section est

$$2^{k_0-k_2}2^{k_1-k_2} = 2^{k_0+k_1-2k_2}.$$

5. Toutes les arêtes qui partent d'un état dans S_n ou S_{2n} ($\mathbf{u}$ seulement est fixé en relation (3.23)) sont des translatés de C_1 dans C_0.
6. Les sections 2 et 3 sont structurellement identiques. En plus les sections 3 et 4 sont les images "par miroir" des sections 2 et 1.
7. Le nombre des sous-treillis parallèles et isomorphes, sans connexions entre eux, est

$$2^{|G_{0/1}|} = 2^{k_0-k_1}.$$

Deuxième partie

Distances minimales

Préambule

Dans ce préambule nous présentons les méthodes utilisées pour le calcul des distances minimales ou de bonnes bornes supérieures des codes *linéaires*. Par la suite nous allons analyser le cas des codes BCH et de leurs duaux.

Tout d'abord, rappelons que trouver une borne supérieure pour la distance minimale dans un code se réduit à la recherche des mots de petit poids. La recherche d'un mot de petit poids dans un code intervient dans la théorie des codes, ainsi que dans la cryptographie (via les systèmes à clé publique qui utilisent les codes correcteurs d'erreurs). Une analyse de ces deux approches est donnée en [18] ; nous rappellerons ici seulement les aspects liés à la théorie des codes. Pour une analyse sur la complexité, voir [99].

Avant tout, précisons qu'un *petit poids* est un poids supposé proche de la distance minimale du code. Cela veut dire que par cette recherche nous pouvons donner des estimations (bornes supérieures) de la distance minimale du code, ce qui est très important car pour beaucoup de familles de codes on ne connaît que des bornes inférieures. Or il s'avère important de savoir si ces bornes sont bonnes, sachant que la distance minimale et, plus généralement, la distribution des poids, caractérisent la performance d'un code.

Un algorithme qui cherche un mot de petit poids sert aussi au décodage en deçà de la capacité de correction. Considérons un code $C[n, k, d]$ de capacité de correction $t = \lfloor (d-1)/2 \rfloor$ et supposons qu'un message $\mathbf{x}$ est un mot de code transmis avec une erreur $\mathbf{e}$ telle que $\mathrm{wt}(\mathbf{e}) \leq t$. Alors le décodage consiste à retrouver $\mathbf{e}$ (donc le mot de code), par recherche du mot de plus petit poids dans le code $C + \mathbf{x}$. Ainsi le problème du décodage d'un code $[n, k]$ est équivalent à la recherche du mot de plus petit poids dans un code $[n, k+1]$.

Rappelons que pour certaines classes de codes les distances minimales sont connues (voir la proposition 1.5 dans le cas des codes de Reed-Muller). Dans certains cas, le calcul des distances minimales est facilité par l'existence de bonnes bornes supérieures ou inférieures [2], par des propriétés de divisibilité vérifiées par les poids des mots du code [63], par le calcul des treillis de poids minimal ([26], [64]). Pourtant, il est montré que le calcul de la distance minimale d pour un code arbitraire est un problème difficile [28]. Un algorithme probabiliste qui donne une bonne borne supérieure pour d est rappelé par la suite ; il présente l'avantage d'être applicable à n'importe quelle classe de codes.

Recherche des mots de poids minimal

Dans ce paragraphe, nous rappelons l'algorithme probabiliste Canteaut-Chabaud qui recherche des mots de poids faible dans de grands codes (pour une description complète de cette approche voir [17]). L'algorithme [18, p.369] est basé sur la méthode de *décodage par ensemble d'information*.

Considérons le code $C[n, k, d]$ de matrice génératrice G. Notons N l'ensemble $\{1, \ldots, n\}$ des indices des coordonnées des mots du code et G_i la i-ème colonne de G. Pour chaque sous-ensemble I de N, $G = (V, W)_I$ désigne la décomposition de la matrice G par rapport à I, c'est-à-dire $V = (G_i)_{i \in I}$ et $W = (G_j)_{j \in N \setminus I}$.

Plus précisément, l'algorithme est le suivant :

Initialisation

On choisit aléatoirement un ensemble d'information I (voir Définition 1.2) et on applique l'élimination de Gauss afin d'obtenir une matrice génératrice en forme systématique $(\mathrm{Id}_k,\, Z)_I$.

Jusqu'à ce qu'un mot de poids w soit trouvé

- on construit une partition *aléatoire* de I, I_1 et I_2, telle que $|I_1| = \lfloor k/2 \rfloor$ et $|I_2| = \lceil k/2 \rceil$. Cela induit une partition des lignes de Z en Z_1 et Z_2 ;
- on choisit aléatoirement un sous-ensemble de l'ensemble de redondance,$L \subset J$, de cardinal σ ;
- pour toute combinaison linéaire Λ_1 de p lignes de Z_1, on calcule sa restriction à L, soit $\Lambda_{1|L}$, et on stocke toutes ces valeurs dans une table de hachage avec 2^σ entrées ; par ailleurs on a
$$\mathrm{wt}(\Lambda_{1|I_1}) = p,\ \mathrm{wt}(\Lambda_{1|I_2}) = 0 \ ;$$
- pour toute combinaison linéaire Λ_2 de p lignes de Z_2, on calcule $\Lambda_{2|L}$ et on stocke toutes ces valeurs dans une table de hachage avec 2^σ entrées ; par ailleurs on a
$$\mathrm{wt}(\Lambda_{2|I_1}) = 0,\ \mathrm{wt}(\Lambda_{2|I_2}) = p \ ;$$
- à partir de la table de hachage, on repère toutes les paires de combinaisons linéaires $(\Lambda_1,\, \Lambda_2)$ telles que $\Lambda_{1|L} = \Lambda_{2|L}$ (donc $\mathrm{wt}((\Lambda_1 + \Lambda_2)_{|L}) = 0$) et on vérifie si
$$\mathrm{wt}((\Lambda_1 + \Lambda_2)_{|J\setminus L}) = w - 2p \ ;$$
si c'est vrai, alors $\mathrm{wt}(\Lambda_1 + \Lambda_2) = w$ et on arrête.
- sinon, on choisit aléatoirement $\lambda \in I$ et $\mu \in J$ tels que $z_{\lambda,\mu} = 1$. On remplace I par $(I\setminus\{\lambda\}) \cup \{\mu\}$, qui est un autre ensemble d'information. On actualise la matrice Z.

La nouveauté apportée par cet algorithme est que *les ensembles d'information successifs ne sont plus indépendants les uns des autres*; ainsi le nombre moyen d'itérations augmente, mais il y a moins d'opérations effectuées à chaque itération. De plus, à la fin de chaque itération, on passe à un autre ensemble d'information en changeant un seul élément, ce qui assure un calcul peu coûteux de la nouvelle matrice Z. Cela a permis notamment de trouver les distances minimales, précédemment inconnues, de six codes BCH de longueur 512.

L'algorithme est itératif. À chaque itération, si on trouve un mot de poids strictement inférieur au plus petit poids rencontré précédemment, le programme affiche le poids et le support du mot. Il prend en entrée la longueur, la dimension du code et le nom du fichier dans lequel est donnée la matrice génératrice. On peut en plus lui indiquer le poids à chercher (près de la valeur anticipée de la distance minimale, par exemple une borne inférieure connue) et un fichier de sortie pour les résultats. Les paramètres optimaux pour $n \leq 512$ sont $p = 1$ et $\sigma = \lceil \log(n)/\log(2) \rceil$.

Nous avons appliqué cet algorithme aux duaux des codes BCH étendus (ou EBCH) de longueur inférieure ou égale à 512. Les résultats obtenus sont présentés dans le chapitre suivant.

4

Application aux duaux des codes BCH

Dans ce chapitre nous considérons des codes BCH *primitifs binaires* et notre intérêt est focalisé sur les distances minimales des duaux de ces codes. À ce jour, seulement six distances minimales restent encore inconnues dans le cas des codes BCH primitifs de longueur au plus 512. La situation est loin d'être la même pour leurs duaux. Pour $n = 128$, les distributions des poids sont données en [26]. Pour $n = 256$, quelques distributions des poids sont accessibles (voir le lien [103]) et récemment a été publiée en [90] une table mise à jour des poids minimaux. L'algorithme utilisé dans [90] est une variation de l'algorithme de Brouwer (utilisé par Magma) pour les codes cycliques. Il prend pour paramètre un entier ℓ plus petit que k et il travaille sur les $k - \ell$ premiers shifts du polynôme générateur, plus précisément sur un sous-code du code de départ. Bien sûr, pour des performances meilleures que Magma, le choix de ℓ est très important.

Parmi les duaux des codes BCH de longueur 256, douze sont tels que seulement une borne supérieure pour les distances minimales est donnée. Pour cette longueur on connaît aussi des bonnes bornes inférieures [2]. Par contre, peu est connu quand $n = 512$, à l'exception des codes BCH de distance construite petite [89]. Concernant les résultats numériques, nous concentrons notre étude sur cette longueur. Deux méthodes sont utilisées. Premièrement, nous appliquons l'algorithme Canteaut-Chabaud de recherche de poids minimum. Deuxièmement, nous utilisons *la décomposition carrée modifiée* (voir Section 2.3), qui peut être appliquée à n'importe quel code affine-invariant.

Le chapitre est structuré de la manière suivante. La première section récapitule les résultats déjà connus. Le résultat principal de [6] est rappelé en Section 4.2 et par la suite nous l'appliquons à la décomposition carrée modifiée des duaux des codes BCH en ordre standard des bits. Nous analysons ensuite l'incidence des propriétés des codes composants sur les distances duales (voir la première *Remarque importante*). Nos principaux résultats sont en Section 4.3. Deux sous-codes s'imposent à l'attention via la décomposition LTSC. Nous commençons par donner pour un de ces sous-codes, noté B, une borne supérieure pour sa dimension (Proposition 4.3). Nous analysons par la suite le rôle des codes Reed-Muller dans la décomposition des duaux des codes BCH (en Section 4.4). Nous montrons que, pour certaines distances construites et pour des longueurs plus petites ou égales à 512, B est un code de Reed-Muller. En Section 4.5, on traite les distances minimales des duaux des codes BCH. Nous utilisons l'algorithme Canteaut-Chabaud sur le code ou sur un

des sous-codes obtenus par la LTSC. Nous obtenons ainsi des nouvelles bornes supérieures pour les distances duales des codes de longueur 512. Pour les codes de longueur 256, nous obtenons les mêmes distances minimales et les mêmes bornes supérieures qu'en [90].

4.1 Distances minimales connues

Dans cette section nous faisons un rappel des résultats connus à ce jour liés au calcul des distances minimales des duaux des codes BCH. Dans [26], une méthode basée sur le treillis associé au code a conduit au calcul des polynômes énumérateurs de tous les codes EBCH de longueur 128. Les polynômes énumérateurs des codes BCH de longueur 127 sont facilement obtenus via le code étendu [70, p. 232]. En passant par la transformée de MacWilliams, on obtient les polynômes énumérateurs de leurs codes duaux et, implicitement, leurs distances minimales. Par ailleurs, ces distances minimales peuvent être aussi calculées avec Magma.

Pour les résultats numériques, nous nous sommes particulièrement intéressés aux longueurs 255 et 511. La première démarche a été de savoir si Kasami*et al.* ont poursuivi leurs calculs pour des longueurs supérieures à 128. Nous avons ainsi trouvé dans [103] les distributions des poids des codes EBCH(256, k), pour $k \leq 63$, $k \geq 207$ (et avec référence théorique [34]). Par conséquent, nous avons pu calculer les polynômes énumérateurs (donc aussi les distances minimales) des duaux de ces codes. Onze parmi ces quinze distances minimales peuvent être également obtenues avec Magma. La distance minimale du code $BCH^{\perp}(255, 32)$ a été aussi obtenue en [89]. Toujours en longueur 255, il est prouvé en [91] que la distance minimale du code $BCH^{\perp}(255, 56)$ est 64.

Comme d'habitude, l'existence de bonnes bornes inférieures peut simplifier la démarche. Le résultat le plus important relève de la programmation de l'algorithme de Schaub [92] que nous rappelons ici brièvement. À chaque mot de code on associe une matrice circulante dont le rang est le poids du mot. Pour calculer la distance minimale du code, il faudrait trouver le rang minimal des matrices associées à tous les mots de code. Le calcul est simplifié par l'utilisation d'une *matrice générique*, dont on cherche un ensemble de lignes indépendantes. Le nombre de ces lignes représente la borne minimale pour le rang, qu'on appelera par la suite*la borne de Schaub*.

Cet algorithme a été implanté par Augot et Levy-dit-Vehel [2] avec des résultats meilleurs que ceux des bornes théoriques. Par exemple, on obtient 16 pour le $BCH^{\perp}(255, 156)$ et comme un mot de poids 16 a été trouvé dans ce code (utilisant Magma ou l'algorithme Canteaut-Chabaud), nous pouvons conclure que la distance minimale est 16. La même conclusion s'impose pour tous les codes $BCH^{\perp}(255, k)$, $164 \leq k \leq 184$, via la propriété suivante :

Lemme 4.1 *La distance minimale d'un code $BCH^{\perp}$ est une fonction décroissante de la distance construite du code BCH.*

Preuve. Soient $\delta_1 \leq \delta_2$ les distances construites de deux codes BCH de polynômes générateurs g_1, resp. g_2. Alors, par la définition d'un code BCH, leurs ensembles de définition (notés I_1 et I_2), satisfont $I_1 \subset I_2$, ce qui implique $g_1 | g_2$. Nous obtenons successivement $BCH(n, \delta_2) \subset BCH(n, \delta_1)$, puis $BCH^{\perp}(n, \delta_1) \subset BCH^{\perp}(n, \delta_2)$ et finalement la conclusion. ◇

Nous devons aussi ajouter que dans certains cas (indiqués par "**" dans l'Annexe A.1), la

borne de Schaub est atteinte et que l'algorithme Canteaut-Chabaud est particulièrement efficace. Enfin, pour douze codes de longueur 511, nous avons pu calculer les distances minimales en utilisant Magma et le lemme 4.1. (se reporter à l'Annexe A.1 pour ces résultats numériques et leurs références).

À la fin de cette section, un petit bilan s'impose : pour 12 codes de longueur 255 et la plupart des codes de longueur 511, la distance minimale reste inconnue. Par la suite, nous présenterons une méthode qui, bien que plus générale, permettra d'affiner les bornes pour ces distances minimales (voir Annexe A.2).

4.2 Sur la construction carrée modifiée des codes BCH

Nous traitons dans cette section le cas des codes *affines-invariants* (voir la définition 1.15), et en particulier les codes EBCH primitifs. Tout d'abord, nous faisons quelques considérations sur l'importance de la construction carrée modifiée (LTSC). Rappelons que la matrice génératrice d'un code $C = \|A/B\|^2$ est de la forme (voir Section 2.3) :

$$\begin{bmatrix} G_{A/B} & \widetilde{G_{A/B}} \\ G_B & 0 \\ 0 & G_B \end{bmatrix}.$$

Remarque importante. Un code mis en forme LTSC devient particulièrement intéressant pour la recherche des mots de petit poids, grâce aux blocs de zéros. La borne suivante sur la dimension k de C étant évidente

$$d \leq \min\{d_B,\, d_{(A/B,\widetilde{A/B})}\}, \tag{4.1}$$

notre intérêt se dirige vers l'étude des codes B, $(A/B, \widetilde{A/B})$ et A. Cela inclut dimensions, distances minimales (ou bornes supérieures) et, bien sûr, comparaisons avec les codes connus. Comparé au code C, le code B est de longueur moitié et de dimension plus petite, donc pour une majorité des cas si $n \leq 512$, ses paramètres peuvent être calculés en utilisant Magma. Vu que dans certains cas on obtient des codes de Reed-Muller, on cherchera à associer les codes RM de façon précise (voir Section 4.4).

Concernant le code $(A/B, \widetilde{A/B})$, on peut obtenir une matrice génératrice telle quek_B colonnes sont nulles en $G_{A/B}$, ainsi qu'en $G_{\widetilde{A/B}}$ (plus précisément, les colonnes correspondant à l'ensemble d'information de B, voir le lemme suivant). Via la relation (2.12), *plus la dimension de B est grande, plus la dimension et la longueur effective du code $(A/B, \widetilde{A/B})$ deviennent petites.* Cela nous permet de réduire la longueur effective de ce code dans le calcul de sa distance minimale.

Notons que k_B est lié à k_A par l'égalité $k_A + k_B = k$, donc une caractérisation des codes A est aussi importante. Les codes A peuvent être comparés avec les codes EBCH$^\perp$; un exemple est indiqué à la fin de la Section 4.4. □

Lemme 4.2 *On peut construire la matrice génératrice du code $(A/B, \widetilde{A/B})$ telle que le nombre des colonnes nulles soit au moins égal à $2k_B$, où k_B est la dimension du code B.*

Preuve. Soit I_B l'ensemble d'information du code B, qui a évidemment pour cardinal k_B. On pourra supposer que chaque colonne dans A/B qui correspond à un élément dans I_B contient un seul 1. Choisissons donc une de ces colonnes et notons-la i.

Les mots de code dans A/B sont des représentants des translatés de B dans A. On considère seulement la colonne i de ce code et on veut montrer qu'on peut se ramener au cas où cette colonne est nulle. En effet, s'il existe un 1, il suffit de considérer la différence du vecteur $\mathbf{v}$ dans A/B qui le contient et du vecteur dans B qui a un 1 sur la colonne i. Ce vecteur restera toujours dans le même coset de B que $\mathbf{v}$. On peut donc toujours annuler les colonnes dans A/B qui correspondent à I_B. On aura les mêmes colonnes nulles dans $\widetilde{A/B}$, d'où la conclusion. ◇

Remarque 4.1 Pour tous les codes EBCH$^\perp$ que nous étudions, la matrice du code complément renvoyée par Magma a $2k_B$ colonnes nulles.

Le reste de la section est dédié à la mise en forme LTSC des codes affines-invariants. On commence par le passage à *l'ordre standard des bits* [6]. Rappelons qu'on travaille avec un code C de longueur 2^m, donc ses coordonnées peuvent être indexées par un élément $X \in \mathbb{F}_{2^m}$. Soit

$$(\alpha_0, \alpha_1, \ldots, \alpha_{m-1}) \tag{4.2}$$

une base de l'espace vectoriel $\mathbb{F}_{2^m}$ sur $\mathbb{F}_2$. Chaque X peut être représenté par le m-uplet $\psi[X] = (b_0, b_1, \ldots, b_{m-1}) \in \mathbb{F}_2^m$, tel que $b_i \in \{0, 1\}$ pour tout $i \in [0, m-1]$ et

$$X = \sum_{i=0}^{m-1} b_i \alpha_i.$$

Mettre un code affine-invariant C dans l'ordre standard des bits consiste à construire un code équivalent *en permutant chaque coordonnée indexée par* X *en position* $j+1$ *dans* C*, où* j *a pour représentation en base* 2 *le* m*-uplet* $\psi[X]$. Par conséquent :

$$\psi[X] = \begin{cases} (b_0, b_1, \ldots, b_{m-2}, 0) & \text{si } X \in F_0 \\ (b_0, b_1, \ldots, b_{m-2}, 1) & \text{si } X \in F_1, \end{cases} \tag{4.3}$$

où F_0 et F_1 sont définis en Définition 2.6. Une fois cette permutation déterminée, nous passons à la décomposition du code permuté, en calculant le code raccourci B (donc la somme directe $B \oplus B$) et par la suite en déduisant le code complément. Cette construction est particulièrement intéressante quand on travaille avec des codes EBCH, vu la construction de leur *matrice de contrôle*. La construction de cette matrice en ordre standard des bits respecte l'algorithme donné en [6, Appendix].

Soit $BCH(n, \delta)$ un code BCH primitif de longueur $n = 2^m - 1$ et de distance construite $\delta = 2e + 1$. Soit $I = \{i_1, i_2, \ldots, i_e\}$ l'ensemble de définition de ce code et $\mathcal{H}_\alpha$ la représentation de sa matrice de contrôle sur $\mathbb{F}_{2^m}$, où α est une racine primitive dans $\mathbb{F}_{2^m}$. La matrice $\mathcal{H}_\alpha$ est donc de la forme :

$$\begin{pmatrix} 1 & \alpha^{i_1} & \alpha^{2i_1} & \ldots & \alpha^{ji_1} & \ldots & \alpha^{(2^m-2)i_1} \\ 1 & \alpha^{i_2} & \alpha^{2i_2} & \ldots & \alpha^{ji_2} & \ldots & \alpha^{(2^m-2)i_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{i_e} & \alpha^{2i_e} & \ldots & \alpha^{ji_e} & \ldots & \alpha^{(2^m-2)i_e} \end{pmatrix}, \tag{4.4}$$

avec $j \in [0, n-1]$. En *ordre classique*, la coordonnée en position j sera notée $X = \alpha^j$. Pour obtenir la matrice de parité du code étendu sur $\mathbb{F}_{2^m}$, $\hat{\mathcal{H}}_\alpha$, il suffit d'ajouter une colonne nulle et une ligne $(1, 1, \ldots, 1)$. La nouvelle coordonnée sera notée par l'élément 0 de $\mathbb{F}_{2^m}$.

Mettons la matrice $\hat{\mathcal{H}}_\alpha$ en *ordre standard des bits*. Cela signifie qu'*on ordonne les éléments de l'ensemble* $\{0\} \cup \{\alpha^j,\ j \in [0, 2^m - 2]\}$ *par rapport aux décompositions en base (4.2)*. Évidemment, la colonne nulle reste la première et notons α^{z_j} la nouvelle coordonnée en position $j+1$, $j \in [0,\ 2^m - 2]$. La matrice ainsi permutée, notée $\hat{\mathcal{H}}'_\alpha$, a pour $(j+1)$-ième colonne le vecteur :

$$(\alpha^{i_1 z_j},\ \alpha^{i_2 z_j},\ \ldots,\ \alpha^{i_e z_j})^t.$$

Pour obtenir la matrice de contrôle *binaire en ordre standard* $\hat{\mathcal{H}}'$, il suffit de remplacer chaque entrée $\alpha^{i_k z_j}$ dans

$$\hat{\mathcal{H}}'_\alpha = \begin{pmatrix} 1 & 1 & 1 & 1 & \ldots & 1 & \ldots & 1 \\ 0 & 1 & \alpha^{i_1 z_1} & \alpha^{i_1 z_2} & \ldots & \alpha^{i_1 z_j} & \ldots & \alpha^{i_1 z_{2^m-2}} \\ 0 & 1 & \alpha^{i_2 z_1} & \alpha^{i_2 z_2} & \ldots & \alpha^{i_2 z_j} & \ldots & \alpha^{i_2 z_{2^m-2}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & \alpha^{i_e z_1} & \alpha^{i_e z_2} & \ldots & \alpha^{i_e z_j} & \ldots & \alpha^{i_e z_{2^m-2}} \end{pmatrix}, \tag{4.5}$$

par sa représentation en la base (4.2).

Remarque importante. Les colonnes sont maintenant indexées par 0, $\alpha^{z_0} = 1$, $\alpha^{z_1} = \alpha = (0, 1, 0, \ldots, 0)$, $\alpha^{z_2} = 1 + \alpha = (1, 1, 0, \ldots, 0)$, $\ldots$, $\alpha^{z_j} = \phi[j+1]$, $\ldots$, $\alpha^{z_{2^m-2}} = \phi[2^m - 1]$, où ϕ est ici la décomposition en base 2 d'un entier. □

Une fois cette matrice construite, nous pouvons énoncer le résultat principal de cette section (voir la définition 2.9 pour la récursivité) :

Théorème 4.1 *[6, Théorème 4] Les codes affines-invariants (et en particulier les EBCH primitifs -et leurs duaux-) sont des codes LTSC réversibles symétriques* récursifs.

La preuve nécessite le résultat auxiliaire suivant :

Lemme 4.3 *Pour tout* $i \in [1, m]$, *on considère deux* m*-uplets :*

- $\sigma^i = (0, \ldots, 0, 1, 0, \ldots, 0)$ *qui a toutes les coordonnées nulles, sauf celle en position* $i-1$ *;*
- $\omega^i = (1, \ldots, 1, 0, \ldots, 0)$ *qui a toutes les* i *premières coordonnées* 1, *et les* $m - i$ *autres sont nulles.*

Soient les permutations affines τ_s^i *définies par*

$$\psi[X] \mapsto \psi[X] + \sigma^i$$

et τ_r^i *définies par*

$$\psi[X] \mapsto \psi[X] + \omega^i.$$

Alors, la permutation affine τ_s^m *échange les* $n/2$ *premières et dernières coordonnées du code, et la permutation* τ_r^m *inverse l'ordre des coordonnées.*

Preuve. (*du lemme*) Fixons la j-ième colonne $j \in [1,\, 2^{m-1}]$ d'indice $\phi[j-1] = (b_0,\, b_1,\, \ldots,\, b_{m-2},\, b_{m-1})$. Alors τ_s^m transforme $(b_0,\, b_1,\, \ldots,\, b_{m-2},\, b_{m-1})$ en $(b_0,\, b_1,\, \ldots,\, b_{m-2},\, b_{m-1}+1)$, donc $\phi[j-1]$ devient $\phi[j-1+2^{m-1}]$. On conclut que τ_s^m échange la j-ième colonne avec la $(j+2^{m-1})$-ième colonne (et réciproquement).

De plus τ_r^m transforme $(b_0,\, b_1,\, \ldots,\, b_{m-2},\, b_{m-1})$ en $(b_0+1,\, b_1+1,\, \ldots,\, b_{m-2}+1,\, b_{m-1}+1)$, donc $\phi[j-1]$ devient

$$\phi[(j-1)+1+2+\ldots+2^{m-1}] = \phi[2^m-1-(j-1)].$$

On conclut que τ_r^m échange la j-ième colonne avec la (2^m-j+1)-ième colonne (et réciproquement). ◇

Preuve. (*du théorème*) La preuve est basée sur l'application successive du lemme précédent, en considérant les permutations affines τ_s^{m-i} et τ_r^{m-i}, pour $i \in \{1,\, 2,\, \ldots,\, m-1\}$ (on a déjà vu le cas $i=0$). Mais le code étant affine-invariant, il est invariant par rapport aux deux permutations. Le résultat du lemme signifie que le code est symétrique et réversible du premier degré. Les valeurs de i plus grandes que 0 justifient la récursivité. Pour la démonstration complète voir [6]. ◇

Remarque 4.2 Tout code affine-invariant de longueur 2^m est de degré de décomposition m.

4.3 Borne sur la dimension du code B

Dans cette section nous considérons les codes cycliques primitifs (de longueur 2^m) en tant que codes dans une algèbre de corps (pour plus de détails voir [21]). Cela nous permet de mener une étude théorique du code B et de donner une borne supérieure de sa dimension, obtenue par la décomposition carrée modifiée des codes affines-invariants (voir le théorème 4.1).

Par la suite nous notons par $\mathbf{F}$ le corps de décomposition $\mathbb{F}_{2^m}$ de $X^{2^m-1}-1$ sur $\mathbb{F}_2$. Nous considérons pour l'espace ambiant de ces codes l'algèbre du groupe additif $\mathbf{F}$ sur $\mathbb{F}_2$. Cette algèbre sera notée $\mathcal{A} = \mathbb{F}_2[\{\mathbf{F}, +\}]$. Tout élément de $\mathcal{A}$ est une somme formelle :

$$\mathbf{x} = \sum_{g \in \mathbf{F}} x_g X^g, \quad \text{avec} \quad x_g \in \mathbb{F}_2.$$

L'addition et la multiplication par scalaires se font par composantes et la multiplication est donnée par la multiplication dans $\mathbf{F}$ [21, p. 974]. Le décalage d'un mot de code $\mathbf{x} = \sum\limits_{g \in \mathbf{F}} x_g X^g$ est le mot $\sum\limits_{g \in \mathbf{F}} x_g X^{\alpha g}$, où α est une racine primitive (2^m-1)-ième de l'unité.

Considérons l'application $\mathbb{F}_2$-linéaire de $\mathcal{A}$ sur $\mathbf{F}$ suivante :

$$\phi_s\left(\sum_{g \in \mathbf{F}} x_g X^g\right) = \sum_{g \in \mathbf{F}} x_g g^s, \tag{4.6}$$

où $0 \le s \le 2^m-1$ et $0^0 = 1$.

Définition 4.1 *Un code C binaire cyclique de longueur n impaire et d'ensemble de définition I est défini par le système d'équations suivantes :*

$$C = \{\mathbf{x} \in \mathcal{A} \ : \ \phi_s(\mathbf{x}) = 0, \quad \forall s \in I\}. \tag{4.7}$$

Le code dual peut être défini de façon analogue, à partir de l'ensemble de définition

$$I^{\perp} = \{s \in [0,\, n-1] \ : \ \ n - s \notin I\}. \tag{4.8}$$

Dès que $0 \notin I$, on peut étendre le code C en utilisant le bit de parité ; l'ensemble de définition du code étendu EC est $I \cup \{0\}$. Par conséquent,

Définition 4.2 *Le code étendu du code cyclique C est défini par les équations suivantes :*

$$\phi_s(\mathbf{x}) = 0, \quad \forall s \in I \quad et \quad \sum_{g \in \mathbf{F}} x_g g^0 = 0. \tag{4.9}$$

Proposition 4.1 *Dans l'espace ambiant $\mathcal{A} = \mathbb{F}_2[\{\mathbf{F}, +\}]$, soit EC un code cyclique étendu de longueur 2^m et d'ensemble de définition I_{EC}. Le code $EC^{\perp}$ a pour ensemble de définition :*

$$I_{EC^{\perp}} = [0,\, 2^m - 1] \backslash \{-s \ : \ s \in I_{EC}\}.$$

Preuve. L'ensemble I_{EC} est un sous-ensemble de $[0,\, 2^m - 1]$, contenant 0 et invariant sous la multiplication par $2 \pmod{2^m - 1}$. D'après la relation (4.8) on a :

$$I_{EC^{\perp}} \quad = \quad \{s \in [0,\, 2^m - 1] \ : \ 2^m - 1 - s \notin I_{EC}\}. \tag{4.10}$$

Mais $t \notin I_{EC^{\perp}}$ implique $2^m - 1 - t \in I_{EC}$ donc $(-t) \in I_{EC}$. Par conséquent, $I_{EC^{\perp}}$ ne contient pas les opposés (modulo $2^m - 1$) des éléments de I_{EC}. ◇

On conclut que l'ensemble de définition de EC étant connu, on peut facilement déduire celui de $\mathrm{EC}^{\perp}$. Pour simplifier, *l'ensemble de définition de $EC^{\perp}$ sera noté ici par $\mathcal{I}$*. Notons que $0 \in \mathcal{I}$.

Par la suite, nous considérons un code affine-invariant EC de longueur 2^m en forme carrée modifiée et nous donnons une borne supérieure pour la dimension du code$B$. Comme il avait été souligné en Section 2.3, la dimension du code B joue un rôle clef dans la recherche des mots de petit poids dans $EC^{\perp}$. Bien sûr, tous ces résultats sont valables dans le cas des codes $\mathrm{EBCH}^{\perp}$.

Nous rappelons que les codes qui nous intéressent, $EC^{\perp}$, sont mis en *ordre standard des bits*, donc les coordonnées sont indexées par des éléments dans $\mathbb{F}_{2^m}$, qui vérifient la relation (4.3). Pour $\mathbf{x} \in \mathrm{EC}^{\perp}$, $\mathbf{x} = \sum_{g \in \mathbf{F}} x_g X^g$, nous choisissons la notation

$$x_g = \begin{cases} y_g & \text{si } g \in F_0 \\ z_g & \text{si } g \in F_1. \end{cases}$$

Pour tout $g \in F_1$ il existe $b_i \in \mathbb{F}_2$, $0 \le i \le m-2$, tel que $g = \sum_{i=0}^{m-2} b_i \alpha_i + \alpha_{m-1}$. Notons $g' = \sum_{i=0}^{m-2} b_i \alpha_i$, donc

$$g = g' + \alpha_{m-1}. \tag{4.11}$$

Maintenant remarquons que le code $B \oplus B$ est un sous-code de $\mathrm{EC}^{\perp}$ de dimension $2k_B$. En tant que sous-code, nous savons déjà qu'il satisfait les équations du code :

$$\sum_{g \in \mathbf{F}} x_g g^s = 0, \quad \forall s \in \mathcal{I}.$$

Mais la dimension du code $B \oplus B$ étant plus petite en pratique, nous essayons de trouver d'autres propriétés pour ce code. Pour cela nous calculons $\phi_s(\mathbf{x})$ pour $s \notin \mathcal{I}$ et $\mathbf{x} = (\mathbf{y}, \mathbf{z}) \in B \oplus B$:

$$\begin{aligned}
\sum_{g \in \mathbf{F}} x_g g^s &= \sum_{g \in F_0} y_g g^s + \sum_{g \in F_1} z_g g^s = \sum_{g \in F_0} y_g g^s + \sum_{g' \in F_0} z_{g'+\alpha_{m-1}} (g' + \alpha_{m-1})^s \\
&= \sum_{g \in F_0} y_g g^s + \sum_{g' \in F_0} z_{g'+\alpha_{m-1}} \sum_{i \preceq s} \binom{s}{i} g'^i (\alpha_{m-1})^{s-i} \\
&= \sum_{g \in F_0} y_g g^s + \sum_{g' \in F_0} z_{g'+\alpha_{m-1}} \sum_{i \preceq s} g'^i (\alpha_{m-1})^{s-i} \\
&= \sum_{g \in F_0} y_g g^s + \sum_{i \preceq s} (\alpha_{m-1})^{s-i} \sum_{g \in F_0} z_{g+\alpha_{m-1}} g^i \\
&= \sum_{g \in F_0} y_g g^s + \sum_{g \in F_0} z_{g+\alpha_{m-1}} g^s + \sum_{i \prec s} (\alpha_{m-1})^{s-i} \sum_{g \in F_0} z_{g+\alpha_{m-1}} g^i \\
&= \sum_{g \in F_0} (y_g + z_{g+\alpha_{m-1}}) g^s + \sum_{i \prec s} (\alpha_{m-1})^{s-i} \sum_{g \in F_0} z_{g+\alpha_{m-1}} g^i.
\end{aligned}$$

Dans les précédentes équations, $\preceq$ désigne *l'ordre partiel*, défini par

$$(i_0, i_1, \ldots, i_{m-1}) \preceq (s_0, s_1, \ldots, s_{m-1}) \text{ si et seulement si } i_j \leq s_j, \ \forall j \in [0, m-1], \tag{4.12}$$

où $(i_0, i_1, \ldots, i_{m-1})$ et $(s_0, s_1, \ldots, s_{m-1})$ sont les décompositions binaires de i, resp. s. Le théorème de Lucas donne

$$\binom{s}{i} = 1 (\mathrm{mod}\, 2) \quad \text{si et seulement si} \quad i \preceq s.$$

Mais $\mathbf{z} \in B$ implique $(0, \mathbf{z}) \in B \oplus B$, donc il existe un $y' \in B$ tel que $y'_g = z_{g+\alpha_{m-1}}$, $\forall g \in F_0$. Nous concluons par

$$\sum_{g \in \mathbf{F}} x_g g^s = \sum_{g \in F_0} (y_g + z_{g+\alpha_{m-1}}) g^s + \sum_{i \prec s} (\alpha_{m-1})^{s-i} \sum_{g \in F_0} y'_g g^i. \tag{4.13}$$

Proposition 4.2 *La restriction de ϕ_s au code $B \oplus B$ est l'application nulle, pour tout $s \in \mathcal{I}$. De plus, avec la notation*

$$S = \{j \notin \mathcal{I} \ : \ \forall i \prec j \Rightarrow i \in \mathcal{I}\}, \tag{4.14}$$

on a :

$$\phi_s(\mathbf{y}, \mathbf{z}) = \sum_{g \in F_0} (y_g + z_{g+\alpha_{m-1}}) g^s, \tag{4.15}$$

pour chaque $s \in S$ et tout $\mathbf{y}, \mathbf{z} \in B$.

Preuve. Pour $s \in S$, la relation (4.13) peut être écrite de la façon suivante :

$$\sum_{g\in\mathbf{F}} x_g g^s = \sum_{g\in F_0} (y_g + z_{g+\alpha_{m-1}}) g^s + \sum_{i\prec s,\, i\in\mathcal{I}} (\alpha_{m-1})^{s-i} \sum_{g\in F_0} y'_g g^i.$$

Étant donné que $(\mathbf{y}', 0) \in \mathrm{EC}^{\perp}$, on a $\sum_{g\in F_0} y'_g g^i = 0$, $\forall i \in \mathcal{I}$, et (4.15) est prouvée. ◇

Remarque 4.3 Si l'ensemble S contient une seule classe cyclotomique, elle est forcément celle du minimum (en ordre partiel) des non-zéros du code. En fait, si $S = (i)$ et il existe j tel que $j \prec i$, via la définition de S on conclut que $j \in \mathcal{I}$.

Proposition 4.3 *Notons par (B, B) le code de matrice génératrice (G_B, G_B). Alors la restriction de ϕ_s à (B, B) est l'application nulle, pour chaque $s \in \mathcal{I} \cup S$. Par conséquent, la dimension de B, k_B, vérifie*

$$k_B = k_{(B,B)} \leq 2^m - (|\mathcal{I}| + |S|).$$

Preuve. Comme (B, B) est un sous-code de $B \oplus B$, la relation (4.15) est vérifiée. Mais $\mathbf{x} = (\mathbf{y}, \mathbf{z}) \in (B, B)$ implique $y_g = z_{g+\alpha_{m-1}}$ pour chaque $g \in F_0$. ◇

Remarque 4.4 La relation $k_B < (2^m - |\mathcal{I}|)/2$ étant évidente, il s'ensuit que la borne précédente est intéressante seulement quand $2^m - (|\mathcal{I}| + |S|) < (2^m - |\mathcal{I}|)/2$ donc si $|S| > (2^m - |\mathcal{I}|)/2$.

Corollaire 4.1 Supposons que pour un $s \in S$ il existe un seul i $(i \in \mathcal{I})$ tel que $i \prec s$. *D'après la définition de S, cela signifie que $i = 0$ et s est une puissance de 2, $s = 2^n$.* Alors, pour tout $(\mathbf{y}, \mathbf{z}) \in (B, B)$, nous trouvons $\mathbf{y}' = \mathbf{y}$ dans la relation (4.13) et $\phi_s(\mathbf{y}, \mathbf{z}) = (\alpha_{m-1})^{s-i} \sum_{g\in F_0} y_g g^i$. Plus précisément,

$$\phi_{2^n}(\mathbf{y}, \mathbf{z}) = (\alpha_{m-1})^{2^n} \mathrm{wt}(\mathbf{y}), \quad \forall\, (\mathbf{y}, \mathbf{z}) \in (B, B).$$

4.4 Codes EBCH$^{\perp}$ versus codes RM

Dans cette section nous étudions les codes obtenus par la décomposition carrée modifiée des codes EBCH$^{\perp}$, en essayant d'identifier des codes connus. Le noyau de cette section sera la relation entre les codes EBCH$^{\perp}$ et B, d'un part, et les codes de Reed-Muller (RM), d'autre part. Nous commençons par rappeler :

Proposition 4.4 *[70, p. 383] Le poinçonné du code de Reed-Muller $RM(r, m)$ est le code cyclique de longueur $2^m - 1$ dont le polynôme générateur a pour racines l'ensemble des α^i, pour tous les i dont la décomposition binaire est de poids $< m - k$.*

Les deux résultats suivants sont bien connus et sont obtenus facilement par une simple inclusion entre les ensembles de définition. Par la suite, (i) désigne la classe cyclotomique de i.

Proposition 4.5 *Le code BCH binaire primitif au sens strict de longueur $2^m - 1$ et de distance construite δ contient le code de Reed-Muller poinçonné d'ordrer $RM^*(r, m)$ pour tout $\delta \leq 2^{m-r} - 1$.*

Proposition 4.6 *La relation* $EBCH(2^m, \delta) \subset RM(r, m)$ *est vraie pour toute valeur* δ *strictement plus grande que le plus grand représentant de poids* $m - r - 1$ *d'une classe cyclotomique modulo* $(2^m - 1)$.

Preuve. En terme d'ensembles de définition, l'inclusion s'écrit :

$$\{s \ : \ \mathrm{wt}(s) \leq m - r - 1\} \subset \bigcup_{s<\delta}(s).$$

Si un élément est dans l'ensemble de gauche, alors toute sa classe cyclotomique y est. Il suffit de considérer seulement un représentant de la même classe cyclotomique, et que le plus grand de ces représentants soit plus petit que δ. ◇

Le passage aux codes duaux donne *le plus grand entier* r_1 *et le plus petit entier* r_2 tels que :

$$RM(r_1, m) \subseteq EBCH^{\perp}(2^m, \delta) \subset RM(r_2, m), \tag{4.16}$$

pour m et δ fixés.

Remarque 4.5 [56] Tous les codes étant en ordre standard des bits, les inclusions suivantes sont vérifiées :

$$RM(r,\, m) \quad \subseteq \quad EBCH(2^m,\, 2^{m-r} - 1), \quad \text{pour} \quad 1 \leq r < m \tag{4.17}$$

$$RM(r,\, m) \quad \subseteq \quad EBCH^{\perp}(2^m,\, q(m, r)), \tag{4.18}$$

pour une fonction $q(m, r)$ telle que $q(m,\, 1) = 5$, $q(m,\, 2) = 2^{\lfloor m/2 \rfloor} + 3$.

Remarque 4.6 On peut améliorer la borne de Schaub [1] pour un des codes étudiés en le comparant avec un code de RM via la proposition 4.5. Ainsi, on peut facilement prouver que tout code EBCH de distance construite $\delta \leq 31$ contient le code RM(3, 8) ; par conséquent, le code EBCH$^{\perp}$(256, 108) est contenu dans le code RM(4, 8). La borne de Schaub donne 26, mais le code RM(4, 8) ne contient pas des mots de poids 26. Ainsi la borne monte à 28 (voir Annexe A.1) .

Dans le tableau suivant nous avons listé les valeurs de δ pour lesquelles $RM(r, m)$ est le plus grand RM inclus dans $EBCH^{\perp}(2^m, \delta)$ pour un r fixé et $m \in \{7, 8, 9\}$.

RM(r, m)	δ du BCH	RM(r, m)	δ du BCH
RM$(1, 7)$	$3 \leq \delta \leq 9$	RM$(5, 8)$	$95 \leq \delta \leq 119$
RM$(2, 7)$	$11 \leq \delta \leq 21$	RM$(6, 8)$	$\delta = 127$
RM$(3, 7)$	$23 \leq \delta \leq 43$	RM$(1, 9)$	$3 \leq \delta \leq 17$
RM$(4, 7)$	$47 \leq \delta \leq 55$	RM$(2, 9)$	$19 \leq \delta \leq 73$
RM$(5, 7)$	$\delta = 63$	RM$(3, 9)$	$75 \leq \delta \leq 85$
RM$(1, 8)$	$3 \leq \delta \leq 17$	RM$(4, 9)$	$87 \leq \delta \leq 171$
RM$(2, 8)$	$19 \leq \delta \leq 37$	RM$(5, 9)$	$175 \leq \delta \leq 219$
RM$(3, 8)$	$39 \leq \delta \leq 85$	RM$(6, 9)$	$223 \leq \delta \leq 239$
RM$(4, 8)$	$87 \leq \delta \leq 91$	RM$(7, 9)$	$\delta = 255$

Sachant que tous les codes en (4.16) sont en ordre standard des bits, nous les mettons en forme carrée modifiée. Rappelons la relation (2.8) :

$$\begin{aligned} G(r,m) &= \begin{bmatrix} G(r,m-1) & G(r,m-1) \\ 0 & G(r-1,m-1) \end{bmatrix} \\ &= \begin{bmatrix} \Delta(r/(r-1),m-1) & \Delta(r/(r-1),m-1) \\ G(r-1,m-1) & 0 \\ 0 & G(r-1,m-1) \end{bmatrix}. \end{aligned}$$

Considérant les codes raccourcis sur les 2^{m-1} premières coordonnées, nous déduisons :

$$RM(r_1-1,\, m-1) \subseteq B(2^m,\delta) \subseteq RM(r_2-1,\, m-1), \tag{4.19}$$

où $B(2^m,\delta)$ désigne le code B obtenu par la décomposition de EBCH$^{\perp}(2^m,\delta)$.

Maintenant nous nous intéressons au calcul exact du code B en remarquant tout d'abord que parfois la décomposition du code $EBCH(2^m,\delta)$ donne un code RM pour le code B. Nous essayons donc de trouver toutes les valeurs de δ pour lesquelles

$$B(2^m,\delta) = RM(r-1,\, m-1). \tag{4.20}$$

Si m est fixé, une méthode consiste en la décomposition de tous les EBCH$^{\perp}(2^m,\delta)$ et le calcul de la dimension (et la distance minimale) de B. Si nous obtenons les paramètres d'un code Reed-Muller, nous vérifions directement l'égalité (par exemple avec Magma), car les deux codes sont tous les deux en ordre standard des bits. Nous présentons dans le tableau suivant toutes les valeurs de δ pour lesquelles la relation (4.20) est vraie (aussi à une ligne de la matrice génératrice près), quand $m \in \{5,6,7,8,9\}$.

B	2^m, δ	B	2^m, δ
RM$(0,4)$	32, $\{3,5\}$	RM$(3,6)+[64,1,12]$	128, 47
RM$(1,4)$	32, 7	RM$(4,6)$	128, 63
RM$(2,4)$	32, 15	RM$(0,7)$	256, $\{3,5,7,9\}$
RM$(0,5)$	64, $\{3,5\}$	RM$(1,7)$	256, $\{19,21\}$
RM$(1,5)$	64, 11	RM$(2,7)$	256, 39
RM$(2,5)+[32,1,8]$	64, 23	RM$(4,7)+[128,1,16]$	256, 95
RM$(3,5)$	64, 31	RM$(5,7)$	256, 127
RM$(0,6)$	128, $\{3,5,7,9\}$	RM$(0,8)$	$512, 3 \le \delta \le 17$
RM$(1,6)$	128, $\{11,13\}$	RM$(1,8)$	512, $\{19, 21, 23, 25\}$
RM$(1,6)+[64,1,28]$	128, $\{15,19\}$	RM$(1,8)+[256,1,128]$	512, $\{27, 29, 31, 35\}$
RM$(2,6)$	128, $\{23,27\}$	RM$(6,8)$	512, 255

Remarque importante. Pour r et m fixés, on peut facilement remarquer que le plus petit δ dans le tableau (s'il existe), est le minimum des δ plus grands que le plus grand représentant de poids r dans une classe cyclotomique modulo (2^m-1) (via la Proposition 4.6). □

Les résultats numériques indiquent que dans la relation (4.19), l'ordre r_1-1 est respecté, mais l'ordre r_2-1 peut être encore réduit dans plusieurs cas. Cela suggère de considérer qu'il suffit

de chercher les cas pour lesquels l'inclusion

$$RM(r_1 - 1,\, m-1) \subseteq B(2^m, \delta) \subseteq RM(r_1,\, m-1) \tag{4.21}$$

est vérifiée. Dès qu'un δ qui vérifie (4.20) est trouvé, on doit aussi essayer les distances construites plus grandes que cette valeur, car dans plusieurs cas le code B est le même pour différents $EBCH^{\perp}(2^m, \delta)$ (voir les résultats numériques en Section A.2).

Par la suite, nous essayons d'appliquer la borne donnée par la proposition 4.3. Nous allons donc nous intéresser de plus près au code (B, B).

Remarque 4.7 Supposons qu'on arrive à montrer une relation du type $(B, B) \subset RM(r, m)$, pour un certain r. Alors l'espace engendré par les lignes de (G_B, G_B) est un sous-espace de l'espace engendré par le lignes de la matrice

$$\begin{pmatrix} G(r, m-1) & G(r, m-1) \\ 0 & G(r-1, m-1) \end{pmatrix}$$

et finalement $B \subseteq RM(r, m-1)$. Le passage de (B, B) à B, réduit donc par conséquent seulement la longueur et non pas l'ordre.

1. pour $\delta \in \{3,\, 2^{m-1}\}$ évidemment,

$$B(2^m, 3) = RM(0, m-1) \quad \text{et} \quad B(2^m, 2^{m-1} - 1) = RM(m-3, m-1);$$

2. pour $\delta = 5$, l'ensemble de définition de $\mathrm{EBCH}^{\perp}(2^m, 5)$ est $[0, 2^m - 1] \backslash \{(-1) \cup (-3)\}$. Mais $-1 = (0111 \ldots 1)$, $-3 = (0011 \ldots 1)$, donc $S = (-3)$. Maintenant (B, B) est un sous-code du code d'ensemble de définition $[0, 2^m - 1] \backslash (-1)$ qui est précisément formé par tous les entiers de poids $< m - 1$, donc $\mathrm{RM}(1, m)$. Cela implique $B \subseteq RM(1, m-1)$. Les calculs donnent $B = RM(0, m-1)$ pour $m \geq 5$.
3. pour $\delta = 7$, l'ensemble de définition de $\mathrm{EBCH}^{\perp}(2^m, 7)$ est $[0, 2^m - 1] \backslash \{(-1) \cup (-3) \cup (-5)\}$. Mais $-1 = (0111 \ldots 1)$, $-3 = (0011 \ldots 1)$, $-5 = (0101 \ldots 1)$, donc $S = (-3) \cup (-5)$ et par conséquent (B, B) est un sous-code du code d'ensemble de définition formé par les entiers de poids $< m - 1$, donc $\mathrm{RM}(1, m)$. Les calculs donnent $B = RM(1, m-1)$ pour $m = 5$ et $B = RM(0, m-1)$ si $m \geq 7$.

Comme nous avons déjà vu dans la section précédente, il est très important que le cardinal de l'ensemble S soit le plus grand possible. Un tel cas est $m = 8$ et $\delta = 19$. L'ensemble de non-zéros du code $\mathrm{EBCH}^{\perp}(256, 19)$ est

$$(15) \cup (31) \cup (47) \cup (61) \cup (63) \cup (95) \cup (111) \cup (119) \cup (127).$$

Sachant que les décompositions binaires sont $15 = (11110000)$, $31 = (11111000)$, $47 = (11110100)$, $61 = (10111100)$, $63 = (11111100)$, $95 = (11111010)$, $111 = (11110110)$, $119 = (11101110)$, $127 = (11111110)$, alors

$$15 \prec 31,\ 15 \prec 47,\ 15 \prec 63,\ 15 \prec 95,\ 15 \prec 111,\ 15 \prec 127$$

et finalement

$$S = (15) \cup (61) \cup (119).$$

Maintenant $I \cup S$ contient tous les entiers de poids ≤ 4 et on déduit

$$(B, B) \subset RM(8-4-1, 8), \quad B \subset RM(3, 7),$$

mais en fait $B = RM(1, 7)$ (voir les précédents tableaux).

Problème ouvert : Trouver tous les codes EBCH$^{\perp}$ pour lesquels le code B est un code RM.

Une autre possibilité pour montrer qu'un code B est un RM est tout simplement de montrer qu'il a la dimension du plus grand RM qu'il contient (voir les précisions précédentes). Pour cela le calcul de la dimension de A est suffisant — voir la relation (2.12). Les codes A sont LTSC et, vu les paramètres de certains d'entre eux, on pourrait penser à une relation avec les duaux des codes EBCH de même longueur. Comme exemple, si $A(2^m, \delta)$ désigne le code A construit via la décomposition LTSC de EBCH$^{\perp}(2^m, \delta)$, nous obtenons que les codes $A(128, 9)$ et EBCH$^{\perp}(64, 11)$ (tous les deux en forme LTSC) sont deux codes $[64, 28, 14]$ qui ne sont pas égaux mais :

$$A(128, 9) \cap EBCH^{\perp}(64, 11) = A(128, 7).$$

Problème ouvert : Est-ce qu'on peut obtenir une relation analogue pour d'autres longueurs et pour d'autres distances construites δ ?

4.5 Résultats numériques

Dans cette section nous analysons les résultats numériques, qui ont été obtenus via l'algorithme Canteaut-Chabaud et la décomposition en forme carrée modifiée des codes EBCH$^{\perp}$.

Les bornes supérieures obtenues en utilisant l'algorithme probabiliste sont marquées par $\mathcal{A}$ en Annexe A.1. Pour quelques codes, nous n'avons pas obtenu $d^{\perp}_{\max}$ par l'application directe de l'algorithme, mais en exploitant la monotonie de la distance minimale (Lemme 4.1). Comme exemple, dès qu'un mot de code de poids 64 a été trouvé dans EBCH$^{\perp}$(512, 37) et dans EBCH$^{\perp}$(512, 57), alors pour tous les codes EBCH$^{\perp}$(512, δ), $37 \leq \delta \leq 57$, la valeur attendue pour $d^{\perp}_{\max}$ est 64. Le même argument conduit à la conclusion que $d^{\perp}_{\max} = 32$ pour δ tel que $87 \leq \delta \leq 107$.

Quelques contradictions dans les résultats apparaissent dans le cas $n \geq 512$, si k est près de $n/2$, car pour toute longueur fixée la complexité de l'algorithme est maximale quand $k = n/2$. Ainsi, dès qu'on trouve un mot de poids 64 dans EBCH$^{\perp}(512, 37)$, nous avons prouvé (via le Lemme 4.1) l'existence des mots de poids plus petits ou égaux à 64 dans tout EBCH$^{\perp}(512, \delta)$ avec $\delta \geq 37$. Cependant, nous ne pouvons pas obtenir directement un tel mot de code dans tous les cas ; par exemple, pour $\delta = 57$ et $\delta = 73$ seulement un mot de poids 82, resp. 66, a été trouvé. Ces valeurs sont gardées dans nos tableaux, même si elles sont affinées par la LTSC (via la borne en (4.1)).

En Annexe A.2 nous listons les décompositions LTSC de tous les codes EBCH$^{\perp}$ de longueur $32 \leq n \leq 512$, en particulier des bornes nouvelles pour les distances duales de ces codes en longueur 512. Les paramètres dans les tableaux sont :

1. δ, la distance construite du code BCH ;
2. k, la dimension du code EBCH$^\perp$ correspondant ;
3. $d^\perp_{\max}$ désigne une borne supérieure de la distance duale (en *italique*) ou la distance duale (obtenue dans tous les cas en utilisant Magma [71]) ; *quand on donne deux valeurs pour $d^\perp_{max}$, la première (obtenue par la LTSC), affine la deuxième (obtenue par l'algorithme probabiliste).*
4. A et B sont les codes obtenus par la décomposition LTSC du code EBCH$^\perp(n, \delta)$ *en ordre standard des bits*;
5. $(A/B, \widetilde{A/B})$ est le code obtenu en considérant le complément de EBCH$^\perp$ par la somme directe $B \oplus B$;
6. l'avant-dernière colonne dans nos tables indique le nombre de colonnes nulles dans une matrice génératrice du code $(A/B, \widetilde{A/B})$;
7. la dernière colonne donne l'ordre du plus grand code RM contenu dans EBCH$^\perp(n, \delta)$.

Considérons la construction LTSC des codes $EBCH^\perp(n, \delta_1)$ et $EBCH^\perp(n, \delta_2)$, qui satisfont

$$\delta_1 < \delta_2 \Rightarrow EBCH^\perp(n, \delta_1) \subset EBCH^\perp(n, \delta_2). \tag{4.22}$$

Ces codes sont mis par la suite en *ordre standard des bits* par la même permutation, et l'inclusion est encore vérifiée. De plus, on peut facilement déduire des inclusions similaires pour les codes correspondants A et B, via leur définition. Utilisant cette monotonie, nous soulignons au passage que tous les codes A (resp. B) de paramètres identiques sont *égaux*.

Remarque 4.8 Les paramètres des codes composants sont obtenus par Magma. Seulement les valeurs en *italique* sont des bornes supérieures et non des distances minimales exactes. Concernant les codes $(A/B, \widetilde{A/B})$, les bornes plus fines (en deuxième position dans la sixième colonne) sont obtenues en appliquant l'algorithme probabiliste ; de plus pour ces codes, une inclusion du type (4.22) n'étant pas envisageable, on doit tester les codes un par un pour évaluer leur distance minimale.

Problème ouvert : Vu les estimations des distances minimales des codes $(A/B, \widetilde{A/B})$ dans l'Annexe A.2, peut-on montrer que leur distance minimale est une fonction décroissante ?

Un autre aspect important est que pour plusieurs codes EBCH$^\perp$, les codes A ou les codes B sont égaux. Une possible explication de ce résultat est donnée par le raisonnement suivant. Supposons que pour une longueur 2^m et un certain δ la décomposition en forme LTSC du code EBCH$^\perp$ est connue. Nous voulons maintenant en déduire la décomposition du code "suivant" (par rapport à δ). Il s'agit juste d'ajouter un bloc de m lignes à la matrice génératrice et d'essayer d'établir si elles vont s'ajouter à la somme directe de $B \oplus B$ et/ou au code complément. En analysant les paramètres des codes B donnés en Annexe A.2, on remarque que dans la plupart des cas c'est *soit* la somme directe, *soit* le code complément qui change de dimension. Dans ces cas, une relation de récurrence sur les dimensions peut être déduite. Il reste à identifier les cas où les deux dimensions changent, de quelle valeur, et de déterminer comment cela dépend des deux δ impliqués.

Remarque importante. Pour tous les codes de longueur 256 pour lesquels on donne en [90] seulement une borne supérieure, nous avons trouvé un mot de poids $d^{\perp}_{\max}$ dans le code $(A/B, \widetilde{A/B})$. De plus, dans tous les cas pour $n \leq 128$ et pour presque tous les cas si $n = 512$, un mot de ce poids a été trouvé dans un des codes $B \oplus B$ (ou tout simplement B) et $(A/B, \widetilde{A/B})$. Ainsi, *ces codes sont potentiellement des sous-codes de poids minimal*

Cependant, il y a aussi des exceptions. Pour $\mathrm{EBCH}^{\perp}(512, 175)$ nous n'avons pas pu trouver un mot de poids 14 en B, donc on conclut qu'il doit se trouver dans un translaté de $B \oplus B$ par rapport au code complément. Deux autres exemples sont $\delta = 117$ et $\delta = 123$. Par contre, pour $\delta = 35$ nous avons trouvé un mot de poids 106 dans le code complément, affinant ainsi la valeur obtenue par l'algorithme de recherche de poids minimum. □

Problème ouvert : Étudier si les codes B et $(A/B, \widetilde{A/B})$ contiennent des mots de petit poids si $n > 512$ et pour d'autres classes de codes ; de plus, analyser si ce résultat dépend de l'ordre standard des bits.

L'ordre du code de Reed-Muller dans la dernière colonne est déduit de la Proposition 4.6 en passant aux codes duaux. Nous insistons encore une fois sur le fait qu'il faut commencer à chercher un code RM dans la cinquième colonne à partir de la ligne qui correspond au changement de l'ordre du code RM dans la dernière colonne.

Les codes RM contenus dans les codes $\mathrm{EBCH}^{\perp}$ donnent des bornes sur les distances minimales. Très fines en longueur 256, elles ne le sont plus en longueur 512, où la distance minimale décroît très vite. Par exemple, en longueur 256 pour $19 \leq \delta \leq 37$ on a $d \leq 64$, et on estime $36 \leq d \leq 60$. Pour $\delta \in \{39, 43, 45\}$, on obtient (via les RM) $d \leq 32$ et nos calculs donnent pour $\delta = 39$ exactement 32.

En longueur 512, pour $55 \leq \delta \leq 73$, on obtient $56 \leq d \leq 64$, tandis que les codes RM donnent 128. De même, pour $75 \leq \delta \leq 85$, on obtient $d \in \{48, 56\}$, tandis que les codes RM donnent la valeur 64 trouvée bien avant ! On suppose que cela vient du fait que les codes B sont beaucoup plus grands que les codes RM, même si c'est plutôt dans le code complément $(A/B, \widetilde{A/B})$ qu'il faudra chercher les mots de poids minimal. Cette observation est renforcée aussi par l'exemple $\delta = 87$: le code B est plus "près" de $\mathrm{RM}(3, 8)$ et on atteint 32 pour B, ainsi que pour $\mathrm{EBCH}^{\perp}$.

4.6 Autour des codes EBCH

Revenons maintenant à l'article de Desaki *et al.* [26] sur les distributions des poids des codes BCH de longueur 128. À partir des résultats numériques, ils ont déduit le résultat suivant :

Théorème 4.2 *Les codes EBCH*$(128, k)$ *et EBCH*$(128, 128-k)^{\perp}$ *ont la même distribution des poids, pour tout* $k \in \{29, 36, 43, 64, 85, 92, 99\}$. *Plus particulièrement, le code EBCH*$(128, 64, 22)$ *est formellement auto-dual (c'est-à-dire le code et son dual ont la même distribution des poids tout en étant différents).*

Remarque 4.9 En utilisant les distributions des poids des autres codes EBCH de longueur 128, nous avons pu calculer les distributions des codes duaux, via la transformée de MacWilliams. Nous avons ainsi trouvé $\mathrm{EBCH}(128, 8) = EBCH(128, 120)^{\perp}$, resp. $\mathrm{EBCH}(128, 15)$ et

EBCH(128, 113)$^\perp$, ont la même distribution des poids, ainsi que EBCH(128, 22), EBCH(128, 106)$^\perp$. Alors le résultat du Théorème 4.2 reste valable pour tout k.

Nous nous sommes intéressés à une justification théorique de ce résultat et à des possibles généralisations, tout cela à travers la construction LTSC pour les codes EBCH (pour plus de détails voir [75]). Pour ce qui est des généralisations par rapport à la longueur, on pourra avoir des propriétés similaires seulement dans le cas m impair [23].

Un cas très facile à étudier est $m = 5$, donc des EBCH de longueur 32. On trouve EBCH(32, 16, 8) qui est *auto-dual*. Rappelons que dans ce cas nous avons EBCH(32, 16, 8)=RM(2, 5) De plus, EBCH(32, 26, 4) =EBCH(32, 6, 16)$^\perp$ et les codes EBCH(32, 21, 6), EBCH (32, 11, 10)$^\perp$ ont la même distribution des poids, donc la propriété décrite par le Théorème 4.2 reste valable pour $n = 32$.

Examinons maintenant de plus près ces codes, en les mettant en forme LTSC. Les paramètres de leurs constructions sont indiqués en Annexe A.2. Pour le code EBCH(32, 16, 8) on obtient $A/B = \widetilde{A/B}$ (ce qui amène à une construction carrée pure) et $A = B^\perp$, donc le code est auto-dual. Pour les codes égaux EBCH(32, 26, 4) =EBCH(32, 6, 16)$^\perp$, les codes A sont égaux, ainsi que les codes B. Pour la paire des codes qui ont la même distribution des poids, les codes A sont égaux, les codes B sont équivalents et, par conséquent, les codes A/B sont équivalents. Mais les codes $A/B \cup \widetilde{A/B}$ ne le sont pas, même s'ils ont la même distribution des poids.

Passons au code formellement auto-dual EBCH(128, 64, 22). On commence par décomposer en forme LTSC le code EBCH(128, 64, 22)$^\perp$; sa matrice génératrice est de la forme :

$$\begin{bmatrix} G_{A/B} & \widetilde{G_{A/B}} \\ G_B & 0 \\ 0 & G_B \end{bmatrix}.$$

Par la suite la matrice génératrice du code EBCH(128, 64, 22) est (d'après le Théorème 2.1) :

$$\begin{bmatrix} G_{B^\perp/A^\perp} & \widetilde{G_{B^\perp/A^\perp}} \\ G_{A^\perp} & 0 \\ 0 & G_{A^\perp} \end{bmatrix}.$$

On veut comparer les grands codes à partir des codes composants, par exemple voir s'ils sont équivalents ou faire apparaître des propriétés qui puissent justifier l'égalité des polynômes énumérateurs. Nous avons notamment obtenu que les codes B et $A^\perp$ sont de paramètres $[64, 15, 24]$, ne sont ni égaux, ni équivalents, mais ont le même polynôme énumérateur :

$$x^{64} + 2352x^{40}y^{24} + 6912x^{36}y^{28} + 14238x^{32}y^{32} + 6912x^{28}y^{36} + 2352x^{40}y^{24} + y^{64}.$$

On conclut que le code A contient un sous-code B de même polynôme énumérateur que $A^\perp$, mais il n'est pas auto-orthogonal (c'est-à-dire inclus dans son dual). Une idée pour approfondir l'étude est de passer au degré de décomposition suivant, donc d'utiliser la récursivité de la construction carrée modifiée.

4.7 Conclusion

Dans ce chapitre, nous donnons des bornes supérieures nouvelles pour les distances duales des codes EBCH binaires de longueur 512. Deux sous-codes sont d'une importance particulière, car ils contiennent très souvent des mots de poids minimum (sauf en trois cas si $n \leq 512$). Ces bornes ont été obtenues avec un algorithme probabiliste, confirmées et même affinées en certains cas par la décomposition carrée modifiée (LTSC).

La LTSC est particulièrement intéressante pour des codes de grande longueur ; notamment elle est facile à construire pour tous les codes EBCH et leurs duaux. Par la suite nous sommes amenés à travailler avec des codes de longueur et dimension réduites (voir *Remarque importante* en Section 4.2). Ainsi, nous proposons quelques pistes pour une approche théorique. Les recherches à venir pourront s'orienter vers la caractérisation des codes A et B, leurs propriétés algébriques et les liens avec d'autres codes affines-invariants.

Troisième partie

Énumérateurs des poids

5

Codes cycliques auto-duaux

Les codes cycliques sont bien connus dans la théorie des codes correcteurs d'erreurs. Nous pouvons citer, par exemple, les codes Bose-Chaudhuri-Hocquenghem (BCH) et les codes résidus quadratiques (QR), qui sont des codes de référence. Pour tous ces codes, définis sur un corps fini $\mathbb{F}$, la longueur n du code et la caractéristique p du corps $\mathbb{F}$ vérifient $p.g.c.d.\,(n,\, p) = 1$. Par contre, pour les codes cycliques *à racines multiples* de longueur n sur $\mathbb{F}$, n est divisible par p. Même si Castagnoli *et al.* ont prouvé en [24] que ces codes ne peuvent pas être meilleurs que les codes cycliques à racines simples, les codes cycliques à racines multiples restent des objets intéressants.

Dans ce chapitre, nous considérons des codes cycliques *binaires* à racines multiples. Sloane et Thompson ont introduit la classe des codes cycliques auto-duaux (à racines multiples) [93]. Notamment, ils ont prouvé que ces codes sont de Type I (voir définition 1.5). Par ailleurs, van Lint a montré que les codes cycliques à racines multiples peuvent être obtenus via la construction dite $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ [65]. Le travail décrit dans ce chapitre est basé sur ces résultats.

Nous utilisons les résultats de van Lint en [65] pour obtenir une *construction carrée* pour *tous* les codes cycliques *binaires* à racines multiples. Nous montrons aussi que la construction $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ joue un rôle clef pour la réduction de la complexité dans le calcul des polynômes énumérateurs. Concernant les codes cycliques auto-duaux (*c.a.d.*), leurs polynômes générateurs sont explicités dans [93]. Mais ces codes étant de Type I [93, Th. 1], nous pouvons calculer certains polynômes énumérateurs des codes *c.a.d.* en utilisant leur code *ombre* [25].

Le chapitre est organisé de la façon suivante. En Section 5.1 nous présentons les notations spécifiques à ce chapitre et rappelons quelques résultats de base. Nous analysons la matrice génératrice des codes cycliques à racines multiples en Section 5.2. Nous généralisons d'abord les résultats de van Lint. Nous déduisons par la suite la construction carrée en Section 5.2.2 ; finalement nous étudions la classe des polynômes générateurs qui sont des polynômes carrés (théorème 5.5).

Dans la Section 5.3, nous restreignons l'étude aux cas des codes *c.a.d.* et particulièrement au calcul de leurs polynômes énumérateurs. Notamment, nous décrivons l'ensemble de ces codes pour des longueurs allant jusqu'à 120. Ces codes n'étant pas tous inéquivalents, nous allons énumérer seulement les représentants des classes d'équivalence. Nous proposons aussi une expression simple

pour les polynômes énumérateurs des codes donnés par une construction carrée (voir théorème 5.8), que nous appliquons par la suite à deux classes spéciales de codes *c.a.d.* (Section 5.3.3). Nous présentons ensuite la méthode de *l'ombre du code*, utilisée pour le calcul des polynômes énumérateurs des codes *c.a.d.* Finalement, nous comparons les différentes méthodes utilisées pour le calcul de ces polynômes ; ainsi, nous avons pu calculer les polynômes énumérateurs de tous les codes *c.a.d.* de longueurs inférieures ou égales à 120, à l'exception de deux codes de longueur 112 et de deux codes de longueur 120. Les résultats numériques sont résumés dans la Section B.

5.1 Préliminaires

Considérons un code *cyclique* C de longueur n et de polynôme générateur $g(x)$. Le *polynôme réciproque* de g sera noté par $\overleftarrow{g(x)}$:

$$\overleftarrow{g(x)} = x^{\deg g} g(x^{-1}).$$

Le code dual $C^{\perp}$ de C est par conséquent un code cyclique de polynôme générateur $(x^n+1)/\overleftarrow{g(x)}$. Alors le polynôme générateur d'un code cyclique auto-dual (*c.a.d.*) de longueur n vérifie :

$$g(x)\overleftarrow{g(x)} = x^n + 1. \tag{5.1}$$

Évidemment, pour tout entier positif n il existe $a \geq 0$ et b impair tels que $n = 2^a b$, et par conséquent

$$x^n + 1 = (x^b + 1)^{2^a}.$$

Définition 5.1 *Soit s un entier tel que $0 \leq s < b$ et soit k le plus petit entier positif tel que $s2^k \equiv s \pmod b$. Notons par α une racine primitive b-ième de l'unité. Le* polynôme minimal *de α^s sur $\mathbb{F}_2$ est*

$$M_s^{(b)}(x) = \prod_{i \in C_s^{(b)}} (x - \alpha^i),$$

où

$$C_s^{(b)} = \{s, 2s, \ldots, 2^{k-1}s\} \pmod b$$

est la classe cyclotomique de s modulo b.

Alors la relation (5.1) devient

$$g(x)\overleftarrow{g(x)} = (x^b + 1)^{2^a} = \left[\prod_{C_s^{(b)}} M_s^{(b)}(x)\right]^{2^a}, \tag{5.2}$$

où chaque terme dans le produit correspond à une classe cyclotomique et apparaît 2^a fois. Par la suite nous allons déduire une formule pour le polynôme générateur des codes *c.a.d.* et prouver le résultat suivant :

Théorème 5.1 *Les codes cycliques auto-duaux sont des codes cycliques à racines multiples.*

Nous rappelons que :

Définition 5.2 Un code cyclique binaire à racines multiples C *est un code cyclique de longueur* $n = 2^a b$ *sur* $\mathbb{F}_2$, *où* $a \geq 1$ *et* b *impair. L'*ensemble de définition *de* C *est l'ensemble de toutes les racines du polynôme générateur, considérées* avec leurs multiplicités.

Définition 5.3 *[93] Une classe cyclotomique est appelée* symétrique *si* $-s \in C_s^{(b)}$ *et* asymétrique *autrement. L'ensemble des classes asymétriques se divise en*paires asymétriques $(C_s^{(b)}, C_{-s}^{(b)})$.

Preuve du théorème 5.1. Le polynôme générateur de tout code *c.a.d.* satisfait (5.2), plus précisément

$$\begin{aligned} g(x)\overleftarrow{g(x)} &= \prod_{C_s^{(b)}\text{sym.}} M_s^{(b)}(x)^{2^a} \prod_{C_s^{(b)}\text{asym.}} M_s^{(b)}(x)^{2^a} \\ &= \prod_{C_s^{(b)}\text{sym.}} M_s^{(b)}(x)^{2^a} \prod_{\text{paires asym.}} M_s^{(b)}(x)^{2^a} M_{-s}^{(b)}(x)^{2^a}. \end{aligned} \tag{5.3}$$

Soit g de la forme

$$g(x) = \prod_{C_s^{(b)}\text{sym.}} M_s^{(b)}(x)^{i_s} \prod_{\text{paires asym.}} M_s^{(b)}(x)^{i_s}\ M_{-s}^{(b)}(x)^{i_{-s}}, \tag{5.4}$$

où les i_s sont des entiers dans l'intervalle $[0, 2^a]$. Par la suite on veut déduire les conditions sur i_s pour que g de la forme (5.4) satisfasse (5.3).

Il est facile de prouver que si $C_s^{(b)}$ est symétrique (donc $C_s^{(b)} = C_{-s}^{(b)}$), alors

$$\overleftarrow{M_s^{(b)}(x)} = M_s^{(b)}(x),$$

tandis que si $C_s^{(b)}$ et $C_{-s}^{(b)}$ forment une paire asymétrique on a :

$$\overleftarrow{M_s^{(b)}(x)} = M_{-s}^{(b)}(x).$$

Donc, suite à (5.4) on a :

$$\overleftarrow{g(x)} = \prod_{C_s^{(b)}\text{sym.}} M_s^{(b)}(x)^{i_s} \prod_{\text{paires asym.}} M_{-s}^{(b)}(x)^{i_s}\ M_s^{(b)}(x)^{i_{-s}}. \tag{5.5}$$

Remplaçant g et $\overleftarrow{g}$ dans (5.3) par (5.4) et (5.5) on déduit :

$$\begin{cases} 2i_s = 2^a & \text{si } C_s^{(b)} \text{ symétrique} \\ i_s + i_{-s} = 2^a & \text{si } (C_s^{(b)}, C_{-s}^{(b)}) \text{ paire asymétrique.} \end{cases} \tag{5.6}$$

En conclusion

$$g(x) = \prod_{C_s^{(b)}\text{sym.}} [M_s^{(b)}(x)]^{2^{a-1}} \prod_{\text{paires asym.}} [M_s^{(b)}(x)]^{i_s}\ [M_{-s}^{(b)}(x)]^{2^a - i_s}, \tag{5.7}$$

où à chaque terme du premier produit correspond une classe cyclotomique$C_s^{(b)}$, à chaque terme du deuxième produit correspond une paire asymétrique $(C_s^{(b)}, C_{-s}^{(b)})$ et i_s est un entier dans l'intervalle $[0, 2^a]$.

La formule (5.7) implique $a \geq 1$ et, via la définition 5.2, on obtient la conclusion. ◇

Dans la section suivante nous allons étudier les codes à racines multiples via leurs polynômes générateurs, pour appliquer par la suite ces résultats à la sous-classe des codes *c.a.d.*

Le polynôme générateur g d'un code cyclique à racines multiples C de longueur $2^a b$ admet une décomposition *unique*

$$g = g_1^{2^a} g_2^{2^a-1} \dots g_{2^a}. \tag{5.8}$$

Les polynômes g_i, $1 \leq i \leq 2^a$, qui ne sont pas égaux à 1 sont *des diviseurs premiers entre eux* de $x^b + 1$ – ce qui implique l'unicité. Notons que les g_i sont sans facteurs carrés, car b est impair.

En utilisant cette décomposition et les constructions introduites dans la Section 2.1, nous obtenons quelques codes particuliers pour lesquels nous adopterons les **notations** suivantes :

- C^i le code cyclique de polynôme générateur $g_1 \dots g_i$;
- $C^{i,j}$ la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de C^i et C^j ;
- $C^{i/j} = C^i/C^j$ pour $i < j$, plus précisément l'ensemble des représentants des translatés de C^j dans C^i ;
- $C^{i,\ell}$ la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de $C^{i,j}$ et $C^{k,\ell}$, pour $i < j < k < \ell$;
- G^i, $G^{i,j}$, $G^{i/j}$, resp. $G^{i,\ell}$ sont les notations similaires pour leurs matrices génératrices.

5.2 Résultats généraux

Dans cette section, nous allons introduire une nouvelle caractérisation pour la matrice génératrice des codes cycliques à racines multiples, basée sur la construction $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ itérative.

5.2.1 Sur la construction $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$

En [65], van Lint a proposé de considérer, *à équivalence près*, les codes cycliques à racines multiples via la construction $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$. Il a montré le théorème suivant dans le cas $n = 2b$, b impair, et a indiqué que ce résultat peut être généralisé pour les longueurs $n = 2^a b$, $a \geq 2$.

Théorème 5.2 *[65] Soit b un entier impair et g_1, g_2 deux diviseurs premiers entre eux de x^b+1. Considérons C^1 le code cyclique binaire de longueur b et générateur g_1 et C^2 le code cyclique binaire de longueur b et générateur $g_1 g_2$. Alors, le code cyclique binaire C de longueur $2b$ et générateur $g_1^2 g_2$ est* équivalent *à la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de C^1 et C^2.*

Maintenant nous allons exposer précisément la généralisation du théorème 5.2. Nous considérons un code cyclique à racines multiples C de longueur $2^a b$ ($a \geq 2$ et b impair) et de polynôme générateur g.

1. Préliminaires :
   - Soit $g_1^{2^a} g_2^{2^a-1} \dots g_{2^a}$ l'unique décomposition de g telle que les polynômes g_i, $i \in [1, 2^a]$, sont des diviseurs premiers entre eux de $x^b + 1$.
   - Pour chaque entier i dans l'intervalle $[1, 2^a]$, on construit le code cyclique binaire de longueur b et générateur $g_1 \dots g_i$, noté C^i.
2. Étapes de la construction :

- *Premier pas :* Pour chaque entier $i \in [1, 2^{a-1}]$, nous construisons la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de C^{2i-1} et C^{2i}, notée $C^{2i-1,\,2i}$.
- *Deuxième pas :* Pour chaque $i \in [1, 2^{a-2}]$, nous construisons la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de $C^{4i-3,\,4i-2}$ et $C^{4i-1,\,4i}$, notée $C^{4i-3,\,4i}$.
- *Troisième pas :* Pour chaque $i \in [1, 2^{a-3}]$, nous construisons la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de $C^{8i-7,\,8i-4}$ et $C^{8i-3,\,8i}$, notée $C^{8i-7,\,8i}$.

  $\vdots$
- *Le a-ième pas :* Nous construisons la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de $C^{1,\,2^{a-1}}$ et $C^{2^{a-1}+1,\,2^a}$, et nous obtenons le code $C^{1,\,2^a}$, qui est *équivalent* à C.

Par la suite, nous donnerons une formule simple pour la matrice génératrice de $C^{1,\,2^a}$. On remarque facilement que le i-ème bloc de lignes contient G^i ou la matrice nulle. Il reste donc à identifier les positions nulles.

Pour cela, nous associerons à la matrice génératrice de la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de deux codes, la matrice

$$G_{(2,2)} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \tag{5.9}$$

et nous soulignons que ces deux matrices ont une et la même position nulle.

Les matrices génératrices intervenant au deuxième pas de la construction seront associées au *produit de Kronecker de deuxième niveau de* $G_{(2,2)}$ défini par

$$G_{(2^2,2^2)} = G_{(2,2)} \otimes G_{(2,2)} = \begin{bmatrix} G_{(2,2)} & G_{(2,2)} \\ 0 & G_{(2,2)} \end{bmatrix}.$$

Évidemment, la matrice génératrice obtenue à la fin de la construction a les mêmes positions nulles que *le produit de Kronecker de a-ième niveau de* $G_{(2,2)}$

$$G_{(2^a,2^a)} = G_{(2^{a-1},2^{a-1})} \otimes G_{(2,2)} = \begin{bmatrix} G_{(2^{a-1},2^{a-1})} & G_{(2^{a-1},2^{a-1})} \\ 0 & G_{(2^{a-1},2^{a-1})} \end{bmatrix}. \tag{5.10}$$

Notons par ℓ^i, pour $i \in [1,\, 2^{a-1}]$, et r^i, pour $i \in [1,\, 2^a]$, la i-ème ligne de $G_{(2^{a-1},2^{a-1})}$, resp. $G_{(2^a,2^a)}$. La formule précédente implique

$$r^i = \begin{cases} (\ell^i, \ell^i) & \text{si } i \in [1, 2^{a-1}], \\ (0, \ell^{i-2^{a-1}}) & \text{si } i \in [2^{a-1}+1, 2^a]. \end{cases} \tag{5.11}$$

Nous avons ainsi prouvé le théorème suivant. Rappelons que C est un code cyclique à racines multiples de longueur $2^a b$ ($a \geq 1$ et b impair) et de générateur g ; la notation $C^{i,j}$ a été expliquée à la fin de la Section 5.1.

Théorème 5.3 *La matrice génératrice de* $C^{1,\,2^a}$ *est :*

$$G^{1,\,2^a} = G^1 \otimes r^1 + G^2 \otimes r^2 + \ldots + G^{2^a} \otimes r^{2^a}, \tag{5.12}$$

où les r^i *ont été précédemment définis. Le code* $C^{1,\,2^a}$ *est équivalent à* C.

Exemple : Construisons la matrice génératrice de $C^{1,2^4}$. Pour cela nous devons connaître la matrice du produit de Kronecker de 4-ième niveau. Nous commençons par calculer

$$G_{(2^2,2^2)} = \begin{bmatrix} G_{(2,2)} & G_{(2,2)} \\ 0 & G_{(2,2)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

et

$$G_{(2^3,2^3)} = G_{(2^2,2^2)} \otimes G_{(2,2)} = \begin{bmatrix} G_{(2^2,2^2)} & G_{(2^2,2^2)} \\ 0 & G_{(2^2,2^2)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

De la même façon on calcule $G_{(2^4,2^4)}$ et on identifie les lignes r^i de cette matrice. En utilisant la formule (5.12), on conclut :

$$G^{1,16} = \begin{bmatrix} G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 & G^1 \\ 0 & G^2 & 0 & G^2 & 0 & G^2 & 0 & G^2 & 0 & G^2 & 0 & G^2 & 0 & G^2 & 0 & G^2 \\ 0 & 0 & G^3 & G^3 & 0 & 0 & G^3 & G^3 & 0 & 0 & G^3 & G^3 & 0 & 0 & G^3 & G^3 \\ 0 & 0 & 0 & G^4 & 0 & 0 & 0 & G^4 & 0 & 0 & 0 & G^4 & 0 & 0 & 0 & G^4 \\ 0 & 0 & 0 & 0 & G^5 & G^5 & G^5 & G^5 & 0 & 0 & 0 & 0 & G^5 & G^5 & G^5 & G^5 \\ 0 & 0 & 0 & 0 & 0 & G^6 & 0 & G^6 & 0 & 0 & 0 & 0 & 0 & G^6 & 0 & G^6 \\ 0 & 0 & 0 & 0 & 0 & 0 & G^7 & G^7 & 0 & 0 & 0 & 0 & 0 & 0 & G^7 & G^7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^9 & G^9 & G^9 & G^9 & G^9 & G^9 & G^9 & G^9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^{10} & 0 & G^{10} & 0 & G^{10} & 0 & G^{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^{11} & G^{11} & 0 & 0 & G^{11} & G^{11} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^{12} & 0 & 0 & 0 & G^{12} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^{13} & G^{13} & G^{13} & G^{13} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^{14} & 0 & G^{14} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^{15} & G^{15} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G^{16} \end{bmatrix}.$$

5.2.2 Plus sur la matrice génératrice

Dans cette section nous obtenons une construction carrée pour tous les codes cycliques binaires à racines multiples. Cela peut être aussi interprété comme une extension de la*construction dite "gluing"* décrite par [84, Th.107, p.177] pour des codes *auto-duaux*.

Théorème 5.4 *Tout code cyclique binaire à racines multiples C de longueur n et polynôme générateur g est équivalent à une construction carrée $C \sim |A/B|^2$, avec A et B deux codes de longueur $n/2$ qui peuvent être calculés précisément en fonction de n et g. En terme de matrices génératrices on a*

$$G \sim \begin{bmatrix} G_B & 0 \\ 0 & G_B \\ G_{A/B} & G_{A/B} \end{bmatrix}, \quad \text{où} \quad G_A = \begin{bmatrix} G_B \\ G_{A/B} \end{bmatrix}. \tag{5.13}$$

La preuve de ce théorème utilise le lemme suivant :

Lemme 5.1 *Soit E_1 et E_2 deux codes cycliques de longueur impaire b et de générateur e_1, resp. e_1e_2. Alors on peut prendre pour code E_1/E_2 le code* cyclique *de générateur $e = (x^b + 1)/e_2$.*

Preuve du lemme. Notons par E le code cyclique de générateur e ; on doit montrer que $E + E_2 = E_1$ et que $E \cap E_2 = \{\mathbf{0}\}$. En écrivant $e = (x^b + 1)e_1/(e_1e_2)$, nous déduisons facilement :

- E est un sous-code de E_1, car e est divisible par e_1 ;
- $k_E = b - \deg e = \deg e_2 = k_{E_1} - k_{E_2}$.

Pour conclure il suffit de vérifier que $E \cap E_2 = \{\mathbf{0}\}$. En vertu du Th. 5.16 de [41, p. 55], le code $E_2 \cap E$ a pour polynôme générateur :

$$\text{p.p.c.m.}\,(e_1e_2,\, e_1(x^b + 1)/(e_1e_2)).$$

Mais e_1, e_2 et $(x^b+1)/(e_1e_2)$ sont premiers entre eux, car x^b+1 est sans facteurs carrés. On conclut que $E_2 \cap E = \{\mathbf{0}\}$ car il a pour générateur $e_1e_2(x^b+1)/(e_1e_2) = x^b+1$, tout en étant de longueur b. ◇

Remarque 5.1 Avec les notations à la fin de la Section 5.1, le lemme 5.1 est vrai pour toute paire de codes C^i et C^j, avec $i < j$.

Preuve du théorème 5.4. Nous commençons par écrire $n = 2^a b$, avec $a \geq 1$ et b impair (voir la définition 5.2). La preuve est basée sur la décomposition unique du générateur g donnée par la relation (5.8).

Pour $a = 1$, le polynôme générateur g se décompose en $g_1^2 g_2$, avec g_1 et g_2 deux diviseurs premiers entre eux de $x^b + 1$. Nous considérons les codes C^1 et C^2 de générateurs g_1, resp. g_1g_2 (voir théorème 5.2). La remarque 2.2 et le lemme 5.1 étant vérifiés par ces codes, nous obtenons la conclusion en choisissant $G_B = G^2$ et $G_{A/B} = G^{1/2}$ (le générateur de $C^{1/2}$ étant $(x^b + 1)/g_2$).

Dans le cas général, pour tout entier $i \in [1, 2^{a-1}]$ nous construisons le code cyclique $C^{i/2^{a-1}+i}$ de générateur

$$\frac{(x^b + 1)g_1 \dots g_i}{g_1 \dots g_{2^{a-1}+i}}.$$

Utilisant les relations (5.11) et (5.12), nous déduisons :

$$\begin{aligned}
G &\sim \sum_{i=1}^{2^{a-1}} G^i \otimes r^i + G^{2^{a-1}+i} \otimes r^{2^{a-1}+i} \\
&= \sum_{i=1}^{2^{a-1}} [G^{2^{a-1}+i} + G^{i/2^{a-1}+i}] \otimes r^i + G^{2^{a-1}+i} \otimes r^{2^{a-1}+i} \\
&= \sum_{i=1}^{2^{a-1}} G^{2^{a-1}+i} \otimes [r^i + r^{2^{a-1}+i}] + G^{i/2^{a-1}+i} \otimes r^i \\
&= \sum_{i=1}^{2^{a-1}} G^{2^{a-1}+i} \otimes [(\ell^i, \ell^i) + (0, \ell^i)] + G^{i/2^{a-1}+i} \otimes (\ell^i, \ell^i) \\
&= \sum_{i=1}^{2^{a-1}} G^{2^{a-1}+i} \otimes [(\ell^i, 0) + (0, \ell^i)] + G^{i/2^{a-1}+i} \otimes (\ell^i, \ell^i) \\
&= \sum_{i=1}^{2^{a-1}} [G^{2^{a-1}+i} \otimes (\ell^i, 0) + G^{2^{a-1}+i} \otimes (0, \ell^i) + G^{i/2^{a-1}+i} \otimes (\ell^i, \ell^i)],
\end{aligned}$$

où l'on somme des matrices génératrices comme cela est expliqué en Section 2.1. La preuve est complète si on considère :

$$G_B = \sum_{i=1}^{2^{a-1}} G^{2^{a-1}+i} \otimes \ell^i = G^{2^{a-1}+1,\, 2^a}$$

et

$$G_{A/B} = \sum_{i=1}^{2^{a-1}} G^{i/2^{a-1}+i} \otimes \ell^i. \tag{5.14}$$

◇

5.2.3 Un cas particulier

Dans cette section, nous restreignons notre étude au cas où toutes les racines de$g$ ont une multiplicité paire. Afin de simplifier notre démarche, nous**notons** par $S(A, B)$ la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ de deux codes A et B, et par $G(A, B)$ sa matrice génératrice.

Lemme 5.2 *Considérons deux codes A et B de même longueur. Alors*

$$S(A \oplus A, B \oplus B) \sim S(A, B) \oplus S(A, B).$$

Preuve. Si nous notons par G_A et G_B les matrices génératrices des codes A et B, nous obtenons

$$G(A \oplus A, B \oplus B) = \begin{bmatrix} G_{A\oplus A} & G_{A\oplus A} \\ 0 & G_{B\oplus B} \end{bmatrix} = \begin{bmatrix} G_A & 0 & G_A & 0 \\ 0 & G_A & 0 & G_A \\ 0 & 0 & G_B & 0 \\ 0 & 0 & 0 & G_B \end{bmatrix} \sim$$

$$\sim \begin{bmatrix} G_A & G_A & 0 & 0 \\ 0 & 0 & G_A & G_A \\ 0 & G_B & 0 & 0 \\ 0 & 0 & 0 & G_B \end{bmatrix} = \begin{bmatrix} G_A & G_A & 0 & 0 \\ 0 & G_B & 0 & 0 \\ 0 & 0 & G_A & G_A \\ 0 & 0 & 0 & G_B \end{bmatrix} = \begin{bmatrix} G(A,B) & 0 \\ 0 & G(A,B) \end{bmatrix}.$$

◇

Le résultat de ce lemme sera appliqué de façon récursive pour prouver le théorème suivant :

Théorème 5.5 *Soit C un code cyclique de longueur $2^a b$ et générateur $g = g_1^{2^a} g_3^{2^a-2} \dots g_{2^a-1}^2$. Soit C' le code cyclique de longueur $2^{a-1}b$ et de générateur g' tel que $g'^2 = g$. Alors le code C est* équivalent à *$C' \oplus C'$; par conséquent, l'énumérateur des poids de C est le carré de l'énumérateur des poids de C'.*

Preuve. Si $a = 1$, via le théorème 5.2, le code C est équivalent à $S(C', C') = C' \oplus C'$. Supposant $a \geq 2$, les codes C et C' sont deux codes cycliques à racines multiples, pour lesquels on applique la construction décrite dans la Section 5.2.1. Pour chacun de ces codes on calcule un code équivalent en a pas (resp. en $a-1$ pas).

En construisant le code équivalent à C, nous adoptons la stratégie suivante : au premier pas, les sommes $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ sont en fait des sommes directes. Aux pas suivants, les sommes$|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ et les sommes directes commutent comme dans le lemme 5.2, et il suffit d'itérer le procédé. Au passage, au pas i de la construction du code équivalent à C on trouve des sommes directes des codes obtenus au pas $i-1$ de la construction du code équivalent à C'.

Construction du code équivalent à C' :

- *Préliminaires :* Comme $g' = g_1^{2^{a-1}} g_3^{2^{a-1}-1} \dots g_{2^a-1}$, on construit, pour tout $i \in [1, 2^{a-1}]$, les codes A^i de générateur $g_1 g_3 \dots g_{2i-1}$.
- *Premier pas :* Pour tout $i \in [1, 2^{a-2}]$ on construit les codes :

$$S(A^{2i-1}, A^{2i}). \tag{5.15}$$

- *Deuxième pas :* Pour tout $i \in [1, 2^{a-3}]$ on construit les sommes $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ des codes en (5.15) :

$$S(S(A^{4i-3}, A^{4i-2}), S(A^{4i-1}, A^{4i})). \tag{5.16}$$

$$\vdots$$

- *Le $(a-1)$-ième pas :* On obtient le code

$$S(S(\dots S((S(A^1, A^2), S(A^3, A^4)), (S(A^5, A^6), S(A^7, A^8))), \dots)). \tag{5.17}$$

Construction du code équivalent à C :

– *Préliminaires :* Comme $g = g_1^{2^a} g_3^{2^a-2} \dots g_{2^a-1}^2$, on construit, pour tout $i \in [1,\, 2^a]$, les codes de générateur $g_1 g_2 \dots g_i$, avec

$$g_{2i} = 1, \quad \forall\, i \in [1,\, 2^{a-1}].$$

A^i étant le code de générateur $g_1 g_3 \dots g_{2i-1}$, on a construit ainsi la suite des codes :

$$A^1,\, A^1, A^2,\, A^2,\, \dots,\, A^{2^{a-1}}, A^{2^{a-1}}. \tag{5.18}$$

– *Premier pas :* Pour $i \in [1, 2^{a-1}]$ on calcule les sommes $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ des codes en (5.18) :

$$S(A^i, A^i) = A^i \oplus A^i. \tag{5.19}$$

– *Deuxième pas :* Pour tout $i \in [1, 2^{a-2}]$, nous appliquons le lemme 5.2 pour les codes A^{2i-1} et A^{2i} et nous obtenons les codes :

$$S(A^{2i-1} \oplus A^{2i-1}, A^{2i} \oplus A^{2i}) = S(A^{2i-1}, A^{2i}) \oplus S(A^{2i-1}, A^{2i}). \tag{5.20}$$

Soulignons que les $S(A^{2i-1}, A^{2i})$ sont exactement les codes obtenus en relation (5.15), donc au premier pas de la construction du code C'.

– *Troisième pas :* Pour chaque $i \in [1, 2^{a-3}]$, nous appliquons le lemme 5.2 aux codes

$$S(A^{4i-3}, A^{4i-2}) \quad \text{et} \quad S(A^{4i-1}, A^{4i})$$

et nous obtenons les codes :

$$\begin{aligned} & S(S(A^{4i-3}, A^{4i-2}) \oplus S(A^{4i-3}, A^{4i-2}),\, S(A^{4i-1}, A^{4i}) \oplus S(A^{4i-1}, A^{4i})) = \\ = \;& S(S(A^{4i-3}, A^{4i-2}), S(A^{4i-1}, A^{4i})) \oplus S(S(A^{4i-3}, A^{4i-2}), S(A^{4i-1}, A^{4i})). \end{aligned}$$

Soulignons qu'on a retrouvé les codes intervenant dans la relation (5.16).

$$\vdots$$

– *Le a-ième pas :* On calcule la somme $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ des deux codes obtenus au pas $a-1$ et, via le lemme 5.2, on obtient la somme directe du code en (5.17) avec lui-même, donc la conclusion.

◇

5.3 Énumérateurs des poids

Dans cette section nous travaillons avec des codes *c.a.d.* binaires de longueur $2^a b$ ($a \geq 1$ et b impair) et polynôme générateur g. Tous les résultats numériques ont été obtenus en utilisant Magma [12].

5.3.1 Énumération des codes

Nous nous sommes intéressés premièrement au nombre des codes *c.a.d.* de longueur $n \leq 120$, inéquivalents et différents de 1. Les polynômes générateurs de ces codes sont calculés en utilisant la formule (5.7), ce qui implique :

Théorème 5.6 *[93] Le nombre des codes* c.a.d. *distincts de longueur* $n = 2^a b$, $a \geq 1$, b *impair, est* $(2^a + 1)^{\delta(b)}$, *où* $\delta(b)$ *est le nombre de paires de classes cyclotomiques asymétriques modulo* b.

Remarque 5.2 Il a été prouvé en [11] que, dans le cas $n = 2b$ (b impair), un code *c.a.d.* sur $\mathbb{F}_2$ peut être obtenu comme image de Gray d'un code *c.a.d.* de longueur b sur $\mathbb{F}_2 + u\mathbb{F}_2$.

Les résultats se trouvent en Section B.1, où l'ensemble de définition I de chaque code est décrit comme réunion de classes cyclotomiques. La classe $C_s^{(b)}$ est notée par (s), afin de simplifier les notations. D'autre part, la notation $(s)^i$, $i > 1$, signifie que la classe (s) est *répétée* i fois.

Nous avons prouvé qu'il existe (*à équivalence près*) :

- Un code pour les longueurs n dans $\{14, 30, 46, 78, 94, 110\}$.
- Deux codes pour $n \in \{28, 60, 92, 102\}$.
- Trois codes pour $n \in \{90, 98\}$.
- Quatre codes pour $n \in \{42, 56, 70, 120\}$.
- Cinq codes pour $n = 62$.
- Huit codes pour $n = 112$.
- Douze codes pour $n = 84$.

Nous vérifions l'inéquivalence de ces codes en calculant le nombre de mots de poids 4, 6 ou 8. Il y a quelques exceptions : deux codes *c.a.d.* [62, 31, 8] et deux codes *c.a.d.* [62, 31, 10] ont les mêmes distributions des poids. Nous prouvons leur inéquivalence en utilisant la fonction IsEquivalent de Magma. La même fonction induit *l'équivalence* des deux codes *c.a.d.* [98, 49, 4]. Pour $n = 62$ et $n = 102$, l'équivalence a été prouvée en utilisant le théorème 1.1.

Pour certains codes, les méthodes précédentes ne peuvent pas être appliquées. Il s'agit des deux codes *c.a.d.* de longueur 90 pour lesquels nous avons réussi à prouver le résultat suivant :

Théorème 5.7 *Les deux codes* c.a.d. [90, 45, 8] *sont équivalents.*

Preuve. Nous utilisons la construction carrée pour ces codes (voir le théorème 5.4). Leurs polynômes générateurs sont décomposés de façon unique : $g = g_1^2 g_2$, où g_1 et g_2 sont des diviseurs premiers entre eux de $x^{45} + 1$. Leurs ensembles de définition sont donnés en Section B.1, et de plus g_2 est identique pour les deux codes. Les codes $C^{1/2}$ sont par conséquent identiques, étant engendrés par $(x^{45} + 1)/g_2$. Via la fonction IsEquivalent de Magma, on vérifie que les codes C^2 sont équivalents, considérant au préalable leurs codes duaux (voir la remarque i1.1). On conclut que les codes *c.a.d.* [90, 45, 8] sont équivalents (cf. la forme de leurs matrices génératrices en relation (5.13)). ◇

Les distributions des poids de tous les codes *c.a.d.* de longueur ≤ 62 peuvent être calculées (par exemple, en utilisant Magma). Par la suite nous allons analyser seulement les codes *c.a.d.* de longueur $n > 62$.

5.3.2 Cas général, complexité

Les polynômes énumérateurs des codes cycliques binaires à racines multiples peuvent être calculés en utilisant la construction carrée de ces codes. Par la suite nous allons exprimer précisément ces polynômes énumérateurs.

Théorème 5.8 *Soit C un code donné par une construction carrée, $C = |A/B|^2$. Le polynôme énumérateur de C vérifie*

$$W_C(x, y) = \sum_{\mathbf{a} \in A/B} W^2_{B+\mathbf{a}}(x, y). \tag{5.21}$$

Autrement dit, W_C peut être calculé dès que les polynômes énumérateurs des $2^{k_{A/B}}$ translatés de B sont connus.

Preuve. La matrice génératrice du code $C = |A/B|^2$ étant de la forme (5.13) :

$$\begin{bmatrix} G_B & 0 \\ 0 & G_B \\ G_{A/B} & G_{A/B} \end{bmatrix},$$

chaque mot du code C peut être exprimé par

$$(\mathbf{b_1} + \mathbf{a}, \mathbf{b_2} + \mathbf{a}) \quad \text{où} \quad \mathbf{b_1}, \mathbf{b_2} \in B \quad \text{et} \quad \mathbf{a} \in A/B.$$

Alors

$$\begin{aligned} W_C(x, y) &= \sum_{\mathbf{v} \in C} x^{n-\mathrm{wt}(\mathbf{v})} y^{\mathrm{wt}(\mathbf{v})} = \sum_{\mathbf{a} \in A/B} \sum_{\mathbf{b_1} \in B} \sum_{\mathbf{b_2} \in B} x^{n-\mathrm{wt}(\mathbf{b_1}+\mathbf{a})-\mathrm{wt}(\mathbf{b_2}+\mathbf{a})} y^{\mathrm{wt}(\mathbf{b_1}+\mathbf{a})+\mathrm{wt}(\mathbf{b_2}+\mathbf{a})} \\ &= \sum_{\mathbf{a} \in A/B} \left[\sum_{\mathbf{b_1} \in B} x^{n/2-\mathrm{wt}(\mathbf{b_1}+\mathbf{a})} y^{\mathrm{wt}(\mathbf{b_1}+\mathbf{a})} \right] \times \left[\sum_{\mathbf{b_2} \in B} x^{n/2-\mathrm{wt}(\mathbf{b_2}+\mathbf{a})} y^{\mathrm{wt}(\mathbf{b_2}+\mathbf{a})} \right] \\ &= \sum_{\mathbf{a} \in A/B} W^2_{B+\mathbf{a}}(x, y). \end{aligned}$$

◇

Corollaire 5.1 *Via le théorème 5.4, la formule (5.21) est valable pour tout code C cyclique à racines multiples, et en particulier pour tout code* c.a.d.

Nous nous sommes aussi intéressés à la complexité du calcul des polynômes énumérateurs ; plus précisément, la complexité est une fonction de $k_{A/B}$. Rappelons que tout code cyclique à racines multiples C de longueur $2^a b$, $a \geq 1$ et b impair, est équivalent à une construction carrée, avec une matrice génératrice de la forme (5.13) et de plus (voir la relation (5.14)) :

$$G_{A/B} = \sum_{i=1}^{2^{a-1}} G^{i/2^{a-1}+i} \otimes \ell^i.$$

Rappelons que :

- le polynôme générateur du code C se décompose de façon unique en $g_1^{2^a} g_2^{2^a-1} \dots g_{2^a}$, où les polynômes g_i, $i \in [1, 2^a]$, sont des diviseurs premiers entre eux de $x^b + 1$.
- les codes C^i sont les codes cycliques de longueur b et polynôme générateur $g_1 g_2 \dots g_i$. Leur matrice génératrice est notée G^i.

Si $|G|$ désigne le nombre de lignes d'une matrice G, on calcule :

$$\begin{aligned}
k_{A/B} &= |G_{A/B}| = \sum_{i=1}^{2^{a-1}} |G^{i/2^{a-1}+i}| = \sum_{i=1}^{2^{a-1}} (|G^i| - |G^{2^{a-1}+i}|) = \sum_{i=1}^{2^{a-1}} (k_{C^i} - k_{C^{2^{a-1}+i}}) = \\
&= \sum_{i=1}^{2^{a-1}} \left[\left(b - \sum_{j=1}^{i} \deg g_j \right) - \left(b - \sum_{j=1}^{2^{a-1}+i} \deg g_j \right) \right] = \sum_{i=1}^{2^{a-1}} \left[\sum_{j=1}^{2^{a-1}+i} \deg g_j - \sum_{j=1}^{i} \deg g_j \right] = \\
&= \sum_{i=1}^{2^{a-1}} \sum_{j=i+1}^{2^{a-1}+i} \deg g_j = \sum_{i=1}^{2^{a-1}} (\deg g_{i+1} + \deg g_{i+2} + \dots + \deg g_{2^{a-1}+i}) = \\
&= \deg g_2 + 2 \deg g_3 + 3 \deg g_4 \dots + 2^{a-1} \deg g_{2^{a-1}+1} + \dots + 2 \deg g_{2^a-1} + \deg g_{2^a}. \quad (5.22)
\end{aligned}$$

Remarque 5.3 La relation (5.21) s'applique pour le calcul des énumérateurs des poids si le nombre des translatés intervenant ne dépasse pas 2^{28}.

5.3.3 Deux cas particuliers

Dans cette section nous présentons deux cas où l'énumérateur des poids d'un code cyclique à racines multiples peut être facilement calculé en utilisant le polynôme énumérateur, supposé connu, d'un code de longueur plus petite.

Premièrement, comme corollaire du théorème 5.5, nous avons obtenu les distributions des poids de quelques codes *c.a.d.* de longueur n et de générateur g'^2, en utilisant les distributions des poids des codes *c.a.d.* de longueur $n/2$ et générateur g', qu'on peut calculer avec MAGMA.

Exemple : Nous avons appliqué cette première méthode à quatre codes de longueurs 84, un code de longueur 92, quatre codes de longueur 112 et deux codes de longueur 120 (voir les résultats en Section B.2).

Deuxièmement, nous utilisons le théorème suivant :

Théorème 5.9 *Soit C un code cyclique de longueur $n = 2b$ (b impair) et générateur $g(x) = (x+1)g_1(x)^2$, g_1 étant un polynôme irréductible sur $\mathbb{F}_2$.*

Alors l'énumérateur des poids de C est égal à la partie de poids pair du carré *de l'énumérateur des poids du code de générateur g_1.*

Preuve. Notons par C^1 et C^2 les codes de polynômes générateurs $g_1(x)$, resp. $(x+1)g_1(x)$; plus précisément, C^2 est le sous-code de poids pair de C^1. La construction décrite dans la Section 5.2.1 implique que C est équivalent à la somme $|u|u+v|$ de C^1 et C^2. Il s'ensuit que $C^{1/2}$ peut être engendré par *n'importe quel mot de code de poids impair* $\mathbf{w} \in C^1$ et la matrice génératrice de C peut être exprimée par (voir la preuve du théorème 5.4) :

$$\begin{bmatrix} G^2 & 0 \\ 0 & G^2 \\ \mathbf{w} & \mathbf{w} \end{bmatrix}.$$

Notons par W_1, W_{1e}, W_{1o} les polynômes énumérateurs de C^1, de sa partie de poids pair, resp. de sa partie de poids impair. Si A_i désigne le nombre des mots de poids i dans C^1, on a

$$W_1(x,y) = \sum_{i=1}^{n} A_i x^{n-i} y^i = \sum_{i\in[1,n],\,\text{pair}} A_i x^{n-i} y^i + \sum_{i\in[1,n],\,\text{impair}} A_i x^{n-i} y^i = W_{1e} + W_{1o}$$

Ensuite, le théorème 5.8 donne :

$$W_C = W_{C^2}^2 + W_{C^2+\mathbf{w}}^2 = (W_{1e})^2 + (W_{1o})^2.$$

Mais $W_1^2 = (W_1^2)_e + (W_1^2)_o$ et par ailleurs

$$W_1^2 = (W_{1e} + W_{1o})^2 = [(W_{1e})^2 + (W_{1o})^2] + 2W_{1e}W_{1o} = W_C + 2W_{1e}W_{1o},$$

donc finalement

$$W_C = (W_1^2)_e.$$

◇

Exemple : Nous avons calculé le polynôme énumérateur de l'unique code *c.a.d.* de longueur 94 comme étant la partie de poids pair du carré de l'énumérateur du code résidu quadratique [47, 24, 11].

5.3.4 L'*ombre* d'un code

Dans cette section, nous utilisons une méthode développée dans [25] pour calculer les énumérateurs des poids des codes cycliques auto-duaux.

Définition 5.4 *Soit C un code auto-dual, C_0 le sous-code de C :*

$$C_0 = \{\mathbf{x} \in C \mid wt(\mathbf{x}) = 0 \,(\mathrm{mod}\, 4)\},$$

et notons $C_2 = C \backslash C_0$. Le code S, dit ombre du code C, *est défini ainsi :*

$$S = \left\{ \mathbf{u} \mid \begin{array}{ll} \mathbf{u}\cdot\mathbf{v} = 0, & si\ \mathbf{v} \in C_0 \\ \mathbf{u}\cdot\mathbf{v} = 1, & si\ \mathbf{v} \in C_2 \end{array} \right\}.$$

Remarque 5.4 On peut facilement prouver que C_0 est un sous-espace de C de codimension 1, donc le sous-code C_2 est *le translaté* de C_0 :

$$C_2 = \mathbf{x} + C_0, \quad \forall\, \mathbf{x} \text{ tel que } \ \mathrm{wt}(\mathbf{x}) = 2 \,(\mathrm{mod}\, 4).$$

Remarque 5.5 Si C est un code de Type II, alors $C_2 = \emptyset$ et $S = C = C_0$. Par conséquent, la construction de *l'ombre* est intéressante seulement dans le cas des codes de Type I.

Nous résumons dans le théorème suivant quelques propriétés importantes du code$S$. Pour plus de détails, notamment pour le cas plus général des codes auto-orthogonaux, voir [25, Th.5] et [41, Th.6, p. 202].

Théorème 5.10 *Soit C un code auto-dual de Type I. L'ombre S du code vérifie :*

1. *S est le translaté du code C ; plus précisément $S = C_0^{\perp} \backslash C$.*
2. *S n'est pas un code linéaire.*
3. *si $W_C(x, y) = \sum_{i=0}^{n} A_i x^{n-i} y^i$ et $W_S(x, y) = \sum_{i=0}^{n} B_i x^{n-i} y^i$ sont les polynômes énumérateurs de C, resp. S, nous pouvons écrire*

$$W_C(x, y) = \sum_{j=0}^{\lfloor n/8 \rfloor} a_j (x^2 + y^2)^{n/2-4j} [x^2 y^2 (x^2 - y^2)^2]^j, \tag{5.23}$$

pour des entiers a_j à déterminer, et déduire

$$W_S(x, y) = \sum_{j=0}^{\lfloor n/8 \rfloor} (-1)^j a_j 2^{n/2-6j} (xy)^{n/2-4j} (x^4 - y^4)^{2j}. \tag{5.24}$$

Éléments de preuve. L'inclusion suivante étant évidente :

$$C_0 \subset C = C^{\perp} \subset C_0^{\perp},$$

et sachant que $|C/C_0| = 2$, il s'ensuit que $C_0^{\perp}$ est une réunion de deux translatés de C, ou bien de quatre translatés de C_0. Il existe donc un $\mathbf{w} \in C_0^{\perp} \backslash C$ tel que :

$$C_0^{\perp} = C + (\mathbf{w} + C).$$

On en déduit que

$$S = \mathbf{w} + C = \mathbf{w} + (C_0 \cup C_2) = (\mathbf{w} + C_0) \cup (\mathbf{w} + C_2),$$

donc l'ombre du code C est un translaté de C et une réunion de deux cosets de C_0. De plus :

$$\mathbf{u} + \mathbf{v} \in \begin{cases} C_0, & \text{si } \mathbf{u}, \mathbf{v} \in \mathbf{w} + C_0 \text{ où } \mathbf{u}, \mathbf{v} \in \mathbf{w} + C_2, \\ C_2, & \text{si } \mathbf{u} \in \mathbf{w} + C_0 \text{ et } \mathbf{v} \in \mathbf{w} + C_2. \end{cases} \tag{5.25}$$

En résumant, la somme de deux vecteurs dans S est dans C, donc S n'est pas linéaire.

◇

Par conséquent, nous pouvons restreindre notre étude au calcul des variables inconnues $a_1, \ldots, a_{\lfloor n/8 \rfloor}$ ($a_0 = 1$). La stratégie est de calculer une partie de ces coefficients en utilisant le code et l'autre en utilisant son *ombre*. Par la suite nous présentons les étapes pour les deux codes *c.a.d.* [102, 51, 6].

1. Nous obtenons a_i, $i \in \{1, 2, \ldots, 8\}$ en utilisant le *code C* :

   nous obtenons A_i, $i \in \{6, 8, 10, 12, 14, 16\}$ en utilisant Magma ;

   nous résolvons un système de 8 équations et 8 inconnues via (5.23).

2. Nous obtenons a_i, $i \in \{9, 10, 11, 12\}$ en utilisant l'*ombre S* :

   nous obtenons B_3, B_7, B_{11}, B_{15} en utilisant Magma et la formule : $S = C_0^{\perp} \backslash C$;

nous résolvons un système de 4 équations et 4 inconnues via (5.24).

Remarque 5.6 (Avantages de la méthode de l'*ombre*)
- $B_i = 0$ sauf si $i \equiv n/2 \pmod 4$ [25, Th.5].
- Le calcul de B_i nécessite seulement le calcul des A_i et du nombre des mots de code de poids i pour $C_0^{\perp}$, mais pas le calcul de S.
- Les A_i et les B_i peuvent être calculés indépendamment.

Cependant, le calcul de A_{16} pour le deuxième code *c.a.d.* de longueur 102 a pris 38 jours sur un Intel-Pentium II 400 MHz (processeur dual), donc cette méthode n'est pas efficace pour le calcul des énumérateurs des poids des codes de longueurs supérieures à 102.

5.3.5 Comparaison des différentes méthodes utilisées

Pour une partie de nos codes, la méthode générale décrite en section 5.3.2 est plus rapide que la méthode de l'*ombre*, même si cette dernière s'applique à tout code *c.a.d.* de longueur $n \leq 102$. Pour trois codes *c.a.d.* de longueur 98, la dimension $k_{A/B}$ du code A/B (défini par le théorème 5.4) est égale à 1 ou 6 (cf. la relation (5.22)), donc on obtient immédiatement les polynômes énumérateurs.

Si $n > 102$ la méthode générale donne en plus les polynômes énumérateurs du code *c.a.d.* [110, 55, 10], de quatre codes *c.a.d.* de longueur 112 et d'un code *c.a.d.* de longueur 120 (2^{28} translatés en (5.21)). Le premier cas spécial en section 5.3.3 s'applique aussi à six codes.

Les distributions de quatre codes *c.a.d.* de longueur 112 ou 120 n'ont pas pu être calculées par ces méthodes (voir la Section B.1). **Ces distributions peuvent être calculées en combinant la méthode de l'*ombre* et le calcul du nombre des mots de petits poids via l'Algorithme décrit en Section 6.4**

6

Codes duadiques

La famille des codes duadiques, qui contient les codes résidus quadratiques (QR), a été introduite en [62] par Leon *et al.* Leurs distances minimales (pour des longueurs $n \leq 241$) ont été données en [86] via un algorithme *probabiliste*. Une autre famille intéressante liée aux codes QR, les codes quadratiques doublement circulants (QDC), a été introduite par Pless sur $\mathbb{F}_3$ [81] (voir aussi Beenker [7]) et par Karlin [49] sur $\mathbb{F}_2$ et a été généralisée par Gaborit sur $\mathbb{F}_q$ [35].

Les codes duadiques (surtout la sous-famille des codes QR) et les codes QDC contiennent beaucoup de codes qui ont les meilleurs paramètres connus à ce jour. Si ces codes sont auto-duaux extrémaux, la distribution des poids est fixée, mais pour la plupart des codes auto-duaux non-extrémaux seulement leurs distances minimales sont connues. Dans ce chapitre nous étudions la distribution des poids de ces codes. Plus précisément, nous donnons les distributions des poids de tous les *codes XQR (QR étendus) de longueur* $n \leq 152$ (sauf si $n = 138$). Nous avons aussi calculé les distributions des poids de quelques codes QDC avec une bonne distance minimale. Dans le cas *ternaire*, on calcule les distributions des meilleurs codes QDC et XQR pour $n \leq 96$.

Nos résultats sont basés sur le calcul du nombre des mots de poids pas très grand de ces codes. Cela est fait via un *algorithme parallélisé* qui a pour noyau une méthode de calcul bien connue dans le cas des codes auto-duaux et qui est plus rapide que MAGMA [71]. On déduit par la suite l'énumérateur des poids via des méthodes classiques.

Pour une simple comparaison avec d'autres classes de codes, notons que, dans le cas binaire, les distributions de tous les codes BCH primitifs de longueur $n \leq 128$ sont connues [26], tandis que celles des codes XQR de longueur 74 ou 90 n'étaient pas connues auparavant.

Le chapitre est organisé de la manière suivante : les Sections 6.1 et 6.2 servent pour le rappel des définitions et des propriétés des codes étudiés. En Section 6.3 nous rappelons les méthodes classiques de calcul des polynômes énumérateurs pour les codes auto-duaux et formellement auto-duaux de poids pairs. En Section 6.4 nous décrivons l'algorithme de calcul du nombre de mots de poids donné dans un code linéaire de paramètres $[n, k, d]$, tel que $n = 2k$. Enfin, en Section 6.5 nous listons les résultats connus et ceux que nous avons obtenus.

Les résultats numériques sont listés en Annexe : en Section C.1 nous donnons le nombre des mots de code de petit poids pour les codes que nous considérons ; enfin nous listons en Section C.2 les énumérateurs des poids de tous les codes XQR binaires de longueur $74 \leq n \leq 152$ (sauf

pour $n = 138$) et de quelques codes QDC, ainsi que de tous les codes ternaires XQR ou QDC de longueur $72 \leq n \leq 96$.

6.1 Codes duadiques et résidus quadratiques

Dans cette section nous rappelons les propriétés de base des codes duadiques et QR. Pour plus de détails voir [85] et [86]. Les codes duadiques forment une famille infinie de codes cycliques de paramètres $[n, (n+1)/2]$ et $[n, (n-1)/2]$ qui contient les codes résidus quadratiques dont la longueur est un nombre premier. Par la suite nous allons utiliser des résultats présentés au Chapitre 1.

Définition 6.1 *Considérons un code C de longueur n sur $\mathbb{F}_q$ et les colonnes du code indexées par $0, 1, \ldots, n-1$. Un mot de code $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$ est de* poids pair *si $\sum_{i=0}^{n-1} y_i = 0$ dans $\mathbb{F}_q$, et de* poids impair *autrement.*

Un code est dit pair *si tous les mots du code sont de poids pair ; le code est dit* impair *s'il contient au moins un mot de poids impair.*

L'ensemble $\mathcal{E}$ des mots de poids pair dans R_n (les classes des polynômes dans $\mathbb{F}_q[x]$ modulo $x^n - 1$) forme un code cyclique $[n, n-1]$ dont le code dual $\mathcal{E}^\perp$ est le code à répétition d'idempotent générateur :

$$j(x) = \frac{1}{n}(1 + x + x^2 + \ldots + x^{n-1}).$$

Par conséquent $\mathcal{E}$ a pour idempotent $1 - j(x)$ (voir [41, th.5.23, p.59]). Les codes duadiques peuvent être définis en termes d'idempotents ou de classes cyclotomiques. Il y a quatre codes duadiques de longueur n, qui se divisent en deux couples : un dit *pair* et l'autre *impair*. Soient $C_1 = \langle e_1(x) \rangle$ et $C_2 = \langle e_2(x) \rangle$, où $e_1(x)$ et $e_2(x)$ sont les idempotents de poids pair associés aux codes cycliques C_1 et C_2. Alors C_1 et C_2 forment la paire des codes duadiques pairs si :

$$e_1(x) + e_2(x) = 1 - j(x),$$

et s'il existe un multiplicateur μ_a ($a < n$, p.g.c.d. $(a, n) = 1$, voir la définition 1.14) tel que

$$C_1\mu_a = C_2 \quad \text{et} \quad C_2\mu_a = C_1.$$

Par ailleurs, cela implique que $C_1 \cap C_2 = \{\mathbf{0}\}$, $C_1 + C_2 = \mathcal{E}$, n est impair et C_1 et C_2 sont chacun de dimension $\frac{n-1}{2}$. On peut associer à C_1 et C_2 la paire des codes duadiques impairs $D_1 = \langle 1 - e_2(x) \rangle$ et $D_2 = \langle 1 - e_1(x) \rangle$.

Les codes duadiques peuvent aussi être définis en termes de classes cyclotomiques modulo$q$. Soit C_1 et C_2 une paire de codes duadiques de poids pair et d'*ensemble de définition* $T_1 = \{0\} \cup S_1$, resp. $T_2 = \{0\} \cup S_2$, où S_1 et S_2 sont des réunions de classes cyclotomiques non-nulles modulo$q$.

Sachant que μ_a interchange C_1 et C_2, et que C_1 et C_2 sont de dimension $\frac{n-1}{2}$, on déduit :

$$\exists a < n, \text{ p.g.c.d. } (a, n) = 1,\ S_1\mu_a = S_2,\ S_2\mu_a = S_1,\ S_1 \cap S_2 = \emptyset \text{ et } S_1 \cup S_2 = \{1, 2, \ldots, n-1\}. \tag{6.1}$$

Définition 6.2 *Soient S_1 et S_2 deux ensembles qui vérifient les conditions (6.1) — appelés* partition de n par μ_a. *Un code cyclique est appelé* duadique (D) *s'il a pour idempotent un des quatre polynômes suivants :*

$$e_1(x) = \sum_{i \in S_1} x^i, \quad e_2(x) = \sum_{i \in S_2} x^i, \quad 1 + e_1(x) \text{ ou } 1 + e_2(x). \tag{6.2}$$

Parmi les propriétés des codes duadiques nous rappelons ici les suivantes (voir [85] et [86]) :

1. Il existe des codes duadiques de longueur n sur $\mathbb{F}_q$ si et seulement si il existe un multiplicateur qui donne une partition de n, donc si et seulement si q est un résidu quadratique modulo n.
2. Si n est un nombre premier $p = \pm 1 \pmod 8$, en prenant S_1 l'ensemble Q des résidus quadratiques non-nuls modulo n, S_2 l'ensemble N des non résidus quadratiques modulo n et $a \in N$ arbitraire, les conditions (6.1) sont satisfaites. Par conséquent, *les codes QR de longueur n premier sont des codes duadiques.*
3. Les codes duadiques *binaires* de longueur n existent si et seulement si

$$n = \prod_i p_i^{a_i}, \quad p_i = \pm 1 \pmod 8 \text{ et premier}, \forall\, i.$$

   Les codes duadiques *ternaires* de longueur n existent si et seulement si

$$n = \prod_i p_i^{a_i}, \quad p_i = \pm 1 \pmod{12} \text{ et premier}, \forall\, i.$$

4. [62] Soit $n = \prod_i p_i^{a_i}, \quad p_i = \pm 1 \pmod 8, \forall\, i$ et définissons $\epsilon = n \pmod 8$. Les codes duadiques d'idempotents $\frac{1-\epsilon}{2} + e_i$ sont de dimension $\frac{n-1}{2}$ et ceux d'idempotents $\frac{1+\epsilon}{2} + e_i$ sont de dimension $\frac{n+1}{2}$, où $i \in \{1, 2\}$.
5. [62] *Tout code binaire cyclique étendu auto-dual est un code duadique.* En particulier, les codes $\mathrm{RM}((m-1)/2, m)$ avec m impair sont des codes duadiques.
6. Soit $\mathcal{D}_1$ et $\mathcal{D}_2$ une paire de codes duadiques impairs de longueur n sur $\mathbb{F}_q$. On peut définir une *extension spéciale* utilisée pour les codes duadiques. Supposons qu'il existe une solution $\gamma \in \mathbb{F}_q$ de l'équation $1 + \gamma^2 n = 0$. Pour C un code linéaire, on définit $\widetilde{C} = \{\widetilde{\mathbf{c}} | \mathbf{c} \in C\}$:

$$\widetilde{\mathbf{c}} = (c_0, c_1, \ldots, c_{n-1}, c_\infty), \text{ où } \mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \text{ et } c_\infty = -\gamma \sum_{i=0}^{n-1} c_i.$$

   Alors :
   – Si μ_{-1} donne la partition pour les codes $\mathcal{D}_1$ et $\mathcal{D}_2$, alors $\widetilde{\mathcal{D}_1}$ et $\widetilde{\mathcal{D}_2}$ sont auto-duaux.
   – Si $\mathcal{D}_1 \mu_{-1} = \mathcal{D}_1$, alors $\widetilde{\mathcal{D}_1}$ et $\widetilde{\mathcal{D}_2}$ sont le dual l'un de l'autre.
   Dans le cas binaire, si $\widetilde{\mathcal{D}_1}$ est auto-dual, alors $\widetilde{\mathcal{D}_1}$ est de Type II si $n = -1 \pmod 8$ et de Type I si $n = 1 \pmod 8$.

6.2 Codes quadratiques doublement circulants

Dans cette section, nous présentons une classe de codes obtenus par une construction doublement circulante (voir la définition 2.12) en utilisant les résidus quadratiques.

Par la suite, nous désignons par $\mathcal{K}$ le corps $\mathbb{F}_2$ ou $\mathbb{F}_3$. Soit r, s et t des éléments de $\mathcal{K}$ et soit p un nombre premier. Nous notons par $Q_p(r,s,t) = (q_{ij})$ la matrice $p \times p$ sur $\mathcal{K}$, dont les lignes et les colonnes sont indexées par i, $j \in \{0, 1, 2, \ldots, p-1\}$. Les entrées q_{ij} sont définies en termes de résidus quadratiques et par la fonction χ (qui n'est pas nécessairement un caractère) :

$$\chi(i) = \begin{cases} r, & \text{si } i = 0, \\ s, & \text{si } i \in Q, \\ t, & \text{si } i \in N. \end{cases}$$

Nous définissons

$$q_{ij} = \chi(j - i).$$

Remarque 6.1 Selon la définition 2.10, la matrice $Q_p(r,s,t)$ est circulante. Si p n'est pas premier, une construction analogue [35] ne produit plus une matrice circulante.

Définition 6.3 *Pour tout α, β et γ dans $\mathcal{K}$, un code est* quadratique doublement circulant (QDC) *si sa matrice génératrice est l'une des deux matrices doublement circulantes suivantes :*

$$\mathcal{P}_p(r,s,t) = (I \mid Q_p(r,s,t)) \tag{6.3}$$

$$\mathcal{B}_p(r,s,t) = \left(\begin{array}{c|c|c|c} 1 & 0\cdots0 & \alpha & \beta\cdots\beta \\ \hline 0 & & \gamma & \\ \vdots & I & \vdots & Q_p(r,s,t) \\ 0 & & \gamma & \end{array}\right) \tag{6.4}$$

On dit que les deux matrices sont en forme doublement circulante pure (6.3), resp. presque doublement circulante (6.4).

Il s'ensuit que les codes doublement circulants sur $\mathbb{F}_p$, p premier, sont de paramètres $[2p, p]$ ou $[2p+2, p+1]$. Définissons :

$$\zeta = \begin{cases} 1 & \text{si } p = 4k+1, \\ -1 & \text{si } p = 4k+3. \end{cases}$$

On travaille toujours avec $\gamma = \zeta\beta$ et, sauf spécification supplémentaire, on considère $\beta = 1$ et $\alpha = r$. Les codes de matrices génératrices $\mathcal{B}_p(r,s,t)$ peuvent être auto-duaux ou isoduaux en fonction de différents paramètres.

6.3 Calcul des énumérateurs des poids

Le calcul du polynôme énumérateur des poids est en général un problème difficile. Mais dans certains cas, le polynôme énumérateur appartient à un certain anneau algébrique. Alors il peut être obtenu à partir du nombre des mots de code de poids donné, *poids pas très grand*.

Dans ce chapitre nous travaillons avec quatre types de codes : de Type II, de Type I, codes binaires formellement auto-duaux pairs et codes ternaires auto-duaux. Cette section sert à rappeler comment calculer les polynômes énumérateurs quand on connaît certains coefficients A_i (voir [87] pour plus de détails sur les codes auto-duaux et leurs polynômes énumérateurs).

Une attention particulière est prêtée aux codes de Type I, car l'introduction du code *ombre* permet de calculer moins de coefficients A_i pour déduire l'énumérateur des poids. Nous allons montrer qu'en fait le poids maximal est très près de la valeur correspondante pour les codes de Type II.

Par la suite nous serons amenés à utiliser les énumérateurs de poids suivants :

1. l'énumérateur des poids du code à répétition $[2, 1, 2]$ (code de Type I) :

$$W_1(x,\, y) = x^2 + y^2, \tag{6.5}$$

2. l'énumérateur du code de Hamming étendu $[8, 4, 4]$ (code de Type II) :

$$W_2(x,\, y) = x^8 + 14x^4y^4 + y^8, \tag{6.6}$$

3. l'énumérateur du code de Golay $[24, 12, 8]$ (code de Type II) :

$$W_3(x,\, y) = x^{24} + 759x^{16}y^8 + 2576x^{12}y^{12} + 759x^8y^{16} + y^{24}, \tag{6.7}$$

4. l'énumérateur du code ternaire $[4, 2, 3]$ (code de Type III) :

$$W_4(x,\, y) = x^4 + 8xy^3, \tag{6.8}$$

5. l'énumérateur du code de Golay ternaire $[12, 6, 6]$ (code de Type III) :

$$W_5(x,\, y) = x^{12} + 264x^6y^6 + 440x^3y^9 + 24y^{12}. \tag{6.9}$$

Nous allons d'abord rappeler que les énumérateurs des poids des codes étudiés dans ce chapitre peuvent être exprimés en fonction des énumérateurs listés précédemment.

Définition 6.4 *Soit $g_1(x, y)$ et $g_2(x, y)$ deux polynômes* homogènes *de degrés r_1, resp. r_2, avec r_1 et r_2 diviseurs d'un entier n. Soit R l'ensemble des paires (i, j) ($i, j \geq 0$ entiers) de solutions de l'équation $r_1 i + r_2 j = n$. Alors le polynôme homogène de degré n :*

$$\sum_{(i,j)\in R} a_i g_1(x, y)^i g_2(x, y)^j \tag{6.10}$$

est une combinaison linéaire *de $g_1(x, y)$ et $g_2(x, y)$.*

Théorème 6.1 *[38]*

1. *l'énumérateur des poids de tout code auto-dual binaire de Type I est une combinaison linéaire de W_1 et W_2 ;*
2. *l'énumérateur des poids de tout code auto-dual binaire de Type II est une combinaison linéaire de W_2 et W_3 ;*

3. *l'énumérateur des poids de tout code auto-dual ternaire de Type III est une combinaison linéaire de W_4 et W_5.*

Remarque 6.2 Tous ces polynômes, ainsi que les énumérateurs des poids correspondants restent inchangés sous l'action d'un groupe, différent selon les cas. Ils sont appelés*invariants*. Pour plus de détails sur *la théorie des invariants* voir [70, Ch.19].

Le calcul se réduit à la résolution d'un système d'équations dont les inconnues sont les variables a_i dans (6.10), en fonction du nombre A_i de mots de poids i. On préfère que *le système à résoudre soit triangulaire supérieur*, par conséquent on choisit de modifier un des deux invariants comme suit :

1. $W_2'(x,y) = \frac{W_1(x,y)^4 - W_2(x,y)}{4} = x^2y^2(x^2-y^2)^2$ remplace W_2 pour les codes de Type I ;
2. $W_3'(x,y) = \frac{W_2(x,y)^3 - W_3(x,y)}{42} = x^4y^4(x^4-y^4)^4$ remplace W_3 pour les codes de Type II ;
3. $W_5'(x,y) = \frac{W_4(x,y)^3 - W_5(x,y)}{24} = y^3(x^3-y^3)^3$ remplace W_5 pour les codes de Type III.

Tous ces remarques conduisent au résultat suivant :

Théorème 6.2 *Les polynômes énumérateurs des poids admettent une écriture comme suit :*

1. *si le code C est binaire de Type I ou formellement auto-dual pair :*

$$W_C(x,y) = \sum_{i=0}^{\lfloor n/8 \rfloor} a_i (x^2+y^2)^{\frac{n}{2}-4i} \{x^2y^2(x^2-y^2)^2\}^i \ ;$$

   il suffit de calculer les $\lfloor n/8 \rfloor + 1$ premiers coefficients A_{2i} (i.e. jusqu'au poids $2\lfloor n/8 \rfloor \sim n/4$) pour déduire W_C.

2. *si le code C est binaire de Type II :*

$$W_C(x,y) = \sum_{i=0}^{\lfloor n/24 \rfloor} a_i (x^8+14x^4y^4+y^8)^{\frac{n}{8}-3i} \{x^4y^4(x^4-y^4)^4\}^i \ ;$$

   il suffit de calculer les $\lfloor n/24 \rfloor + 1$ premiers coefficients A_{4i} (i.e. jusqu'au poids $4\lfloor n/24 \rfloor \sim n/6$) pour déduire W_C.

3. *si le code C est ternaire de Type III :*

$$W_C(x,y) = \sum_{i=0}^{\lfloor n/12 \rfloor} a_i (x^4+8xy^3)^{\frac{n}{12}-3i} \{y^3(x^3-y^3)^3\}^i \ ;$$

   il suffit de calculer les $\lfloor n/12 \rfloor + 1$ premiers coefficients A_{3i} (i.e. jusqu'au poids $3\lfloor n/12 \rfloor \sim n/4$) pour déduire W_C.

Nous allons approfondir l'étude des codes binaires de Type I en utilisant le code*ombre*, qu'on a déjà présenté en Section 5.3.4. On montrera que cela nous permet de calculer seulement environ $n/24$ coefficients, comme dans le cas des codes de Type II.

Le polynôme énumérateur du code *ombre* S est lié au polynôme énumérateur de C par les coefficients a_i car :

$$W_S(x,y) = \sum_{i=0}^{\lfloor n/8 \rfloor} a_i (-1)^i 2^{\frac{n}{2}-6i} (xy)^{\frac{n}{2}-4i} (x^4-y^4)^{2i} = x^n + B_1 x^{n-1} y + \ldots + B_n y^n. \qquad (6.11)$$

La formule précédente implique que les seuls coefficients de S qui peuvent être non-nuls sont les B_i avec $i = ss \pmod 4$, où $ss = n/2 \pmod 4$. La forme particulière du polynôme xy dans la formule (6.11) implique aussi que les coefficients B_{ss+4i}, $0 \leq i \leq \lfloor n/8 \rfloor$, de W_S fixent les inconnues $a_{\lfloor n/8 \rfloor - i}$, $0 \leq i \leq \lfloor n/8 \rfloor$.

Le code C_0 étant donné par la Définition 5.4, notons que $S = C_0^{\perp} \backslash C$ (voir Théorème 5.10), donc l'énumérateur des poids de S peut être calculé par

$$W_S(x,y) = W_{C_0^{\perp}}(x,y) - W_C(x,y),$$

avec à peu près la même complexité que W_C.

Supposons maintenant qu'on soit capable de calculer le nombre de mots d'un code de rendement 1/2 jusqu'au poids $2m$, donc de déduire les coefficients $a_1, \ldots, a_m$. Il nous reste à déterminer $a_{\lfloor n/8 \rfloor}, \ldots, a_{m+2}, a_{m+1}$ via le code *ombre*. Cela demande le calcul de B_{ss}, B_{ss+4}, $\ldots$, $B_{ss+4(\lfloor n/8 \rfloor - m - 1)}$ et implique que $ss + 4(\lfloor n/8 \rfloor - m - 1)$ soit d'ordre (et plus petit que) $2m$. Ensuite on a :

$$2m = \lceil (ss + 4\lfloor n/8 \rfloor - 4)/3 \rceil.$$

Enfin $2m$ est d'ordre $n/6$, la même approximation étant obtenue pour les codes de Type II.

Exemple : Soit C un code auto-dual de Type I de longueur 90. Le nombre de variables a_i nécessaires pour le calcul de W_C est $1 + \lfloor 90/8 \rfloor = 12$ (a_0 est toujours 1), $ss = 1$ et $2m = 14$. Le nombre de mots de C de poids $2, 4, \ldots, 14$ donne les coefficients $a_1, a_2, \ldots, a_7$ et le nombre de mots de S de poids 1, 5, 9 et 13 donne les coefficients a_{11}, a_{10}, a_9 et a_8. Par exemple, pour le code duadique de longueur 90 et $S_1 = \{0, 1, 3, 5, 13\}$ on trouve $B_1 = 1$, $B_5 = B_9 = 0$, $B_{13} = 89$.

Corollaire 6.1 *(*Bornes sur la distance minimale*) Soit $C[n, n/2]$ un code auto-dual ou formellement auto-dual de distance minimale d_C. Les bornes connues pour la distance minimale sont ([87] ou [84, p. 138]) :*

1. *si C est binaire de Type I :*

$$d_C \leq \begin{cases} 4\lfloor \frac{n}{24} \rfloor + 4, & si\ n \neq 22 \pmod{24}, \\ 4\lfloor \frac{n}{24} \rfloor + 6, & si\ n = 22 \pmod{24}\ ; \end{cases}$$

2. *si C est binaire de Type II :*

$$d_C \leq 4\lfloor \frac{n}{24} \rfloor + 4\ ;$$

3. *si C est binaire formellement auto-dual :*

$$d_C \leq 2\lfloor \frac{n}{8} \rfloor + 2\ ;$$

4. *si C est ternaire de Type III :*

$$d_C \leq 3\lfloor \frac{n}{12} \rfloor + 3.$$

Définition 6.5 *Un code auto-dual qui, selon son type, atteint une des précédentes bornes, est appelé* extrémal. *Un code est* optimal *si son poids minimum est le plus grand connu pour sa longueur et sa dimension.*

Soulignons que si le code est extrémal, son polynôme énumérateur est unique et donc fixé par la théorie des invariants. C'est pour cela que nous ne nous intéressons dans ce chapitre qu'à des codes non-extrémaux.

6.4 Un algorithme de calcul du nombre des mots de poids donné

La section précédente nous a montré que le calcul du polynôme énumérateur des poids se réduit au calcul d'un nombre suffisant des mots de petit poids. Pour le comptage des mots de poids w fixé nous utilisons un algorithme qui est usuellement employé pour des codes auto-duaux.

Si $C[n,k,d]$ est un code linéaire sur $\mathbb{F}_q$ avec $n = 2k$, on calcule deux matrices génératrices de C, $G' = (I', A')$ et $G'' = (A'', I'')$, ayant des ensembles d'information disjoints de rang maximal, i.e. les deux matrices diagonales I' et I'' sont toutes les deux de rang k. Les matrices G' et G'' existent toujours si C est auto-dual ; de même pour tous les codes non auto-duaux que nous considérons dans ce chapitre.

Pour $1 \leq i \leq k$ notons par G'_i et G''_i la i-ème ligne de G', resp. G''. Pour déterminer le nombre de mots du code de poids w nous comptons :

1. pour tout t tel que $1 \leq t \leq w/2$ et tous les $a_i \in \mathbb{F}_q^*$, le nombre de mots $\mathbf{c}$ de poids w qui sont des combinaisons linéaires de t lignes de G' :

$$\mathbf{c} = a_1 G'_{i_1} + a_2 G'_{i_2} + \cdots + a_t G'_{i_t}\,,$$

2. pour tout t tel que $1 \leq t < w/2$ et tous les $a_i \in \mathbb{F}_q^*$, le nombre de mots $\mathbf{c}$ de poids w qui sont des combinaisons linéaires de t lignes de G'' :

$$\mathbf{c} = a_1 G''_{i_1} + a_2 G''_{i_2} + \cdots + a_t G''_{i_t}\,.$$

Cette méthode a été généralisée par Zimmermann *et al.* [9] pour le calcul de la distance minimale des codes linéaires, pour toutes valeurs de k et n. L'adaptation au calcul du nombre de mots de poids donné des codes binaires ou ternaires de rendement $1/2$ est due à A. Wassermann et est présentée par la suite.

Pour une sélection fixée $i_1, i_2, \ldots, i_t$ de lignes de la matrice génératrice, nous devons tester toutes les combinaisons linéaires $a_1 G'_{i_1} + a_2 G'_{i_2} + \cdots + a_t G'_{i_t}$ et $a_1 G''_{i_1} + a_2 G''_{i_2} + \cdots + a_t G''_{i_t}$, avec $a_i \in \mathbb{F}_q^*$, $1 \leq i \leq t$. Comme on peut fixer $a_1 = 1$, cela réduit à $(q-1)^{t-1}$ le nombre des combinaisons à tester pour chaque sélection de t lignes. Le nombre des mots ainsi obtenus sera par la suite multiplié par $(q-1)$.

La génération de toutes les combinaisons de t lignes $i_1, i_2, \ldots, i_t$ à partir des k lignes de G', resp. G'', est réalisée via l'algorithme *revolving door* dû à Nijenhuis et Wilf [79]. Comme toute énumération suivant le *Code de Gray*, à chaque pas on change seulement une ligne de l'ensemble $\{i_1, i_2, \ldots, i_t\}$.

La suite de t-uplets ainsi obtenue possède la propriété clef que la combinaison $i_1, i_2, \ldots, i_t$ de lignes de la matrice génératrice est visitée après que exactement :

$$m = \binom{i_t + 1}{t} - \binom{i_{t-1} + 1}{t-1} + \cdots + (-1)^t \binom{i_2 + 1}{2} - (-1)^t \binom{i_1 + 1}{1} - t \,(\mathrm{mod} 2) \qquad (6.12)$$

autres combinaisons ont été visitées (voir Lüneburg [67]).

Cela permet une parallélisation facile du problème : pour $1 \leq t \leq w/2$ on partage le calcul de $\binom{k}{t}$ pas d'énumération en $\lfloor \binom{k}{t}/T \rfloor$ gros morceaux de taille T. Par conséquent, la r-ième tâche pour $1 \leq r \leq \lfloor \binom{k}{t}/T \rfloor$ commence au pas d'énumération $(r-1)T$. La formule (6.12) nous permet de déduire facilement la combinaison $i_1, i_2, \ldots, i_t$ des lignes de la matrice génératrice en fonction de m et t. Pour la distribution des tâches on utilise `autoson` de McKay [69].

Cet algorithme est beaucoup plus rapide que la méthode de calcul employée par MAGMA [71] via la commande `NumberOfWords`. Cette méthode agissant pour des codes arbitraires, on utilise par défaut seulement un ensemble d'information. Cela signifie que MAGMA énumère $O(\binom{k}{w})$ mots de code, alors que pour notre méthode (adaptée aux codes de longueur $n = 2k$) seulement $O(\binom{k}{w/2})$ pas sont nécessaires. De plus, l'algorithme peut être parallélisé sur un nombre arbitraire d'ordinateurs. Les résultats donnés ici ont été obtenus par des calculs simultanés sur 40 ordinateurs. Enfin, notre implémentation semble être au moins deux fois plus rapide que celle de MAGMA, version V2.9-6.

6.5 Résultats numériques

Dans les tables présentées dans cette section, nous listons en premier la longueur et le type des codes. Si l'énumérateur des poids est connu, on donne la méthode de calcul :

1. "ex" signifie que le code est extrémal, donc d'énumérateur connu ;
2. "M" signifie que l'énumérateur est calculable avec Magma ;
3. "$\star$" désigne un résultat que nous avons obtenu, et qui n'était pas connu auparavant ;
4. "-" désigne un résultat inconnu.

Dans la liste des codes binaires tels que $n \leq 152$, seul l'énumérateur du code de longueur 138 n'est pas encore calculé. Pour les codes ternaires nous indiquons seulement notre contribution. Pour les résultats détaillés voir l'Annexe C.

Codes binaires

n	C	W_C
8	XQR	ex.
18	XQR	M
24	XQR	M
32	XQR & D	ex. & M
40	QDC	ex.
42	XQR	M
48	XQR	ex.
50	D	M
56	QDC	ex.
64	QDC	ex.
72	XQR	M
74	XQR & D	$\star$
80	XQR	ex.
88	QDC	ex.
90	XQR & D	$\star$
98	XQR	$\star$
104	XQR	ex.
108	QDC	$\star$
114	XQR & D	$\star$
120	D & QDC	$\star$
128	XQR & D	$\star$
138	XQR	–
152	XQR & D	$\star$

Codes ternaires

n	C	W_C
72	XQR	$\star$
84	XQR & QDC	$\star$
88	QDC	$\star$
96	QDC	$\star$

A

Distances minimales des codes EBCH$^\perp$

A.1 Résultats et méthodes utilisées

Dans cette section nous listons les distances minimales (connues auparavant ou nouvelles) et les méthodes utilisées pour les obtenir, à l'exception de la construction carrée modifiée (à laquelle sera largement consacrée la section suivante). Dans les tableaux suivants, les paramètres sont :

1. δ, la distance construite du code BCH ;
2. k, la dimension du code BCH$^\perp$ correspondant ;
3. d désigne une borne supérieure ou la distance duale du code BCH$^\perp$; elle reste inchangée par le passage au code EBCH$^\perp$.

Le symbole $**$ désigne une borne de Schaub atteinte. Les résultats sont marqués par $\mathcal{M}$ s'ils sont obtenus avec Magma, par $\mathcal{A}$ si c'est l'algorithme Canteaut-Chabaud qui a été utilisé et par "we" si on connaît le polynôme énumérateur, tandis que "monotonie" signifie que le lemme 4.1 est utilisé. Pour plus de détails voir les Sections 4.1 et 4.5.

Longueur 255

δ	k	d	Remarques	Méthodes utilisées	Borne de Schaub
3	8	128**	dist min	we, $\mathcal{M}$	128
5	16	112**	dist min	we, $\mathcal{M}$	112
7	24	96**	dist min	we, $\mathcal{M}$	96
9	32	96	dist min	we, $\mathcal{M}$, [89]	88
11	40	64**	dist min	we, $\mathcal{A}$, $\mathcal{M}$	64
13	48	64**	dist min	we, $\mathcal{A}$, $\mathcal{M}$	64
15	56	64	dist min	[91]	60
17	64	60	borne sup	$\mathcal{A}$	42
19	68	60	borne sup	$\mathcal{A}$	42
21	76	56	borne sup	$\mathcal{A}$	40
23	84	48	borne sup	$\mathcal{A}$	32
25	92	48	borne sup	$\mathcal{A}$	32
27	100	48	borne sup	$\mathcal{A}$	32
29	108	40	borne sup	$\mathcal{A}$	28
31	116	36	borne sup	$\mathcal{A}$	26
37	124	36	borne sup	$\mathcal{A}$	22
39	132	32	borne sup	$\mathcal{A}$	22
43	140	28	borne sup	$\mathcal{A}$	20
45	148	26	borne sup	$\mathcal{A}$	20
47	156	16**	dist min	($\mathcal{A}$ ou $\mathcal{M}$)+[2]	16
51	164	16**	dist min	$\mathcal{A}$, monotonie	16
53	168	16**	dist min	$\mathcal{A}$, monotonie	16
55	176	16	dist min	monotonie	
59	184	16	dist min	monotonie	
61	192	16	dist min	we	
63	200	12	dist min	we	
85	208	12	dist min	we	
87	210	12	dist min	we	
91	218	10	dist min	$\mathcal{M}$, we	
95	226	8	dist min	$\mathcal{M}$, we	
111	234	6	dist min	$\mathcal{M}$, we	
119	242	4	dist min	$\mathcal{M}$, we	
127	246	4	dist min	$\mathcal{M}$, we	

Longueur 511

δ	k	d	Remarques	Méthodes utilisées
3	9	256	dist min	$\mathcal{M}$
5	18	240	dist min	$\mathcal{M}$
7	27	224	dist min	$\mathcal{M}$
9	36	196	dist min	$\mathcal{M}$
11	45	192	dist min	$\mathcal{M}$
13	54	184	borne sup	$\mathcal{A}$
15	63	168	borne sup	$\mathcal{A}$
17	72	164	borne sup	$\mathcal{A}$
19	81	128	borne sup	$\mathcal{A}$
21	90	128	borne sup	monotonie
23	99	128	borne sup	monotonie
25	108	128	borne sup	$\mathcal{A}$
27	117	112	borne sup	$\mathcal{A}$
29	126	112	borne sup	$\mathcal{A}$
31	135	108	borne sup	$\mathcal{A}$
35	144	108	borne sup	$\mathcal{A}$
37	153	64	borne sup	$\mathcal{A}$
39	162	64	borne sup	monotonie
41	171	64	borne sup	$\mathcal{A}$
43	180	64	borne sup	monotonie
45	189	64	borne sup	monotonie
47	198	64	borne sup	monotonie
51	207	64	borne sup	$\mathcal{A}$
53	216	64	borne sup	$\mathcal{A}$
55	225	64	borne sup	$\mathcal{A}$
57	234	82	borne sup	$\mathcal{A}$
59	243	56	borne sup	$\mathcal{A}$
61	252	56	borne sup	$\mathcal{A}$
63	261	70	borne sup	$\mathcal{A}$
73	270	66	borne sup	$\mathcal{A}$
75	273	64	borne sup	$\mathcal{A}$
77	282	62	borne sup	$\mathcal{A}$
79	291	48	borne sup	$\mathcal{A}$
83	300	54	borne sup	$\mathcal{A}$
85	309	52	borne sup	$\mathcal{A}$

δ	k	d	Remarques	Méthodes utilisées
87	318	32	borne sup	$\mathcal{A}$
91	327	32	borne sup	$\mathcal{A}$
93	336	32	borne sup	monotonie
95	345	32	borne sup	monotonie
103	354	32	borne sup	monotonie
107	363	32	borne sup	$\mathcal{A}$
109	372	28	borne sup	$\mathcal{A}$
111	381	28	borne sup	$\mathcal{A}$
117	390	26	borne sup	$\mathcal{A}$
119	399	24	borne sup	$\mathcal{A}$
123	408	22	borne sup	$\mathcal{A}$
125	417	20	borne sup	$\mathcal{A}$
127	426	16	borne sup	$\mathcal{A}$
171	435	16	borne sup	$\mathcal{A}$
175	444	14	borne sup	$\mathcal{A}$
183	453	8	dist min	$\mathcal{M}$+monotonie
187	462	8	dist min	$\mathcal{M}$+monotonie
191	471	8	dist min	$\mathcal{M}$+monotonie
219	480	8	dist min	$\mathcal{M}$+monotonie
223	483	8	dist min	$\mathcal{M}$
239	492	6	dist min	$\mathcal{M}$
255	501	4	dist min	$\mathcal{M}$

A.2 Décomposition LTSC

Dans les tableaux suivants, les paramètres sont :

1. δ, la distance construite du code BCH ;
2. k, la dimension du code EBCH$^{\perp}$ correspondant ;
3. $d^{\perp}_{\max}$ désigne une borne supérieure de la distance duale (en *italique*) ou la distance duale (obtenue dans tous les cas en utilisant Magma [71]) ;
4. A et B sont les codes obtenus par la décomposition LTSC du code EBCH$^{\perp}(n, \delta)$ *mis en ordre standard des bits*;
5. $(A/B, \widetilde{A/B})$ est le code obtenu en considérant le complément de EBCH$^{\perp}$ par la somme directe $B \oplus B$;
6. l'avant-dernière colonne dans nos tables indique le nombre de colonnes nulles dans une matrice génératrice du code $(A/B, \widetilde{A/B})$. Son importance a été analysée en Section 4.2 ;
7. la dernière colonne donne l'ordre du plus grand code de Reed-Muller contenu dans EBCH$^{\perp}(n, \delta)$. Son importance a été analysée en Section 4.4.

Pour toutes les longueurs, les valeurs en *italique* sont des bornes supérieures et non pas des distances minimales exactes. En troisième colonne une notation du type *106*(*108*) signifie que la borne 108 obtenue par l'algorithme probabiliste à été affinée par la borne 106, obtenue par la LTSC. Par contre, en sixième colonne, c'est l'algorithme probabiliste qui donne une borne meilleure ; par exemple, dans le code $[512, 101, \mathit{32}(\mathit{28})]$ un mot de poids 28 a été trouvé par l'algorithme probabiliste. De plus, quand on note $[512, 76, 22-24]$, cela signifie qu'on a prouvé par Magma que la distance minimale appartient à l'intervalle $[22, 24]$.

Les codes marqués par "$*$" sont des codes de Reed-Muller, *à une ligne de la matrice génératrice près*. Par exemple, $[64, 7+1, 28]^\star$ signifie que G_B est la matrice génératrice du code RM$[64, 7, 28]$, à une ligne près.

Pour plus de détails voir la Section 4.5.

A.2.1 Longueur 32

δ	k	$d^\perp_{\max}$	A	B	$(A/B, \widetilde{A/B})$	nb. col. nulles	RM
3	6	16	[16,5,8]*	[16,1,16]*	[32,4,16]	2	1
5	11	12	[16,10,4]	[16,1,16]*	[32,9,12]	2	1
7	16	8	[16,11,4]*	[16,5,8]*	[32,6,8]	10	2
11	21	6	[16,15,2]*	[16,6,6]	[32,9,6]	12	2
15	26	4	[16,15,2]*	[16,11,4]*	[32,4,4]	22	3

A.2.2 Longueur 64

δ	k	$d^\perp_{\max}$	A	B	$(A/B, \widetilde{A/B})$	nb. col. nulles	RM
3	7	32	[32,6,16]*	[32,1,32]*	[64,5,32]	2	1
5	13	24	[32,12,8]	[32,1,32]*	[64,11,24]	2	1
7	19	16	[32,16,8]*	[32,3,16]	[64,13,16]	6	1
9	25	14	[32,22,4]	[32,3,16]	[64,19,14]	6	1
11	28	14	[32,22,4]	[32,6,16]*	[64,16,14]	12	2
13	34	12	[32,26,4]*	[32,8,12]	[64,18,12]	16	2
15	40	8	[32,26,4]*	[32,14,8]	[64,12,8]	28	2
21	46	8	[32,31,2]*	[32,15,8]	[64,16,8]	30	2
23	48	6	[32,31,2]*	[32,16+1,6]*	[64,14,6]	34	3
27	54	4	[32,31,2]*	[32,23,4]	[64,8,4]	46	3
31	57	4	[32,31,2]*	[32,26,4]*	[64,5,4]	52	4

A.2.3 Longueur 128

δ	k	$d^{\perp}_{\max}$	A	B	$(A/B, \widetilde{A/B})$	nb.col.nulles	RM
3	8	64	[64,7]*	[64,1,64]*	[128,6,64]	2	1
5	15	56	[64,14]	[64,1,64]*	[128,13,56]	2	1
7	22	48	[64,21]	[64,1,64]*	[128,20,48]	2	1
9	29	44	[64,28]	[64,1,64]*	[128,27,44]	2	1
11	36	32	[64,29]	[64,7,32]*	[128,22,32]	14	2
13	43	32	[64,36]	[64,7,32]*	[128,29,32]	14	2
15	50	28	[64,42]*	[64,7+1,28]*	[128,34,28]	16	2
19	57	22	[64,49]	[64,7+1,28]*	[128,41,22]	16	2
21	64	22	[64,49]	[64,15,24]	[128,34,22]	30	2
23	71	16	[64,49]	[64,22,16]*	[128,27,16]	44	3
27	78	16	[64,56]	[64,22,16]*	[128,34,16]	44	3
29	85	14	[64,57]*	[64,28,14]	[128,29,14]	56	3
31	92	12	[64,57]*	[64,35,12]	[128,22,12]	70	3
43	99	10	[64,63]*	[64,36,10]	[128,27,10]	72	3
47	106	8	[64,63]*	[64,42+1,8]*	[128,20,8]	86	4
55	113	6	[64,63]*	[64,50,6]	[128,13,6]	100	4
63	120	4	[64,63]*	[64,57,4]*	[128,6,4]	114	5

A.2.4 Longueur 256

δ	k	$d^{\perp}_{\max}$	A	B	$(A/B, \widetilde{A/B})$	nb.col.nulles	RM
3	9	128	[128,8,64]*	[128,1,128]*	[256,7,128]	2	1
5	17	112	[128,16,48]	[128,1,128]*	[256,15,112]	2	1
7	25	96	[128,24,32]	[128,1,128]*	[256,23,96]	2	1
9	33	96	[128,32,32]	[128,1,128]*	[256,31,96]	2	1
11	41	64	[128,37,*32*]	[128,4,64]	[256,33,64]	8	1
13	49	64	[128,45,*24*]	[128,4,64]	[256,41,64]	8	1
15	57	64	[128,53,*24*]	[128,4,64]	[256,49,64]	8	1
17	65	*60*	[128,61,18]	[128,4,64]	[256,57,*60*]	8	1
19	69	*60*	[128,61,18]	[128,8,64]*	[256,53,*60*]	16	2
21	77	*56*	[128,69,*16*]	[128,8,64]*	[256,61,*56*]	16	2
23	85	*48*	[128,72,*16*]	[128,13,48]	[256,59,*48*]	26	2
25	93	*48*	[128,80,*12*]	[128,13,48]	[256,67,*48*]	26	2
27	101	*48*	[128,80,*12*]	[128,21,48]	[256,59,*48*]	42	2
29	109	*40*	[128,88,8]	[128,21,48]	[256,67,*40*]	42	2
31	117	*36*	[128,96,8]	[128,21,48]	[256,75,*36*]	42	2
37	125	*36*	[128,104,6]	[128,21,48]	[256,83,*36*]	42	2
39	133	*32*	[128,104,6]	[128,29,32]*	[256,75,*32*]	58	3

δ	k	$d^{\perp}_{\max}$	A	B	$(A/B, \widetilde{A/B})$	nb.col.nulles	RM
43	141	*28*	[128,104,6]	[128,37,32]	[256,67,*28*]	74	3
45	149	*26*	[128,107,6]	[128,42,28]	[256,65,*26*]	84	3
47	157	16	[128,107,6]	[128,50,16]	[256,57,24]	100	3
51	165	16	[128,115,4]	[128,50,16]	[256,65,20]	100	3
53	169	16	[128,115,4]	[128,54,16]	[256,61,16]	108	3
55	177	16	[128,115,4]	[128,62,16]	[256,53,16]	124	3
59	185	16	[128,120,4]*	[128,65,16]	[256,55,16]	130	3
61	193	16	[128,120,4]*	[128,73,16]	[256,47,16]	146	3
63	201	12	[128,120,4]*	[128,81,12]	[256,39,12]	162	3
85	209	12	[128,127,2]*	[128,82,12]	[256,45,12]	164	3
87	211	12	[128,127,2]*	[128,84,12]	[256,43,12]	168	4
91	219	10	[128,127,2]*	[128,92,10]	[256,35,10]	184	4
95	227	8	[128,127,2]*	[128,99+1,8]*	[256,27,8]	200	5
111	235	6	[128,127,2]*	[128,108,6]	[256,19,6]	216	5
119	243	4	[128,127,2]*	[128,116,4]	[256,11,4]	232	5
127	247	4	[128,127,2]*	[128,120,4]*	[256,7,4]	240	6

A.2.5 Longueur 512

δ	k	$d^{\perp}_{\max}$	A	B	$(A/B, \widetilde{A/B})$	nb.col.nulles	RM
3	10	256	[256,9]*	[256,1,256]*	[512,8]	2	1
5	19	240	[256,18]	[256,1,256]*	[512,17]	2	1
7	28	224	[256,27]	[256,1,256]*	[512,26]	2	1
9	37	196	[256,36]	[256,1,256]*	[512,35]	2	1
11	46	192	[256,45]	[256,1,256]*	[512,44,*192*]	2	1
13	55	*184*	[256,54]	[256,1,256]*	[512,53,*184*]	2	1
15	64	*168*	[256,63]	[256,1,256]*	[512,62,*168*]	2	1
17	73	*164*	[256,72]	[256,1,256]*	[512,71,*168(164)*]	2	1
19	82	*128*	[256,73]	[256,9,128]*	[512,64,*128*]	18	2
21	91	*128*	[256,82]	[256,9,128]*	[512,73]	18	2
23	100	*128*	[256,91]	[256,9,128]*	[512,82]	18	2
25	109	*128*	[256,100]	[256,9,128]*	[512,91,*128*]	18	2
27	118	*112*	[256,108]	[256,9+1,112]*	[512,98,*112*]	20	2
29	127	*112*	[256,117]	[256,9+1,112]*	[512,107,*112*]	20	2
31	136	*108*	[256,126]	[256,9+1,112]*	[512,116,*108*]	20	2
35	145	*106(108)*	[256,135]	[256,9+1,112]*	[512,125,*106*]	20	2

δ	k	$d^{\perp}_{\max}$	A	B	$(A/B, \widetilde{A/B})$	nb.col.nulles	RM
37	154	*64*	[256,135]	[256,19,64]	[512,116,*64*]	38	2
39	163	*64*	[256,138]	[256,25,64]	[512,113]	50	2
41	172	*64*	[256,147]	[256,25,64]	[512,122]	50	2
43	181	*64*	[256,147]	[256,34,64]	[512,113]	68	2
45	190	*64*	[256,156]	[256,34,64]	[512,122]	68	2
47	199	*64*	[256,165]	[256,34,64]	[512,131]	68	2
51	208	*64*	[256,174]	[256,34,64]	[512,140]	68	2
53	217	*64*	[256,181]	[256,36,64]	[512,145]	72	2
55	226	*64*	[256,181]	[256,45,64]	[512,136,*64*]	90	2
57	235	*64(82)*	[256,190]	[256,45,64]	[512,145,*64*]	90	2
59	244	*56*	[256,190]	[256,54,56]	[512,136,*56*]	108	2
61	253	*56*	[256,199]	[256,54,56]	[512,145,*56*]	108	2
63	262	*56(70)*	[256,208]	[256,54,56]	[512,154]	108	2
73	271	*56(66)*	[256,217]	[256,54,56]	[512,163]	108	2
75	274	*56(64)*	[256,217]	[256,57,*56*]	[512,160]	114	3
77	283	*56(62)*	[256,217]	[256,66,*56*]	[512,151,*62*]	132	3
79	292	*48*	[256,217]	[256,75,*48*]	[512,142,*48*]	150	3
83	301	*48(54)*	[256,217]	[256,84,*48*]	[512,133,*48*]	168	3
85	310	*48(52)*	[256,217]	[256,93,*48*]	[512,124,*48*]	186	3
87	319	*32*	[256,217]	[256,102,*32*]	[512,115,*32*]	204	4
91	328	*32*	[256,225]	[256,103,*32*]	[512,122,*32*]	206	4
93	337	*32*	[256,228]	[256,109,*32*]	[512,119,*32*]	218	4
95	346	*32*	[256,228]	[256,118,*32*]	[512,110,*32*]	236	4
103	355	*32*	[256,237]	[256,118,*32*]	[512,119,*32*]	236	4
107	364	*32*	[256,237]	[256,127,*32*]	[512,110,*32*]	254	4
109	373	*28*	[256,237]	[256,136,*28*]	[512,101,*32(28)*]	272	4
111	382	*28*	[256,237]	[256,145,*28*]	[512,92,*28*]	290	4
117	391	*26*	[256,246]	[256,145,*28*]	[512,101,*28*]	290	4
119	400	*24*	[256,246]	[256,154,*24*]	[512,92,*24*]	308	4
123	409	*22*	[256,247]*	[256,162,*24*]	[512,85,20-24]	324	4
125	418	*20*	[256,247]*	[256,171,*20*]	[512,76,22-24]	342	4
127	427	*16*	[256,247]*	[256,180,*16*]	[512,67,20]	360	4
171	436	*16*	[256,255]*	[256,181,*16*]	[512,74,20]	362	4
175	445	*14*	[256,255]*	[256,190,*16*]	[512,65,16]	380	5
183	454	8	[256,255]*	[256,199,8]	[512,56,14]	398	5
187	463	8	[256,255]*	[256,208,8]	[512,47,12]	416	5
191	472	8	[256,255]*	[256,217,8]	[512,38,8]	434	5
219	481	8	[256,255]*	[256,226,8]	[512,29,8]	452	5
223	484	8	[256,255]*	[256,229,8]	[512,26,8]	458	6
239	493	6	[256,255]*	[256,238,6]	[512,17,6]	476	6
255	502	4	[256,255]*	[256,247,4]*	[512,8,4]	494	7

B

Codes cycliques auto-duaux

Dans cette section, nous énumérons tous les codes *c.a.d.* de longueur $n \leq 120$ inéquivalents et non-triviaux (voir Section 5.3.1). I désigne l'ensemble de définition du code, donné ici comme réunion de classes cyclotomiques. Nous notons par $\mathcal{M}$ un polynôme énumérateur obtenu en utilisant MAGMA, $\mathcal{G}$ un résultat obtenu via la méthode générale décrite en Section 5.3.2, *1*, resp. *2*, les cas spéciaux considérés en Section 5.3.3 et $\mathcal{S}$ si on utilise le code *ombre*. Le symbole "–" désigne un résultat inconnu. Quand on utilise plusieurs méthodes, la première est plus rapide que la deuxième (la deuxième que la troisième). Nous indiquons aussi les valeurs de $k_{A/B}$ pour tout code de longueur $n \geq 84$, comme mesure de la complexité du calcul (voir Section 5.3.2).

B.1 Énumération des codes

n	k	d	I	Équiv.	Méthodes utilisées
14	7	4	$(0)(1)^2$		$\mathcal{M}$, *2*
28	14	4	$(0)^2(1)^4$ $(0)^2(1)^3(3)$	inéq.	$\mathcal{M}$, *1* $\mathcal{M}$
30	15	6	$(0)(1)^2(3)(5)$		$\mathcal{M}$
42	21	4 4 6 8	$(0)(3)^2(5)^2(7)$ $(0)(1)(3)^2(5)(7)$ $(0)(1)^2(3)(7)(9)$ $(0)(1)^2(3)^2(7)$	inéq.	$\mathcal{M}$
46	23	8	$(0)(1)^2$		$\mathcal{M}$, *2*
56	28	4 4 4 6	$(0)^4(1)^8$ $(0)^4(1)^6(3)^2$ $(0)^4(1)^5(3)^3$ $(0)^4(1)^7(3)$	inéq.	$\mathcal{M}$
60	30	4 6	$(0)^2(1)^3(3)^2(5)^2(7)$ $(0)^2(1)^4(3)^2(5)^2$	inéq.	$\mathcal{M}$ $\mathcal{M}$, *1*
62	31	6	$(0)(1)^2(3)(5)(7)(11)$ $(0)(1)(3)^2(5)(11)(15)$ $(0)(1)(3)(7)(11)^2(15)$	éq.	$\mathcal{M}$
		8	$(0)(1)^2(3)^2(11)^2$ $(0)(1)^2(3)^2(5)^2$ $(0)(1)^2(7)^2(11)^2$	éq.	
			$(0)(1)^2(5)^2(7)^2$	inéq.	
		10	$(0)(1)^2(3)^2(5)(11)$ $(0)(1)^2(3)(7)(11)^2$ $(0)(1)(3)^2(5)^2(15)$	éq.	
			$(0)(1)^2(3)(5)^2(7)$ $(0)(1)^2(5)(7)^2(11)$ $(0)(1)(3)^2(11)^2(15)$	éq.	
70	35	4 4 6 8	$(0)(3)^2(5)^2(7)$ $(0)(1)(3)(5)^2(7)$ $(0)(3)^2(5)(7)(15)$ $(0)(1)^2(5)^2(7)$	inéq.	$\mathcal{S}$, $\mathcal{G}$
78	39	6	$(0)(3)(7)^2(13)$		$\mathcal{S}$, $\mathcal{G}$

n	k	d	I	Équiv.	Méthodes utilisées	$k_{A/B}$
84	42	4	$(0)^2(1)(3)^2(5)^3(7)^2(9)^2$	inéq.	$\mathcal{S}$	30
			$(0)^2(1)(3)^3(5)^3(7)^2(9)$		$\mathcal{S}$, $\mathcal{G}$	24
			$(0)^2(1)^2(3)^3(5)^2(7)^2(9)$		$\mathcal{S}$	36
			$(0)^2(1)^3(3)^3(5)(7)^2(9)$		$\mathcal{S}$, $\mathcal{G}$	24
			$(0)^2(1)^2(3)^4(5)^2(7)^2$		*1*, $\mathcal{S}$	30
			$(0)^2(3)^4(5)^4(7)^2$		*1*, $\mathcal{G}$, $\mathcal{S}$	6
		6	$(0)^2(3)^2(5)^4(7)^2(9)^2$		*1*, $\mathcal{S}$, $\mathcal{G}$	18
			$(0)^2(3)^3(5)^4(7)^2(9)$		$\mathcal{S}$, $\mathcal{G}$	12
			$(0)^2(1)(3)^4(5)^3(7)^2$		$\mathcal{S}$, $\mathcal{G}$	18
		8	$(0)^2(1)^4(3)^3(7)^2(9)$		$\mathcal{S}$, $\mathcal{G}$	12
			$(0)^2(1)^4(3)^4(7)^2$		*1*, $\mathcal{G}$, $\mathcal{S}$	6
			$(0)^2(1)^3(3)^4(5)(7)^2$		$\mathcal{S}$, $\mathcal{G}$	18
90	45	4	$(0)(1)(3)^2(5)(7)(9)(15)$	inéq.	$\mathcal{S}$	37
		6	$(0)(1)^2(3)(5)(9)(15)(21)$		$\mathcal{S}$, $\mathcal{G}$	21
		8	$(0)(1)^2(3)^2(5)(9)(15)$	éq.	$\mathcal{S}$, $\mathcal{G}$	13
			$(0)(3)^2(5)(7)^2(9)(15)$			
92	46	4	$(0)^2(1)^3(5)$	inéq.	$\mathcal{S}$, $\mathcal{G}$	24
		8	$(0)^2(1)^4$		*1*, $\mathcal{G}$, $\mathcal{S}$	2
94	47	12	$(0)(1)^2$		*2*, $\mathcal{G}$, $\mathcal{S}$	1
98	49	4	$(0)(1)^2(7)^2$	éq.	$\mathcal{G}$, $\mathcal{S}$	1
			$(0)(3)^2(7)^2$			
		4	$(0)(1)(3)(7)^2$	inéq.	$\mathcal{S}$	43
			$(0)(1)^2(7)(21)$		$\mathcal{G}$, $\mathcal{S}$	6
102	51	6	$(0)(3)(5)^2(9)(17)(19)^2$	éq.	$\mathcal{G}$, $\mathcal{S}$	19
			$(0)(3)(9)(11)^2(17)(19)^2$			
		6	$(0)(3)(5)(9)(11)(17)(19)^2$	éq.	$\mathcal{S}$	35
			$(0)(1)(3)(5)^2(9)(17)(19)$			
110	55	10	$(0)(1)^2(5)(11)$		$\mathcal{G}$	15
112	56	4	$(0)^8(1)^{16}$	inéq.	*1*, $\mathcal{G}$	8
			$(0)^8(1)^{12}(3)^4$		*1*	32
			$(0)^8(1)^{11}(3)^5$		–	38
			$(0)^8(1)^{10}(3)^6$		*1*	44
			$(0)^8(1)^9(3)^7$		–	50
		6	$(0)^8(1)^{15}(3)$		$\mathcal{G}$	14
			$(0)^8(1)^{14}(3)^2$		*1*, $\mathcal{G}$	20
			$(0)^8(1)^{13}(3)^3$		$\mathcal{G}$	26
120	60	4	$(0)^4(1)^6(3)^4(5)^4(7)^2$	inéq.	*1*	4
		4	$(0)^4(1)^5(3)^4(5)^4(7)^3$		–	52
		6	$(0)^4(1)^8(3)^4(5)^4$		*1*, $\mathcal{G}$	28
		6	$(0)^4(1)^7(3)^4(5)^4(7)$		–	36

B.2 Distributions des poids

Dans chaque tableau, la distribution des poids d'un code *c.a.d.* de paramètres $[n, k, d]$ est donnée comme indiqué ci-après, où A_i est le nombre des mots de code de poids i.

n k d I			
A_0	A_6	A_{12}	...
A_2	A_8	A_{14}	...
A_4	A_{10}	A_{16}	...

Nous donnons seulement la première moitié de chaque distribution car$A_i = A_{n-i}$ pour $k < i \leq n$.

$n = 62$ $k = 31$ $d = 6$ $I = (0)(1)^2(3)(5)(7)(11)$							
1	0	1085	22568	311333	8049336	91075985	326038160
0	155	5208	82615	1581000	31535959	197095024	417943395

$n = 62$ $k = 31$ $d = 8$ $I = (0)(1)^2(7)^2(11)^2$							
$n = 62$ $k = 31$ $d = 8$ $I = (0)(1)^2(5)^2(7)^2$							
1	0	930	17360	252743	8082816	92323580	321819680
0	0	0	24025	1614480	32783554	192876544	423946111

$n = 62$ $k = 31$ $d = 10$ $I = (0)(1)^2(3)^2(5)(11)$							
$n = 62$ $k = 31$ $d = 10$ $I = (0)(1)^2(3)(5)^2(7)$							
1	0	0	2046	253673	8579746	90300520	325634540
0	0	186	26195	1726390	31945624	195388164	419884739

$n = 70$ $k = 35$ $d = 4$ $I = (0)(3)^2(5)^2(7)$					
1	245	65170	5978770	228446855	3077043739
0	2205	340065	22461894	616936950	5085164070
70	13790	1526350	76478710	1496314970	6569095330

$n = 70$ $k = 35$ $d = 4$ $I = (0)(1)(3)(5)^2(7)$					
1	875	203350	8823850	200628470	3161244394
0	6510	841650	23872254	602009730	5099569860
70	40250	2939650	65917390	1542412910	6471357970

$n = 70$ $k = 35$ $d = 6$ $I = (0)(3)^2(5)(7)(15)$					
1	35	8155	1493170	203070245	3225252485
0	315	45435	9341346	650096685	5066814480
0	1687	247520	49280140	1629892225	6344325260

$n = 70$ $k = 35$ $d = 6$ $I = (0)(3)^2(5)(7)(15)$					
1	35	8155	1493170	203070245	3225252485
0	315	45435	9341346	650096685	5066814480
0	1687	247520	49280140	1629892225	6344325260

$n = 78$ $k = 39$ $d = 6$ $I = (0)(3)(7)^2(13)$						
1	13	1443	880165	288612571	12504256869	78830706591
0	0	9126	7686627	1262928969	28469649429	96501446535
0	117	82719	52793169	4431965967	52526886633	

$n = 84$ $k = 42$ $d = 4$ $I = (0)^2(1)(3)^2(5)^3(7)^2(9)^2$					
1	1554	1860705	505262072	69887628258	697484864076
0	12768	7517440	2011454592	165503767296	767373824832
21	83314	29991549	7461354552	324140581674	
224	424416	122806656	24671188864	523507543104	

$n = 84$ $k = 42$ $d = 4$ $I = (0)^2(1)(3)^3(5)^3(7)^2(9)$					
1	1218	1195425	485243640	70118324514	697926886188
0	7056	6070400	1984726464	165334780800	768399402336
21	35938	27458109	7503526584	323448956138	
168	208872	116186784	24876301856	522993644208	

$n = 84$ $k = 42$ $d = 4$ $I = (0)^2(1)^2(3)^3(5)^2(7)^2(9)$					
1	13902	14747733	1396887912	65734567794	706594138068
0	105840	55733888	3454118976	157611758976	781027337376
105	656782	183952461	8853684456	316732978562	
1512	3373992	535065888	24312872864	523024927152	

$n = 84$ $k = 42$ $d = 4$ $I = (0)^2(1)^3(3)^3(5)(7)^2(9)$					
1	210	328545	425414136	70496336610	699033966540
0	672	1084160	1977010560	164928933120	771306403776
21	12082	12019581	7728262200	321815287850	
0	56832	81467904	25351585280	521518287360	

$n = 84$ $k = 42$ $d = 4$ $I = (0)^2(1)^2(3)^4(5)^2(7)^2$					
1	1953	2245278	721240212	68849380614	700411416558
0	12852	10289888	2453477040	163005736992	771979417656
42	88249	46976433	8024077860	321456200576	
378	482922	195593160	24836165480	523020160236	

$n = 84$ $k = 42$ $d = 4$ $I = (0)^2(3)^4(5)^4(7)^2$					
1	3234	3225915	915438951	68534060805	709798638255
0	20664	14639660	3102665496	156417769620	789148566240
84	111475	63880614	9626703381	310569565061	
294	654156	252550410	26971506590	518177537766	

$n = 84$ $k = 42$ $d = 6$ $I = (0)^2(3)^2(5)^4(7)^2(9)^2$					
1	336	424284	313590536	70868975814	694496747448
0	2730	1704304	1597509144	168051888144	762650569596
0	21112	9065217	6880884984	326941918044	
56	109752	54532128	24428812576	524051784144	

$n = 84$ $k = 42$ $d = 6$ $I = (0)^2(3)^3(5)^4(7)^2(9)$					
1	462	407568	302212932	71002653834	694684231248
0	2520	1935248	1573365696	168008847216	763118448336
0	13762	9827181	6883362396	326628918392	
42	82242	53540928	24521594384	523793035332	

$n = 84$ $k = 42$ $d = 6$ $I = (0)^2(1)(3)^4(5)^3(7)^2$					
1	21	55524	291237156	71016365406	694885124262
0	84	397376	1588243440	167869716672	763555829112
0	1645	4948713	6952949172	326360945876	
0	8112	43090320	24611753264	523620503952	

$n = 84$ $k = 42$ $d = 8$ $I = (0)^2(1)^4(3)^3(7)^2(9)$					
1	84	84000	290490732	71052111354	694915883508
0	126	459872	1582050792	167896934688	763692566580
0	2632	5326461	6944968836	326302760168	
0	14664	44057496	24618489224	523523337624	

$n = 84$ $k = 42$ $d = 8$ $I = (0)^2(1)^4(3)^4(7)^2$					
1	840	490476	280405944	71454866070	695268056280
0	882	1028720	1510981752	168284717328	765467824428
0	19936	9560817	6858873288	325639354124	
0	109920	55951392	24678530912	522246394656	

$n = 84$ $k = 42$ $d = 8$ $I = (0)^2(1)^3(3)^4(5)(7)^2$					
1	147	181524	310292556	70878494190	694563405690
0	756	1176224	1600114992	167999800416	762774907320
0	5131	7941465	6900550812	326850756260	
42	30330	52042536	24461501960	524009506860	

$n = 90$ $k = 45$ $d = 4$ $I = (0)(1)(3)^2(5)(7)(9)(15)$					
1	9450	13473945	1813223640	134549925570	3398737384476
0	75789	55369610	4893055965	389003419800	4858971923130
45	508410	199384461	13732516440	960741998650	5805038357820
945	2839770	634463865	42473581464	1981324531170	

$n = 90$ $k = 45$ $d = 6$ $I = (0)(1)^2(3)(5)(9)(15)(21)$					
1	1035	771660	396777660	135020302110	3407685515226
0	5544	5001395	1827739305	401074673580	4841492300100
0	19635	24386481	8750492460	984621157300	5767378825050
105	98955	98422020	37536240444	2006273314350	

$n = 90$ $k = 45$ $d = 8$ $I = (0)(1)^2(3)^2(5)(9)(15)$					
$n = 90$ $k = 45$ $d = 8$ $I = (0)(3)^2(5)(7)^2(9)(15)$					
1	90	60840	253839000	136579092030	3402688124556
0	324	494090	1641902985	402533266680	4841872667340
0	1125	4089456	8764218180	983669295520	5774223013110
0	5805	33698880	38214655284	2001707619120	

$n = 92$ $k = 46$ $d = 4$ $I = (0)^2(1)^3(5)$				
1	0	9230705	49450497536	5308318676738
0	1771	71810048	172957605194	8253282611200
23	32384	472024147	517834149888	10774826946446
0	138391	2592321536	1323428029990	11779745437440
253	1036288	12173390757	2879080866816	

$n = 92$ $k = 46$ $d = 8$ $I = (0)^2(1)^4$				
1	0	7820736	40076129016	5347988400216
0	5152	133384176	159838165718	8241818764256
0	128018	529706744	513761216416	10725893263372
0	1537228	1651478752	1349868788360	11721627880172
2024	1303456	8058354865	2933929700240	

$n = 94$ $k = 47$ $d = 12$ $I = (0)(1)^2$					
1	0	713460	175954464	187722670014	8215339885456
0	0	0	1542500520	627078284520	13622492744520
0	25944	4661272	9258332600	1762462322168	19095552850440
0	0	18696976	46210744745	4161657460928	22639226329636

$n = 98$ $k = 49$ $d = 4$ $I = (0)(1)^2(7)^2$ or $I = (0)(3)^2(7)^2$				
1	0	109115160	121563518370	16605392913634
0	158466	327770114	390873902874	31812060417396
98	0	2061969588	1182621666597	52694947099340
0	4775589	8187782155	3176594460620	75685521734424
4459	5764801	32761408344	7615108136664	92146834111962

$n = 98$ $k = 49$ $d = 4$ $I = (0)(1)(3)(7)^2$				
1	237699	1027280541	201879508383	16280275231962
0	1750035	3641955135	467838217770	32193208514110
147	10739673	11532815567	1124801897093	54008425059582
2401	56359359	32824779981	2850045492574	76331495567866
26313	256807187	84641298917	7136052227178	90746960941182

$n = 98$ $k = 49$ $d = 4$ $I = (0)(1)^2(7)(21)$				
1	28910	146848933	128537856734	16461115997690
0	175273	649193013	403745371442	31895117048984
98	1117501	2694359570	1170637962781	53472407893618
343	5961683	10531138947	3107048572264	76153598061814
4459	30318015	38153578267	7523488401086	91107066819230

$n = 102$ $k = 51$ $d = 6$ $I = (0)(3)(5)^2(9)(17)(19)^2$						
1	0	16218	150131080	273470729823	33237035230752	297983470977964
0	0	116144	1270039944	1167443713152	73051704884960	348018171261408
0	136	1179120	9020333280	4210404590080	136719099234864	
17	1224	14573658	54066974624	12848063695407	218326519158768	

$n = 102$ $k = 51$ $d = 6$ $I = (0)(3)(5)(9)(11)(17)(19)^2$						
1	1020	2494818	633593740	263512910463	33300267896952	297978473504284
0	10200	11915504	2221557144	1153264391232	73056233697200	348177038839968
0	69496	49878000	9742160760	4212762748480	136612449095664	
17	444924	185947938	51560749964	12890785236687	218190709698168	

$n = 110$ $k = 55$ $d = 10$ $I = (0)(1)^2(5)(11)$						
1	0	8635	58191210	295899700360	90210198378520	2256803553280430
0	11	20350	662879635	1583684406920	253023043311830	3483758610875092
0	0	350735	6087524190	7164578410495	609286540729800	4650262600409470
0	110	4130890	46446936855	27528994147160	1262733186278025	5371694359511260

$n = 112$ $k = 56$ $d = 4$ $I = (0)^8(1)^{16}$					
1	38528	294063504	441869725304	138914640519536	5573828834412048
0	259308	1422983520	1595063914710	352521183659848	8157381952434064
112	1758912	6543623758	5393586983456	821341812891096	10289818749042336
392	10055388	28384660416	17075449225772	1741833277670752	11124583528113300
5880	56030520	115306237312	50531142515952	3315675672194893	

$n = 112$ $k = 56$ $d = 4$ $I = (0)^8(1)^{12}(3)^4$					
1	10976	76870248	179873111936	116070021577958	5698123712305920
0	65100	389548992	726459687177	329632399197472	8167789171987312
28	424320	1945455652	2822921257856	825474818560644	10139386656787584
224	2622459	9362523072	10495435006868	1807707005133568	10897461851848824
1722	14706048	42366010792	36536174474016	3445067285711611	

$n = 112$ $k = 56$ $d = 4$ $I = (0)^8(1)^{10}(3)^6$					
1	61824	569034424	521362451456	127309905116870	5658665764046848
0	509796	2461532416	1755971601545	333310537633920	8178512239701328
84	3602432	10009247332	5529146255360	808764816448652	10205615685461504
896	22224251	38974449920	16378348618044	1764277992128512	10987875543117816
6874	119712768	145894188664	46436462144128	3387582961225211	

$n = 112$ $k = 56$ $d = 6$ $I = (0)^8(1)^{15}(3)$					
1	4704	36764532	74641397936	109047338114680	5727172500183456
0	31948	178163832	360668095809	324871903703704	8165424749827912
0	220704	825104896	1741986261344	831171894751800	10098372196991792
56	1258131	3680361048	7853467343036	1830718035281408	10838898741901440
840	7000112	16242238852	31443379911888	3481073914998827	

$n = 112$ $k = 56$ $d = 6$ $I = (0)^8(1)^{14}(3)^2$					
1	2184	17700592	66084867632	109185216286538	5729096199693664
0	12432	79547104	348888945881	324611249807208	8166647909834848
0	98960	405850396	1746027619280	830366760194720	10097186737619072
56	711179	2237065376	7924891411920	1830072512601760	10836337135356936
406	3901296	12266839984	31617376658392	3481743584014619	

$n = 112$ $k = 56$ $d = 6$ $I = (0)^8(1)^{13}(3)^3$					
1	1176	9858240	61009204928	109229542640974	5729893832476800
0	8512	52315536	341794164201	324461948048392	8167108845099904
0	53376	293423956	1749851723648	830011407732864	10096682863790432
56	326235	1724200464	7966413234752	1829845444808192	10835298421714200
210	1862336	10326533056	31701780996648	3482080665601979	

$n = 120$ $k = 60$ $d = 4$ $I = (0)^4(1)^6(3)^4(5)^4(7)^2$					
1	205660	6459958575	24089676909390	15073908815471625	168056732014715344
0	1517520	28377105120	79254129926160	31546771846319952	
30	10039725	122012187480	255659941531575	57717931330788330	
560	59168880	502980933600	800913328847280	92225798295135680	
3795	307296426	1950458359485	2360269740830820	128768071390771155	
25872	1444336800	7060702657440	6326612233734000	157243432752002880	

$n = 120$ $k = 60$ $d = 6$ $I = (0)^4(1)^8(3)^4(5)^4$					
1	28630	1184060325	11918910240600	15248630149040400	167471194716366184
0	180240	6587372400	50090807526900	31879738212957492	
0	1525650	35035593900	200170926790170	58090618745804400	
140	10077900	164859641520	722644319744400	92434630633864400	
1380	51305016	699069058275	2294610160744650	128648094559335255	
7392	235421400	2858699184600	6345105496867200	156795136288865760	

C

Codes duadiques et QDC

C.1 Poids minimum

Dans cette section nous donnons le nombre des mots A_i nécessaires pour le calcul des énumérateurs des poids des codes auxquels nous nous sommes intéressés. Notons qu'un espace vide dans les tableaux signifie que le coefficient concerné n'est pas nécessaire pour le calcul des polynômes énumérateurs. Pour montrer l'efficacité de l'algorithme, on n'a pas pris en compte, pour les codes de Type I, les améliorations apportées par l'utilisation du code *ombre* (voir Section 6.3) : ainsi, quelques coefficients de plus pourraient être supprimés des tables.

L'énumération des codes duadiques est faite suivant l'ensemble S_1 des classes cyclotomiques [86]. Chaque classe cyclotomique est donnée par son plus petit représentant. Tous les codes listés ici sont inéquivalents (voir [86] et [47]) ; toutes les distances minimales sont exactes et sont les mêmes que celles probabilistes en [86], ainsi que celles exactes obtenues en [94]. Dans tous nos tableaux, 'I','II' ou 'iso' désignent un code de Type I, de Type II ou isodual. La notation XQ_{n-1} est pour un code résidu quadratique étendu de longueur n et $\mathcal{B}_p(r,s,t)$ est un code QDC construit comme en Section 6.2.

C.1.1 Codes binaires

n	S_1	Type	Poids							
			10	12	14	16	18	20	22	24
74	0, 1, 3, 5, 11	I	73	876	876	163812	504576			
74	0, 1, 3, 5, 13	I	73	0	0	174324	515088			
74	0, 1, 5, 9, 17	iso	0	657	8103	86724	1061274			
74	XQ_{73}	iso	0	0	8103	89133	1093905			
90	0, 1, 3, 5, 13	I	0	89	0	18601	94963	4529833	14729055	
90	0, 1, 3, 5, 19	I	0	89	0	18601	94963	4529833	14729055	
90	0, 1, 3, 11, 33	iso	0	0	0	14685	203632	2910567	30374454	
90	XQ_{89}	iso	0	0	0	0	274120	2819520	30530115	
98	XQ_{97}	iso	0	0	0	28518	80801	931588	15038492	166982396

n	S_1	Type	Poids			
			20	22	24	26
108	$\mathcal{B}_{53}(0,1,0)$	iso	260442	5804136	74455407	746650008

n	S_1	Type	Poids						
			16	18	20	22	24	26	28
114	XQ_{113}	iso	25764	0	450870	2795394	36685789	468994974	4741799790
114	0, 1, 3	iso	0	4746	139216	2499560	38153885	473211004	4773745342

n	S_1	Type	Poids	
			16	20
120	$\mathcal{B}_{59}(0,0,1)$	II	0	71862
128	1, 3, 9, 11, 13, 15, 21, 27, 47	II	94488	0
128	1, 3, 5, 9, 11, 13, 15, 21, 27	II	127	23114
128	1, 3, 9, 13, 15, 19, 21, 29, 47	II	1143	59563
128	1, 3, 7, 9, 11, 13, 19, 21, 47	II	3048	57785
128	1, 3, 7, 9, 11, 13, 21, 27, 47	II	127	8890
128	1, 3, 5, 7, 9, 13, 19, 21, 29	II	1016	0
128	1, 3, 15, 21, 23, 27, 29, 47, 55	II	127	25781
128	1, 3, 5, 7, 9, 19, 21, 23, 29	II	127	14224
128	1, 3, 5, 7, 9, 11, 21, 23, 27	II	127	17780
128	1, 3, 7, 9, 11, 21, 23, 27, 47	II	1016	113792
128	1, 3, 7, 9, 11, 19, 21, 23, 47	II	381	36449
128	1, 3, 7, 9, 13, 21, 27, 29, 47	II	127	22225
128	1, 3, 5, 9, 15, 21, 23, 27, 29	II	889	23114
128	1, 3, 9, 13, 15, 21, 27, 29, 47	II	0	33782
128	1, 3, 5, 9, 13, 15, 21, 27, 29	II	0	26670
128	1, 3, 9, 15, 21, 23, 27, 29, 47	II	0	19558
128	1, 3, 9, 11, 15, 21, 23, 27, 47	II	0	14224
128	1, 3, 7, 9, 21, 23, 27, 29, 47	II	0	38227
128	1, 3, 5, 9, 13, 15, 19, 21, 29	II	0	22225
128	1, 3, 9, 11, 13, 15, 19, 21, 47	II	0	22225
128	1, 3, 9, 15, 19, 21, 23, 29, 47	II	0	21336
128	1, 3, 5, 9, 15, 19, 21, 23, 29	II	0	24892
128	1, 3, 11, 13, 15, 21, 27, 47, 55	II	0	21336
128	1, 3, 5, 7, 9, 11, 13, 19, 21	II	0	29337
128	1, 3, 5, 7, 11, 13, 21, 27, 55	II	0	14224
128	1, 3, 5, 7, 21, 23, 27, 29, 55	II	0	25781
128	1, 3, 5, 7, 9, 21, 23, 27, 29	II	0	25781
128	1, 3, 5, 7, 9, 13, 21, 27, 29	II	0	32004
128	XQ_{127}	II	0	56896
128	1, 3, 5, 7, 9, 11, 13, 21, 27	II	0	14224

n	S_1	Type	Poids							
			4	6	8	10	12	14	16	18
120	1, 13, 17, 21	iso	119	0	6664	0	233240	0	6285818	8000132
120	1, 7, 11, 51	iso	0	476	595	476	97342	242760	621061	12995752
120	1, 7, 13, 17	iso	0	0	952	0	28560	0	787780	1000076
120	1, 7, 11, 17	iso	0	0	0	0	1428	0	1904	79968

n		Poids					
		20	22	24	26	28	30
120	cont.	152410678	678868344	3828465501	26066879076	95327385454	618071235908
120	cont.	56207032	246474228	1749528837	8831883928	43093426089	214787987092
120	cont.	19046545	84857472	479437315	3275719188	12450412366	84684304628
120	cont.	366996	2950724	33511590	325076584	3231268047	29780853844

n	S_1	Type	Poids						
			22	24	26	28	30	32	34
138	XQ_{137}	iso	321402	2356948	21533934	490138050	6648308245	77865259105	?

n	S_1	Type	Poids	
			20	24
152	1, 3, 5, 11, 17	II	1661	315590
152	1, 3, 5, 11, 15	II	755	290675
152	XQ_{151}	II	28690	717230
152	1, 3, 7, 11, 15	II	0	253680

C.1.2 Codes ternaires

n	S_1	Poids		
		18	21	24
72	XQ_{71}	357840		
84	XQ_{83}	0	2368488	
84	$\mathcal{B}_{41}(0,1,-1)$	0	1259520	
88	$\mathcal{B}_{43}(1,1,-1)$	0	635712	
96	$\mathcal{B}_{47}(0,1,-1)$	0	0	15358848

C.2 Distributions des poids des codes XQR et QDC

Dans cette section nous listons les distributions de poids de tous les codes XQR binaires pour$74 \leq n \leq 152$, à l'exception de $n = 138$, auquel cas on n'a pas pu calculer le nombre de mots de poids34. Nous donnons aussi les distributions des meilleurs codes connus de Type III pour des longueurs$72 \leq n \leq 96$ et de quelques codes QDC.

Dans les tableaux suivants, la distribution des poids d'un code$C$ est donnée coefficient par coefficient, suivant la divisibilité du code, et du haut vers le bas. Comme dans le tableau suivant, pour les codes de Type II on indique seulement les A_i dont $i \equiv 0 \pmod 4$, commençant par $i = 0$.

$C\,[n,k,d]$		
A_0	A_{12}	...
A_4	A_{16}	...
A_8	A_{20}	...

En plus, dans le cas binaire, nous donnons seulement une moitié de la distribution car le mot$\mathbf{1}$ est dans le code, donc $A_i = A_{n-i}$ pour $n/2 < i \leq n$. Pour les codes de Type I ou isoduaux, nous n'indiquons que les A_i, avec $i \equiv 0 \pmod 2$ et dans le cas ternaire tous les A_i, $i \equiv 0 \pmod 3$.

C.2.1 Codes binaires

XQ_{73} [74, 37, 14]						
1	0	0	1093905	254815041	6824006274	24737670308
0	0	8103	8481140	961185963	13019584383	
0	0	89133	53285328	2871214319	19988042838	

XQ_{89} [90, 45, 18]					
1	0	0	249109665	136522171665	3403264592178
0	0	274120	1650476520	402352364565	4842229755990
0	0	2819520	8806726665	983530112015	5773480681050
0	0	30530115	38261417997	2001805012350	

XQ_{97} [98, 49, 16]				
1	0	931588	50926874040	17462837266490
0	0	15038492	233809599765	33526663755928
0	0	166982396	894066458642	54576674050892
0	28518	1358454930	2860927585152	75458314701252
0	80801	9150616832	7695094033816	88704970251121

$\mathcal{B}_{53}(0, 1, 0)$ [108, 54, 20]					
1	0	260442	48988609776	72678166237737	2057126174405640
0	0	5804136	296249593887	192293022909528	2564427604488075
0	0	74455407	1505946930696	436002802265628	2759782492680368
0	0	746650008	6456853758147	849260837522448	
0	0	6776021115	23472658198080	1423731100290057	

XQ_{113} [114, 57, 16]					
1	0	450870	40222705272	130792381305999	9146861126267592
0	0	2795394	282354497019	410299386601373	13906428976796370
0	0	36685789	1670741392215	1108537936510704	18376732004342040
0	25764	468994974	8378734842802	2586605780563854	21121413533212377
0	0	4741799790	35791510616904	5223754097520833	

C.2.2 Codes binaires doublement pairs

$\mathcal{B}_{59}(0, 0, 1)$ [120, 60, 20]					
1	0	38266515	18824858869630	30531593455793640	335200293307691448
0	0	6114942900	397449398883096	116023986363853260	
0	71862	475562216745	4630515150103920	25725775470657619 5	

XQ_{127} [128, 64, 20]					
1	0	12886944	18071400219840	94116098389459776	4051982879111857536
0	0	2935906752	552526728803616	549827720843995776	5193576979998072614
0	56896	320496430380	9491105395362624	1920594821540759360	

XQ_{151} [152, 76, 20]				
1	0	39390352685	542987434102295230	2949674479653927921105
0	28690	5498418925790	9222363801666354938	8446025592483033745430
0	717230	430930711970510	98458872937411257395	1584056476023984384 4500
0	164250450	19714914844819780	670740325520625252110	1952736465900603994 7448

C.2.3 Codes ternaires

XQ_{71} [72, 36, 18]				
1	0	3526708233024	33060319439473536	354030140213568
0	357840	59355639700560	44727780092682960	15690598310400
0	29772288	607803719012256	34777050459847488	230439513024
0	2688809760	3798836081510400	14918073393482560	701366400
0	125992569472	14443540225111008	3328268946790464	242112

XQ_{83} [84, 42, 21]				
1	0	1352212685085528	26733400018682175840	14629832742157920
0	2368488	20473212184431552	29675353243188471504	381318300016752
0	284708592	202449078688331520	20292322166081839440	3368236422096
0	27749368752	1310283952206924240	8274112484895023040	6324380216
0	1592706743376	5538706800932028360	1923875803617010488	168
0	57955685753952	15191881867463438592	239704188855678792	

$\mathcal{B}_{41}(0, 1, -1)$ [84, 42, 21]				
1	0	1352235251475360	26733400409832932928	14629839379331400
0	1259520	20473152007392000	29675352917229507264	381316825089312
0	307996920	202449207639130560	20292322391745737760	3368469305376
0	27516485472	1310283726543025920	8274112355944224000	6301091888
0	1594181670816	5538707126890992600	1923875863794050040	1109136
0	57949048580472	15191881476312681504	239704166289288960	

$\mathcal{B}_{43}(1, 1, -1)$ [88, 44, 21]				
1	0	1294153960207680	149357302340678914944	2082732674601397104
0	635712	25036242546864384	244362022826028476160	138180567030830016
0	120126864	321440977188929664	256787210074634276928	4155805506743616
0	13724386368	2750936673293043840	169471411253708788224	46760218984160
0	973104610368	15702733362100701840	68144098773399841872	137405652384
0	44026206285264	59598753754070017344	16023500794040249664	41268480

$\mathcal{B}_{47}(0, 1, -1)$ [96, 48, 24]				
1	0	505749122830203648	19981992658967285787648	454433488827015168
0	15358848	7072363949850708480	19059746824746748028736	7545675935390208
0	2714166528	68123018965819051008	11815486076969836320768	43161251173632
0	277096863744	452631266154565769472	463585469159763259660 8	53269682688
0	18611299710528	2071548979724323089408	111164397425203012070 4	2075712
0	828013100655744	6502086773568887367168	155474680695461541888	
0	24803529161845248	13891892673289405272576	11895672055160067200	

Bibliographie

[1] D. Augot, P. Charpin et N. Sendrier, "Studying the locator polynomial of minimum weight codewords of BCH codes," *IEEE Trans. Inform. Theory,* vol. 38, pp. 960–973, 1992.

[2] D. Augot et F. Levy-dit-Vehel, "Bounds on the Minimum Distance of the Duals of the BCH Codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 4, pp. 1257–1260, July 1996.

[3] D. Augot et N. Sendrier, "Idempotents and the BCH Bound," *IEEE Trans. Inform. Theory*, vol. 40, no. 1, pp. 204–207, Jan. 1994.

[4] C. Bachoc, "On harmonic weight enumerators of binary codes," *Designs, Codes and Cryptography,* vol. 18, pp. 11–28, 1999.

[5] Y. Berger et Y. Be'ery, "Bounds on the Trellis Size of Linear Block Codes," *IEEE Trans. Inform. Theory*, vol. 39, no. 1, pp. 203–209, Jan. 1993.

[6] Y. Berger et Y. Be'ery, "The Twisted Squaring Construction, Trellis Complexity and Generalized Weights of BCH and QR Codes," *IEEE Trans. Inform. Theory,* vol. 42, no. 6, pp. 1817–1827, Nov. 1996.

[7] G. F. M. Beenker, "A note on Extended Quadratic Residue Codes over GF(9) and Their Ternary Images", *IEEE Trans. Inform. Theory,* vol. 30, pp. 403–405, 1984.

[8] E. R. Berlekamp, *Algebraic Coding Theory*, New York, McGraw-Hill, 1968.

[9] A. Betten, H. Fripertinger, A. Kerber, A. Wassermann, et K.-H. Zimmermann, *Codierungstheorie – Konstruktion und Anwendung linearer Codes,* Springer, Heidelberg, 1998.

[10] T. Blackmore et G.H. Norton, "Matrix-Product Codes over$\mathbb{F}_q$," *AAECC*, vol. 12, pp. 477–500, 2001.

[11] A. Bonnecaze et P. Udaya, "Cyclic Self-Dual Codes over$\mathbb{F}_2 + u\mathbb{F}_2$," *IEEE Trans. Inform. Theory*, vol. 45, no. 4, pp. 1250–1255, May 1999.

[12] W. Bosma et J. Cannon, *The Magma Computational Algebra System for Algebra, Number Theory and Geometry (Version 2.9),* University of Sydney, School of Mathematics and Statistics, Computational Algebra Group, May 2002.

[13] R.C. Bose and D.K. Ray-Chaudhuri, "On a class of error correcting binary group codes", *Inform. and Control*, vol. 3, pp. 68–79, 1960.

[14] R.C. Bose and D.K. Ray-Chaudhuri, "Further results on error correcting binary group codes", *Inform. and Control*, vol. 3, pp. 279–290, 1960.

[15] A. E. Brouwer, "Bounds on the size of linear codes", *Handbook of Coding Theory*, ed. V.S. Pless and W.C. Huffman. Elsevier Science, North-Holland, pp. 295–461, 1998.

[16] A. Brouwer et T. Verhoeff, "An updated table of minimum distance bounds for binary linear codes", *IEEE Trans. Inform. Theory*, vol. 39, no. 2, pp. 662–677, March 1993.

[17] A. Canteaut, "Attaques de cyptosystèmes à mots de poids faible et construction de fonctions t-résilientes," Thèse de doctorat, Univ. Paris VI, 1996.

[18] A. Canteaut et F. Chabaud, "A New Algorithm for Finding Minimum-Weight Words in a Linear Code : Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511," *IEEE Trans. Inform. Theory*, vol. 44, no. 1, pp. 367–378, Jan. 1998.

[19] A. Canteaut, "Programmation en langage C," Cours de DEA de Limoges, 2000.

[20] L. Carlitz et S. Uchiyama, "Bounds for exponential sums", *Duke Math. J.*, vol. 24, pp. 37–41, Dec. 1957.

[21] P. Charpin, "Open problems on cyclic codes," *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds., Elsevier Science, North-Holland, 1998.

[22] P. Charpin, "Codes cycliques étendus invariants sous le groupe affine," Thèse de doctorat d'État, Univ. Paris VII, 1987.

[23] P. Charpin et F. Levy-dit-Vehel, "On Self-Dual Affine-Invariant Codes," *Journal of Comb. Theory A.*, vol. 67, no. 2, pp. 223–244, Aug. 1994.

[24] G. Castagnoli, J. L. Massey, Ph. A. Shoeller et N. von Seemann, "On Repeated-Root Cyclic Codes," *IEEE Trans. Inform. Theory*, vol. 37, no. 2, pp. 337–342, Mar. 1991.

[25] J. H. Conway et N. J. A. Sloane, "A New Upper Bound on the Minimal Distance of Self-Dual Codes," *IEEE Trans. Inform. Theory*, vol. 36, no. 6, pp. 1319–1333, Nov. 1990.

[26] Y. Desaki, T. Fujiwara et T. Kasami, "The Weight Distributions of Extended Binary Primitive BCH Codes of Length 128," *IEEE Trans. Inform. Theory*, vol. 43, no. 4, pp. 1364–1371, July 1997.

[27] S. T. Dougherty et M. Harada, "Shadow Optimal Self-Dual Codes," *Kyushu J. Math.*, vol. 53, pp. 223–227, 1999.

[28] I. Dumer, D. Micciancio et M. Sudan, "Hardness of Approximating the Minimum Distance of a Linear Code," *IEEE Trans. Inform. Theory*, vol. 49, no. 1, pp. 22–37, Jan. 2003.

[29] S. Fedorenko et E. Krouk, "About Block Circulant Representation of Linear Codes," *Proccedings of Sixth International Workshop on Algebraic and Combinatorial Coding Theory*, Pskov, Russia, 1998, pp. 116–118.

[30] S. Fedorenko, "On the Structure of Linear Block Codes Given the Group of Symmetry," *Proc. IEEE Inter. Workshop on Concatenated Codes*, Ulm, Germany, 1999.

[31] S. Fedorenko, "The table decoders of quadratic-residue codes," *ACCT'2000*, pp. 137–140.

[32] G.D. Forney, Jr., "Coset Codes-Part 2 : Binary Lattices and Related Codes", *IEEE Trans. Inform. Theory*, vol. 34, no. 5, pp. 1152–1187, Sept. 1988.

[33] G.D. Forney, Jr., "Dimension/Length Profiles and Trellis Complexity of Linear Block Codes", *IEEE Trans. Inform. Theory*, vol. 40, no. 6, pp. 1741–1752, Nov. 1994.

[34] T. Fujiwara et T. Kasami, "The Weight Distribution of(256, k) Extended Binary Primitive BCH Code with $k \leq 63$, $k \geq 207$," *Technical Report of IEICE, IT97-46*, pp. 29–33, Sept. 1997.

[35] Ph. Gaborit, "Quadratic Double Circulant Codes over Fields," *J. Combin. Theory Ser. A,* vol. 97, pp. 85–107, 2002.

[36] Ph. Gaborit, C.-S. Nedeloaia et A. Wassermann, "Weight Enumerators of Duadic and Quadratic Residue Codes", ISIT'04, Chicago, USA, 27 juin - 2 juillet 2004, p. 485.

[37] Ph. Gaborit, C.-S. Nedeloaia et A. Wassermann, "On the Weight Enumerators of Duadic and Quadratic Residue Codes", *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 402–407, Jan. 2005.

[38] A. M. Gleason, "Weight polynomials of self-dual codes and the MacWilliams identities," *Actes Congres Intern. de Mathématique,* 3, 1970, Gauthier-Villars, pp. 211–215, Paris, 1971.

[39] M. Goossens, F. Mittelbach, A. Samarin, *The LaTeX Companion. Reading*, Addison-Wesley, 1994.

[40] M. Goossens, F. Mittelbach, S. Rahtz, *The LaTeX Companion. Graphics*, Addison-Wesley, 1997.

[41] *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman Editors, Elsevier Science, North-Holland, 1998.

[42] H. Helgert et R. Stinaff, "Shortened BCH codes," *IEEE Trans. Inform. Theory,* vol. IT-19, pp. 818–820, 1973.

[43] A. Hocquenghem, *Codes correcteurs d'erreurs*, Chiffres, vol. 2, pp. 147–156, 1959, Paris.

[44] X. Hou, "GL$(m, 2)$ Acting on $R(r, m)/R(r-1, m)$," *Discrete Mathematics,* vol. 49, pp. 99–122, 1996.

[45] X. Hou, "Cubic bent functions," *Discrete Mathematics*, vol. 189, pp. 149–161, 1998

[46] W.C. Huffman, "Decomposing and Shortening Codes using Automorphisms," *IEEE Trans. Inform. Theory,* vol. IT-32, no. 6, pp. 833–836, Nov. 1986.

[47] W.C. Huffman, V. Job et V. Pless, "Multipliers and Generalized Multipliers of Cyclic Objects and Cyclic Codes," *J.Comb. Theory A,* vol. 62, 1993.

[48] R.A. Jenson, "A Double Circulant Presentation for Quadratic Residue Codes," *IEEE Trans. Inform. Theory,* vol. 26, no. 2, pp. 223–227, 1980.

[49] M. Karlin, "New binary coding results by circulant," *IEEE Trans. Inform. Theory,* vol. 15, pp. 81–92, 1969.

[50] T. Kasami, S. Lin et W. Peterson, "Some results on cyclic codes which are invariant under the affine group," *Tech. Rep. AFCRL-66-622,* Air Force Cambridge Res. Labs., Bedford, MA, 1966.

[51] T. Kasami, S. Lin et W. Peterson, "New Generalizations of the Reed-Muller Codes. Part I : Primitive Codes," *IEEE Trans. Inform. Theory,* vol. IT-14, no. 2, pp 189–199, Mar. 1968.

[52] T. Kasami, S. Lin et W. Peterson, "Polynomial Codes," *IEEE Trans. Inform. Theory,* vol. 14, no. 6, pp. 807–814, Nov. 1968.

[53] T. Kasami et S. Lin, "Some results on the minimum weight of primitive BCH codes," *IEEE Trans. Inform. Theory,* vol. IT-18, pp. 824–825, 1972.

[54] T. Kasami et N. Tokura, "Some remarks on BCH bounds and minimum weights of primitive binary BCH codes," *IEEE Trans. Inform. Theory,* vol. IT-15, pp. 408–413, 1969.

[55] T. Kasami, N. Tokura et S. Azumi, "On the Weight Enumeration of Weight Less than$2.5d$ of Reed-Muller Codes," *Information and Control,* vol. 30, no. 4, pp. 380–395, Apr. 1976.

[56] T. Kasami, T. Takata, T. Fujiwara, S. Lin, "On the Optimum Bit Orders with Respect to the State Complexity of Trellis Diagrams for Binary Linear Codes", *IEEE Trans. Inform. Theory*, vol. 39, no. 1, pp. 242–245 , Jan. 1993.

[57] T. Kasami, T. Takata, T. Fujiwara, S. Lin, "On Complexity of Trellis Structure of Linear Block Codes", *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 1057–1064, May 1993.

[58] F. R. Kschischang et V. Sorokine, "On the Trellis Structure of Block Codes," *IEEE Trans. Inform. Theory*, vol. 41, no. 6, pp. 1924–1937, Nov. 1995.

[59] A. Lafourcade, A. Vardy, "Optimal Sectionalization of a Treillis", *IEEE Trans. Inform. Theory*, vol. 42, no. 3, pp. 689–703, May 1996.

[60] P. Lee et E. Brickell, "An observation on the security of McEliece's public-key cyptosystem," *Advances in Cryptology-EUROCRYPT'88,* C. Günter, Ed. New York : Springer-Verlag, pp. 275-280, 1988.

[61] J. Leon, "A probabilistic algorithm for computing minimum weights of large error-correcting codes," *IEEE Trans. Inform. Theory,* vol. 34, pp. 1354–1359, 1988.

[62] J. Leon, J. Masley et V. Pless, "Duadic Codes", *IEEE Trans. Inform. Theory,* vol. IT-30, pp. 709–714, 1984.

[63] F. Levy-dit-Vehel, "Bounds on the Minimum Distance of the Duals of Extended BCH codes over $\mathbb{F}_p$," *Applicable Algebra in Engineering Communication and Computing (AAECC),* New York : Springer-Verlag, vol. 6, no. 3, pp. 175–190, 1995.

[64] S. Lin, T. Kasami, T. Fujiwara et M. Fossorier, *Trellis and Trellis-Based Decoding Algorithms for Linear Block Codes*, Kluwer Academic Publishers, 1998.

[65] J. H. van Lint, "Repeated-Root Cyclic Codes," *IEEE Trans. Inform. Theory*, vol. 37, no. 2, pp. 343–345, Mar. 1991.

[66] J. H. van Lint et R. M. Wilson, "On the Minimum Distance of Cyclic Codes," *IEEE Trans. Inform. Theory*, vol. 32, no. 1, pp. 23–40, Jan. 1986.

[67] H. Lüneburg, "Gray-Codes", *Abh. Math. Sem. Hamburg,* vol. 52, pp. 208–227, 1982.

[68] R. McEliece, "On the BCJR Trellis for Linear Block Codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 4, pp. 1072–1092, July 1996.

[69] B. D. McKay, "*autoson*—a distributed batch system for UNIX workstation networks (version 1.3)", Tech. Rep. TR-CS-96-03, Computer Science Department, Australian National University, 1996.

[70] F. J. MacWilliams et N. J. A. Sloane, *The Theory of Error-Correcting Codes,* Amsterdam, North-Holland, 1977.

[71] MAGMA page web : http ://www.maths.usyd.edu.au :8000/u/magma/

[72] MEDICIS page web : http ://www.medicis.polytechnique.fr

[73] O. Moreno et C.J. Moreno, "The MacWilliams-Sloane Conjecture on the Tightness of the Carlitz-Uchiyama Bound and the Weights of Duals of BCH Codes," *IEEE Trans. Inform. Theory*, vol. 40, no. 6, pp. 1894–1907, Nov. 1994.

[74] D.J. Muder, "Minimal Trellis for Block Codes", *IEEE Trans. Inform. Theory*, vol. 34, no. 5, pp. 1049–1053, Sept. 1988.

[75] C.-S. Nedeloaia, "Sur un algorithme de calcul des polynômes énumérateurs des poids," Mémoire de stage de DEA, Univ de Limoges, 2001.

[76] C.-S. Nedeloaia, "On Weight Distributions of Cyclic Self-Dual Codes," ISIT'02, Lausanne, Suisse, juin-juillet 2002, p. 232.

[77] C.-S. Nedeloaia, "Weight Distributions of Cyclic Self-Dual Codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 6, pp. 1582–1591, June 2003.

[78] C.-S. Nedeloaia, "Upper Bounds on the Dual Distances of EBCH Codes," Rapport de recherche no. 5477, INRIA Rocquencourt, 14 pages, Jan. 2005.

[79] A. Nijenhuis et H. S. Wilf, *Combinatorial Algorithms*, 2nd ed., Academic Press, New York, 1978.

[80] W.W. Peterson, E.J. Weldon, Jr., *Error-Correcting Codes*, MIT Press, 1972.

[81] V. Pless, "Symmetry Codes over GF(3) and new five designs", *J. Combin. Theory Ser. A,* vol. 12, pp. 119–142, 1972.

[82] V. Pless, "Q-Codes", *J. Combin. Theory Ser. A,* vol. 43, pp. 258–276, 1986.

[83] V. Pless, "Duadic Codes and Generalizations," *Proceedings of EUROCODE '92*, Springer-Verlag, pp. 3–15.

[84] V. Pless, *Introduction to the Theory of Error Correcting Codes,* 3rd ed., Wiley, New York, 1998.

[85] V. Pless, W. C. Huffman et R. A. Brualdi, "An Introduction to Algebraic Codes",*Handbook of Coding Theory*, ed. V. S. Pless and W. C. Huffman, Elsevier-Science, North-Holland, pp. 3–139, 1998.

[86] V. Pless, J. Masley et J. Leon, "On Weights in Duadic Codes", *J. Combin. Theory Ser. A,* vol. 44, pp. 6–21, 1987.

[87] E. M. Rains et N. J. A. Sloane, "Self-Dual Codes", *Handbook of Coding Theory,* ed. V. S. Pless and W. C. Huffman. Elsevier Science, North-Holland, pp. 177–294, 1998.

[88] F. Rodier, "On the Spectra of the Duals of Binary BCH Codes of Designed Distance$\delta = 9$," *IEEE Trans. Inform. Theory*, vol. 38, no. 2, pp. 478–479, March 1992.

[89] F. Rodier, "On the Minimum Distance of the Duals of 4-Error Correcting Extended BCH Codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1677–1678, July 1999.

[90] M. Sala, "Upper Bounds on the Dual Distance of BCH$(255, k)$," *Designs, Codes and Cryptography,* vol. 30, pp. 159–168, 2003.

[91] M. Sala et A. Tamponi, "A Linear Programming Estimate of the Weight Distribution of BCH$(255, k)$," *IEEE Trans. Inform. Theory*, vol. 46, no. 6, pp. 2235–2237, Sept. 2000.

[92] T. Schaub, "A linear complexity approach to cyclic codes," dissertation, Swiss Federal Institute of Technology, Zurich, 1988.

[93] N. J. A. Sloane et J. G. Thompson, "Cyclic Self-Dual Codes", *IEEE Trans. Inform. Theory*, vol. IT-29, no. 3, pp. 364–366, May 1983.

[94] M. H. M. Smid, "Duadic Codes", *IEEE Trans. Inform. Theory,* vol. 33, pp. 432–433, 1987.

[95] J. Stern, "A Method for finding codewords of small weight," *Coding Theory and Applications,* G. Cohen and J. Wolfmann, Eds. New York : Springer-Verlag, pp. 106–113, 1989.

[96] M. Sugino, Y. Ienaga, N. Tokura et T. Kasami, "Weight Distribution of $(128, 64)$ Reed-Muller Code," *IEEE Trans. Inform. Theory,* vol. IT-17, no. 5, pp. 627–628, Sep. 1971. sw

[97] T. Sugita, T. Kasami et T. Fujiwara, "The Weight Distribution of the Third-Order Reed-Muller Code of Length 512", *IEEE Trans. Inform. Theory,* vol. 42, no. 5, pp. 1622–1625, Sept. 1996.

[98] H.C.A. van Tilborg, "On Weights in Codes," *TH-Report 71-WSK-03,* Tech. Univ. Eindhoven, Netherlands, Dec. 1971.

[99] A. Vardy, "The intractibility of computing the minimum distance of a code", *IEEE Trans. Inform. Theory*, vol. 43, pp. 1757-1766, 1997.

[100] A. Vardy, "Trellis Structure of Codes", *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds., Chapitre 24, pp. 1989–2117, Elsevier Science, North-Holland, 1998.

[101] A. Vardy et Y. Be'ery, "Maximum-Likelihood Soft Decision Decoding of BCH codes," *IEEE Trans. Inform. Theory*, vol. 40, no. 2, pp. 546–554, March 1994.

[102] M. van der Vlugt, "Non-BCH Triple-Error-Correcting Codes", *IEEE Trans. Inform. Theory*, vol. 42, no. 5, pp. 1612–1614, Sept. 1996.

[103] Weight Distribution tables : http ://www.logic.cs.hiroshima-cu.ac.jp/wd

[104] J. K. Wolf, "Efficient maximum-likelihood decoding of linear block codes using a trellis", *IEEE Trans. Inform. Theory*, vol. 24, pp. 76–84, 1978.

Résumé

Cette thèse est dédiée à l'étude des codes linéaires binaires cycliques ou auto-duaux, en utilisant la décomposition de leurs matrices génératrices. Cela nous amène à l'obtention des mots de petit poids voire au calcul exact des polynômes énumérateurs. Afin de simplifier la matrice génératrice, nous passons de l'ordre cyclique à l'ordre standard des bits. Cela nous permet de mettre en forme carrée modifiée récursive tous les codes affines-invariants et de donner de nouvelles bornes supérieures pour les distances duales des codes BCH binaires de longueur 512.

Par la suite, nous montrons que tout code cyclique binaire à racines multiples est équivalent à une construction carrée. En particulier tous les codes cycliques auto-duaux (*c.a.d.*) dont toutes les racines ont une multiplicité paire sont des sommes directes. Notre étude aboutit à l'énumération et au calcul des polynômes énumérateurs de tous les codes *c.a.d.* de longueur $n \leq 120$. Enfin, nous étudions certains codes auto-duaux, dont les plus connus sont les codes résidus quadratiques. En utilisant un algorithme de calcul du nombre des mots de petit poids et la théorie des invariants, nous avons obtenu les énumérateurs des poids de tous les codes duadiques de longueur $n \leq 152$, à une exception près.

Mots-clés: code linéaire, polynôme énumérateur des poids, code cyclique, code auto-dual, constructions carré et carrée modifiée, théorie des invariants, mot de petit poids, ordre standard.

Abstract

This thesis is dedicated to the study of binary linear codes, cyclic or self-dual, by using the decomposition of their generator matrices. This enables us to obtain minimum-weight words and to calculate weight enumerators. In order to simplify the generator matrix, we pass from the cyclic order to the standard bit order. We deduce a recursive twisted squaring construction of all affine-invariant codes and we give new upper bounds on the dual distances of binary extended BCH codes of length 512.

On the other hand, we prove that any binary repeated-root cyclic code is equivalent to a squaring construction. In particular, any cyclic self-dual code (*c.a.d.*) with all roots even-repeated is a direct sum. Our study succeeds by enumerating all *c.a.d.* codes of length $n \leq 120$ and by giving their weight enumerators. Finally, we study some classes of self-dual codes, the most famous being the quadratic-residue codes. By using an algorithm for finding the number of low-weight words and the invariant theory, we obtain the weight enumerators of all duadic codes of length $n \leq 152$, up to one code.

Keywords: linear code, weight enumerator, cyclic code, self-dual code, (twisted) squaring construction, invariant theory, low-weight word, standard bit order.