THÈSE de DOCTORAT de l'UNIVERSITÉ PARIS 6

Spécialité : INFORMATIQUE

Présentée par

Nicolas SENDRIER

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ PARIS 6

Titre de la thèse :

Codes Correcteurs d'Erreurs à Haut Pouvoir de Correction

Soutenue le 20 décembre 1991, devant le jury composé de :

Daniel LAZARD	*Président*
Bernard COURTEAU	*Rapporteur*
Michel MINOUX	*Rapporteur*
Gérard COHEN	*Rapporteur*
Paul CAMION	*Directeur*
Pascale CHARPIN	
Thomas ERICSON	
Harold MATTSON	

Remerciements

Je tiens à remercier d'abord Bernard COURTEAU qui a largement, et heureusement, débordé de son rôle de rapporteur en me suggérant de nombreuses améliorations tant sur le fond que sur la forme de ce document.

Je l'associe également aux remerciements que j'adresse aux autres rapporteurs Michel MINOUX et Gérard COHEN, qui ont accepté la charge de ce rapport malgré les délais très courts qui leur étaient imposés.

Merci à Daniel LAZARD pour la gentillesse qu'il a toujours témoignée à mon égard et pour avoir accepté de présider le jury de cette thèse.

Les trois mois que j'ai passés en Suède à l'université de Linköping au début de l'année 1990, ont été pour moi une occasion unique de prendre du recul sur mes travaux et en même temps d'en élargir la portée. Merci donc à Thomas ERICSON et au groupe "Data Transmission" pour leur compétence et la sympathie avec laquelle ils m'ont acceuilli.

Je remercie Paul CAMION et Pascale CHARPIN d'avoir suscité chez moi un grand intérêt pour la recherche. Merci également pour leurs nombreux et précieux conseils et pour la sérénité qu'ils apportent tous les jours au projet CODES.

Merci enfin à Harold MATTSON dont la présence dans le jury m'honore plus que je ne saurais le dire.

Outre aux artisans et aux juges de cette thèse, je tiens à exprimer ma gratitude à de nombreuses personnes, tout d'abord à Hervé CHABANNE qui a été mon premier, et non le moindre, relecteur.

Merci également à Daniel AUGOT qui, outre notre fructueuse collaboration avec Pascale CHARPIN, a su faire preuve en toutes circonstances de sa désormais légendaire jovialité.

L'informatique ne me serait sans doute pas apparue si intéressante sans Jean-Marc STEYAERT, je l'en remercie ainsi que Philipe FLAJOLET pour sa disponibilité et pour avoir guidé mes premiers pas à l'INRIA.

Merci pour leurs qualités humaines et leur aide constante à François MORAIN, Bruno SALVY et Paul ZIMMERMANN, "Wizards" aux compétences multiples dont j'ai toujours su user et abuser.

Pour leur amitié et leur assistance lors de mes passages au LITP, je remercie Annick VALIBOUZE et Renaud RIOBOO.

Je remercie Christian DESARMÉNIEN pour sa gentillesse et sa patience envers un ingénieur système débutant, merci également à Christine DUBOIS et Christelle GUIZIOU pour leur présence discrète mais efficace au secrétariat.

Ma gratidude va aussi à André MONTPETIT, dont le passage au projet CODES fut trop court mais restera dans nos mémoires.

Mes excuses enfin à tous ceux que je n'aurais pas dû oublier.

Table des matières

Notations

- $A[X]$ l'anneau des polynômes à une indéterminée de l'anneau A.
- $\mathbb{F}_q$ le corps fini à q éléments.
- $\lfloor r \rfloor$, la partie entière par défaut d'un entier r.
- Pour $p(X) \in A[X]$, on note $[X^i]p(X)$ le coefficient de X^i dans $p(X)$.
- Pour un ensemble E, on notera $|E|$ son cardinal.
- $\frac{1}{2}\mathbb{Z}$ dénote l'ensemble des demi-entiers.
- Pour toutes matrices M et M' ayant le même nombre de ligne $(M \mid M')$ représentera la matrice dont la i-ième ligne est constituée de la concaténation de la i-ième ligne de M et de la i-ième ligne de M'.
- Soit E un ensemble, on note $\mathcal{P}(E)$ l'ensemble des parties de E.
- On notera $A + B$ la somme de deux espaces vectoriels A et B, i.e. l'espace vectoriel engendré par $A \cup B$. On dira que la somme est directe si $A \cap B = \{0\}$ et on notera $A \oplus B$.

Introduction générale

1 Historique

Ce travail fait suite à une étude menée à bien par le projet CODES de l'INRIA pour la société Alcatel Thomson Faisceaux Herziens (cf. [21]). Il s'agissait de concevoir une configuration de codage pour combattre les effets d'un brouilleur impulsionnel et d'en évaluer les performances. La configuration proposée reposait sur un schéma de concaténation classique de Forney [36].

Les résultats des ces travaux ont mis en évidence la nécessité de procédés de construction de codes de grande capacité de correction à partir de codes de petite longueur, ceux-ci se prétant mieux aux algorithmes de décodage rapides, ainsi que l'intérêt de développer des outils combinatoires d'analyse de performance de tels codes. Le sujet de cette thèse s'est donc imposé naturellement.

Une grande partie de cette thèse, à savoir les chapitres III et IV, a été effectuée dans le cadre d'une étude pour la DRET, en cours, qui traite des codes permettant de lutter contre de très hauts niveaux de bruit, c'est-à-dire de 10 à 30% de taux d'altération (effacements ou erreurs).

2 Hypothèses

Nous supposerons en général pour l'évaluation des configurations de codage que le canal de transmission est symétrique et sans mémoire. C'est-à-dire que les erreurs seront indépendantes, et chaque symbole transmis aura la même probabilité d'être transformé en chacun des autres.

Cette hypothèse est forte, et est inadaptée au cas du brouilleur impulsionnel par exemple. Nous supposerons donc que, si besoin est, les mots de codes ont été entrelacés dans une table suffisamment grande, de façon à assurer la validité du modèle choisi de canal.

3 Nécessité de codes de grande longueur

La conception de codes à haut pouvoir de correction implique des conditions nécessaires sur les paramètres de ces codes. Tout d'abord, il est clair que le taux de transmission devra être faible, le théorème de Shannon ([67, p.22]) nous donne une borne supérieure de ce taux pour un canal donné.

Une seconde condition nécessaire porte sur la longueur du code, celle-ci doit être grande. La justification que donne Blahut de ce fait dans l'introduction de [11] est la suivante :

> "Whenever a message consist of a large number of bits, it is better in principle to use a single block code of large blocklength than to use a succession of codewords from a shorter block code. The nature of statistical fluctuations is such that a random pattern of errors usually exhibits some clustering of errors. Some segments of the random pattern contain more than the average number of errors, and some segments contain less. Hence, long codewords are considerably less sensitive to random errors than are shorter codewords of the same rate; but, of course, the encoder and decoder may be more complex. As an example, suppose that 1000 information bits are transmitted with a (fictitious) 2000-bit binary codeword that can correct 100 bit errors. Compare this with a scheme for transmitting 100 bits at a time with a 200-bit binary codeword that can correct 10 bit errors per block. Ten such blocks are needed to transmit 1000 bits. This latter scheme can also correct a total of 100 errors, but only if they are properly distributed – 10 errors to a 200-bit block. The first scheme can correct 100 errors no matter how they are distributed within the 2000-bit codeword. It is far more powerful.
>
> This heuristic argument can be given a sound theoretical footing, but that is not our purpose here. We only wish to make plausible the fact that good codes are of long blocklength, and that very good codes are of very long blocklength. Such codes can be very hard to find and when found may require complex devices to implement the encoding and decoding operations."

4 Distance minimale des codes BCH binaires

Nous nous sommes donc intéréssé aux codes de grande longueur. Ceux-ci posent comme le suggère Blahut le problème de leur complexité de décodage. En particulier les codes BCH que nous étudions dans le chapitre II ont une complexité de décodage relativement importante, en $O(n^2)$ opérations élémentaires pour un code de longueur n, ces opérations s'effectuant dans un corps fini de taille $O(n)$ ($n+1$ pour un code BCH primitif). La distance minimale des codes BCH binaire en grande longueur - 255 et au delà - n'est pas connue dans tous les cas. Nous nous sommes intéréssé au calcul de cette distance minimale pour les codes BCH binaires primitifs au sens strict de longueur 255 et 511.

Nous développons une méthode de recherche de la distance minimale des codes cycliques à l'aide des identités de Newton. Ces identités nous procurent un système d'équations qui doit être vérifié par les fonctions symétriques élémentaires et les fonctions puissances symétriques des localisateurs d'un mot dont on peut fixer le poids. Les codes cycliques pouvant être définis par des conditions portant sur les fonctions puissances élémentaires de ses mots, ces conditions sont facile à intégrer aux équations de Newton.

Cela nous permet de donner une condition nécessaire et suffisante pour l'existence d'un mot de poids donné dans un code. Cette condition est l'existence de solution à un système d'équation dont on peut donner l'expression pour un code et un poids donné.

Dans le cas des codes BCH binaires primitifs au sens strict, ce système prend une forme suffisemment simple pour autoriser une résolution à l'aide du logiciel de calcul formel MAPLE. Nous avons pu prouver l'absence de solutions de poids égal à la distance construite pour certains codes ; il s'agit, en longueur 255, des codes $B(255, 59)$ et $B(255, 61)$ dont les distances minimales valent respectivement 61 et 63, et, en longueur 511, du code $B(511, 123)$ dont la distance minimale vaut 127. Nous avons pu également montrer pour certains codes de longueur 511 l'existence de solutions de poids égal à la distance construite.

5 Outils d'analyse de performance

Outre les codes BCH binaires, nous nous sommes intéressé à deux types de construction permettant d'obtenir des codes de grande longueur. Ces deux méthodes, codes concaténés et codes produit, ont l'avantage de fournir des codes longs dont la complexité de décodage est relativement faible.

Bien que cela ne soit pas parfaitement exact, on peut estimer que ces codes permettent de multiplier les paramètres de leur codes composants, en ajoutant seulement leur complexités de décodage. Le principal problème qui se pose alors est d'arriver à évaluer le plus précisement possible les performances de ces codes, afin de savoir s'il peuvent répondre au problème que nous nous posons, à savoir corriger des taux d'erreur élevés.

Pour évaluer les performances de ces codes, il nous a fallu d'abord définir avec précision ce qu'était un algorithme de décodage ; une application de l'espace ambiant dans le code et qui conserve le code. La propriété clé d'une telle application est sa borne, on dira qu'un algorithme est borné par un entier e, s'il corrige toute erreur de poids strictement inférieur à $\frac{e}{2}$, et on dira que cette borne est stricte si l'algorithme ne corrige pas d'erreur de poids supérieur.

En utilisant la combinatoire énumérative, nous pouvons décrire complètement un algorithme de décodage $\gamma : \mathbb{F}_q^n \to C \cup \{\infty\}$ d'un code C de longueur n sur un alphabet $\mathbb{F}_q$ à l'aide de trois polynômes

- le polynôme des motifs corrigibles

$$P_0(x) = \sum_{m \in \gamma^{-1}(0)} x^{w_H(m)},$$

- le polynôme des motifs non corrigibles

$$P_1(x) = \sum_{m \in \gamma^{-1}(\infty)} x^{w_H(m)},$$

- le polynôme des motifs erronés

$$P_2(x) = \sum_{m \in \gamma^{-1}(C \setminus \{0\})} x^{w_H(m)}.$$

Ces polynômes étant liés par la relation

$$P_0(x) + P_1(x) + P_2(x) = (1 + (q-1)x)^n,$$

puisqu'ils énumèrent tous les mots de l'espace.

Pour la correction d'erreur et d'effacement simultanement, nous introduisons la métrique de Hamming généralisée Δ_H ; le symbole ∞ représente un effacement, et on attribue à ces effacement le poids $\frac{1}{2}$, c'est-à-dire en longueur 1, on a $\Delta_H(\infty, x) = \frac{1}{2}$ pour $x \neq \infty$. De cette manière nous obtenons une distance et en outre nous traduisons l'idée intuitive qu'un effacement "coûte" 2 fois moins cher à corriger qu'une erreur.

De la même manière que précédemment, cela nous permet de définir les polynômes énumérateurs des motifs d'erreur et d'effacement, mais avec des polynômes de $\mathbb{Z}[x^{\frac{1}{2}}]$ car la distance de Hamming généralisée est à valeur dans les demi-entiers.

Enfin nous définissons la capacité de correction d'un code pour un algorithme de décodage donné. Cette grandeur a la même dimension qu'une distance minimale, et sert à mesurer les performances d'un algorithme corrigeant au delà de la distance minimale.

La capacité de correction d'un algorithme est le plus grand entier d^* tel que les performances d'un code hypothétique de distance minimale d^* possédant un algorithme de décodage borné strictement par d^* soient inférieures à celle du code mesuré. Bien entendu une telle définition nécessite que le canal de transmission, c'est-à-dire pour nous la probabilité d'erreur par symbole, soit donnée.

Dans la suite cette capacité de correction pourra être mesurée soit à l'aide d'outils théoriques, dans le cas des codes concaténés, soit à l'aide de simulations, pour les codes produit.

6 Construction et décodage des codes concaténés

L'idée des codes concaténés date de 1966 avec l'ouvrage de Forney [36]. Il s'agit de remplacer les symboles d'un code "externe" par des mots d'un code "interne", le cardinal de l'ensemble des symboles du codes externe devra donc être égal au cardinal du code interne.

Le décodage "naïf" d'un code concaténé consiste à décoder successivement le code interne pour obtenir des symboles du code externe, provoquant d'éventuelles erreurs, puis le code externe. Il existe un algorithme plus performant, dit de Block-Zyablov, basé sur les idées du décodage souple de Forney ; il s'agit de décoder plusieurs fois l'ensemble des mots du code interne en dégradant progressivement les performances du code interne. Ceci a pour effet d'augmenter le nombre d'echecs du décodage interne (qui correspondent à des effacements pour le code externe), et de diminuer le nombre de décodages internes erronés (qui correspondent à des erreurs pour le code externe). Le résultat important, qui garantit le fonctionnement de cet algorithme, est que pour tout motif d'erreur de poids inférieur à la moitié de la distance minimale du code concaténé, il existe un compromis pour le décodage interne qui fourni un motif d'erreur et d'effacement corrigible par le code externe.

Nous avons pu en utilisant les outils d'evaluation définis dans le chapitre I obtenir une évaluation des performances de ces deux algorithmes lorsque les algorithmes de décodage des codes composants sont bornés.

Enfin nous définissons un troisième algorithme de décodage des codes concaténés d'ordre 1, le décodage partiel. Ce décodage sera utile pour définir celui des codes concaténés généralisés d'abord, puis celui des codes produit. On peut résumer son fonctionnement de la façon suivante ; nous disposons pour décoder le code concaténé $\mathcal{C}$ sur $\mathbb{F}_q$ de $\mathcal{E}$ et $\mathcal{B}'$ d'un algorithme de décodage du code externe $\mathcal{E}$, mais pour le code interne nous ne possédons d'algorithme que pour pour un surcode $\mathcal{B}$ de $\mathcal{B}'$. Un algorithme de décodage peut néanmions être défini en utilisant une projection $\pi : \mathcal{B} \to \mathcal{B}'$. Le décodage partiel Ψ aura, en outre, la propriété fondamentale suivante :

$$\forall \mathbf{y} \in \mathbb{F}_q^{n_b n_e}, \ \forall \mathbf{v} \in \mathcal{V}^{n_e}, \quad \Psi(\mathbf{y}+\mathbf{v}) = \Psi(\mathbf{y}),$$

où n_b est la longueur de $\mathcal{B}$ et de $\mathcal{B}'$, n_e est la longueur de $\mathcal{E}$, et $\mathcal{V}$ est un supplémentaire de $\mathcal{B}'$ dans $\mathcal{B}$ égal au noyau de π.

La concaténation d'ordre supérieur est une idée des russes Block, Zyablov et Zinoviev, et consiste à utiliser plusieurs codes externes et plusieurs code internes. Nous donnons dans ce travail une définition des codes concaténés généralisés, dans le cas linéaire, comme une somme directe de codes concaténés d'ordre 1.

Cette définition nous permet de définir le décodage de ces codes de façon récursive ; décoder un code concaténé d'ordre m est équivalent à décoder partiellement un code concaténé d'ordre 1, puis à decoder un code concaténé d'ordre $m-1$.

7 Les codes produit

Les codes produit ont été introduits par Elias en 1954 sous le nom de codes itérés. Ils sont définis à l'aide de deux codes composants sur le même alphabet. Le code produit de C_1 et C_2, noté $C_1 \otimes C_2$, est l'ensemble des matrices dont chaque ligne est un élément de C_1, et chaque colonne est un élément de C_2. Contrairement aux codes concaténés les deux codes codes composants ont un rôle symétrique, cette propriété sera utilisée pour le décodage.

Le premier algorithme de décodage que nous présentons, et que nous appelons algorithme "élémentaire" consiste à décoder dans l'ordre chaque ligne puis chaque colonne jusqu'à obtenir un mot de code. Cet algorithme à été simulé et s'est avéré très efficace pour des produits de codes de Reed-Solomon sur $\mathbb{F}_{16}$.

Le second algorithme que nous présentons est l'algorithme de Reddy-Robinson. Pour introduire cet algorithme, nous montrons tout d'abord qu'un code produit peut s'exprimer comme somme directe de codes concaténés d'ordre 1. Cette somme n'a pas les mêmes propriétés que celle définissant un code concaténés généralisé, mais permet de décrire le décodage d'un code produit comme le décodage en parallèle de codes concaténés d'ordre 1 par un algorithme de décodage partiel.

L'algorithme de Reddy-Robinson a pour principal défaut de ne pas tenir compte de la symétrie du code produit, cela nous a amené après étude des résultats des simulations à concevoir un algorithme plus performant défini comme une version itérative de l'algorithme de Reddy-Robinson.

Nous avons pu obtenir par des simulations des résultats intéressants sur les performances des codes produit. Les algorithmes décrits plus haut permettent en effet de corriger des motifs d'erreur très au delà de la distance minimale du code. Par exemple nous avons pu corriger pour le code $RS(15,7,9) \otimes RS(15,7,9)$ plus de 99% des motifs d'erreur de poids 85, alors que la distance minimale de ce code est de 81, et donc n'autorise le décodage d'erreur avec certitude que jusqu'au poids d'erreur 40. Pour ce même code nous avons obtenu une capacité de correction d'erreur de 179 pour un taux d'erreur résiduel de 10^{-5} (ce qui correspond à un taux d'erreur de 29% pour des symboles de $\mathbb{F}_{16}$) alors que la borne de Singleton valable pour tout code en bloc donne, pour les même paramètres, une distance minimale inférieure ou égale 177.

Cela signifie que pour obtenir un taux d'erreur résiduel de 10^{-5}, on ne peut pas trouver de code en bloc de même longueur, de même dimension et ayant un algorithme de décodage borné strictement par sa distance minimale, meilleur que le code $RS(15,7,9) \otimes RS(15,7,9)$.

Enfin nous concluons l'exposé sur les codes produit en montrant que leur complexité de décodage reste très raisonnable. En effet pour une même quantité d'information transmise, la complexité de décodage est sensiblement la même que celle des codes composants.

Chapitre I

Quelques notions de théorie des codes

La section 1 de ce chapitre rappelle les principales notions sur les corps finis nécessaires à la définition des codes cycliques dans la section 2.

La section 3 traite du problème du décodage. On donnera en particulier une définition d'un algorithme de décodage comme une application à valeur dans la réunion du code et du symbole ∞ (qui représente un echec), et qui conserve le code. Nous enumérons ensuite les propriétés que peut avoir un tel algorithme. A l'aide de la distance de Hamming généralisée, nous pourrons élargir cette définition et ces propriétés au décodage d'erreur et d'effacement simultanement. La section 4 detaille le décodage des codes BCH à l'aide de l'algorithme d'Euclide étendu.

Enfin dans la section 5 nous reprenons la notion de polynôme des motifs corrigibles introduite par Paul Camion dans [21], et nous la généralisons. Cet outil de combinatoire énumérative permet de décrire avec précision les performances d'un algorithme de décodage, et ces polynômes que nous décrivons peuvent être calculé dans certain cas particuliers. Nous définissons également la capacité de correction d'un code pour un algorithme donné qui permet de mesurer les performances d'un algorithme qui décode au-delà de la distance minimale du code.

1 Rappels sur les corps finis

Nous rappellons dans cette section les notions sur les corps finis et leurs extensions qui seront utiles dans ce travail. Les énoncés donnés ici sont largement inspirés de [63], ouvrage auquel nous renvoyons le lecteur s'il désire les preuves complètes des résultats exposés ci-dessous.

1.1 Corps finis et extensions

Définition I.1 *Soit $p > 1$ un nombre premier. On note $\mathbb{F}_p$ le corps fini à p éléments $\mathbb{Z}/p\mathbb{Z}$.*

Définition I.2 *Soient K un sous-corps du corps K' et d la dimension de K' en tant que K-espace vectoriel. On dit que K' est une extension de degré d de K.*

Théorème I.1 *Soit F un corps fini de cardinal $q > 1$. Alors*

1. *$q = p^m$, où p est un nombre premier et m un entier positif.*
2. *F est unique à isomorphisme près.*

Le corps de cardinal $q = p^m$, noté $\mathbb{F}_q$ est une extension de degré m du corps premier $\mathbb{F}_p$.

Proposition I.1 *Pour tout entier $m > 1$, $\mathbb{F}_{q^m}$ est une extension de degré m de $\mathbb{F}_q$.*

Théorème I.2 *Le groupe multiplicatif de $\mathbb{F}_q$, noté $\mathbb{F}_q^*$ est cyclique.*

Définition I.3 *Un générateur de $\mathbb{F}_q^*$ est appelé* élément primitif *de $\mathbb{F}_q$.*

Théorème I.3 *Soit $f(X) \in \mathbb{F}_q[X]$ un polynôme irréductible de degré m. Alors $f(X)$ possède une racine α dans $\mathbb{F}_{q^m}$. De plus les racines de $f(X)$ sont simples et sont données par les éléments distincts suivants $\alpha, \alpha^q, \alpha^{q^2}, \dots, \alpha^{q^{m-1}}$ de $\mathbb{F}_{q^m}$ qui sont appelés les* conjugés *de α pour $\mathbb{F}_q$.*

Si α est un élément primitif de $\mathbb{F}_{q^m}$, alors ses conjugés sont aussi des éléments primitifs. On dira alors que $f(X)$ est un polynôme primitif.

Théorème I.4 *Soit $\mathbb{F}_q$ un corps fini, et soient m et m' deux entiers strictement positifs.*

$$\mathbb{F}_{q^{m'}} \text{ est isomorphe à un sous-corps de } \mathbb{F}_{q^m} \Leftrightarrow m`|m.$$

De plus $\mathbb{F}_{q^{m'}}$ est isomorphe à tout sous-corps $\{\beta \in \mathbb{F}_{q^m} \mid \beta^{q^{m'}} = \beta\}$.

Remarque I.1 Si $\alpha \in \mathbb{F}_{q^m}$ est dans le sous-corps $\mathbb{F}_{q^{m'}}$ de $\mathbb{F}_{q^m}$, alors ses conjugés seront $\alpha, \alpha^q, \alpha^{q^2}, \dots, \alpha^{q^{m'-1}}$ et $m'|m$.

1.2 Représentation polynomiale d'un corps fini

Si $f(X) \in \mathbb{F}_q[X]$, on note $(f(X))$ l'idéal de $\mathbb{F}_q[X]$ engendré par $f(X)$.

Proposition I.2 *Soient un entier $m > 1$ et $f(X) \in \mathbb{F}_q[X]$ un polynôme irréductible de degré m. Alors l'anneau quotient $\mathbb{F}_q[X]/(f(X))$ est un corps fini de cardinal q^m, isomorphe à $\mathbb{F}_{q^m}$.*

Cette proposition nous fournit une représentation des éléments de $\mathbb{F}_{q^m}$ par des polynômes à coefficient dans $\mathbb{F}_q$ de degré au plus $m-1$ en une racine α de $f(X)$.

Proposition I.3 *Soient un entier $m > 1$ et $f(X) \in \mathbb{F}_q[X]$ un polynôme irréductible primitif de degré m. L'ensemble des restes de la division par $f(X)$ des éléments de l'ensemble $\{1, X, X^2, \ldots, X^{q^m-1}\}$ est égal à l'anneau quotient $\mathbb{F}_q[X]/(f(X))$.*

Dans le cas où $f(X)$ est primitif, on a donc une représentation de $\mathbb{F}_{q^m}$ par des polynômes de degré au plus $m-1$ en une racine α de $f(X)$, et une représentation cyclique de $\mathbb{F}_{q^m}^*$ par les puissances de α.

Définition I.4 *Soit $n | q^m - 1$. On appelle* racine n-ième de l'unité sur $\mathbb{F}_q$, *un élément de $\mathbb{F}_{q^m}$ dont l'ordre divise n, on appelle* racine n-ième primitive de l'unité sur $\mathbb{F}_q$, *un élément d'ordre n de $\mathbb{F}_{q^m}$, en particulier si $n = q^m - 1$, une racine primitive n-ième de l'unité sur $\mathbb{F}_q$ est un élément primitif de $\mathbb{F}_{q^m}$.*

Définition I.5 *Soit $n | q^m - 1$. Pour tout entier i, $0 \leq i < n$, on définit la* classe cyclotomique *de i modulo n sur $\mathbb{F}_q$ comme l'ensemble*

$$\mathrm{cl}(i) = \{i, qi, q^2 i, \ldots, q^{m-1} i\} \bmod n.$$

Le cardinal d'une classe cyclotomique modulo n sur $\mathbb{F}_q$ sera un diviseur de m, où m est le plus petit entier tel que $n | q^m - 1$. En fait, si α est une racine primitive n-ième de l'unité sur $\mathbb{F}_q$, alors la classe cyclotomique de i est l'ensemble des logarithmes en base α des conjugés de α^i, c'est-à-dire que si $\mathbb{F}_{q^{m'}}$ est le plus petit sous-corps de $\mathbb{F}_{q^m}$ contenant α^i, alors

$$\mathrm{cl}(i) = \{i, qi, q^2 i, \ldots, q^{m'-1} i\} \bmod n,$$

l'ensemble des conjugés de α^i étant

$$\{\alpha^i, \alpha^{qi}, \alpha^{q^2 i}, \ldots, \alpha^{q^{m'-1} i}\},$$

et de plus ces deux ensembles ont pour cardinal m', qui est un diviseur de m en vertu du théorème I.4

Définition I.6 *Soit $\alpha \in \mathbb{F}_{q^m}$. Le* polynôme minimal *de α dans $\mathbb{F}_q$ est le polynôme unitaire de plus bas degré $f(X) \in \mathbb{F}_q[X]$ vérifiant $f(\alpha) = 0$.*

Proposition I.4 *Soit $\alpha \in \mathbb{F}_{q^m}$. Le polynôme minimal de α est le polynôme $f(X)$ scindé dans $\mathbb{F}_{q^m}$ dont les racines sont les conjugés de α pour $\mathbb{F}_q$*

$$f(X) = (X - \alpha)(X - \alpha^q) \dots (X - \alpha^{q^{m'-1}}),$$

$\mathbb{F}_{q^{m'}}$ étant le plus petit sous-corps de $\mathbb{F}_{q^m}$ contenant α.

Corollaire I.1 *Si α est un élément primitif de $\mathbb{F}_{q^m}$, son polynôme minimal est primitif de degré m.*

2 Codes correcteurs – Définitions et propriétés

Nous rappelons dans cette section les principales notions de théorie des codes, et en particulier la définition des codes BCH qui font l'objet du second chapitre de ce travail, et qui seront également utilisés pour illustrer les deux chapitres suivant.

Tous les résultats exposés dans cette section sont extraits de [67].

2.1 Définitions

Soient n et q deux entiers positifs, soit A un alphabet à q éléments.

Définition I.7 *Soient A^n l'ensemble des mots de longueur n sur A, $\mathbf{x} = (x_0, x_1, \dots, x_{n-1}) \in A^n$ et $\mathbf{y} = (y_0, y_1, \dots, y_{n-1}) \in A^n$.*

- *La* distance de Hamming *entre $\mathbf{x}$ et $\mathbf{y}$ est*

  $$d_H(\mathbf{x}, \mathbf{y}) = |\{i \mid 0 \leq i \leq n-1,\ x_i \neq y_i\}|.$$

  on vérifie que d_H est bien une métrique et on appelle alors espace de Hamming sur A *l'ensemble A^n muni de la métrique d_H.*

- *Si A est un groupe, le* poids de Hamming $w_H(\mathbf{x})$ *d'un mot $\mathbf{x} \in A^n$ est le nombre de ses coordonnées non nulles*

  $$w_H(\mathbf{x}) = d_H(\mathbf{x}, \mathbf{0}),$$

  où $\mathbf{0}$ est le mot de A^n ayant toutes ses coordonnées égales à l'elément neutre de A.

- *Un* code C *sur A est une partie non vide de l'espace de Hamming A^n, n est appelé* longueur *du code, les éléments de C sont appelés* mots de code.

Définition I.8 *On appelle* distance minimale *d'un code C sur A, l'entier*

$$d = min\{d_H(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\},$$

si A est un groupe, on appelle poids minimal *d'un code C, l'entier*

$$min\{w_H(\mathbf{x}) \mid \mathbf{x} \in C, \mathbf{x} \neq \mathbf{0}\}.$$

Définition I.9 *On appelle* code parfait *un code tel que l'ensemble des boules de rayon* $\lfloor\frac{d-1}{2}\rfloor$ *centrées en tous les éléments du code forment une partition de l'espace de Hamming* A^n.

Définition I.10 *Un code* C *est dit* linéaire *sur* A*, si* A *est un corps et* C *un sous-espace-vectoriel de* A^n*. La dimension de* C *sur* A *est appelée la* dimension *du code* C.

On notera $C(n,k,d)$ *un code linéaire* C *de longueur* n *de dimension* k *et de distance minimale* d*. On dira que* n*,* k *et* d *sont les* paramètres *du code.*

Remarque I.2 Pour un code linéaire le *poids minimal* est égal à la *distance minimale.*

Définition I.11 *Une* matrice génératrice G *d'un code linéaire* C *est une matrice* $k\times n$ *dont les lignes forment une base de* C.

2.2 Les codes BCH

L'alphabet est égal à $\mathbb{F}_q$, le corps à q éléments. Soit n un entier. On note $\mathcal{R}_n = \mathbb{F}_q[X]/(X^n-1)$ l'anneau des polynômes à coefficients dans $\mathbb{F}_q$ modulo l'idéal engendré par X^n-1. Soit α une racine primitive n-ième de l'unité sur $\mathbb{F}_q$.

2.2.1 Définitions

Définition I.12 (codes cycliques) *Un* code cyclique *de longueur* n *sur* $\mathbb{F}_q$ *est un idéal de* $\mathcal{R}_n$.

Remarque I.3

1. Un code cyclique est linéaire.
2. Un code cyclique est invariant par permutation circulaire de ses coordonnées dans la base $(1,X,X^2,\ldots,X^{n-1})$de $\mathcal{R}_n$. La permutation circulaire d'une position à droite correspond à la multiplication par X modulo X^n-1

$$\begin{aligned} & X\left(a_0+a_1X+\ldots+a_{n-2}X^{n-2}+a_{n-1}X^{n-1}\right) \\ = \;& a_0X+a_1X^2+\ldots+a_{n-2}X^{n-1}+a_{n-1}X^n \\ = \;& a_{n-1}+a_0X+a_1X^2+\ldots+a_{n-2}X^{n-1} \bmod X^n-1. \end{aligned}$$

Théorème I.5 *Soit* C *un code cyclique de longueur* n *sur* $\mathbb{F}_q$*, on a*

i. *Il existe un unique polynôme unitaire* $g(X)$ *de degré minimal dans* C.

ii. C *est l'idéal engendré par* $g(X)$*,* $g(X)$ *est appelé* polynôme générateur *de* C.

iii. $g(X)$ *est un facteur de* X^n-1.

iv. *Tout* $c(X)\in C$ *s'écrit de façon unique* $c(X)=f(X)g(X)$ *dans* $\mathbb{F}_q[X]$*. La dimension de* C *est* $n-r$ *où* $r=\deg g(X)$.

Définition I.13 (codes BCH) *Soit C un code cyclique le longueur n sur $\mathbb{F}_q$ et soit $g(X)$ son polynôme générateur. C est un code BCH de* distance construite δ *si δ est le plus grand entier vérifiant*

$$\exists\, b \in \mathbb{Z},\ g(\alpha^b) = g(\alpha^{b+1}) = \ldots = g(\alpha^{b+\delta-2}) = 0,$$

où α est une racine primitive n-ième de l'unité dans l'extension $\mathbb{F}_{q^m}$ de $\mathbb{F}_q$, m étant le plus petit entier tel que $m|q^m - 1$.

Définition I.14 *Un code BCH est dit* primitif *s'il existe un entier m tel que $n = q^m - 1$, il est dit BCH* au sens strict *si $b = 1$. On notera $B(n, \delta)$ le code BCH binaire primitif au sens strict de longueur n et de distance construite δ.*

Remarque I.4 $B(n, \delta)$ est l'ensemble des polynômes de $\mathcal{R}_n$ admettant comme racines $\alpha, \alpha^2, \ldots, \alpha^{\delta-1}$.

Définition I.15 *Soit un code cyclique C de longueur n sur $\mathbb{F}_q$. On appelle* zéro *de C, toute puissance de α qui soit racine de $g(X)$.*

Définition I.16 *Soit un code cyclique C de longueur n sur $\mathbb{F}_q$.* L'ensemble de définition *de C, noté $I(C)$, est égal à l'ensemble des exposants des zéros de C :*

$$I(C) = \{i \mid g(\alpha^i) = 0\}.$$

Remarque I.5

1. $g(X)$ divise $X^n - 1$, donc toutes ses racines sont des racines n-ièmes de l'unité sur $\mathbb{F}_q$, les zéros du code sont donc bien des puissances de α.
2. $I(C)$ est l'ensemble des entiers i tels que $\forall\, c(X) \in C$, $c(\alpha^i) = 0$, si C est un code BCH de longueur n sur $\mathbb{F}_q$ et de distance construite δ, $I(C)$ contient la réunion des classes cyclotomiques modulo n sur $\mathbb{F}_q$ des entiers $b, b+1, \ldots, b+\delta-2$.

Définition I.17 *Le code de Reed-Muller raccourci $\mathcal{R}^*(\nu, m)$ de longueur $n = q^m - 1$ sur $\mathbb{F}_q$, d'ordre $\nu \in [1, m(q-1)[$, est le code cyclique dont l'ensemble de définition est*

$$I(C_\nu(m, q)) = \{i \in [1..n[\, \mid w_q(i) < m(q-1) - \nu\}.$$

Où w_q le q-poids est défini pour tout $s \in [1, n]$, par

$$w_q(s) = \sum_{i=0}^{m-1} s_i,$$

où les s_i sont définis par l'écriture de s en base q, $s = \sum_{i=0}^{m-1} s_i q^i$.

2.2.2 Distance construite et distance minimale des code BCH

Théorème I.6 (borne BCH) *Soit un code BCH de distance construite δ, sa distance minimale d vérifie*

$$d \geq \delta.$$

C'est ce théorème qui justifie le terme de distance construite donné à δ. Il faut noter de plus que pour les codes BCH primitifs au sens strict cette borne est atteinte la plupart du temps.

Le plus petit code BCH binaire primitif au sens strict dont la distance minimale est strictement supérieure à la distance construite est le code $B(127, 29)$ dont la distance minimale vaut 31, c'est le seul en longueur 127. En longueur 255 tous les codes BCH primitifs au sens strict ont pour distance minimale leur distance construite, sauf éventuellement $B(255, 59)$ et $B(255, 61)$. Nous montrons dans le chapitre II que ces deux codes dépassent leur distance construite, leur distance minimale vaut respectivement 61 et 63. Le plus petit code BCH binaire primitif au sens strict dépassant sa distance construite de plus de 2 est le code $B(511, 123)$ dont la distance minimale vaut 127, ce resultat a également été obtenu dans le cadre de ce travail, et est exposé dans le chapitre II.

Proposition I.5 *La distance minimale d'un code BCH binaire primitif au sens strict est impaire.*

Proposition I.6 *La distance construite d'un code BCH primitif au sens strict de longueur n sur $\mathbb{F}_q$ est l'élement minimal d'une classe cyclotomique modulo n sur $\mathbb{F}_q$.*

2.3 Les codes de Reed-Solomon

Définition I.18 *Un code de Reed-Solomon est un code BCH sur $\mathbb{F}_q$ de longueur $n = q - 1$. Si ce code a pour dimension k et distance minimale d, nous le noterons $RS(n, k, d)$.*

Proposition I.7 (borne de Singleton) *[67, p. 33] Si C est un code linéaire de paramètres (n, k, d), alors*

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

Définition I.19 *Un code de longueur n, de dimension k, de distance minimale d est MDS (*de l'anglais "maximum distance separable"*) s'il atteint la borne de Singleton*

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

Proposition I.8 *Les codes de Reed-Solomon sont MDS.*

2.4 Polynôme énumérateur des poids

Définition I.20 *Soit C un code linéaire de longueur n et soit A_i le nombre de mots de poids i. On appelle polynôme énumérateur des poids de C, le polynôme*

$$W_C(x,y) = \sum_{i=0}^{n} A_i x^{n-i} y^i.$$

Définition I.21 *Soit C un code linéaire (n,k,d). Le* code dual *de C, noté $C^\perp$, est l'orthogonal de C pour le produit scalaire usuel*

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=0}^{n} x_i y_i.$$

$C^\perp$ est un code linéaire $(n, n-k, d^\perp)$, la distance minimale $d^\perp$ de $C^\perp$ est appelée distance duale *de C.*

Théorème I.7 (transformée de MacWilliams) *Soit C un code linéaire sur $\mathbb{F}_q$, on a*

$$|C|\, W_{C^\perp}(x,y) = W_C(x+(q-1)y, x-y) \tag{I.1}$$

qui est appelé identité de MacWilliams, *et puisque $(C^\perp)^\perp = C$*

$$|C^\perp|\, W_C(x,y) = W_{C^\perp}(x+(q-1)y, x-y) \tag{I.2}$$

Si A'_i est le nombre de mot de poids i de $C^\perp$, en dérivant la relation (I.2) s fois par rapport à la variable y, on obtient en $x=1, y=1$

$$\sum_{i=s}^{n} \binom{i}{s} \frac{A_i}{q^k} = \left(\frac{q-1}{q}\right)^s \sum_{i=0}^{s} \left(\frac{1}{1-q}\right)^i \binom{n-i}{n-s} A'_i \tag{I.3}$$

en particulier lorsque $s < d^\perp$ l'équation (I.3) donne

$$\sum_{i=s}^{n} \binom{i}{s} \frac{A_i}{q^k} = \binom{n}{s} \left(\frac{q-1}{q}\right)^s. \tag{I.4}$$

Ces relations sont appelées identités de Pless.

La distribution des poids des codes MDS

Proposition I.9 *[67, pp. 319–321] Soit $C(n,k,d)$ un code MDS sur $\mathbb{F}_q$, A_w le nombre de mots de poids w, on a*

$$\begin{aligned} A_w &= \binom{n}{w} \sum_{j=0}^{w-d} (-1)^j \binom{w}{j} (q^{w-d+1-j} - 1) \\ &= \binom{n}{w} (q-1) \sum_{j=0}^{w-d} (-1)^j \binom{w-1}{j} q^{w-d-j}. \end{aligned} \tag{I.5}$$

En particulier pour les poids d et $d+1$

$$A_d = (q-1)\binom{n}{d} \tag{I.6}$$

$$A_{d+1} = (q-1)(q-d)\binom{n}{d+1} \tag{I.7}$$

3 Théorie algébrique du décodage

Dans toute cette section on considère un code linéaire $C(n,k,d)$ sur $\mathbb{F}_q$.

3.1 Position du problème

Etant donné

- que $\mathbf{x} \in C$ est le "message transmis",
- que $\mathbf{x}$ est perturbé dans un canal bruité par l'erreur $\mathbf{e} \in \mathbb{F}_q^n$,
- que $\mathbf{y} = \mathbf{x} + \mathbf{e}$, le "message reçu", est le seul mot auquel le décodeur a accès,

le problème du décodage est de retrouver $\mathbf{x}$ à partir de $\mathbf{y}$.

Un algorithme de décodage de C devra donc prendre comme argument un élément de $\mathbb{F}_q^n$, et s'il se termine, rendre un élément du code C. Il devra également être déterministe, c'est-à-dire qu'un mot de l'espace $\mathbb{F}_q^n$ sera toujours décodé de la même manière. Nous proposons la définition suivante.

Définition I.22 *Soit un code linéaire $C(n,k,d)$ sur $\mathbb{F}_q$, un* algorithme de décodage d'erreur de C *est une application*

$$\begin{array}{rccc} \gamma & : & \mathbb{F}_q^n & \longrightarrow & C \cup \{\infty\} \\ & & \mathbf{y} & \longmapsto & \gamma(\mathbf{y}) \end{array}$$

telle que $\forall\, \mathbf{x} \in C$, $\gamma(\mathbf{x}) = \mathbf{x}$. Le fait que $\gamma(\mathbf{y}) = \infty$ signifiant que $\mathbf{y}$ n'a pas été décodé.

Soit e un entier positif, on dira que γ est borné par e, *si*

$$\forall\, \mathbf{y} \in \mathbb{F}_q^n,\ \forall\, \mathbf{x} \in C,\quad d_H(\mathbf{x},\mathbf{y}) < \frac{e}{2} \Rightarrow \gamma(\mathbf{y}) = \mathbf{x}$$

où d_H est la métrique de Hamming sur $\mathbb{F}_q$, on dira que γ est borné strictement par e *si on a de plus*

$$\gamma(\mathbf{y}) = \mathbf{x} \neq \infty \Rightarrow d_H(\mathbf{x},\mathbf{y}) < \frac{e}{2}$$

On dira que γ respecte la métrique d, *si*

$$\forall\, \mathbf{y} \in \mathbb{F}_q^n,\ \forall\, \mathbf{x} \in C,\quad \gamma(\mathbf{y}) \in C \Rightarrow d(\gamma(\mathbf{y}),\mathbf{y}) \le d(\mathbf{x},\mathbf{y})$$

On dira que γ est un algorithme linéaire *si*

$$\forall \mathbf{y} \in \mathbb{F}_q^n, \ \forall \mathbf{x} \in C, \quad \gamma(\mathbf{y}+\mathbf{x}) = \gamma(\mathbf{y}) + \mathbf{x},$$

avec la convention $\infty + \mathbf{x} = \infty$.

Remarque I.6 Une confusion est possible entre un algorithme linéaire et un algorithme de complexité linéaire. Nous reservons, dans tout ce travail, le terme d'algorithme linéaire à la propriété énoncée ci-dessus.

Proposition I.10 *Tout code C admet un algorithme de décodage d'erreur borné par sa distance minimale, respectant la distance de Hamming, et qui se termine pour toute entrée, on dit d'un tel algorithme de décodage qu'il est à* vraisemblance maximale *(*maximum likelyhood decoding*).*

preuve : Soit l'algorithme de décodage γ suivant

$$\forall \mathbf{y} \in \mathbb{F}_q^n, \ \gamma(\mathbf{y}) = \text{“le mot de } C \text{ le plus proche de } \mathbf{y} \text{ pour } d_H\text{”}$$

puisque C est fini, on peut parcourir le code pour trouver ce mot le plus proche, en cas de non unicité, les conflits peuvent être réglés en munissant C d'une relation d'ordre.

Notons que γ se termine pour toute entrée et respecte la distance de Hamming par définition.

Soit d la distance minimale de C, soient $\mathbf{y} \in \mathbb{F}_q^n$ et $\mathbf{x} \in C$, tels que

$$d_H(\mathbf{x}, \mathbf{y}) < \frac{d}{2}.$$

Alors $\mathbf{x}$ est le mot de C le plus proche de $\mathbf{y}$, sinon la définition de la distance minimale serait contredite, donc γ est borné par d. □

Notons que cet algorithme n'est pas unique en général, il suffit en fait qu'il existe un mot de $\mathbf{y} \in \mathbb{F}_q^n$ qui soit à égale distance de deux mots du code, auquel cas on a au moins deux algorithmes distincts à vraisemblance maximale.

Proposition I.11 *Soit un code linéaire $C(n, k, d)$ sur $\mathbb{F}_q$, muni d'un algorithme de décodage γ, la meilleure borne possible pour γ est*

$$2\left\lfloor\frac{d-1}{2}\right\rfloor + 1.$$

preuve : Notons que $2\lfloor\frac{d-1}{2}\rfloor + 1$ vaut $d+1$ si d est impair, et d si d est pair.

Soient $\mathbf{x}, \mathbf{x}' \in C$ tels que $d_H(\mathbf{x}, \mathbf{x}') = d$. Il existe alors $\mathbf{y} \in \mathbb{F}_q^n$ tel que

$$d_H(\mathbf{x}, \mathbf{y}) = \frac{d-1}{2}, \quad \text{et} \quad d_H(\mathbf{x}', \mathbf{y}) = \frac{d+1}{2},$$

si d est impair, ou

$$d_H(\mathbf{x}, \mathbf{y}) = \frac{d}{2}, \quad \text{et} \quad d_H(\mathbf{x}', \mathbf{y}) = \frac{d}{2},$$

si d est pair.

Donc si d est impair γ ne peut être borné par $d+2$, et si d est pair γ ne peut être borné par $d+1$. $\square$

3.2 Matrice de parité – Syndrome

Définition I.23 *On appelle* matrice de parité *du code $C(n,k,d)$ une matrice H à $n-k$ lignes et n colonnes sur $\mathbb{F}_q$, telle que $C = \ker H = \{\mathbf{x} \mid H\mathbf{x}^t = 0\}$.*

La matrice H détermine le code C. Remarquons que par un théorème fondamental d'algèbre linéaire, les lignes de H forment une base de l'orthogonal de C dans $\mathbb{F}_q^n$ pour le produit scalaire usuel.

Définition I.24 *Soit $\mathbf{y} \in \mathbb{F}_q^n$. Le* syndrome *de $\mathbf{y}$ est le vecteur de $\mathbb{F}_q^{n-k}$*

$$S(\mathbf{y}) = H\mathbf{y}^t.$$

l'application $S : \mathbb{F}_q^n \to \mathbb{F}_q^{n-k}$ ainsi définie est $\mathbb{F}_q$-linéaire.

H induit une relation d'équivalence sur $\mathbb{F}_q^n$

$$\mathbf{x} \,\mathcal{R}\, \mathbf{y} \Leftrightarrow H\mathbf{x}^t = H\mathbf{y}^t \Leftrightarrow H(\mathbf{x}-\mathbf{y})^t = 0 \Leftrightarrow \mathbf{x}-\mathbf{y} \in C.$$

Chacune des classes de cette relation d'équivalence est appelée “classe latérale” ou “translaté” de C.

Le code C est la classe de l'élément nul de $\mathbb{F}_q^n$. Plus généralement, deux éléments sont dans un même translaté si et seulement si ils ont le même syndrome.

Proposition I.12 *Il existe au plus un mot de poids $< d/2$ dans un translaté donné.*

preuve : Soient $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ tels que $\mathbf{x} \,\mathcal{R}\, \mathbf{y}$, $w_H(\mathbf{x}) < d/2$ et $w_H(\mathbf{y}) < d/2$. Alors $\mathbf{x}-\mathbf{y} \in C$ et $w_H(\mathbf{x}-\mathbf{y}) = d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{0}) + d(\mathbf{0},\mathbf{y}) < d/2 + d/2 = d$, en vertu de l'inégalité triangulaire. Donc $\mathbf{x} = \mathbf{y}$. $\square$

3.3 Décodage des codes linéaires

A partir de la matrice de parité et des syndromes on peut définir l'algorithme de décodage γ suivant :

1. Dans tout translaté $\mathbf{y} + C$, on choisit un élément de plus petit poids, $e(\mathbf{y})$ que l'on met dans une table indexée par le syndrome de $\mathbf{y}$. Remarquons que $e(\mathbf{y}+\mathbf{x}) = e(\mathbf{y})$ pour tout $\mathbf{x} \in C$

2. Si $\mathbf{y}$ est le mot reçu, on calcule son syndrome $S(\mathbf{y})$, et on lit $e(\mathbf{y})$ dans la table.

3. l'algorithme retourne $\gamma(\mathbf{y}) = \tilde{\mathbf{x}} = \mathbf{y} - e(\mathbf{y})$.

Proposition I.13 *γ est un algorithme à vraisemblance maximale, donc il est borné par sa distance minimale, et respecte la distance de Hamming. De plus cet algorithme est linéaire.*

preuve : Soit $\mathbf{y} \in \mathbb{F}_q^n$, $\tilde{\mathbf{e}}$ un élément de plus petit poids de la classe de $\mathbf{y}$

- $\gamma(\mathbf{y}) \in C$, en effet $\mathbf{y} \mathcal{R} \tilde{\mathbf{e}}$, donc $\gamma(\mathbf{y}) = \mathbf{y} - \tilde{\mathbf{e}} \mathcal{R} \mathbf{0}$.
- $\forall \mathbf{x} \in C$, $\mathbf{y} - \mathbf{x} \mathcal{R} \tilde{\mathbf{e}}$, donc $d_H(\mathbf{x}, \mathbf{y}) = w_H(\mathbf{y} - \mathbf{x}) \geq w_H(\tilde{\mathbf{e}}) = d_H(\gamma(\mathbf{y}), \mathbf{y})$, donc $\gamma(\mathbf{y})$ est un élément du code réalisant le minimum de la distance à $\mathbf{y}$.
- L'algorithme est linéaire. En effet, pour tout $\mathbf{x} \in C$, $\mathbf{y} \in \mathbb{F}_q^n$

$$\gamma(\mathbf{y} + \mathbf{x}) = \mathbf{y} + \mathbf{x} - e(\mathbf{y} + \mathbf{x}) = \mathbf{y} + \mathbf{x} - e(\mathbf{y}) = \gamma(\mathbf{y}) + \mathbf{x},$$

puisque $e(\mathbf{y} + \mathbf{x}) = e(\mathbf{y})$.

□

Le décodage des codes linéaires se ramène donc à la recherche d'un mot de plus petit poids d'un translaté donné par son syndrome. Cette recherche est généralement difficile, mais dans certains cas, comme celui des codes BCH, on possède un algorithme qui fonctionne pour certains translatés.

3.4 Décodage d'erreur et d'effacement simultanément

Un effacement étant un symbole erroné dont on connait la position, on convient de substituer le symbole ∞ en cette position. Nous sommes donc amené à raisonner sur des vecteurs dont les coordonnés sont dans $\mathbb{F}_q \cup \{\infty\}$. Nous étendons l'addition dans $\mathbb{F}_q$ en considérant ∞ comme un élément absorbant.

Dans la proposition suivante nous définissons une distance de Hamming généralisée sur $\mathbb{F}_q \cup \{\infty\}$qui diffère de celle définie par Wei dans [98], bien que portant le même nom.

Proposition I.14 *Soit $\delta_H : (\mathbb{F}_q \cup \{\infty\})^2 \to \{0, \frac{1}{2}, 1\}$ l'application définie par*

$$\delta_H(a, b) = \begin{cases} 0 & si\ a = b \\ \frac{1}{2} & si\ a \neq b,\ \ et\ a = \infty,\ \ ou\ b = \infty \\ 1 & si\ a \neq b,\ \ et\ a \neq \infty,\ b \neq \infty \end{cases}$$

Si $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in (\mathbb{F}_q \cup \{\infty\})^n$, on définit $\Delta_H : (\mathbb{F}_q \cup \{\infty\})^n \to \frac{1}{2}\mathbb{Z}$ par

$$\Delta_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \delta_H(x_i, y_i),$$

Δ_H est une distance sur $(\mathbb{F}_q \cup \{\infty\})^n$, on l'appelera distance de Hamming généralisée *sur $\mathbb{F}_q$.*

preuve : δ_H est une distance sur $\mathbb{F}_q \cup \{\infty\}$, en effet

1. $\delta_H(a,b) = 0 \Leftrightarrow a = b$
2. $\delta_H(a,b) = \delta_H(b,a)$
3. soit $a, b, c \in \mathbb{F}_q \cup \{\infty\}$, distincts, on a
   (a) $a \neq \infty$, $b \neq \infty$, et $c \neq \infty$, δ_H restreint à $\mathbb{F}_q$ est la distance de Hamming (en longueur 1 !), donc
   $$\delta_H(a,c) \leq \delta_H(a,b) + \delta_H(b,c).$$
   (b) $a = \infty$, ou $c = \infty$, alors
   $$\delta_H(a,c) = \frac{1}{2} \leq \delta_H(a,b) + \delta_H(b,c) = \frac{3}{2}.$$
   (c) $b = \infty$
   $$\delta_H(a,b) + \delta_H(b,c) = 1 \geq \delta_H(a,c).$$
   si a, b, c ne sont pas distincts, l'inégalité triangulaire est triviale (l'un des trois termes au moins est nul, les autres sont égaux).

De la relation
$$\Delta_H(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} \delta_H(x_i, y_i),$$
on déduit que Δ_H est également une distance. □

Définition I.25 *On appellera* poids de Hamming généralisé *d'un mot* $\mathbf{x} \in (\mathbb{F}_q \cup \{\infty\})^n$*, noté* $\Omega_H(\mathbf{x})$*, sa distance de Hamming généralisée au mot nul*
$$\Omega_H(\mathbf{x}) = \Delta_H(\mathbf{x}, \mathbf{0}).$$

Remarque I.7

1. La métrique Δ_H restreinte à $\mathbb{F}_q^n$ coïncide avec la métrique de Hamming. En particulier la distance minimale est conservée.
2. Le symbole ∞ représentant un effacement, si $\mathbf{y} \in (\mathbb{F}_q \cup \{\infty\})^n$ est effacé en ρ positions (i.e. $\mathbf{y}$ possède ρ fois le symbole ∞ dans son écriture), et diffère de $\mathbf{x} \in \mathbb{F}_q^n$ en ν des positions restantes, alors
   $$\Delta_H(\mathbf{x},\mathbf{y}) = \frac{2\nu + \rho}{2}.$$
3. Le poids 1/2 qui est donné aux effacements, est le plus petit tel que Δ_H soit une distance, tout en conservant la même distance minimale pour le code, et il correspond bien a l'idée intuitive qu'un effacement "coûte" 2 fois moins cher à corriger qu'une erreur.

Définition I.26 *Soit un code linéaire $C(n,k,d)$ sur $\mathbb{F}_q$. Un* algorithme de décodage d'erreur et d'effacement de C *est une application*

$$\begin{array}{rcl}\gamma \ : \ (\mathbb{F}_q \cup \{\infty\})^n & \longrightarrow & C \cup \{\infty\} \\ \mathbf{y} & \longmapsto & \gamma(\mathbf{y})\end{array}$$

telle que $\forall\, \mathbf{x} \in C$, $\gamma(\mathbf{x}) = \mathbf{x}$. Le fait que $\gamma(\mathbf{y}) = \infty$ signifiant que $\mathbf{y}$ n'a pas été décodé.

Soit e un entier positif. On dira que γ est borné par e*, si*

$$\forall\, \mathbf{y} \in (\mathbb{F}_q \cup \{\infty\})^n,\ \forall\, \mathbf{x} \in C, \quad \Delta_H(\mathbf{x},\mathbf{y}) < \frac{e}{2} \Rightarrow \gamma(\mathbf{y}) = \mathbf{x}$$

où Δ_H est la distance de Hamming généralisée sur $\mathbb{F}_q$. On dira que γ est borné strictement par e *si on a de plus*

$$\gamma(\mathbf{y}) = \mathbf{x} \neq \infty \Rightarrow \Delta_H(\mathbf{x},\mathbf{y}) < \frac{e}{2}.$$

On dira que γ respecte la distance d *si*

$$\forall\, \mathbf{y} \in (\mathbb{F}_q \cup \{\infty\})^n,\ \forall\, \mathbf{x} \in C, \quad \gamma(\mathbf{y}) \in C \Rightarrow d(\gamma(\mathbf{y}),\mathbf{y}) \leq d(\mathbf{x},\mathbf{y}).$$

On dira que γ est un algorithme linéaire *si*

$$\forall\, \mathbf{y} \in (\mathbb{F}_q \cup \{\infty\})^n,\ \forall\, \mathbf{x} \in C, \quad \gamma(\mathbf{y}+\mathbf{x}) = \gamma(\mathbf{y}) + \mathbf{x},$$

(on convient que $\mathbf{x} + \infty = \infty$ pour tout $x \in (\mathbb{F}_q \cup \{\infty\})^n$).

Remarque I.8 Le symbole ∞ est utilisé dans deux contexte différents, pour représenter l'effacement d'un symbole de $\mathbb{F}_q$ et pour représenter un echec au décodage.

Proposition I.15 *Tout code C admet un algorithme de décodage d'erreur et d'effacement borné par sa distance minimale, respectant la distance de Hamming généralisée, et qui se termine pour toute entrée, on dit d'un tel algorithme de décodage qu'il est à* vraisemblance maximale (maximum likelyhood decoding).

preuve : Soit l'algorithme de décodage γ suivant

$$\forall\, \mathbf{y} \in (\mathbb{F}_q \cup \{\infty\})^n,\ \gamma(\mathbf{y}) = \text{“le mot de } C \text{ le plus proche de } \mathbf{y} \text{ pour } \Delta_H\text{”}$$

puisque C est fini, on peut parcourir le code pour trouver ce mot le plus proche, en cas de non unicité, les conflits peuvent être réglés en munissant C d'une relation d'ordre.

- γ se termine pour toute entrée et respecte la distance de Hamming généralisée par définition.
- Soit d la distance minimale de C, soient $\mathbf{y} \in (\mathbb{F}_q \cup \{\infty\})^n$ et $\mathbf{x} \in C$, tels que

  $$\Delta_H(\mathbf{x},\mathbf{y}) < \frac{d}{2}.$$

  Alors $\mathbf{x}$ est le mot de C le plus proche de $\mathbf{y}$, sinon la définition de la distance minimale serait contredite, donc γ est borné par d.

□

Proposition I.16 *Soit un code linéaire $C(n,k,d)$ sur $\mathbb{F}_q$, muni d'un algorithme de décodage d'erreur et d'effacement γ, la meilleure borne possible pour γ est la distance minimale d de C.*

preuve :
Soit $\mathbf{x} \in C$, tel que $w_H(\mathbf{x}) = d$. Alors il existe $\mathbf{y} \in (\mathbb{F}_q \cup \{\infty\})^n$ tel que

$$\Delta_H(\mathbf{x}, \mathbf{y}) = \frac{d}{2}, \quad \text{et} \quad \Delta_H(\mathbf{0}, \mathbf{y}) = \frac{d}{2}.$$

En effet, supposons sans perte de généralité que $\mathbf{x} = (x_1, \ldots, x_d, 0, \ldots, 0)$. Si d est pair, on choisit

$$\mathbf{y} = (x_1, \ldots, x_{\frac{d}{2}}, 0, \ldots, 0),$$

et si d est impair, on choisit

$$\mathbf{y} = (x_1, \ldots, x_{\frac{d-1}{2}}, \infty, 0, \ldots, 0).$$

Donc γ ne peut pas être borné par $d+1$. □

4 Décodage des codes BCH

Soit α une racine n-ième primitive de l'unité dans le corps $\mathbb{F}_{q^m}$ où $n | q^m - 1$. $B_q(n, \delta, b)$ est le code BCH sur $\mathbb{F}_q$ de longueur n, et de distance construite δ, défini par

$$B_q(n,\delta,b) = \{c(X) \in \mathbb{F}_q[X]/(X^n - 1) \;\mid\; c(\alpha^b) = c(\alpha^{b+1}) = \ldots = c(\alpha^{b+\delta-2}) = 0\}.$$

Ce code admet pour matrice de parité :

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

Les sections 4.1, 4.2 et 4.3 sont inspirées de [67], la section 4.4 de [75]. Toutefois la présentation basée sur le théorème I.9 est originale.

4.1 Polynômes localisateur et évaluateur

Soit $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}_q^n$ de poids w et soient :

$$a_{i_1}, a_{i_2}, \ldots, a_{i_w}$$

les composantes non nulles de ce vecteur. On associe à **a** les éléments suivants de $\mathbb{F}_{q^m}$:

$$X_1 = \alpha^{i_1}, X_2 = \alpha^{i_2}, \ldots, X_w = \alpha^{i_w}$$

appelés *localisateurs* de **a**, ainsi que les éléments suivants de $\mathbb{F}_q$:

$$Y_1 = a_{i_1}, Y_2 = a_{i_2}, \ldots, Y_w = a_{i_w}$$

appelés *évaluateurs* de **a**.

Définition I.27 *Le* polynôme localisateur *de* **a** *est le polynôme :*

$$\sigma(z) = \prod_{i=1}^{w}(1 - X_i z) = \sum_{i=0}^{w} \sigma_i z^i \in \mathbb{F}_{q^m}[z].$$

Les racines de $\sigma(z)$ sont les inverses des localisateurs de **a**.

Les coefficients σ_i de $\sigma(z)$ sont les fonctions symétriques élémentaires des X_i :

$$\begin{cases} \sigma_1 = -(X_1 + \ldots + X_w), \\ \sigma_2 = X_1X_2 + X_1X_3 + \ldots + X_{w-1}X_w, \\ \vdots \\ \sigma_w = (-1)^w X_1 \ldots X_w. \end{cases}$$

Définition I.28 *Pour tout entier b, on définit un* polynôme évaluateur $\omega_b(z) \in \mathbb{F}_{q^m}[z]$ *de* **a** *par :*

$$\omega_b(z) = \sum_{i=1}^{w} Y_i X_i^b \prod_{\substack{j=1 \\ j \neq i}}^{w} (1 - X_j z).$$

On a pour tout i, $1 \leq i \leq w$:

$$Y_i = \frac{X_i^{-b}\omega_b(X_i^{-1})}{\prod_{j \neq i}(1 - X_j X_i^{-1})}.$$

De la connaissance de $\sigma(z)$ et $\omega_b(z)$ on peut donc déterminer entièrement le vecteur **a**.

Théorème I.8 *Soit* $\mathbf{a} = (a_0, \ldots, a_{n-1}) \in \mathbb{F}_q^n$, *$\sigma(z)$ le polynôme localisateur de* **a**, *$\omega_b(z)$ son polynôme évaluateur. On associe à* **a** *le polynôme $a(z) = a_0 + a_1 z + \ldots + a_{n-1}z^{n-1}$, et pour tout entier j, on pose $A_j = a(\alpha^j)$. On a*

$$\omega_b(z) = S_b(z)\sigma(z) \tag{I.8}$$

où

$$S_b(z) = \sum_{j=0}^{\infty} A_{j+b} z^j.$$

preuve : On a :

$$\begin{aligned}
\frac{\omega_b(z)}{\sigma(z)} &= \sum_{i=1}^{w} \frac{Y_i X_i^b}{1 - zX_i}, \\
&= \sum_{i=1}^{w} Y_i X_i^b \sum_{j=0}^{\infty} (zX_i)^j, \\
&= \sum_{j=0}^{\infty} z^j \sum_{i=1}^{w} Y_i X_i^{j+b}, \\
&= \sum_{j=0}^{\infty} z^j \sum_{i=0}^{n-1} a_i \alpha^{i(j+b)}, \\
&= \sum_{j=0}^{\infty} A_{j+b} z^j,
\end{aligned}$$

ce qui démontre le théorème. □

Remarque I.9 Le théorème I.8 reste vrai si l'un ou plusieurs des Y_i est nul.

Théorème I.9 *Soient $S(z) \in \mathbb{F}_{q^m}[z]$ et δ un entier positif. Pour tout entier positif ou nul ν, on considère le système d'indéterminées $f(z)$ et $g(z)$ dans $\mathbb{F}_{q^m}[z]$:*

$$\left\{ \begin{array}{l} g(z) \equiv f(z)S(z) \bmod z^{\delta-1} \\ \deg f \leq \nu \\ \deg g + \nu < \delta - 1. \end{array} \right. \tag{I.9}$$

Il existe au plus, à un scalaire multiplicatif près, un couple $(f(z), g(z))$ de polynômes premiers entre eux solution de ce système. Toute solution $(f'(z), g'(z))$ de (I.9) est alors de la forme :

$$\left\{ \begin{array}{l} f'(z) = \lambda(z)f(z) \\ g'(z) = \lambda(z)g(z). \end{array} \right.$$

preuve : Soient $(f(z), g(z))$ et $(f'(z), g'(z))$ deux couples solutions de (I.9), avec $f(z)$ et $g(z)$ premiers entre eux. On a

$$g(z)f'(z) \equiv f(z)S(z)f'(z) \equiv f(z)g'(z) \bmod z^{\delta-1},$$

or $\deg(gf') < \delta - 1$ et $\deg(fg') < \delta - 1$, donc

$$g(z)f'(z) = f(z)g'(z),$$

et puisque $f(z)$ et $g(z)$ sont premiers entre eux, $f(z) \mid f'(z)$ et $g(z) \mid g'(z)$, donc

$$\left\{ \begin{array}{l} f'(z) = \lambda(z)f(z) \\ g'(z) = \lambda(z)g(z) \end{array} \right.$$

ce qui prouve le théorème. □

Proposition I.17 *Soient un entier positif δ et $\mathbf{a} \in \mathbb{F}_q^n$ de poids $w \leq \lfloor \frac{\delta-1}{2} \rfloor$. Les polynômes localisateur et évaluateur de $\mathbf{a}$, $\sigma(z)$ et $\omega_b(z)$ sont solutions de (I.9) pour $S(z) = S_b(z) \bmod z^{\delta-1}$ et $\nu = \lfloor \frac{\delta-1}{2} \rfloor$, et sont premiers entre eux.*

preuve : On a $\deg \sigma = w \leq \lfloor \frac{\delta-1}{2} \rfloor$ et $\deg \omega_b < w \leq \delta - 1 - \lfloor \frac{\delta-1}{2} \rfloor$. Le théorème I.8 nous assure que $(\sigma(z), \omega_b(z))$ est solution de (I.9). De plus $\sigma(z)$ est scindé dans $\mathbb{F}_{q^m}$ et aucune de ses racines n'est racine de $\omega_b(z)$, donc $\mathrm{pgcd}(\sigma(z), \omega_b(z)) = 1$. □

4.2 L'algorithme d'Euclide étendu

Théorème I.10 (Algorithme d'Euclide) *Soient $r_{-1}(z)$ et $r_0(z)$ tels que $\deg r_0 \leq \deg r_{-1}$. On construit deux suites $(r_i(z))_{i\geq 1}$ et $(q_i(z))_{i\geq 1}$ telles que :*

$$\begin{array}{rcllcl}
r_{-1}(z) & = & q_1(z)r_0(z) + r_1(z), & \deg r_1 & < & \deg r_0 \\
r_0(z) & = & q_2(z)r_1(z) + r_2(z), & \deg r_2 & < & \deg r_1 \\
r_1(z) & = & q_3(z)r_2(z) + r_3(z), & \deg r_3 & < & \deg r_2 \\
 & \vdots & & & \vdots & \\
r_{j-2}(z) & = & q_j(z)r_{j-1}(z) + r_j(z), & \deg r_j & < & \deg r_{j-1} \\
r_{j-1}(z) & = & q_{j+1}(z)r_j(z). & & &
\end{array}$$

Alors $r_j(z)$, le dernier reste non nul de ces divisions est le pgcd de $r_{-1}(z)$ et $r_0(z)$.

La preuve de ce théorème est omise.

Soient les deux suites de polynômes $(U_i(z))_i$ et $(V_i(z))_i$ définies par :

$$\begin{array}{ll}
U_{-1}(z) = 0, & U_0(z) = 1, \\
V_{-1}(z) = 1, & V_0(z) = 0,
\end{array}$$

$$\begin{array}{l}
U_i(z) = U_{i-2}(z) - q_i(z)U_{i-1}(z), \\
V_i(z) = V_{i-2}(z) - q_i(z)V_{i-1}(z).
\end{array}$$

Proposition I.18 *Pour tout $i \geq 0$ on a :*

$$\left\{\begin{array}{l}
r_i(z) = U_i(z)r_0(z) + V_i(z)r_{-1}(z) \\
\deg U_i = \deg r_{-1} - \deg r_{i-1} \\
U_i(z)V_{i-1}(z) - U_{i-1}(z)V_i(z) = (-1)^i \\
\quad \textit{(donc } U_i(z) \textit{ et } V_i(z) \textit{ sont premiers entre eux)}
\end{array}\right. \tag{I.10}$$

preuve : (I.10) est vrai pour $i = 0$, supposons la propriété vérifiée pour tous les entiers inférieurs ou égaux à i.

1. D'après le théorème I.10 on a

$$r_{i-1}(z) = q_{i+1}(z)r_i(z) + r_{i+1}(z),$$

en appliquant (I.10) à i et $i-1$, il vient

$$U_{i-1}(z)r_0(z) + V_{i-1}(z)r_{-1}(z) = q_{i+1}(z)(U_i(z)r_0(z) + V_i(z)r_{-1}(z)) + r_{i+1}(z),$$

en regroupant et à l'aide de la définition de $U_i(z)$ et $V_i(z)$

$$r_{i+1}(z) = (U_{i-1}(z) - q_{i+1}(z)U_i(z))r_0(z) + (V_{i-1}(z) - q_{i+1}(z)V_i(z))r_{-1}(z),$$

$$r_{i+1}(z) = U_{i+1}(z)r_0(z) + V_{i+1}(z)r_{-1}(z).$$

2. On a

$$U_{i+1}(z) = U_{i-1}(z) - q_{i+1}(z)U_i(z),$$

et $\deg U_{i-1} \leq \deg U_i$, d'où

$$\deg U_{i+1} = \deg q_{i+1} + \deg U_i.$$

Or $\deg q_{i+1} = \deg r_{i-1} - \deg r_i$, ce qui prouve en utilisant l'hypothèse de récurrence

$$\deg U_{i+1} = \deg r_{-1} - \deg r_i.$$

3. En utilisant la définition de $U_i(z)$ et $V_i(z)$ et l'hypothèse de récurrence, on a

$$\begin{aligned} U_{i+1}(z)V_i(z) - U_i(z)V_{i+1}(z) &= (U_{i-1}(z) - q_{i+1}(z)U_i(z))V_i(z) \\ &\quad -U_i(z)(V_{i-1}(z) - q_{i+1}(z)V_i(z)) \\ &= -(U_i(z)V_{i-1}(z) - U_{i-1}(z)V_i(z)) \\ &= (-1)^{i+1}. \end{aligned}$$

Ceci achève la démonstration par récurrence de la proposition. □

4.3 L'algorithme de décodage

Dans toute cette section, on considère le code BCH $C = B_q(n, \delta, b) \subset \mathbb{F}_q^n$ de matrice de parité H, et on pose $t = \lfloor \frac{\delta-1}{2} \rfloor$.

4.3.1 Quel problème doit on résoudre?

Supposons $\mathbf{a} \in C$ le mot émis et $\mathbf{m} = \mathbf{a} + \mathbf{e} \in \mathbb{F}_q^n$ le mot reçu. Connaissant $\mathbf{m}$ on veut déterminer $\mathbf{e} = (e_0, \ldots, e_{n-1})$, en supposant que le poids de $\mathbf{e}$ est suffisamment petit.

Le décodeur a accès à $H\mathbf{m}^t = H\mathbf{e}^t$, ce qui revient à connaitre les valeurs en α^b, α^{b+1}, ..., $\alpha^{b+\delta-2}$ du polynôme d'erreur $e(z) = e_0 + e_1 z + \ldots + e_{n-1}z^{n-1}$. Si l'on se réfère à la section 4.1, il suffit de calculer les valeurs des polynômes localisateur et évaluateur, $\sigma(z)$ et $\omega_b(z)$, de $\mathbf{e}$.

4.3.2 Description de l'algorithme

Théorème I.11 *Soit* $\mathbf{e} = (e_0, \ldots, e_{n-1}) \in \mathbb{F}_q^n$ *de poids* $w \leq t$*, on considère les polynômes localisateur et évaluateur de* $\mathbf{e}$*,* $\sigma(z)$ *et* $\omega_b(z)$*, ainsi que la série formelle*

$$S_b(z) = \sum_{i=0}^{\infty} e(\alpha^{i+b}) z^i,$$

où $e(z) = e_0 + e_1 z + \ldots + e_{n-1} z^{n-1}$.

On applique l'algorithme d'Euclide aux polynômes $r_0(z) = S_b(z) \bmod z^{\delta-1}$ *et* $r_{-1}(z) = z^{\delta-1}$*, et on considère les trois suites de polynômes* $(r_i(z))_i$*,* $(U_i(z))_i$ *et* $(V_i(z))_i$ *définies dans* 4.2.

1. *Il existe un rang* k *tel que* $\deg r_{k-1} \geq \delta - 1 - t$*, et* $\deg r_k < \delta - 1 - t$*,*

2. *les polynômes localisateur et évaluateur de* $\mathbf{e}$ *sont donnés par :*

$$\sigma(z) = \frac{U_k(z)}{U_k(0)} \quad \text{et} \quad \omega_b(z) = \frac{r_k(z)}{U_k(0)}.$$

preuve : Pour tout i positif, on a que

$$\begin{cases} r_i(z) = U_i(z) r_0(z) + V_i(z) z^{\delta-1} \\ \deg U_i = \delta - 1 - \deg r_{i-1} \\ \operatorname{pgcd}(U_i(z), V_i(z)) = 1 \end{cases} \tag{I.11}$$

que les degrés des $r_i(z)$ sont décroissants, et qu'il existe un rang à partir duquel la suite des $r_i(z)$ est nulle. De plus $\deg r_{-1} = \delta - 1 \geq \delta - 1 - t$, donc il existe un rang k pour lequel $\deg r_{k-1} \geq \delta - 1 - t$ et $\deg r_k < \delta - 1 - t$. On aura alors :

$$\begin{cases} r_k(z) = U_k(z) S_b(z) \bmod z^{\delta-1} \\ \deg U_k = \deg r_{-1} - \deg r_{k-1} \leq t \\ \deg r_k < \delta - 1 - t, \end{cases}$$

d'après le théorème I.9 et la proposition I.17, on peut affirmer que :

$$\begin{cases} U_k(z) = \lambda(z) \sigma(z) \\ r_k(z) = \lambda(z) \omega_b(z). \end{cases}$$

D'après le théorème I.8 on a

$$\omega_b(z) = \sigma(z) S_b(z),$$

et puisque $S_b(z) = r_0(z) \bmod z^{\delta-1}$, il existe un polynôme $\mu(z)$ tel que

$$\omega_b(z) = \sigma(z) r_0(z) + \mu(z) z^{\delta-1}.$$

D'après (I.11)

$$r_k(z) = U_k(z) r_0(z) + V_k(z) z^{\delta-1}$$

donc

$$\lambda(z)\omega_b(z) = \lambda(z)\sigma(z)r_0(z) + V_k(z)z^{\delta-1}$$

et

$$\lambda(z)\mu(z)z^{\delta-1} = V_k(z)z^{\delta-1}.$$

On a donc un polynôme $\lambda(z)$ qui vérifie :

$$\begin{cases} \lambda(z) \mid U_k(z) \\ \lambda(z) \mid V_k(z) \end{cases}$$

et comme $U_k(z)$ et $V_k(z)$ sont premiers entre eux, $\lambda(z)$ est une constante non nulle qui est égale à $\lambda(0) = U_k(0)$ puisque $\sigma(0) = 1$. □

Donc si $\mathbf{m} = \mathbf{a} + \mathbf{e}$ est connu, avec $\mathbf{a} \in C$ et $\mathbf{e}$ de poids au plus t, alors il est possible de déterminer $\mathbf{e}$, en effet :

- Soient $m(z)$ le polynôme associé à $\mathbf{m}$ et $e(z)$ le polynôme associé à $\mathbf{e}$. On a

$$m(\alpha^b) = e(\alpha^b), \ldots, m(\alpha^{b+\delta-2}) = e(\alpha^{b+\delta-2}).$$

  Donc si $m(z)$ est connu, $S_b(z) \bmod z^{\delta-1} = \sum_{i=0}^{\delta-2} e(\alpha^{b+i})z^i$ est connu.

- On applique alors l'algorithme d'Euclide étendu comme décrit dans le théorème I.11, ce qui nous permet de construire les polynômes localisateur et évaluateur de $\mathbf{e}$.

- d'après la définition I.28 du polynôme évaluateur, cela nous suffit à déterminer $\mathbf{e}$ et donc $\mathbf{a}$.

Proposition I.19 *L'algorithme d'Euclide étendu pour la correction d'erreur dans un code BCH de distance construite δ est un algorithme borné strictement par δ et respectant la distance de Hamming.*

preuve : L'algorithme d'Euclide est borné par δ d'après le théorème I.11, cette borne est stricte car dans tous les cas, le degré de σ étant au plus égal à t, on corrigera au plus t erreurs.

L'algorithme d'Euclide respecte la distance de Hamming car si $\mathbf{y}$ est décodé en $\mathbf{x} \in C$, alors $d_H(\mathbf{x}, \mathbf{y}) \leq t$, et pour tout $\mathbf{x}' \in C$ différent de $\mathbf{x}$, on a

$$d_H(\mathbf{x}', \mathbf{y}) \geq d_H(\mathbf{x}, \mathbf{x}') - d_H(\mathbf{x}, \mathbf{y}) \geq d - t > t.$$

□

Remarque I.10 Il existe un autre algorithme de décodage des codes BCH, l'algorithme de Berlekamp-Massey. J-L. Dornstetter a montré dans [30] que cet algorithme était équivalent à l'algorithme d'Euclide étendu, et P. Camion a proposé dans [19] une version itérative de l'algorithme d'Euclide qui reproduit pas à pas l'algorithme de Berlekamp-Massey.

4.4 Décodage d'erreur et d'effacement par l'algorithme d'Euclide étendu

Nous nous posons le problème du décodage d'un mot d'un code BCH de distance construite δ contenant ν erreurs et ρ effacements.

Commencons par remplacer les effacements par l'elément zéro. Le problème que l'on a à résoudre est alors le suivant : décoder une erreur $\mathbf{e}$ de poids $\nu+\rho$, sachant que l'on connait ρ des $\nu+\rho$ positions d'erreurs.

Dans toute cette section, on considèrera le vecteur $\mathbf{e}=(e_0,\ldots,e_{n-1})\in\mathbb{F}_q^n$ vérifiant pour deux entiers positifs ou nuls ν et ρ :

- ν positions de $\mathbf{e}$ appellées *positions d'erreur*, sont non nulles, leurs localisateurs sont :

$$X_1=\alpha^{i_1}, X_2=\alpha^{i_2},\ldots,X_\nu=\alpha^{i_\nu},$$

  et les valeurs respectives de $\mathbf{e}$ en ces positions sont :

$$Y_1=e_{i_1}, Y_2=e_{i_2},\ldots,Y_\nu=e_{i_\nu}$$

- $n-\nu-\rho$ positions de $\mathbf{e}$ sont nulles
- les ρ positions restantes ont une valeur quelconque, et son appelées *positions d'effacement*, leurs localisateurs sont :

$$X'_1=\alpha^{i'_1}, X'_2=\alpha^{i'_2},\ldots,X'_\rho=\alpha^{i'_\rho},$$

  et les valeurs respectives de $\mathbf{e}$ en ces positions sont :

$$Y'_1=e_{i'_1}, Y'_2=e_{i'_2},\ldots,Y'_\rho=e_{i'_\rho}$$

Définition I.29 *Le polynôme localisateur des erreurs de* $\mathbf{e}$ *est le polynôme :*

$$\sigma(z)=\prod_{i=1}^{\nu}(1-X_iz).$$

Le polynôme localisateur des effacements de $\mathbf{e}$ *est le polynôme :*

$$\sigma'(z)=\prod_{i=1}^{\rho}(1-X'_iz).$$

Le polynôme localisateur de $\mathbf{e}$ *est le polynôme :*

$$\Sigma(z)=\prod_{i=1}^{\nu}(1-X_iz)\prod_{i=1}^{\rho}(1-X'_iz)=\sigma(z)\sigma'(z).$$

Le polynôme évaluateur de $\mathbf{e}$ *est le polynôme :*

$$\Omega_b(z)=\sum_{i=1}^{\nu}{X_i}^bY_i\prod_{j\neq i}(1-X_jz)\prod_{j=1}^{\rho}(1-X'_jz)+\sum_{i=1}^{\rho}X_i'^bY'_i\prod_{j=1}^{\nu}(1-X_jz)\prod_{j\neq i}(1-X'_jz).$$

Ces définitions des polynômes localisateur et évaluateur coïncident avec celles données dans la section 4.1, à la difference près que certains des localisateurs peuvent correspondre à des positions nulles.Notons que le polynôme $\sigma'(z)$ est connu.

D'après la remarque I.9, le théorème I.8 reste valable et on a

$$\Omega_b(z) = \Sigma(z)S_b(z) = \sigma(z)\sigma'(z)S_b(z) \tag{I.12}$$

où

$$S_b(z) = \sum_{i=0}^{\infty} e(\alpha^{b+i})z^i$$

et $e(z)$ est le polynôme

$$e(z) = e_0 + e_1 z + \ldots + e_{n-1}z^{n-1}.$$

Nous avons l'analogue de la proposition I.17 :

Proposition I.20 *Supposons que*

$$2\nu + \rho < \delta$$

alors, les polynômes $\sigma(z)$ et $\Omega_b(z)$ sont solution de (I.9) *pour*

$$S(z) = \sigma'(z)S_b(z) \bmod z^{\delta-1},$$

de plus $\sigma(z)$ et $\Omega_b(z)$ sont premiers entre eux.

preuve : D'après (I.12), on a

$$\Omega_b(z) \equiv \sigma(z)S(z) \bmod z^{\delta-1}.$$

D'autre part $\deg\sigma = \nu$ par définition, et $\deg\Omega_b < \nu + \rho$, donc $\deg\Omega_b + \nu < 2\nu + \rho \leq \delta - 1$.

Comme $\sigma(z)$ est scindé dans $\mathbb{F}_{q^m}$, et qu'aucune de ses racines n'est racine de $\Omega_b(z)$, donc $\sigma(z)$ et $\Omega_b(z)$ sont premiers entre eux. □

Théorème I.12 *On applique l'algorithme d'Euclide aux polynômes $r_{-1}(z) = z^{\delta-1}$ et $r_0(z) = S_b(z)\sigma'(z) \bmod z^{\delta-1}$, et on considère les trois suites de polynômes $(r_i(z))_i$, $(U_i(z))_i$ et $(V_i(z))_i$ définies dans* 4.2. *Si $2\nu + \rho < \delta$ alors :*

1. *Il existe un rang k tel que $\deg r_{k-1} \geq \delta - 1 - \nu$, et $\deg r_k < \delta - 1 - \nu$*

2. *les polynômes localisateur d'erreur et évaluateur de* **e** *sont donnés par :*

$$\sigma(z) = \frac{U_k(z)}{U_k(0)} \quad \text{et} \quad \Omega_b(z) = \frac{r_k(z)}{U_k(0)}.$$

preuve : cf. preuve du théorème I.11. □

Ce théorème montre donc que l'on peut utiliser l'algorithme d'Euclide pour corriger ν erreurs et ρ effacements à condition que

$$2\nu + \rho < \delta$$

où δ est la distance construite du code BCH considéré.

Bien entendu on retrouve le résultat de la section 4.3 pour $\rho = 0$.

Proposition I.21 *L'algorithme d'Euclide étendu pour corriger simultanement des erreurs et des effacements dans un code BCH de distance construite δ est borné strictement par δ et respecte la distance de Hamming généralisée.*

Remarque I.11 Les resultats de cette section ainsi que ceux de la section précédente donnent un algorithme de décodage borné par la distance construite. Lorsque la distance minimale est strictement supérieure à la distance construite, on ne dispose plus de l'algorithme d'Euclide si l'on veut un algorithme borné par la distance minimale.

5 Outils pour l'évaluation d'algorithmes de décodage

5.1 Polynôme des motifs d'erreur corrigibles

Définition I.30 *On appellera énumérateur des poids d'un sous-ensemble E de $\mathbb{F}_q^n$ le polynôme de $\mathbb{Z}[x]$*

$$\sum_{m \in E} x^{w_H(m)}.$$

Définition I.31 *Un* motif d'erreur *d'un code C de longueur n linéaire sur $\mathbb{F}_q$ possédant un algorithme de décodage d'erreur γ, est un élément de $\mathbb{F}_q^n$.*

Un motif d'erreur m de C sera dit corrigible par γ si

$$\forall\, c \in C,\ \ \gamma(c+m) = c.$$

Lorsque γ est linéaire, le motif d'erreur m est corrigible ssi $\gamma(m) = 0$. Dans la suite de cette section nous ne nous interesserons qu'aux algorithmes linéaires. Dans ce cas, l'ensemble des motifs d'erreur corrigibles par γ est $\gamma^{-1}(0)$.

Définition I.32 *Soient C un code de longueur n linéaire sur $\mathbb{F}_q$ et γ un algorithme de décodage linéaire de C. Le* polynôme des motifs corrigibles *de γ est l'énumérateur des poids $P_0(x)$ de l'ensemble des motifs d'erreur corrigibles par γ, c'est-à-dire*

$$P_0(x) = \sum_{m \in \gamma^{-1}(0)} x^{w_H(m)} = \sum_{i=0}^{n} a_{0,i} x^i$$

où

$$a_{0,i} = |\{m \in \mathbb{F}_q^n \mid w_H(m) = i,\ \gamma(m) = 0\}|$$

est le nombre de motifs d'erreur de poids i corrigibles par γ.

De la même manière nous définissons un polynôme décrivant les motifs non corrigibles et un polynôme des motifs corrigés de façon erronée.

Définition I.33 *Soit C un code de longueur n linéaire sur $\mathbb{F}_q$, soit γ un algorithme de décodage de C. Le* polynôme des motifs non corrigibles *de γ est un polynôme $P_1(x) \in \mathbb{Z}[x]$*

$$P_1(x) = \sum_{m \in \gamma^{-1}(\infty)} x^{w_H(m)} = \sum_{i=0}^{n} a_{1,i} x^i$$

où

$$a_{1,i} = |\{m \in \mathbb{F}_q^n \mid w_H(m) = i,\ \gamma(m) = \infty\}|$$

est le nombre de motifs d'erreur de poids i pour lesquels γ échoue.

Le polynôme des motifs erronés *de γ est un polynôme $P_2(x) \in \mathbb{Z}[x]$*

$$P_2(x) = \sum_{m \in \gamma^{-1}(C \setminus \{0\})} x^{w_H(m)} = \sum_{i=0}^{n} a_{2,i} x^i$$

où

$$a_{2,i} = |\{m \in \mathbb{F}_q^n \mid w_H(m) = i,\ \gamma(m) \in C \setminus \{0\}\}|$$

est le nombre de motifs d'erreur de poids i pour lesquels γ fournit un mot de C erroné.

Bien entendu, on a

$$P_0(x) + P_1(x) + P_2(x) = \sum_{i=0}^{n} \binom{n}{i} (q-1)^i x^i = (1 + (q-1)x)^n,$$

c'est-à-dire que ces trois polynômes prennent en compte tous les motifs d'erreur possibles.

Les deux propositions suivantes sont évidentes.

Proposition I.22 *Si l'algorithme γ est borné strictement par e, on a*

$$P_0(x) = \sum_{i=0}^{\frac{e-1}{2}} \binom{n}{i} (q-1)^i x^i.$$

Proposition I.23 *Si l'algorithme γ est à vraisemblance maximale, alors*

$$P_1(x) = 0.$$

5.1.1 Codes possédant un algorithme de décodage borné strictement

Soit un code $C(n,k,d)$ linéaire sur $\mathbb{F}_q$, muni d'un algorithme de décodage borné strictement par e, on pose

$$t = \lfloor \frac{e-1}{2} \rfloor.$$

On a dans ce cas d'après la proposition I.22

$$P_0(x) = \sum_{i=0}^{t} \binom{n}{i} (q-1)^i x^i,$$

et on peut calculer également les valeurs de $P_1(x)$ et $P_2(x)$ si on connait la distribution des poids du code.

Un motif d'erreur sera corrigé de façon erronée si et seulement si il se trouve à une distance inférieure à t d'un mot non nul du code.

Proposition I.24 *Le polynôme énumérateur des poids d'une boule de rayon r autour d'un mot de poids w de $\mathbb{F}_q^n$ est égal à*

$$\sum_{0 \leq i+j \leq r} \binom{w}{i} (1+(q-2)x)^i x^{w-i} \binom{n-w}{j} (q-1)^j x^j. \tag{I.13}$$

preuve : Soit $m \in \mathbb{F}_q^n$ un mot de poids w. Soient $\mathrm{supp}(m)$ le support de m, et $\overline{\mathrm{supp}(m)}$ son complémentaire. On a

- $\binom{w}{i}(1+(q-2)x)^i x^{w-i}$ est l'énumérateur des poids de la restriction à $\mathrm{supp}(m)$ des mots de la forme $m+u$ tels que $w_H(u) = i$ et $\mathrm{supp}(u) \subset \mathrm{supp}(m)$,
- $\binom{n-w}{j}(q-1)^j x^j$ est l'énumérateur des poids de la restriction à $\overline{\mathrm{supp}(m)}$ des mots de la forme $m+u$ tels que $w_H(u) = j$ et $\mathrm{supp}(u) \cap \mathrm{supp}(m) = \emptyset$.

Donc l'ensemble des mots à distance exactement r de m a pour énumérateur des poids

$$\sum_{i+j=r} \binom{w}{i} (1+(q-2)x)^i x^{w-i} \binom{n-w}{j} (q-1)^j x^j.$$

On en déduit (I.13). □

On note $f_{w,t}(x)$ l'énumérateur des poids d'une boule de rayon t centrée en un mot de poids w, on a

$$f_{w,t}(x) = \sum_{0 \leq i+j \leq t} \binom{w}{i} \binom{n-w}{j} (q-1)^j (1+(q-2)x)^i x^{w-i+j}$$

Proposition I.25 *Le polynôme des motifs erronés de γ, est égal à*

$$P_2(x) = \sum_{w=d}^{n} A_w \, f_{w,t}(x),$$

où A_w est le nombre de mots de poids w dans C.

preuve : Les boules de rayons t centrées en tous les éléments du code sont deux à deux disjointes car $t = \lfloor \frac{e-1}{2} \rfloor \leq \lfloor \frac{d-1}{2} \rfloor$, pour obtenir l'enumérateur des poids des motifs incorrectement corrigés par γ, il suffit donc d'additionner les énumérateurs des poids des boules de rayons t centrées en tous les éléments non nuls du code

$$P_2(x) = \sum_{m \in C \setminus \{0\}} f_{w_H(m),t}(x) = \sum_{w=d}^{n} A_w \, f_{w,t}(x).$$

□

On peut à partir de cette proposition en déduire le polynôme des motifs non corrigibles

$$P_1(x) = \sum_{i=t+1}^{n} \binom{n}{i} (q-1)^i x^i - P_2(x).$$

5.1.2 Codes MDS

Pour les codes MDS on connait la distribution des poids, donc si un tel code possède un algorithme de décodage borné strictement par sa distance minimale, on connait tous ses polynômes de description de motifs d'erreur.

Pour un tel code $C(n, k, d)$, on a

$$A_w = \binom{n}{w}(q-1)\sum_{i=0}^{w-d}(-1)^i\binom{w-1}{i}q^{w-d-i}.$$

Exemple : Prenons Le code de Reed-Solomon $RS(15, 7, 9)$ sur $\mathbb{F}_{16}$ muni de l'algorithme d'Euclide étendu qui est bien borné strictement par la distance minimale, le calcul nous donne

$$P_0(x) = 1 + 225\,x + 23625\,x^2 + 1535625\,x^3 + 69103125\,x^4$$

$$\begin{aligned} P_1(x) &= 2270943675\,x^5 + 56407826475\,x^6 + 1082821986675\,x^7 + 16217629824825\,x^8 \\ &\quad + 189224739979275\,x^9 + 1703273910373035\,x^{10} + 11612517003134775\,x^{11} \\ &\quad + 58063467021999075\,x^{12} + 200988339181361550\,x^{13} \\ &\quad + 430689502969894350\,x^{14} + 430689473127942210\,x^{15} \end{aligned}$$

$$\begin{aligned} P_2(x) &= 9459450\,x^5 + 602251650\,x^6 + 16658091450\,x^7 + 274571347050\,x^8 \\ &\quad + 3184273692600\,x^9 + 28407212673840\,x^{10} + 194399744912100\,x^{11} \\ &\quad + 971116718235300\,x^{12} + 3362142996372825\,x^{13} \\ &\quad + 7204387410965025\,x^{14} + 7204417252917165\,x^{15} \end{aligned}$$

Ce calcul a été effectué en Maple sur une station de travail SUN Sparcstation2 et a demandé environ 1 seconde de temps de calcul.

5.2 Polynôme des motifs d'erreur et d'effacement corrigibles

Nous pouvons définir pour un code possédant un algorithme de décodage d'erreur et d'effacement, les mêmes polynômes que précédemment, en utilisant la distance de Hamming généralisée.

Dans cette section, on appellera poids d'un mot $m \in (\mathbb{F}_q \cup \{\infty\})^n$ son poids de Hamming généralisé, on appelera support de m, et on notera $\mathrm{supp}(m)$, l'ensemble des positions non nulles de m.

Ici, nous étendons la définition de l'énumérateur des poids au poids de Hamming généralisé.

Définition I.34 *Soit Ω_H le poids de Hamming généralisé. On appellera énumérateur des poids d'un sous-ensemble E de $(\mathbb{F}_q \cup \{\infty\})^n$, le polynôme de $\mathbb{Z}[x^{\frac{1}{2}}]$*

$$\sum_{m \in E} x^{\Omega_H(m)}.$$

Définition I.35 *Un* motif d'erreur et d'effacement *d'un code C de longueur n linéaire sur $\mathbb{F}_q$ possédant un algorithme de décodage d'erreur et d'effacement γ, est un élément de $(\mathbb{F}_q \cup \{\infty\})^n$.*

Un motif d'erreur et d'effacement m de C sera dit corrigible par γ si

$$\forall\, c \in C,\ \ \gamma(c+m) = c.$$

Lorsque γ est linéaire, le motif d'erreur et d'effacement e est corrigible ssi $\gamma(e) = 0$. Dans la suite de cette section on ne s'interessera qu'aux algorithmes linéaires.

Définition I.36 *Soient C un code de longueur n linéaire sur $\mathbb{F}_q$ et γ un algorithme de décodage d'erreur et d'effacement linéaire de C.*

Le polynôme des motifs corrigibles *de γ est l'énumérateur des poids $P_0(x)$ de l'ensemble des motifs d'erreur et d'effacement corrigibles par γ, c'est-à-dire*

$$P_0(x) = \sum_{m \in \gamma^{-1}(0)} x^{\Omega_H(m)} = \sum_{0 \leq i \leq n,\ i \in \frac{1}{2}\mathbb{Z}} a_{0,i} x^i$$

où

$$a_{0,i} = |\{m \in (\mathbb{F}_q \cup \{\infty\})^n \mid \Omega_H(m) = i,\ \gamma(m) = 0\}|$$

est le nombre de motifs d'erreur et d'effacement de poids i corrigibles par γ.

De la même manière, on peut définir un polynôme décrivant les motifs non corrigibles et un polynôme des motifs corrigés de façon erronée.

Définition I.37 *Soit C un code de longueur n linéaire sur $\mathbb{F}_q$, soit γ un algorithme de décodage d'erreur et d'effacement de C.*

Le polynôme des motifs non corrigibles *de γ est un polynôme $P_1(x) \in \mathbb{Z}[x^{\frac{1}{2}}]$*

$$P_1(x) = \sum_{m \in \gamma(\infty)} x^{\Omega_H(m)} = \sum_{0 \leq i \leq n,\ i \in \frac{1}{2}\mathbb{Z}} a_{1,i} x^i$$

où

$$a_{1,i} = |\{m \in \mathbb{F}_q^n \mid \Omega_H(m) = i,\ \gamma(m) = \infty\}|$$

est le nombre de motifs d'erreur et d'effacement de poids i pour lesquels γ échoue.

Le polynôme des motifs erroné de γ est un polynôme $P_2(x) \in \mathbb{Z}[x^{\frac{1}{2}}]$

$$P_2(x) = \sum_{m \in \gamma(C \setminus \{0\})} x^{\Omega_H(m)} = \sum_{0 \leq i \leq n,\ i \in \frac{1}{2}\mathbb{Z}} a_{2,i} x^i$$

où

$$a_{2,i} = |\{m \in \mathbb{F}_q^n \mid \Omega_H(m) = i,\ \gamma(m) \in C \setminus \{0\}\}|$$

est le nombre de motifs d'erreur et d'effacement de poids i pour lesquels γ fourni un mot de C erroné.

Bien entendu, on a

$$P_0(x) + P_1(x) + P_2(x) = (1 + (q-1)x + x^{\frac{1}{2}})^n,$$

c'est-à-dire que ces trois polynômes prennent en compte tous les motifs d'erreur et d'effacement possibles.

Proposition I.26 *Si l'algorithme γ est borné strictement par e, on a*

$$P_0(x) = \sum_{0 \leq i < \frac{e}{2},\ i \in \frac{1}{2}\mathbb{Z}} [x^i](1 + (q-1)x + x^{\frac{1}{2}})^n,$$

où $[x^i]P$ désigne le coefficient de x^i dans l'expression P.

5.2.1 Codes possédant un algorithme de décodage borné strictement

Soit un code $C(n,k,d)$ linéaire sur $\mathbb{F}_q$, muni d'un algorithme de décodage d'erreur et d'effacement borné strictement par e, on pose

$$t = \frac{e-1}{2} \in \frac{1}{2}\mathbb{Z}.$$

Comme dans le cas des algorithmes de décodage d'erreur, $P_0(x)$ est connu dans ce cas, et on peut obtenir à partir de la distribution des poids de C la valeur de $P_1(x)$ et $P_2(x)$.

Un motif d'erreur et d'effacement sera corrigé de façon erronée si et seulement si il se trouve à une distance de Hamming généralisée inférieure à t d'un mot non nul du code.

Proposition I.27 *Le polynôme énumérateur des poids d'une boule de rayon r autour d'un mot de poids de Hamming w de $\mathbb{F}_q^n$ est égal à*

$$\sum_{0 \leq k \leq r} [y^k] \left(\left(\sum_{i=0}^{n} \binom{w}{i} (1 + x^{\frac{1}{2}} y^{\frac{1}{2}} + (q-2)xy)^i x^{w-i} \right) \left(\sum_{j=0}^{n} \binom{n-w}{j} (x^{\frac{1}{2}} y^{\frac{1}{2}} + (q-1)xy)^j \right) \right). \tag{I.14}$$

preuve : Soit $m \in \mathbb{F}_q^n$ un mot de poids w. Soient $\mathrm{supp}(m)$ le support de m, et $\overline{\mathrm{supp}(m)}$ son complémentaire. On a

- $\binom{w}{i}(y + x^{\frac{1}{2}} y^{\frac{1}{2}} + (q-2)xy)^i x^{w-i}$ est :
  - pour $y = 1$, le polynôme énumérateur des poids de la restriction à $\mathrm{supp}(m)$ des mots $u + m$ tels que $|\mathrm{supp}(m)| = i$ et $\mathrm{supp}(u) \subset \mathrm{supp}(m)$,
  - et pour $x = 1$, le polynôme énumérateur des poids de la restriction à $\mathrm{supp}(m)$ des mots u tels que $|\mathrm{supp}(m)| = i$ et $\mathrm{supp}(u) \subset \mathrm{supp}(m)$,
- $\binom{n-w}{j}(x^{\frac{1}{2}} y^{\frac{1}{2}} + (q-1)xy)^j$ est
  - pour $y = 1$ l'énumérateur des poids de la restriction à $\overline{\mathrm{supp}(m)}$ des mots $u + m$ tels que $|\mathrm{supp}(u)| = j$ et $\mathrm{supp}(u) \cap \mathrm{supp}(m) = \emptyset$,

– pour $x = 1$ l'énumérateur des poids de la restriction à $\overline{\mathrm{supp}(m)}$ des mots u tels que $|\mathrm{supp}(u)| = j$ et $\mathrm{supp}(u) \cap \mathrm{supp}(m) = \emptyset$.

Donc l'ensemble des mots à distance de Hamming généralisée exactement r de c a pour énumérateur

$$[y^r]\left(\sum_{i=0}^{n}\binom{w}{i}(y + x^{\frac{1}{2}}y^{\frac{1}{2}} + (q-2)xy)^i x^{w-i}\right)\left(\sum_{j=0}^{n}\binom{n-w}{j}(x^{\frac{1}{2}}y^{\frac{1}{2}} + (q-1)xy)^j\right).$$

On en déduit (I.14). □

On note $g_{w,t}(x)$ l'énumérateur des poids d'une boule de rayon t centrée en un mot de poids w, on a

$$g_{w,t}(x) = [y^t]\left(\sum_{i=0}^{n}\binom{w}{i}(y + x^{\frac{1}{2}}y^{\frac{1}{2}} + (q-2)xy)^i x^{w-i}\right)\left(\sum_{j=0}^{n}\binom{n-w}{j}(x^{\frac{1}{2}}y^{\frac{1}{2}} + (q-1)xy)^j\right).$$

Proposition I.28 *Le polynôme des motifs erronés de γ, est égal à*

$$P_2(x) = \sum_{w=d}^{n} A_w\, g_{w,t}(x),$$

où A_w est le nombre de mots de poids w dans C.

preuve : Les boules de rayons t centrées en tous les éléments du code sont deux à deux disjointes car $t = \lfloor\frac{e-1}{2}\rfloor \leq \lfloor\frac{d-1}{2}\rfloor$. Pour obtenir l'enumérateur des poids des motifs incorrectement corrigés par γ, il suffit donc d'additionner les énumérateurs des poids des boules de rayons t centrées en tous les éléments non nuls du code

$$P_2(x) = \sum_{c\in C\setminus\{0\}} g_{\Omega_H(c),t}(x) = \sum_{w=d}^{n} A_w\, g_{w,t}(x).$$

□

On peut à partir de cette proposition en déduire le polynôme des motifs non corrigibles à l'aide de

$$P_0(x) + P_1(x) + P_2(x) = (1 + (q-1)x + x^{\frac{1}{2}})^n.$$

5.3 Capacité de correction d'un algorithme de décodage

Soit $C(n, k, d)$ un code linéaire sur $\mathbb{F}_q$ muni d'un algorithme de décodage linéaire. Soit $P_0(x)$ le polynôme des motifs corrigibles de cet algorithme.

Dans un canal q-aire symétrique de probabilité d'erreur totale p (la probabilité de transition d'un état à un autre vaut $\frac{p}{q-1}$), la probabilité de décodage correct est égale à

$$P_{cor}(p) = (1-p)^n P_0\left(\frac{p}{(q-1)(1-p)}\right) = \sum_{i=0}^{n} a_{0,i}\left(\frac{p}{q-1}\right)^i (1-p)^{n-i}$$

Définition I.38 *Soit $C(n, k, d)$ un code linéaire sur $\mathbb{F}_q$, muni d'un algorithme de décodage linéaire γ dont le polynôme des motifs corrigibles est $P_0(x)$.*

On pose

$$P_{0,t^*}(x) = \sum_{i=0}^{t^*} \binom{n}{i} (q-1)^i x^i.$$

La capacité de correction *du code C pour l'algorithme γ et pour une probabilité d'erreur p, est le plus grand entier $d^* = 2t^* + 1$ tel que*

$$(1-p)^n P_{0,t^*}(\frac{p}{(q-1)(1-p)}) \leq (1-p)^n P_0(\frac{p}{(q-1)(1-p)})$$

La capacité de correction de C est donc la distance minimale d^* d'un code de même longueur n sur $\mathbb{F}_q$, possédant un algorithme de décodage borné strictement par d^*, et ayant des performances de décodage voisines de celles de C pour une probabilité d'erreur par symbole donnée.

Il s'agit en quelque sorte d'une "distance minimale pratique".

Chapitre II

Distance minimale des codes BCH binaires

Dans ce chapitre nous dévellopons une partie d'un travail mené en commun avec Daniel Augot et Pascale Charpin (cf. [4], [5] et [6])

La première section nous permet de définir les identités de Newton pour un code cyclique donné C, c'est-à-dire une suite d'identités vérifiées par les fonctions symétriques élémentaires et les fonctions puissances symétriques des localisateurs d'un mot de C.

Dans la seconde section nous définissons les équations des Newton, dont l'écriture est identique à celle des identités de Newton, mais que nous considérons comme un système d'équations à résoudre. Nous établissons alors une équivalence entre la consistance de ce système et l'existence de mots de poids donné dans un code cyclique.

Enfin dans la section 3 nous donnons les principaux résultats obtenus à l'aide des théorèmes de la section 2. En particulier nous avons pu donner des distances minimales inconnues de codes BCH primitifs au sens strict de longueur 255 et 511.

Dans tout ce chapitre nous utiliserons les notations suivantes :

- $B(n, \delta)$ est le code BCH binaire primitif au sens strict de distance construite δ et de longueur $n = 2^m - 1$.
- $\mathbb{F}_{2^m}$ est le corps fini de cardinal 2^m et α est une racine n-ième primitive de l'unité dans $\mathbb{F}_{2^m}$.
- A tout vecteur $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}_{2^m}^n$ on identifie le polynôme $a(z) = a_0 + a_1 z + \ldots + a_{n-1} z^{n-1} \in \mathbb{F}_{2^m}[z]$.

1 Les identités de Newton

1.1 Définitions

Définition II.1 *Soit un entier s quelconque, on appelle* classe cyclotomique de s modulo n, *et on note* cl(s), *l'ensemble*

$$cl(s) = \{s, 2s, 2^2 s \ldots, 2^{m-1} s \bmod n\}.$$

Notons que $\forall\, s \in \mathbb{Z}$, $\forall\, i \in \mathrm{cl}(s)$, on a $\mathrm{cl}(i) = \mathrm{cl}(s)$.

Définition II.2 *Le* polynôme de Mattson-Solomon *associé à* $\mathbf{a} \in \mathbb{F}_{2^m}^n$, *est le polynôme* $A(z) \in \mathbb{F}_{2^m}[z]$

$$A(z) = \sum_{i=1}^{n} A_i z^{n-i} \tag{II.1}$$

où

$$A_i = a(\alpha^i) = \sum_{j=0}^{n-1} a_j \alpha^{ij}$$

Notons que l'on peut étendre la définition des coefficients A_i du polynôme de Mattson-Solomon pour tout i dans $\mathbb{Z}$.

Théorème II.1 *Soit $A(z)$ le polynôme de Mattson-Solomon associé à $\mathbf{a} \in \mathbb{F}_{2^m}^n$, on a la formule d'inversion*

$$a(z) = \sum_{i=0}^{n-1} A(\alpha^i) z^i.$$

preuve : On a

$$A(\alpha^i) = \sum_{j=1}^{n} A_j \alpha^{i(n-j)} = \sum_{j=1}^{n} A_j \alpha^{-ij} = \sum_{j=1}^{n} (\sum_{l=0}^{n-1} a_l \alpha^{lj}) \alpha^{-ij} = \sum_{l=0}^{n-1} a_l (\sum_{j=1}^{n} \alpha^{(l-i)j}),$$

or $\forall\,\beta \in \mathbb{F}_{2^m}$,

$$(\beta - 1)(\sum_{j=1}^{n} \beta^j) = \beta^{n+1} - \beta = 0.$$

Donc

$$l \neq i \Rightarrow \alpha^{l-i} - 1 \neq 0 \Rightarrow \sum_{j=1}^{n} \alpha^{(l-i)j} = 0,$$

et enfin

$$A(\alpha^i) = na_i = a_i$$

car $n = 2^m - 1 = 1 \bmod 2$. □

Proposition II.1 *Soit* $\mathbf{a} \in \mathbb{F}_2^n$ *un vecteur binaire. Soit* A_i *le* i*-ième coefficient du polynôme de Mattson-Solomon associé à* $\mathbf{a}$*. Pour tout* $i \in \mathbb{Z}$*, on a*

1. $A_{i+n} = A_i$

2. $A_{2i \bmod n} = A_i^2$

preuve :

1. $A_{i+n} = a(\alpha^{i+n}) = a(\alpha^i) = A_i$

2. $A_{2i \bmod n} = A_{2i} = a(\alpha^{i^2}) = a(\alpha^i)^2$, car $\mathbb{F}_{2^m}$ est de caractéristique 2, les coefficients de $\mathbf{a}$ sont dans $\mathbb{F}_2$.

□

C'est-à-dire qu'il y a exactement un A_i significatif par classe cyclotomique,

$$\forall\,j \in \mathrm{cl}(i),\ j = 2^l i \bmod n,\ \text{donc } A_j = A_i^{2^l},$$

en particulier, on a

$$A_i = 0 \Leftrightarrow \forall\,j \in \mathrm{cl}(i),\ A_j = 0. \qquad \text{(II.2)}$$

On également le corollaire suivant

Corollaire II.1 *Soit* $\mathbf{a} \in \mathbb{F}_2^n$ *un vecteur binaire. Soit* A_i *le* i*-ième coefficient du polynôme de Mattson-Solomon associé à* $\mathbf{a}$*. Pour tout* $i \in \mathbb{Z}$*, l'ensemble*

$$\{A_j,\ j \in \mathrm{cl}(i)\}$$

est exactement l'ensemble des conjugés de A_i *dans* $\mathbb{F}_{2^m}$*.*

Définition II.3 *Soient* $X_1, X_2, \ldots, X_w$*,* w *élément distincts de* $\mathbb{F}_{2^m}$*, pour tout* $k \in \mathbb{Z}$*, la* k*-ième* fonction puissance symétrique *des* $(X_i)_{1 \leq i \leq w}$*, notée* A_k *est définie par*

$$A_k = \sum_{i=1}^{w} X_i^k.$$

Notons tout de suite que cette définition n'entraine aucun conflit de notation avec les coefficients du polynôme de Mattson-Solomon, comme le montre la proposition suivante.

Proposition II.2 *Si* $\mathbf{a} \in \mathbb{F}_2^m$ *est un mot de localisateurs* $X_1, X_2, \ldots, X_w$*, alors pour tout* $k \in \mathbb{Z}$*, le* k*-ième coefficient du polynôme de Mattson-Solomon de* $\mathbf{a}$ *est égal à la* k*-ième fonction puissance symétrique des* $(X_i)_{1 \leq i \leq w}$*.*

preuve : Soient

$$X_1 = \alpha^{i_1}, X_2 = \alpha^{i_2}, \ldots, X_w = \alpha^{i_w},$$

les localisateurs de $\mathbf{a}$. Puisque $\mathbf{a} \in \mathbb{F}_2^m$, $a_i = 1$ pour $i = i_1, \ldots, i_w$, et vaut 0 sinon.

Le k-ième coefficient du polynôme de Mattson-Solomon est égal à

$$a(\alpha^k) = \sum_{i=0}^{n-1} a_i(\alpha^k)^i = \sum_{j=1}^{w} (\alpha^k)^{i_j} = \sum_{j=1}^{w} (\alpha^{i_j})^k = \sum_{j=1}^{w} X_j^k,$$

qui est bien la k-ième fonction puissance symétrique des X_i. □

Définition II.4 *Soient* $X_1, X_2, \ldots, X_w$*,* w *élément distincts de* $\mathbb{F}_{2^m}$*. Pour tout* k*,* $0 \leq k \leq w$*, la* k*-ième* fonction symétrique élémentaire *des* $(X_i)_{1 \leq i \leq w}$*, notée* σ_k *est définie par*

$$\sigma_k = \sum_{\substack{I \in \mathcal{P}([1, w]) \\ |I| = k}} (-1)^k \prod_{i \in I} X_i.$$

Proposition II.3 (identités de Newton) *Soient* $X_1, X_2, \ldots, X_w$*,* w *élément distincts de* $\mathbb{F}_{2^m}$*,* σ_i *les fonctions symétriques élémentaires des* X_i*,* A_i *les fonctions puissances symétriques des* X_i*, on a les relations suivantes*

$$\begin{array}{rll} 0 < r \leq w, & I_r : & A_r + \sum_{1 \leq i < r} A_{r-i}\sigma_i + r\sigma_r = 0 \\ r > w, & I_r : & A_r + \sum_{1 \leq i \leq w} A_{r-i}\sigma_i = 0 \end{array} \tag{II.3}$$

appelées identités de Newton.

preuve : Soit $\mathbf{a}$ le mot de $\mathbb{F}_2^n$ admettant les X_i, $1 \leq i \leq w$ comme localisateurs. Alors $\sigma(z)$ le polynôme localisateur de $\mathbf{a}$ vérifie

$$\sigma(z) = \prod_{i=1}^{w} (1 - X_i z) = \sum_{i=0}^{w} \sigma_i z^i$$

où les σ_i sont les fonctions symétriques élémentaires des X_i

$$\left\{ \begin{array}{l} \sigma_1 = -(X_1 + \ldots + X_w), \\ \sigma_2 = X_1X_2 + X_1X_3 + \ldots + X_{w-1}X_w, \\ \vdots \\ \sigma_w = (-1)^w X_1 \ldots X_w. \end{array} \right.$$

Un tel mot $\mathbf{a}$ est unique car ses coordonnées sont dans $\mathbb{F}_2$. D'après le théorème I.8 page 28, pour $b = 1$, on a

$$\omega(z) = S(z)\sigma(z) \tag{II.4}$$

où

$$\omega(z) = \sum_{i=1}^{w} X_i \prod_{j \neq i}(1 - zX_j), \quad S(z) = \sum_{i=1}^{\infty} A_i z^{i-1}.$$

Remarquons que $\omega(z)$ est l'opposé de la dérivée de $\sigma(z)$, donc

$$\omega(z) + \sum_{i=1}^{w} i\sigma_i z^{i-1} = 0. \tag{II.5}$$

Le produit $S(z)\sigma(z)$ vaut

$$S(z)\sigma(z) = \sum_{i=1}^{\infty} A_i z^{i-1} \sum_{i=0}^{w} \sigma_i z^i = \sum_{r=1}^{\infty}(\sum_{i=0}^{r-1} A_{r-i}\sigma_i) z^{r-1},$$

donc en utilisant (II.4) et (II.5) on a

$$\sum_{r=1}^{w} r\sigma_r z^{r-1} + \sum_{r=1}^{\infty}(\sum_{i=0}^{r-1} A_{r-i}\sigma_i) z^{r-1} = 0. \tag{II.6}$$

La série donnée par le terme de gauche de l'égalité (II.6) étant identiquement nulle, on en déduit pour $r \leq w$

$$\sum_{i=0}^{r-1} A_{r-i}\sigma_i + r\sigma_r = 0$$

donc, puisque $\sigma_0 = 1$,

$$A_r + \sum_{i=1}^{r-1} A_{r-i}\sigma_i + r\sigma_r = 0$$

et pour $r > w$

$$\sum_{i=0}^{r-1} A_{r-i}\sigma_i = 0$$

donc, puisque $\sigma_0 = 1$, et $\sigma_i = 0$ pour $i > w$

$$A_r + \sum_{i=1}^{w} A_{r-i}\sigma_i = 0.$$

□

1.2 Les identités de Newton pour un code BCH

Proposition II.4 *Soit $\mathbf{a} \in \mathbb{F}_{2^m}^n$, de poids w, et de localisateurs $X_1, X_2, \ldots, X_w$. Pour tout $k \in \mathbb{Z}$, on note A_k la k-ième fonction puissance symétrique des X_i.*

Soit C un code cyclique binaire de longueur n, d'ensemble de définition $I(C)$. Alors on a

$$\mathbf{a} \in C \Leftrightarrow \forall\, k \in I(C),\ A_k = 0 \tag{II.7}$$

preuve : D'après la proposition II.2, pour tout k, A_k est le k-ième coefficient du polynôme de Mattson-Solomon associé à $\mathbf{a}$, donc $A_k = a(\alpha^k)$, or on a

$$\mathbf{a} \in C \Leftrightarrow \forall\, k \in I(C),\ a(\alpha^k) = 0$$

par définition de $I(C)$, d'où le résultat. □

Remarquons que les équations (II.2) et (II.7) prouvent que l'ensemble de définition d'un code binaire de longueur n est une réunion de classes cyclotomiques modulo n. En particulier, cela nous permet de décrire $B(n, \delta)$ comme le code cyclique dont l'ensemble de définition est la réunion des classes cyclotomiques

$$I(B(n,\delta)) = \mathrm{cl}(1) \cup \mathrm{cl}(2) \cup \ldots \cup \mathrm{cl}(\delta - 1).$$

Théorème II.2 *Soit $B(n, \delta)$ le code BCH primitif au sens strict de longueur n et de distance construite δ, soit $\mathbf{a} \in \mathbb{F}_2^n$ de poids $w \geq \delta$ et soit $\sigma(z) = 1 + \sum_{i=1}^{w} \sigma_i z^i$ son polynôme localisateur. On a*

$$\mathbf{a} \in B(n,\delta) \Leftrightarrow \forall\, i \text{ impair},\ 0 < i < \delta,\ \sigma_i = 0.$$

preuve : A_i et σ_i, les fonctions puissances élémentaires et les fonctions symétriques élémentaires des localisateurs de $\mathbf{a}$, vérifient les équations de Newton.

Si $\mathbf{a} \in B(n, \delta)$, pour tout $i < \delta$, on a

$$I_i \ :\ i\sigma_i = 0,$$

donc en caractéristique 2 on a $\sigma_i = 0$ lorsque i est impair.

Réciproquement on suppose $\sigma_i = 0$, pour $i < \delta$ impair, montrons par récurrence que $A_i = 0$, pour tout $i < \delta$. On a

$$I_1 \ :\ A_1 + \sigma_1 = 0,$$

donc puisque $\sigma_1 = 0$, $A_1 = 0$.

Supposons que $\forall\, j < i < \delta$, $A_j = 0$, puisque $\delta \leq w$, la i-ième equation de Newton s'écrit

$$I_i \ :\ A_i + \sum_{j=1}^{i-1} \sigma_j A_{i-j} + i\sigma_i = 0,$$

donc en utilisant l'hypothèse de récurrence

$$I_i \ :\ A_i + i\sigma_i = 0.$$

Si i est pair, alors $i = 0$ en caractéristique 2, sinon $\sigma_i = 0$, dans tous les cas $A_i = 0$, donc $\mathbf{a} \in B(n, \delta)$. □

Proposition II.5 *Si $\mathbf{a} \in B(n, \delta)$ est de poids δ, alors $A_\delta \neq 0$, et de plus si $\mathrm{pgcd}(\delta, n) = 1$, il existe une permutation cyclique de $\mathbf{a}$ dont la δ-ième fonction puissance symétrique est égale à 1.*

preuve : Si $\mathbf{a} \in B(n, \delta)$ est de poids δ, on a $A_\delta \neq 0$, sinon $\mathbf{a}$ appartiendrait au code dont l'ensemble de définition est $I(B(n, \delta)) \cup \mathrm{cl}(\delta)$, code qui a une borne BCH strictement supérieure à δ, ce qui contredit $w_H(\mathbf{a}) = \delta$.

Soient $X_1, \ldots, X_\delta$ les localisateurs de $\mathbf{a}$,

$$A_\delta = \sum_{i=1}^{\delta} X_i^\delta = \alpha^j,$$

permuter circulairement $\mathbf{a}$ revient exactement à multiplier les localisateurs par une puissance de α.

Si $\mathrm{pgcd}(\delta, n) = 1$, alors d'après Bezout,

$$\exists\, l \in \mathbb{Z},\ l\,\delta \equiv -j \bmod n,$$

donc le mot $\tilde{\mathbf{a}}$ de localisateurs $\alpha^l X_1, \ldots, \alpha^l X_w$ a pour δ-ième fonction puissance élémentaire

$$\tilde{A}_\delta = \sum_{i=1}^{w} (\alpha^l X_i)^\delta = \alpha^{-j} \sum_{i=1}^{w} X_i^\delta = 1$$

et puisque $B(n, \delta)$ est cyclique, $\tilde{\mathbf{a}} \in B(n, \delta)$. □

2 Les équations de Newton

Définition II.5 *Soit un entier w. Pour tout r, on considère les équations eq_r d'indeterminées $(A_i)_{i>0}$, et $(\sigma_i)_{1 \le i \le w}$ sur $\mathbb{F}_{2^m}$, définies par*

$$\begin{array}{rl} 0 < r \le w, & eq_r \ :\ A_r + \sum_{1 \le i < r} A_{r-i}\sigma_i + r\sigma_r = 0 \\ r > w, & eq_r \ :\ A_r + \sum_{1 \le i \le w} A_{r-i}\sigma_i = 0 \end{array} \tag{II.8}$$

Lorsque les A_i et les σ_i sont respectivement les fonctions puissances élémentaires et les fonctions symétriques élémentaires de w éléments $X_1, \ldots, X_w$ de $\mathbb{F}_{2^m}$, alors eq_r est la r-ième identités de Newton notée I_r dans la section précédente.

Lemme II.1 *Soit $(\sigma_i)_{0 < i \le w}$ une suite d'éléments de $\mathbb{F}_{2^m}$. Pour tout rang $r > w$, il existe une unique famille $(A_i)_{0 < i < r}$ vérifiant les équations de Newton $(eq_i)_{0 < i < r}$.*

preuve : Les w premières équation de Newton s'écrivent

$$\begin{pmatrix} 1 & 0 & \ldots & \ldots & 0 \\ \sigma_1 & 1 & \ddots & & \vdots \\ \sigma_2 & \sigma_1 & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ \sigma_{w-1} & \sigma_{w-2} & \sigma_{w-3} & \ldots & 1 \end{pmatrix} \begin{pmatrix} A_1 \\ A_2 \\ A_3 \\ \vdots \\ A_w \end{pmatrix} = \begin{pmatrix} \sigma_1 \\ 2\sigma_2 \\ 3\sigma_3 \\ \vdots \\ w\sigma_w \end{pmatrix} \tag{II.9}$$

il s'agit d'un système de Cramer donnant $(A_i)_{0<i\leq w}$ en fonction de $(\sigma_i)_{0<i\leq w}$. Et pour tout i, $w < i < r$ on a

$$eq_i\,:\,A_i + \sum_{1\leq j\leq w} A_{i-j}\sigma_j = 0$$

qui nous donne une définition par récurrence des A_i restants.

Donc lorsque les $(\sigma_i)_{0<i\leq w}$ sont fixé, il existe une unique famille $(A_i)_{0<i<r}$ vérifiant les équations de Newton $(eq_i)_{0<i<r}$. □

Proposition II.6 *Soit un entier $r > w$, soient $(\sigma_i)_{0<i\leq w}$ et $(A_i)_{0<i<r}$ vérifiant les équations de Newton $(eq_i)_{0<i<r}$. Soit le polynôme $\sigma(z) = 1 + \sum_{1\leq i\leq w} \sigma_i z^i$. Si $\sigma(z)$ est scindé dans $\mathbb{F}_{2^m}$, et s'écrit*

$$\sigma(z) = \prod_{j=1}^{w}(1 - X_j z),\ \text{avec}\ \forall\, j,\ 1 \leq j \leq w,\ X_j \in \mathbb{F}_{2^m},$$

alors les éléments A_i solutions des équations $(eq_i)_{0<i\leq r}$ sont les fonctions puissances symétriques des X_i, c'est-à-dire

$$\forall\, i,\ 1 < i < r,\ A_i = \sum_{j=1}^{w} X_j^i.$$

preuve : Il est clair que les fonctions symétriques élémentaires et les fonctions puissances symétriques des X_i sont solution des équations de Newton. L'unicité de la solution des équations $(eq)_{0<i<r}$, lorsque les σ_i sont donnés (lemme II.1) nous assure donc le résultat. □

2.1 Les équations de Newton pour un code cyclique

Lemme II.2 *Soit le système d'indéterminées $(\sigma_i)_{0<i\leq w}$ et $(A_i)_{0<i<n+w}$ sur $\mathbb{F}_{2^m}$,*

$$(S_w(C))\quad \begin{cases} 0 < r \leq w,\ eq_r\ :\ A_r + \sum_{1\leq i<r} A_{r-i}\sigma_i + r\sigma_r = 0 \\ w < r < n+w,\ eq_r\ :\ A_r + \sum_{1\leq i\leq w} A_{r-i}\sigma_i = 0 \\ \forall\, i \in I(C),\ A_i = 0 \\ \sigma(z)\mid z^n - 1 \end{cases} \tag{II.10}$$

où $\sigma(z) = 1 + \sum_{1\leq i\leq w} \sigma_i z^i$.

1. *Si $(\sigma_i)_{0<i\leq w}$ et $(A_i)_{0<i<n+w}$ sont une solution de $(S_w(C))$ alors les σ_i sont les fonctions symétriques élémentaires et les A_i les fonctions puissances symétriques d'un mot $\mathbf{a} \in C$ de poids w.*

2. *Si $\mathbf{a}$ est un mot de C de poids w, alors les fonctions symétriques élémentaires et les fonctions puissances symétriques des localisateurs de $\mathbf{a}$ sont solution de $(S_w(C))$.*

preuve :

1. Soit $(\sigma_i)_{0<i\leq w}$ et $(A_i)_{0<i<n+w}$ une solution de $(S_w(C))$.

- On a $n = 2^m - 1$, donc puisque $\sigma(z)$ divise $z^n - 1$, et $\sigma(0) = 1$, $\sigma(z)$ peut être considéré comme le polynôme localisateur d'un certain mot $\mathbf{a} \in \mathbb{F}_2^n$ de poids w.
- D'après la proposition II.6, les A_i sont alors les fonctions puissances symétriques des localisateurs de $\mathbf{a}$.
- Enfin puisque pour tout i dans $I(C)$, $A_i = 0$, on a bien $\mathbf{a} \in C$ d'après la proposition II.4.

2. Supposons qu'il existe un mot $\mathbf{a}$ de poids w dans C, soient $X_1, \dots, X_w$ les localisateurs de $\mathbf{a}$.

   - Les fonctions puissances symétriques A_i, et les fonctions symétriques élémentaires σ_i des X_i sont solution des équations de Newton $(eq_r)_{0<r<n+w}$ d'après la proposition II.3.
   - $\mathbf{a} \in C$, donc $A_i = 0$, pour $i \in I(C)$.
   - Enfin $\sigma(z)$, en tant que polynôme localisateur de $\mathbf{a}$, divise $z^n - 1$, car $n = 2^m - 1$.

   Les σ_i et les A_i sont donc solution du système $(S_w(C))$.

□

Théorème II.3 *Soient C un code cyclique binaire de longueur n, $I(C)$ son ensemble de définition, soit w un entier positif et $(S_w(C))$ le système défini par* (II.10).

Le code C admet des mots de poids w si et seulement si le système $(S_w(C))$ admet des solutions.

preuve : D'après le lemme II.2, toute solution de $(S_w(C))$ nous donne un mot de C de poids w, et réciproquement tout mot de C de poids w donne une solution de $(S_w(C))$. □

L'existence de mots de poids donné dans un code cyclique est donc liée à la consistance d'un système d'équations polynomiales sur $\mathbb{F}_{2^m}$. Le corollaire suivant nous sera utile pour démontrer l'absence de mots d'un poids donné dans un code.

Corollaire II.2 *S'il n'existe pas de solution au système*

$$\begin{cases} eq_r, \ 0 < r < n + w \\ A_i = 0, \ i \in I(C), \end{cases}$$

alors il n'existe pas de mots de poids w dans C.

Proposition II.7 *Toute solution $(\sigma_i)_{0<i\leq w}$ et $(A_i)_{0<i<n+w}$ de $(S_w(C))$ vérifie les propriétés*

- $\forall\, i,\ 0 < i < w,\ A_{n+i} = A_i,$
- $A_n = w,$
- $\forall\, i,\ 0 < i < n,\ A_{2i \bmod n} = A_i^2.$

preuve : D'après la proposition II.6, toute solution de $(S_w(C))$ est telle les A_i soient les fonctions puissances symétriques de w éléments distincts de $\mathbb{F}_{2^m}$, les propriétés énoncées s'en déduisent immédiatement. □

Corollaire II.3 *Soit J un ensemble de représentants des classes cyclotomiques modulo n sur $\mathbb{F}_2$, soit i un entier et soit $j \in J$ un représentant de la classe de i. Il existe $s \in \mathbb{Z}$ tel que $i = 2^s j$ et donc*

$$A_i = A_j^{2^s}.$$

Donc le système $(S_w(C))$ est équivalent à un système dont les seules indéterminées sont $(\sigma_i)_{1\leq i\leq w}$ et $(A_i)_{i\in J'}$, où J' est un ensemble de représentants des classes cyclotomiques qui ne sont pas dans $I(C)$.

2.2 Le cas des codes BCH primitifs au sens strict

On se place dans le cas où $C = B(n,\delta)$, le code BCH binaire primitif au sens strict de longueur n et de distance construite δ, soit $I(B(n,\delta))$ son ensemble de définition, on a

$$I(B(n,\delta)) = \mathrm{cl}(1) \cup \mathrm{cl}(2) \ldots \cup \mathrm{cl}(\delta - 1),\ \ \delta \notin I(B(n,\delta)).$$

On notera (S_δ) le système $(S_\delta(B(n,\delta)))$,

$$(S_\delta)\quad \begin{cases} 0 < r \leq \delta,\ eq_r\ :\ A_r + \sum_{1\leq i<r} A_{r-i}\sigma_i + r\sigma_r = 0 \\ \delta < r < n+\delta,\ eq_r\ :\ A_r + \sum_{1\leq i\leq \delta} A_{r-i}\sigma_i = 0 \\ \forall\, i \in \bigcup_{j=1}^{\delta-1} \mathrm{cl}(j),\ \ A_i = 0 \\ \sigma(z) \mid z^n - 1 \end{cases}$$

où $\sigma(z) = 1 + \sum_{1\leq i\leq \delta} \sigma_i z^i$.

Proposition II.8 *Toute solution $(\sigma_i)_{0<i\leq\delta}$ et $(A_i)_{0<i<n+\delta}$ de (S_δ), vérifie*

- *$\sigma_i = 0$ pour i impair, $0 < i < \delta$.*
- *$A_n = 1$.*
- *$A_\delta \neq 0$.*

De plus si $\mathrm{pgcd}(n,\delta) = 1$, *et s'il existe des solutions à (S_δ), alors il existe une solution telle que $A_\delta = 1$.*

preuve :

- D'après le lemme II.2, les σ_i sont les fonctions symétriques élémentaires d'un mot $\mathbf{a} \in B(n,\delta)$. Le théorème II.2 nous assure alors que $\sigma_i = 0$ pour i impair, $0 < i < \delta$.
- $A_n = \sum_{i=1}^{\delta} X_i^n = \delta = 1$, car δ est impair.

- S'il existe une solution à (S_δ), d'après le lemme II.2, les solutions A_i sont les fonctions puissances symétriques d'un mot $\mathbf{a} \in B(n,\delta)$ de poids δ, d'après la proposition II.5, $A_\delta \neq 0$.

La même proposition II.5 nous assure aussi que si $\mathrm{pgcd}(n,\delta) = 1$, il existe une permutation circulaire de $\mathbf{a}$ dont la δ-ième fonction puissance symétrique est égale à 1. Donc d'après le lemme II.2, il existe une solution de (S_δ) telle que $A_\delta = 1$. □

Proposition II.9 *Les équations de Newton pour $B(n,\delta)$ et pour le poids δ, d'indices impairs r, $\delta+2 \leq r \leq 2\delta-1$, forment un système linéaire triangulaire, inversible lorsque $A_\delta \neq 0$, donnant les $(\sigma_i)_{0<i\leq\delta}$ d'indices pairs en fonction des A_i.*

preuve : Pour tout j, $1 \leq j \leq (\delta-1)/2$, on a

$$eq_{\delta+2j} \;:\; A_{\delta+2j} + \sum_{i=1}^{j} \sigma_{2i} A_{2(j-i)+\delta} = 0.$$

Si on écrit le système constitué de ces équations, on obtient

$$\begin{pmatrix} A_\delta & 0 & \dots & 0 \\ A_{\delta+2} & A_\delta & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A_{2\delta-3} & \dots & A_{\delta+2} & A_\delta \end{pmatrix} \begin{pmatrix} \sigma_2 \\ \sigma_4 \\ \vdots \\ \sigma_{\delta-1} \end{pmatrix} = \begin{pmatrix} A_{\delta+2} \\ A_{\delta+2} \\ \vdots \\ A_{2\delta-1} \end{pmatrix}$$

système qui est inversible si $A_\delta \neq 0$. □

Proposition II.10 *Les équations de Newton pour $B(n,\delta)$ et pour le poids δ, d'indices impairs r, $n+2 \leq r \leq n+\delta-1$, forment un système linéaire triangulaire, inversible lorsque $A_0 \neq 0$, donnant les $(\sigma_i)_{0<i\leq\delta}$ d'indices pairs en fonction des A_i.*

preuve : Pour tout j, $1 \leq j \leq (\delta-1)/2$

$$eq_{n+2j} \;:\; A_0\sigma_{2j} + \sum_{i=1}^{\frac{\delta-1}{2}-j} \sigma_{2(i+j)} A_{-2i} + A_{-\delta+2j}\sigma_\delta = 0$$

Si on écrit le système constitué de ces équations, on obtient

$$\begin{pmatrix} A_0 & A_{-2} & \dots & A_{-\delta+3} \\ 0 & A_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & A_{-2} \\ 0 & \dots & 0 & A_0 \end{pmatrix} \begin{pmatrix} \sigma_2 \\ \sigma_4 \\ \vdots \\ \sigma_{\delta-1} \end{pmatrix} = \begin{pmatrix} A_{-\delta+2}\sigma_\delta \\ A_{-\delta+4}\sigma_\delta \\ \vdots \\ A_{-1}\sigma_\delta \end{pmatrix}$$

système qui est inversible si $A_0 \neq 0$. □

3 Utilisation des équations de Newton

3.1 Recherche de contradictions

3.1.1 Comment prouver qu'un code dépasse sa distance construite?

L'ensemble des résultats que nous avons exposé jusquà présent a pour but de simplifier la résolution des équations de Newton. Nous allons nous interesser plus particulièrement aux codes BCH primitifs au sens strict. Lorsque l'on veut montrer qu'un tel code dépasse sa distance construite δ, il suffit de prouver qu'il n'existe pas de mots de poids δ dans le code.

D'après le théorème II.3 il suffit pour cela de démontrer l'absence de solution à un certain système d'équations sur $\mathbb{F}_{2^m}$

$$(S_\delta) \quad \begin{cases} 0 < r \leq \delta,\ eq_r\ :\ A_r + \sum_{1 \leq i < r} A_{r-i}\sigma_i + r\sigma_r = 0 \\ \delta < r < n+\delta,\ eq_r\ :\ A_r + \sum_{1 \leq i \leq \delta} A_{r-i}\sigma_i = 0 \\ \forall\, i \in I(B(n,\delta)),\ A_i = 0 \\ \sigma(z) \mid z^n - 1 \end{cases}$$

d'indéterminées $(\sigma_i)_{0<i\leq\delta}$ et $(A_i)_{0<i<n+\delta}$, où $\sigma(z) = 1 + \sum_{i=1}^{\delta} \sigma_i z^i$.

Les propositions II.7 et II.8, nous assurent que toute solution de (S_δ) vérifie également

$$\forall\, i,\ n < i < n+\delta,\ A_{n+i} = A_i \qquad \text{(II.11)}$$

$$\forall\, i,\ 0 < i \leq n,\ A_{2i \bmod n} = A_i^2 \qquad \text{(II.12)}$$

$$A_\delta \neq 0 \qquad \text{(II.13)}$$

$$A_n = A_0 = 1 \qquad \text{(II.14)}$$

Soit J l'ensemble des minima des classes cyclotomiques modulo n sur $\mathbb{F}_2$, d'après le corollaire II.3 la condition (II.12) est équivalente à

$$\forall\, i,\ 0 < i \leq n,\ j = \min \mathrm{cl}(i),\ A_i = A_j^{ij^{-1} \bmod n}. \qquad \text{(II.15)}$$

Notons que deux éléments i et j sont dans la même classe cyclotomique modulo n sur $\mathbb{F}_2$ si et seulement si $ij^{-1} \bmod n$ est une puissance de 2.

Nous décrivons ici la procédure qui nous a permis d'obtenir les preuves des résultats énoncés plus loin sur les distances minimales des codes $B(255, 59)$, $B(255, 61)$ et $B(511, 123)$. Nous supposerons que n et δ sont premiers entre eux.

On procède comme suit

1. on écrit les equations de Newton pour le poids δ,
2. on prend en compte le code $B(n,\delta)$: $\sigma_i = 0$ pour $i < \delta$ impair, et $A_i = 0$ pour $0 < i < \delta$,
3. on calcule J l'ensemble des minima des classes cyclotomiques modulo n et à l'aide des conditions (II.11) et (II.12), on élimine tous les A_i sauf pour $i \in J$,

4. on pose $A_\delta = 1$ et $A_0 = 1$,

5. on résoud le système inversible donné par la proposition II.9 ou la proposition II.10, qui nous donne les σ_i en fonction des A_i ,

6. le système résultant a pour seules indéterminées les A_i pour $i \in J$, et est polynomial. On essaie de prouver qu'il est inconsistant en tentant de le résoudre pas à pas.

En annexe A, cette procédure est utilisée avec succès dans les trois cas cités plus haut.

3.1.2 Résultats nouveaux sur les codes BCH

Nous allons présenter dans ce paragraphe deux exemples de codes dont la distance minimale était inconnue.

En longueur 255, tous les codes BCH primitifs au sens strict atteignent leur bornes BCH, sauf les codes de distance construite 59 et 61. Il est conjecturé que ces deux codes ont une distance minimale qui depasse leur distance construite, mais la preuve d'un tel fait demandait jusqu'à présent de passer en revue une grande partie du code, et cette recherche est très couteuse (cf.[29]).

Proposition II.11 *1. Le code $B(255,59)$ a pour distance minimale 61,*

2. le code $B(255,61)$ a pour distance minimale 63.

preuve : J-L. Dornstetter donne dans [29] un mot de poids 61 dans $B(255,59)$ ainsi qu'un mot de poids 63 dans $B(255,61)$.

Il reste donc a montrer qu'il n'existe pas de mots de poids égal à la distance construite pour chacun de ces deux codes. Cette preuve est donnée dans l'annexe A. □

Lemme II.3 *Soit $n = 2^m - 1$. Les éléments de la classe cyclotomique d'un entier $i \in [1, n[$ modulo n sur $\mathbb{F}_2$ sont les entiers dont l'écriture en base 2 est une permutation circulaire des m termes de l'écriture de i.*

preuve : Soit

$$i = \sum_{j=0}^{m-1} a_j 2^j,$$

l'écriture en base 2 de l'entier i. On a

$$2i = \sum_{j=0}^{m-1} a_j 2^{j+1} \equiv a_{m-1} + \sum_{j=1}^{m-1} a_{j-1} 2^j \bmod n,$$

d'où le résultat. □

Lemme II.4 *Le code $B(511,123)$ est inclus dans le code de Reed-Muller raccourci $\mathcal{R}^*(4,9)$.*

preuve : Le code $\mathcal{R}^*(4,9)$ a pour ensemble de définition

$$I(\mathcal{R}^*(4,9)) = \{i \in [1..511[\mid w_2(i) < 5\}.$$

Nous voulons montrer que

$$I(\mathcal{R}^*(4,9)) \subset I(B(n,\delta)),$$

ou, de façon équivalente, que

$$\forall\, i \in [1,n[,\ w_2(i) < 5 \Rightarrow \min(\mathrm{cl}(i)) < 123.$$

Par convention on représente une classe cyclotomique par son plus petit élément, et on munit l'ensemble des classes cyclotomiques modulo n de l'ordre induit par l'ordre naturel sur $\mathbb{N}$. La plus grande classe cyclotomique de 2-poids ≤ 4 est la classe représentée par l'entier dont l'écriture en base 2 est $(1,0,1,0,1,0,1,0,0)$, c'est-à-dire 85.

En effet, d'après le lemme II.3 un entier est dans $I(B(511,123))$, si et seulement si toutes les permutations circulaires de son écriture en base 2 y sont aussi, or un examen rapide permet de se convaincre que tout entier de 2-poids 4 ou moins écrit en base 2 peut être permuté pour donner un entier inferieur ou égal à 85.

Donc $B(511,123) \subset B(511,87) \subset \mathcal{R}^*(4,9)$. □

Proposition II.12 *Le code $B(511,123)$ a pour distance minimale 127.*

preuve : Puisque $B(511,123) \subset \mathcal{R}^*(4,9)$, la même inclusion reste vraie pour les codes étendus en ajoutant un bit égal à la somme des autres, et d'après [67, Cor. 13,p. 447], le code de Reed-Muller d'ordre 4 et de longueur 512 a des poids multiples de 4. Donc la distance minimale de $B(511,123)$ est un multiple de 4 moins 1.

On a $B(511,127) \subset B(511,123)$ et $B(511,127)$ atteint sa distance construite, donc la distance minimale de $B(511,123)$ est 123 ou 127. Nous donnons en annexe A la preuve que cette distance minimale n'est pas 123. □

3.2 Recherche de solutions particulières : les idempotents

3.2.1 Définition et caractérisation des idempotents

Définition II.6 *Le support d'un mot* $\mathbf{a} \in \mathbb{F}_2^n$ *noté* $\mathrm{supp}(x)$ *est l'ensemble des positions de ses coordonnées non nulles numérotée de 0 à $n-1$.*

Le support d'un mot d'un code cyclique est égal à l'ensemble des logarithmes en base α de ses localisateurs.

Définition II.7 *Soient* $\mathbf{a} \in \mathbb{F}_2^n$ *et* $a(z) \in \mathbb{F}_2[z]$ *le polynôme associé à* $\mathbf{a}$. $\mathbf{a}$ *est un idempotent si et seulement si* $a(z)^2 = a(z) \bmod (z^n - 1)$.

Lemme II.5 *Soit* $\mathbf{a}$ *un mot de* $\mathbb{F}_2^n$*, les assertions suivantes sont equivalentes*

(i) $\mathbf{a}$ *est un idempotent*

(ii) *le support de* $\mathbf{a}$ *est la réunion de classes cyclotomiques modulo* n

(iii) *le polynôme localisateur de* $\mathbf{a}$ *est à coefficients dans* $\mathbb{F}_2$

(iv) *les fonctions puissances symétriques des localisateurs de* $\mathbf{a}$ *sont dans* $\mathbb{F}_2$

preuve : Soient $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}_{2^m}^n$ et $a(z)$ son polynôme associé.

(i)⇒(ii) On a $a(z) = a(z)^2$, donc pour tout i

$$i \in \mathrm{supp}(\mathbf{a}) \Rightarrow 2i \in \mathrm{supp}(\mathbf{a}^2) = \mathrm{supp}(\mathbf{a}),$$

d'où $i \in \mathrm{supp}(\mathbf{a}) \;\Rightarrow\; \mathrm{cl}(\mathbf{i}) \subset \mathrm{supp}(\mathbf{a})$.

(ii)⇒(iii) Les racines du polynôme localisateur $\sigma(z)$ de $\mathbf{a}$ sont les inverses des localisateurs, donc les localisateurs sont une réunion de classes de conjugaisons, or pour tout entier i

$$\prod_{j \in \mathrm{cl}(i)} (1 - \alpha^j z) \in \mathbb{F}_2[z],$$

car c'est le polynôme minimal de α^{-i}, donc $\sigma(z) \in \mathbb{F}_2[z]$.

(iii)⇒(iv) Si $\sigma(z) \in \mathbb{F}_2[z]$, alors pour tout i, $\sigma_i \in \mathbb{F}_2$, et d'après la preuve du lemme II.1 page 51, on a par récurrence $A_j \in \mathbb{F}_2$, pour tout j

(iv)⇒(i) Soit $A(z)$ le polynôme de Mattson-Solomon de $\mathbf{a}$. D'après [67, Th. 22, page 240], on a

$$a(z)^2 = a(z) \Leftrightarrow A(z) * A(z) = A(z),$$

où $*$ représente le produit composante par composante. Puisque $A_j^2 = A_j$ pour tout j, on a bien $A(z) * A(z) = A(z)$ et donc, $a(z)^2 = a(z)$.

□

3.2.2 Recherche d'idempotents de poids minimal

Il s'agit ici de prouver qu'un code BCH binaire primitif au sens strict $B(n, \delta)$ de longueur $n = 2^m - 1$ et de distance construite δ, a pour distance minimale $d = \delta$. Pour cela il suffit prouver qu'il existe un mot de poids δ ou de poids $\delta + 1$.

Proposition II.13 *Soit le système*

$$(S'_w) \quad \begin{cases} 0 < r \leq w, \; eq_r \;:\; A_r + \sum_{1 \leq i < r} A_{r-i}\sigma_i + r\sigma_r = 0 \\ w < r < n, \; eq_r \;:\; A_r + \sum_{1 \leq i \leq w} A_{r-i}\sigma_i = 0 \\ \forall\, i \in I(B(n,\delta)), \; A_i = 0 \\ \sigma(z) \mid z^n - 1 \\ \forall\, i, \; 0 \leq i < n, \; A_i^2 = A_i \end{cases}$$

Si (S'_w) *admet une solution pour* $w = \delta$ *ou* $w = \delta + 1$, *alors* $B(n, \delta)$ *a pour distance minimale* δ.

preuve : Si (S'_w) admet une solution, alors d'après le lemme II.2 et le lemme II.5 $B(n,\delta)$ admet un mot idempotent de poids w. Puisque $w \in \{\delta, \delta+1\}$ et puisque la distance minimale d de $B(n,\delta)$ est impaire, on en conclut que $d = \delta$. □

Remarque II.1 Par la proposition ci-dessus, nous limitons la recherche de mots de poids δ ou $\delta + 1$ dans $B(n,\delta)$ aux seuls idempotents, ce qui donne une forme particulièrement simple aux équations de Newton et autorise une résolution relativement rapide. Il est bien sûr possible qu'un code atteigne sa distance minimale sans que ce soit le cas pour un idempotent, nous ne pourrons alors pas conclure.

Théorème II.4 *Le code $B(511,\delta)$ contient des idempotents de poids δ ou $\delta+1$, donc atteint sa distance construite si*

$$\delta \in \{19, 39, 45, 53, 57, 79, 83, 91, 103\}.$$

preuve : Pour chacun des codes de l'énoncé, nous avons obtenu en résolvant les équations de Newton, un mot $\mathbf{x}$ idempotent de poids δ ou $\delta+1$. D'après le lemme II.5 le support de $\mathbf{x}$ est réunion de classes cyclotomiques, et puisque $\mathbf{x} \in \mathbb{F}_2^n$, ce support le définit entièrement.

On a $\mathbf{x} \in B(n,\delta)$ pour

$$\begin{array}{rl}
\delta = 19,\ \omega(\mathbf{x}) = 19, & \mathrm{supp}(\mathbf{x}) = \mathrm{cl}(0) \cup \mathrm{cl}(23) \cup \mathrm{cl}(91) \\
\delta = 39,\ \omega(\mathbf{x}) = 39, & \mathrm{supp}(\mathbf{x}) = \mathrm{cl}(63) \cup \mathrm{cl}(87) \cup \mathrm{cl}(117) \cup \mathrm{cl}(127) \cup \mathrm{cl}(219) \\
\delta = 45,\ \omega(\mathbf{x}) = 45, & \mathrm{supp}(\mathbf{x}) = \mathrm{cl}(17) \cup \mathrm{cl}(37) \cup \mathrm{cl}(57) \cup \mathrm{cl}(93) \cup \mathrm{cl}(103) \\
\delta = 53,\ \omega(\mathbf{x}) = 54, & \mathrm{supp}(\mathbf{x}) = \mathrm{cl}(17) \cup \mathrm{cl}(31) \cup \mathrm{cl}(41) \cup \mathrm{cl}(45) \cup \mathrm{cl}(103) \cup \mathrm{cl}(117) \\
\delta = 57,\ \omega(\mathbf{x}) = 57, & \mathrm{supp}(\mathbf{x}) = \mathrm{cl}(29) \cup \mathrm{cl}(43) \cup \mathrm{cl}(51) \cup \mathrm{cl}(55) \cup \mathrm{cl}(61) \cup \mathrm{cl}(63) \\
 & \quad \cup\, \mathrm{cl}(219) \\
\delta = 79,\ \omega(\mathbf{x}) = 79, & \mathrm{supp}(\mathbf{x}) = \mathrm{cl}(0) \cup \mathrm{cl}(3) \cup \mathrm{cl}(13) \cup \mathrm{cl}(39) \cup \mathrm{cl}(41) \cup \mathrm{cl}(61) \cup \mathrm{cl}(73) \\
 & \quad \cup\, \mathrm{cl}(77) \cup \mathrm{cl}(107) \cup \mathrm{cl}(117) \cup \mathrm{cl}(219) \\
\delta = 83,\ \omega(\mathbf{x}) = 84, & \mathrm{supp}(\mathbf{x}) = \mathrm{cl}(11) \cup \mathrm{cl}(15) \cup \mathrm{cl}(23) \cup \mathrm{cl}(43) \cup \mathrm{cl}(53) \cup \mathrm{cl}(79) \\
 & \quad \cup\, \mathrm{cl}(123) \cup \mathrm{cl}(183) \cup \mathrm{cl}(191) \cup \mathrm{cl}(219) \\
\delta = 91,\ \omega(\mathbf{x}) = 91, & \mathrm{supp}(\mathbf{x}) = \mathrm{cl}(0) \cup \mathrm{cl}(7) \cup \mathrm{cl}(13) \cup \mathrm{cl}(25) \cup \mathrm{cl}(37) \cup \mathrm{cl}(41) \cup \mathrm{cl}(59) \\
 & \quad \cup\, \mathrm{cl}(61) \cup \mathrm{cl}(117) \cup \mathrm{cl}(175) \cup \mathrm{cl}(239) \\
\delta = 103,\ \omega(\mathbf{x}) = 103, & \mathrm{supp}(\mathbf{x}) = \mathrm{cl}(0) \cup \mathrm{cl}(7) \cup \mathrm{cl}(13) \cup \mathrm{cl}(19) \cup \mathrm{cl}(27) \cup \mathrm{cl}(31) \cup \mathrm{cl}(87) \\
 & \quad \cup\, \mathrm{cl}(91) \cup \mathrm{cl}(95) \cup \mathrm{cl}(191) \cup \mathrm{cl}(219) \cup \mathrm{cl}(223) \cup \mathrm{cl}(255).
\end{array}$$

Donc pour tous ces codes la distance construite est atteinte. □

La procédure pour la recherche d'idempotents solution des équations de Newton est donnée en annexe A. Il est bon de préciser que cette procédure a échoué pour $B(511,\delta)$, et $\delta \in \{29, 37, 41, 43, 51, 59, 61, 75, 77, 85, 87, 107\}$.

Nous donnons dans le tableau II.1, la liste des codes BCH primitifs au sens strict de longueur 511, et la valeur de leur distance minimale ou de la meilleure borne connue.

n	k	δ	d	dans	n	k	δ	d		dans
511	502	3	3	[56]	511	241	73	73		[75]
	493	5	5	[56]		238	75	≥ 75		—
	484	7	7	[56]		229	77	≥ 77		—
	475	9	9	**		220	79	79		*
	466	11	11	[56]		211	83	83		*
	457	13	13	[47]		202	85	≥ 85		—
	448	15	15	[56]		193	87	≥ 87		—
	439	17	17	**		184	91	91		*
	430	19	19	*		175	93	95	#	[59]
	421	21	21	[75]		166	95	95		[56]
	412	23	23	[56]		157	103	103		*
	403	25	25	[47]		148	107	≥ 107		—
	394	27	27	[56]		139	109	111	#	[59]
	385	29	≥ 29	—		130	111	111		[56]
	376	31	31	[56]		121	117	119	#	[59]
	367	35	35	[75]		112	119	119		[56]
	358	37	≥ 37	—		103	123	127	##	*
	349	39	39	*		94	125	127	#	[59]
	340	41	≥ 41	—		85	127	127		[56]
	331	43	≥ 43	—		76	171	171		**
	322	45	45	*		67	175	175		**
	313	47	47	[56]		58	183	183		**
	304	51	≥ 51	—		49	187	187		**
	295	53	53	*		40	191	191		[56]
	286	55	55	[56]		31	219	219		[75]
	277	57	57	*		28	223	223		[56]
	268	59	≥ 59	—		19	239	239		[56]
	259	61	≥ 61	—		10	255	255		[56]
	250	63	63	[56]						

\# $d = \delta + 2$
\## $d = \delta + 4$
* Résultat obtenu par les équations de Newton
** Résultat obtenu par une recherche exhaustive

Tableau II.1 : Distance minimale des codes BCH de longueur 511

Chapitre III

Codes concaténés

Les codes concaténés ont été introduits par Forney dans [36] en 1966. Ces travaux ont ensuite été poursuivis par les russes Block, Zinoviev et Zyablov, qui ont abouti à un algorithme de décodage performant, s'inspirant des idées de Forney sur le décodage souple, et surtout aux codes concaténés généralisés [12], [103] et [104].

Nous nous sommes volontairement limité ici au cas linéaire. La plupart des résultats restent cependant valables dans le cas général. Cette limitation permet de donner des définitions qui nous semblent plus simples que celles données généralement. En particulier nous définissons, dans la section 3, les codes concaténés géneralisés comme une somme directe de codes concaténés d'ordre 1, cette représentation ayant déjà été utilisée par Jensen dans un cas particulier (cf. [50]).

Dans la section 1, outre la définition des codes concaténés d'ordre 1 et la description de leur décodage, nous introduisons la notion de décodage partiel qui nous permet dans la section 3 de donner une description récursive simple du décodage des codes concaténés géneralisés, description que Bossert donne en filigrane dans [14].

Dans la section 2, nous utilisons une approche originale pour l'évaluation des performances des codes concaténés d'ordre 1, faisant appel à la combinatoire énumérative. De telles évaluations avaient été faites sous une forme légèrement différente pour l'algorithme naïf dans [21]. Nous avons pu étendre ces résultats à tous les algorithmes décrits dans la section 1.

Soit un corps fini $\mathbb{F}_q$. Pour tout couple d'entiers n_b et n_e, nous identifierons chaque élément de $\mathbb{F}_q^{n_b n_e}$ à un élément de $(\mathbb{F}_q^{n_b})^{n_e}$. En particulier lorsque l'on posera $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathbb{F}_q^{n_b n_e}$, cela signifiera que pour tout i, $1 \leq i \leq n_e$, on a $x_i \in \mathbb{F}_q^{n_b}$. Le vecteur $\mathbf{x}$ aura comme composantes, dans l'ordre, les composantes de x_1, puis celles de x_2, jusqu'à celles de x_{n_e}.

1 Codes concaténés d'ordre 1 – Algorithmes de décodage

1.1 Définition

Soit un code linéaire $\mathcal{B} = \mathcal{B}(n_b, k_b, d_b)$ sur $\mathbb{F}_q$, $\mathcal{B}$ et $\mathbb{F}_{q^{k_b}}$ sont isomorphes en tant que $\mathbb{F}_q$-espaces vectoriels. Fixons un isomorphisme

$$\begin{array}{cccc} \theta & : & \mathbb{F}_{q^{k_b}} & \longrightarrow & \mathcal{B} \\ & & x & \longmapsto & \theta(x). \end{array} \tag{III.1}$$

θ permet de représenter de façon unique tout élément de $\mathbb{F}_{q^{k_b}}$ par un élément du code $\mathcal{B}$, et réciproquement.

Soit n_e un entier positif. On définit l'isomorphisme $\mathbb{F}_q$-linéaire Θ comme suit :

$$\begin{array}{cccc} \Theta & : & \mathbb{F}_{q^{k_b}}^{n_e} & \longrightarrow & \mathcal{B}^{n_e} \\ & & \mathbf{x} = (x_1, \ldots, x_{n_e}) & \longmapsto & \Theta(\mathbf{x}) = (\theta(x_1), \ldots, \theta(x_{n_e})) \end{array} \tag{III.2}$$

Définition III.1 *Soient $\mathcal{B} = \mathcal{B}(n_b, k_b, d_b)$ un code linéaire sur $\mathbb{F}_q$, $\mathcal{E} = \mathcal{E}(n_e, k_e, d_e)$ un code linéaire sur $\mathbb{F}_{q^{k_b}}$ et θ l'isomorphisme défini par* (III.1).

Le code concaténé admettant $\mathcal{B}$ comme code interne et $\mathcal{E}$ comme code externe est le code $\mathcal{C}$ sur $\mathbb{F}_q$ constitué des mots de $\mathcal{E}$ dans lesquels les symboles de $\mathbb{F}_{q^{k_b}}$ sont remplacés par des mots de $\mathcal{B}$ à l'aide de θ

$$\mathcal{C} = \{(\theta(x_1), \theta(x_2), \ldots, \theta(x_{n_e})) \mid (x_1, x_2, \ldots, x_{n_e}) \in \mathcal{E}\} = \Theta(\mathcal{E}).$$

On dira en abrégé que $\mathcal{C}$ est le code concaténé de $\mathcal{B}$ et de $\mathcal{E}$.

Remarque III.1 Le code $\mathcal{E}$ introduit dans la définition ci-dessus est un espace vectoriel sur $\mathbb{F}_{q^{k_b}}$ de dimension k_e, mais puisque $\mathbb{F}_{q^{k_b}}$ peut être considéré comme un espace vectoriel sur $\mathbb{F}_q$ de dimension k_b, on peut considérer $\mathcal{E}$ comme un espace vectoriel sur $\mathbb{F}_q$ de dimension $k_b k_e$.

Proposition III.1 *Soit Θ l'isomorphisme défini par* (III.2). *Si $\mathcal{C} = \Theta(\mathcal{E})$ est le code concaténé de $\mathcal{B}(n_b, k_b, d_b)$ et $\mathcal{E}(n_e, k_e, d_e)$, alors $\mathcal{C}$ est un code linéaire sur $\mathbb{F}_q$ de longueur $n_b n_e$, de dimension $k_b k_e$, et dont la distance minimale est au moins égale à $d_b d_e$.*

preuve : Puisque Θ est injectif et $\mathbb{F}_q$-linéaire, $\mathcal{C}$ a pour dimension $\dim_{\mathbb{F}_q}(\mathcal{E}) = k_b k_e$.

Montrons que tout mot non nul du code concaténé a un poids supérieur ou égal à $d_b d_e$. Soit $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathcal{E}$. Alors

$$w_H(\Theta(\mathbf{x})) = \sum_{i=1}^{n_e} w_H(\theta(x_i))$$

où w_H est le poids de Hamming sur $\mathbb{F}_q$. Si $\mathbf{x} \neq 0$, alors au moins d_e des x_i sont non nuls et pour chacun de ces x_i, $\theta(x_i) \in \mathcal{B} \setminus \{0\}$, donc $w_H(\theta(x_i)) \geq d_b$, d'où le résultat. □

Remarque III.2 Les codes concaténés sont définis ici comme des codes linéaires, Forney les a introduits dans [36] de façon plus générale.

Dans le cas non linéaire il suffit de pouvoir mettre en bijection l'ensemble des mots du code interne avec l'alphabet du code externe. Les résultats sur les paramètres donnés dans la proposition III.1 restent vrais en remplaçant la dimension par le nombre d'éléments.

1.2 Algorithme de décodage naïf

Supposons donnés :

- $\mathcal{B} = \mathcal{B}(n_b, k_b, d_b)$ le code interne sur $\mathbb{F}_q$ muni d'un algorithme de décodage d'erreur γ borné par la distance minimale,
- $\mathcal{E} = \mathcal{E}(n_e, k_e, d_e)$ le code externe sur $\mathbb{F}_{q^{k_b}}$ est muni d'un algorithme de décodage d'erreur et d'effacement Φ borné par la distance minimale,
- $\mathcal{C} = \Theta(\mathcal{E})$ le code concaténé de $\mathcal{B}$ et $\mathcal{E}$.

Proposition III.2 *Soit l'application* Γ

$$\begin{array}{rrcl} \Gamma \ : & (\mathbb{F}_q^{n_b})^{n_e} & \longrightarrow & (\mathbb{F}_{q^{k_b}} \cup \{\infty\})^{n_e} \\ & \mathbf{y} = (y_1, \ldots, y_{n_e}) & \longmapsto & \Gamma(\mathbf{y}) = (\theta^{-1}(\gamma(y_1)), \ldots, \theta^{-1}(\gamma(y_{n_e}))) \end{array} \tag{III.3}$$

(avec la convention $\theta^{-1}(\infty) = \infty$).

Alors $\Psi = \Theta \circ \Phi \circ \Gamma$ est un algorithme de décodage d'erreur du code concaténé $\mathcal{C}$ de $\mathcal{B}$ et de $\mathcal{E}$, borné par $\frac{d_b d_e}{2}$, autrement dit, pour tout $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$ et pour tout $\mathbf{x} \in \mathcal{C}$, on a

$$d_H(\mathbf{x}, \mathbf{y}) < \frac{d_b d_e}{4} \Rightarrow \Psi(\mathbf{y}) = \mathbf{x}.$$

preuve : Soient $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathcal{C}$, $\mathbf{y} = (y_1, \ldots, y_{n_e}) \in \mathbb{F}_q^{n_b n_e}$. Posons pour tout i

$$w_i = d_H(x_i, y_i),$$

et, par hypothèse,

$$d_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n_e} w_i < \frac{d_b d_e}{4}.$$

Les distances Δ_H et δ_H sont définies sur $\mathbb{F}_{q^{k_b}} \cup \{\infty\}$. Supposons que $\Psi(\mathbf{y}) = \Theta(\Phi(\Gamma(\mathbf{y}))) \neq \mathbf{x}$. Puisque Φ est borné par d_e, on a (cf. définition I.26)

$$\Delta_H(\Theta^{-1}(\mathbf{x}), \Gamma(\mathbf{y})) = \sum_{i=1}^{n_e} \delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma(y_i))) \geq \frac{d_e}{2}. \tag{III.4}$$

$$\begin{tikzcd}
& (\mathcal{B}\cup\{\infty\})^{n_e} \arrow[dr, "(\theta^{-1})^{n_e}"] & \\
\mathbb{F}_q^{n_b n_e} \equiv (\mathbb{F}_q^{n_b})^{n_e} \arrow[ur, "\gamma^{n_e}"] \arrow[rr, "\Gamma"] \arrow[d, "\Psi"] & & (\mathbb{F}_{q^{k_b}}\cup\{\infty\})^{n_e} \arrow[d, "\Phi"] \\
\mathcal{C}\cup\{\infty\} & & \mathcal{E}\cup\{\infty\} \arrow[ll, "\Theta"']
\end{tikzcd}$$

Figure III.1 : Décodage naïf d'un code concaténé

Par définition, $\delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma(y_i))) \neq 0$ si et seulement si $\gamma(y_i) \neq x_i$, et puisque γ est borné par d_b, on peut écrire

$$\delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma(y_i))) \neq 0 \Rightarrow w_i = d_H(x_i, y_i) \geq \frac{d_b}{2}.$$

Si $I = \{i \mid \delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma(y_i))) \neq 0\}$, on a pour tout $i \in I$

$$w_i \geq \frac{d_b}{2},$$

de plus puisque δ_H est à valeur dans $\{0, \frac{1}{2}, 1\}$ et que

$$\sum_{i=1}^{n_e} \delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma(y_i))) = \sum_{i\in I} \delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma(y_i))) \geq \frac{d_e}{2},$$

le cardinal de I est au moins égal à $\frac{d_e}{2}$, d'où

$$\sum_{i=1}^{n_e} w_i \geq \frac{d_b d_e}{4}$$

ce qui contredit l'hypothèse. On a donc bien $\Psi(\mathbf{y}) = \mathbf{x}$. $\square$

Cette borne de $\frac{d_b d_e}{2}$ n'est pas excellente, puisque qu'on peut espérer atteindre $d_b d_e$. L'exemple suivant montre qu'il existe, pour au moins un code, des motifs d'erreur de poids inférieur à $\frac{d_b d_e}{2}$, et que Ψ ne peut pas décoder.

Exemple :

Considérons le code concaténé défini par les données suivantes :

- Le code interne $\mathcal{B}$ est le code de Hamming binaire $H(7,4,3)$, γ est un algorithme de décodage à vraisemblance maximale, capable de corriger toute erreur de poids 1.
- Le code externe $\mathcal{E}$ est le code de Reed-Solomon $RS(15,13,3)$ sur $\mathbb{F}_{16}$, Φ est l'algorithme d'Euclide étendu, capable de corriger tout motif de ν erreurs et ρ effacements tel que $2\nu+\rho < 3$, et aucun autre (*i.e.* Φ *est borné strictement par la distance minimale du code*).

Le code concaténé résultant possède une distance minimale supérieure ou égale à 9. Montrons qu'il existe des motifs d'erreur de poids 4 qui ne sont pas corrigibles par Ψ.

Prenons comme mot émis le mot nul, et supposons que le mot reçu soit $\mathbf{y} = (y_1, y_2, 0, \ldots, 0) \in (\mathbb{F}_2^4)^{15}$, tel que $w_H(y_1) = w_H(y_2) = 2$. Ce mot $\mathbf{y}$ est à distance 4 du mot émis. Par exemple

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{15} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Le code de Hamming étant un code 1-correcteur parfait, $\gamma(y_1)$ et $\gamma(y_2)$ sont deux éléments non nuls de $H(7,4,3)$, et donc $\theta^{-1}(\gamma(y_1))$ et $\theta^{-1}(\gamma(y_2))$ sont deux éléments non nuls de $\mathbb{F}_{16}$. Dans notre exemple

$$\Gamma(\mathbf{y}) = (\beta, \beta, 0, \ldots, 0), \text{ où } \beta \in \mathbb{F}_{16}^*.$$

Le décodeur externe Φ doit donc décoder un mot qui contient 2 erreurs ; or

$$\Delta_H(\Gamma(\mathbf{y}), \mathbf{0}) = 2 > 3/2 \;\Rightarrow\; \Phi(\Gamma(\mathbf{y})) \neq \mathbf{0}$$

car Φ est borné strictement par 3. Donc le décodage va échouer, c'est-à-dire aboutir

- soit à ∞,
- soit à un mot non nul du code concaténé à distance ≥ 5 du mot reçu.

1.3 Algorithme de Block-Zyablov

Soit $\mathcal{C}$ le code concaténé de $\mathcal{B}$ et $\mathcal{E}$ munis comme plus haut des données suivantes :

- $\mathcal{B} = \mathcal{B}(n_b, k_b, d_b)$ le code interne est muni d'un algorithme de décodage γ, borné par la distance minimale d_b.

- $\mathcal{E} = \mathcal{E}(n_e, k_e, d_e)$ le code externe est muni d'un algorithme de décodage d'erreur et d'effacement Φ borné par la distance minimale d_e.

Pour tout nombre réel r, $0 \leq r < \frac{d_b}{2}$, on définit l'algorithme de décodage suivant γ_r de $\mathcal{B}$

$$\forall\, y \in \mathbb{F}_q^{n_b},\ \ \gamma_r(y) = \begin{cases} \gamma(y) & \text{si } d_H(\gamma(y), y) \leq r \\ \infty & \text{sinon} \end{cases},$$

on en déduit l'application Γ_r

$$\begin{array}{rccl} \Gamma_r \ \ : & \mathbb{F}_q^{n_b n_e} & \longrightarrow & (\mathbb{F}_{q^{k_b}} \cup \{\infty\})^{n_e} \\ & \mathbf{y} = (y_1, \ldots, y_{n_e}) & \longmapsto & \Gamma_r(\mathbf{y}) = (\theta^{-1}(\gamma_r(y_1)), \ldots, \theta^{-1}(\gamma_r(y_{n_e}))) \end{array}$$

et l'algorithme de décodage de $\mathcal{C}$, $\Phi_r = \Theta \circ \Phi \circ \Gamma_r$

$$\begin{array}{rccl} \Phi_r \ \ : & \mathbb{F}_q^{n_b n_e} & \longrightarrow & \mathcal{C} \cup \{\infty\} \\ & \mathbf{y} & \longmapsto & \Phi_r(\mathbf{y}) = \Theta(\Phi(\Gamma_r(\mathbf{y}))) \end{array}.$$

Remarque III.3 On a défini de cette manière une infinité de décodeurs, cependant un nombre finis d'entre eux seulement sont distincts, en effet

$$\forall\, s \in [r, r+1[\ ,\ \ \Phi_s = \Phi_r.$$

On a donc exactement $t+1$ décodeurs distincts $\Phi_0, \Phi_1, \ldots, \Phi_t$, où $t = \lfloor \frac{d_b - 1}{2} \rfloor$.

Définition III.2 *Soit d_b un entier strictement positif. Pour tout $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathbb{F}_q^{n_b n_e}$, et tout $\mathbf{y} = (y_1, \ldots, y_{n_e}) \in \mathbb{F}_q^{n_b n_e}$, on définit $D(\mathbf{x}, \mathbf{y})$, par*

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n_e} \min(d_H(x_i, y_i), d_b).$$

Proposition III.3 *D est une distance sur $\mathbb{F}_q^{n_b n_e}$, dominée par la distance de Hamming,*

$$\forall\, \mathbf{x} \in \mathbb{F}_q^{n_b n_e},\ \forall\, \mathbf{y} \in \mathbb{F}_q^{n_b n_e}, \quad D(\mathbf{x}, \mathbf{y}) \leq d_H(\mathbf{x}, \mathbf{y}).$$

preuve : l'application $d : (x, y) \mapsto \min(d_H(x, y), d_b)$ est une distance sur $\mathbb{F}_q^{n_b}$. En effet, on a bien

1. $d(x, y) = 0 \Leftrightarrow \min(d_H(x, y), d_b) = 0 \Leftrightarrow d_H(x, y) = 0 \Leftrightarrow x = y$,
2. $d(x, y) = \min(d_H(x, y), d_b) = d(y, x)$,
3. si $d_H(x, y) \geq d_b$ ou $d_H(y, z) \geq d_b$, alors

$$d(x, y) + d(y, z) \geq d_b \geq d(x, z),$$

sinon $d(x, y) = d_H(x, y)$ et $d(y, z) = d_H(y, z)$, donc

$$d(x, y) + d(y, z) = d_H(x, y) + d_H(y, z) \geq d_H(x, z) \geq d(x, z).$$

Pour $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathbb{F}_q^{n_b n_e}$ et $\mathbf{y} = (y_1, \ldots, y_{n_e}) \in \mathbb{F}_q^{n_b n_e}$, on a que

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n_e} \min(d_H(x_i, y_i), d_b) = \sum_{i=1}^{n_e} d(x_i, y_i),$$

ce qui entraine que D est une distance. De plus pour tout i, $d(x_i, y_i) \leq d_H(x_i, y_i)$, donc $D(\mathbf{x}, \mathbf{y}) \leq d_H(\mathbf{x}, \mathbf{y})$. □

Proposition III.4 *Soit $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$. Pour tout entier r, $0 \leq r \leq t = \lfloor \frac{d_b - 1}{2} \rfloor$, on pose*

$$\tilde{\mathbf{y}}_r = \Phi_r(\mathbf{y}) \in \mathcal{C} \cup \{\infty\}.$$

Alors tous les r pour lesquels $D(\tilde{\mathbf{y}}_r, \mathbf{y}) < \frac{d_b d_e}{2}$ (par convention $D(\infty, \mathbf{y}) = \infty$), donnent la même valeur $\tilde{\mathbf{y}}$ de $\tilde{\mathbf{y}}_r$.

On peut alors définir le décodeur de Block-Zyablov de $\mathcal{C}$, noté Ψ :

$$\forall\, \mathbf{y} \in \mathbb{F}_q^{n_b n_e},\ \Psi(\mathbf{y}) = \begin{cases} \tilde{\mathbf{y}}_r & si\ \exists\, r,\ tel\ que\ D(\tilde{\mathbf{y}}_r, \mathbf{y}) < \frac{d_b d_e}{2} \\ \infty & sinon \end{cases}. \tag{III.5}$$

preuve : Si $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathcal{C}$ et $\mathbf{x}' = (x'_1, \ldots, x'_{n_e}) \in \mathcal{C}$ sont distincts, alors d_e au moins de ces n_e composantes sont distinctes et, pour chacune d'elles $\min(d_H(x_i, x'_i), d_b) = d_b$ puisque $x_i, x'_i \in \mathcal{B}$. Donc

$$D(\mathbf{x}, \mathbf{y}) + D(\mathbf{x}', \mathbf{y}) \geq D(\mathbf{x}, \mathbf{x}') \geq d_b d_e,$$

et on ne peut donc pas avoir simultanément $D(\mathbf{x}, \mathbf{y})$ et $D(\mathbf{x}', \mathbf{y})$ inférieurs à $\frac{d_b d_e}{2}$. □

Théorème III.1 *Soit $\mathcal{C}$ le code concaténé de $\mathcal{B}(n_b, k_b, d_b)$ et $\mathcal{E}(n_e, k_e, d_e)$ muni d'un décodeur de Block-Zyablov Ψ. Alors Ψ est un algorithme de décodage de $\mathcal{C}$ borné par $d_b d_e$.*

Autrement dit, pour tout $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$ et pour tout $\mathbf{x} \in \mathcal{C}$ on a

$$d_H(\mathbf{x}, \mathbf{y}) < \frac{d_b d_e}{2} \Rightarrow \Psi(\mathbf{y}) = \mathbf{x}.$$

preuve : (*due à T. Ericson [35]*)

Soient $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathcal{C}$, $\mathbf{y} = (y_1, \ldots, y_{n_e}), \mathbf{z} = (z_1, \ldots, z_{n_e}) \in \mathbb{F}_q^{n_b n_e}$. On note

1. $\forall\, i,\ 1 \leq i \leq n_e,\quad w_i = d_H(x_i, y_i)$.

2. D la distance de la définition III.2

$$D(\mathbf{y}, \mathbf{z}) = \sum_{i=1}^{n_e} \min(d_H(y_i, z_i), d_b).$$

3. Définissons la fonction $E : \mathbb{R}^* \times [0, \frac{d_b}{2}[\to \{0,1,2\}$ par

$$E(w,r) = \begin{cases} 0 & 0 \leq w \leq r \\ 1 & r < w < d_b - r \\ 2 & w \geq d_b - r \end{cases}$$

où $w \in \mathbb{R}^*$ et $r \in [0, \frac{d_b}{2}[$.

Lemme III.1 *Pour tout réel r, $0 \leq r \leq \frac{d_b}{2}$ et pour tout réel $w > 0$, E vérifie les deux propriétés suivantes*

$$\sum_{i=1}^{n_e} E(w_i, r) \geq 2\Delta_H(\Theta^{-1}(\mathbf{x}), \Gamma_r(\mathbf{y})), \tag{III.6}$$

$$\int_0^{\frac{d_b}{2}} E(w,r)dr = \min(w, d_b). \tag{III.7}$$

preuve : (du lemme)

- Montrons (III.6). Pour tout i distinguons deux cas :
  1. $\delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma_r(y_i))) = 1/2$: dans ce cas $\gamma_r(y_i) = \infty$ car $x_i \neq \infty$, donc $w_i = d_H(x_i, y_i) > r$, sinon on aurait $\gamma_r(y_i) = x_i \neq \infty$. On en conclut que $E(w_i, r) \geq 1$, par définition de E.
  2. $\delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma_r(y_i))) = 1$: dans ce cas on a $\gamma_r(y_i) = \gamma(y_i) \in \mathcal{B}$ et $\gamma(y_i) \neq x_i$, donc par l'inégalité triangulaire

     $$w_i = d_H(y_i, x_i) \geq d_H(\gamma(y_i), x_i) - d_H(y_i, \gamma(y_i)) \geq d_b - r.$$

     En effet $\gamma(y_i) \in \mathcal{B}$, et $x_i \in \mathcal{B}$ étant distincts, on a $d_H(\gamma(y_i), x_i) \geq d_b$ et $d_H(y_i, \gamma(y_i)) \leq r$, par définition de γ_r. Donc $E(w_i, r) = 2$ par définition de E.

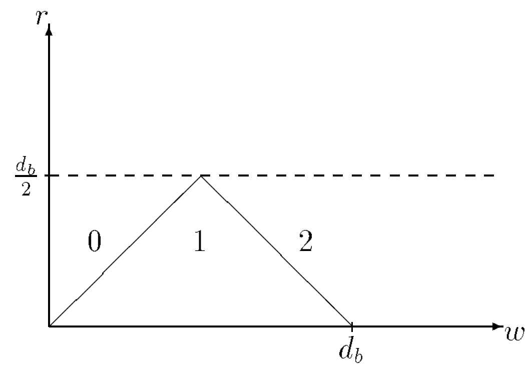
  Dans ces deux cas on a donc toujours

  $$E(w_i, r) \geq 2\delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma_r(y_i))),$$

  ce qui entraine (III.6) puisque par définition

  $$\Delta_H(\Theta^{-1}(\mathbf{x}), \Gamma_r(\mathbf{y})) = \sum_{i=1}^{n_e} \delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma_r(y_i))).$$

- L'égalité (III.7) se déduit aisément de la figure suivante qui donne les valeurs de $E(w,r)$ dans le plan (w,r)



□

Supposons maintenant que $d_H(\mathbf{x},\mathbf{y}) < \frac{d_b d_e}{2}$ et montrons que $\Psi(\mathbf{y}) = \mathbf{x}$. En utilisant les équations (III.6) et (III.7), la définition des w_i et le fait que la distance D est dominée par la distance de Hamming, il vient

$$\begin{aligned} D(\mathbf{x},\mathbf{y}) &= \sum_{i=1}^{n_e} \min(w_i, d_b), \\ & \sum_{i=1}^{n_e} \int_0^{\frac{d_b}{2}} E(w_i, r)dr, \\ & \int_0^{\frac{d_b}{2}} \sum_{i=1}^{n_e} E(w_i, r)dr \geq d_b \min_r \{\Delta_H(\Theta^{-1}(\mathbf{x}), \Gamma_r(\mathbf{y}))\}. \end{aligned}$$

Donc

$$\begin{aligned} \frac{d_b d_e}{2} &> d_b \min_r \{\Delta_H(\Theta^{-1}(\mathbf{x}), \Gamma_r(\mathbf{y}))\}, \\ \frac{d_e}{2} &> \min_r \{\Delta_H(\Theta^{-1}(\mathbf{x}), \Gamma_r(\mathbf{y}))\} \end{aligned}$$

et il existe un nombre réel r_0 tel que

$$\Delta_H(\Theta^{-1}(\mathbf{x}), \Gamma_{r_0}(\mathbf{y})) < \frac{d_e}{2}.$$

D'après la remarque III.3, on peut supposer que r_0 est un entier $0 \leq r_0 \leq t$. Comme Φ est borné par d_e et $\Theta^{-1}(\mathbf{x}) \in \mathcal{E}$, on a

$$\Phi_{r_0}(\mathbf{y}) = \Theta(\Phi(\Gamma_{r_0}(\mathbf{y}))) = \mathbf{x}$$

et la proposition III.4 nous assure alors que

$$\Psi(\mathbf{y}) = \mathbf{x}.$$

□

On a en fait un résultat plus fort en utilisant la distance D plutôt que la distance de Hamming.

Corollaire III.1 *L'algorithme de Block-Zyablov est borné strictement par $d_b d_e$ pour la métrique D, c'est-à-dire que pour tout $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$ et pour tout $\mathbf{x} \in \mathcal{C}$*

$$D(\mathbf{x}, \mathbf{y}) < \frac{d_b d_e}{2} \Leftrightarrow \Psi(\mathbf{y}) = \mathbf{x}.$$

preuve : Pour obtenir que Ψ est borné par $d_b d_e$ pour D, il suffit de reprendre telle quelle la preuve précédente en utilisant comme hypothèse $D(\mathbf{x}, \mathbf{y}) < \frac{d_b d_e}{2}$. Cette borne est stricte par la définition même du décodeur de Block-Zyablov (propositon III.4). □

L'algorithme de décodage peut se décrire comme suit

Algorithme $\mathcal{A}_1$ (de Block-Zyablov) *Soit* $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$.

1. *Pour tout entier r, $0 \leq r \leq t$, on calcule*

$$\tilde{\mathbf{y}}_r = \Phi_r(\mathbf{y}) \in \mathcal{C} \cup \{\infty\}.$$

2. *On calcule l'ensemble*

$$R = \{r \mid D(\tilde{\mathbf{y}}_r, \mathbf{y}) < \frac{d_b d_e}{2}\}.$$

3. *Si $R \neq \emptyset$, l'algorithme retourne*

$$\Psi(\mathbf{y}) = \tilde{\mathbf{y}}_{r_0} \text{ pour } r_0 \in R$$

sinon l'algorithme retourne

$$\Psi(\mathbf{y}) = \infty.$$

La validité de cet algorithme est assurée par la proposition III.4. En particulier le choix de $r_0 \in R$ est indifférent.

Proposition III.5 *L'algorithme $\mathcal{A}_1$ respecte la distance D, mais ne respecte pas la distance de Hamming.*

preuve : L'algorithme $\mathcal{A}_1$ respecte la distance D par définition. En effet, soit $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$ et soit $\mathbf{x} \in \mathcal{C}$. Si $\Psi(\mathbf{y}) \in \mathcal{C}$ et $D(\mathbf{x}, \mathbf{y}) \leq D(\Psi(\mathbf{y}), \mathbf{y}) < \frac{d_b d_e}{2}$ alors $\Psi(\mathbf{y}) = \mathbf{x}$ par l'inégalité triangulaire.

En revanche $\mathcal{A}_1$ ne respecte pas la distance de Hamming, comme le montre le contre-exemple suivant :

- Le code interne est le code de Hamming binaire $H(7, 4, 3)$, muni d'un algorithme de décodage d'erreur à vraisemblance maximale, capable de corriger tout motif de poids 1.

- Le code externe est le code de Reed-Solomon $RS(15,11,5)$ sur $\mathbb{F}_{16}$, muni d'un algorithme de décodage d'erreur et d'effacement borné strictement par 5.

On choisit le code de Hamming cyclique de matrice génératrice

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

Soit α un élément générateur de $\mathbb{F}_{16}^*$, de polynôme minimal $1+\alpha+\alpha^4$. Pour β dans $\mathbb{F}_{16}$, $\theta(\beta)$ sera l'image par G de son écriture dans la base $\{1,\alpha,\alpha^2,\alpha^3\}$ comme le montre la table suivante :

$$\begin{array}{lcllll}
 & & & \theta & & \\
0 & = & 0 & \longrightarrow & 0000 & 000 \\
\alpha & = & \alpha & \longrightarrow & 0100 & 011 \\
\alpha^2 & = & \alpha^2 & \longrightarrow & 0010 & 111 \\
\alpha^5 & = & \alpha+\alpha^2 & \longrightarrow & 0110 & 100 \\
\alpha^3 & = & \alpha^3 & \longrightarrow & 0001 & 101 \\
\alpha^9 & = & \alpha+\alpha^3 & \longrightarrow & 0101 & 110 \\
\alpha^6 & = & \alpha^2+\alpha^3 & \longrightarrow & 0011 & 010 \\
\alpha^{11} & = & \alpha+\alpha^2+\alpha^3 & \longrightarrow & 0111 & 001
\end{array}
\qquad
\begin{array}{lcllll}
 & & & \theta & & \\
1 & = & 1 & \longrightarrow & 1000 & 110 \\
\alpha^4 & = & 1+\alpha & \longrightarrow & 1100 & 101 \\
\alpha^8 & = & 1+\alpha^2 & \longrightarrow & 1010 & 001 \\
\alpha^{10} & = & 1+\alpha+\alpha^2 & \longrightarrow & 1110 & 010 \\
\alpha^{14} & = & 1+\alpha^3 & \longrightarrow & 1001 & 011 \\
\alpha^7 & = & 1+\alpha+\alpha^3 & \longrightarrow & 1101 & 000 \\
\alpha^{13} & = & 1+\alpha^2+\alpha^3 & \longrightarrow & 1011 & 100 \\
\alpha^{12} & = & 1+\alpha+\alpha^2+\alpha^3 & \longrightarrow & 1111 & 111
\end{array}$$

Nous choisissons pour $RS(15,11,5)$ le code de Reed-Solomon dont les zéros sont α, α^2, α^3 et α^4. Son polynôme générateur vaut

$$g(z) = (z-\alpha)(z-\alpha^2)(z-\alpha^3)(z-\alpha^4) = \alpha^{10} + \alpha^3 z + \alpha^6 z^2 + \alpha^{13} z^3 + z^4.$$

Soit $\mathbf{x}$ le mot de $RS(15,11,5)$ dont l'écriture polynomiale est

$$\alpha^3 + \alpha^{12} z + \alpha^3 z^2 + \alpha^8 z^3 + \alpha^7 z^6 = \left(\alpha^8 + \alpha^5 z + \alpha^7 z^2\right) g(z),$$

soit

$$\mathbf{x} = (\alpha^3, \alpha^{12}, \alpha^3, \alpha^8, 0, 0, \alpha^7, 0, \dots)$$

Considérons le mot $\Theta(\mathbf{x})$ du code concaténé de $H(7,4,3)$ et $RS(15,11,5)$ et le mot $\mathbf{y} \in \mathbb{F}_2^{105}$ défini comme suit :

$$\Theta(\mathbf{x}) = \begin{pmatrix} 0001101 \\ 1111111 \\ 0001101 \\ 1010001 \\ 0000000 \\ 0000000 \\ 1101000 \\ 0000000 \\ \vdots \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 0000001 \\ 0111111 \\ 0000100 \\ 1000001 \\ 0000000 \\ 0000000 \\ 0000000 \\ 0000000 \\ \vdots \end{pmatrix} = \Theta(\mathbf{x}) + \begin{pmatrix} 0001100 \\ 1000000 \\ 0001001 \\ 0010000 \\ 0000000 \\ 0000000 \\ 1101000 \\ 0000000 \\ \vdots \end{pmatrix}.$$

On a

$$\Phi_0(\mathbf{y}) = \Phi_1(\mathbf{y}) = \mathbf{0}.$$

En effet,

$$\Gamma_0(\mathbf{y}) = (\infty, \infty, \infty, \infty, 0, \ldots, 0) \ \text{ et } \ \Delta_H(\mathbf{0}, \Gamma_0(\mathbf{y})) = 4 \cdot \frac{1}{2} = 2 < \frac{5}{2},$$

$$\Gamma_1(\mathbf{y}) = (0, \alpha^{12}, 0, \alpha^8, 0, \ldots, 0) \ \text{ et } \ \Delta_H(\mathbf{0}, \Gamma_1(\mathbf{y})) = 2 < \frac{5}{2}.$$

et

$$D(\mathbf{y}, \mathbf{0}) = \sum_{i=1}^{15} \min(w_H(y_i), 3) = 1 + 3 + 1 + 2 = 7 < \frac{15}{2}.$$

L'algorithme $\mathcal{A}_1$ donne donc $\Psi(\mathbf{y}) = \mathbf{0}$, et pourtant

$$d_H(\mathbf{y}, \Theta(\mathbf{x})) = 9 < d_H(\mathbf{y}, \mathbf{0}) = 10.$$

□

Remarque III.4 En remplaçant la condition 2 dans l'algorithme $\mathcal{A}_1$ par

2bis. Soit r_0, l'une des valeurs de r telles que

$$d_H(\tilde{\mathbf{y}}_{r_0}, \mathbf{y}) < \frac{d_b d_e}{2}.$$

on obtient un algorithme de décodage moins performant, mais qui respecte la distance de Hamming.

Décodage au delà de la distance minimale par l'algorithme de Block-Zyablov L'algorithme de Block-Zyablov $\mathcal{A}_1$ pour le décodage d'un code concaténé d'ordre 1 décode au delà de la distance minimale pour la métrique de Hamming. Ce n'est pas le cas pour la distance D.

Il est possible de remplacer la conditionnelle de l'algorithme $\mathcal{A}_1$ par une condition moins contraignante. Par exemple, considérons l'algorithme

Algorithme $\mathcal{A}_2$ (de Block-Zyablov étendu) *Soit* $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$.

1. *Pour tout entier r, $0 \leq r \leq t$, on calcule*

$$\tilde{\mathbf{y}}_r = \Phi_r(\mathbf{y}) \in \mathcal{C} \cup \{\infty\}.$$

2. *On calcule l'ensemble*

$$R = \{r \mid d_H(\tilde{\mathbf{y}}_r, \mathbf{y}) = \min_{1 \leq i \leq t}(d_H(\tilde{\mathbf{y}}_i, \mathbf{y}))\}.$$

3. *Si $R \neq \emptyset$, l'algorithme retourne*

$$\Psi(\mathbf{y}) = \tilde{\mathbf{y}}_{r_0} \text{ pour } r_0 \in R$$

sinon l'algorithme retourne

$$\Psi(\mathbf{y}) = \infty.$$

Ici, il serait inapproprié d'utiliser D plutôt que d_H, car lorsque l'on considère l'ensemble des $t+1$ branches, on veut choisir celle qui aboutit en donnant le mot le plus proche du code pour la distance de Hamming et non pour la distance D.

Remarque III.5 On n'a pas, dans le cas de l'algorithme $\mathcal{A}_2$, d'assurance sur l'unicité des $\tilde{\mathbf{y}}_r$, il est possible que deux valeurs de r donnent des résultats différents et que ces deux mots soient à la même distance de $\mathbf{y}$. Pour conserver le déterminisme de l'algorithme, ainsi que sa linéarité, on peut par exemple choisir toujours le plus grand r, tel que $d_H(\tilde{\mathbf{y}}_r, \mathbf{y})$ soit minimal.

Proposition III.6 *L'algorithme $\mathcal{A}_2$ ne respecte ni la distance de Hamming, ni la distance D.*

preuve : Reprenons pour le contre-exemple le même code concaténé que dans la preuve de la proposition III.5 avec le même mot

$$\mathbf{x} = (\alpha^3, \alpha^{12}, \alpha^3, \alpha^8, 0, 0, \alpha^7, 0, \dots).$$

Considérons le mot $\Theta(\mathbf{x})$ du code concaténé de $H(7,4,3)$ et $RS(15,11,5)$ et le mot $\mathbf{y} \in \mathbb{F}_2^{105}$ défini comme suit :

$$\Theta(\mathbf{x}) = \begin{pmatrix} 0001101 \\ 1111111 \\ 0001101 \\ 1010001 \\ 0000000 \\ 0000000 \\ 1101000 \\ 0000000 \\ \vdots \end{pmatrix}, \quad \text{et} \quad \mathbf{y} = \begin{pmatrix} 1011101 \\ 0111111 \\ 0001111 \\ 1000000 \\ 0000000 \\ 0000000 \\ 0000000 \\ 0000000 \\ \vdots \end{pmatrix} = \Theta(\mathbf{x}) + \begin{pmatrix} 1010000 \\ 1000000 \\ 0000010 \\ 0010001 \\ 0000000 \\ 0000000 \\ 1101000 \\ 0000000 \\ \vdots \end{pmatrix}$$

On a

$$\Phi_0(\mathbf{y}) = \mathbf{0} \text{ et } \Phi_1(\mathbf{y}) = \infty.$$

En effet,

$$\Gamma_0(\mathbf{y}) = (\infty, \infty, \infty, \infty, 0, \ldots, 0) \quad \text{et} \quad \Delta_H(\mathbf{0}, \Gamma_0(\mathbf{y})) = 2 < \frac{5}{2}$$

et

$$\Gamma_1(\mathbf{y}) = (\alpha^{13}, \alpha^{12}, \alpha^3, 0, 0, \ldots, 0) \quad \text{et} \quad \Phi(\Gamma_1(\mathbf{y})) = \infty.$$

Donc $\Psi(\mathbf{y}) = \mathbf{0}$ et pourtant,

$$D(\mathbf{y}, \Theta(\mathbf{x})) = 9 < D(\mathbf{y}, \mathbf{0}) = 10$$

et

$$d_H(\mathbf{y}, \Theta(\mathbf{x})) = 9 < d_H(\mathbf{y}, \mathbf{0}) = 16.$$

□

1.4 Décodage partiel d'un code concaténé

Supposons donnés :

- $\mathcal{B} = \mathcal{B}(n_b, k_b, d_b)$ un code linéaire sur $\mathbb{F}_q$ muni d'un algorithme de décodage γ, linéaire et borné par la distance minimale d_b, on suppose de plus que cet algorithme respecte la distance de Hamming.

- $\mathcal{B}' = \mathcal{B}'(n_b, k'_b, d'_b) \subset \mathcal{B}$ le code interne.

- $\mathcal{E} = \mathcal{E}(n_e, k_e, d_e)$ sur $\mathbb{F}_{q^{k'_b}}$ le code externe est muni d'un algorithme de décodage d'erreur et d'effacement Φ borné par la distance minimale d_e.

- θ un isomorphisme $\mathbb{F}_q$-linéaire

$$\begin{array}{rcccl} \theta & : & \mathbb{F}_{q^{k'_b}} & \longrightarrow & \mathcal{B}' \\ & & x & \longmapsto & \theta(x). \end{array} \tag{III.8}$$

  On pose $\Theta = \theta^{n_e}$

- $\mathcal{C} = \Theta(\mathcal{E})$ le code concaténé de $\mathcal{B}'$ et $\mathcal{E}$.

- $\mathcal{V}$ tel que $\mathcal{B}' \oplus \mathcal{V} = \mathcal{B}$. On note π la projection de $\mathcal{B}$ sur $\mathcal{B}'$ parallélement à $\mathcal{V}$

$$\begin{array}{rcccl} \pi & : & \mathcal{B} & \longrightarrow & \mathcal{B}' \\ & & x & \longmapsto & \pi(x) \end{array} \tag{III.9}$$

1.4.1 Définition

A partir de l'algorithme γ et de π, on peut définir un algorithme de décodage d'erreur γ' de $\mathcal{B}'$, borné par d_b, par $\gamma' = \pi \circ \gamma$.

Comme pour le décodage de Block-Zyablov des codes concaténés d'ordre 1, on peut définir les algorithmes de décodage d'erreur suivants :

Pour tout nombre réel r, $0 \leq r < \frac{d_b}{2}$, on définit l'algorithme de décodage γ_r de $\mathcal{B}'$

$$\forall\, y \in \mathbb{F}_q^{n_b},\ \gamma_r(y) = \begin{cases} \gamma'(y) = \pi(\gamma(y)) & \text{si } d_H(\gamma(y), y) \leq r \\ \infty & \text{sinon} \end{cases},$$

on en déduit l'application Γ_r

$$\begin{array}{rccl} \Gamma_r\ : & \mathbb{F}_q^{n_b n_e} & \longrightarrow & (\mathbb{F}_{q^{k'_b}} \cup \{\infty\})^{n_e} \\ & \mathbf{y} = (y_1, \ldots, y_{n_e}) & \longmapsto & \Gamma_r(\mathbf{y}) = (\theta^{-1}(\gamma_r(y_1)), \ldots, \theta^{-1}(\gamma_r(y_{n_e}))) \end{array}$$

et l'algorithme de décodage de $\mathcal{C}$ défini par $\Phi_r = \Theta \circ \Phi \circ \Gamma_r$

$$\begin{array}{rccl} \Phi_r\ : & \mathbb{F}_q^{n_b n_e} & \longrightarrow & \mathcal{C} \cup \{\infty\} \\ & \mathbf{y} & \longmapsto & \Phi_r(\mathbf{y}) = \Theta(\Phi(\Gamma_r(\mathbf{y}))) \end{array}. \tag{III.10}$$

Remarque III.6 La définition des Φ_r n'est pas strictement équivalente à celle donnée dans la section 1.3. En effet dans la définition de γ_r, la condition pour que $\gamma_r(y) = \gamma'(y)$ devrait être $d_H(\gamma'(y), y) \leq r$ et non $d_H(\gamma(y), y) \leq r$.

Notons également que si l'on change de projection $\pi : \mathcal{B} \to \mathcal{B}'$, on modifie la définition du décodage.

Ces différences n'altèrent pas, comme nous le verrons, les propriétés de l'algorithme.

Proposition III.7 *Soit $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$. Pour tout entier r, $0 \leq r \leq t = \lfloor \frac{d_b - 1}{2} \rfloor$, on pose*

$$\tilde{\mathbf{y}}_r = \Phi_r(\mathbf{y}) \in \mathcal{C} \cup \{\infty\}.$$

Alors tous les r pour lesquels $d_H(\tilde{\mathbf{y}}_r, \mathbf{y}) < \frac{d_b d_e}{2}$ (par convention $d_H(\infty, \mathbf{y}) = \infty$), donnent la même valeur $\tilde{\mathbf{y}}$ de $\tilde{\mathbf{y}}_r$.

On peut alors définir un décodeur Ψ de $\mathcal{C}$

$$\forall\, \mathbf{y} \in \mathbb{F}_q^{n_b n_e},\ \Psi(\mathbf{y}) = \begin{cases} \tilde{\mathbf{y}}_r & si\ \exists\, r,\ d_H(\tilde{\mathbf{y}}_r, \mathbf{y}) < \frac{d_b d_e}{2} \\ \infty & sinon \end{cases}. \tag{III.11}$$

preuve : La preuve est identique à celle de la proposition III.4. □

Définition III.3 *L'algorithme de décodage Ψ donné par la proposition précédente sera appelé* algorithme de décodage partiel de $\mathcal{C}$ par rapport à $\mathcal{B}$.

$$
\begin{tikzcd}
(\mathcal{B} \cup \{\infty\})^{n_e} \arrow[r, "\pi^{n_e}"] & (\mathcal{B}' \cup \{\infty\})^{n_e} \arrow[d, "(\theta^{-1})^{n_e}"] \\
\mathbb{F}_q^{n_b n_e} \equiv (\mathbb{F}_q^{n_b})^{n_e} \arrow[u, "\gamma^{n_e}"'] \arrow[ur, "{\gamma_r}^{n_e}"] \arrow[r, "\Gamma_r"] \arrow[d, "\Phi_r"] & (\mathbb{F}_{q^{k'_b}} \cup \{\infty\})^{n_e} \arrow[d, "\Phi"] \\
\mathcal{C} \cup \{\infty\} & \mathcal{E} \cup \{\infty\} \arrow[l, "\Theta"']
\end{tikzcd}
$$

Figure III.2 : Décodage partiel d'un code concaténé

Remarque III.7 Nous ne donnons pas à l'algorithme Ψ défini dans la proposition III.7, le nom de décodeur partiel de Block-Zyablov de $\mathcal{C}$, nous préférons réserver cette appellation à un algorithme plus général qui sera décrit dans la proposition III.10.

Théorème III.2 *Soit $\mathcal{C}$ le code concaténé de $\mathcal{B}' = \mathcal{B}'(n_b, k'_b, d'_b)$ et $\mathcal{E} = \mathcal{E}(n_e, k_e, d_e)$ et soit $\mathcal{B} = \mathcal{B}(n_b, k_b, d_b)$ un sur-code de $\mathcal{B}'$. L'algorithme de décodage partiel Ψ de $\mathcal{C}$ par rapport à $\mathcal{B}$ est borné par $d_b d_e$.*

Autrement dit, pour tout $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$ et pour tout $\mathbf{x} \in \mathcal{C}$, on a

$$d_H(\mathbf{x}, \mathbf{y}) < \frac{d_b d_e}{2} \Rightarrow \Psi(\mathbf{y}) = \mathbf{x}.$$

preuve : La preuve est identique à la celle du théorème III.1, en remplaçant $\mathcal{B}$ par $\mathcal{B}'$ et γ par γ', exception faite de la propriété (III.6)

$$\forall\, r \text{ réel},\ 0 \leq r < \frac{d_b}{2},\quad \sum_{i=1}^{n_e} E(w_i, r) \geq 2\Delta_H(\Theta^{-1}(\mathbf{x}), \Gamma_r(\mathbf{y})). \tag{III.6\textit{bis}}$$

qui reste vraie, mais pour laquelle il nous faut donner une justification différente :

Pour tout i distinguons deux cas :

1. $\delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma_r(y_i))) = 1/2$: dans ce cas $\gamma_r(y_i) = \infty$ car $x_i \neq \infty$, donc $w_i = d_H(x_i, y_i) > r$, sinon on aurait $\gamma_r(y_i) = x_i \neq \infty$. On en conclut que $E(w_i, r) \geq 1$, par définition de E.

2. $\delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma_r(y_i))) = 1$: dans ce cas on a $\gamma_r(y_i) = \gamma'(y_i) = \pi(\gamma(y_i)) \in \mathcal{B}$ et $\gamma_r(y_i) \neq x_i$, donc également $\gamma(y_i) \in \mathcal{B}$ et $\gamma(y_i) \neq x_i$, et par l'inégalité triangulaire

$$w_i = d_H(y_i, x_i) \geq d_H(\gamma(y_i), x_i) - d_H(y_i, \gamma(y_i)) \geq d_b - r.$$

En effet $\gamma(y_i) \in \mathcal{B}$, et $x_i \in \mathcal{B}$ étant distincts, on a $d_H(\gamma(y_i), x_i) \geq d_b$ et $d_H(y_i, \gamma(y_i)) \leq r$, par définition de γ_r. Donc $E(w_i, r) = 2$ par définition de E.

Dans ces deux cas on a donc toujours

$$E(w_i, r) \geq 2\delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma_r(y_i))),$$

ce qui entraine (III.6*bis*) puisque par définition

$$\Delta_H(\Theta^{-1}(\mathbf{x}), \Gamma_r(\mathbf{y})) = \sum_{i=1}^{n_e} \delta_H(\theta^{-1}(x_i), \theta^{-1}(\gamma_r(y_i))).$$

□

1.4.2 Décodage partiel de Block-Zyablov

Nous désirons un algorithme plus général que celui exposé ci-dessus. Si Ψ est un algorithme de décodage partiel de $\mathcal{C}$ par rapport à $\mathcal{B}$, nous voulons que Ψ vérifie la propriété supplémentaire

$$\forall\, \mathbf{y} \in \mathbb{F}_q^{n_b n_e},\ \forall\, \mathbf{x} \in \mathcal{C},\ \ \forall\, \mathbf{v} \in \mathcal{V}^{n_e},\ d_H(\mathbf{x}+\mathbf{v}, \mathbf{y}) < \frac{d_b d_e}{2} \Rightarrow \Psi(\mathbf{y}) = \mathbf{x}. \tag{III.12}$$

Proposition III.8 *Soit $\mathcal{C}$ le code concaténé de $\mathcal{B}'$ et $\mathcal{E}$. Si pour tout nombre réel r, $0 \le r < \frac{d_b}{2}$, Φ_r est défini par* (III.10)*, alors on a*

$$\forall\, \mathbf{y} \in \mathbb{F}_q^{n_b n_e},\ \forall\, \mathbf{v} \in \mathcal{V}^{n_e},\ \Phi_r(\mathbf{y}+\mathbf{v}) = \Phi_r(\mathbf{y}).$$

preuve : Comme $\mathcal{V} \subset \mathcal{B}$, et γ est linéaire, on peut écrire

$$\forall\, y \in \mathbb{F}_q^{n_b},\ \forall\, v \in \mathcal{V},\ \gamma(y+v) = \gamma(y) + v,$$

et

$$\forall\, y \in \mathbb{F}_q^{n_b},\ \forall\, v \in \mathcal{V},\ \gamma'(y+v) = \pi(\gamma(y+v)) = \pi(\gamma(y)+v) = \gamma'(y).$$

On en déduit immédiatement pour tout r

$$\forall\, y \in \mathbb{F}_q^{n_b},\ \forall\, v \in \mathcal{V},\ \gamma_r(y+v) = \gamma_r(y)$$

puisque $d_H(\gamma(y+v), y+v) = d_H(\gamma(y), y)$ Le résultat s'ensuit. □

Dans la proposition III.7 qui décrit le décodage partiel d'un code concaténé, on est amené à calculer pour tout entier r, $0 \le r \le t$, le vecteur $\tilde{\mathbf{y}}_r = \Phi_r(\mathbf{y})$. Si $d_H(\mathbf{x}, \mathbf{y}) < \frac{d_b d_e}{2}$, alors il existe une valeur de r telle que $\tilde{\mathbf{y}}_r = \mathbf{x}$. Pour trouver ce r, il suffit de vérifier que $d_H(\tilde{\mathbf{y}}_r, \mathbf{y}) < \frac{d_b d_e}{2}$.

Si on suppose que l'on a seulement $d_H(\mathbf{x}+\mathbf{v}, \mathbf{y}) < \frac{d_b d_e}{2}$ pour un certain $\mathbf{v} \in \mathcal{V}^{n_e}$, alors, d'après la proposition III.8, il existe une branche r de l'algorithme telle que $\tilde{\mathbf{y}}_r = \mathbf{x}$. En effet

$$d_H(\mathbf{x}, \mathbf{y}-\mathbf{v}) < \frac{d_b d_e}{2} \ \Rightarrow\ \exists\, r_0,\ \Phi_{r_0}(\mathbf{y}-\mathbf{v}) = \mathbf{x},$$

et $\Phi_{r_0}(\mathbf{y}) = \mathbf{x}$ puisque $\Phi_{r_0}(\mathbf{y}-\mathbf{v}) = \Phi_{r_0}(\mathbf{y})$.

Il n'est pourtant pas certain que $d_H(\tilde{\mathbf{y}}_{r_0}, \mathbf{y}) < \frac{d_b d_e}{2}$. Si l'on prend par exemple $\mathbf{v} = (v_1, v_2, \ldots, v_{n_e})$, avec $v_i \neq 0$ pour au moins d_e coordonnées, alors $w_H(\mathbf{v}) \ge d_b d_e$ et on a

$$d_H(\tilde{\mathbf{y}}_{r_0}, \mathbf{y}) \ge d_H(\tilde{\mathbf{y}}_{r_0}, \tilde{\mathbf{y}}_{r_0}+\mathbf{v}) - d_H(\mathbf{y}, \tilde{\mathbf{y}}_{r_0}+\mathbf{v}) > w_H(\mathbf{v}) - \frac{d_b d_e}{2}$$

donc

$$d_H(\tilde{\mathbf{y}}_{r_0}, \mathbf{y}) > \frac{d_b d_e}{2}.$$

Il faut donc trouver un critère autre que la distance de Hamming pour repérer la bonne branche.

Soit $\mathbf{y} = (y_1, \ldots, y_{n_e}) \in \mathbb{F}_q^{n_b n_e}$, on pose

$$\forall\, i,\ w_i = \begin{cases} \frac{d_b}{2} & \text{si } \gamma(y_i) = \infty \\ \min(\frac{d_b}{2}, d_H(y_i, \gamma(y_i))) & \text{sinon} \end{cases}.$$

Pour tout $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathcal{B}'^{n_e}$, on pose

$$\forall\, i,\ W_i(x_i) = \begin{cases} w_i & \text{si } x_i = \gamma'(y_i) = \pi(\gamma(y_i)) \\ d_b - w_i & \text{sinon} \end{cases}$$

et

$$W_{\mathbf{y}}(\mathbf{x}) = \sum_{i=1}^{n_e} W_i(x_i).$$

Proposition III.9

$$\forall\, \mathbf{y} \in \mathbb{F}_q^{n_b n_e},\ \forall\, \mathbf{x} \in \mathcal{C},\ \forall\, \mathbf{v} \in \mathcal{V}^{n_e},\quad W_{\mathbf{y}}(\mathbf{x}) \le d_H(\mathbf{y}, \mathbf{x} + \mathbf{v}).$$

preuve : Soient

$$\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathcal{C},\quad \mathbf{v} = (v_1, \ldots, v_{n_e}) \in \mathcal{V}^{n_e},\quad \mathbf{y} = (y_1, \ldots, y_{n_e}) \in \mathbb{F}_q^{n_b n_e},$$

tels que

$$d_H(\mathbf{y}, \mathbf{x} + \mathbf{v}) < \frac{d_b d_e}{2}.$$

Posons

$$\mathbf{z} = (z_1, \ldots, z_{n_e}) = \mathbf{x} + \mathbf{v},$$

et notons que pour tout i, $x_i, v_i \in \mathcal{B}$, donc $z_i \in \mathcal{B}$.

Pour tout i, on a $W_i(x_i) \le d_H(z_i, y_i)$, en effet

- si $x_i = \gamma'(y_i)$ (donc $W_i(x_i) = w_i$)
  - si $z_i = \gamma(y_i)$, alors
  $$d_H(z_i, y_i) = d_H(\gamma(y_i), y_i) \ge w_i = W_i(x_i)$$
  - si $z_i \ne \gamma(y_i) \in \mathcal{B}$
  $$d_H(z_i, y_i) \ge d_H(\gamma(y_i), y_i) \ge w_i = W_i(x_i)$$
  car γ respecte la distance de Hamming.
- sinon $x_i \ne \gamma'(y_i)$, donc $\gamma(y_i) \ne z_i$ (et $W_i(x_i) = d_b - w_i$)

– si $\gamma(y_i) = \infty$, alors $w_i = \frac{d_b}{2}$, et donc puisque γ est borné par d_b

$$d_H(z_i, y_i) \geq \frac{d_b}{2} = d_b - w_i = W_i(x_i)$$

– si $z_i \neq \gamma(y_i) \in \mathcal{B}$ et $w_i = \frac{d_b}{2}$, γ est borné par d_b, donc

$$d_H(z_i, y_i) \geq \frac{d_b}{2} = d_b - w_i = W_i(x_i)$$

– si $z_i \neq \gamma(y_i) \in \mathcal{B}$ et $w_i = d_H(y_i, \gamma(y_i))$, par l'inégalité triangulaire, on a

$$d_H(z_i, y_i) \geq d_H(z_i, \gamma(y_i)) - d_H(y_i, \gamma(y_i)) \geq d_b - w_i = W_i(x_i)$$

donc

$$W_{\mathbf{y}}(\mathbf{x}) = \sum_{i=1}^{n_e} W_i(x_i) \leq d_H(\mathbf{z}, \mathbf{y}).$$

$\square$

Proposition III.10 *Soit* $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$. *Pour tout entier* r, $0 \leq r \leq t = \lfloor \frac{d_b - 1}{2} \rfloor$, *posons*

$$\tilde{\mathbf{y}}_r = \Phi_r(\mathbf{y}) \in \mathcal{C} \cup \{\infty\}.$$

Alors tous les r *pour lesquels*

$$W_{\mathbf{y}}(\tilde{\mathbf{y}}_r) < \frac{d_b d_e}{2}$$

donnent la même valeur $\tilde{\mathbf{y}}$ *de* $\tilde{\mathbf{y}}_r$, *on pose alors*

$$\Psi(\mathbf{y}) = \tilde{\mathbf{y}},$$

et si aucun r *ne vérifie* $W_{\mathbf{y}}(\tilde{\mathbf{y}}_r) < \frac{d_b d_e}{2}$, *on pose*

$$\Psi(\mathbf{y}) = \infty.$$

On dira que Ψ *est un décodeur partiel de Block-Zyablov de* $\mathcal{C}$ *par rapport à* $\mathcal{B}$.

preuve : Soient $\mathbf{x} = (x_1, \ldots, x_{n_e})$ et $\mathbf{x}' = (x'_1, \ldots, x'_{n_e})$ deux éléments distincts de $\mathcal{C}$, soit $J \subset [1, n_e]$, l'ensemble des positions pour lesquelles $\Theta^{-1}(\mathbf{x}) \in \mathcal{E}$ et $\Theta^{-1}(\mathbf{x}') \in \mathcal{E}$ diffèrent,

$$\forall\, i \in J,\ x_i \neq x'_i.$$

Le cardinal de J est au moins égal à d_e, la distance minimale de $\mathcal{E}$. Pour $i \in J$, on ne peut pas avoir simultanément $W_i(x_i) = w_i$ et $W_i(x'_i) = w_i$, sinon $x_i = \gamma'(y_i) = x'_i$, donc

$$\forall\, i \in J,\ W_i(x_i) + W_i(x'_i) \in \{w_i + (d_b - w_i), 2(d_b - w_i)\},$$

et donc

$$\forall\, i \in J,\ W_i(x_i) + W_i(x'_i) \geq d_b,$$

donc en sommant sur J, on a

$$W_{\mathbf{y}}(\mathbf{x}) + W_{\mathbf{y}}(\mathbf{x}') \geq d_b d_e.$$

Ces deux nombres ne peuvent donc pas être simultanément strictement inférieur à $\frac{d_b d_e}{2}$, d'où le résultat. □

L'algorithme de décodage peut se décrire comme suit.

Algorithme $\mathcal{A}_3$ *Soit* $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$.

1. *Pour tout entier* r, $0 \leq r \leq t$, *on calcule*

$$\tilde{\mathbf{y}}_r = \Phi_r(\mathbf{y}) \in \mathcal{C} \cup \{\infty\}.$$

2. *On calcule l'ensemble*

$$R = \{r \mid W_{\mathbf{y}}(\tilde{\mathbf{y}}_r) < \frac{d_b d_e}{2}\}.$$

3. *Si* $R \neq \emptyset$, *l'algorithme retourne*

$$\Psi(\mathbf{y}) = \tilde{\mathbf{y}}_{r_0} \textit{ pour } r_0 \in R,$$

sinon l'algorithme retourne

$$\Psi(\mathbf{y}) = \infty.$$

Proposition III.11 *Soit* Ψ *défini par l'algorithme* $\mathcal{A}_3$, *on a*

$$\forall\, \mathbf{y} \in \mathbb{F}_q^{n_b n_e},\ \forall\, \mathbf{x} \in \mathcal{C},\quad \forall\, \mathbf{v} \in \mathcal{V}^{n_e},\ d_H(\mathbf{x}+\mathbf{v}, \mathbf{y}) < \frac{d_b d_e}{2} \Rightarrow \Psi(\mathbf{y}) = \mathbf{x}.$$

preuve : Soient

$$\mathbf{y} \in \mathbb{F}_q^{n_b n_e},\ \mathbf{x} \in \mathcal{C},\ \mathbf{v} \in \mathcal{V}^{n_e},$$

tels que

$$d_H(\mathbf{x}+\mathbf{v}, \mathbf{y}) < \frac{d_b d_e}{2}.$$

D'après la proposition III.9,

$$W_{\mathbf{y}}(\mathbf{x}) \leq d_H(\mathbf{x}+\mathbf{v}, \mathbf{y}) < \frac{d_b d_e}{2}$$

et d'après le théorème III.2, on a

$$\Psi(\mathbf{y}-\mathbf{v}) = \mathbf{x},$$

donc il existe r_0 tel que

$$\Phi_{r_0}(\mathbf{y} - \mathbf{v}) = \mathbf{x}.$$

D'après la proposition III.8, on a donc

$$\Phi_{r_0}(\mathbf{y}) = \mathbf{x}$$

et enfin, d'après la proposition III.10

$$\Psi(\mathbf{y}) = \Phi_{r_0}(\mathbf{y}) = \mathbf{x}.$$

□

Proposition III.12 *L'algorithme $\mathcal{A}_3$ ne respecte pas la distance de Hamming.*

preuve : Pour construire un contre-exemple nous considérons le cas trivial $\mathcal{B} = \mathcal{B}'$. Le code concaténé est celui de la preuve de la proposition III.5. Soit le mot de $RS(15, 11, 5)$:

$$\mathbf{x} = (\alpha^{10}, \alpha^3, \alpha^6, \alpha^{13}, 1, 0, \dots)$$

Considérons le mot $\Theta(\mathbf{x})$ du code concaténé de $H(7, 4, 3)$ et $RS(15, 11, 5)$, et le mot $\mathbf{y} \in \mathbb{F}_2^{105}$ définit comme suit :

$$\Theta(\mathbf{x}) = \begin{pmatrix} 1110010 \\ 0001101 \\ 0011010 \\ 1011100 \\ 1000110 \\ 0000000 \\ \vdots \end{pmatrix}, \quad \text{et} \quad \mathbf{y} = \begin{pmatrix} 1110011 \\ 0000001 \\ 0001000 \\ 1111100 \\ 0000000 \\ 0000000 \\ \vdots \end{pmatrix} = \Theta(\mathbf{x}) + \begin{pmatrix} 0000001 \\ 0001100 \\ 0010010 \\ 0100000 \\ 1000110 \\ 0000000 \\ \vdots \end{pmatrix}$$

On a

$$\Phi_0(\mathbf{y}) = \Phi_1(\mathbf{y}) = \mathbf{0},$$

en effet,

$$\Gamma_0(\mathbf{y}) = (\infty, \infty, \infty, \infty, 0, 0, \dots, 0) \ \text{ et } \ \Delta_H(\mathbf{0}, \Gamma_0(\mathbf{y})) = 2 < \frac{5}{2},$$

$$\Gamma_1(\mathbf{y}) = (\alpha^{10}, 0, 0, \alpha^{13}, 0, 0, \dots, 0) \ \text{ et } \ \Delta_H(\mathbf{0}, \Gamma_1(\mathbf{y})) = 2 < \frac{5}{2}.$$

Donc $\Psi(\mathbf{y}) = \mathbf{0}$ et pourtant

$$W_1(0) = 3 - 1 = 2, \ W_2(0) = 1, \ W_3(0) = 1, \ W_4(0) = 3 - 1 = 2, \ \forall\, i > 4, \ W_i(0) = 0,$$

donc

$$W_{\mathbf{y}}(\mathbf{0}) = 6 < \frac{15}{2}$$

et

$$d_H(\mathbf{y}, \Theta(\mathbf{x})) = 9 < d_H(\mathbf{y}, \mathbf{0}) = 12.$$

□

2 Evaluation des performances des codes concaténés d'ordre 1

- Soit $\mathcal{C}$ le code concaténé admettant comme code interne $\mathcal{B}(n_b, k_b, d_b)$ sur $\mathbb{F}_q$, et comme code externe $\mathcal{E}(n_e, k_e, d_e)$ sur $\mathbb{F}_{q^{k_b}}$.
- Soit γ un algorithme de décodage d'erreur de $\mathcal{B}$, soit Φ un algorithme de décodage d'erreur et d'effacement de $\mathcal{E}$, ces deux algorithmes étant bornés par la distance minimale.
- Soient $P_0(x)$, $P_1(x)$, et $P_2(x)$, les polynômes des motifs corrigibles, des motifs non corrigibles et des motifs erronés de $\mathcal{B}$ pour l'algorithme γ.
- Soit Δ_H la distance de Hamming généralisée sur $\mathbb{F}_{q^{k_b}} \cup \{\infty\}$, Ω_H le poids de Hamming généralisé sur $\mathbb{F}_{q^{k_b}} \cup \{\infty\}$, d_H la distance de Hamming sur $\mathbb{F}_q$, et w_H le poids de Hamming sur $\mathbb{F}_q$.

2.1 Distribution des poids des codes concaténés d'ordre 1

Lemme III.2 *Pour tout entier r, $1 \leq r \leq n_e - d_e$, on a*

$$\sum_{w=d_b d_e}^{d_b(d_e+r)-1} A_w(\mathcal{C}) \leq \sum_{i=d_e}^{d_e+r-1} A_i(\mathcal{E}).$$

preuve : Soit $\mathbf{c} \in \mathcal{C}$ de poids w strictement inférieur à $d_b(d_e+r)$. Le vecteur $\Theta^{-1}(\mathbf{c})$ possède au plus $d_e + r - 1$ coordonnées non nulles, dans le cas contraire puisque $\mathcal{B}$ a pour distance minimale d_b, on aurait $w \geq d_b(d_e + r)$, d'où l'inégalité. $\square$

Proposition III.13 *Soit un entier w tel que $d_b d_e \leq w \leq n_b n_e$. On a*

$$A_w(\mathcal{C}) \leq \sum_{i=d_e}^{\lfloor \frac{w}{d_b} \rfloor} A_i(\mathcal{E}).$$

preuve : Notons d'abord que pour $w \geq d_b n_e$, l'inégalité est triviale puisqu'on a alors $A_w(\mathcal{C}) \leq |\mathcal{E}| = |\mathcal{C}|$.

Si $w < d_b n_e$, on pose $r = \lfloor \frac{w}{d_b} \rfloor - d_e + 1$. On a alors $d_b d_e \leq w < d_b(d_e + r)$ et en appliquant le lemme III.2 pour cette valeur de r on obtient

$$\sum_{j=d_b d_e}^{d_b(d_e+r)-1} A_j(\mathcal{C}) \leq \sum_{i=d_e}^{d_e+r-1} A_i(\mathcal{E}) = \sum_{i=d_e}^{\lfloor \frac{w}{d_b} \rfloor} A_i(\mathcal{E}),$$

donc en particulier

$$A_w(\mathcal{C}) \leq \sum_{i=d_e}^{\lfloor \frac{w}{d_b} \rfloor} A_i(\mathcal{E}).$$

$\square$

2.2 Evaluation de l'algorithme naïf

Soit $\Psi = \Phi \circ \Gamma$ l'algorithme de décodage d'erreur de $\mathcal{C}$ présenté dans la section 1.2 de ce chapitre, où $\Gamma : (m_1, \ldots, m_{n_e}) \mapsto (\gamma(m_1), \ldots, \gamma(m_{n_e}))$.

Lemme III.3

$$\sum_{\mathbf{m} \in \mathbb{F}_q^{n_b n_e}} x^{w_H(\mathbf{m})} y^{2\Omega_H(\Gamma(\mathbf{m}))} = (P_0(x) + P_1(x)y + P_2(x)y^2)^{n_e}. \tag{III.13}$$

preuve : Soit $\mu : \mathbb{F}_{q^{k_b}} \cup \{\infty\} \to \{0, \frac{1}{2}, 1\}$ l'application définie par

$$\mu(a) = \begin{cases} 0 & \text{si } a = 0 \\ \frac{1}{2} & \text{si } a = \infty \\ 1 & \text{si } a \neq 0, \text{ et } a \neq \infty. \end{cases}$$

Dans la notation du chapitre I, $\mu(a) = \delta_H(a, 0)$.

Pour $\mathbf{a} = (a_1, \ldots, a_{n_e}) \in (\mathbb{F}_{q^{k_b}} \cup \{\infty\})^{n_e}$, on a

$$\Omega_H(\mathbf{a}) = \sum_{i=1}^{n} \mu(a_i).$$

Si l'on revient aux définitions des polynômes énumérateurs des motifs d'erreur pour le code interne, on a

$$\begin{aligned} P_0(x) &= \sum_{m \in \mathbb{F}_q^{n_e},\ \gamma(m)=0} x^{w_H(m)}, \\ P_1(x) &= \sum_{m \in \mathbb{F}_q^{n_e},\ \gamma(m)=\infty} x^{w_H(m)}, \\ P_2(x) &= \sum_{m \in \mathbb{F}_q^{n_e},\ \gamma(m) \in \mathcal{B} \setminus \{0\}} x^{w_H(m)} \end{aligned}$$

et ces trois polynômes énumèrent tous les motifs possibles. On a donc

$$\sum_{m \in \mathbb{F}_q^{n_b}} x^{w_H(m)} y^{2\mu(\gamma(m))} = P_0(x) + P_1(x)y + P_2(x)y^2 n$$

et en élevant à la puissance n_e le premier terme de cette égalité, on obtient

$$\Big(\sum_{m \in \mathbb{F}_q^{n_b}} x^{w_H(m)} y^{2\mu(\gamma(m))} \Big)^{n_e} = \sum_{(m_1, \ldots, m_{n_e}) \in (\mathbb{F}_q^{n_b})^{n_e}} x^{\sum_{i=1}^{n_e} w_H(m_i)} y^{2 \sum_{i=1}^{n_e} \mu(\gamma(m_i))}.$$

Si on pose $\mathbf{m} = (m_1, \ldots, m_{n_e})$, on a

$$w_H(\mathbf{m}) = \sum_{i=1}^{n_e} w_H(m_i) \quad \text{et} \quad \Omega_H(\Gamma(\mathbf{m})) = \sum_{i=1}^{n_e} \mu(\gamma(m_i)),$$

ce qui achève la démonstration de (III.13). $\square$

Proposition III.14 *Soit le polynôme $Q(x,y) \in \mathbb{Z}[x,y]$ défini par*

$$Q(x,y) = (P_0(x) + P_1(x)y + P_2(x)y^2)^{n_e}.$$

Si Φ est borné strictement *par d_e, alors*

$$P_{\mathcal{C}}(x) = \sum_{i=0}^{d_e-1} [y^i]Q(x,y)$$

est le polynôme des motifs corrigibles de $\mathcal{C}$ pour l'algorithme Ψ.

preuve : D'après le lemme III.3, on a

$$Q(x,y) = \sum_{\mathbf{m} \in \mathbb{F}_q^{n_b n_e}} x^{w_H(\mathbf{m})} y^{2\Omega_H(\Gamma(\mathbf{m}))}$$

et puisque Φ est borné strictement par d_e, on peut écrire

$$\Psi(\mathbf{m}) = \Phi(\Gamma(\mathbf{m})) = \mathbf{0} \Leftrightarrow 2\Omega_H(\Gamma(\mathbf{m})) < d_e,$$

donc

$$\sum_{i=0}^{d_e-1} [y^i]Q(x,y) = \sum_{\mathbf{m} \in \mathbb{F}_q^{n_b n_e}, \Psi(\mathbf{m})=\mathbf{0}} x^{w_H(\mathbf{m})},$$

qui est exactement la définition du polynôme des motifs corrigibles. □

2.3 Evaluation de l'algorithme de Block-Zyablov

Soit Ψ un décodeur de Block-Zyablov (algorithme $\mathcal{A}_1$, page 72) de $\mathcal{C}$.

Nous rappelons que pour tout $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathbb{F}_q^{n_b n_e}$ et tout $\mathbf{y} = (y_1, \ldots, y_{n_e}) \in \mathbb{F}_q^{n_b n_e}$, on définit $D(\mathbf{x}, \mathbf{y})$ par

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n_e} \min(d_H(x_i, y_i), d_b).$$

Cette définition dépend de la distance minimale d_b du code interne.

Lemme III.4 *Soit $P_{\mathcal{C}}(x)$ le polynôme des motifs corrigibles de $\mathcal{C}$ pour l'algorithme de Block-Zyablov, on a*

$$P_{\mathcal{C}}(x) = \sum_{\mathbf{m} \in \mathbb{F}_q^{n_b n_e}, D(\mathbf{m}, \mathbf{0}) < \frac{d_b d_e}{2}} x^{w_H(\mathbf{m})}$$

preuve : L'algorithme de Block-Zyablov est borné strictement par $d_b d_e$ pour la distance D, donc

$$\forall\, \mathbf{m} \in \mathbb{F}_q^{n_b n_e},\ \Psi(\mathbf{m}) = 0 \Leftrightarrow D(\mathbf{m}, \mathbf{0}) < \frac{d_b d_e}{2},$$

d'où le résultat. □

Proposition III.15 *On pose*

$$Q(x,y) = \left(\sum_{m \in \mathbb{F}_q^{n_b}} x^{w_H(m)} y^{\min(w_H(m),d_b)} \right)^{n_e}.$$

Le polynôme des motifs corrigibles de $\mathcal{C}$ *pour l'algorithme* $\mathcal{A}_1$ *de Block-Zyablov est égal à*

$$P_{\mathcal{C}}(x) = \sum_{i=0}^{\frac{d_b d_e}{2}} [y^i] Q(x,y).$$

preuve :

$$\left(\sum_{m \in \mathbb{F}_q^{n_b}} x^{w_H(m)} y^{\min(w_H(m),d_b)} \right)^{n_e} = \sum_{(m_1,\ldots,m_{n_e}) \in \mathbb{F}_q^{n_b n_e}} x^{\sum_{i=1}^{n_e} w_H(m_i)} y^{\sum_{i=1}^{n_e} \min(w_H(m_i),d_b)},$$

donc

$$Q(x,y) = \sum_{\mathbf{m} \in \mathbb{F}_q^{n_b n_e}} x^{w_H(\mathbf{m})} y^{D(\mathbf{m},\mathbf{0})}.$$

En utilisant le lemme III.4, on obtient le résultat. □

2.4 Evaluation de l'algorithme de Block-Zyablov étendu

Soit Ψ un décodeur de Block-Zyablov étendu (algorithme $\mathcal{A}_2$, page 75) de $\mathcal{C}$. On pose $t = \lfloor \frac{d_b}{2} \rfloor$, rappelons que pour décrire Ψ on définit pour tout entier r, $0 \leq r \leq t$,

- un algorithme de décodage de $\mathcal{B}$

$$\forall\, m \in \mathbb{F}_q^{n_b},\ \gamma_r(m) = \begin{cases} \gamma(m) & \text{si } d_H(\gamma(m),m) \leq r \\ \infty & \text{sinon} \end{cases},$$

- une application Γ_r

$$\begin{array}{cccc} \Gamma_r : & \mathbb{F}_q^{n_b n_e} & \longrightarrow & (\mathbb{F}_{q^{k_b}} \cup \{\infty\})^{n_e} \\ & \mathbf{m} = (m_1,\ldots,m_{n_e}) & \longmapsto & \Gamma_r(\mathbf{m}) = (\theta^{-1}(\gamma_r(m_1)),\ldots,\theta^{-1}(\gamma_r(m_{n_e}))) \end{array}$$

  où θ est un isomorphisme fixé de $\mathbb{F}_{q^{k_b}}$ sur $\mathcal{B}$,

- et un algorithme de décodage $\Phi_r = \Theta \circ \Phi \circ \Gamma_r$ du code concaténé $\mathcal{C}$

$$\begin{array}{cccc} \Phi_r : & \mathbb{F}_q^{n_b n_e} & \longrightarrow & \mathcal{E} \cup \{\infty\} \\ & \mathbf{m} & \longmapsto & \Phi_r(\mathbf{m}) = \Theta(\Phi(\Gamma_r(\mathbf{m}))). \end{array}$$

Pour tout entier r, $0 \leq r \leq n_e$, on notera

$$P_{0,r}(x) = [x^r]P_0(x)x^r$$

le monôme de degré r de $P_0(x)$, c'est-à-dire le polynôme énumérateur des motifs corrigibles de poids r, et

$$P_{2,r}(x) = \sum_{m \in \mathbb{F}_q^{n_b},\ \gamma(m) \in C \setminus \{0\},\ d_H(m,\gamma(m))=r} x^{w_H(m)}$$

le polynôme énumérateur des motifs d'erreur qui sont corrigés de façon erronée en exactement r positions.

Le polynôme

$$\sum_{i=0}^{r} P_{0,i}(x)$$

est le polynôme des motifs corrigibles de $\mathcal{B}$ pour γ_r, le polynôme

$$\sum_{i=0}^{r} P_{2,i}(x)$$

est le polynôme des motifs erronés de $\mathcal{B}$ pour γ_r et le polynôme

$$P_1(x) + \sum_{i=r+1}^{n_e} (P_{2,i}(x) + P_{0,i}(x))$$

est le polynôme des motifs non corrigibles de $\mathcal{B}$ pour γ_r.

Remarque III.8 Notons que l'on a

$$\sum_{i=0}^{n_b} (P_{0,i}(x) + P_{2,i}(x)) + P_1(x) = (1 + (q-1))^{n_b}$$

c'est-à-dire que tous les motifs sont énumérés par ces polynômes une et une seule fois.

Lemme III.5

$$\sum_{\mathbf{m} \in \mathbb{F}_q^{n_b n_e}} x^{w_H(\mathbf{m})} \prod_{j=0}^{t} y_j^{2\Omega_H(\Gamma_j(\mathbf{m}))} =$$
$$\left(\sum_{i=0}^{t} P_{0,i}(x) \prod_{j=0}^{i-1} y_j + P_1(x) \prod_{j=0}^{t} y_j + \sum_{i=0}^{t} P_{2,i}(x) \prod_{j=0}^{i-1} y_j \prod_{j=i}^{t} y_j^2 \right)^{n_e} \tag{III.14}$$

preuve : On pose

$$P(x, y_0, \ldots, y_t) = \sum_{i=0}^{t} P_{0,i}(x) \prod_{j=0}^{i-1} y_j + P_1(x) \prod_{j=0}^{t} y_j + \sum_{i=0}^{t} P_{2,i}(x) \prod_{j=0}^{i-1} y_j \prod_{j=i}^{t} y_j^2.$$

Pour tout motif $m \in \mathbb{F}_q^{n_b}$, on a

- m est corrigible pour γ_i si et seulement si $2\mu(\gamma_i(m)) = 0$,
- m est non corrigible pour γ_i si et seulement si $2\mu(\gamma_i(m)) = 1$,
- m est erroné pour γ_i si et seulement si $2\mu(\gamma_i(m)) = 2$.

Par définition on a

- pour tout i, $0 \leq i \leq t$, $P_{0,i}(x)$ énumère des motifs qui sont corrigibles pour $\gamma_i, \ldots, \gamma_t$, et non corrigibles pour $\gamma_0, \ldots, \gamma_{i-1}$, c'est-à-dire des motifs tels que $2\mu(\gamma_j(m)) = 0$ pour $i \leq j \leq t$, et $2\mu(\gamma_j(m)) = 1$ pour $0 \leq j \leq i-1$,
- pour tout i, $0 \leq i \leq t$, $P_{2,i}(x)$ énumère des motifs non corrigibles pour $\gamma_0, \ldots, \gamma_{i-1}$, et erronés pour $\gamma_i, \ldots, \gamma_t$, c'est-à-dire des motifs tels que $2\mu(\gamma_j(m)) = 1$ pour $0 \leq j \leq i-1$, et $2\mu(\gamma_j(m)) = 2$ pour $i \leq j \leq t$,
- $P_1(x)$ énumère des motifs non corrigibles pour $\gamma_0, \ldots, \gamma_t$, c'est-à-dire des motifs tels que $2\mu(\gamma_j(m)) = 1$ pour $0 \leq j \leq t$.

D'après la remarque III.8 tous les élément de $\mathbb{F}_q^{n_b}$ sont compté dans un et un seul des polynômes $(P_{0,i}(x))_{0 \leq i \leq t}$, $(P_{2,i})_{0 \leq i \leq t}$ et $P_1(x)$, on a donc

$$\sum_{m \in \mathbb{F}_q^{n_b}} x^{w_H(m)} \prod_{j=0}^{t} y^{2\mu(\gamma_j(m))} = P(x, y_0, \ldots, y_t).$$

En élevant le terme de droite de cette égalité à la puissance n_e on obtient

$$\left(\sum_{m \in \mathbb{F}_q^{n_b}} x^{w_H(m)} \prod_{j=0}^{t} y^{2\mu(\gamma_j(m))} \right)^{n_e} = \sum_{(m_1, \ldots, m_{n_e}) \in \mathbb{F}_q^{n_b n_e}} x^{\sum_{i=1}^{n_e} w_H(m_i)} \prod_{j=0}^{t} y^{2 \sum_{i=1}^{n_e} \mu(\gamma_j(m_i))}$$

et enfin, il est clair que

$$\sum_{(m_1, \ldots, m_{n_e}) \in \mathbb{F}_q^{n_b n_e}} x^{\sum_{i=1}^{n_e} w_H(m_i)} \prod_{j=0}^{t} y^{2 \sum_{i=1}^{n_e} \mu(\gamma_j(m_i))} = \sum_{\mathbf{m} \in \mathbb{F}_q^{n_b n_e}} x^{w_H(\mathbf{m})} \prod_{j=0}^{t} y_j^{2\Omega_H(\Gamma_j(\mathbf{m}))}.$$

On en déduit (III.14). □

Définition III.4 *On dira que le polynôme* $p(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ *est inférieur au polynôme* $q(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ *et on notera*

$$p(x_1, \ldots, x_n) \preceq q(x_1, \ldots, x_n)$$

si pour tout $(i_1, \ldots, i_n) \in \mathbb{N}^n$,

$$[\prod_{r=1}^{n} x_r^{i_r}] p(x_1, \ldots, x_n) \leq [\prod_{r=1}^{n} x_r^{i_r}] q(x_1, \ldots, x_n).$$

2.4.1 Majoration des performances de l'algorithme

Proposition III.16 *Soit $Q(x, y_0, \ldots, y_t)$ le polynôme de $\mathbb{Z}[x, y_0, \ldots, y_t]$ défini par*

$$Q(x, y_0, \ldots, y_t) = P(x, y_0, \ldots, y_t)^{n_e},$$

où

$$P(x, y_0, \ldots, y_t) = \sum_{i=0}^{t} P_{0,i}(x) \prod_{j=0}^{i-1} y_j + P_1(x) \prod_{j=0}^{t} y_j + \sum_{i=0}^{t} P_{2,i}(x) \prod_{j=0}^{i-1} y_j \prod_{j=i}^{t} y_j^2.$$

Soit $P_{\mathcal{C}}(x)$ le polynôme des motifs corrigibles de $\mathcal{C}$ pour l'algorithme de Block-Zyablov étendu. Si Φ est borné strictement *par d_e, alors*

$$P_{\mathcal{C}}(x) \preceq \sum_{\substack{(s_0,\ldots,s_t)\in\mathbb{N}^{t+1} \\ \min_{0\leq j\leq t}(s_j)<d_e}} [\prod_{j=0}^{t} y_j^{s_j}] Q(x, y_0, \ldots, y_t). \tag{III.15}$$

preuve : Si $\Psi(\mathbf{m}) = \mathbf{0}$, alors il existe r, $0 \leq r \leq t$, tel que $2\Omega_H(\Gamma_r(\mathbf{m})) < d_e$, donc

$$\left\{\mathbf{m} \in \mathbb{F}_q^{n_b n_e} \mid \Psi(\mathbf{m}) = \mathbf{0}\right\} \subset \left\{\mathbf{m} \in \mathbb{F}_q^{n_b n_e} \mid \min_{0\leq r\leq t}(2\Omega_H(\Gamma_r(\mathbf{m}))) < d_e\right\}.$$

On a donc

$$P_{\mathcal{C}}(x) = \sum_{\mathbf{m}\in\mathbb{F}_q^{n_b n_e}, \Psi(\mathbf{m})=\mathbf{0}} x^{w_H(\mathbf{m})} \preceq \sum_{\mathbf{m}\in\mathbb{F}_q^{n_b n_e}, \min_{0\leq r\leq t}(2\Omega_H(\Gamma_r(\mathbf{m})))<d_e} x^{w_H(\mathbf{m})}.$$

Or d'après le lemme III.5, on a

$$Q(x, y_0, \ldots, y_t) = \sum_{\mathbf{m}\in\mathbb{F}_q^{n_b n_e}} x^{w_H(\mathbf{m})} \prod_{j=0}^{t} y_r^{2\Omega_H(\Gamma_r(\mathbf{m}))},$$

donc

$$\sum_{\substack{(s_0,\ldots,s_t)\in\mathbb{N}^{t+1} \\ \min_{0\leq j\leq t}(s_j)<d_e}} [\prod_{j=0}^{t} y_j^{s_j}] Q(x, y_0, \ldots, y_t) = \sum_{\mathbf{m}\in\mathbb{F}_q^{n_b n_e}, \min_{0\leq r\leq t}(2\Omega_H(\Gamma_r(\mathbf{m})))<d_e} x^{w_H(\mathbf{m})},$$

d'où (III.15). □

Remarque III.9 (**Pourquoi n'a-t'on-qu'une inégalité?**) Les motifs que nous énumérons dans la proposition ci-dessus sont ceux vérifiant :

> "il existe une des $t+1$ branches de l'algorithme de Block-Zyablov qui aboutit au mot nul."

Cela signifie que le motif est énuméré dans un monôme dont le degré en l'un au moins des y_i est inférieur strictement à d_e.

Les motifs corrigibles sont ceux vérifiant :

> "il existe une des $t+1$ branches de l'algorithme de Block-Zyablov qui aboutit au mot nul *et aucune des autres n'aboutit à un mot de code plus proche.*"

On peut envisager que $\Phi_0(\mathbf{m}) = \mathbf{0}$ et $\Psi(\mathbf{m}) \neq \mathbf{0}$, comme le montre l'exemple suivant.

Exemple : Nous reprenons le code concaténé admettant comme code interne le code de Hamming binaire $H(7,4,3)$ et comme code externe le code de Reed-Solomon $RS(15,11,5)$ (cf. 72)

Soit $\mathbf{c}$ le mot de $RS(15,11,5)$ dont l'ecriture polynomiale est

$$1 + z^4 + z^6 + z^7 + z^8 = \left(\alpha^5 + \alpha^{13} z + \alpha^{11} z^2 + \alpha^6 z^3 + z^4\right) g(z)$$

et l'écriture vectorielle

$$\mathbf{c} = (\alpha^3, \alpha^{12}, \alpha^3, \alpha^8, 0, 0, \alpha^7, 0, \dots)$$

Considérons le mot $\Theta(\mathbf{c})$ du code concaténé de $H(7,4,3)$ et $RS(15,11,5)$, et le mot $\mathbf{m} \in \mathbb{F}_2^{105}$

$$\Theta(\mathbf{c}) = \begin{pmatrix} 1000110 \\ 0000000 \\ 0000000 \\ 0000000 \\ 1000110 \\ 0000000 \\ 1000110 \\ 1000110 \\ 1000110 \\ 0000000 \\ \vdots \end{pmatrix}, \quad \text{et} \quad \mathbf{m} = \begin{pmatrix} 0000000 \\ 0000000 \\ 0000000 \\ 0000000 \\ 0000110 \\ 0000000 \\ 0000110 \\ 0000110 \\ 0000110 \\ 0000000 \\ \vdots \end{pmatrix} = \Theta(\mathbf{c}) + \begin{pmatrix} 1000110 \\ 0000000 \\ 0000000 \\ 0000000 \\ 1000000 \\ 0000000 \\ 1000000 \\ 1000000 \\ 1000000 \\ 0000000 \\ \vdots \end{pmatrix}$$

On a

$$\Phi_0(\mathbf{m}) = \mathbf{0} \quad \text{et} \quad \Phi_1(\mathbf{m}) = \mathbf{c}.$$

En effet,

$$\Gamma_0(\mathbf{m}) = (0,0,0,0,\infty,0,\infty,\infty,\infty,0,\dots) \quad \text{et} \quad \Delta_H(\mathbf{0}, \Gamma_0(\mathbf{m})) = 2 < \frac{5}{2},$$

$$\Gamma_1(\mathbf{m}) = (0,0,0,0,1,0,1,1,,1,0,\dots) \quad \text{et} \quad \Delta_H(\mathbf{c}, \Gamma_1(\mathbf{m})) = 1 < \frac{5}{2}$$

et

$$d_H(\mathbf{m}, \Theta(\mathbf{c})) = 7 \quad \text{et} \quad d_H(\mathbf{m}, \mathbf{0}) = 8,$$

ce qui donne

$$\Psi(\mathbf{m}) = \Theta(\mathbf{c}) \neq \mathbf{0}.$$

Donc dans ce cas particulier, on a un motif $\mathbf{m}$ qui est corrigible par Φ_0 et erroné par Ψ, donc pour le code concaténé de cet exemple l'inégalité (III.15) est stricte.

2.4.2 Minoration des performances de l'algorithme

Il semble malheureusement difficile d'obtenir une borne inférieure pertinente pour les performances de l'algorithme de Block-Zyablov étendu. Nous pensons toutefois que la borne supérieure obtenue plus haut est proche de la valeur exacte.

On peut tenter de majorer le nombre de motifs corrigibles pour l'un des Φ_i et erroné pour Ψ. Ce nombre est majoré par le nombre de motifs tels qu'une branche de l'algorithme aboutit au mot nul et une au moins des autres branches aboutit à un mot de code non nul.

Un tel motif $\mathbf{m}$ vérifie

$$\exists\, \mathbf{c} \in \mathcal{C} \setminus \{0\},\ w_H(\mathbf{m}) \geq d_H(\mathbf{m}, \mathbf{c}).$$

Autrement dit, $\mathbf{m}$ est dans une boule de rayon $w_H(\mathbf{m})$ centrée en un mot non nul du code. On pose pour $\mathbf{c} \in \mathcal{C}$,

$$E_r(\mathbf{c}) = \{\mathbf{m} \in \mathbb{F}_q^{n_b n_e} \mid w_H(\mathbf{m}) = r,\ d_H(\mathbf{m}, \mathbf{c}) \leq r\}.$$

Le cardinal de $E_r(\mathbf{c})$ est connu et ne dépend que de $w = w_H(\mathbf{c})$; on le notera donc $E_r(w)$. L'enumérateur des poids d'une boule de rayon r centrée en un mot de poids w est égal à

$$f_{w,r}(x) = \sum_{0 \leq i+j \leq r} \binom{w}{i} \binom{n-w}{j} (q-1)^j (1 + (q-2)x)^i x^{w-i+j}$$

et

$$|E_r(w)| = [x^r] f_{w,r}(x).$$

Le nombre de motifs de poids r corrigibles par l'un des Φ_i mais pas par Ψ, est borné par

$$\sum_{w=d_b d_e}^{2r} A_w(\mathcal{C}) |E_r(w)|.$$

En utilisant le lemme III.2, le nombre de motifs de poids r corrigibles pour l'un des Φ_i mais pas par Ψ, est borné par

$$\max_{d_b d_e \leq w \leq 2r} |E_r(w)| \sum_{i=d_e}^{\lfloor \frac{2r}{d_b} \rfloor} A_i(\mathcal{E}).$$

Ce nombre peut être calculé lorsque l'on connait la distribution des poids du code externe.

2.5 Exemple : Un code $\mathcal{C}(105, 44, 15)$ binaire

- Le code interne est le code de Hamming binaire $H(7, 4, 3)$, muni d'un algorithme de décodage d'erreur à vraisemblance maximale, capable de corriger tout motif d'erreur de poids 1.

- Le code externe est le code de Reed-Solomon $RS(15, 11, 5)$ sur $\mathbb{F}_{16}$, muni d'un algorithme de décodage d'erreur et d'effacement borné strictement par 5.

- Le code concaténé de $\mathcal{B}$ et $\mathcal{E}$, est un code de longueur 105 de dimension 44 et de distance minimale 15.

Pour le code de Hamming on a

$$P_0(x) = 1 + 7x, \quad P_1(x) = 0, \quad P_2(x) = 21x^2 + 35x^3 + 35x^4 + 21x^5 + 7x^6 + x^7$$

et

$$P_{0,0}(x) = 1, \quad P_{0,1}(x) = 7x, \quad P_{2,0}(x) = 7x^3 + 7x^4 + x^7, \quad P_{2,1}(x) = 21x^2 + 28x^3 + 28x^4 + 7x^6.$$

La table III.1 donne pour chaque poids w, le rapport du nombre de motif corrigibles au nombre total de motifs de poids w, et ce pour les trois algorithmes dont l'évaluation est décrite dans cette section.

On notera que la différence entre l'algorithme naïf et l'algorithme de Block-Zyablov étendu est très faible, puisqu'elle ne dépasse pas 2% en valeur relative.

La borne inférieure de l'algorithme de Block-Zyablov étendu ne fourni pas de valeurs interessantes, sauf pour le poids $w = 8$, pour lequel on obtient une proportion de décodage correct supérieure ou égale à 0.9477.

2.6 Evaluation du décodage partiel des codes concaténés

Tous les résultats obtenus pour l'algorithme de Block-Zyablov étendu sont valables pour l'algorithme partiel de Block-Zyablov.

En effet, la seule différence entre ces deux décodages réside dans la fonction qui permet de choisir parmi les branches qui ont abouti. L'encadrement obtenu dans 2.4 n'a pas fait intervenir la façon dont le choix entre les branches s'opère.

Proposition III.17 *Nous reprenons les notations de la section 1.4.*

Le motif d'erreur $\mathbf{e} \in \mathbb{F}_q^{n_b n_e}$ *est corrigible pour le code concaténé* $\mathcal{C}$ *par l'algorithme de décodage partiel de Block-Zyablov* Ψ *si et seulement si*

$$\forall \mathbf{x} \in \mathcal{C},\ \exists \mathbf{v} \in \mathcal{V}^{n_e},\ \Psi(\mathbf{x} + \mathbf{v} + \mathbf{e}) = \mathbf{x}.$$

preuve : Cette propriété ce déduit aisement de la linéarité de Ψ, et de la proposition III.8. $\square$

3 Codes concaténés généralisés

3.1 Définition

3.1.1 Codes concaténés d'ordre 2

Soient deux codes linéaires sur $\mathbb{F}_q$ de même longueur $\mathcal{B}_1(n_b, k_{1,b} + k_{2,b}, d_{1,b})$ et $\mathcal{B}_2(n_b, k_{2,b}, d_{2,b})$, tels que $\mathcal{B}_2 \subset \mathcal{B}_1$.

w	naïf	B-Z étendu (borne supérieure)	B-Z
< 6	1	1	1
6	0.9973	1	1
7	0.9835	1	1
8	0.9436	0.9498	$0.6974\,10^{-2}$
9	0.8625	0.8641	$0.3878\,10^{-3}$
10	0.7345	0.7348	$0.1371\,10^{-4}$
11	0.5706	0.5707	$0.2533\,10^{-6}$
12	0.3969	0.3969	$0.1786\,10^{-8}$
13	0.2429	0.2429	$0.6238\,10^{-10}$
14	0.1287	0.1287	$0.1459\,10^{-11}$
15	$0.5826\,10^{-1}$	$0.5826\,10^{-1}$	$0.1716\,10^{-13}$
16	$0.2222\,10^{-1}$	$0.2222\,10^{-1}$	0
17	$0.7106\,10^{-2}$	$0.7106\,10^{-2}$	0
18	$0.1909\,10^{-2}$	$0.1909\,10^{-2}$	0
19	$0.4345\,10^{-3}$	$0.4345\,10^{-3}$	0
20	$0.8400\,10^{-4}$	$0.8400\,10^{-4}$	0
21	$0.1375\,10^{-4}$	$0.1375\,10^{-4}$	0
22	$0.1885\,10^{-5}$	$0.1885\,10^{-5}$	0
23	$0.2120\,10^{-6}$	$0.2120\,10^{-6}$	0
24	$0.1881\,10^{-7}$	$0.1881\,10^{-7}$	0
25	$0.1238\,10^{-8}$	$0.1238\,10^{-8}$	0
26	$0.5382\,10^{-10}$	$0.5382\,10^{-10}$	0
27	$0.1160\,10^{-11}$	$0.1160\,10^{-11}$	0
> 27	0	0	

Tableau III.1 :Performance, pour les différents algorithmes de décodage, des codes concaténés binaires (105,44) ayant pour code interne le code de Hamming (7,4)

Considérons un sous-code $\mathcal{B}'_1$ de $\mathcal{B}_1$ tel que

$$\mathcal{B}_1 = \mathcal{B}'_1 \oplus \mathcal{B}_2.$$

$\mathcal{B}'_1$ est un code linéaire sur $\mathbb{F}_q$ de longueur n_b, de dimension $k_{1,b}$ et de distance minimale $d'_{1,b} \geq d_{1,b}$.

Soient θ_1 et θ_2 les isomorphismes $\mathbb{F}_q$-linéaires

$$\begin{array}{rcccl} \theta_1 & : & \mathbb{F}_{q^{k_{1,b}}} & \longrightarrow & \mathcal{B}'_1 \\ \theta_2 & : & \mathbb{F}_{q^{k_{2,b}}} & \longrightarrow & \mathcal{B}_2 \end{array}$$

Soit n_e un entier positif, on définit les isomorphismes $\mathbb{F}_q$-linéaires Θ_1 et Θ_2 par

$$\begin{array}{rcccl} \Theta_1 & : & \mathbb{F}^{n_e}_{q^{k_{1,b}}} & \longrightarrow & \mathcal{B}'^{\,n_e}_1 \\ & & \mathbf{x} = (x_1, \ldots, x_{n_e}) & \longmapsto & \Theta_1(\mathbf{x}) = (\theta_1(x_1), \ldots, \theta_1(x_{n_e})) \\ \Theta_2 & : & \mathbb{F}^{n_e}_{q^{k_{2,b}}} & \longrightarrow & \mathcal{B}^{n_e}_2 \\ & & \mathbf{x} = (x_1, \ldots, x_{n_e}) & \longmapsto & \Theta_2(\mathbf{x}) = (\theta_2(x_1), \ldots, \theta_2(x_{n_e})) \end{array}$$

Soient $\mathcal{E}_1(n_e, k_{1,e}, d_{1,e})$ un code linéaire sur $\mathbb{F}_{q^{k_{1,b}}}$ et $\mathcal{E}_2(n_e, k_{2,e}, d_{2,e})$ un code linéaire sur $\mathbb{F}_{q^{k_{2,b}}}$, on définit

- $\mathcal{C}_1 = \Theta_1(\mathcal{E}_1)$ le code concaténé d'ordre 1 de $\mathcal{B}'_1$ et $\mathcal{E}_1$.
- $\mathcal{C}_2 = \Theta_2(\mathcal{E}_2)$ le code concaténé d'ordre 1 de $\mathcal{B}_2$ et $\mathcal{E}_2$.

Proposition III.18 *La somme des codes $\mathcal{C}_1$ et $\mathcal{C}_2$ sur $\mathbb{F}_q$ est directe ; on pose*

$$\mathcal{C} = \mathcal{C}_1 \oplus \mathcal{C}_2.$$

$\mathcal{C}$ est le code concaténé d'ordre 2 *admettant $\mathcal{B}_1$ et $\mathcal{B}_2$ comme codes internes, et $\mathcal{E}_1$ et $\mathcal{E}_2$ comme codes externes.*

preuve : Soit $\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,n_e}) \in \mathcal{E}_1$ et $\mathbf{x}_2 = (x_{2,1}, \ldots, x_{2,n_e}) \in \mathcal{E}_2$, alors

$$\Theta_1(\mathbf{x}_1) = \Theta_2(\mathbf{x}_2) \Rightarrow \forall\, i,\ \theta_1(x_{1,i}) = \theta_2(x_{2,i}),$$

or $\theta_1(x_{1,i}) \in \mathcal{B}'_1$ et $\theta_2(x_{2,i}) \in \mathcal{B}_2$, et la somme de $\mathcal{B}'_1$ et $\mathcal{B}_2$ est directe. □

Proposition III.19 *Soit Θ l'isomorphisme $\mathbb{F}_q$-linéaire suivant*

$$\begin{array}{rcccl} \Theta & : & \mathbb{F}^{n_e}_{q^{k_{1,b}}} \times \mathbb{F}^{n_e}_{q^{k_{2,b}}} & \longrightarrow & \mathcal{B}^{n_e}_1 \\ & & (\mathbf{x}_1, \mathbf{x}_2) & \longmapsto & \Theta(\mathbf{x}_1, \mathbf{x}_2) = \Theta_1(\mathbf{x}_1) + \Theta_2(\mathbf{x}_2) \\ & & & & \Theta(\mathbf{x}_1, \mathbf{x}_2) = (\theta_1(x_{1,1}) + \theta_2(x_{2,1}), \ldots, \theta_1(x_{1,n_e}) + \theta_2(x_{2,n_e})) \end{array}$$

où $\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,n_e})$, et $\mathbf{x}_2 = (x_{2,1}, \ldots, x_{2,n_e})$.

Alors $\mathcal{C}$ le code concaténé d'ordre 2 de $\mathcal{B}_1, \mathcal{B}_2$, et $\mathcal{E}_1, \mathcal{E}_2$ est égal à

$$\mathcal{C} = \Theta\left(\mathcal{E}_1 \times \mathcal{E}_2\right)$$

où $\mathcal{E}_1 \times \mathcal{E}_2$ est considéré comme un $\mathbb{F}_q$-espace vectoriel.

$\mathcal{C}$ est un code linéaire sur $\mathbb{F}_q$, de longueur $n_b n_e$, de dimension $k_{1,b}k_{1,e} + k_{2,b}k_{2,e}$ et dont la distance minimale est supérieure ou égale à $\min(d_{1,b}d_{1,e}, d_{2,b}d_{2,e})$.

preuve : cf. preuve plus générale de la proposition III.21. □

3.1.2 Codes concaténés d'ordre m

Soient m codes linéaires sur $\mathbb{F}_q$ de même longueur

$$1 \leq s \leq m,\ \mathcal{B}_s(n_b, k_{s,b} + k_{(s+1),b} + \ldots + k_{m,b}, d_{s,b}),$$

tels que $\mathcal{B}_m \subset \mathcal{B}_{m-1} \subset \ldots \subset \mathcal{B}_2 \subset \mathcal{B}_1$.

Pour tout s, $1 \leq s \leq m$, on choisit un code $\mathcal{B}'_s$ tel que

- $s < m$, $\mathcal{B}_s = \mathcal{B}'_s \oplus \mathcal{B}_{s+1}$,
- $\mathcal{B}'_m = \mathcal{B}_m$.

Pour tout s, $1 \leq s \leq m$, $\mathcal{B}'_s$ est un code linéaire sur $\mathbb{F}_q$ de longueur n_b et de dimension $k_{s,b}$. De plus, on a

$$\mathcal{B}_1 = \bigoplus_{s=1}^{m} \mathcal{B}'_s. \tag{III.16}$$

Pour tout s, $1 \leq s \leq m$, fixons θ_s un isomorphisme $\mathbb{F}_q$-linéaire

$$\theta_s \ :\ \mathbb{F}_{q^{k_{s,b}}} \longrightarrow \mathcal{B}'_s.$$

Si n_e un entier positif, on définit alors l'isomorphisme $\mathbb{F}_q$-linéaire Θ_s

$$\begin{array}{rccl} \Theta_s \ : & \mathbb{F}_{q^{k_{s,b}}}^{n_e} & \longrightarrow & {\mathcal{B}'_s}^{n_e} \\ & \mathbf{x} = (x_1, \ldots, x_{n_e}) & \longmapsto & \Theta_s(\mathbf{x}) = (\theta_s(x_1), \ldots, \theta_s(x_{n_e})). \end{array}$$

Pour tout code linéaire $\mathcal{E}_s(n_e, k_{s,e}, d_{s,e})$ sur $\mathbb{F}_{q^{k_{s,b}}}$, Θ_s nous permet donc de définir

$$\mathcal{C}_s = \Theta_s(\mathcal{E}_s)$$

le code concaténé d'ordre 1 de $\mathcal{B}'_s$ et $\mathcal{E}_s$.

Proposition III.20 *La somme des codes $(\mathcal{C}_s)_{1\leq s\leq m}$ sur $\mathbb{F}_q$ est directe.*

preuve : Pour tout s, soit $\mathbf{x}_s = (x_{s,1}, \ldots, x_{s,n_e}) \in \mathcal{E}_s$, alors

$$\sum_{s=1}^{m} \Theta_s(\mathbf{x}_s) = 0 \Rightarrow \forall\, i,\ \sum_{s=1}^{m} \theta_s(x_{s,i}) = 0.$$

Or $\theta_s(x_{s,i}) \in \mathcal{B}'_s$ pour tout s, et la somme $\bigoplus_{s=1}^{m} \mathcal{B}'_s$ est directe, donc pour tout s et pour tout i, $x_{s,i} = 0$, donc pour tout s, $\mathbf{x}_s = \mathbf{0}$, d'où le résultat. □

Définition III.5 *Soient $(\mathcal{B}_s(n_b, \sum_{i=s}^{m} k_{i,b}, d_{s,b}))_{1\leq s\leq m}$, une suite décroissante de m codes linéaires sur $\mathbb{F}_q$, appelés codes internes, et $(\mathcal{E}_s(n_e, k_{s,e}, d_{s,e}))_{1\leq s\leq m}$, une suite de m codes linéaires sur $\mathbb{F}_{q^{k_{s,b}}}$, appelés codes externes.*

Pour tout s, $1 \leq s < m$, choisissons un code $\mathcal{B}'_s$ tel que $\mathcal{B}'_s \oplus \mathcal{B}_{s+1} = \mathcal{B}_s$ et $\mathcal{B}'_m = \mathcal{B}_m$.

Pour tout s, $1 \leq s \leq m$, notons $\mathcal{C}_s$ le code concaténé d'ordre 1 de $\mathcal{B}'_s$ et $\mathcal{E}_s$. Le code concaténé généralisé $\mathcal{C}$ d'ordre m admettant les $\mathcal{B}_s$ comme codes internes et les $\mathcal{E}_s$ comme codes externes, est défini comme étant la somme des $\mathcal{C}_s$

$$\mathcal{C} = \bigoplus_{s=1}^{m} \mathcal{C}_s.$$

Proposition III.21 *Soit Θ l'isomorphisme $\mathbb{F}_q$-linéaire suivant*

$$\begin{array}{rcccl} \Theta & : & \mathbb{F}_{q^{k_{1,b}}}^{n_e} \times \ldots \times \mathbb{F}_{q^{k_{m,b}}}^{n_e} & \longrightarrow & \mathcal{B}_1^{n_e} \\ & & (\mathbf{x}_1, \ldots, \mathbf{x}_m) & \longmapsto & \Theta(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \Theta_1(\mathbf{x}_1) + \ldots + \Theta_m(\mathbf{x}_m). \end{array}$$

Alors $\mathcal{C}$ le code concaténé d'ordre m des $\mathcal{B}_s$ et des $\mathcal{E}_s$ est égal à

$$\mathcal{C} = \Theta\,(\prod_{s=1}^{m} \mathcal{E}_s) \qquad \text{(III.17)}$$

où $\prod_{s=1}^{m} \mathcal{E}_s$ est considéré comme un $\mathbb{F}_q$-espace vectoriel.

$\mathcal{C}$ est un code linéaire sur $\mathbb{F}_q$, de longueur $n_b n_e$, de dimension $\sum_{s=1}^{m} k_{s,b} k_{s,e}$, dont la distance minimale est supérieure ou égale à $\min_{1 \leq s \leq m}(d_{s,b} d_{s,e})$.

preuve : D'après la définition III.5, tout élément $\mathbf{x}$ de $\mathcal{C}$ s'écrit de façon unique comme somme d'éléments de $\mathcal{C}_s$, $1 \leq s \leq m$

$$\mathbf{x} = \Theta_1(\mathbf{x}_1) + \ldots + \Theta_m(\mathbf{x}_m), \text{ où } \forall\, s,\ \mathbf{x}_s \in \mathcal{E}_s$$

d'où (III.17).

La dimension de $\mathcal{C}_s$ est $k_{s,b} k_{s,e}$ d'après la proposition III.19. Donc la dimension de $\mathcal{C}$ est $\sum_{s=1}^{m} k_{s,b} k_{s,e}$.

Montrons par récurrence sur m l'énoncé suivant

($\mathcal{P}_m$) Tout code concaténé d'ordre m a une distance minimale $\geq \min\limits_{1 \leq s \leq m} (d_{s,b} d_{s,e})$.

- ($\mathcal{P}_1$) est vrai car $\mathcal{C}$ est alors un code concaténé d'ordre 1.
- Supposons ($\mathcal{P}_{m-1}$) vrai. Soit $\mathbf{x} = (x_1, \ldots, x_{n_e}) \in \mathcal{C}$ et $\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,n_e})$ sa projection sur $\mathcal{C}_1$ parallèlement à $\bigoplus_{s=2}^{m} \mathcal{C}_s$. On distingue deux cas

  $\mathbf{x}_1 \neq \mathbf{0}$ **:** dans ce cas, $d_{1,e}$ au moins des coordonnées de $\mathbf{x}_1$ sont non nulles car $\mathbf{x}_1$ est un mot de $\mathcal{E}_1$, code dont la distance minimale est $d_{1,e}$. Comme $x_{1,i} \neq 0 \Rightarrow x_i \neq 0$, $d_{1,e}$ au moins des x_i sont non nuls. Puisque $\mathcal{B}_1$ a pour distance minimale $d_{1,b}$ et que $x_i \in \mathcal{B}_1$, on a

  $$w_H(\mathbf{x}) = \sum_{i=1}^{n_e} w_H(x_i) \geq d_{1,b} d_{1,e}.$$

  $\mathbf{x}_1 = 0$ **:** dans ce cas on a $\mathbf{x} \in \bigoplus_{s=2}^{m} \mathcal{C}_s$, qui est un code concaténé d'ordre $m-1$, donc en utilisant l'hypothèse de récurrence

  $$w_H(\mathbf{x}) \geq \min_{2 \leq s \leq m} (d_{s,b} d_{s,e}).$$

$\square$

3.2 Décodage des codes concaténés d'ordre m

Soit un entier $m > 1$. Pour tout s, $1 \leq s \leq m$, on considère

- le code interne $\mathcal{B}_s$ muni d'un algorithme de décodage d'erreur γ_s borné par la distance minimale, on suppose de plus que cet algorithme respecte la distance de Hamming,
- le code externe $\mathcal{E}_s$ muni d'un algorithme de décodage d'erreur et d'effacement Φ_s borné par la distance minimale,
- $\mathcal{B}'_s$ un sous code de $\mathcal{B}_s$ et $\mathcal{C}_s$ le code concaténé d'ordre 1 de $\mathcal{B}'_s$ et $\mathcal{E}_s$,
- enfin $\mathcal{C}$ est le code concaténé d'ordre m admettant les $\mathcal{B}_s$ comme codes internes, et les $\mathcal{E}_s$ comme codes externes. C'est-à-dire $\mathcal{C} = \bigoplus_{s=1}^{m} \mathcal{C}_s$.

Nous allons décrire un algorithme de décodage par récurrence. Plus précisement, nous allons montrer que le décodage d'un code concaténé d'ordre m est équivalent au décodage partiel d'un code concaténé d'ordre 1 puis au décodage d'un code concaténé d'ordre $m-1$. Cet algorithme est directement inspiré de celui donné par V. Zinoviev [104] dans le cas non linéaire.

Proposition III.22 *Soit Ψ_1 un décodeur partiel de Block-Zyablov du code concaténé $\mathcal{C}_1$ de $\mathcal{B}'_1$ et $\mathcal{E}_1$ par rapport à $\mathcal{B}_1$ (cf. proposition III.10). Soit Ψ_2 un algorithme de décodage du code concaténé d'ordre $m-1$, $\bigoplus_{s=2}^{m} \mathcal{C}_s$.*

On considère l'algorithme de décodage Ψ défini par

$$\forall\, \mathbf{y} \in \mathbb{F}_q^{n_b n_e},\ \ \Psi(\mathbf{y}) = \begin{cases} \infty & si\ \Psi_1(\mathbf{y}) = \infty \\ \Psi_1(\mathbf{y}) + \Psi_2(\mathbf{y} - \Psi_1(\mathbf{y})) & sinon \end{cases}$$

Si Ψ_2 est borné par $\min_{2\leq s\leq m}(d_{sb}d_{se})$, alors Ψ est borné par $\min_{1\leq s\leq m}(d_{sb}d_{se})$.

preuve : Soient $\mathbf{x} \in \mathcal{C}$ et $\mathbf{y} \in \mathbb{F}_q^{n_b n_e}$ tels que

$$d_H(\mathbf{x}, \mathbf{y}) < \min_{1\leq s\leq m}(\frac{d_{sb}d_{se}}{2}).$$

On a

$$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 + \ldots + \mathbf{x}_m,$$

avec $\forall\, s$, $\mathbf{x}_s \in \mathcal{C}_s$ et on pose

$$\mathbf{x}' = \mathbf{x}_2 + \ldots + \mathbf{x}_m.$$

Puisque Ψ_2 est borné par $\min_{2\leq s\leq m}(d_{sb}d_{se})$, on a

$$\forall\, \mathbf{z} \in \mathbb{F}_q^{n_b n_e},\ d_H(\mathbf{x}', \mathbf{z}) < \min_{2\leq s\leq m}(\frac{d_{sb}d_{se}}{2}) \Rightarrow \Psi_2(\mathbf{z}) = \mathbf{x}' = \mathbf{x}_2 + \ldots + \mathbf{x}_m. \tag{III.18}$$

Puisque Ψ_1 est un décodeur partiel de Block-Zyablov de $\mathcal{C}_1$ par rapport à $\mathcal{B}_1$, on a d'après la proposition III.11

$$\forall\, \mathbf{z} \in \mathbb{F}_q^{n_b n_e},\ d_H(\mathbf{x} = \mathbf{x}_1 + \mathbf{x}', \mathbf{z}) < \frac{d_{1,b} d_{1,e}}{2} \Rightarrow \Psi_1(\mathbf{z}) = \mathbf{x}_1, \tag{III.19}$$

car $\mathbf{x}' \in \mathcal{B}_2^{n_e}$ et $\mathcal{B}_1 = \mathcal{B}_1' \oplus \mathcal{B}_2$.

Donc si

$$d_H(\mathbf{x}, \mathbf{y}) < \min_{1 \leq s \leq m} (\frac{d_{sb} d_{se}}{2}),$$

donc d'après (III.19), $\Psi_1(\mathbf{y}) = \mathbf{x}_1$. On pose alors

$$\mathbf{y}' = \mathbf{y} - \Psi_1(\mathbf{y}) = \mathbf{y} - \mathbf{x}_1.$$

Ce vecteur vérifie

$$d_H(\mathbf{x}', \mathbf{y}') = d_H(\mathbf{x} - \mathbf{x}_1, \mathbf{y} - \mathbf{x}_1) = d_H(\mathbf{x}, \mathbf{y}) < \min_{1 \leq s \leq m} (\frac{d_{sb} d_{se}}{2}),$$

donc d'après (III.18),

$$\Psi_2(\mathbf{y}') = \mathbf{x}' = \mathbf{x}_2 + \ldots + \mathbf{x}_m$$

et

$$\Psi(\mathbf{y}) = \mathbf{x}_1 + \mathbf{x}_2 + \ldots + \mathbf{x}_m = \mathbf{x}.$$

□

Définition III.6 *Soit $\mathcal{C} = \mathcal{C}_1 \oplus \ldots \oplus \mathcal{C}_m$ un code concaténé d'ordre m, on appelle algorithme de Block-Zyablov de $\mathcal{C}$, l'algorithme Ψ défini récursivement par*

1. *Si $m = 1$, Ψ est un décodeur de Block-Zyablov du code concaténé $\mathcal{C}$ d'ordre 1.*

2. *Si $m > 1$, Ψ est défini par*

$$\forall\, \mathbf{y} \in \mathbb{F}_q^{n_b n_e},\ \Psi(\mathbf{y}) = \begin{cases} \infty & si\ \Psi_1(\mathbf{y}) = \infty \\ \Psi_1(\mathbf{y}) + \Psi_2(\mathbf{y} - \Psi_1(\mathbf{y})) & sinon \end{cases}$$

où Ψ_1 est un décodeur partiel de Block-Zyablov du code concaténé de $\mathcal{B}_1'$ et $\mathcal{E}_1$ par rapport à $\mathcal{B}_1$, et Ψ_2 est un décodeur de Block-Zyablov du code concaténé d'ordre $m - 1$ $\mathcal{C}_2 \oplus \ldots \oplus \mathcal{C}_m$.

Proposition III.23 *Soit Ψ un décodeur de Block-Zyablov du code concaténé $\mathcal{C}$ d'ordre m. Alors Ψ est borné par $\min_{1 \leq s \leq m}(d_{sb} d_{se})$.*

preuve : C'est une conséquence immédiate de la proposition III.22. □

3.3 Evaluation des codes concaténés généralisés

Proposition III.24 *Soit un code concaténé d'ordre m,*

$$\mathcal{C} = \bigoplus_{i=1}^{m} \mathcal{C}_i.$$

Pour tout i, on note Ψ_i le décodeur de partiel Block-Zyablov du code concaténé $\mathcal{C}_i$.

Soit Ψ le décodeur de Block-Zyablov du code concaténé $\mathcal{C}$ d'ordre m.

Un motif d'erreur $\mathbf{e} \in \mathbb{F}_q^{n_b n_e}$ est corrigible pour $\mathcal{C}$ par Ψ, si et seulement si pour tout i, il est corrigible pour $\mathcal{C}_i$ par Ψ_i.

preuve : Nous allons montrer ce résultat par récurrence sur l'ordre m.

- il est vrai à l'ordre 1 car $\Psi = \Psi_1$.
- Supposons le résultat vrai pour tout code concaténé jusqu'à l'ordre $m-1$.

  Soit Ψ' le décodeur de Block-Zyablov du code concaténé $\mathcal{C}' = \bigoplus_{i=2}^{m} \mathcal{C}_i$ d'ordre $m-1$. Il suffit de montrer que $\mathbf{e} \in \mathbb{F}_q^{n_b n_e}$ est corrigible pour $\mathcal{C}$ par Ψ ssi il est corrigible pour $\mathcal{C}_1$ par Ψ_1 et pour $\mathcal{C}'$ par Ψ'.

  Soit $\mathbf{e} \in \mathbb{F}_q^{n_b n_e}$, $\mathbf{x}' \in \mathcal{C}'$, $\mathbf{x}_1 \in \mathcal{C}_1$, et $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}' \in \mathcal{C}$.

  - Si $\mathbf{e}$ est corrigible par Ψ, alors

    $$\Psi(\mathbf{x} + \mathbf{e}) = \mathbf{x} = \mathbf{x}_1 + \mathbf{x}',$$

    donc

    $$\Psi_1(\mathbf{x}_1 + \mathbf{x}' + \mathbf{e}) = \mathbf{x}_1, \text{ et } \Psi'(\mathbf{x}' + \mathbf{e}) = \mathbf{x}',$$

    car la décomposition $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}'$ est unique. Donc $\mathbf{e}$ est corrigible par Ψ', et d'après la proposition III.17, il l'est également par Ψ_1.
  - Réciproquement si

    $$\Psi_1(\mathbf{x}_1 + \mathbf{e}) = \mathbf{x}_1, \text{ et } \Psi'(\mathbf{x}' + \mathbf{e}) = \mathbf{x}',$$

    alors on a également

    $$\Psi_1(\mathbf{x}_1 + \mathbf{x}' + \mathbf{e}) = \mathbf{x}_1,$$

    et on en déduit immédiatement que $\mathbf{e}$ est corrigible pour Ψ.

□

Chapitre IV

Codes Produit

Les codes produit ont été introduits par Elias en 1954 [33], qui les appelait alors "codes itérés". Ces idées ont été reprises ensuite par Calingaert en 1961 [17], puis par Burton et Weldon en 1965 [15], qui ont été les premiers à utiliser le terme "code produit" et qui se sont intéressés plus particulièrement au produit de codes cycliques. On peut ensuite citer Goethals [38], Kasami [52] et Gore [42] qui ont utilisé, entre les années 67 et 70, la structure des codes produit pour obtenir des résultats sur la métrique de certains codes cycliques. L'essentiel des travaux significatifs sur les codes produit ont été théoriques, et mettent en relief ces codes en tant qu'objets algébriques.

Des idées sur le décodage des codes produit sont données dès leurs origines, mais il a fallu attendre 1972 pour que Reddy et Robinson [79] donnent un algorithme de décodage borné par la distance minimale, en s'inspirant des idées de Forney sur le décodage GMD (*Generalised Minimum Distance*) [36].

Dans ce chapitre nous commencons par énoncer dans la section 1 la définition ainsi que les propriétés qui nous ont semblé importantes des codes produit.

La section 2 traite du décodage des codes produit avec en particulier deux algorithmes, un premier "élémentaire" borné par la moitié de la distance minimale seulement, et l'algorithme de Reddy-Robinson que nous généralisons pour un corps quelconque ; celui donné par Reddy et Robinson dans [79] se limite au cas binaire. Cette généralisation utilise des techniques proches de celles du décodage de Block-Zyablov des codes concaténés.

Dans les sections 3 et 4, les codes produit sont évalués pour le décodage des effacements seuls puis celui des erreurs seules. Ces évaluations utilisent les outils mis en place dans le chapitre I, à savoir le polynôme des motifs corrigibles et la capacité de correction.

La section 5 décrit le résultat de simulations très poussées menées sur des produits de codes de Reed-Solomon. Nous constatons que la capacité de correction de certains de ces codes est très élevée, en tout cas très supérieure à leur distance minimale. En particulier pour un taux d'erreur résiduel de 10^{-5} le code $RS(15, 7, 9) \otimes RS(15, 7, 9)$ a une capacité de correction égale à 179, supérieure à la borne de Singleton, alors que sa distance minimale n'est que de 81.

Enfin la section 6 donne quelque éléments permettant d'estimer la complexité de décodage des codes produit. Nous montrons que celle-ci est faible, et même comparable par symbole transmis a celle des codes composants nettement moins performants.

1 Définitions – Propriétés

1.1 Définitions

Dans toute cette section, on considère C_1 and C_2, deux codes linéaires sur $\mathbb{F}_q$ de paramètres $C_1(n_1, k_1, d_1)$ et $C_2(n_2, k_2, d_2)$ pour la métrique de Hamming.

Définition IV.1 *Le code produit $C_1 \otimes C_2$ de C_1 et C_2 est l'ensemble des matrices M de taille $n_2 \times n_1$ vérifiant :*

- *chaque ligne de M est un mot de code de C_1,*
- *chaque colonne de M est un mot de code de C_2.*

1.1.1 Représentation des codes produit comme polynômes à deux indéterminées

Pour tout entier n, $\mathbb{F}_q^n$ est isomorphe en tant que $\mathbb{F}_q$-espace vectoriel à l'espace des polynômes à une indeterminée sur $\mathbb{F}_q$ de degré au plus $n-1$. Tout mot de $\mathbb{F}_q^n$

$$\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$$

est identifié au polynôme

$$a(X) = a_0 + a_1 X + \ldots + a_{n-1} X^{n-1}.$$

De même, on peut identifier, en tant que $\mathbb{F}_q$-espace vectoriel, l'ensemble des matrices de taille $n_2 \times n_1$ à l'ensemble des polynômes à deux indéterminées X et Y sur $\mathbb{F}_q$, de degré au plus $n_1 - 1$ en X, et $n_2 - 1$ en Y. La matrice

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & \ldots & a_{0,n_1-1} \\ a_{1,0} & a_{1,1} & \ldots & a_{1,n_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_2-1,0} & a_{n_2-1,1} & \ldots & a_{n_2-1,n_1-1} \end{pmatrix}$$

sera ainsi identifiée au polynôme

$$A(X,Y) = \sum_{0 \leq i < n_2, 0 \leq j < n_1} a_{ij} X^j Y^i.$$

Cette identification nous permet de définir le code produit $C_1 \otimes C_2$ comme un ensemble de polynômes

$$A(X,Y) \in C_1 \otimes C_2 \Leftrightarrow A(X,Y) = \sum_{i=0}^{n_2-1} x_i(X) Y^i = \sum_{j=0}^{n_1-1} y_j(Y) X^j \qquad \text{(IV.1)}$$

où

$$\forall\, i,\ 0 \leq i < n_2,\ x_i(X) \in C_1 \ \text{ et } \ \forall\, j,\ 0 \leq j < n_1,\ y_j(Y) \in C_2.$$

1.1.2 Les paramètres des codes produit

L'espace vectoriel des matrices $n \times m$ sur $\mathbb{F}_q$ sera noté $\mathcal{M}_q(n,m)$.

Nous allons montrer ici que le code produit $C_1 \otimes C_2$ est un code linéaire sur $\mathbb{F}_q$ de paramètres

$$N = n_1 n_2, \quad K = k_1 k_2, \quad D = d_1 d_2,$$

où N, K et D sont respectivement la longueur, la dimension et la distance minimale du code produit.

Un schéma de codage simple. Soient G_1 et G_2 des matrices génératrices de C_1 et C_2 sous forme systématique,

$$G_i = \left(\begin{array}{c|c} I_{k_i} & P_i \end{array} \right), \quad i = 1, 2.$$

Où I_k est la matrice identité de $\mathcal{M}_q(k,k)$ et $P_i \in \mathcal{M}_q(n_i - k_i, k_i)$. Soit l'application $\mathbb{F}_q$-linéaire

$$\begin{array}{rcccl} \Phi & : & \mathcal{M}_q(k_2,k_1) & \longrightarrow & \mathcal{M}_q(n_2,n_1) \\ & & X & \longmapsto & \Phi(X) = \begin{pmatrix} X & XP_1 \\ {}^tP_2X & {}^tP_2XP_1 \end{pmatrix} \end{array}$$

Proposition IV.1 *Le morphisme Φ est injectif et l'image de $\mathcal{M}_q(k_2,k_1)$ par Φ est égale au code produit $C_1 \otimes C_2$.*

preuve : $\Phi(X) = 0 \Leftrightarrow X = 0$ est clair. Soit $X \in \mathcal{M}_q(k_2,k_1)$, chacune des lignes de $\Phi(X)$ est de la forme

$$\left(\begin{array}{c|c} \mathbf{x} & \mathbf{x}P_1 \end{array} \right) = \mathbf{x}\,G_1 \in C_1, \text{ où } \mathbf{x} \in \mathbb{F}_q^{n_1}.$$

Le résultat s'obtient pour les colonnes par transposition. □

Corollaire IV.1 *La dimension de $C_1 \otimes C_2$ est $K = k_1 k_2$.*

Remarque IV.1 En plus de la dimension des codes produit, la proposition IV.1 nous fourni avec le morphisme Φ un algorithme de codage.

La distance minimale. Pour determiner la distance minimale du code produit $C_1 \otimes C_2$ nous utiliserons la représentation des mots par des polynômes à deux indéterminées.

Proposition IV.2 *Soient $x(X) \in C_1$ et $y(Y) \in C_2$, alors*

$$A(X,Y) = x(X)y(Y) \in C_1 \otimes C_2,$$

où le produit $x(X)y(Y)$ est le produit usuel des polynômes. De plus on a

$$w_H(A) = w_H(x)w_H(y).$$

preuve : Posons

$$\begin{aligned} x(X) &= x_0 + x_1 X + \ldots + x_{n_1-1} X^{n_1-1} \\ y(Y) &= y_0 + y_1 Y + \ldots + y_{n_2-1} Y^{n_2-1} \end{aligned}$$

on a alors

$$A(X,Y)x(X)y(Y) = \sum_{i=0}^{n_2-1} x(X) y_i Y^i = \sum_{j=0}^{n_1-1} y(Y) x_j X^j$$

donc $A(X,Y)$ vérifie (IV.1), ce qui prouve son appartenance au code produit.

Le poids de Hamming d'un polynôme est égal à son nombre de monômes, on a donc bien $w_H(A) = w_H(x) w_H(y)$. □

Le corollaire suivant est immédiat.

Corollaire IV.2 *La distance minimale de $C_1 \otimes C_2$ est $D = d_1 d_2$.*

1.1.3 Dual d'un code produit

On pose

$$\mathcal{C}(N,K,D) = C_1 \otimes C_2,\ N = n_1 n_2,\ K = k_1 k_2,\ D = d_1 d_2.$$

$\mathcal{C}^\perp$ le code dual de $\mathcal{C}$ est un code linéaire sur $\mathbb{F}_q$, de longueur N et de dimension $N - K$.

Proposition IV.3 *Le code dual de $\mathcal{C}$ est égal au $\mathbb{F}_q$-espace vectoriel*

$$\mathcal{C}^\perp = (C_1^\perp \otimes \mathbb{F}_q^{n_2}) + (\mathbb{F}_q^{n_1} \otimes C_2^\perp).$$

preuve : Notons pour l'instant

$$\mathcal{C}' = (C_1^\perp \otimes \mathbb{F}_q^{n_2}) + (\mathbb{F}_q^{n_1} \otimes C_2^\perp),$$

nous allons montrer que $\mathcal{C}' = \mathcal{C}^\perp$.

Le produit scalaire de deux matrices de $\mathcal{M}_q(n_2, n_1)$ est égal à la somme des produits scalaires des lignes, ou de façon équivalente, à la somme des produits scalaires des colonnes, c'est-à-dire, si $M, M' \in \mathcal{M}_q(n_2, n_1)$

$$M = \begin{pmatrix} \mathbf{m}_0 \\ \vdots \\ \mathbf{m}_{n_2-1} \end{pmatrix},\quad M' = \begin{pmatrix} \mathbf{m}'_0 \\ \vdots \\ \mathbf{m}'_{n_2-1} \end{pmatrix},\quad \langle M, M' \rangle = \sum_{i=0}^{n_2-1} \langle \mathbf{m}_i, \mathbf{m}'_i \rangle. \tag{IV.2}$$

Pour montrer $\mathcal{C}' \subset \mathcal{C}^\perp$, il suffit de montrer $C_1^\perp \otimes \mathbb{F}_q^{n_2} \subset \mathcal{C}^\perp$, l'autre inclusion étant symétrique.

Soit $M' \in C_1^\perp \otimes \mathbb{F}_q^{n_2}$, M' est une matrice dont les lignes sont dans $C_1^\perp$, les colonnes étant quelconque. D'après (IV.2) son produit scalaire avec un élément $M \in C_1 \otimes C_2$ vaut

$\sum_{i=0}^{n_2-1}\langle \mathbf{m}_i, \mathbf{m}'_i\rangle$, où pour tout i, $\mathbf{m}_i \in C_1$ et $\mathbf{m}'_i \in C_1^\perp$, donc $\langle M, M'\rangle = 0$, et $M' \in \mathcal{C}^\perp$. D'où l'inclusion $\mathcal{C}' \subset \mathcal{C}^\perp$.

On a

$$(C_1^\perp \otimes \mathbb{F}_q^{n_2}) \cap (\mathbb{F}_q^{n_1} \otimes C_2^\perp) = C_1^\perp \otimes C_2^\perp,$$

en effet une matrice qui a simultanement toutes ses lignes dans $C_1^\perp$, et toutes ses colonnes dans $C_2^\perp$, est par définition dans le produit de ces deux codes.

Calculons la dimension de $\mathcal{C}'$,

$$\dim \mathcal{C}' = \dim(C_1^\perp \otimes \mathbb{F}_q^{n_2}) + \dim(\mathbb{F}_q^{n_1} \otimes C_2^\perp) - \dim((C_1^\perp \otimes \mathbb{F}_q^{n_2}) \cap (\mathbb{F}_q^{n_1} \otimes C_2^\perp)),$$

donc

$$\dim \mathcal{C}' = (n_1 - k_1)n_2 + n_1(n_2 - k_2) - (n_1 - k_1)(n_2 - k_2) = n_1 n_2 - k_1 k_2$$

qui est précisément la dimension de $\mathcal{C}^\perp$, d'où le résultat. □

Définition IV.2 *On appellera* ensemble de positions d'information *d'un code linéaire sur $\mathbb{F}_q$ de dimension k un ensemble de k positions telles que la projection du code sur ces positions soit égale à $\mathbb{F}_q^k$.*

Proposition IV.4 *Soit $C(n, k, d)$ un code linéaire sur $\mathbb{F}_q$ de distance duale $d^\perp$, la restriction de C à tout ensemble de $r \leq d^\perp - 1$ positions contient chaque r-uple exactement q^{k-r} fois, et $d^\perp - 1$ est le plus grand entier ayant cette propriété.*

preuve : cf. [67, Ch. 5, Th. 8, p. 139]. □

Remarque IV.2 Cette propriété est montrée pour un code non nécessairement linéaire par Delsarte dans [28] et est équivalente à dire que l'ensemble des mots d'un code forment un tableau orthogonal de force $d^\perp - 1$ (cf. [13] et [67, p. 328]).

Corollaire IV.3 *Soit un code $C(n, k, d)$ sur $\mathbb{F}_q$ de matrice génératrice G et de distance duale $d^\perp$. On a*

1. *Toute famille de $r \leq d^\perp - 1$ colonnes de G est libre.*
2. *Toute famille libre de k colonnes de G correspond à un ensemble de positions d'information de C.*
3. *Tout ensemble de $r \leq d^\perp - 1$ positions est inclus dans un ensemble de positions d'information de C.*

preuve : Ces propriétés sont des résultats classique d'algèbre linéaire, conséquences directes de la proposition IV.4. □

Lemme IV.1 *Soient $d_1^\perp$ la distance duale de C_1 et $d_2^\perp$ la distance duale de C_2.*

La restriction de $C_1 \otimes C_2$ à toute sous matrice de taille $(d_2^\perp - 1) \times (d_1^\perp - 1)$ contient chaque matrice de $\mathcal{M}_q(d_2^\perp - 1, d_1^\perp - 1)$ exactement $q^{k_1 k_2 - (d_2^\perp - 1)(d_1^\perp - 1)}$ fois.

preuve : Considèrons un support d'éléments de $C_1 \otimes C_2$ constitué d'une sous-matrice $(d_2^\perp - 1) \times (d_1^\perp - 1)$. Ce support correspond au choix d'un ensemble de $d_1^\perp - 1$ positions de colonnes et d'un ensemble de $d_2^\perp - 1$ positions de lignes.

L'ensemble des $d_1^\perp - 1$ colonnes est inclus d'après la corollaire IV.3 dans un ensemble de k_1 colonnes correspondant à des positions d'informations de C_1, de même l'ensemble des $d_2^\perp - 1$ lignes est inclus dans un ensemble de k_2 lignes correspondant à des positions d'informations de C_2.

Supposons sans perte de généralité que les positions d'informations de C_1 (resp. C_2) sont les k_1 (resp. k_2) premières. Lorsque l'on projette $C_1 \otimes C_2$ sur la sous-matrice $k_2 \times k_1$ décrite plus haut, on obtient exactement une fois chaque matrice $k_2 \times k_1$ sur $\mathbb{F}_q$. Comme la matrice $(d_2^\perp - 1) \times (d_1^\perp - 1)$ choisie initialement est incluse dans cette matrice $k_2 \times k_1$, on a le résultat. □

Proposition IV.5 *La distance duale de $\mathcal{C} = C_1 \otimes C_2$ est égale à* $\min(d_1^\perp, d_2^\perp)$.

preuve : Soit $D^\perp$ la distance duale de $\mathcal{C}$, c'est-à-dire la distance minimale de $\mathcal{C}^\perp = (C_1^\perp \otimes \mathbb{F}_q^{n_2}) \oplus (\mathbb{F}_q^{n_1} \otimes C_2^\perp)$.

Il existe un élément non nul de $C_1^\perp \otimes \mathbb{F}_q^{n_2}$ de poids $d_1^\perp$. De même il existe un élément non nul de $\mathbb{F}_q^{n_1} \otimes C_2^\perp$ de poids $d_2^\perp$, donc $D^\perp \leq \min(d_1^\perp, d_2^\perp)$.

Soit un mot de $\mathcal{C}^\perp$ de poids $< \min(d_1^\perp, d_2^\perp)$, ce mot a son support inclus dans une matrice $(d_2^\perp - 1) \times (d_1^\perp - 1)$. Or d'après le lemme IV.1, on peut trouver un mot de $C_1 \otimes C_2$ ayant des valeurs arbitraires sur cette sous-matrice.

Le mot de $\mathcal{C}^\perp$ doit être orthogonal à tous les mots du code, donc dans le cas que nous considérons, il doit être nul, et donc $D^\perp \geq \min(d_1^\perp, d_2^\perp)$. □

1.2 Distribution des poids d'un code produit

Nous considérons les codes $C_1(n_1, k_1, d_1)$, $C_2(n_2, k_2, d_2)$ et $C_1 \otimes C_2 = \mathcal{C}(N, K, D)$ linéaires sur $\mathbb{F}_q$, avec $N = n_1 n_2$, $K = k_1 k_2$, $D = d_1 d_2$.

1.2.1 Les poids faibles

Pour un code donné C, nous noterons $A_i(C)$, le nombre de mots de C de poids i.

Lemme IV.2 *Soient $C(n, k, d)$ un code linéaire sur $\mathbb{F}_q$ $\mathbf{x}, \mathbf{y} \in C$ de poids d. Si $\mathbf{x}$ et $\mathbf{y}$ ont même support, alors ils sont proportionnels.*

preuve : Soient $\mathbf{x} = (x_0, \ldots, x_{n-1}), \mathbf{y} = (y_0, \ldots, y_{n-1}) \in C$ de poids d et de même support. Soit i dans le support commun. On a $x_i \neq 0$ et le mot $\mathbf{z} = y_i x_i^{-1} \mathbf{x} - \mathbf{y} \in C$ s'annule en position i, d'où $w_H(\mathbf{z}) < d$ et $\mathbf{z} = \mathbf{0}$, d'où le résultat. □

Proposition IV.6 *Le nombre de mots de code de $C = C_1 \otimes C_2$ de poids $D = d_1 d_2$ est*

$$A_D(\mathcal{C}) = \frac{1}{q-1} A_{d_1}(C_1) A_{d_2}(C_2), \tag{IV.3}$$

et il n'y a aucun autre poids inférieur à $D + \min(d_1, d_2)$*, c'est-à-dire*

$$A_w(\mathcal{C}) = 0 \tag{IV.4}$$

pour tout w *tel que* $D < w < D + \min(d_1, d_2)$.

preuve :

1. $w = D$.

   Tout mot de poids $d_1 d_2$ a pour support une sous-matrice $d_2 \times d_1$, sinon on a plus de d_2 lignes non nulles ou plus de d_1 colonnes non nulles et le poids est supérieur à $d_1 d_2$.

   D'après le lemme IV.2, toutes les lignes sont proportionnelles et toutes les colonnes sont proportionnelles. Donc un mot de poids minimal de $C_1 \otimes C_2$ sera de la forme

   $$\Theta(\mathbf{x}_1, \mathbf{x}_2) = \begin{pmatrix} x_{1,0}{}^t\mathbf{x}_2 & x_{1,1}{}^t\mathbf{x}_2 & \dots & x_{1,n_1-1}{}^t\mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} x_{2,0}\mathbf{x}_1 \\ x_{2,1}\mathbf{x}_1 \\ \vdots \\ x_{2,n_2-1}\mathbf{x}_1 \end{pmatrix},$$

   où

   $$\begin{array}{l} \mathbf{x}_1 = (x_{1,0}, x_{1,1}, \dots, x_{1,n_1-1}) \in C_1, \\ \mathbf{x}_2 = (x_{2,0}, x_{2,1}, \dots, x_{2,n_2-1}) \in C_2. \end{array}$$

   D'après l'équivalence

   $$\Theta(\mathbf{x}_1, \mathbf{x}_2) = \Theta(\mathbf{x}'_1, \mathbf{x}'_2) \Leftrightarrow \exists\, \lambda \in \mathbb{F}_q^*, \begin{cases} \mathbf{x}_1 = \lambda \mathbf{x}'_1 \\ \lambda \mathbf{x}_2 = \mathbf{x}'_2 \end{cases} \tag{IV.5}$$

   on peut affirmer que chaque mot de poids $d_1 d_2$ est obtenu exactement $q-1$ fois par la construction Θ, d'où (IV.3).

2. $D < w < D + \min(d_1, d_2)$.

   Un mot de poids $> D$ doit avoir soit une ligne de poids $> d_1$, soit une colonne de poids $> d_2$, donc doit avoir soit plus de d_1 colonnes non nulles, soit plus de d_2 lignes non nulles, donc dans tous les cas le poids d'un tel mot est supérieur ou égal à $D + \min(d_1, d_2)$.

$\square$

Corollaire IV.4 *Si* C_1 *et* C_2 *sont MDS, et* $\mathcal{C} = C_1 \otimes C_2$*, alors*

$$A_D(\mathcal{C}) = (q-1)\binom{n_1}{d_1}\binom{n_2}{d_2}. \tag{IV.6}$$

preuve : Le nombre de mot de poids d d'un code MDS (n, k, d) est [67, p. 320] $(q-1)\binom{n}{d}$.
$\square$

Il est possible de connaître la distribution des poids des codes produit jusqu'à $d_1 d_2 + \max(d_1, d_2)$.

Proposition IV.7 *[92] Si $d_2 \leq d_1$, alors $\forall\, i$, $0 < i < d_1/d_2$, on a*

$$A_{(d_1+i)d_2}(\mathcal{C}) = \frac{1}{q-1} A_{d_1+i}(C_1) A_{d_2}(C_2), \tag{IV.7}$$

et les autres poids sont nuls

$$\forall\, w,\ d_1 d_2 < w < d_1 d_2 + d_1,\ d_2 \not| w,\ A_w(\mathcal{C}) = 0.$$

preuve : $w < d_1 d_2 + d_1$, il y a exactement d_2 lignes non nulles, sinon le poids serait supérieur ou égal à $d_1 d_2 + d_1$. On pose $i = w/d_2 - d_1$, chacune des lignes a un poids égal à $d_1 + i$, sinon une des colonnes aurait un poids $< d_2$, donc i est entier, de plus d'après le lemme IV.2 toutes les colonnes sont proportionnelles, donc toutes les lignes également, tous les mots de poids w sont donc obtenu à l'aide de Θ, et chacun d'eux est obtenus $q-1$ fois de cette manière, donc pour tout $i < d_1/d_2$ on a (IV.7). □

1.2.2 Les moments de la distribution des poids

Proposition IV.8 *Soit $\mathcal{C}$ un code de longueur n sur $\mathbb{F}_q$, et soit $d^\perp$ sa distance duale.*

Les $d^\perp - 1$ premiers moments de la loi de distribution des poids de $\mathcal{C}$ sont égaux aux moments de la distributions des poids de $\mathbb{F}_q^n$.

preuve : Les identités de Pless [67, p. 131] nous donnent les différents moments de la distribution des poids de $\mathcal{C}$:

$$\forall\, s,\ \sum_{i=s}^{n} \binom{i}{s} \frac{A_i(\mathcal{C})}{q^k} = \frac{1}{q^s} \sum_{i=0}^{s} (-1)^i \binom{n-i}{s-i} A_i(\mathcal{C}^\perp) \tag{IV.8}$$

donc pour $s < d^\perp$:

$$\sum_{i=s}^{n} \binom{i}{s} \frac{A_i(\mathcal{C})}{q^k} = \frac{1}{q^s} \binom{n}{s} \tag{IV.9}$$

si $\mathcal{C} = \mathbb{F}_q^n$, alors $\mathcal{C}^\perp = \{0\}$, donc (IV.8) nous donne

$$\forall\, s,\ \sum_{i=s}^{n} \binom{i}{s} \frac{A_i(\mathbb{F}_q^n)}{q^n} = \frac{1}{q^s} \binom{n}{s} \tag{IV.10}$$

les équations (IV.9) et (IV.10) nous donnent l'égalité des $d^\perp - 1$ premiers moments binomiaux, donc le résultat est également vrai pour les moments polynômiaux, donc

$$\forall\, s,\ \frac{1}{q^n} \sum_{i=s}^{n} i^s A_i(\mathbb{F}_q^n) = \frac{1}{q^k} \sum_{i=s}^{n} i^s A_i(\mathcal{C}).$$

□

1.3 Code produit et codes concaténés

On suppose que C_1 et C_2 sont sous forme systématique et ont pour matrices génératrices

$$G_1 = \left(\; I_{k_1} \mid P_1 \; \right), \text{ et } G_2 = \left(\; I_{k_2} \mid P_2 \; \right)$$

Proposition IV.9 *Le produit de codes est distributif par rapport à la somme directe. Autrement dit, si C_1, C_2 et C_2' sont des codes linéaires et que la somme de C_2 et C_2' est directe, alors :*

$$C_1 \otimes (C_2 \oplus C_2') = (C_1 \otimes C_2) \oplus (C_1 \otimes C_2').$$

preuve : Il est clair, si l'on revient à la définition, que $C_1 \otimes C_2 \subset C_1 \otimes (C_2 \oplus C_2')$, et de même que $C_1 \otimes C_2' \subset C_1 \otimes (C_2 \oplus C_2')$. L'égalité des dimensions nous donne l'égalité de la somme de ces $\mathbb{F}_q$-espaces vectoriels.

Enfin la somme est directe car si $M \in C_1 \otimes C_2$ et $M \in C_1 \otimes C_2'$ alors toutes les colonnes de M sont dans $C_2 \cup C_2' = \{0\}$, et donc $M = 0$. On a bine $C_1 \otimes C_2 \cup C_1 \otimes C_2' = \{0\}$. □

Proposition IV.10 *On suppose que le code C_2 est sous forme systématique. Pour tout i, $0 \leq i < k_2$, soit $\mathbf{e}_i$ l'élément de C_2 dont les k_2 premières composantes sont nulles, execpté la i-ème qui vaut 1.*

Soit $C_{2,i}$ le code engendré par $\mathbf{e}_i$, $C_{2,i}$ est un code linéaire sur $\mathbb{F}_q$ dont les paramètres sont $(n_2, 1, d_{2,i} \geq d_2)$. On a

$$\bigoplus_{i=0}^{k_2-1} C_{2,i} = C_2, \quad \textit{et} \quad C_1 \otimes C_2 = \bigoplus_{i=0}^{k_2-1} C_1 \otimes C_{2,i}.$$

preuve : $(\mathbf{e}_i)_{0 \leq i \leq k_2-1}$, est une base de C_2 et les $C_{2,i}$ sont 2 à 2 disjoints, on a donc la première somme directe. La deuxième se déduit de la proposition IV.9. □

Proposition IV.11 *Soient les isomorphismes*

$$\begin{array}{rcrcl} \theta_i & : & \mathbb{F}_q & \longrightarrow & C_{2,i} \\ & & x & \longmapsto & x\,\mathbf{e}_i \end{array}$$

et

$$\begin{array}{rcccl} \Theta_i & : & \mathbb{F}_q^{n_1} & \longrightarrow & {C_{2,i}}^{n_1} \\ & & \mathbf{x} = (x_1, \ldots, x_{n_1}) & \longmapsto & \Theta_i(\mathbf{x}) = (\theta_i(x_1), \ldots, \theta_i(x_{n_1})). \end{array}$$

$\mathcal{C}_i = \{\Theta_i(\mathbf{x}) \mid \mathbf{x} \in C_1\}$ *est le code concaténé de $C_{2,i}$ et C_1, on a $\mathcal{C}_i = C_1 \otimes C_{2,i}$ et donc*

$$\mathcal{C} = C_1 \otimes C_2 = \bigoplus_{i=0}^{k_2-1} \mathcal{C}_i. \tag{IV.11}$$

preuve : L'égalité $\mathcal{C}_i = C_1 \otimes C_{2,i}$ est évidente lorsqu'on représente l'élément $\Theta_i(\mathbf{x})$ par la matrice

$$\left(x_1\,{}^t\mathbf{e}_i \mid \ldots \mid x_{n_1}\,{}^t\mathbf{e}_i\right) = \mathbf{x}\,{}^t\mathbf{e}_i \in \mathcal{M}_q(n_2, n_1)$$

où le produit dans le terme de droite de l'égalité est un produit de matrice, c'est-à-dire où l'on considère que ${C_{2,i}}^{n_1} \subset \mathcal{M}_q(n_2, n_1)$. □

Bien entendu on peut également définir cette somme directe en intervertissant les rôles de C_1 et C_2.

Corollaire IV.5 *Le code produit $C_1 \otimes C_2$ est égal au code concaténé généralisé d'ordre k_2 dont les k_2 codes internes sont*

$$\mathcal{B}_r = \bigoplus_{i=1}^{r} C_{2,i}$$

pour $r \in \{1, \ldots, k_2\}$ et dont les k_2 codes externes sont égaux à C_1.

2 Décodage des codes produit

2.1 L'algorithme élémentaire

Nous supposons que nous possédons des algorithmes de décodage Ψ_1 et Ψ_2 pour C_1 et pour C_2 respectivement.

Dans une matrice $n_2 \times n_1$ d'élements de $\mathbb{F}_q$, lorsque nous parlerons de décoder une ligne ou une colonne, il s'agira de

- corriger la ligne (resp. la colonne) dans la matrice si Ψ_1 (resp. Ψ_2) appliqué à cette ligne (resp. cette colonne) retourne un mot de C_1 (resp. C_2).
- ne pas modifier la matrice sinon.

Nous considerons l'algorithme de décodage suivant :

Algorithme $\mathcal{A}_4$

Étape 1. *décoder successivement chaque ligne puis chaque colonne à l'aide de Ψ_1, puis Ψ_2.*

Étape 2. *si la matrice est un mot du code produit* → SUCCES

Étape 3. *si la matrice a déjà été obtenue précédemment à cette étape* → ECHEC

Étape 4. → **Etape 1.**

Remarque IV.3

1. L'étape 3 doit être différente en pratique, on ne peut en effet pas garder en mémoire le résultat après chaque itération.

2. L'algorithme élémentaire en une seule passe, c'est-à-dire en ne conservant que l'étape 1, s'apparente à l'algorithme naïf de décodage des codes concténés.

Proposition IV.12 *Si Ψ_1 et Ψ_2 sont linéaires, alors l'algorithme $\mathcal{A}_4$ est linéaire.*

preuve : l'algorithme $\mathcal{A}_4$ est une suite de décodages de lignes et de colonnes par Ψ_1 et Ψ_2, si ces deux décodeurs sont linéaires alors il en est de même pour l'algorithme $\mathcal{A}_4$. □

Proposition IV.13 *Si Ψ_1 l'algorithme de décodage de C_1 est borné par l_1 impair, et Ψ_2 l'algorithme de décodage de C_2 est borné par l_2 impair, alors l'algorithme $\mathcal{A}_4$ est borné par $\frac{(l_1+1)(l_2+1)}{2}$. Autrement dit, tout motif d'erreur de poids inférieur strictement à $\frac{(l_1+1)(l_2+1)}{4}$ est corrigible.*

preuve : Un motif d'erreur non corrigible par Ψ_1 (resp. Ψ_2) est de poids au moins égal à $\frac{l_1+1}{2}$ (resp. $\frac{l_2+1}{2}$).

Soit un motif d'erreur pour le code produit de poids inférieur strictement à $\frac{(l_1+1)(l_2+1)}{4}$, au plus $\frac{l_2-1}{2}$ lignes ne sont pas corrigibles par Ψ_1, sinon le poids du motif serait au moins égal à $\frac{l_2+1}{2}\frac{l_1+1}{2}$, correspondant à au moins $\frac{l_2+1}{2}$ lignes dans lesquelles l'erreur est de poids au moins $\frac{l_1+1}{2}$.

Donc pour chaque colonne on a au plus $\frac{l_2-1}{2}$ erreurs, et le motif est corrigible. □

Proposition IV.14 *S'il existe un motif d'erreur non corrigible par Ψ_1 de poids w_1 et un motif d'erreur non corrigible par Ψ_2 de poids w_2, alors il existe un motif non corrigible de poids $w_1 w_2$ pour l'algorithme $\mathcal{A}_4$*

preuve : Pour $i = 1, 2$, soit $e_i(X)$ la représentation polynomiale d'un motif de poids w_i non corrigible par Ψ_i, on pose $m_i(X) = \Psi_i(e_i(X))$ si $\Psi_i(e_i(X)) \neq \infty$, $m_i(X) = e_i(X)$ sinon. Notons que $\Psi_i(m_i(X)) = m_i(X)$.

Le motif d'erreur du code produit dont la représentation polynomiale est $e_1(X)e_2(Y)$ est non corrigible, en effet après correction des lignes le motif devient $m_1(X)e_2(Y)$, après correction des colonnes il devient $m_1(X)m_2(Y)$, et il ne sera plus modifié par la suite. L'image de $e_1(X)e_2(Y)$ par l'algorithme est donc non nulle, et ce motif a pour poids $w_1 w_2$. □

Corollaire IV.6 *Quels que soient les algorithmes Ψ_1 et Ψ_2, $\frac{(d_1+1)(d_2+1)}{2}$ est la meilleure borne possible pour l'algorithme $\mathcal{A}_4$.*

preuve : Il existe toujours des motifs non corrigibles de poids $\leq \frac{d_1+1}{2}$ pour Ψ_1, ainsi que des motifs non corrigibles de poids $\leq \frac{d_2+1}{2}$ pour Ψ_2, donc il existe toujours un motif de poids $\leq \frac{(d_1+1)(d_2+1)}{4}$ non corrigible par $\mathcal{A}_4$.

Enfin si Ψ_1 et Ψ_2 sont bornés par la distance minimale, alors $\mathcal{A}_4$ est borné par $\frac{(d_1+1)(d_2+1)}{2}$.
□

2.2 L'algorithme de Reddy-Robinson

L'algorithme de Reddy-Robinson est plus sophistiqué que l'algorithme élémentaire, en revanche, il est borné par la distance minimale, donc est capable de corriger certain motifs non corrigibles par l'algorithme élémentaire.

On définit pour tout i, $0 \leq i < n_2$, le code $C_{2,i} \subset C_2$ engendré par un vecteur $\mathbf{e}_i$ dont la i-ième position est égale à 1.

Soient les isomorphismes

$$\begin{array}{rcccc} \theta_i & : & \mathbb{F}_q & \longrightarrow & C_{2,i} \\ & & x & \longmapsto & x\,\mathbf{e}_i \end{array}$$

et $\Theta_i = \theta_i^{n_1}$.

Comme dans la section 1.3, le code produit $\mathcal{C}_i = C_1 \otimes C_{2,i} = \Theta_i(C_1)$, est égal au code concaténé de $C_{2,i}$ et C_1. Et puisque $C_{2,i} \subset C_2$ on peut définir le décodeur partiel de Block-Zyablov de $\mathcal{C}_i$ par rapport à C_2 que nous noterons Ψ_i.

La construction de ce décodeur partiel nécessite la connaissance d'un supplémentaire $\overline{C}_{2,i}$ de $C_{2,i}$ dans C_2. En effet, la projection π_i de C_2 dans $C_{2,i}$ parallèlement à $\overline{C}_{2,i}$ est utilisée dans la définition de Ψ_i (cf. proposition III.7 page III.7).

Prenons pour $\overline{C}_{2,i}$ l'ensemble des mots de C_2 dont la i-ième composante est nulle. $\overline{C}_{2,i}$ est un $\mathbb{F}_q$-espace vectoriel et on a bien $C_{2,i} \oplus \overline{C}_{2,i} = C_2$.

Définition IV.3 *Définissons Ψ pour $M \in \mathcal{M}_q(n_2, n_1)$ par*

$$\Psi(M) = \begin{pmatrix} \Theta_0^{-1}(\Psi_0(M)) \\ \vdots \\ \Theta_{n_2-1}^{-1}(\Psi_{n_2-1}(M)) \end{pmatrix}.$$

L'algorithme de décodage d'erreur Ψ est appelé décodeur de Reddy-Robinson *de $\mathcal{C}$.*

On a le résultat suivant.

Proposition IV.15 *Le décodeur de Reddy-Robinson d'un code produit $\mathcal{C} = C_1(n_1, k_1, d_1) \otimes C_2(n_2, k_2, d_2)$ est borné par la distance minimale $d_1 d_2$.*

preuve : Pour tout i, $0 \leq i < n_2$, Ψ_i le décodeur partiel de Block-Zyablov de $\mathcal{C}_i$ est borné par $d_1 d_2$. Ici nous identifions $\mathbb{F}_q^{n_2 n_1} \equiv \mathcal{M}_q(n_2, n_1)$, et $\overline{C}_{2,i}^{\,n_1} \equiv \mathbb{F}_q^{n_1} \otimes \overline{C}_{2,i}$. La proriété (III.12) s'énonce alors pour tout $M \in \mathcal{M}_q(n_2, n_1)$, pour tout $\mathbf{x} \in \mathcal{C}_i$ et pour tout $M' \in \mathbb{F}_q^{n_1} \otimes \overline{C}_{2,i}$

$$d_H(\mathbf{x} + M', M) < \frac{d_1 d_2}{2} \Rightarrow \Psi_i(M) = \mathbf{x}. \tag{IV.12}$$

Soit la matrice $A \in C_1 \otimes C_2$,

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,n_1-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,n_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_2-1,0} & a_{n_2-1,1} & \dots & a_{n_2-1,n_1-1} \end{pmatrix}.$$

On veut montrer que Ψ est borné par $d_1 d_2$, c'est-à-dire

$$\forall\, M \in \mathcal{M}_q(n_2, n_1),\ d_H(A, M) < \frac{d_1 d_2}{2} \Rightarrow \Psi(M) = A.$$

Soit $M \in \mathcal{M}_q(n_2, n_1)$, tel que

$$d_H(A, M) < \frac{d_1 d_2}{2}.$$

Posons $\mathbf{a}_i = (a_{i,0}, \ldots, a_{i,n_1-1})$, la matrice $A - \Theta_i(\mathbf{a}_i)$ est dans $C_1 \otimes \overline{C}_{2,i}$, car $A \in C_1 \otimes C_2$ et $\Theta_i(\mathbf{a}_i) \in C_1 \otimes C_2$, et ces deux matrices ont la même i-ième ligne. Donc $\Psi_i(M) = \Theta_i(\mathbf{a}_i)$ pour tout i. On a donc par définition

$$\Psi(M) = \begin{pmatrix} \Theta_0^{-1}(\Psi_0(M)) \\ \vdots \\ \Theta_{n_2-1}^{-1}(\Psi_{n_2-1}(M)) \end{pmatrix} = \begin{pmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_{n_2-1} \end{pmatrix} = A.$$

□

Cet algorithme est donné dans le cas binaire dans [79]. On peut en donner une description différente, qui est une généralisation de l'algorithme donné dans cet article par Reddy et Robinson pour $\mathbb{F}_2$.

Le mot reçu est une matrice $n_2 \times n_1$.

1. On décode à l'aide d'un algorithme de décodage de C_2 les n_1 colonnes, soit e_j le nombre d'erreurs décodées dans la j-ième colonne, on pose

$$\forall\, j,\ 0 \le j < n_1,\ w_j = \begin{cases} e_j & \text{si } e_j \le \frac{d_2}{2} \\ \frac{d_2}{2} & \text{si } e_j > \frac{d_2}{2} \text{ ou si le décodage a échoué.} \end{cases}$$

2. On décode ensuite chacune des lignes à l'aide d'un décodeur d'erreur et d'effacement de C_1, en tenant compte des w_j obtenus pour chaque colonne.

   Comme pour l'algorithme de Block-Zyablov, il y aura plusieurs branches. Dans chacune d'elle, on remplace progressivement par des effacements les positions les moins "sûres", c'est-à-dire celles correspondant aux w_j les plus grands.

   Plus précisement, si $\mathbf{y} = (y_0, \ldots, y_{n_1-1})$ est la i-ième ligne après décodage des colonnes, on choisira la branche donnant comme résultat un $\mathbf{x} = (x_0, \ldots, x_{n_1-1}) \in C_1$ tel que

$$W_{\mathbf{y}}(\mathbf{x}) < \frac{d_1 d_2}{2},$$

   où

$$W_{\mathbf{y}}(\mathbf{x}) = \sum_{j=0}^{n_1-1} W_j(x_j),$$

   et

$$\forall\, j,\ W_j(x_j) = \begin{cases} w_j & \text{si } x_j = y_j \\ d_2 - w_j & \text{sinon} \end{cases}$$

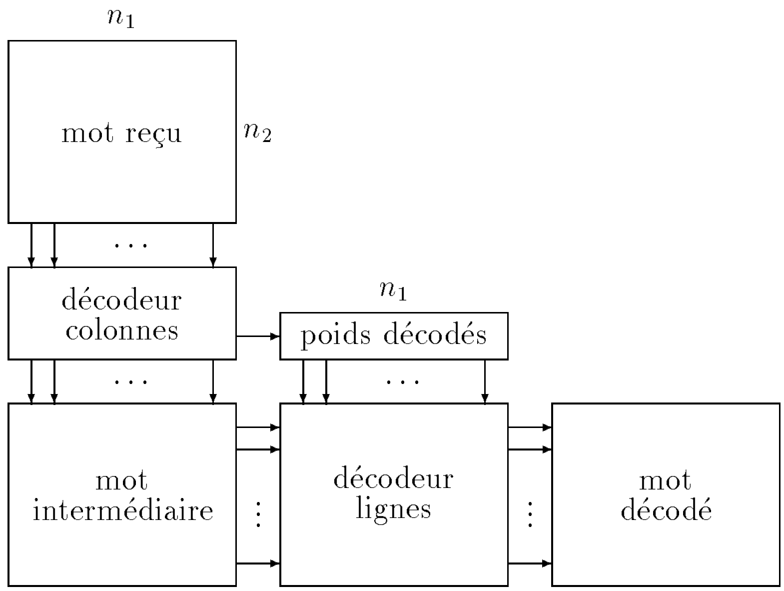


Figure IV.1 : Décodeur de Reddy-Robinson

Les deux étapes de cet algorithme sont résumée dans le diagramme de la figure IV.1

Remarque IV.4 Dans l'algorithme de Reddy-Robinson les lignes et les colonnes ne jouent pas un rôle symétrique comme c'est le cas dans l'algorithme élémentaire. Il est donc possible, si le décodage a échoué dans un sens, de recommencer après avoir transposé la matrice et en permutant les décodeurs des lignes et des colonnes.

3 Polynôme des motifs d'effacement corrigibles

Considérons le code produit $\mathcal{C} = C_1(n_1, k_1, d_1) \otimes C_2(n_2, k_2, d_2)$, et supposons que les seules perturbations pouvant intervenir dans un mot de code sont des effacements.

La notion de schème et de schème corrigible que nous définissons dans cette section est proche de la définition de motif que nous donnons dans le chapitre I.

3.1 Algorithme de décodage d'effacement

On considère γ_1 et γ_2 deux algorithmes de décodage d'effacement de C_1 et C_2 capable de corriger $d_1 - 1$ et $d_2 - 1$ effacements respectivement et jamais plus.

On considère l'algorithme de décodage suivant :

Algorithme $\mathcal{A}_5$

Étape 1. *décoder successivement chaque ligne puis chaque colonne à l'aide de γ_1 ou γ_2.*

Étape 2. *si la matrice ne contient plus d'effacements* $\rightarrow$ SUCCES

Étape 3. *si la matrice n'a pas été modifiée lors de l'étape 1* $\rightarrow$ ECHEC

Étape 4. $\rightarrow$ **Etape 1.**

On est assuré de la terminaison de cette algorithme car le nombre d'effacements va décroître strictement jusqu'à un certain rang, à partir duquel le motif d'effacement va rester le même, si ce motif est nul, alors le mot est décodé, sinon l'algorithme échoue. Dans tous les cas l'algorithme s'arrète à ce rang.

3.2 Schèmes corrigibles

Définition IV.4

1. *Un schème est un élément de $\mathcal{M}_2(n_2, n_1)$.*
2. *Le* poids *d'un schème est son nombre de positions égales à "1".*
3. *On appelle* grille *d'un schème la plus petite sous-matrice $m_2 \times m_1$ contenant le support du schème. On dit que la grille est de* taille $m_2 \times m_1$.

Remarque IV.5 Nous utilisons $\mathcal{M}_2(n_2, n_1)$, c'est-à-dire l'ensemble des matrices $n_2 \times n_1$ sur $\mathbb{F}_2$ pour décrire les schèmes. Nous n'utilisons cette définition que par commodité ; en effet pour les énoncés donnés ici, il suffit que le schème soit un tableau à n_2 lignes et n_1 colonnes sur un alphabet de 2 symboles.

Définition IV.5 *On dit que le schème M contient le schème M' et on note $M \supset M'$, si le support de M contient le support de M'.*

Soit un couple d'entiers (t_1, t_2), vérifiant $0 < t_1 \leq n_1$ et $0 < t_2 \leq n_2$.

Définition IV.6 *On dit qu'un schème M* est réductible *pour le couple (t_1, t_2) s'il possède une ligne non nulle de poids $\leq t_1$, ou une colonne non nulle de poids $\leq t_2$. Soit M' le schème obtenu en remplaçant cette ligne ou cette colonne par des "0", on note $M \rightarrow M'$.*

Remarque IV.6 Toutes les définitions et les propositions qui suivent sont valables pour un couple d'entier (t_1, t_2). Afin d'alléger ces énoncés nous avons préféré rendre cette référence implicite.

Définition IV.7 *On dit que le schème M* est réductible en M' en r étapes*, et on note $M \succeq M'$, s'il existe une famille de schèmes $(M_i)_{0\leq i\leq r}$, $r \geq 0$, tels que*

$$M = M_0 \to M_1 \to \ldots \to M_{r-1} \to M_r = M'.$$

Proposition IV.16 *La relation $\succeq$ est une relation d'ordre partiel sur $\mathcal{M}_2(n_2, n_1)$.*
L'ordre $\succeq$ est plus fin que l'ordre $\supset$, c'est-à-dire $M \succeq M' \Rightarrow M \supset M'$.

La preuve de cet énoncé est omise.

Définition IV.8

1. *On dit que M est* irréductible *si $M \succeq M' \Rightarrow M' = M$ (i.e. M est un élément minimal pour la relation $\succeq$).*

2. *On note $M' \rhd M$ si $M' \succeq M$ et M irréductible.*

Proposition IV.17 *Pour tout schème M, si $M \rhd M'$ et $M \rhd M''$ alors $M' = M''$. Le schème M' est alors appelé* réduit *de M et est noté $\widehat{M}$.*

preuve : Soit $M \to M_1$ la première étape de $M \rhd M'$.
Puisque $M \supset M''$, on a également $M_1 \supset M''$, sinon M'' serait réductible. L'égalité $M' = M''$ se montre ainsi aisement par récurrence sur le nombre d'étapes de $M \rhd M'$. □

Définition IV.9

1. *On dit que M est* corrigible *si $\widehat{M} = 0$.*

2. *On dit que M est* non corrigible *si $\widehat{M} \neq 0$.*

Proposition IV.18 *Soient M et M' deux schèmes. On a*

$$\left.\begin{array}{r} M \supset M' \\ M' = \widehat{M'} \end{array}\right\} \Rightarrow \widehat{M} \supset M'.$$

preuve : M est réduit en $\widehat{M}$ en r étapes, nous allons montrer le résultat par récurrence sur r.

- Si $r = 0$, $M = \widehat{M}$ est irréductible et le résultat est vrai.

- Supposons le résultat vrai pour $r-1$, $M \to M_1$ et M_1 est réductible en $\widehat{M}$ en $r-1$ étapes. Il suffit de montrer que $M_1 \supset M'$ pour avoir le résultat.

  Tout élément du support de M' est dans une ligne de poids $> t_1$ et une colonne de poids $> t_2$ dans M', sinon M' serait réductible. Puisque $M \supset M'$ c'est le cas aussi pour M.

  Donc aucun des éléments du support de M appartenant au support de M' n'est affecté par la réduction $M \to M_1$, donc $M_1 \supset M'$.

□

Corollaire IV.7 *Soient M et M' deux schèmes. On a*

$$M \supset M' \Rightarrow \widehat{M} \supset \widehat{M'}.$$

Proposition IV.19

1. *Un schème est non corrigible si et seulement si il contient un schème irréductible non nul.*

2. *Soit un schème irréductible dont la grille a pour taille $m_2 \times m_1$, alors $m_1 > t_1$, $m_2 > t_2$ et le schème est de poids $\geq \max(m_2(t_1+1), m_1(t_2+1))$.*

3. *Tout schème irréductible non nul est de poids $\geq (t_1+1)(t_2+1)$.*

preuve :

1. Soit un schème M non corrigible, alors $M \supset \widehat{M} \neq 0$. D'autre part si $M \supset M'$, et M' est irréductible non nul, alors $\widehat{M} \supset M'$, donc $\widehat{M} \neq 0$.

2. Soit un schème irréductible dont la grille a pour taille $m_2 \times m_1$, on a $m_1 > t_1$ et $m_2 > t_2$, sinon le schème serait réductible. De plus chacunes des m_2 lignes de la grille a un poids $\geq (t_1+1)$, et chacunes des m_1 colonnes a un poids $\geq (t_2+1)$, donc le schème a un poids $\geq \max(m_2(t_1+1), m_1(t_2+1))$.

3. Il résulte immédiatement du point précédent que tout schème irréductible non nul est de poids au moins $(t_1+1)(t_2+1)$.

□

3.3 Quelques résultats sur les motifs de poids faible

A un motif d'erreur ou d'effacement, on fait correspondre le schème qui comporte des "1" en tous les éléments de son support. On confondra dans la suite le motif et le schème.

Définition IV.10 *On appellera* séquence de décodage *d'effacement pour un motif d'effacement du code produit $\mathcal{C} = C_1 \otimes C_2$, une suite finie de décodage effectif d'une ligne par γ_1 ou d'une colonne par γ_2.*

On dira qu'une séquence de décodage est achevée *pour un motif d'effacement si aucun effacement ne peut plus être corrigé par le décodage d'une ligne ou d'une colonne.*

On dira qu'une séquence de décodage est correctrice *pour un motif d'effacement M si le motif résultant de M par la séquence est le motif nul.*

Proposition IV.20 *Soit M un motif d'effacement pour le code produit $C_1 \otimes C_2$, M est corrigible pour l'algorithme $\mathcal{A}_5$ si et seulement si son schème correspondant est corrigible pour $(d_1 - 1, d_2 - 1)$.*

preuve : Il est clair qu'une réduction "$\rightarrow$" est strictement équivalente au décodage d'une ligne ou d'une colonne, donc $M \rhd 0$ si et seulement si le motif d'effacement M est corrigible. □

Corollaire IV.8 *Toute séquence de décodage achevée agissant sur un même motif d'effacement aboutit au même résultat. Le schème correspondant à ce résultat est irréductible.*

Proposition IV.21 *Il n'existe aucun motif d'effacement non corrigible de poids $< d_1 d_2$. Le nombre de motifs d'effacement non corrigibles de poids $d_1 d_2$ est égal à*

$$\binom{n_1}{d_1}\binom{n_2}{d_2}.$$

preuve : Le schème correspondant à un motif non corrigible contient un schème irréductible, donc d'après la proposition IV.19, il est de poids $\geq d_1 d_2$.

Soit un motif d'effacement non corrigible de poids $d_1 d_2$, on considére le schème correspondant, il est de poids $d_1 d_2$, donc d'après la proposition IV.19 il est irréductible et sa grille est de taille $d_2 \times d_1$, donc son support est égal à sa grille.

Le nombre de motifs d'effacement corrigibles de poids $d_1 d_2$ est donc égal au nombre de grille de taille $d_2 \times d_1$, soit

$$\binom{n_1}{d_1}\binom{n_2}{d_2}.$$

□

Proposition IV.22 *Le nombre de motifs d'effacement non corrigibles de poids w,*

$$d_1 d_2 \leq w < d_1 d_2 + \min(d_1, d_2)$$

est égal à

$$\binom{n_1}{d_1}\binom{n_2}{d_2}\binom{n_1 n_2 - d_1 d_2}{w - d_1 d_2}.$$

preuve : Tout schème non corrigible contient un schème irréductible non nul, la grille de ce schème irréductible est de taille $d_2 \times d_1$, sinon son poids serait $\geq d_2 d_1 + \min(d_1, d_2)$.

Le support de tout motif d'effacement de poids $w < d_1 d_2 + \min(d_1, d_2)$ contient donc une sous-matrice $d_2 \times d_1$, et $w - d_1 d_2$ autres effacements ont une position quelconque parmi $n_1 n_2 - d_1 d_2$. D'où le résultat. □

3.3.1 Cas particulier : $C_1 = C_2$

On suppose que $C_1 = C_2 = C(n, k, d)$.

Proposition IV.23 *Le nombre de motifs d'effacement non corrigibles de poids w tel que*

$$d^2 + d \leq w < d^2 + 2d,$$

est majoré par

$$\binom{n}{d}^2 \binom{n^2 - d^2}{w - d^2} + \binom{n}{d+1}^2 (d+1)! \binom{n^2 - d^2 - d}{w - d^2 - d}.$$

Nous allons commencer par établir les trois résultats suivants :

Lemme IV.3 *Tout motif d'effacement non corrigible de poids $w < d^2 + 2d$ contient un schème irréductible dont la grille est de taille $m_2 \times m_1$ avec $d \leq m_1 \leq d+1$ et $d \leq m_2 \leq d+1$.*

preuve : Tout motif non corrigible M contient un schème irréductible non nul M'. Soit $m_2 \times m_1$ la taille de M', si $m_1 \geq d + 2$ ou $m_2 \geq d + 2$. D'après la proposition IV.19, le poids w' de M' vérifie $w' \geq d(d + 2)$. Or $w' \leq w$ donc $m_1 < d + 2$ et $m_2 < d + 2$. □

Lemme IV.4 *Tout motif d'effacement non corrigible de poids $w < d^2 + 2d$ contient un schème irréductible dont la grille est de taille $d \times d$ ou $(d + 1) \times (d + 1)$.*

preuve : En effet, $m_2 \times m_1 \in \{d \times d, d \times (d+1), (d+1) \times d, (d+1) \times (d+1)\}$. Et un schème irréductible dont la grille a pour taille $d \times (d+1)$ ou $(d+1) \times d$ contient un schème irréductible dont la grille a une taille $d \times d$. Donc un motif d'effacement non corrigible contient un schème irréductible dont la grille est de taille $d \times d$ ou $(d + 1) \times (d + 1)$. □

Lemme IV.5 *Tout schème irréductible dont la grille est de taille $(d+1) \times (d+1)$ contient un schème irréductible de poids $d^2 + d$ dont la grille est de taille $(d+1) \times (d+1)$ et dont le support est le complémentaire par rapport à la grille d'une matrice de permutation $(d + 1) \times (d + 1)$.*

preuve : On considère un schème irréductible dont la grille est de taille $(d + 1) \times (d + 1)$, chacune des $d + 1$ lignes et $d + 1$ colonnes de ce schème comporte au moins d "1" et au plus $d+1$. Donc le complémentaire du schème par rapport à sa grille est une matrice $(d+1) \times (d+1)$ comportant au plus un "1" dans chaque ligne et chaque colonne. C'est-à-dire que son support est inclus dans celui d'une matrice de permutation $(d + 1) \times (d + 1)$. □

preuve : (de la proposition IV.23)

Le nombre de schèmes irréductibles dont la grille est de taille $d \times d$ est égal à $\binom{n}{d}^2$. Donc le nombre de motifs d'effacement de poids w contenant un schème irréductible dont la grille est de taille $d \times d$ est majoré par

$$\binom{n}{d}^2 \binom{n^2 - d^2}{w - d^2},$$

On obtient seulement une majoration car certains motifs sont comptés plusieurs fois. Par exemple un motif dont le support est une sous-matrice $d \times (d+1)$ est compté $d+1$ fois.

Le nombre de schèmes irréductibles dont la grille est de taille $(d+1) \times (d+1)$ et dont le support est le complémentaire par rapport à la grille d'une matrice de permutation $(d+1) \times (d+1)$ est égal à $\binom{n}{d+1}^2 (d+1)!$. Donc le nombre de motifs d'effacement de poids w contenant un tel schème est majoré par

$$\binom{n}{d+1}^2 (d+1)! \binom{n^2 - d^2 - d}{w - d^2 - d},$$

d'où le résultat. $\square$

Exemple : Soit $RS(14, 7, 8)$ le code de Reed-Solomon de paramètres $(14, 7, 8)$ sur $\mathbb{F}_{16}$, on considère le code produit $RS(14, 7, 8) \otimes RS(14, 7, 8)$, ce code est linéaire sur $\mathbb{F}_{16}$, de paramètres $(196, 49, 64)$.

Le plus petit motif d'effacement non corrigible pour ce code est de poids 64, et on en aura $\binom{14}{8}^2 = 9018009$. Le nombre total de motifs d'effacement de poids 64 étant $\binom{196}{64}$, la proportion de motifs de poids 64 non corrigibles vaudra :

$$\frac{\binom{14}{8}^2}{\binom{196}{64}} \approx 2.10^{-46}.$$

On connait également le nombre de motifs d'effacement non corrigibles jusqu'au poids 72 exclus, ces valeurs sont données dans le tableau IV.1.

On constate que ces valeurs sont très faibles, on peut même affirmer que "en pratique" tous les motifs d'effacement de poids < 80 sont corrigibles.

3.4 Une borne théorique

Soit $P_0(x)$ le polynôme des motifs d'effacement de C_1 corrigibles par γ_1 et $P_1(x)$ le polynôme des motifs d'effacement de C_1 non corrigibles par γ_1.

Proposition IV.24 *Tout motif d'effacement de $C_1 \otimes C_2$ possédant strictement moins de d_2 lignes non corrigible par γ_1 est corrigible.*

Le polynôme énumérateur des poids de ces motifs est égal à :

$$P(x) = \sum_{i=0}^{d_2 - 1} \binom{n_2}{i} P_1(x)^i P_0(x)^{n_2 - i}.$$

poids	nombre de motifs non corrigibles	total	rapport
64	$\binom{14}{8}^2$	$\binom{196}{64}$	$\approx 2\,10^{-46}$
65	$\binom{14}{8}^2\binom{132}{1}$	$\binom{196}{65}$	$\approx 10^{-44}$
66	$\binom{14}{8}^2\binom{132}{2}$	$\binom{196}{66}$	$\approx 5\,10^{-43}$
67	$\binom{14}{8}^2\binom{132}{3}$	$\binom{196}{67}$	$\approx 10^{-41}$
68	$\binom{14}{8}^2\binom{132}{4}$	$\binom{196}{68}$	$\approx 2\,10^{-40}$
69	$\binom{14}{8}^2\binom{132}{5}$	$\binom{196}{69}$	$\approx 3\,10^{-39}$
70	$\binom{14}{8}^2\binom{132}{6}$	$\binom{196}{70}$	$\approx 3\,10^{-38}$
71	$\binom{14}{8}^2\binom{132}{7}$	$\binom{196}{71}$	$\approx 3\,10^{-37}$
72	$\leq \binom{14}{8}^2\binom{132}{8} + \binom{14}{9}^2 9!$	$\binom{196}{72}$	$\leq 3\,10^{-36}$
73	$\leq \binom{14}{8}^2\binom{132}{9} + \binom{14}{9}^2 9!\binom{124}{1}$	$\binom{196}{73}$	$\leq 2\,10^{-35}$
74	$\leq \binom{14}{8}^2\binom{132}{10} + \binom{14}{9}^2 9!\binom{124}{2}$	$\binom{196}{74}$	$\leq 2\,10^{-34}$
75	$\leq \binom{14}{8}^2\binom{132}{11} + \binom{14}{9}^2 9!\binom{124}{3}$	$\binom{196}{75}$	$\leq 10^{-33}$
76	$\leq \binom{14}{8}^2\binom{132}{12} + \binom{14}{9}^2 9!\binom{124}{4}$	$\binom{196}{76}$	$\leq 8\,10^{-33}$
77	$\leq \binom{14}{8}^2\binom{132}{13} + \binom{14}{9}^2 9!\binom{124}{5}$	$\binom{196}{77}$	$\leq 5\,10^{-32}$
78	$\leq \binom{14}{8}^2\binom{132}{14} + \binom{14}{9}^2 9!\binom{124}{6}$	$\binom{196}{78}$	$\leq 3\,10^{-31}$
79	$\leq \binom{14}{8}^2\binom{132}{15} + \binom{14}{9}^2 9!\binom{124}{7}$	$\binom{196}{79}$	$\leq 10^{-30}$

Tableau IV.1 : Motifs d'effacements non corrigibles pour le code $RS(14,7,8) \otimes RS(14,7,8)$

preuve : Soit un motif d'effacement de $C_1 \otimes C_2$ vérifiant les hypothèses de l'énoncé. On corrige d'abord ses lignes. A l'issue de cette opération il reste au plus $d_2 - 1$ lignes contenant des effacements. Donc chaque colonne est corrigible car elle contient au plus $d_2 - 1$ effacements.

Soient $P_0(x)$ le polynôme des motifs d'effacement de C_1 corrigibles par γ_1 et $P_1(x)$ le polynôme des motifs d'effacement de C_1 non corrigibles par γ_1. Le nombre de motifs d'effacement de $C_1 \otimes C_2$ possédant exactement i lignes non corrigibles est égal à :

$$\binom{n_2}{i} P_1(x)^i P_0(x)^{n_2 - i}.$$

En sommant ce résultat pour $i = 0, \ldots, d_2 - 1$, on obtient l'énumérateur des poids désiré, soit

$$P(x) = \sum_{i=0}^{d_2-1} \binom{n_2}{i} P_1(x)^i P_0(x)^{n_2 - i}.$$

□

L'énumérateur des poids des motifs possédant au plus $d_2 - 1$ lignes non corrigibles est inférieur au polynôme des motifs corrigibles de $C_1 \otimes C_2$ par l'algorithme $\mathcal{A}_5$. C'est-à-dire que pour tout i, $0 \leq i \leq n_1 n_2$, si S_i est le nombre de motifs d'effacement corrigible de poids i, alors

$$S_i \geq [x^i]P(x).$$

La proposition ci-dessus nous donne donc une borne inférieure du nombre de motifs corrigibles.

Exemple : Appliquons ce résultat au code $RS(14, 7, 8) \otimes RS(14, 7, 8)$, les polynômes des motifs d'effacement corrigibles et non corrigibles valent respectivement

$$P_0(x) = 3003x^8 + 2002x^9 + 1001x^{10} + 364x^{11} + 91x^{12} + 14x^{13} + x^{14},$$

et

$$P_1(x) = 1 + 14x + 91x^2 + 364x^3 + 1001x^4 + 2002x^5 + 3003x^6 + 3432x^7.$$

Ces deux polynômes nous permettent de calculer une borne supérieure du nombre de motifs non corrigibles de poids compris entre 64 et 100. Les résultats sont donnés dans le tableau IV.2. On constate que ces bornes sont très éloignées de celles obtenues au paragraphe précédent, et qui sont données dans le tableau IV.1.

Nous ne donnons pas les valeurs pour des poids supérieurs à 100 celles-ci devennant trop élevées n'ont pas d'intéret pratique. On verra dans la section 5 que pour ce code les valeurs obtenues par simulation sont nettement inférieures (cf. tableau IV.4).

3.5 Mise en place de la simulation – capacité de correction

Pour le décodage d'effacement nous prenons comme modèle un canal à effacement symétrique sans mémoire dans lequel chaque symbole transmis a la même probabilité p d'être effacé.

3.5.1 Taux d'effacement résiduel

Définition IV.11 *On appelle* taux d'effacement résiduel *d'un code pour un algorithme de décodage d'effacement et un canal donné, la probabilité pour qu'un mot de code transmis à travers le canal ne soit pas décodé.*

Proposition IV.25 *Soit un code linéaire C de longueur n sur $\mathbb{F}_q$ muni d'un algorithme γ de décodage d'effacements et soit $P(x)$ le polynôme des motifs d'effacement corrigibles de C pour γ. $Q(x) = (1 + x)^n - P(x)$ est le polynôme des motifs non corrigibles.*

Le taux d'effacement résiduel de C pour un canal de probabilité d'effacement p est égal à :

$$p_{eff} = (1 - p)^n Q\left(\frac{p}{1 - p}\right) = 1 - (1 - p)^n P\left(\frac{p}{1 - p}\right). \quad \text{(IV.13)}$$

poids	proportion de motifs non corrigibles	poids	proportion de motifs non corrigibles
< 64	0	64	$< 6\,10^{-22}$
65	$< 2\,10^{-20}$	66	$< 5\,10^{-19}$
67	$< 8\,10^{-18}$	68	$< 9\,10^{-17}$
69	$< 9\,10^{-16}$	70	$< 7\,10^{-15}$
71	$< 5\,10^{-14}$	72	$< 3\,10^{-13}$
73	$< 10^{-12}$	74	$< 7\,10^{-12}$
75	$< 3\,10^{-11}$	76	$< 10^{-10}$
77	$< 5\,10^{-10}$	78	$< 2\,10^{-9}$
79	$< 6\,10^{-9}$	80	$< 2\,10^{-8}$
81	$< 6\,10^{-8}$	82	$< 2\,10^{-7}$
83	$< 5\,10^{-7}$	84	$< 10^{-6}$
85	$< 4\,10^{-6}$	86	$< 9\,10^{-6}$
87	$< 2\,10^{-5}$	88	$< 5\,10^{-5}$
89	$< 10^{-4}$	90	$< 2\,10^{-4}$
91	$< 5\,10^{-4}$	92	$< 9\,10^{-4}$
93	$< 2\,10^{-3}$	94	$< 3\,10^{-3}$
95	$< 6\,10^{-3}$	96	$< 10^{-2}$
97	$< 2\,10^{-2}$	98	$< 3\,10^{-2}$
99	$< 4\,10^{-2}$	100	$< 7\,10^{-2}$
> 100	—		

Tableau IV.2 : Borne théorique pour les motifs d'effacement non corrigibles du code $RS(14,7,8) \otimes RS(14,7,8)$

preuve : Si S_i est le nombre de motifs d'effacement corrigibles de poids i, le taux d'effacement résiduel est égal à

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

On a $P(x) = \sum_{i=0}^{n} S_i x^i$, et on en déduit facilement le résultat. □

3.5.2 La simulation

Il n'est malheureusement pas toujours possible de déterminer théoriquement le polynôme des motifs d'effacement corrigibles; en particulier pour les codes produit, seuls les motifs de poids faible (proche de la distance minimale) peuvent être décrits de façon à être comptés facilement.

Le polynôme des motifs corrigibles peut être calculé par simulation, pour chaque poids d'effacement i, on tire aléatoirement un grand nombre de motif M, soit $\tilde{S}_i$ le nombre de motifs corrigibles, on obtient alors une estimation du nombre S_i de motifs d'effacement corrigibles pour un code de longueur n,

$$S_i \approx \binom{n}{i} \frac{\tilde{S}_i}{M}.$$

Nous avons mis en place une simulation pour les codes produit à partir de l'algorithme $\mathcal{A}_5$, le résultat suivant limite le champ d'investigation pour les poids des motifs d'effacement.

Proposition IV.26 *On suppose que C_1 et C_2 ont tous deux un algorithme de correction d'effacement capable de corriger tous les motifs de poids strictement inférieur à d_1 et d_2 respectivement, et aucun autre.*

Soit $P(x) = \sum_{i=0}^{n_1 n_2} S_i x^i$ le polynôme des motifs d'effacement corrigibles de $\mathcal{C} = C_1 \otimes C_2$ pour l'algorithme $\mathcal{A}_5$, on a

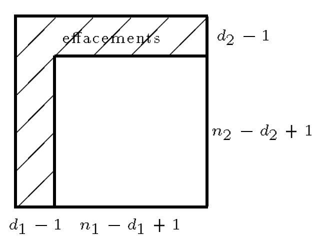
$$\begin{aligned} 0 \leq i < d_1 d_2, \quad & S_i = \binom{n_1 n_2}{i}, \\ d_1 d_2 \leq i \leq U, \quad & 0 < S_i < \binom{n_1 n_2}{i}, \\ U < i \leq n_1 n_2, \quad & S_i = 0, \end{aligned}$$

où $U = n_2 n_2 - (n_1 - d_1 + 1)(n_2 - d_2 + 1)$.

preuve : Si $i < d_1 d_2$, tous les motifs sont corrigibles d'après la proposition IV.21, donc $S_i = \binom{n_1 n_2}{i}$.

Si $i \geq d_1 d_2$. Comme il existe des motifs non corrigibles de poids $d_1 d_2$, cela reste vrai à fortiori pour le poids i, donc $S_i < \binom{n_1 n_2}{i}$.

Si $i = U$, le motif d'effacement dont le support est constitué des $d_2 - 1$ premières lignes et des $d_1 - 1$ premières colonnes est de poids U et est corrigible,



donc $S_U > 0$.

- Si C_1 et C_2 sont MDS, alors $k_1 = n_1 - d_1 + 1$ et $k_2 = n_2 - d_2 + 1$, donc $U = n_1 n_2 - k_1 k_2$. Et si un motif d'effacement a un poids $> U$, il reste moins de $k_1 k_2$ symboles d'information, ce qui ne peut pas être suffisant pour un code de dimension $k_1 k_2$.
- Les hypothèses faites sur les décodeurs de C_1 et C_2 (i.e. pas de décodage au delà de la distance minimale) impliquent que tout se passe comme dans le cas MDS.

donc, si $i > U$, $S_i = 0$. □

3.5.3 Décodage au delà de la distance minimale

La proposition IV.26, nous assure que le décodage est possible jusqu'à la distance minimale, mais surtout qu'il est possible au delà. Pour mesurer le gain apporté par les motifs de poids élevé nous utiliserons la capacité de correction définie dans la section 5.3 du chapitre I.

Pour un algorithme de décodage d'effacement la définition de la capacité de décodage est la suivante.

Définition IV.12 *Soit C un code linéaire sur $\mathbb{F}_q$ de longueur n et soit $P(x)$ son polynôme des motifs d'effacement corrigibles pour un algorithme de décodage d'effacement $\mathcal{A}$. On pose*

$$P_{d^*}(x) = \sum_{i=0}^{d^*-1} \binom{n}{i} x^i.$$

La capacité de correction *de C pour l'algorithme $\mathcal{A}$ et pour la probabilité d'effacement p est le plus grand entier d^* tel que*

$$P_{d^*}(\frac{p}{1-p}) \leq P(\frac{p}{1-p}) < P_{d^*+1}(\frac{p}{1-p}).$$

Le taux d'effacement résiduel du code C pour un canal de probabilité d'effacement p, dont le polynôme des motifs d'effacement corrigibles est égal à $P(x)$, est égal à $1-(1-p)^n P(\frac{p}{1-p})$.

D'autre part $P_{d^*}(x)$ est le polynôme des motifs d'effacement corrigibles d'un code fictif C^* de longueur n, de distance minimale d^* et muni d'un algorithme de décodage d'effacement corrigeant exactement tous les motifs de poids inférieur à sa distance minimale.

La capacité de correction de C est donc la distance minimale du code C^* tel que son taux d'effacement résiduel soit immédiatement supérieur à celui de C, pour une probabilité d'effacement donnée p.

4 Polynôme des motifs d'erreur corrigibles

4.1 L'algorithme élémentaire

4.1.1 Quel lien avec les effacements ?

Soient $C_1(n_1, k_1, d_1)$ et $C_2(n_2, k_2, d_2)$ deux codes linéaires sur $\mathbb{F}_q$, γ_1 et γ_2 des algorithmes de décodage d'erreur bornés strictement par la distance minimale. On pose

$$t_1 = \frac{d_1 - 1}{2} \text{ et } t_2 = \frac{d_2 - 1}{2}.$$

Définition IV.13 *On appellera* séquence de décodage *d'erreur pour un motif d'erreur du code produit $\mathcal{C} = C_1 \otimes C_2$, une suite finie de décodage effectif d'une ligne par γ_1 ou d'une colonne par γ_2.*

On dira qu'une séquence de décodage est achevée *pour un motif d'erreur si aucune erreur ne peut plus être corrigé par le décodage d'une ligne ou d'une colonne.*

On dira qu'une séquence de décodage est correctrice *pour un motif d'erreur M si le motif résultant de M par la séquence est le motif nul.*

Soient C_1^* et C_2^*, deux codes linéaires sur $\mathbb{F}_q$ de longueur n_1 et n_2 respectivement, muni d'algorithmes de décodage d'effacement γ_1^* et γ_2^* capable de décoder exactement tous les motifs d'effacement de poids t_1 et t_2 ou moins respectivement.

Proposition IV.27 *Soit M un motif d'erreur pour $C_1 \otimes C_2$, soit M^* le motif d'effacement ayant même support que M.*

Si le motif d'effacement M^* est corrigible pour le code produit $\mathcal{C}^* = C_1^* \otimes C_2^*$ muni de l'algorithme $\mathcal{A}_5$ construit à l'aide de γ_1^* et γ_2^*, alors il existe une séquence de décodage correctrice pour M.

preuve : On a $\mathcal{C} = C_1(n_1, k_1, d_1) \otimes C_2(n_2, k_2, d_2)$ et $\mathcal{C}^* = C_1^*(n_1, k_1^*, t_1+1) \otimes C_2^*(n_2, k_2^*, t_2+1)$, c'est-à-dire que la capacité de décodage d'erreur de C_i est la même que la capacité de décodage d'effacement de C_i^*, pour $i = 1, 2$.

On considère le schème correspondant à M et M^*, puisque M^* est un motif d'effacement corrigible, d'après la proposition IV.20, ce schème est corrigible pour (t_1, t_2), il existe donc une suite de réduction permettant de corriger le schème.

Les algorithmes de décodage d'erreur γ_1 et γ_2 de C_1 et C_2 peuvent corriger tout motif d'erreur de poids inférieur ou égal respectivement à t_1 et t_2, donc la suite de réduction nous permet de définir une séquence de décodage d'erreur correctrice pour M. □

Remarque IV.7

1. L'existence d'une séquence de décodage d'erreur correctrice ne garantit pas que le motif soit corrigible.

2. Il n'y a pas équivalence entre un motif d'erreur corrigible et un motif d'effacement corrigible ayant même support, comme l'illustrent les deux exemples ci-dessous.

   Exemple : Dans les exemples suivants on considère des codes binaires.

   1. Un motif d'effacement non corrigible pour $\mathcal{C}^*$ peut être corrigible pour $\mathcal{C}$ par l'algorithme élémentaire : considérons le produit de deux codes binaires 2-correcteurs et le motif d'erreur

$$M = \begin{pmatrix} 1111100\dots \\ 1110000\dots \\ 1110000\dots \\ 0000000\dots \\ \dots \end{pmatrix}$$

      on suppose que le motif d'erreur $111000\dots$ n'est pas corrigible pour C_1, et que $0111110\dots$ est un mot de C_1. On a alors après correction des lignes

$$M \xrightarrow{li} \begin{pmatrix} 0111110\dots \\ 1110000\dots \\ 1110000\dots \\ 0000000\dots \\ \dots \end{pmatrix}$$

puis après correction des colonnes

$$M \xrightarrow{li} (\ ) \xrightarrow{col} \begin{pmatrix} 0110000\ldots \\ 0110000\ldots \\ 0110000\ldots \\ 0??0000\ldots \\ \ldots \end{pmatrix}$$

dans l'étape suivante, correction des lignes, il reste au plus 2 erreurs dans chaque ligne donc M est corrigible.

D'autre part M contient un motif 3×3 et est donc non corrigible pour un algorithme de décodage d'effacement corrigeant 2 effacements seulement.

Remarquons que l'ordre de décodage a une importance dans le cas des erreurs, si on décode dans M d'abord les colonnes on a

$$M \xrightarrow{col} \begin{pmatrix} 1110000\ldots \\ 1110000\ldots \\ 1110000\ldots \\ 0000000\ldots \\ \ldots \end{pmatrix} \xrightarrow{li} \begin{pmatrix} 1110000\ldots \\ 1110000\ldots \\ 1110000\ldots \\ 0000000\ldots \\ \ldots \end{pmatrix} \longrightarrow \text{fin}$$

2. A l'inverse il existe des motifs d'erreur non corrigible pour $\mathcal{C}$ qui sont des motifs d'effacement corrigibles pour $\mathcal{C}^*$, considérons le motif

$$M = \begin{pmatrix} 0111000\ldots \\ 0111100\ldots \\ 0011110\ldots \\ 0001110\ldots \\ 0000000\ldots \\ 0000000\ldots \\ \ldots \end{pmatrix}$$

le code C_1 est le même que dans l'exemple précédent, et on suppose que $111110\ldots$ est un mot de C_2, on a alors par décodage d'erreur des lignes puis des colonnes

$$M \xrightarrow{li} \begin{pmatrix} 0111110\ldots \\ 0111110\ldots \\ 0111110\ldots \\ 0111110\ldots \\ 0000000\ldots \\ 0000000\ldots \\ \ldots \end{pmatrix} \xrightarrow{col} \begin{pmatrix} 0111110\ldots \\ 0111110\ldots \\ 0111110\ldots \\ 0111110\ldots \\ 0111110\ldots \\ 0000000\ldots \\ \ldots \end{pmatrix} \longrightarrow \text{fin}$$

ce motif donc n'est pas corrigible.

Il est clair d'autre part que si on considère M comme un motif d'effacement, il est corrigible.

4.2 L'algorithme de Reddy-Robinson

On suppose ici que γ_2 est un algorithme de décodage d'erreur pour C_2 borné strictement par d_2, et que γ_1 est un algorithme de décodage d'erreur et d'effacement pour C_1 borné strictement par d_1.

Proposition IV.28 *Soit M un motif d'erreur pour $\mathcal{C}$, si M possède au moins d_1 colonnes non corrigibles par γ_2, alors M est non corrigible pour l'algorithme de Reddy-Robinson.*

preuve : L'algorithme de Reddy-Robinson procède de la façon suivante :

1. il décode chacune des n_1 colonnes numérotée de 0 à $n_1 - 1$. Lors de chacun de ces décodages, il décode e_i erreurs,

2. pour chacune des n_2 lignes, on effectue $t_2 + 1$ décodages, numérotés de 0 à t_2. Le j-ième décodage consiste à remplacer par un effacement toutes les positions i pour lesquelles le i-ième décodage de colonne a échoué, ou telles que $e_i > j$.

Donc l'algorithme échouera systématiquement si le nombre de colonnes non corrigibles par γ_2 est au moins égal à d_1. En effet, dans ce cas, pour toutes les lignes et pour les $t_2 + 1$ branches il y aura au moins d_1 effacements ce qui empèchera la correction car γ_1 est borné strictement par d_1. □

Le polynôme énumérateur de ces motifs non corrigibles est égal à

$$\sum_{i=d_1}^{n_1} P_1(x)^i (P_2(x) + P_0(x))^{n_1 - i},$$

où $P_0(x)$, $P_1(x)$ et $P_2(x)$ sont respectivement les polynômes des motifs corrigibles, non corrigibles et erronés de C_2 pour γ_2. Si $P_{\mathcal{C}}(x)$ est le polynôme des motifs corrigibles de $\mathcal{C}$, alors

$$P_{\mathcal{C}}(x) \preceq (1 + (q-1)x)^{n_1 n_2} - \sum_{i=d_1}^{n_1} P_1(x)^i (P_2(x) + P_0(x))^{n_1 - i}.$$

Exemple : Pour le code $RS(14,7,8) \otimes RS(14,7,8)$, on peut obtenir une borne inférieure du nombre de motifs d'erreur non corrigibles par l'algorithme de Reddy-Robinson à l'aide de la proposition ci-dessus. Les valeurs obtenues par le calcul sont données dans le tableau IV.3.

Ce tableau nous permet de donner une borne supérieure de la capacité de correction d'erreur du code pour cet algorithme, on obtient une borne de 105, lorsque le taux d'erreur résiduel (cf. définition IV.14) est voisin de 10^{-5}, ce qui correspond à une probabilité d'erreur $p = 0.147$. A titre de comparaison, on verra que les simulations pour le même code avec l'algorithme élémentaire donnent le même taux d'erreur résiduel pour $p = 0.191$; ce qui correspond à une capacité de correction d'erreur $d^* = 125$.

w	$1 - S_w/\binom{196}{w}$	w	$1 - S_w/\binom{196}{w}$	w	$1 - S_w/\binom{196}{w}$
31	0	46	≥ 0.0033	61	≥ 0.018
32	$\geq 6.5\,10^{-11}$	47	≥ 0.0050	62	≥ 0.017
33	$\geq 1.2\,10^{-9}$	48	≥ 0.0072	63	≥ 0.016
34	$\geq 1.1\,10^{-8}$	49	≥ 0.0097	64	≥ 0.015
35	$\geq 7.9\,10^{-8}$	50	≥ 0.012	65	≥ 0.014
36	$\geq 4.0\,10^{-7}$	51	≥ 0.015	66	≥ 0.014
37	$\geq 1.7\,10^{-6}$	52	≥ 0.017	67	≥ 0.013
38	$\geq 6.1\,10^{-6}$	53	≥ 0.019	68	≥ 0.012
39	$\geq 1.9\,10^{-5}$	54	≥ 0.020	69	≥ 0.011
40	$\geq 5.2\,10^{-5}$	55	≥ 0.021	70	≥ 0.010
41	$\geq 1.3\,10^{-4}$	56	≥ 0.021	71	≥ 0.010
42	$\geq 2.9\,10^{-4}$	57	≥ 0.021	72	≥ 0.0095
43	$\geq 6.1\,10^{-4}$	58	≥ 0.021	73	≥ 0.0089
44	≥ 0.0011	59	≥ 0.020	74	≥ 0.0084
45	≥ 0.0020	60	≥ 0.019	75	≥ 0.0079

Tableau IV.3 :Borne inférieure de la proportion de motifs d'erreur non corrigibles pour le décodage du code $RS(14,7,8) \otimes RS(14,7,8)$ par l'algorithme de Reddy-Robinson

4.3 Simulation – capacité de correction

4.3.1 Taux d'erreur résiduel

Définition IV.14 *On appelle* taux d'erreur résiduel *d'un code pour un algorithme de décodage d'erreur et un canal donné, la probabilité pour qu'un mot de code transmis à travers le canal ne soit pas décodé.*

Proposition IV.29 *Soit un code linéaire C de longueur n sur $\mathbb{F}_q$ muni d'un algorithme γ de décodage d'erreur, soit $P(x)$ le polynôme des motifs d'erreur corrigibles de C pour γ, $Q(x) = (1+(q-1)x)^n - P(x)$ est le polynôme des motifs non corrigibles.*

Le taux d'erreur résiduel de C pour un canal de probabilité d'erreur p est égal à :

$$p_{err} = (1-p)^n Q(\frac{p}{(q-1)(1-p)}) = 1 - (1-p)^n P(\frac{p}{(q-1)(1-p)}). \tag{IV.14}$$

preuve : Si S_i est le nombre de motifs d'erreur corrigibles de poids i, le taux d'erreur résiduel est égal à

$$1 - \sum_{i=0}^{n} S_i \left(\frac{p}{q-1}\right)^i (1-p)^{n-i},$$

on a $P(x) = \sum_{i=0}^{n} S_i x^i$, et on en déduit facilement le résultat. □

4.3.2 La simulation

Comme pour les effacements, étant donné l'impossibilité d'obtenir une bonne évaluation des performances du code à l'aide de résultats théoriques, nous avons fait appel à des simulations.

L'algorithme élémentaire Nous donnons d'abord ici une proposition qui détermine l'ensemble des poids d'erreur qui devront faire l'objet de la simulation.

Proposition IV.30 *Nous supposons que $C_1(n_1, k_1, d_1)$ et $C_2(n_2, k_2, d_2)$ possèdent un algorithme de décodage d'erreur borné strictement par la distance minimale, on pose*

$$t_1 = \lfloor \frac{d_1 - 1}{2} \rfloor, \ \textit{et } t_2 = \lfloor \frac{d_2 - 1}{2} \rfloor.$$

Soit e_i la proportion de motifs d'erreur de $C_1 \otimes C_2$ de poids i corrigibles par l'algorithme élémentaire, on a

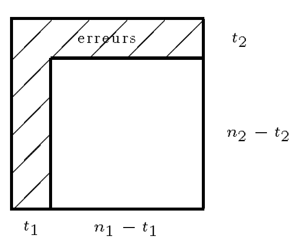
$$0 \le i < L, \qquad e_i = 1, \tag{IV.15}$$
$$L \le i \le U, \qquad 0 < e_i < 1, \tag{IV.16}$$

où

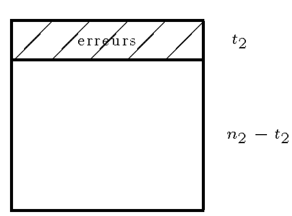
$$\begin{cases} U = n_1 n_2 - (n_1 - t_1)(n_2 - t_2), \\ L = (t_1 + 1)(t_2 + 1). \end{cases}$$

preuve : La proposition IV.13 nous assure que $e_i = 1$ pour $i < L$, et le corollaire IV.6 que $e_i < 1$ pour $i \ge L$.

Il existe des motifs d'erreur de poids U corrigible, considérons par exemple le motif



où la zone hachurée est le support du motif. Toutes les lignes sauf les t_2 premières sont corrigibles, donc après décodage des lignes on a un motif de la forme



où la zone hachurée contient le reste du motif d'erreur, chacune des colonnes contient donc au plus t_2 erreurs et est donc corrigible. □

La recherche du plus grand poids d'un motif corrigible est difficile, tout comme la présentation d'un contre-exemple rigoureux montrant que cette borne n'est pas U.

En outre, les valeurs des e_i pour $i > U$ ne sont significatives dans le calcul du taux d'erreur résiduel pour des valeurs élevées de la probabilité d'erreur, valeurs pour lesquelles les performances sont mauvaises.

L'algorithme de Reddy-Robinson

Proposition IV.31 *Nous supposons que $C_1(n_1, k_1, d_1)$ (resp. $C_2(n_2, k_2, d_2)$) possèdent un algorithme de décodage d'erreur borné strictement par la distance minimale, on pose*

$$t_2 = \lfloor \frac{d_2 - 1}{2} \rfloor.$$

On suppose de plus qu'il existe un motif d'erreur de poids n_2 non corrigible par γ_2.

Soit e_i la proportion de motifs d'erreur de $C_1 \otimes C_2$ de poids i corrigibles par l'algorithme de Reddy-Robinson, on a

$$0 \leq i < L, \qquad e_i = 1, \tag{IV.17}$$

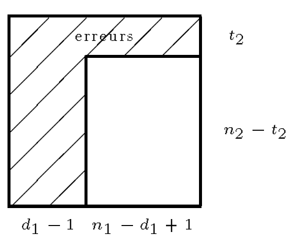
$$L \leq i \leq U, \qquad 0 < e_i < 1, \tag{IV.18}$$

où

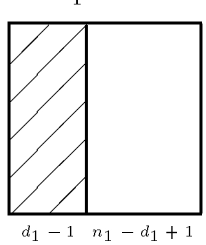
$$\begin{cases} U = n_1 n_2 - (n_1 - d_1 + 1)(n_2 - t_2), \\ L = \frac{d_1 d_2}{2}. \end{cases}$$

preuve : L'algorithme de Reddy-Robinson est borné par la distance minimale, donc $e_i = 1$ pour $i < L$. Comme il n'existe pas de meilleure borne pour l'algorithme, on a $e_L < 1$.

Il existe des motifs d'erreur de poids U corrigibles, considérons par exemple le motif



où la zone hachurée est le support du motif. On suppose de plus que chacune des $d_1 - 1$ premières colonnes est non corrigible. Après décodage des colonnes on a un motif de la forme



où la zone hachurée contient des effacements. Pour chaque ligne, la branche Ψ_{t_2} devra corriger un motif d'erreur et d'effacement comprenant $d_1 - 1$ effacements et aucune erreur. Chaque ligne est donc corrigible car aucune des autres branches Ψ_i, $i < t_2$ ne peut aboutir; il y aurait en effet n_1 effacements dans les t_2 premières lignes.

Le motif décrit est donc corrigible, et $e_U > 0$. □

Comme pour l'algorithme élémentaire, il est difficile, et peu utile d'un point de vue pratique, de trouver la borne supérieure exacte des motifs d'erreur corrigibles par l'algorithme de Reddy-Robinson.

4.3.3 Décodage au delà de la distance minimale

Pour évaluer le décodage au delà de la distance minimale, on utilise la capacité de correction d'erreur.

Définition IV.15 *Soit C un code linéaire sur $\mathbb{F}_q$ de longueur n et soit $P(x)$ son polynôme des motifs d'erreur corrigibles pour un algorithme de décodage d'erreur $\mathcal{A}$. On pose*

$$P_{t^*}(x) = \sum_{i=0}^{t^*} \binom{n}{i} (q-1)^i x^i.$$

La capacité de correction *de C pour l'algorithme $\mathcal{A}$ et pour la probabilité d'erreur p est le plus grand entier impair $d^* = 2t^* + 1$ tel que*

$$P_{t^*}\left(\frac{p}{(q-1)(1-p)}\right) \leq P\left(\frac{p}{(q-1)(1-p)}\right) < P_{t^*+1}\left(\frac{p}{(q-1)(1-p)}\right).$$

Le taux d'erreur résiduel de C pour un canal de probabilité d'erreur p, dont le polynôme des motifs d'effacement corrigibles est égal à $P(x)$, est égal à $1 - (1-p)^n P(\frac{p}{(q-1)(1-p)})$.

D'autre part $P_{t^*}(x)$ est le polynôme des motifs d'erreur corrigibles d'un code fictif C^* de longueur n, de distance minimale $d^* = 2t^* + 1$, et muni d'un algorithme de décodage d'erreur borné strictement par sa distance minimale.

La capacité de correction de C pour un algorithme et pour un canal donné est donc la distance minimale d'un code C^* qui, utilisé dans le même canal et ayant un algorithme borné strictement par sa distance minimale, aurait un taux d'erreur résiduel voisin.

5 Résultats des simulations

Soient $C_1(n_1, k_1, d_1)$ et $C_2(n_2, k_2, d_2)$ deux codes linéaires sur $\mathbb{F}_q$. Soit $\mathcal{C} = C_1 \otimes C_2$ le code produit de C_1 et C_2.

5.1 Effacements

Pour la correction d'effacement, nous faisons l'hypothèse suivante sur les décodeurs des codes composants :

- γ_1 et γ_2 sont des algorithmes de décodage d'effacement de C_1 et C_2 capables de corriger tous les motifs de poids inférieur strictement à d_1 et d_2 respectivement et aucun autre.

Dans le cadre de cette hypothèse, ni la dimension des codes, ni le cardinal de $\mathbb{F}_q$ n'ont d'influences sur les performances de décodage du code produit.

Nous avons calculé par une simulation la capacité de correction du code $RS(14,7,8) \otimes RS(14,7,8)$, le tableau IV.4 donne une évaluation de la proportion de motifs corrigibles. Pour chaque poids 10^6 motifs ont été générés aléatoirement. Pour les poids inférieurs à 120 tous les motifs d'effacement générés ont été corrigés.

w	$\tilde{S}_w/M$	w	$\tilde{S}_w/M$	w	$\tilde{S}_w/M$
< 120	1.000000	129	0.989830	139	0.344820
120	0.999998	130	0.980354	140	0.240351
121	0.999995	131	0.964168	141	0.153589
122	0.999980	132	0.938254	142	0.088032
123	0.999929	133	0.899242	143	0.044455
124	0.999846	134	0.844023	144	0.019035
125	0.999582	135	0.770268	145	0.006589
126	0.998985	136	0.678731	146	0.001676
127	0.997665	137	0.572586	147	0.000249
128	0.994961	138	0.458300	> 147	0

Tableau IV.4 :$RS(14,7,8) \otimes RS(14,7,8)$, simulation pour la correction d'effacement

Ces résultats de simulation nous ont permis de donner une estimation du polynôme des motifs corrigibles du code,

$$P(x) = \sum_{w=0}^{196} \binom{196}{w} \frac{\tilde{S}_w}{M} x^w,$$

qui permet de calculer le taux d'effacement résiduel p_{eff} pour toute valeur p de la probabilité d'effacement du canal

$$p_{eff} = (1-p)^{196} P(\frac{p}{1-p}),$$

ces résultats sont donnés dans le tableau IV.5.

Dans le tableau IV.6, nous donnons la valeur de la capacité de correction d'effacement pour des valeurs données du taux d'effacement résiduel. On notera que la capacité de correction varie peu, et donc constitue, au moins pour le code $RS(14,7,8) \otimes RS(14,7,8)$, une mesure pertinente des performances du code.

p	p_{eff}	p	p_{eff}	p	p_{eff}
0,45	$1{,}97.10^{-10}$	0,54	$0{,}24.10^{-4}$	0,63	0,0321
0,46	$1{,}01.10^{-9}$	0,55	$0{,}68.10^{-4}$	0,64	0,0549
0,47	$0{,}44.10^{-8}$	0,56	$1{,}77.10^{-4}$	0,65	0,0893
0,48	$1{,}83.10^{-8}$	0,57	$0{,}43.10^{-3}$	0,66	0,1381
0,49	$0{,}70.10^{-7}$	0,58	$1{,}01.10^{-3}$	0,67	0,2032
0,50	$0{,}25.10^{-6}$	0,59	$0{,}22.10^{-2}$	0,68	0,2847
0,51	$0{,}87.10^{-6}$	0,60	$0{,}47.10^{-2}$	0,69	0,3803
0,52	$0{,}28.10^{-6}$	0,61	$0{,}94.10^{-2}$	0,70	0,4853
0,53	$0{,}86.10^{-5}$	0,62	$1{,}78.10^{-2}$		

Tableau IV.5 : Taux d'effacement résiduel du code $RS(14,7,8) \otimes RS(14,7,8)$ pour diverses valeurs de la probabilité d'effacement p

p	d^*	p_{eff}
0.652	136	$\approx 10^{-1}$
0.610	135	$\approx 10^{-2}$
0.579	135	$\approx 10^{-3}$
0.553	134	$\approx 10^{-4}$
0.531	134	$\approx 10^{-5}$
0.511	133	$\approx 10^{-6}$
0.492	133	$\approx 10^{-7}$

Tableau IV.6 :Capacités de correction d'effacement pour un code $C(14,7,8) \otimes C(14,7,8)$ pour différents taux d'effacement résiduels

Enfin pour certains codes produit ayant des codes composants égaux de longueur 14 ou 15 et dont la distance minimale varie de 3 à 9, nous donnons dans le tableau IV.7 la capacité de correction d'effacement lorsque le taux d'effacement résiduel est voisin de 10^{-5}.

On notera que la capacité de correction d'effacement est dans ces exemples toujours très supérieure à la distance minimale.

5.2 Erreurs

5.2.1 Premier exemple : $RS(14,7,8) \otimes RS(14,7,8)$

Pour le décodage d'erreur, nous avons, de la même manière que pour les effacements, calculé le taux d'erreur résiduel (tableau IV.9) et la capacité de correction (tableau IV.10) du code $RS(14,7,8) \otimes RS(14,7,8)$ à partir des résultats obtenus en simulant le décodage d'erreur par l'algorithme élémentaire (tableau IV.8).

On constate alors que les variations de la capacité de correction d'erreur sont plus im-

$n = 14$				
	p	d^*	p_{eff}	paramètres (cas MDS)
$d = 3$	0.078	34	$< 10^{-5}$	(196,144,9)
$d = 4$	0.189	62	$< 10^{-5}$	(196,121,16)
$d = 5$	0.284	84	$< 10^{-5}$	(196,100,25)
$d = 6$	0.370	102	$< 10^{-5}$	(196,81,36)
$d = 7$	0.452	118	$< 10^{-5}$	(196,64,49)
$d = 8$	0.531	134	$< 10^{-5}$	(196,49,64)

$n = 15$				
	p	d^*	p_{eff}	paramètres (cas MDS)
$d = 3$	0.071	35	$< 10^{-5}$	(225,169,9)
$d = 4$	0.180	67	$< 10^{-5}$	(225,144,16)
$d = 5$	0.269	90	$< 10^{-5}$	(225,121,25)
$d = 6$	0.349	110	$< 10^{-5}$	(225,100,36)
$d = 7$	0.426	128	$< 10^{-5}$	(225,81,49)
$d = 8$	0.501	145	$< 10^{-5}$	(225,64,64)
$d = 9$	0.573	160	$< 10^{-5}$	(225,49,81)

Tableau IV.7 :capacité de décodage d'effacement du code $C(n, k, d) \otimes C(n, k, d)$ pour $n = 14$ et $n = 15$

portantes que celles de la capacité de correction d'effacement.

5.2.2 Autres exemples

Nous avons simulé le décodage d'autre codes produit, nous ne présenterons pas comme pour le code précédent tous les détails, mais seulement la capacité de correction d'erreur lorsque le taux d'erreur résiduel a des valeurs entre 10^{-4} et 10^{-6}. Dans les résultats donnés par le tableau IV.11, on remarquera que lorsque le taux d'erreur résiduel vaut 10^{-5} le code $RS(15, 7, 9) \otimes RS(15, 7, 9)$ a une capacité de correction égale à 179. Cette valeur est très élevée, puisque la borne de Singleton nous affirme qu'un code de longueur 225 et de dimension 49 a une distance minimale au plus égale à 177.

Donc il n'existe, pour obtenir un taux d'erreur résiduel de 10^{-5}, aucun code en bloc de même longueur et de même taux de transmission, dont l'algorithme de décodage soit borné strictement par la distance minimale, qui ait des performances équivalentes ou supérieure au code $RS(15, 7, 9) \otimes RS(15, 7, 9)$.

5.2.3 Amélioration de l'algorithme élémentaire

Les résultats de ces simulations, et en particulier le fait que l'algorithme élémentaire soit plus performant amène une remarque

w	$\check{S}_w/M$	w	$\check{S}_w/M$	w	$\check{S}_w/M$
<50	1.00000	59	0.98937	69	0.38256
50	0.99994	60	0.98252	70	0.25701
51	0.99983	61	0.97046	71	0.15092
52	0.99978	62	0.95159	72	0.07540
53	0.99943	63	0.92268	73	0.02980
54	0.99936	64	0.87870	74	0.00710
55	0.99894	65	0.81776	75	0.00140
56	0.99781	66	0.73343	>75	0
57	0.99629	67	0.63188		
58	0.99372	68	0.51199		

Tableau IV.8 :$RS(14,7,8) \otimes RS(14,7,8)$, correction d'erreur par l'algorithme élémentaire

p	p_{err}	p	p_{err}
0.15	$1.85\,10^{-8}$	0.23	$0.75\,10^{-3}$
0.16	$1.08\,10^{-7}$	0.24	$1.89\,10^{-3}$
0.17	$0.53\,10^{-6}$	0.25	$0.44\,10^{-2}$
0.18	$0.23\,10^{-5}$	0.26	$0.96\,10^{-2}$
0.19	$0.87\,10^{-5}$	0.27	0.0194
0.20	$0.30\,10^{-4}$	0.28	0.0367
0.21	$0.95\,10^{-4}$	0.29	0.0646
0.22	$0.28\,10^{-3}$	0.30	0.1066

Tableau IV.9 :$RS(14,7,8) \otimes RS(14,7,8)$, taux d'erreur résiduel, pour différentes valeurs de la probabilité d'erreur par symbole p

p	p_{err}	d^*
0.159	$\approx 10^{-7}$	121
0.174	$\approx 10^{-6}$	123
0.191	$\approx 10^{-5}$	125
0.210	$\approx 10^{-4}$	129
0.233	$\approx 10^{-3}$	131
0.260	$\approx 10^{-2}$	133
0.298	$\approx 10^{-1}$	135

Tableau IV.10 :$RS(14,7,8) \otimes RS(14,7,8)$, capacité de correction d'erreur pour différents taux d'erreur résiduels p_{err}

p_{err}	$RS(14,10,5)\otimes RS(14,10,5)$ p	d^*	$RS(14,8,7)\otimes RS(14,8,7)$ p	d^*
10^{-4}	0.070	59	0.194	121
10^{-5}	0.055	55	0.174	117
10^{-6}	0.043	51	0.159	115

p_{err}	$RS(14,7,8)\otimes RS(14,7,8)$ p	d^*	$RS(15,7,9)\otimes RS(15,7,9)$ p	d^*
10^{-4}	0.210	129	0.284	181
10^{-5}	0.191	125	0.265	179
10^{-6}	0.174	123	0.249	177

Tableau IV.11 : Capacité de correction de certains codes produit de Reed-Solomon sur $\mathbf{F}_{16}$
n

> L'algorithme de Reddy-Robinson a une meilleure borne que l'algorithme élémentaire : il est meilleur pour les motifs de poids proches de la moitié de la distance minimale, et pourtant il est moins performant en moyenne.

Il existe donc des motifs corrigibles par l'un de ces algorithmes et pas par l'autre. Comment combiner ces deux algorithmes de façon à obtenir un gain sensible? L'idée de l'amélioration est la suivante :

> On voudrait pouvoir corriger les motifs M tels qu'il existe une séquence de décodage d'erreur $M \to M'$, avec M' corrigible par l'algorithme de Reddy-Robinson.

En pratique nous allons itérer un algorithme proche de l'algorithme de Reddy-Robinson.
Nous étendons l'algorithme de Reddy-Robinson en rajoutant pour le décodage des lignes une branche supplémentaire sans effacement. Cet algorithme est ensuite itéré. La procédure complète est décrite en annexe C.

Cette amélioration est notable, en particulier pour le code $RS(14,7,8)\otimes RS(14,7,8)$ et un taux d'erreur résiduel de 10^{-5}, la capacité de correction passe de 125 à 129, pour une probabilité d'erreur de $p = 0.191$ et $p = 0.199$ respectivement (cf. tableau IV.12).

p	p_{err}	d^*
0.183	$\approx 10^{-6}$	129
0.199	$\approx 10^{-5}$	129
0.217	$\approx 10^{-4}$	131
0.239	$\approx 10^{-3}$	133

Tableau IV.12 :$RS(14,7,8)\otimes RS(14,7,8)$, capacité de correction d'erreur pour une version améliorée du décodage

6 Remarques sur la complexité du décodage

Cette section décrit l'une des propriétés fondamentale des codes produit ; le faible coût algorithmique de leur décodage.

6.1 Algorithme élémentaire

L'algorithme élémentaire de décodage des codes produits ne permet pas un décodage en temps constant, les bornes supérieures de la complexité de décodage d'un mot sont relativement mauvaises, en effet certaines configurations d'erreur peuvent provoquer des cycles dans le décodage, cycles difficile à détecter. Ces configurations sont cependant rares, et la complexité en moyenne reste très satisfaisante.

6.1.1 Etude théorique

Il est difficile de donner avec précision la complexité du décodage. Celle-ci dépend du nombre d'itérations de l'algorithme, nombre qui est susceptible de beaucoup varier.

On peut cependant dire que le nombre de décodages par ligne et par colonne est de l'ordre de 1, c'est-à-dire que si la complexité de décodage d'une ligne est λ_1 et la complexité de décodage d'une colonne est λ_2, la complexité de décodage d'un mot du code produit est de l'ordre de

$$\Lambda = n_1\lambda_2 + n_2\lambda_1,$$

et si $n_1 = n_2 = n$ et $\lambda_1 = \lambda_2 = O(n^a)$, alors

$$\Lambda = O(n^{a+1}) = O(N^{\frac{a+1}{2}}),$$

où $N = n^2$ est la longueur du code produit.

Bien entendu ce calcul est approximatif, les complexité de décodage n'étant pas en général en $O(n^a)$. En revanche, il souligne bien que la complexité de décodage est relativement faible.

Le cas des codes de Reed-Solomon Dans le cas où les codes composants sont des codes de Reed-Solomon, on peut faire une évaluation plus précise.

La complexité de décodage d'un code de Reed-Solomon de longueur n et de dimension k sur $\mathbb{F}_q$ est équivalente à $\lambda(n-k)^2$. Donc si $C_1 = C_2 = RS(n,k,d)$, la complexité de décodage d'un mot du code produit, c'est-à-dire de k^2 symboles de $\mathbb{F}_q$ est

$$\Lambda = 2\lambda\mu n(n-k)^2,$$

où μ est le nombre moyen de décodage par ligne et par colonne. Et si on pose $R = k/n$, on obtient

$$\Lambda = 2\lambda\mu n^3(1-R)^2.$$

Calculons maintenant à titre de comparaison la complexité d'un code de Reed-Solomon de longueur n sur $\mathbb{F}_q$ et ayant le même taux de transmission que le code produit, c'est-à-dire une dimension égale à k^2/n. Pour transmettre k^2 symboles d'information de ce code il faut n mots de code, la compléxité de décodage est donc égale à

$$\Lambda' = \lambda n(n - \frac{k^2}{n})^2,$$

soit encore

$$\Lambda' = \lambda n^3(1 - R^2)^2.$$

Le rapport Λ/Λ' est égale à $2\mu/(1+R)^2$. Par exemple si $\mu = 3/2$ et $R = 1/2$, ce rapport vaut $4/3$, c'est-à-dire que le code produit est trés légèrement plus complexe à décoder, alors que ses performances sont, au moins pour les exemples que nous avons traités, très supérieures.

Par exemple le code $RS(15,3,13)$ a un taux d'erreur résiduel de 0.14 pour $p = 0.265$, au lieu de 10^{-5} pour le code produit $RS(15,7,9) \otimes RS(15,7,9)$.

6.1.2 Complexité de décodage en fonction du taux d'erreur

Nous donnons ici des évaluations précises de la complexité du décodage d'erreur du code produit $RS(14,7,8) \otimes RS(14,7,8)$, pour l'algorithme élémentaire.

L'algorithme de décodage des codes produits est obtenu par itérations de l'algorithme de décodage des lignes et des colonnes, dans le cas du code produit $RS(14,7,8) \otimes RS(14,7,8)$, le décodage d'une ligne ou d'une colonne est effectué par l'algorithme d'Euclide étendu qui peut être décomposé en 3 étapes :

1. Calcul du syndrome (ici évaluation d'un polynôme en 7 points).

2. Calcul des polynômes localisateur et évaluateur (par l'algorithme d'Euclide étendu)

3. Calcul de l'erreur (calcul des racines d'un polynôme, puis évaluation d'une polynôme en chacune de ces racines – ici on a au maximum 3 racines)

Les coût de ces différentes étapes dépendent des choix fait pour la mise œuvre. L'étape 1 a un coût constant faible par rapport au coût des étapes 2 et 3. L'étape 2 a un coût égal au nombre de divisions euclidienne neccessaires à l'obtention des polynômes localisateur et évaluateur, ce nombre etant sensiblement égal au nombre d'erreur corrigées. L'étape 3 a un coût proportionnel au nombre d'erreur décodées (c'est-à-dire le degré du polynôme localisateur).

L'étape 2 étant plus couteuse que l'étape 3, du moins lorsque le corps fini est de petite taille, nous avons choisi comme mesure de la complexité de décodage le nombre de divisions euclidiennes effectuées au total au cours du décodage d'un mot du code produit.

Pour tous les poids entre 0 et 196, nous avons effectué 10^4 tirages aléatoires de motifs d'erreur, et compté le nombre de divisions euclidiennes, les résultats sont donnés dans la table IV.13.

A partir de ces chiffres nous avons calculé le nombre moyen de divisions euclidiennes pour un canal $q - aire$ ($q = 16$) symétrique, et pour des probabilités d'erreur variant entre 0% et 100%. Ces résultats sont donnés dans la table IV.14

poids	nb. de div.								
0	0	1	10000	2	19961	3	29859	4	39737
5	49610	6	59456	7	69373	8	79305	9	89366
10	99553	11	109602	12	120349	13	130735	14	141503
15	152718	16	163748	17	175433	18	187050	19	198928
20	211632	21	224014	22	236835	23	249753	24	263714
25	277481	26	291164	27	305885	28	320479	29	334736
30	350600	31	366009	32	381828	33	397757	34	413929
35	431520	36	448248	37	466064	38	484002	39	502951
40	523426	41	542747	42	564763	43	585612	44	607924
45	631636	46	656505	47	681247	48	706533	49	735301
50	764845	51	795718	52	827268	53	860652	54	898911
55	935244	56	974575	57	1017447	58	1062003	59	1109045
60	1159444	61	1209635	62	1257765	63	1309160	64	1350967
65	1389294	66	1408849	67	1416338	68	1402355	69	1365319
70	1312110	71	1252676	72	1192086	73	1134084	74	1087754
75	1055104	76	1027666	77	1001691	78	97763	79	96106
80	95585	81	93238	82	91880	83	90647	84	90417
85	89110	86	88142	87	86894	88	86510	89	85921
90	85443	91	84480	92	84520	93	83817	94	83159
95	83026	96	82851	97	82973	98	82212	99	81752
100	81496	101	81677	102	80911	103	80852	104	81022
105	80663	106	80644	107	80204	108	80250	109	79989
110	80010	111	80022	112	79951	113	79832	114	79840
115	79680	116	79597	117	79623	118	79663	119	79548
120	79276	121	79353	122	79566	123	79666	124	79499
125	79341	126	79428	127	79313	128	79352	129	79493
130	79440	131	79331	132	79175	133	79409	134	79327
135	79469	136	79308	137	79326	138	79396	139	79252
140	79327	141	79420	142	79205	143	79346	144	79297
145	79135	146	79308	147	79379	148	79391	149	79307
150	79419	151	79367	152	79495	153	79207	154	79340
155	79351	156	79278	157	79448	158	79266	159	79415
160	79560	161	79282	162	79344	163	79377	164	79226
165	79232	166	79291	167	79328	168	79409	169	79205
170	79400	171	79273	172	79417	173	79166	174	79302
175	79436	176	79178	177	79396	178	79464	179	79273
180	79414	181	79484	182	79401	183	79367	184	79291
185	79138	186	79361	187	79241	188	79291	189	79251
190	79213	191	79241	192	79235	193	79283	194	79355
195	79469	196	79285						

Tableau IV.13 : Nombre de divisions pour 10000 tirages, et pour tous les poids d'erreur

p (en %)	nb. de div.								
0	0	1	1.952	2	3.893	3	5.838	4	7.803
5	9.806	6	11.862	7	13.987	8	16.194	9	18.492
10	20.890	11	23.392	12	26.003	13	28.725	14	31.563
15	34.524	16	37.619	17	40.865	18	44.284	19	47.905
20	51.761	21	55.890	22	60.337	23	65.149	24	70.372
25	76.040	26	82.153	27	88.654	28	95.401	29	102.143
30	108.521	31	114.105	32	118.455	33	121.210	34	122.174
35	121.367	36	119.019	37	115.525	38	111.353	39	106.954
40	102.694	41	98.815	42	95.439	43	92.593	44	90.241
45	88.317	46	86.746	47	85.461	48	84.405	49	83.529
50	82.800	51	82.189	52	81.676	53	81.244	54	80.883
55	80.582	56	80.332	57	80.128	58	79.961	59	79.827
60	79.719	61	79.633	62	79.566	63	79.513	64	79.471
65	79.438	66	79.411	67	79.390	68	79.372	69	79.358
70	79.346	71	79.337	72	79.331	73	79.327	74	79.327
75	79.330	76	79.335	77	79.340	78	79.345	79	79.349
80	79.351	81	79.349	82	79.345	83	79.338	84	79.330
85	79.323	86	79.318	87	79.317	88	79.320	89	79.327
90	79.338	91	79.349	92	79.356	93	79.349	94	79.327
95	79.296	96	79.270	97	79.255	98	79.259	99	79.313
100	79.318								

Tableau IV.14 : Nombre moyen de divisions par décodage en fonction du taux d'erreur

6.1.3 Complexité et performances

Il peut être intéressant de considérer comment varient la complexité de décodage et les performances du codes en fonction du poids de l'erreur (figure IV.2) ou du taux d'erreur d'un canal q-aire symétrique (figure IV.3, les courbes données sont celles du code $RS(14,7,8) \otimes RS(14,7,8)$. Dans les courbes la complexité a été divisée par 3×28; 28 etant la somme du nombre de lignes et de colonnes, et 3 etant le nombre maximal de divisions effectuées par l'algorithme d'Euclide étendu pour décoder le code $RS(14,7,8)$.

Une complexité 1 correspond donc grossièrement à un décodage par ligne et par colonne.

On notera que dans les domaines où l'algorithme est efficace, c'est-à-dire lorsque le taux de correction est voisin de 1, la complexité varie de façon sensiblement linéaire par rapport au taux d'erreur.

6.2 Algorithme de Reddy-Robinson et dérivé

6.2.1 Algorithme de Reddy-Robinson

A l'inverse de l'algorithme élémentaire l'algorithme de Reddy-Robinson se décode en temps constant, on peut détailler sa complexité :

1. chaque colonne est décodée 1 fois,

2. chaque ligne est décodée $(d_2+1)/2$ fois.

On considère que les autres étapes de la procédure de décodage ont une complexité non significative.

Le cas Reed-Solomon On suppose que C_1 et C_2 sont égaux à un code de Reed-Solomon $RS(n,k,d)$. On suppose que ce dernier code a une complexité de décodage de $\lambda(n-k)^2$, la complexité de décodage du code produit vaut alors

$$\Lambda = n\frac{d+1}{2}\lambda(n-k)^2 + \lambda n(n-k)^2,$$

on a $d = n-k+1$, et on pose $R = k/n$,

$$\Lambda = \lambda n^3(\frac{n-k}{2}+1)(1-R)^2 + \lambda n^3(1-R)^2,$$

soit encore

$$\Lambda = \lambda n^4\frac{(1-R)^3}{2} + 2\lambda n^3(1-R)^2.$$

Bien que cette complexité soit asymptotiquement supérieure à celle de l'algorithme élémentaire, pour les codes étudiés (i.e. codes de Reed-Solomon de longueur 15) les complexités sont sensiblement équivalentes.

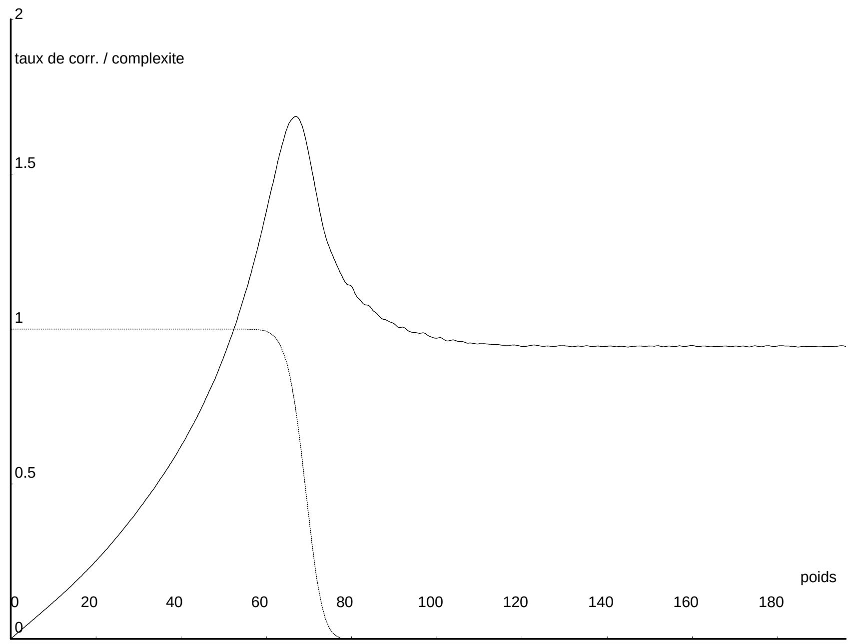


Figure IV.2 : $RS(14,7,8) \otimes RS(14,7,8)$, taux de correction et complexité, en abscisse le poids

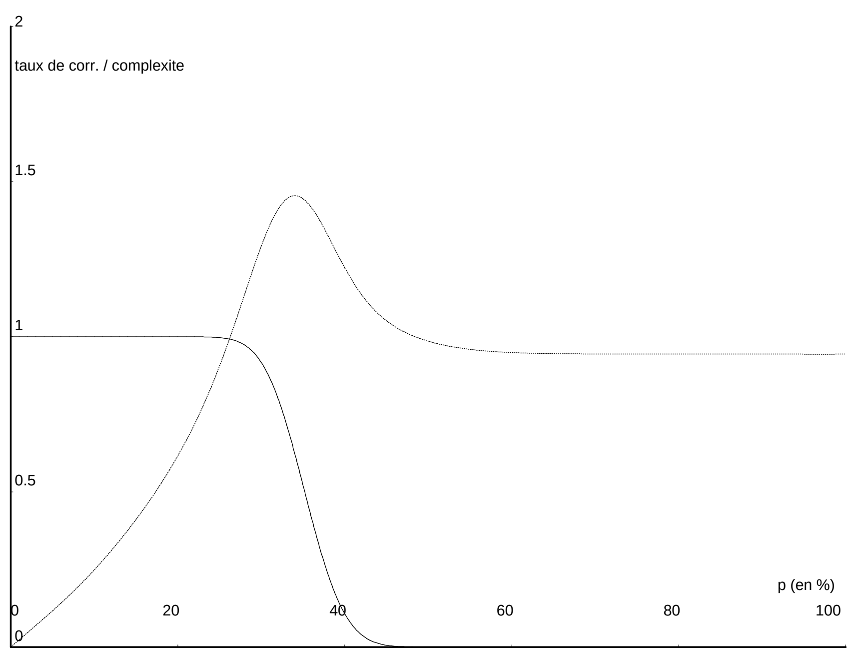


Figure IV.3 : $RS(14,7,8) \otimes RS(14,7,8)$, taux de correction et complexité, en abscisse le taux d'erreur (en %)

6.2.2 Algorithme de Reddy-Robinson itéré

Enfin, nous ajoutons un dernier mot sur la complexité de l'algorithme dérivé des deux premiers (cf. annexe C).

La complexité de cet algorithme va être fonction du nombre moyen d'itérations, chacune de ces itération ayant une complexité sensiblement égale à celle de l'algorithme de Reddy-Robinson.

Ce nombre, comme dans le cas de l'algorithme élémentaire est difficile a déterminer de façon théorique. Notons que pour les codes dont nous avons simulé le décodage, le nombre moyen d'itérations est très proche de celui obtenu pour l'algorithme élémentaire.

C'est-à-dire que le nombre de décodages des codes composants est sensiblement le même, mais que chacun de ces décodages a demandé respectivement $t_2 + 2$ et $t_1 + 2$ appels à phi_1 et phi_2, où t_1 et t_2 sont le nombre d'erreurs corrigibles par ϕ_1 le décodeur d'erreur des lignes et ϕ_2 le décodeur d'erreur des colonnes.

Le cas des codes de Reed-Solomon Si l'on suppose que les codes composants sont deux codes de Reed-Solomon identiques $RS(n, k, d)$, si $t = \lfloor d/2 \rfloor$, alors la complexité de décodage est sensiblement égale à

$$\Lambda = 2\lambda\mu(t+2)n^3(1-R)^2,$$

en reprenant les notations de la section 6.1.1.

Conclusions et perspectives

1 Décodage au delà de la distance minimale

Les codes que nous avons étudiés dans ce travail corrigent certains motifs d'erreur de poids supérieur à la moitié de la distance minimale. Or pour juger des performances d'un algorithme de décodage on utilise généralement sa borne, la distance minimale étant la meilleure borne pour un code donné. On jugera généralement qu'un code est bon s'il a une bonne distance minimale.

Pour juger des codes décodant au-delà de la distance minimale nous avons utilisé la capacité de correction. Bien qu'ayant la même dimension qu'une distance minimale, il convient de prendre des précaution sur son interpretation :

- la capacité de correction dépend du taux d'erreur du canal,
- lorsque ce taux d'erreur tend vers ∞ la capacité de correction tend vers la borne de l'algorithme, si mauvaise soit-elle,
- si l'on souhaite comparer deux codes, il faut comparer les capacités de correction de leur meilleur algorithme, et non la distance minimale de l'un et la capacité de correction de l'autre.

Si l'on souhaite pourtant se faire une idée des performances d'un code à partir de sa capacité de correction d^*, il faut donc bien préciser que l'hypothétique code de même longueur, de même dimension et de distance minimale d^*, n'aura des performances du même ordre que si son meilleur algorithme de décodage, dont la complexité algorithmique est acceptable, est borné *strictement* par sa distance minimale.

Tous les codes BCH, et donc en particulier les codes de Reed-Solomon sont dans ce cas, en effet le meilleur algorithme (raisonnable) connu, est l'algorithme d'Euclide étendu (ou de façon équivalente l'algorithme de Berlekamp-Massey [30]) qui est borné strictement par la distance construite.

Dans cette optique, le principal intéret des codes produit, et dans une moindre mesure des codes concaténés, est qu'ils disposent d'un algorithme naturel capable de corriger nettement au delà de la distance minimale.

2 Quels codes produit ?

Quelques tentatives de décodage de produits de code de Hamming ont été effectués, les résultats on été relativement décevants. En effet, pour obtenir un taux d'erreur résiduel de 10^{-5} le produit d'un code de Hamming de longueur 15 par lui-même a une capacité de correction de 17, alors que des codes BCH de même longueur 225, et de dimension voisine 117 ou 125 ont pour distance minimale 29 ou 27.

Considérons un autre produit de code binaire, le code $B(15,7) \otimes B(15,7)$, où $B(15,7)$ est le code BCH primitif au sens strict de longueur 15, de distance construite 7 et de dimension 5. Nous avons obtenu pour un taux d'erreur résiduel de 10^{-5} une capacité de correction de 115 pour ce code. Ce résultat est bien meilleur puisque qu'aucun code BCH ne peut atteindre ces valeurs.

Il convient cependant de préciser que les simulations faites pour ces deux codes n'ont pas été poussées très loin. Les valeurs données pour la capacité de correction sont donc probablement légèrement surévaluées.

En tenant compte de ces résultats ainsi que de ceux exposés dans la thèse, on peut dégager deux propriétés empiriques souhaitables pour construire de bons codes produit :

1. les codes composants ne doivent pas être parfait, car l'algorithme risque alors d'ajouter des erreurs plus souvent qu'il n'en corrige,

2. le taux de transmission doit être faible.

Cela indique que les les codes produit ne peuvent pas avoir des paramètres quelconques, et ne sont pas adaptés à tous les types de bruits, mais bien a ceux que nous avons considérés dans ce travail. Il nous semble de toute façon que les codes produit n'ont pas bénéficié d'une attention suffisante en ce qui concerne leurs applications au codage, cela est probablement du à leur mauvaise distance minimale, distance minimale qui devient même exécrable asymptotiquement. Cet argument n'est plus valable dés que l'on utilise des critères plus adaptés à une utilisation pratique comme la capacité de correction, ou même des critères théoriques plus complets comme la distribution des poids, en effet les résultats de la section 1.2 du chapitre IV prouvent que le nombre de mot de poids minimal est très faible par rapport au cardinal du code.

3 Codes concaténés *vs.* codes produit

Une question peut sembler naturelle après ce travail : des codes produit et des codes concaténés, quelle est la meilleure construction?

La réponse n'est évidemment pas simple, d'abord parceque nous n'avons pas encore suffisemment d'éléments pour répondre, et ensuite parcequ'il est difficile de construire un code produit et un code concaténé qui soit comparable.

Pour faire une comparaison équitable, il faudrait obtenir par chacune de ces constructions un code de même longueur, de même dimension, défini sur le même corps et ayant une complexité de décodage comparable.

De plus on a vu pour les codes produit que ceux-ci étaient relativement meilleurs pour des taux de transmission faible, qu'en est il des codes concaténés, et les taux de transmission optimaux sont-ils les même ? On peut penser par exemple que les deux construction vont être adaptées à des applications différentes.

Enfin existe-il une construction plus performante combinant le produit et la concaténation, par exemple un code produit utilisé comme code externe d'un code concaténé ?

Annexe A

Résolution des équations de Newton en MAPLE

Nous présentons ici des preuves de resultats annoncés dans le chapitre II

1 Recherche de contradictions

Chacune des trois preuves présentées ici a été obtenue à l'aide du logiciel de calcul formel MAPLE. Le lecteur se rendra compte aisément que ces calculs étaient impossibles "à la main".

Chacune de ces preuves a été obtenue à la suite d'une session intéractive de l'ordre de quelques heures, mais ont neccessité seulement quelques minutes de temps calcul sur des SUN 3 et SUN 4, et une vérification de la validité des résultats à l'aide du même logiciel peut et a été faite très rapidement.

1.1 Le code $B(255, 59)$ n'admet pas de mots de poids 59

Nous considérons les équations de Newton eq_r pour $0 < r \leq n = 255$, pour le code $B(255, 59)$, et pour le poids $\delta = 59$. Nous voulons montrer qu'il n'existe pas de mots de code de poids δ.

D'après la propriété de conjugaison des fonctions puissances symétriques, et en tenant compte de l'ensemble de définition du code, les seules fonctions puissances symétriques inconnues sont

$$A_{59},\ A_{61},\ A_{63},\ A_{85},\ A_{87},\ A_{91},\ A_{95},\ A_{111},\ A_{119},\ A_{127},$$

et puisque 255 et 59 sont premier entre eux, on peut supposer $A_{59} = 1$.

On a $eq_{59}\ :\ \sigma_{59} = A_{59} = 1$, et les σ_i pour i impair, $1 \leq i < 59$ sont nuls.

Les équations de Newton eq_i, pour i impair, $61 \leq i \leq 119$ nous fournissent un système triangulaire linéaire donnant les σ_i, pour i pair, $2 \leq i \leq 58$ ce système est formé des 29 équations suivantes

$$eq_{61}\ :\ A_{61} + \sigma_2 = 0$$

$$
\begin{aligned}
eq_{63} &: A_{63} + A_{61}\sigma_2 + \sigma_4 = 0\\
eq_{65} &: A_{63}\sigma_2 + A_{61}\sigma_4 + \sigma_6 = 0\\
eq_{67} &: A_{63}\sigma_4 + A_{61}\sigma_6 + \sigma_8 = 0\\
eq_{69} &: A_{63}\sigma_6 + A_{61}\sigma_8 + \sigma_{10} = 0\\
eq_{71} &: A_{63}\sigma_8 + A_{61}\sigma_{10} + \sigma_{12} = 0\\
eq_{73} &: A_{63}\sigma_{10} + A_{61}\sigma_{12} + \sigma_{14} = 0\\
eq_{75} &: A_{63}\sigma_{12} + A_{61}\sigma_{14} + \sigma_{16} = 0\\
eq_{77} &: A_{63}\sigma_{14} + A_{61}\sigma_{16} + \sigma_{18} = 0\\
eq_{79} &: A_{61}^{64} + A_{63}\sigma_{16} + A_{61}\sigma_{18} + \sigma_{20} = 0\\
eq_{81} &: A_{61}^{64}\sigma_2 + A_{63}\sigma_{18} + A_{61}\sigma_{20} + \sigma_{22} = 0\\
eq_{83} &: A_{61}^{64}\sigma_4 + A_{63}\sigma_{20} + A_{61}\sigma_{22} + \sigma_{24} = 0\\
eq_{85} &: A_{85} + A_{61}^{64}\sigma_6 + A_{63}\sigma_{22} + A_{61}\sigma_{24} + \sigma_{26} = 0\\
eq_{87} &: A_{87} + A_{85}\sigma_2 + A_{61}^{64}\sigma_8 + A_{63}\sigma_{24} + A_{61}\sigma_{26} + \sigma_{28} = 0\\
eq_{89} &: A_{87}\sigma_2 + A_{85}\sigma_4 + A_{61}^{64}\sigma_{10} + A_{63}\sigma_{26} + A_{61}\sigma_{28} + \sigma_{30} = 0\\
eq_{91} &: A_{91} + A_{87}\sigma_4 + A_{85}\sigma_6 + A_{61}^{64}\sigma_{12} + A_{63}\sigma_{28} + A_{61}\sigma_{30} + \sigma_{32} = 0\\
eq_{93} &: A_{87}^{4} + A_{91}\sigma_2 + A_{87}\sigma_6 + A_{85}\sigma_8 + A_{61}^{64}\sigma_{14} + A_{63}\sigma_{30} + A_{61}\sigma_{32} + \sigma_{34} = 0\\
eq_{95} &: A_{95} + A_{87}^{4}\sigma_2 + A_{91}\sigma_4 + A_{87}\sigma_8 + A_{85}\sigma_{10} + A_{61}^{64}\sigma_{16} + A_{63}\sigma_{32} + A_{61}\sigma_{34} + \sigma_{36} = 0\\
eq_{97} &: A_{95}\sigma_2 + A_{87}^{4}\sigma_4 + A_{91}\sigma_6 + A_{87}\sigma_{10} + A_{85}\sigma_{12} + A_{61}^{64}\sigma_{18} + A_{63}\sigma_{34} + A_{61}\sigma_{36} + \sigma_{38} = 0\\
eq_{99} &: A_{95}\sigma_4 + A_{87}^{4}\sigma_6 + A_{91}\sigma_8 + A_{87}\sigma_{12} + A_{85}\sigma_{14} + A_{61}^{64}\sigma_{20} + A_{63}\sigma_{36} + A_{61}\sigma_{38} + \sigma_{40} = 0\\
eq_{101} &: A_{95}\sigma_6 + A_{87}^{4}\sigma_8 + A_{91}\sigma_{10} + A_{87}\sigma_{14} + A_{85}\sigma_{16} + A_{61}^{64}\sigma_{22} + A_{63}\sigma_{38} + A_{61}\sigma_{40} + \sigma_{42} = 0\\
eq_{103} &: 1 + A_{95}\sigma_8 + A_{87}^{4}\sigma_{10} + A_{91}\sigma_{12} + A_{87}\sigma_{16} + A_{85}\sigma_{18} + A_{61}^{64}\sigma_{24} + A_{63}\sigma_{40} + A_{61}\sigma_{42}\\
&\quad +\sigma_{44} = 0\\
eq_{105} &: \sigma_2 + A_{95}\sigma_{10} + A_{87}^{4}\sigma_{12} + A_{91}\sigma_{14} + A_{87}\sigma_{18} + A_{85}\sigma_{20} + A_{61}^{64}\sigma_{26} + A_{63}\sigma_{42} + A_{61}\sigma_{44}\\
&\quad +\sigma_{46} = 0\\
eq_{107} &: A_{91}^{32} + \sigma_4 + A_{95}\sigma_{12} + A_{87}^{4}\sigma_{14} + A_{91}\sigma_{16} + A_{87}\sigma_{20} + A_{85}\sigma_{22} + A_{61}^{64}\sigma_{28} + A_{63}\sigma_{44}\\
&\quad +A_{61}\sigma_{46} + \sigma_{48} = 0\\
eq_{109} &: A_{91}^{4} + A_{91}^{32}\sigma_2 + \sigma_6 + A_{95}\sigma_{14} + A_{87}^{4}\sigma_{16} + A_{91}\sigma_{18} + A_{87}\sigma_{22} + A_{85}\sigma_{24} + A_{61}^{64}\sigma_{30} + A_{63}\sigma_{46}\\
&\quad +A_{61}\sigma_{48} + \sigma_{50} = 0\\
eq_{111} &: A_{111} + A_{91}^{4}\sigma_2 + A_{91}^{32}\sigma_4 + \sigma_8 + A_{95}\sigma_{16} + A_{87}^{4}\sigma_{18} + A_{91}\sigma_{20} + A_{87}\sigma_{24} + A_{85}\sigma_{26} + A_{61}^{64}\sigma_{32}\\
&\quad +A_{63}\sigma_{48} + A_{61}\sigma_{50} + \sigma_{52} = 0\\
eq_{113} &: A_{111}\sigma_2 + A_{91}^{4}\sigma_4 + A_{91}^{32}\sigma_6 + \sigma_{10} + A_{95}\sigma_{18} + A_{87}^{4}\sigma_{20} + A_{91}\sigma_{22} + A_{87}\sigma_{26} + A_{85}\sigma_{28}\\
&\quad +A_{61}^{64}\sigma_{34} + A_{63}\sigma_{50} + A_{61}\sigma_{52} + \sigma_{54} = 0\\
eq_{115} &: A_{111}\sigma_4 + A_{91}^{4}\sigma_6 + A_{91}^{32}\sigma_8 + \sigma_{12} + A_{95}\sigma_{20} + A_{87}^{4}\sigma_{22} + A_{91}\sigma_{24} + A_{87}\sigma_{28} + A_{85}\sigma_{30}\\
&\quad +A_{61}^{64}\sigma_{36} + A_{63}\sigma_{52} + A_{61}\sigma_{54} + \sigma_{56} = 0\\
eq_{117} &: A_{87}^{16} + A_{111}\sigma_6 + A_{91}^{4}\sigma_8 + A_{91}^{32}\sigma_{10} + \sigma_{14} + A_{95}\sigma_{22} + A_{87}^{4}\sigma_{24} + A_{91}\sigma_{26} + A_{87}\sigma_{30}\\
&\quad +A_{85}\sigma_{32} + A_{61}^{64}\sigma_{38} + A_{63}\sigma_{54} + A_{61}\sigma_{56} + \sigma_{58} = 0
\end{aligned}
$$

$$
\begin{aligned}
eq_{119} \ : \ & A_{119} + A_{87}^{16}\sigma_2 + A_{111}\sigma_8 + A_{91}^{4}\sigma_{10} + A_{91}^{32}\sigma_{12} + \sigma_{16} + A_{95}\sigma_{24} + A_{87}^{4}\sigma_{26} + A_{91}\sigma_{28} \\
& +A_{87}\sigma_{32} + A_{85}\sigma_{34} + A_{61}^{64}\sigma_{40} + A_{63}\sigma_{56} + A_{61}\sigma_{58} = 0
\end{aligned}
$$

la résolution de ce système nous donne l'expression des σ_i en fonction des A_i

$$
\begin{aligned}
\sigma_2 &:= A_{61} \\
\sigma_4 &:= A_{61}^2 + A_{63} \\
\sigma_6 &:= A_{61}^3 \\
\sigma_8 &:= A_{63}A_{61}^2 + A_{63}^2 + A_{61}^4 \\
\sigma_{10} &:= A_{61}A_{63}^2 + A_{61}^5 \\
\sigma_{12} &:= A_{63}^3 + A_{63}A_{61}^4 + A_{61}^6 \\
\sigma_{14} &:= A_{61}^7 \\
\sigma_{16} &:= A_{63}^4 + A_{63}^2A_{61}^4 + A_{63}A_{61}^6 + A_{61}^8 \\
\sigma_{18} &:= A_{61}A_{63}^4 + A_{63}^2A_{61}^5 + A_{61}^9 \\
\sigma_{20} &:= A_{61}^{64} + A_{63}^5 + A_{63}^3A_{61}^4 + A_{63}A_{61}^8 + A_{61}^2A_{63}^4 + A_{61}^{10} \\
\sigma_{22} &:= A_{61}^3A_{63}^4 + A_{61}^{11} \\
\sigma_{24} &:= A_{61}^{66} + A_{63}^6 + A_{63}^2A_{61}^8 + A_{61}^2A_{63}^5 + A_{63}A_{61}^{10} + A_{61}^{12} \\
\sigma_{26} &:= A_{85} + A_{61}A_{63}^6 + A_{63}^2A_{61}^9 + A_{61}^{13} \\
\sigma_{28} &:= A_{87} + A_{61}^{64}A_{63}^2 + A_{61}^{68} + A_{63}^7 + A_{63}^3A_{61}^8 + A_{63}A_{61}^{12} + A_{61}^{14} \\
\sigma_{30} &:= A_{85}A_{61}^2 + A_{61}^{15} \\
\sigma_{32} &:= A_{91} + A_{87}A_{61}^2 + A_{61}^{70} + A_{63}^8 + A_{63}^4A_{61}^8 + A_{63}^2A_{61}^{12} + A_{63}A_{61}^{14} + A_{61}^{16} \\
\sigma_{34} &:= A_{87}^4 + A_{85}A_{63}^2 + A_{85}A_{61}^4 + A_{61}A_{63}^8 + A_{63}^4A_{61}^9 + A_{63}^2A_{61}^{13} + A_{61}^{17} \\
\sigma_{36} &:= A_{95} + A_{61}^{18} + A_{61}^{72} + A_{63}^9 + A_{91}A_{61}^2 + A_{87}A_{63}^2 + A_{87}A_{61}^4 + A_{61}^{64}A_{63}^4 + A_{61}^{68}A_{63}^2 + A_{63}^5A_{61}^8 \\
&\quad +A_{63}^3A_{61}^{12} + A_{63}A_{61}^{16} + A_{61}^2A_{63}^8 + A_{63}^4A_{61}^{10} \\
\sigma_{38} &:= A_{85}A_{61}^6 + A_{61}^3A_{63}^8 + A_{63}^4A_{61}^{11} + A_{61}^{19} + A_{87}^4A_{61}^2 \\
\sigma_{40} &:= A_{61}^{128} + A_{63}^6A_{61}^8 + A_{63}^2A_{61}^{16} + A_{61}^2A_{63}^9 + A_{63}^5A_{61}^{10} + A_{61}^4A_{63}^8 + A_{61}^{66}A_{63}^4 + A_{95}A_{61}^2 \\
&\quad +A_{91}A_{63}^2 + A_{91}A_{61}^4 + A_{61}^{74} + A_{63}^{10} + A_{61}^{20} + A_{87}A_{61}^6 + A_{63}A_{61}^{18} \\
\sigma_{42} &:= A_{85}A_{63}^2A_{61}^4 + A_{87}^4A_{63}^2 + A_{87}^4A_{61}^4 + A_{85}A_{63}^4 + A_{85}A_{61}^8 + A_{63}^6A_{61}^9 + A_{63}^2A_{61}^{17} + A_{61}^5A_{63}^8 \\
&\quad +A_{61}A_{63}^{10} + A_{61}^{129} + A_{61}^{21} \\
\sigma_{44} &:= 1 + A_{87}A_{63}^2A_{61}^4 + A_{63}A_{61}^{128} + A_{61}^{22} + A_{61}^{76} + A_{63}^{11} + A_{95}A_{63}^2 + A_{95}A_{61}^4 + A_{91}A_{61}^6 \\
&\quad +A_{87}A_{63}^4 + A_{87}A_{61}^8 + A_{61}^{64}A_{63}^6 + A_{61}^{72}A_{63}^2 + A_{63}^7A_{61}^8 + A_{63}^3A_{61}^{16} + A_{61}^4A_{63}^9 + A_{63}A_{61}^{20} \\
&\quad +A_{61}^6A_{63}^8 \\
\sigma_{46} &:= A_{85}A_{61}^2A_{63}^4 + A_{61}^7A_{63}^8 + A_{61}^{23} + A_{87}^4A_{61}^6 + A_{85}A_{61}^{10} \\
\sigma_{48} &:= A_{91}A_{63}^2A_{61}^4 + A_{87}A_{61}^2A_{63}^4 + A_{63}^{12} + A_{61}^{24} + A_{61}^{132} + A_{61}^{78} + A_{61}^2 + A_{91}^{32} + A_{95}A_{61}^6 + A_{91}A_{63}^4 \\
&\quad +A_{91}A_{61}^8 + A_{87}A_{61}^{10} + A_{63}A_{61}^{22} + A_{63}^4A_{61}^{16} + A_{61}^4A_{63}^{10} + A_{63}^2A_{61}^{20} + A_{61}^6A_{63}^9 \\
\sigma_{50} &:= A_{91}^4 + A_{85}A_{61}^{12} + A_{61}A_{63}^{12} + A_{63}^4A_{61}^{17} + A_{61}^5A_{63}^{10} + A_{63}^2A_{61}^{21} + A_{85}A_{63}^2A_{61}^8 + A_{87}^4A_{63}^2A_{61}^4 \\
&\quad +A_{61}^{25} + A_{61}^{133} + A_{87}^4A_{63}^4 + A_{87}^4A_{61}^8 + A_{85}A_{63}^6
\end{aligned}
$$

$$
\begin{aligned}
\sigma_{52} \ := \ & A_{61}^4 + A_{63}^2 + A_{61}^{64}A_{63}^8 + A_{61}^{72}A_{63}^4 + A_{61}^{76}A_{63}^2 + A_{63}A_{61}^{24} + A_{63}A_{61}^{132} + A_{63}^5A_{61}^{16} + A_{61}^4A_{63}^{11} \\
& +A_{63}^3A_{61}^{20} + A_{61}^2A_{63}^{12} + A_{63}^4A_{61}^{18} + A_{61}^{80} + A_{63}^{13} + A_{61}^{26} + A_{95}A_{63}^2A_{61}^4 + A_{91}A_{61}^2A_{63}^4 \\
& +A_{87}A_{63}^2A_{61}^8 + A_{95}A_{63}^4 + A_{87}A_{61}^{12} + A_{91}^{32}A_{61}^2 + A_{95}A_{61}^8 + A_{87}A_{63}^6 + A_{91}A_{61}^{10} + A_{85}^2 \\
& +A_{111} \\
\sigma_{54} \ := \ & A_{87}^4A_{61}^2A_{63}^4 + A_{85}A_{61}^{14} + A_{61}A_{85}^2 + A_{61}^3A_{63}^{12} + A_{63}^4A_{61}^{19} + A_{61}^{27} + A_{87}^4A_{61}^{10} + A_{91}^4A_{61}^2 \\
\sigma_{56} \ := \ & A_{91}A_{63}^2A_{61}^8 + A_{95}A_{61}^2A_{63}^4 + A_{63}^{14} + A_{87}^2 + A_{61}^{82} + A_{61}^{136} + A_{61}^{28} + A_{111}A_{61}^2 + A_{91}^{32}A_{63}^2 \\
& +A_{91}^{32}A_{61}^4 + A_{95}A_{61}^{10} + A_{91}A_{63}^6 + A_{91}A_{61}^{12} + A_{87}A_{61}^{14} + A_{61}^{128}A_{63}^4 + A_{61}^{66}A_{63}^8 + A_{61}^{74}A_{63}^4 \\
& +A_{63}A_{85}^2 + A_{63}^2A_{61}^{24} + A_{63}^6A_{61}^{16} + A_{61}^2A_{63}^{13} + A_{63}^5A_{61}^{18} + A_{63}A_{61}^{26} + A_{61}^6 \\
\sigma_{58} \ := \ & A_{85}A_{63}^2A_{61}^{12} + A_{85}A_{63}^4A_{61}^8 + A_{87}^4A_{63}^2A_{61}^8 + A_{87}^{16} + A_{61}^{129}A_{63}^4 + A_{61}A_{63}^{14} + A_{61}A_{87}^2 \\
& +A_{63}^2A_{61}^{25} + A_{63}^6A_{61}^{17} + A_{61}^{137} + A_{61}^{29} + A_{91}^4A_{63}^2 + A_{91}^4A_{61}^4 + A_{87}^4A_{63}^6 + A_{87}^4A_{61}^{12} \\
& +A_{85}A_{63}^8 + A_{85}A_{61}^{16}
\end{aligned}
$$

Ces valeurs des σ_i sont remplacées par leurs valeurs dans les équations restantes. Après remplacement et développement, les équations sont triées par ordre croissant de taille, la taille d'une équation étant égale au nombre de ses monômes, on obtient la liste suivante

> 180, 188, 196, 192, 186, 184, 204, 200, 190, 252, 189, 119, 178, 182, 187, 212, 121, 194, 208, 220, 236, 198, 125, 185, 244, 191, 193, 202, 248, 197, 216, 228, 224, 123, 133, 195, 129, 206, 232, 250, 240, 205, 137, 141, 201, 181, 183, 246, 254, 218, 127, 199, 210, 173, 179, 203, 157, 253, 131, 214, 177, 149, 171, 234, 249, 139, 145, 153, 222, 242, 230, 135, 251, 169, 238, 245, 155, 165, 213, 221, 209, 207, 237, 241, 217, 226, 147, 175, 161, 143, 243, 163, 151, 211, 167, 247, 233, 235, 219, 159, 229, 225, 239, 227, 215, 231, 223

Pour montrer l'inconsistance du système nous allons procéder comme suit

- nous examinons les équations une par une dans l'ordre donné ci-dessus, jusqu'à obtenir une équation résoluble
- Aprés résolution d'une équation, nous recommencons l'examen depuis le debut de la liste.
  à chaque étape de la résolution, nous prenons en compte toutes les informations obtenues jusqu'alors

Montrons d'abord que $A_{61} \neq 0$

Supposons que $A_{61} = 0$. Alors

$$eq_{196}: \ \mathbf{A_{85}^3} = 0 \ \Rightarrow \ \mathbf{A_{85}} := 0,$$

$$eq_{208}: \ \mathbf{A_{87}^6} = 0 \ \Rightarrow \ \mathbf{A_{87}} := 0,$$

$$eq_{236}: \ 1 = 0,$$

donc $A_{61} \neq 0$.

Nous donnons ici, dans l'ordre de résolution, toutes les équations "résolubles" ainsi que l'information que nous en tirons

$$eq_{180} \quad : \quad A_{61}^{8}A_{85}A_{63}^{4} + A_{85}A_{63}^{8} + A_{61}^{4}\mathbf{A_{91}^{4}} + A_{61}^{29} + A_{61}^{3}A_{85}^{2} + A_{61}^{16}A_{85} + A_{61}^{12}A_{87}^{4} + A_{87}^{16} = 0$$

$$\Rightarrow \ \mathbf{A_{91}} \ := \ A_{61}A_{85}A_{63} + A_{61}^{254}A_{85}A_{63}^{2} + A_{61}^{2}A_{87} + A_{61}^{70} + A_{61}^{191}A_{85}^{2} + A_{61}^{3}A_{85} + A_{61}^{254}A_{87}^{4}$$

$$\begin{aligned} eq_{196} \quad : \quad & \mathbf{A_{95}^{2}}A_{61}^{3} + A_{87}^{8}A_{61}A_{63}^{2} + A_{85}^{2}A_{61}A_{63}^{6} + A_{87}^{2}A_{61}^{3}A_{63}^{4} + A_{85}^{3} + A_{61}^{131}A_{63}^{8} + A_{61}^{139}A_{63}^{4} \\ & + A_{61}^{130}A_{87}^{4} + A_{61}^{5}A_{85}^{2}A_{63}^{4} = 0 \end{aligned}$$

$$\begin{aligned} \Rightarrow \ \mathbf{A_{95}} \ := \ & A_{61}^{126}A_{85}^{3} + A_{61}^{254}A_{87}^{4}A_{63} + A_{61}^{254}A_{85}A_{63}^{3} + A_{87}A_{63}^{2} + A_{61}A_{85}A_{63}^{2} + A_{61}^{64}A_{63}^{4} \\ & + A_{61}^{68}A_{63}^{2} + A_{61}^{191}A_{87}^{2} \end{aligned}$$

$$eq_{192} : \ \mathbf{A_{85}^{3}} = 0 \ \Rightarrow \ \mathbf{A_{85}} = 0$$

$$eq_{200} : \ 1 = 0$$

□

1.2 Le code $B(255, 61)$ n'admet pas de mots de poids 61

Nous considérons les équations de Newton eq_r pour $0 < r \leq n = 255$, pour le code $B(255, 61)$, et pour le poids $\delta = 61$. Nous voulons montrer qu'il n'existe pas de mots de code de poids δ.

D'après la propriété de conjugaison des fonctions puissances symétriques, et en tenant compte de l'ensemble de définition du code, les seules fonctions puissances symétriques inconnues sont

$$A_{61}, \ A_{63}, \ A_{85}, \ A_{87}, \ A_{91}, \ A_{95}, \ A_{111}, \ A_{119}, \ A_{127},$$

et puisque 255 et 61 sont premier entre eux, on peut supposer $A_{61} = 1$.

On a $eq_{61} \ : \ \sigma_{61} = A_{61} = 1$, et les σ_i pour i impair, $1 \leq i < 61$ sont nuls.

Les équations de Newton eq_i, pour i impair, $63 \leq i \leq 123$ nous fournissent un système triangulaire linéaire donnant les σ_i, pour i pair, $2 \leq i \leq 60$ ce système est formé des 30 équations suivantes

$$\begin{aligned} eq_{63} \ &: \ A_{63} + \sigma_2 = 0 \\ eq_{65} \ &: \ A_{63}\sigma_2 + \sigma_4 = 0 \\ eq_{67} \ &: \ A_{63}\sigma_4 + \sigma_6 = 0 \\ eq_{69} \ &: \ A_{63}\sigma_6 + \sigma_8 = 0 \\ eq_{71} \ &: \ A_{63}\sigma_8 + \sigma_{10} = 0 \end{aligned}$$

$$
\begin{aligned}
eq_{73} &: A_{63}\sigma_{10} + \sigma_{12} = 0 \\
eq_{75} &: A_{63}\sigma_{12} + \sigma_{14} = 0 \\
eq_{77} &: A_{63}\sigma_{14} + \sigma_{16} = 0 \\
eq_{79} &: 1 + A_{63}\sigma_{16} + \sigma_{18} = 0 \\
eq_{81} &: \sigma_2 + A_{63}\sigma_{18} + \sigma_{20} = 0 \\
eq_{83} &: \sigma_4 + A_{63}\sigma_{20} + \sigma_{22} = 0 \\
eq_{85} &: A_{85} + \sigma_6 + A_{63}\sigma_{22} + \sigma_{24} = 0 \\
eq_{87} &: A_{87} + A_{85}\sigma_2 + \sigma_8 + A_{63}\sigma_{24} + \sigma_{26} = 0 \\
eq_{89} &: A_{87}\sigma_2 + A_{85}\sigma_4 + \sigma_{10} + A_{63}\sigma_{26} + \sigma_{28} = 0 \\
eq_{91} &: A_{91} + A_{87}\sigma_4 + A_{85}\sigma_6 + \sigma_{12} + A_{63}\sigma_{28} + \sigma_{30} = 0 \\
eq_{93} &: A_{87}^4 + A_{91}\sigma_2 + A_{87}\sigma_6 + A_{85}\sigma_8 + \sigma_{14} + A_{63}\sigma_{30} + \sigma_{32} = 0 \\
eq_{95} &: A_{95} + A_{87}^4\sigma_2 + A_{91}\sigma_4 + A_{87}\sigma_8 + A_{85}\sigma_{10} + \sigma_{16} + A_{63}\sigma_{32} + \sigma_{34} = 0 \\
eq_{97} &: A_{95}\sigma_2 + A_{87}^4\sigma_4 + A_{91}\sigma_6 + A_{87}\sigma_{10} + A_{85}\sigma_{12} + \sigma_{18} + A_{63}\sigma_{34} + \sigma_{36} = 0 \\
eq_{99} &: A_{95}\sigma_4 + A_{87}^4\sigma_6 + A_{91}\sigma_8 + A_{87}\sigma_{12} + A_{85}\sigma_{14} + \sigma_{20} + A_{63}\sigma_{36} + \sigma_{38} = 0 \\
eq_{101} &: A_{95}\sigma_6 + A_{87}^4\sigma_8 + A_{91}\sigma_{10} + A_{87}\sigma_{14} + A_{85}\sigma_{16} + \sigma_{22} + A_{63}\sigma_{38} + \sigma_{40} = 0 \\
eq_{103} &: A_{95}\sigma_8 + A_{87}^4\sigma_{10} + A_{91}\sigma_{12} + A_{87}\sigma_{16} + A_{85}\sigma_{18} + \sigma_{24} + A_{63}\sigma_{40} + \sigma_{42} = 0 \\
eq_{105} &: A_{95}\sigma_{10} + A_{87}^4\sigma_{12} + A_{91}\sigma_{14} + A_{87}\sigma_{18} + A_{85}\sigma_{20} + \sigma_{26} + A_{63}\sigma_{42} + \sigma_{44} = 0 \\
eq_{107} &: A_{91}^{32} + A_{95}\sigma_{12} + A_{87}^4\sigma_{14} + A_{91}\sigma_{16} + A_{87}\sigma_{20} + A_{85}\sigma_{22} + \sigma_{28} + A_{63}\sigma_{44} + \sigma_{46} = 0 \\
eq_{109} &: A_{91}^4 + A_{91}^{32}\sigma_2 + A_{95}\sigma_{14} + A_{87}^4\sigma_{16} + A_{91}\sigma_{18} + A_{87}\sigma_{22} + A_{85}\sigma_{24} + \sigma_{30} + A_{63}\sigma_{46} \\
&\quad +\sigma_{48} = 0 \\
eq_{111} &: A_{111} + A_{91}^4\sigma_2 + A_{91}^{32}\sigma_4 + A_{95}\sigma_{16} + A_{87}^4\sigma_{18} + A_{91}\sigma_{20} + A_{87}\sigma_{24} + A_{85}\sigma_{26} + \sigma_{32} \\
&\quad +A_{63}\sigma_{48} + \sigma_{50} = 0 \\
eq_{113} &: A_{111}\sigma_2 + A_{91}^4\sigma_4 + A_{91}^{32}\sigma_6 + A_{95}\sigma_{18} + A_{87}^4\sigma_{20} + A_{91}\sigma_{22} + A_{87}\sigma_{26} + A_{85}\sigma_{28} + \sigma_{34} \\
&\quad +A_{63}\sigma_{50} + \sigma_{52} = 0 \\
eq_{115} &: A_{111}\sigma_4 + A_{91}^4\sigma_6 + A_{91}^{32}\sigma_8 + A_{95}\sigma_{20} + A_{87}^4\sigma_{22} + A_{91}\sigma_{24} + A_{87}\sigma_{28} + A_{85}\sigma_{30} + \sigma_{36} \\
&\quad +A_{63}\sigma_{52} + \sigma_{54} = 0 \\
eq_{117} &: A_{87}^{16} + A_{111}\sigma_6 + A_{91}^4\sigma_8 + A_{91}^{32}\sigma_{10} + A_{95}\sigma_{22} + A_{87}^4\sigma_{24} + A_{91}\sigma_{26} + A_{87}\sigma_{30} + A_{85}\sigma_{32} \\
&\quad +\sigma_{38} + A_{63}\sigma_{54} + \sigma_{56} = 0 \\
eq_{119} &: A_{119} + A_{87}^{16}\sigma_2 + A_{111}\sigma_8 + A_{91}^4\sigma_{10} + A_{91}^{32}\sigma_{12} + A_{95}\sigma_{24} + A_{87}^4\sigma_{26} + A_{91}\sigma_{28} + A_{87}\sigma_{32} \\
&\quad +A_{85}\sigma_{34} + \sigma_{40} + A_{63}\sigma_{56} + \sigma_{58} = 0 \\
eq_{121} &: A_{119}\sigma_2 + A_{87}^{16}\sigma_4 + A_{111}\sigma_{10} + A_{91}^4\sigma_{12} + A_{91}^{32}\sigma_{14} + A_{95}\sigma_{26} + A_{87}^4\sigma_{28} + A_{91}\sigma_{30} + A_{87}\sigma_{34} \\
&\quad +A_{85}\sigma_{36} + \sigma_{42} + A_{63}\sigma_{58} + \sigma_{60} = 0
\end{aligned}
$$

la résolution de ce système nous donne l'expression des σ_i en fonction des A_i

$$
\begin{aligned}
\sigma_2 &:= A_{63} \\
\sigma_4 &:= A_{63}^2 \\
\sigma_6 &:= A_{63}^3
\end{aligned}
$$

$$
\begin{aligned}
\sigma_8 &:= A_{63}^4\\
\sigma_{10} &:= A_{63}^5\\
\sigma_{12} &:= A_{63}^6\\
\sigma_{14} &:= A_{63}^7\\
\sigma_{16} &:= A_{63}^8\\
\sigma_{18} &:= 1 + A_{63}^9\\
\sigma_{20} &:= A_{63}^{10}\\
\sigma_{22} &:= A_{63}^2 + A_{63}^{11}\\
\sigma_{24} &:= A_{85} + A_{63}^{12}\\
\sigma_{26} &:= A_{87} + A_{63}^4 + A_{63}^{13}\\
\sigma_{28} &:= A_{85}A_{63}^2 + A_{63}^{14}\\
\sigma_{30} &:= A_{91} + A_{87}A_{63}^2 + A_{63}^6 + A_{63}^{15}\\
\sigma_{32} &:= A_{87}^4 + A_{85}A_{63}^4 + A_{63}^{16}\\
\sigma_{34} &:= A_{95} + A_{91}A_{63}^2 + A_{87}A_{63}^4 + A_{63}^8 + A_{63}^{17}\\
\sigma_{36} &:= A_{87}^4A_{63}^2 + A_{85}A_{63}^6 + 1 + A_{63}^{18}\\
\sigma_{38} &:= A_{95}A_{63}^2 + A_{91}A_{63}^4 + A_{87}A_{63}^6 + A_{63}^{10} + A_{63} + A_{63}^{19}\\
\sigma_{40} &:= A_{87}^4A_{63}^4 + A_{85}A_{63}^8 + A_{63}^{20}\\
\sigma_{42} &:= A_{95}A_{63}^4 + A_{91}A_{63}^6 + A_{87}A_{63}^8 + A_{63}^{12} + A_{63}^{21}\\
\sigma_{44} &:= A_{87}^4A_{63}^6 + A_{85}A_{63}^{10} + A_{63}^4 + A_{63}^{22}\\
\sigma_{46} &:= A_{91}^{32} + A_{95}A_{63}^6 + A_{91}A_{63}^8 + A_{87}A_{63}^{10} + A_{63}^{14} + A_{63}^5 + A_{63}^{23}\\
\sigma_{48} &:= A_{91}^4 + A_{87}^4A_{63}^8 + A_{85}^2 + A_{85}A_{63}^{12} + A_{63}^{24}\\
\sigma_{50} &:= A_{111} + A_{91}^{32}A_{63}^2 + A_{95}A_{63}^8 + A_{91}A_{63}^{10} + A_{87}A_{63}^{12} + A_{63}^{16} + A_{63}A_{85}^2 + A_{63}^{25}\\
\sigma_{52} &:= A_{63}^8 + A_{87}^2 + A_{63}^{26} + A_{91}^4A_{63}^2 + A_{87}^4A_{63}^{10} + A_{85}A_{63}^{14}\\
\sigma_{54} &:= 1 + A_{63}^9 + A_{91}^{32}A_{63}^4 + A_{63}^{27} + A_{95}A_{63}^{10} + A_{91}A_{63}^{12} + A_{87}A_{63}^{14} + A_{63}A_{87}^2 + A_{111}A_{63}^2 + A_{63}^{18}\\
\sigma_{56} &:= A_{91}^4A_{63}^4 + A_{87}^{16} + A_{63}^{28} + A_{87}^4A_{63}^{12} + A_{85}^2A_{63}^4 + A_{85}A_{63}^{16}\\
\sigma_{58} &:= A_{111}A_{63}^4 + A_{119} + A_{63}^{29} + A_{91}^{32}A_{63}^6 + A_{95}A_{63}^{12} + A_{91}A_{63}^{14} + A_{87}A_{63}^{16} + A_{85}^2A_{63}^5 + A_{63}^{20}\\
\sigma_{60} &:= A_{85} + A_{91}^2 + A_{87}^2A_{63}^4 + A_{85}A_{63}^{18} + A_{87}^{16}A_{63}^2 + A_{91}^4A_{63}^6 + A_{87}^4A_{63}^{14} + A_{63}^{30} + A_{63}^{12}
\end{aligned}
$$

Ces valeurs des σ_i sont remplacées par leurs valeurs dans les équations restantes. Après remplacement et développement les équations sont triées par ordre croissant de taille, la taille d'une équation étant égale au nombre de ses monômes, on obtient la liste suivante

186, 190, 188, 194, 198, 192, 202, 123, 184, 189, 191, 196, 206, 254, 127, 195, 200, 210, 135, 193, 187, 199, 214, 125, 131, 204, 252, 197, 222, 203, 238, 129, 139, 208, 250, 143, 201, 218, 133, 137, 226, 230, 246, 248, 234, 185, 212, 242, 207, 141, 181, 216, 236, 244, 151, 183, 205, 220, 232, 147, 224, 159, 179, 211, 240, 149, 175, 155, 145, 157, 209, 173, 163, 167, 228, 153, 171, 215, 253, 251, 165, 169, 161, 177, 213, 223, 239, 247, 249, 243, 217, 235, 237, 245, 219, 221, 229, 231, 233, 227, 225, 241

Pour montrer l'inconsistance du système nous allons procéder comme suit

- nous examinons les équations une par une dans l'ordre donné ci-dessus, jusqu'à obtenir une équation résoluble

- Aprés résolution d'une équation, nous recommencons l'examen depuis le debut de la liste.
  à chaque étape de la résolution, nous prenons en compte toutes les informations obtenues jusqu'alors

Nous donnons ici, dans l'ordre de résolution, toutes les équations "résolubles" ainsi que l'information que nous en tirons

$$eq_{186} \quad : \quad A_{87}^8 + A_{85}^2 A_{63}^8 + A_{85} A_{63}^{20} + A_{87}^{16} A_{63}^4 + A_{91}^4 A_{63}^8 + A_{87}^4 A_{63}^{16} + A_{63}^{32} + \mathbf{A_{95}^4} = 0$$

$$\Rightarrow \quad \mathbf{A_{95}} \quad := \quad A_{87}^2 + A_{85}^2 A_{63}^2 + A_{85} A_{63}^5 + A_{87}^4 A_{63} + A_{91} A_{63}^2 + A_{87} A_{63}^4 + A_{63}^8$$

$$\begin{aligned} eq_{188} \quad : \quad & A_{87}^8 A_{63} + A_{91}^2 A_{63}^3 + A_{87}^2 A_{63}^7 + A_{85}^2 + A_{85}^2 A_{63}^9 + A_{91} + A_{87} A_{63}^2 + A_{63}^6 + A_{63}^{15} \\ & + \mathbf{A_{127}} = 0 \end{aligned}$$

$$\Rightarrow \quad \mathbf{A_{127}} \quad := \quad A_{87}^8 A_{63} + A_{91}^2 A_{63}^3 + A_{87}^2 A_{63}^7 + A_{85}^2 + A_{85}^2 A_{63}^9 + A_{91} + A_{87} A_{63}^2 + A_{63}^6 + A_{63}^{15}$$

$$eq_{194} : \ \mathbf{A_{85}^3} + 1 = 0 \ \Rightarrow \ \mathbf{A_{85}} \neq 0$$

$$eq_{198} : \ A_{85} \mathbf{A_{87}^2} + A_{63}^2 = 0 \ \Rightarrow \ \mathbf{A_{87}} := A_{63} A_{85}$$

$$\begin{aligned} eq_{187} \quad : \quad & 1 + A_{91} A_{85} A_{63}^{18} + A_{85}^2 A_{91} A_{63}^6 + A_{91}^4 A_{63}^{21} + A_{63}^2 + A_{63}^{142} + A_{63}^{130} A_{85} + A_{111}^{128} A_{63}^2 \\ & + A_{91}^{16} A_{63}^3 + A_{85}^2 A_{63}^{39} + \mathbf{A_{119}^8} + A_{91}^6 A_{63}^9 + A_{91}^2 A_{63}^{33} + A_{91}^4 A_{63}^3 + A_{85}^2 A_{91}^2 \\ & + A_{85}^2 A_{63}^{21} + A_{63}^{36} + A_{63}^{15} A_{85} + A_{91} A_{63}^{30} + A_{91}^3 + A_{85} A_{63}^{24} + A_{85}^2 A_{63}^{15} A_{91}^4 \\ & + A_{91}^2 A_{63}^{21} A_{85} + A_{63}^9 A_{85} A_{91}^4 + A_{91}^4 A_{63}^{12} + A_{91}^2 A_{63}^6 + A_{91}^5 A_{63}^6 + A_{85}^2 A_{91}^4 A_{63}^6 \\ & + A_{85}^2 A_{63}^{30} + A_{63}^{45} + A_{91} A_{85} + A_{91} A_{63}^{12} + A_{63}^3 A_{85}^2 = 0 \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad \mathbf{A_{119}} \quad := \quad & 1 + A_{111}^{16} A_{63}^{64} + A_{85} A_{63}^{228} + A_{91}^{160} A_{63}^{192} + A_{91}^{32} A_{85}^2 A_{63}^{66} + A_{63}^{64} + A_{85} A_{63}^{225} A_{91}^{128} \\ & + A_{63}^{80} A_{85}^2 + A_{63}^{225} A_{85}^2 + A_{91}^{32} A_{63}^{195} + A_{85} A_{63}^{195} + A_{91}^2 A_{63}^{96} + A_{91}^{64} A_{85} + A_{63}^{132} \\ & + A_{91}^{128} A_{63}^{96} + A_{91}^{192} A_{63}^{33} + A_{91}^{64} A_{63}^{162} A_{85}^2 + A_{63}^{96} A_{85} + A_{63}^{33} A_{85}^2 A_{91}^{128} + A_{85} A_{91}^{128} A_{63}^{192} \\ & + A_{91}^{32} A_{63}^{129} + A_{63}^{209} + A_{91}^{128} A_{63}^{162} + A_{63}^{165} + A_{85} A_{91}^{32} A_{63}^{192} + A_{63}^3 A_{85}^2 + A_{91}^{128} A_{63}^{129} \\ & + A_{91}^{64} A_{63}^{36} + A_{85} A_{63}^{162} + A_{91}^{64} A_{63}^{192} + A_{91}^{96} + A_{91}^{32} A_{85}^2 \end{aligned}$$

$$\begin{aligned}
eq_{189} \quad : \quad & A_{91}^2 A_{63}^{34} + A_{91}^4 A_{85} A_{63}^{10} + A_{91} A_{63}^{31} + A_{63} A_{91}^3 + A_{91}^4 A_{63}^4 + A_{85} A_{63}^{25} + A_{85}^2 A_{63}^{22} \\
& + A_{91} A_{85} A_{63}^{19} + A_{63}^7 A_{91}^2 + A_{91}^4 A_{63}^{13} + A_{91}^5 A_{63}^7 + A_{85}^2 A_{63}^{31} A_{63} + A_{63}^2 + A_{63}^{128} A_{91} \\
& + A_{63}^{37} + \mathbf{A_{111}^4} + A_{91} A_{63}^{13} + A_{91}^4 A_{63}^{22} + A_{63}^{128} A_{85}^2 + A_{91}^2 A_{63} A_{85}^2 + A_{85}^2 A_{63}^4 \\
& + A_{85} A_{63}^{16} + A_{63}^3 + A_{63}^{134} + A_{85}^2 A_{63}^{40} + A_{91}^6 A_{63}^{10} + A_{63}^{46} + A_{85}^2 A_{91} A_{63}^7 \\
& + A_{85}^2 A_{63}^{16} A_{91}^4 + A_{91}^2 A_{63}^{22} A_{85} + A_{85}^2 A_{91}^4 A_{63}^7 + A_{63} A_{91} A_{85} = 0
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \ \mathbf{A_{111}} \ := \ & A_{91} A_{85}^2 A_{63}^4 + A_{63}^{133} A_{85}^2 + A_{91} A_{85} A_{63}^{130} + A_{91}^{64} A_{85} A_{63}^{196} + A_{91}^{128} A_{63}^{64} A_{85}^2 \\
& + A_{85}^2 A_{91}^{64} A_{63}^{193} + A_{63}^{70} A_{85} + A_{63}^{64} + A_{63}^{128} + A_{63}^{139} + A_{85}^2 A_{63}^{10} + A_{91}^{129} A_{63}^{130} \\
& + A_{91}^{128} A_{63}^{133} A_{85} + A_{85}^2 A_{91} A_{63}^{193} + A_{63} A_{85}^2 + A_{63}^{32} A_{85}^2 + A_{63}^{64} A_{91}^{64} A_{85} + A_{63}^{73} \\
& + A_{91} A_{63} + A_{85} A_{63}^4 + A_{63}^{133} A_{91} + A_{91}^{65} A_{63}^{193} + A_{85}^2 A_{63}^{199} + A_{91}^{64} A_{63}^{67} + A_{63}^{161} \\
& + A_{91}^{64} A_{63}^{199} + A_{91} A_{63}^{67} + A_{91}^{128} A_{63}^{136} + A_{63}^{193} A_{91}^{128} + A_{63}^{32} A_{91}^{64} + A_{63}^{192} + A_{63}^{64} A_{91}^{192}
\end{aligned}$$

$$eq_{199} : \ \mathbf{A_{91}^{16}} = 0 \ \Rightarrow \ \mathbf{A_{91}} := 0$$

$$eq_{203} : \ 1 = 0$$

□

1.3 Le code $B(511, 123)$ n'admet pas de mots de poids 123

Nous ne donnerons pas ici une preuve complète, seulement les principales étapes menant à la contradiction. Les indications que nous donnons sont suffisantes pour reconstituer la preuve.

Les σ_i peuvent être tous obtenus en fonction des A_i, nous supposons que cette opération a été effectuée et que les substitutions correspondantes ont été faites dans les équations de Newton. Nous commencons par montrer que $A_{125} \neq 0$, en effet si on suppose $A_{125} = 0$, alors

$$eq_{396} \ \Rightarrow \ \mathbf{A_{171}} := 0,$$

puis

$$eq_{416} \ \Rightarrow \ \mathbf{A_{175}} := 0,$$

et

$$eq_{492} : \ 1 = 0.$$

Voici dans les étapes de la résolution

$$eq_{372} \ \Rightarrow \ \mathbf{A_{187}} := \ldots$$

$$eq_{388} \ \Rightarrow \ \mathbf{A_{191}} := \ldots$$

$$eq_{392} \ \Rightarrow \ \mathbf{A_{171}} := \ldots$$

$$eq_{404} \Rightarrow \mathbf{A_{175}} := \ldots$$
$$eq_{412} \Rightarrow \mathbf{A_{125}} := \ldots$$
$$eq_{420} + eq_{428} \Rightarrow 1 = 0$$

ce qui constitue la contradiction. □

2 Recherche d'idempotents

La recherche des idempotents dans un code à l'aide des équations de Newton prend une forme particulièrement simple, car les fonctions symétriques élémentaires et les fonctions puissances symétriques d'un idempotent sont dans $\mathbb{F}_2$.

Nous procéderons comme suit pour la résolution

1. on écrit les équations de Newton pour le poids considéré,
2. on prend en compte les propriétés du code $B(n, \delta)$, c'est-à-dire $\sigma_i = 0$ pour $i < \delta$ impair, et $A_i = 0$ pour $0 < i < \delta$,
3. J l'ensemble des minima des classes cyclotomiques, on élimine tous les A_i sauf pour $i \in J$,
4. on pose $A_0 = w \bmod 2$ et si $\mathrm{pgcd}(n, \delta) = 1$, $A_\delta = 1$,
5. on résoud le système inversible donné par la proposition II.9 ou la proposition II.10,
6. le système résultant a pour seules indéterminées les A_i pour $i \in J$, et est polynomial de degré 1 en chacun des A_i,
7. pour les 8 premiers A_i, on essaie toutes les valeurs possibles, nous avons donc 256 systèmes à résoudre.

Chacun de ces 256 systèmes peut être résolu en un temps relativement court, dans la plupart des cas le système est inconsistant, et la contradiction apparait alors rapidement, c'est-à-dire dans un temps de l'ordre de la dizaine de secondes. Chaque fois que le système trouve une solution ce qui prend un temps de l'ordre de la minute, il reste à vérifier que $\sigma(z) | z^n - 1$.

La méthode utilisée pour la recherche des solutions peut sembler peu adaptée pour résoudre des équations algébriques. Cependant il faut préciser que la recherche de contradiction dans les équations de Newton est possible sans relations supplémentaires, car en général l'inconsistance apparait rapidement et ne nécessite pas la totalité du système. La recherche d'une solution particulière demande en revanche la résolution du système dans sa totalité, ce qui demande un espace mémoire très important en longueur 511.

Le compromis temps/mémoire que nous avons utilisé a été choisi empiriquement après plusieurs essais, et nous a semblé être le meilleur pour la longueur 511.

Annexe B

Mise en œuvre de l'algorithme d'Euclide étendu en MAPLE et en C

1 Les problèmes posés

1.1 Les calculs dans les corps finis

Le décodage des codes BCH par l'algorithme d'Euclide nécessite des calculs dans une extension de corps fini, cette extension sera celle dans laquelle le polynôme générateur est entièrement scindé, si par exemple le code défini sur $\mathbb{F}_q$ est tel que $n = q^m - 1$ avec $q = p^r$, l'extension devra être de degré $r \times m$ sur le corps premier $\mathbb{F}_p$.

En général les librairies des langages de programmation et des systèmes de calcul formel ne prévoient pas le calcul dans les extensions de corps finis (sauf Scratchpad avec quelques aménagements cf. [3]), or ces calculs posent certains problèmes algorithmiques:

(i) Il faut, pour pouvoir définir l'extension, disposer d'un polynôme irréductible primitif de degré convenable.

(ii) Il faut ensuite choisir une représentation des éléments du corps. Soit α une racine primitive du corps:

- on peut choisir de représenter les éléments du corps comme des puissances de α, c'est-à-dire en fait que l'on utilise le logarithme en base α, la multiplication est alors très simple, puisqu'il s'agit simplement d'une addition modulaire des exposants. L'addition par contre est difficile, il faut résoudre l'équation: $\alpha^k = \alpha^i + \alpha^j$, or cette équation est d'une grande complexité algorithmique.
- la seconde solution consiste à représenter les éléments du corps comme des polynômes en α de degré inférieur au degré de l'extension. Les opérations élémentaires sur le corps deviennent alors une addition modulaire de polynômes pour l'addition, et une division modulaire de polynômes pour la multiplication.

(iii) Enfin, il faudra que les opérations élémentaires puissent être effectuées rapidement. Dans le cadre de ce travail il n'est pas nécessaire de pouvoir utiliser de grand corps, nous nous sommes donc intéressé à des corps de cardinal relativement faible.

1.2 L'algorithmique

La mise en œuvre de l'algorithme d'Euclide pour le décodage des codes BCH demande essenciellement 3 fonctions en plus des opérations élémentaires: la division de deux polynômes, l'évaluation d'un polynôme en un élément du corps et le calcul des racines d'un polynôme. Nous discuterons des problèmes et des choix algorithmiques faits pour ces fonctions dans les sections suivantes.

2 Mise en œuvre en *C*

Pour la mise en œuvre en C de l'algorithme d'Euclide pour le décodage des codes de Reed-Solomon, nous avons choisi de représenter les éléments du corps par leur logarithme discret en base α, où α est un élément générateur du groupe multiplicatif du corps étendu, plus précisement, nous avons utilisé la convention suivante: "0" représente le zéro du corps, "1" l'unité et "i" l'élément α^{i-1}. Ceci nous permet d'obtenir une opération de multiplication simple:

$$[i] \times [j] = \begin{cases} [((i-1)+(j-1) \bmod (q-1))+1] & \text{si } i \neq 0 \text{ et } j \neq 0 \\ [0] & \text{sinon} \end{cases}$$

Pour l'addition nous avons choisi d'utiliser le logarithme de Zech Z défini pour tout entier n par:

$$1 + \alpha^n = \alpha^{Z(n)}$$

l'addition de α^i et α^j sécrit alors:

$$\alpha^i + \alpha^j = \alpha^i(1+\alpha^{j-i}) = \alpha^{i+Z(j-i)}$$

donc:

$$[i] + [j] = \begin{cases} [((i-1)+Z(j-i) \bmod (q-1))+1] & \text{si } i \neq 0 \\ [j] & \text{sinon} \end{cases}$$

Il suffit donc de mettre en table la fonction Z, pour obtenir des opérations élémentaires rapides.

Le calcul des racines d'un polynôme dans $\mathbb{F}_q$ est effectué par examen successif de tous les éléments du corps, cela n'a bien sur été possible que parceque nous nous sommes limités à des corps de petite taille, et nécessite bien entendu une opération d'évaluation d'un polynôme en un élément du corps rapide.

Pour l'évaluation d'un polynôme nous avons utilisé en C l'algorithme de Horner qui permet d'évaluer un polynôme de degré d en d additions et d multiplications

La mise en table du logarithme de Zech ainsi que ces choix algorithmiques nous ont permis d'obtenir en C un décodage relativement rapide, c'est-à-dire par exemple pour le code de Reed-Solomon de longueur 15 et de dimension 7 le décodage d'un mot de code prend environ $1/100^{\text{ième}}$ de seconde, et pour le code RS(255,223) environ 1/2 seconde. Ces calculs ont été effectués sur une station de travail SUN 3/60 12Mo.

3 Mise en œuvre en *MAPLE*

3.1 Les choix

Nous avons choisi, pour *MAPLE* de représenter les éléments d'une extension d'un corps fini premier par des polynômes ayant pour indéterminée un élément primitif a de l'extension. La principale motivation de ce choix est que les polynômes sont des objets très adaptés au langage *MAPLE*. Nous générons cependant la table des logarithmes des élément du corps, ainsi que la table des puissances de a, cela nous limite à des corps de petite taille (jusqu'à $\mathbb{F}_{256}$ en pratique), mais était necessaire pour obtenir des temps de calcul raisonnables.

l'addition: l'addition de deux éléments du corps se ramène à une addition de polynômes suivie d'un modulo.

la multiplication il faut multiplier les deux polynômes en a puis calculer le reste du polynôme produit par le polynôme générateur du corps. En pratique, pour des raisons d'efficacité nous effectuons le produit des polynômes dans le corps de base puis substituons aux monômes de degré trop elevé leur valeur dans le corps. par exemple pour $\mathbb{F}_{16}$: défini avec le polynôme $a^4 + a + 1$:

$$\begin{aligned}
(a^3 + a)(a^2 + a + 1) &\Rightarrow a^5 + a^4 + 2a^3 + a^2 + a \\
&\Rightarrow (a^2 + a) + (a + 1) + 2a^3 + a^2 + a \\
&\Rightarrow 2a^3 + 2a^2 + 3a + 1 \\
&\Rightarrow a + 1.
\end{aligned}$$

le calcul des racines d'un polynôme comme dans notre mise en œuvre en *C* nous calculons les racines d'un polynôme par examen de tous les éléments du corps.

évaluation d'un polynôme nous n'avons pas utilisé ici l'algorithme de Horner, nous avons préféré une autre méthode qui utilise mieux les possibilités de *MAPLE*: les opérations d'expansion d'un polynôme et de substitution d'une indéterminée sont ecrites en *C* et compilées, donc extrèmement rapides, le problème se ramenait donc essentiellement, après avoir substitué à l'indéterminée une puissance de a, à la la réduction d'un polynôme de degré élevé en a en un polynôme de degré inférieur au degré de l'extension. Cette opération est effectuée par la fonction *rat*, qui transforme le monôme de plus haut degré d'un polynôme en a en un polynôme de degré convenable, et se rappelle recursivement.

3.2 Le module "corps-finis"

Une description des fonctions les plus importantes de ce module est donnée en section 3.3.

L'initialisation d'une extension de corps premier se fait par l'appel à la fonction *GF*, dont l'argument sera le cardinal du corps. La variable a est reservée pour représenter un élément générateur du corps. Le polynôme servant à définir le corps n'est pas calculé, et doit donc être fourni par l'utilisateur, soit en l'ajoutant au fichier "Primitif" directement dans les sources, soit en utilisant la fonction *Primitif* au cours d'une session, ceci devra de toute façon être fait avant l'appel à *GF*. Le polynôme générateur du corps est stoqué dans la variable *tellrat*. Nous noterons $\mathbb{F}_p$ le corps premier et $\mathbb{F}_q$ l'extension (on a $q = p^m$).

L'initialisation du corps peut être longue puisque c'est au cours de cette étape que sera calculée la table des logarithmes. Cependant, ce calcul ne sera effectué q'une seule fois car la table est sauvée dans un fichier dans les librairies. À titre d'exemple, l'initialisation du corps $\mathbb{F}_{1024}$ prends la première fois environ 13 secondes et 3 secondes les fois suivantes.

Les fonctions du module sont sommairement décrites plus bas, nous nous contenterons de décrire un peu plus en détail la fonction *rat* qui réduit une fraction rationnelle de $\mathbf{Q}(a, x_1, \ldots, x_s)$, en un élément de $\mathbb{F}_q(x_1, \ldots, x_n)$. *rat* se rappelle pour le numérateur et le dénominateur, puis pour chacun de ces polynômes, elle s'appelle récursivement en réduisant à chaque fois le nombre d'indéterminée jusqu'à obtenir un polynôme en a. Ce polynôme en a est à son tour réduit en remplaçant successivement le monôme de plus haut degré en a en un polynôme de degré convenable en utilisant la table des puissances de a qui est calculée à l'initialisation du corps. La fonction *rat* représente les éléments du corps par des polynômes de degré inférieur au degré de l'extension, le module possède également la fonction *rexp* qui remplit la même fonction que *rat*, mais représente les éléments du corps comme des puissances de a.

En plus des fonctions necessaires à la mise en œuvre de l'algorithme d'Euclide, nous avons complété ce module avec les fonctions suivantes:

le calcul du pgcd de deux polynômes à une indéterminée

la factorisation d'un polynôme à une indéterminée

le calcul du polynôme minimal d'un élément du corps

le calcul de la classe cyclotomique d'un élément du corps

le calcul de l'ordre d'un élément du corps

La factorisation utilise l'algorithme de Berlekamp, et n'est donc utilisable que pour de petits corps.

Le calcul du polynôme minimal utilise l'algorithme suivant:
soit $p = resultant(x - e, tellrat, a)$ le résultant des polynômes $x - e$ (où $e \in \mathbb{F}_q$) et $tellrat$ (le polynôme générateur du corps) par rapport à la variable a, calculé dans $\mathbb{F}_p[x]$ (on considère $e \in \mathbb{F}_p[a]$,alors le polynôme minimal de e est le plus grand diviseur de p n'ayant aucun facteur multiple.

3.3 Documentation du module corps-fini en MAPLE

FONCTION : **GF**

APPEL : $GF(q)$;

ARGUMENTS : q – puissance de premier . $q = p^r$

DESCRIPTION :

- Affecte à la variable *tellrat* la valeur d'un polynôme générateur du corps de base. Ce polynôme est défini par l'appel *Primitif(p,r)*.
- L'indéterminée de ce polynôme est a qui représente un élément générateur du corps ayant *tellrat* comme polynôme minimal.
- Le degré de *tellrat* est égal à r.
- La variable a est réservée pour toute la suite, et ne doit donc pas être affectée.
- Affecte à la variable *Modulus* la caractéristique du corps.
- Affecte à la variable *argGF* le cardinal du corps $(= q)$.
- ATTENTION, la fonction *Primitif*, qui renvoie un polynôme irreductible primitif n'effectue aucun calcul et prend en fait ses valeur dans une table qu'il convient de compléter le cas échéant.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $GF(27)$;

$$3, a^3 + 2a + 1, 27$$

VOIR AUSSI : Primitif, algebraic, rat, rexp.

FONCTION : **Primitif**

APPEL : $Primitif(p, m)$;

ARGUMENTS : p un nombre premier, m un entier positif.

DESCRIPTION :

- $Primitif$ est une table qui contient des polynômes irreductibles primitifs, l'appel $Primitif(p, m)$; retourne un polynôme en a ou bien n'est pas évalué.
- si l'on désire faire des calculs dans une extension de degré m d'un corps fini premier $\mathbb{F}_p$, et que $Primitif(p, m)$ n'est pas défini, il faut affecter à $Primitif(p, m)$ la valeur d'un polynôme irréductible primitif $p(a)$: $Primitif(p, m) := p(a)$; (attention il est nécessaire d'utiliser l'indéterminée a)
- Si l'on désire ajouter un polynôme définitivement, on peut ajouter la ligne correspondante dans le fichier source *Primitif*.

EXEMPLES :

- $Primitif(2, 4)$;
$$a^4 + a + 1$$
- $Primitif(5, 2)$;
$$Primitif(5, 2)$$
- $Primitif(5, 2) := a^2 + a + 1$;
$$Primitif(5, 2) := a^2 + a + 1$$
- $Primitif(5, 2)$;
$$a^2 + a + 1$$
- $Primitif(2, 4) := a^4 + a^3 + 1$;
$$Primitif(2, 4) := a^4 + a^3 + 1$$
- $Primitif(2, 4)$;
$$a^4 + a^3 + 1$$

VOIR AUSSI : GF.

FONCTION : **rat**

APPEL : $rat(p)$;

ARGUMENTS : p est un polynôme ou une fraction rationnelle à coefficients dans le corps defini par GF.

DESCRIPTION :

- Transforme une fraction rationnelle quelconque (avec un nombre arbitraire d'indéterminées) en une fraction rationnelle à coefficients dans $\mathbb{F}_q$.
- Les éléments du corps sont représentés comme des polynômes en a de degré inférieur au degré de *tellrat*.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $rat(3a^4x^2 + a^7 + 1)$;

$$(a+1)x^2 + a^3 + a$$

VOIR AUSSI : rexp, algebraic.

FONCTION : **rexp**

APPEL : $rexp(p)$;

ARGUMENTS : p est un polynôme ou une fraction rationnelle à coefficients dans le corps defini par GF.

DESCRIPTION :

- Transforme une fraction rationnelle quelconque (avec un nombre arbitraire d'indéterminées) en une fraction rationnelle à coefficients dans $\mathbb{F}_q$.
- Les éléments du corps sont représentés comme des puissances de a.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $rexp(3a^4x^2 + a^7 + 1)$;

$$a^4x^2 + a^9$$

VOIR AUSSI : rat, algebraic.

FONCTION : **algebraic**

APPEL : *algebraic*();

ARGUMENTS : sans argument la fonction utilisée pour l'évaluation sera *rat*, si un argument est fourni (quelconque) alors cette fonction sera *rexp*.

DESCRIPTION :

- Cette fonction crée un nouvel environnement dans lequel la fonction *rexp* (ou *rat*) est appliquée au résultat de chaque évaluation.
- Le mot-clé "off" permet de quitter ce mode d'évaluation.

VOIR AUSSI : rexp, rat.

FONCTION : **Log**

APPEL : *Log*(x);

ARGUMENTS : x un élément du corps - i.e. un polynôme en a.

DESCRIPTION : Calcule le logarithme de x en base a.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $Log(1 + a)$;

$$4$$

- $Log(a^{17})$;

$$2$$

—

FONCTION : **inverse**

APPEL : *inverse*(x);

ARGUMENTS : x un élément quelconque du corps.

DESCRIPTION : calcule l'inverse de x.

FONCTION : **erem,equo**

APPEL : $erem(p_1, p_2)$; ou $erem(p_1, p_2, 'qu')$;
$equo(p_1, p_2)$; ou $equo(p_1, p_2, 're')$;

ARGUMENTS : p_1 et p_2 sont des polynômes à coefficients dans le corps $\mathbb{F}_q$, les arguments qu et re sont optionnels et doivent être des variables non-affectées.

DESCRIPTION :

- la fonction $erem$ retourne le reste de la division de p_1 par p_2, la fonction $equo$ retourne le quotient de la division de p_1 par p_2.
- si un troisième argument est fourni à $erem$ ou $equo$, les valeur du quotient et du reste seront assignées respectivement à qu et à re.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $equo(x^2 + ax + a^2 + 1, x + a^3 + a + 1)$;

$$x + a^3 + 1$$

- $erem(x^2 + ax + a^2 + 1, x + a^3 + a + 1, 'qu')$;

$$a^3 + 1$$

- qu;

$$x + a^3 + 1$$

VOIR AUSSI : egcd, efactor.

FONCTION : **egcd**

APPEL : $egcd(p_1, p_2)$;

ARGUMENTS : p_1 et p_2 sont des polynômes à coefficients dans le corps $\mathbb{F}_q$.

DESCRIPTION : *egcd* retourne le plus grand commun diviseur des polynômes p_1 et p_2 calculé dans le corps $\mathbb{F}_q$.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $egcd(x^2 + (a^2 + a + 1)x + a^3 + a^2, x + a^2)$;

$$x + a^2$$

VOIR AUSSI : erem, equo, efactor.

—

FONCTION : **efactor**

APPEL : $efactor(p)$;

ARGUMENTS : p est un polynôme à une indeterminée à coefficients dans le corps $\mathbb{F}_q$,

DESCRIPTION : $efactor$ factorise le polynôme p dans le corps $\mathbb{F}_q$ par l'algorithme de Berlekamp. **ATTENTION** cet algorithme n'est utilisable que pour des corps de petit cardinal.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $efactor(x^3 + (a^2 + a + 1)x^2 + (a^3 + a^2 + a)x + a^3)$

$$(x + 1)(x + a)(x + a^2)$$

VOIR AUSSI : erem, equo, egcd.

FONCTION : **pmin**

APPEL : $pmin(e, x)$;

ARGUMENTS : e un élément du corps $\mathbb{F}_q$, x une indeterminée.

DESCRIPTION : retourne le polynôme minimal de e dans le corps premier $\mathbb{F}_p$, l'indéterminée du polynôme sera x.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $pmin(a, x)$;

$$x^4 + x + 1$$

VOIR AUSSI : classecycl.

—

FONCTION : **classecycl**

APPEL : $classecycl(i)$;

ARGUMENTS : i un entier modulo $argGF - 1$.

DESCRIPTION : retourne l'ensemble des logarithmes des éléments de la classe cyclotomique de a^i.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $classecycl(3)$;

$$3, 6, 9, 12$$

VOIR AUSSI : pmin.

FONCTION : **RS**

APPEL : $RS(n,k,l)$; ou $RS(n,k,p)$; ou $RS(n,k,l,1)$; ou $RS(n,k,p,1)$;

ARGUMENTS : n, k la longueur et la dimension du code de Reed-Solomon, l une liste (de longueur k), p un polynôme de degré inférieur ou egal à $k-1$.

DESCRIPTION :

- la fonction RS code avec le code de Reed-Solomon de longueur n et de dimension k la liste ou le polynôme donné en troisième argument et retourne le résultat sous le même format.
- si un quatrième argument est fourni (optionnel et quelconque), les éléments du corps dans le résultat seront exprimés sous forme d'une puissance de l'élément primitif a.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $RS(15, 7, x + a)$;

$$x^9 + ax^8 + (a^2 + a)x^7 + (a^3 + a + 1)x^6 + ax^4 + (a^3 + a^2 + a + 1)x^3 + (a^3 + a^2 + 1)x^2 + (a^3 + a^2 + a + 1)x + a^2 + a + 1$$

- $RS(15, 7, [a, 1, 0, 0, 0, 0, 0], 1)$;

$$[a^{10}, a^{12}, a^{13}, a^{12}, a, 0, a^{13}, a^5, a, 1, 0, 0, 0, 0, 0]$$

VOIR AUSSI : genpRS, decRS.

FONCTION : **genpRS**

APPEL : genpRS(n,k);

ARGUMENTS : n entier inférieur strictement à $argGF$, k entier inférieur strictement à n.

DESCRIPTION : retourne un polynôme générateur d'un code de Reed-Solomon de longueur n et de dimension k.

VOIR AUSSI : RS, decRS.

—

FONCTION : **decRS**

APPEL : $decRS(n, k, p, z)$;

ARGUMENTS : n, k entiers, p un polynôme de $\mathbf{F}_q[z]$ de degré inférieur ou égal à $n - 1$.

DESCRIPTION : décode si possible le polynôme p comme un élément du code de Reed-Solomon de longueur n et de dimension k.

EXEMPLES :

- $GF(16)$;

$$2, a^4 + a + 1, 16$$

- $decRS(15, 11, x + 1, x)$;

$$0$$

VOIR AUSSI : RS, genpRS.

Annexe C

Mise en œuvre du décodage des codes produit

La mise en œuvre du décodage des codes produit est relativement aisée, en effet celle-ci s'avère économique tant au niveau de la taille des programmes que de la quantité de mémoire nécessaire.

On fera dans cette annexe deux hypothèses

1. les codes composants sont égaux, de paramètres (n, k, d),
2. on corrige les erreurs seules.

D'autre part nous ne décrirons ici que l'algorithme le plus performant que nous ayons utilisé pour nos simulations, que nous appellerons algorithme de Reddy-Robinson itératif.

1 Description de l'algorithme de Reddy-Robinson itératif

Soit le code produit $\mathcal{C} = C(n,k,d) \otimes C(n,k,d)$. On pose

$$t = \lfloor \frac{d}{2} \rfloor.$$

Soit ϕ un algorithme de décodage d'erreur et d'effacement de C.

1.1 Algorithme de Reddy-Robinson étendu

Soit une matrice $M \in \mathcal{M}_q(n,n)$. Soit $\mathbf{c}_i$, $i \in \{0, \ldots, n-1\}$ sa i-ième colonne. On définit

1. Les entiers w_i

$$\forall\, i,\ 0 \leq i \leq n-1,\ w_i = \begin{cases} d_H(\mathbf{c}_i, \phi(\mathbf{c}_i) & \text{si } d_H(\mathbf{c}_i, \phi(\mathbf{c}_i) \leq \frac{d}{2} \\ \frac{d}{2} & \text{si } d_H(\mathbf{c}_i, \phi(\mathbf{c}_i) > \frac{d}{2} \\ & \textit{ou} \\ & \text{si le décodage de } \mathbf{c}_i \text{ a échoué} \end{cases}$$

2. La fonction $W_{\mathbf{y}}(\mathbf{x})$

$$\forall\, \mathbf{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{F}_q^n,\ \forall\, \mathbf{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{F}_q^n,$$

$$W_{\mathbf{y}}(\mathbf{x}) = \sum_{i=0}^{n-1} W_i(x_i),$$

avec

$$\forall\, i,\ 0 \leq i \leq n-1,\ W_i(x_i) = \begin{cases} w_i & \text{si } x_i = y_i \\ d - w_i & \text{sinon} \end{cases}$$

3. Les décodeurs ϕ_j

$$\forall\, j \in \{0, 1, \ldots, t, \infty\},\ \phi_j(\mathbf{y}) = \phi(\mathbf{y}_j),$$

où $\mathbf{y}_j$ est le vecteur $\mathbf{y}$ dans lequel on a remplacé par un effacement toutes les positions i telles que $w_i \geq j$.

Algorithme $\mathcal{A}_6$ (Reddy-Robinson étendu) *Soit $M \in \mathcal{M}_q(n, n)$,*

Étape 1. *On décode les colonnes de M à l'aide de ϕ.*

On peut alors définir les w_i, les décodeurs ψ_j, et les fonctions $W_{\mathbf{y}}$

Étape 2. *On remplace les valeurs décodées dans M.*

Étape 3. *On effectue ensuite pour chaque ligne les $t+2$ décodages d'erreur et d'effacement $\phi_0, \ldots, \phi_t, \phi_\infty$.*

Étape 4. *Pour chaque ligne $\mathbf{y}$, on choisit parmi les ϕ_j qui aboutissent à un mot de C l'un de ceux réalisant le minimum de $W_{\mathbf{y}}(\phi_j(\mathbf{y}))$.*

Étape 5. *On remplace, s'il y a lieu, dans M les lignes ainsi obtenues et l'algorithme retourne M.*

Les étapes 3 à 5 de cet algorithme nous permettent de définir un algorithme que nous appellerons également, par abus de langage, algorithme de Reddy-Robinson étendu et que nous noterons ψ.

Algorithme $\mathcal{A}_7$ *Soient $M \in \mathcal{M}_q(n, n)$ et $w_0, \ldots, w_{n-1} \in \{0, \ldots, t, \frac{d}{2}\}$,*

Étape 0. *On définit les décodeurs ψ_j, et les fonctions $W_{\mathbf{y}}$ à partir des w_i.*

Étape 3. *On effectue ensuite pour chaque ligne les $t+2$ décodages d'erreur et d'effacement $\phi_0, \ldots, \phi_t, \phi_\infty$.*

Étape 4. *Pour chaque ligne $\mathbf{y}$, on choisit parmi les ϕ_j qui aboutissent à un mot de C l'un de ceux réalisant le minimum de $W_{\mathbf{y}}(\phi_j(\mathbf{y}))$.*

Étape 5. *On remplace, s'il y a lieu, dans M les lignes ainsi obtenues et l'algorithme retourne M.*

1.2 Algorithme de Reddy-Robinson itératif

Soit ψ l'algorithme défini par $\mathcal{A}_7$. L'algorithme de Reddy-Robinson itératif est défini comme suit :

Algorithme $\mathcal{A}_8$ *Soit $M \in \mathcal{M}_q(n,n)$,*

Étape 1. *On décode les colonnes de M à l'aide de ϕ.*

Étape 2. *on calcule les w_i et on remplace les valeurs décodées dans M.*

Étape 3. $M' := \psi(M, w_0, \ldots, w_{n-1})$.

Étape 4. *Si $M' \in \mathcal{C} \longrightarrow$ SUCCES.*

Étape 5. *Si le nombre d'itérations est superieur ou égal à MAX ou si $M = M' \longrightarrow$ ECHEC.*

Étape 6. *On affecte à w_i la distance de Hamming entre la i-ième ligne de M et la i-ième ligne de M'.*

Étape 7. $M := transpose(M')$.

Étape 8. $\longrightarrow$ **Étape 3.**

On notera que cet algorithme possède un paramètre : MAX, qui est le nombre maximal d'itérations effectuées.

Remarque C.1

1. *On peut déterminer a peu de frais une borne supérieure pratique du nombre d'itérations en cas de décodage correct du motif. Cette borne supérieure constituerait une bonne valeur pour le paramètre MAX.*

   En effet, on peut supposer, si le motif est corrigible, qu'à chaque itération l'une au moins des lignes va être correctement corrigée, et pour la première fois. C'est-à-dire qu'on aura alors au plus $2n$ itérations.

2. *Ce paramètre permet également de limiter la complexité du décodage, mais en diminuant les performances.*

2 Programme en C

```
#include <stdio.h>

#define CARD 16

static int somme[CARD][CARD] = {
    { 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15},
```

```
    { 1,  0,  5,  9, 15,  2, 11, 14, 10,  3,  8,  6, 13, 12,  7,  4},
    { 2,  5,  0,  6, 10,  1,  3, 12, 15, 11,  4,  9,  7, 14, 13,  8},
    { 3,  9,  6,  0,  7, 11,  2,  4, 13,  1, 12,  5, 10,  8, 15, 14},
    { 4, 15, 10,  7,  0,  8, 12,  3,  5, 14,  2, 13,  6, 11,  9,  1},
    { 5,  2,  1, 11,  8,  0,  9, 13,  4,  6, 15,  3, 14,  7, 12, 10},
    { 6, 11,  3,  2, 12,  9,  0, 10, 14,  5,  7,  1,  4, 15,  8, 13},
    { 7, 14, 12,  4,  3, 13, 10,  0, 11, 15,  6,  8,  2,  5,  1,  9},
    { 8, 10, 15, 13,  5,  4, 14, 11,  0, 12,  1,  7,  9,  3,  6,  2},
    { 9,  3, 11,  1, 14,  6,  5, 15, 12,  0, 13,  2,  8, 10,  4,  7},
    {10,  8,  4, 12,  2, 15,  7,  6,  1, 13,  0, 14,  3,  9, 11,  5},
    {11,  6,  9,  5, 13,  3,  1,  8,  7,  2, 14,  0, 15,  4, 10, 12},
    {12, 13,  7, 10,  6, 14,  4,  2,  9,  8,  3, 15,  0,  1,  5, 11},
    {13, 12, 14,  8, 11,  7, 15,  5,  3, 10,  9,  4,  1,  0,  2,  6},
    {14,  7, 13, 15,  9, 12,  8,  1,  6,  4, 11, 10,  5,  2,  0,  3},
    {15,  4,  8, 14,  1, 10, 13,  9,  2,  7,  5, 12, 11,  6,  3,  0}
};

static int produit[CARD][CARD] = {
    { 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
    { 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15},
    { 0,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,  1},
    { 0,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,  1,  2},
    { 0,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,  1,  2,  3},
    { 0,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,  1,  2,  3,  4},
    { 0,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,  1,  2,  3,  4,  5},
    { 0,  7,  8,  9, 10, 11, 12, 13, 14, 15,  1,  2,  3,  4,  5,  6},
    { 0,  8,  9, 10, 11, 12, 13, 14, 15,  1,  2,  3,  4,  5,  6,  7},
    { 0,  9, 10, 11, 12, 13, 14, 15,  1,  2,  3,  4,  5,  6,  7,  8},
    { 0, 10, 11, 12, 13, 14, 15,  1,  2,  3,  4,  5,  6,  7,  8,  9},
    { 0, 11, 12, 13, 14, 15,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10},
    { 0, 12, 13, 14, 15,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11},
    { 0, 13, 14, 15,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12},
    { 0, 14, 15,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13},
    { 0, 15,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14}
};

#define TRUE 1
#define FALSE 0
#define NOMOD 2
#define ERONE 3
#define LON 14
#define DIM 7
#define DMIN 8
#define B 0
```

```
#define NB_ITER 2*LON

main(argc,argv)
int argc;
char **argv;
{
    FILE *fichier;
    char nom_du_fichier[80];

    int i,j,poids_erreur,debut,fin,succes,essai,cpt,res;
    int mat[LON][LON],matorig[LON][LON],w[LON];
    int decoder();
    float resultat;
    void creermat(),copie(),transposer();

    debut = atoi(argv[1]);
    fin = atoi(argv[2]);
    cpt = atoi(argv[3]);

    strcpy(nom_du_fichier,"r");
    strcat(nom_du_fichier,argv[1]);
    strcat(nom_du_fichier,"-");
    strcat(nom_du_fichier,argv[2]);
    strcat(nom_du_fichier,"_");
    strcat(nom_du_fichier,argv[3]);
    fichier = fopen(nom_du_fichier,"w");

    fprintf(fichier,"Code produit sur GF(%d) de parametres (%d,%d,%d)\n"
    ,CARD,LON*LON,DIM*DIM,DMIN*DMIN);
    fprintf(fichier
    ,"Code composant = code BCH sur GF(%d) de parametres (%d,%d,%d)\n"
    ,CARD,LON,DIM,DMIN);
    fprintf(fichier,"\tzeros du code = ");
    for(i=B ; i<B+DMIN-1 ; ++i)
fprintf(fichier,"%-4.1d",i);
    fprintf(fichier,"\n\n");
    fflush(fichier);
    for(poids_erreur=debut ; poids_erreur<=fin ; ++poids_erreur) {
succes=0;
essai=0;
for(i=0 ; i<cpt ; ++i) {
    ++essai;
    creermat(mat,poids_erreur);
```

```
    copie(matorig,mat,LON*LON);
    for(j=0 ; j<LON ; ++j) {
w[j] = 2*dec_li(mat[j],0,0);
    }
    transposer(mat);
    if(egal(mat,0,LON*LON)==TRUE) {
++succes;
break;
    }
    for(j=0 ; j<NB_ITER ; ++j) {
res = decoder(mat,w);
if(res==TRUE) {
    ++succes;
    break;
}
else if(res==NOMOD) {
    break;
}
else if(res==ERRONE) {
    fprintf(fichier,"mauvais decodage\n");
    imprimer(mat,LON,fichier);
    imprimer(matorig,LON,fichier);
    break;
}
    }
}
resultat = (float) succes / essai;
fprintf(fichier,"nombre d'essais:\t%d\n",cpt);
fprintf(fichier,"poids de l'erreur:\t%d\n%f\n",poids_erreur,resultat);
fflush(fichier);
    }
}

void creermat(mat,poids)
int mat[][LON],poids;
{
    int m[LON*LON];
    void initialiser(),modifier(),copie();

    copie(mat,0,LON*LON);
    initialiser(m);
    modifier(mat,m,poids);
}
```

```
void modifier(mat,m,poids)
int mat[][LON];
int m[],poids;
{
    int i,j;

    for(i=LON*LON ; poids>0 ; --poids)
{
    j=random()%i--;
    mat[(m[j]/LON)][(m[j]%LON)]=(random()%(CARD-1))+1;
    m[j]=m[i];
}
}

void initialiser(m)
int m[];
{
    int i;

    for(i=0 ;  i<LON*LON ; ++i)
m[i]=i;
}

void transposer(mat)
int mat[][LON];
{
    int x;
    int i,j;

    for(i=1 ; i<LON ; ++i)
for(j=0 ; j<i ; ++j) {
    x=mat[i][j];
    mat[i][j]=mat[j][i];
    mat[j][i]=x;
}
}

int dec_li(mdc,eff,neff)
int mdc[],eff[],neff;
{
    int inv,i,nberreurs,poids;
    extern int nbdec;
    int li[LON],sigma[LON],omega[LON],zeros[CARD];
    int r[LON][LON],u[LON][LON],q[LON][LON],aux[LON][LON];
```

```
    int degre(),inverse(),evaluer(),prodaux(),eltcode(),calculsyndrome();
    void copie(),diviser(),psomme(),pproduit(),prodscal();

    if(neff>=DMIN)
return(-1);
    nbdec++;
    if(neff>0)
for(i=0 ; i<neff ; ++i)
    mdc[eff[i]] = 0;
    poids = dist_H(mdc,0,LON);
    if(2*poids+neff<DMIN) {
copie(mdc,0,LON);
return(poids);
    }
    nberreurs = 0;
    copie(r,0,LON*LON);
    if(calculsyndrome(r[1],mdc)!=0) {
if(neff>0) {
    copie(omega,0,LON);
    copie(sigma,r[1],LON);
    for(i=0 ; i<neff ; ++i) {
omega[1] = eff[i]+1;
pproduit(sigma,omega,r[1]);
r[1][DMIN-1] = 0;
copie(sigma,r[1],LON);
    }
}
copie(li,mdc,LON);
copie(q,0,LON*LON);
copie(u,0,LON*LON);
copie(aux,0,LON*LON);
copie(zeros,0,CARD);
r[0][DMIN-1]=1;
u[1][0]=1;
for(i=0 ; i<DMIN-1 ; ++i) {
    if(degre(r[i+1],LON)<((DMIN+neff)/2))
break;
    else {
copie(r[i+2],r[i],LON);
diviser(r[i],r[i+1],q[i],r[i+2]);
pproduit(q[i],u[i+1],aux[i]);
psomme(aux[i],u[i],u[i+2]);
    }
}
```

```
if(u[i+1][0]==0) {
    return(-1);
}
inv=inverse(u[i+1][0]);
copie(sigma,u[i+1],LON);
prodscal(sigma,inv);
copie(omega,r[i+1],LON);
prodscal(omega,inv);
for(i=0 ; i<neff ; ++i)
    zeros[eff[i]+1] = 1;
for(i=1 ; i<CARD ; ++i) {
    if(evaluer(sigma,i)==0) {
inv = inverse(i);
if((inv>LON) || (zeros[inv]!=0))
    return(-1);
zeros[inv] = 1;
++nberreurs;
    }
}
if(nberreurs<degre(sigma,LON))
    return(-1);
for(i=1 ; i<=LON ; ++i) {
    if(zeros[i]!=0)
li[i-1]=somme[li[i-1]][produit[puissance(inverse(i),B)]
       [produit[evaluer(omega,inverse(i))]
[inverse(prodaux(zeros,i))]]];
}
if(eltcode(li)==TRUE) {
    copie(mdc,li,LON);
    return(nberreurs);
}
else
    return(-1);
    }
    else
return(0);
}

int calculsyndrome(s,li)
int *s,*li;
{
    int i,mod;
    int evaluer();
```

```
    mod=0;
    for(i=0 ; i<DMIN-1 ; ++i) {
s[i]=evaluer(li,i+B+1);
if(s[i]!=0)
    mod=1;
    }
    return(mod);
}

int prodaux(t,x)
int *t,x;
{
    int i,r;
    int inverse();

    r=1;
    for(i=1 ; i<=LON ; ++i)
if(t[i]==1&&i!=x)
    r=produit[r][somme[1][produit[inverse(x)][i]]];
    return(r);
}

int eltcode(mdc)
int *mdc;
{
    int i;

    for(i=1 ; i<=DMIN-1 ; ++i)
if(evaluer(mdc,i+B)!=0)
    return(FALSE);
    return(TRUE);
}

int decoder(mat,w)
int mat[][LON],w[LON];
{

    int r,i,j,p,decode,modif,meilleure_branche,sw,meilleur_score,score;
    int ww[LON];
    int res[(DMIN-1)/2+2][LON][LON];
    int eff[(DMIN-1)/2+2][LON],neff[(DMIN-1)/2+2];
    int dec_li();
    void transposer(),copie(),imprimer();
```

```
    sw = 0;
    for(i=0 ; i<LON ; ++i) {
if(w[i]<0)
    w[i] = DMIN;
sw += w[i];
    }
    decode = TRUE;
    modif = FALSE;
    for(r=0 ; r<(DMIN-1)/2+1 ; ++r) {
copie(res[r],mat,LON*LON);
for(i=0,neff[r]=0 ; i<LON ; ++i) {
    if(w[i]>2*r)
eff[r][neff[r]++] = i;
    if(neff[r]>=DMIN)
break;
}
    }
    copie(res[(DMIN-1)/2+1],mat,LON*LON);
    neff[(DMIN-1)/2+1] = 0;
    for(i=0 ; i<LON ; ++i) {
meilleur_score = LON*LON;
meilleure_branche = -1;
for(r=(DMIN-1)/2+1 ; r>=0 ; --r) {
    p = dec_li(res[r][i],eff[r],neff[r]);
    if(p==0) {
meilleure_branche = r;
break;
    }
    else if(p>0) {
score = sw;
for(j=0 ; j<LON ; ++j) {
    if(res[r][i][j]!=mat[i][j]) {
score += 2*(DMIN-w[j]);
    }
}
if(score<DMIN*DMIN) {
    meilleure_branche = r;
    break;
}
else if(score==sw) {
    meilleure_branche = r;
    break;
}
else if(score<meilleur_score) {
```

```
    meilleure_branche = r;
    meilleur_score = score;
}
    }
}
if(meilleure_branche>=0) {
    ww[i] = 0;
    for(j=0 ; j<LON ; ++j)
if(mat[i][j]!=res[meilleure_branche][i][j])
    ww[i] += 2;
    if(ww[i]>0) {
modif = TRUE;
if(ww[i]>DMIN)
    ww[i] = DMIN;
copie(mat[i],res[meilleure_branche][i],LON);
    }
}
else {
    ww[i] = DMIN;
    decode = FALSE;
}
    }
    for(i=0 ; i<LON ; ++i) {
w[i] = ww[i];
    }
    transposer(mat);
    if(modif==FALSE)
return(NOMOD);
    if((decode==TRUE) && (egal(mat,0,LON*LON)==FALSE)) {
for(i=0 ; i<LON ; ++i) {
    if(eltcode(mat[i])==FALSE)
return(FALSE);
}
return(ERRONE);
    }
    return(decode);
}

void diviser(p,d,q,r)
int *p,*d,*q,*r; /*initialement p=r et q=0 */
{
    int degd,degr,inv,i,j;
    int degre(),inverse(),imp();
```

```
    degd=degre(d,LON);
    degr=degre(r,LON);
    if(degd>=0) {
inv=inverse(d[degd]);
for(i=degr ; i>=degd ; --i) {
    if(r[i]==0)
;
    else {
q[i-degd]=produit[r[i]][inv];
r[i]=0;
for(j=i-1 ; j>=i-degd ; --j)
    r[j]=somme[r[j]][produit[q[i-degd]][d[degd-i+j]]];
    }
}
    }
    else
printf("erreur : division par 0\n");
}

void psomme(p,q,s)
int *p,*q,*s;
{
    int i,deg;
    int degre(),max();

    deg=max(degre(p,LON),degre(q,LON));
    for(i=0 ; i<=deg ; ++i)
s[i]=somme[p[i]][q[i]];
}

void pproduit(p,q,s)
int *p,*q,*s;
{
    int i,j,degp,degq;
    int degre();

    degp=degre(p,LON);
    degq=degre(q,LON);
    for(i=0 ; i<=degp ; ++i)
for(j=0 ; j<=degq ; ++j)
    s[i+j]=somme[s[i+j]][produit[p[i]][q[j]]];
}
```

```
void prodscal(p,a)
int *p,a;
{
    int i,deg;
    int degre();

    deg=degre(p,LON);
    for(i=0 ; i<=deg ; ++i)
p[i]=produit[p[i]][a];
}

int max(x,y)
int x,y;
{
    if(x>y)
return(x);
    return(y);
}


int inverse(x)
int x;
{
    if(x==1)
return(1);
    else
return(CARD+1-x);
}

int evaluer(p,x)
int *p,x;
{
    int i,y,deg;
    int puissance(),degre();

    y=0;
    deg=degre(p,LON);
    for(i=0 ; i<=deg ; ++i)
y=somme[y][produit[p[i]][puissance(x,i)]];
    return(y);
}

int puissance(x,i)
int x,i;
```

```
{
    if(x==0||x==1)
return(x);
    return(((x-1)*i)%(CARD-1)+1);
}

void copie(x,y,taille)
int *x,*y,taille;
{
    if(y==0)
for(; taille>0 ; --taille,++x)
    *x = 0;
    else
for(; taille>0 ; --taille,++x,++y)
    *x = *y;
}

int degre(p,d)
int *p,d;
{

    for(;d>0;--d)
if(p[d-1]!=0)
    break;
    return(d-1);
}

int egal(x,y,taille)
int *x,*y,taille;
{
    if(y==0) {
for(; taille>0 ; --taille,++x)
    if(*x != 0)
return(FALSE);
return(TRUE);
    }
    else {
for(; taille>0 ; --taille,++x,++y)
    if(*x != *y)
return(FALSE);
return(TRUE);
    }
}
```

```
int dist_H(x,y,taille)
int *x,*y,taille;
{
    int d;

    d = 0;
    if(y==0) {
for(; taille>0 ; --taille,++x)
    if(*x != 0)
++d;
return(d);
    }
    else {
for(; taille>0 ; --taille,++x,++y)
    if(*x != *y)
++d;
return(d);
    }
}

int imprimer(mat,taille,stream)
int *mat,taille;
FILE *stream;
{
    int i;

    for(i=0 ; i<taille ; ++i,mat+=taille)
imp_li(mat,taille,stream);
    fprintf(stream,"\n");
}

int imp_li(m,taille,stream)
int *m,taille;
FILE *stream;
{
    for(; taille>0 ; --taille,++m)
fprintf(stream,"%-4.1d",*m);
    fprintf(stream,"\n");
}
```

Bibliographie

[1] Abramson (N.). – Encoding and decoding cyclic code groups. *Problems of Information Transmission*, vol. 6, n° 2, 1970, pp. 148–154.

[2] Assmus (E.F.) et Key (J.D.). – Affine and projective planes. *Discrete Mathematics*, vol. 83, 1990, pp. 161–187.

[3] Augot (D.). – Exemple d'utilisation du constructeur FiniteFieldExtension de Scratchpad. – Rapport de DEA LAP (LITP Paris VI), octobre 1989.

[4] Augot (D.), Charpin (P.) et Sendrier (N.). – The minimum distance of some binary codes via the newton's identities. *In : EUROCODE'90*, éd. par Charpin (P.) et Cohen (G.). pp. 65–73. – Springer-Verlag, 1990.

[5] Augot (D.), Charpin (P.) et Sendrier (N.). – *Studying the Locator Polynomials of Minimum Weight Codewords of BCH Codes.* – Rapport de Recherche n° 1488, INRIA, juillet 1991. A paraître dans IEEE Transaction on Information Theory.

[6] Augot (D.), Charpin (P.) et Sendrier (N.). – Sur une classe de polynômes scindés de l'algèbre $\mathbb{F}_{2^m}[Z]$. *Compte Rendu de l'Academie des Sciences de Paris*, t. 312, Série I, 1991, pp. 649–651.

[7] Bahl (L.R.) et Chien (R.T.). – Single - and multiple - burst-correcting properties of a class of cyclic product codes. *IEEE Transaction on Information Theory*, vol. 17, 1971, pp. 594–600.

[8] Berlekamp (E.R.). – The weight enumerators for certain subcodes of the second order binary Reed-Muller codes. *Information and Control*, vol. 17, 1970, pp. 485–500.

[9] Berlekamp (E.R.). – *Algebraic Coding Theory.* – Aegen Park Press, 1984.

[10] Berlekamp (E.R.) et Justesen (J.). – Some long cyclic linear binary codes are not so bad. *IEEE Transaction on Information Theory*, vol. 20, 1974, pp. 351–356.

[11] Blahut (R.E.). – *Theory and Practice of Error Control Codes.* – Addison-Wesley, 1984.

[12] Block (E.L.) et Zyablov (V.V.). – Coding of generalized concatenated codes. *Problemy Peredachi Informatsii*, vol. 10, n° 3, 1974, pp. 45–50.

[13] Bose (R.C.) et Bush (R.C.). – Orthogonal arrays of strength two and three. *Ann. Math. Stat.*, vol. 23, 1952, pp. 508–524.

[14] Bossert (M.). – *Concatenation of Block Codes.* – Rapport technique, DFG, 1988.

[15] Burton (H.O.) et Weldon (E.J.). – Cyclic product codes. *IEEE Transaction on Information Theory*, vol. 11, 1965, pp. 433–439.

[16] Calabro (D.) et Wolf (J.K.). – On the synthesis of two dimesnional arrays with desirable correlation properties. *Information and Control*, 1968.

[17] Calingaert (P.). – Two-dimensional parity checking. *J. ACM*, vol. 8, 1961, pp. 186–200.

[18] Camion (P.). – A proof of some properties of reed-muller codes by means of the normal basis theorem. *In : Proceedings of the Conference on Combinatorial Mathematics and Its Applications*, éd. par Bose (R.C.) et Dowling (T.A.). – Chapel Hill, N.C., avril 1967.

[19] Camion (P.). – *An Iterative Euclidean Algorithm.* – Rapport de Recherche n° 844, INRIA, mai 1988.

[20] Camion (P.). – Majority decoding of repetition codes for the r-ary symmetric channel. *Sankhya : The Indian Journal of statistics*, vol. 54,A, 1991.

[21] Camion (P.) et Politano (J.L.). – Evaluation of a coding design for a very noisy chanel. *In : Coding Theory and Applications*, éd. par Cohen (G.) et Wolfmann (J.). – Springer Verlag, LNCS n° 388, 1988.

[22] Cerveira (A.G.). – On a class of wide-sense binary BCH codes whose minimum distances exceed the BCH bound. *IEEE Transaction on Information Theory*, septembre 1968, pp. 784–785.

[23] Charpin (P.). – Codes cycliques étendus affines-invariants et antichaînes d'un ensemble partiellement ordonné. *Discrete Mathematics*, vol. 80, 1990, pp. 229–247.

[24] Charpin (P.). – On a class of primitive BCH-codes. *IEEE Transaction on Information Theory*, vol. 36, n° 1, janvier 1990, pp. 222–228.

[25] Chien (R.T.) et Ng (S.W.). – Dual product codes for correction of multiple low-density burst errors. *IEEE Transaction on Information Theory*, vol. 19, 1973, pp. 672–677.

[26] Cohen (G.). – *Distance Minimale et Enumération des Poids des Codes Linéaires.* – Thèse de 3e cycle, ENST, avril 1976.

[27] Cohen (G.). – On the minimum distance of some BCH codes. *IEEE Transaction on Information Theory*, vol. 26, 1980, p. 363.

[28] Delsarte (P.). – Four fondamental parameters of a code and their combinatorial significance. *Information and Control*, vol. 23, 1973, pp. 407–438.

[29] Dornstetter (J-L.). – Quelques resultats sur les codes BCH binaires en longueur 255. – ENST rapport de stage, annexe, juillet 1982.

[30] Dornstetter (J.L.). – On the equivalence between berlekamp's and euclid's algorithms. *IEEE Transaction on Information Theory*, vol. 33, n° 3, mai 1987.

[31] Duc (N.Q.). – On the lin-weldon majority-logic decoding algorithm for product codes. *IEEE Transaction on Information Theory*, vol. 19, 1973, pp. 581–583.

[32] Duc (N.Q.) et Skattebol (L.V.). – Further results on majority-logic decoding of product codes. *IEEE Transaction on Information Theory*, vol. 18, 1972, pp. 308–310.

[33] Elias (P.). – Error-free coding. *IEEE Transaction on Information Theory*, vol. 4, 1954, pp. 29–37.

[34] Ericson (T.). – Concatenated codes – principles and possibilities. *In : AAECC-4*. – 1986.

[35] Ericson (T.). – *A Simple Analysis of the Block-Zyablov Decoding Algorithm*. – Rapport technique n° LiTH-ISY-I-0819, Departement of Electronical Engineering, Linköping University, novembre 1986.

[36] Forney (G.D. Jr.). – *Concatenated Codes*. – Cambridge, Massachussets, The MIT Press, 1966.

[37] Forney (G.D. Jr.). – Generalized minimum distance decoding. *IEEE Transaction on Information Theory*, vol. 12, 1966, pp. 125–131.

[38] Goethals (J.M.). – Factorization of cyclic codes. *IEEE Transaction on Information Theory*, vol. 13, 1967, pp. 242–246.

[39] Goethals (J.M.). – A polynomial approach to linear codes. *Information and Control*, 1967.

[40] Goldberg (M.). – Augmentation techniques for a class of product codes. *IEEE Transaction on Information Theory*, vol. 19, 1973, pp. 666–672.

[41] Gordon (B.). – On the existence of perfect maps. *IEEE Transaction on Information Theory*, vol. 13, 1967, pp. 242–246.

[42] Gore (W.C.). – Further results on product codes. *IEEE Transaction on Information Theory*, vol. 16, 1970, pp. 446–451.

[43] Hartmann (C.R.) et Tzeng (K.K.). – On some class of cyclic codes of composite length. *IEEE Transaction on Information Theory*, novembre 1973, pp. 820–823.

[44] Hartmann (C.R.P.) et Tzeng (K.K.). – A bound for cyclic codes of composite length. *IEEE Transaction on Information Theory*, vol. 18, 1972, pp. 307–308.

[45] Hartmann (C.R.P.), Tzeng (K.K.) et Chien (R.T.). – Some results on the minimum distance structure of cyclic codes. *IEEE Transaction on Information Theory*, vol. 18, 1972, pp. 402–409.

[46] Helgert (H.J.) et Stinaff (R.D.). – Minimum-distance bound for binary linear codes. *IEEE Transaction on Information Theory*, vol. 19, n° 3, mai 1973, pp. 344–356.

[47] Helgert (H.J.) et Stinaff (R.D.). – Shortened BCH codes. *IEEE Transaction on Information Theory*, novembre 1973, pp. 818–820.

[48] Ikai (T.) et Kojima (Y.). – Two-dimensional cyclic codes. *Electronics and Communication in Japan*, vol. 57A, n° 4, 1974, pp. 27–35.

[49] Imai (H.). – Two-dimensional fire codes. *IEEE Transaction on Information Theory*, vol. 19, 1973, pp. 796–806.

[50] Jensen (J.M.). – The concatenated structure of cyclic and abelian codes. *IEEE Transaction on Information Theory*, vol. 31, n° 6, novembre 1985, pp. 783–793.

[51] Karpovsky (M.G.). – On the weight distribution of binary linear codes. *IEEE Transaction on Information Theory*, vol. 25, n° 1, janvier 1979, pp. 105–108.

[52] Kasami (T.). – Some lower bounds on the minimum weight of cyclic codes of composite length. *IEEE Transaction on Information Theory*, vol. 14, n° 6, novembre 1968, pp. 814–818.

[53] Kasami (T.). – The weight enumerators for several classes of subcodes of the 2nd order binary Reed-Muller codes. *Information and Control*, vol. 18, 1971, pp. 369–394.

[54] Kasami (T.). – Constructing and decomposition of cyclic codes of composite length. *IEEE Transaction on Information Theory*, vol. 20, 1974, pp. 680–683.

[55] Kasami (T.) et Lin (S.). – The construction of a class of majority-logic decodable codes. *IEEE Transaction on Information Theory*, vol. 17, 1971, pp. 600–610.

[56] Kasami (T.) et Lin (S.). – Some results on the minimum weight of primitive BCH codes. *IEEE Transaction on Information Theory*, novembre 1972, pp. 824–825.

[57] Kasami (T.), Lin (S.) et Peterson (W.W.). – Some results on cyclic codes which are invariant under the affine group and their applications. *Information and Control*, vol. 11, 1967, pp. 475–496.

[58] Kasami (T.), Lin (S.) et Peterson (W.W.). – New generalisations of the Reed-Muller codes – Part I: Primitive codes. *IEEE Transaction on Information Theory*, vol. 14, n° 2, mars 1968, pp. 189–199.

[59] Kasami (T.) et Tokura (N.). – Some remarks on BCH bounds and minimum weights of binary primitive BCH codes. *IEEE Transaction on Information Theory*, vol. 15, n° 3, mai 1969, pp. 408–413.

[60] Kuroda (H.). – Note on error-control systems using product codes. *Electronics and Communication in Japan*, vol. 57A, n° 4, 1974, pp. 22–28.

[61] Lee (Y.L.) et Cheng (M.C.). – Cyclic mappings of product codes. *IEEE Transaction on Information Theory*, vol. 21, 1975, pp. 233–235.

[62] Leon (J.S.). – A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transaction on Information Theory*, vol. 34, n° 5, septembre 1988, pp. 1354–1359.

[63] Lidl (R.) et Niederreiter (H.). – *Finite Fields*. – Cambridge University Press, 1983.

[64] Lin (S.) et Weldon (E.J.). – Further results on cyclcic product codes. *IEEE Transaction on Information Theory*, vol. 16, 1970, pp. 452–495.

[65] Lum (V.) et Chien (R.T.). – On the minimum distance of Bose-Chaudhuri-Hocquenghem codes. *SIAM Journal on Applied Mathematics*, vol. 16, n° 6, novembre 1968, pp. 1325–1337.

[66] MacWilliams (F.J.) et Sloane (N.J.). – Pseudo random sequences and arrays. *Proceedings IEEE*, vol. 64, 1976, pp. 1715–1729.

[67] MacWilliams (F.J.) et Sloane (N.J.A.). – *The Theory of Error Correcting Codes*. – North-Holland, 1986.

[68] McEliece (R.J.). – Weight congruencs for p-ary cyclic codes. *Discrete Mathematics*, vol. 3, 1972, pp. 177–192.

[69] Nakamura (K.) et Iwadare (Y.). – Data scramblers for multilevel pulse sequences. *Nec Research and Development*, vol. 26, juillet 1972, pp. 53–63.

[70] Nomura (T.) et Fukuda (A.). – Linear recurring planes and two-dimensional cyclic codes. *Electronics and Communication in Japan*, vol. 54A, n° 3, 1974, pp. 22–28.

[71] Nomura (T.), Miyakawa (H.), Imai (H.) et Fukuda (A.). – A method of construction and some properties of planes having maximum area matrix. *Electronics and Communication in Japan*, vol. 54A, n° 4, 1971, pp. 18–25.

[72] Nomura (T.), Miyakawa (H.), Imai (H.) et Fukuda (A.). – Some properties of the gb-plane and its extension to three-dimensional space. *Electronics and Communication in Japan*, vol. 54A, n° 8, 1971, pp. 27–34.

[73] Nomura (T.), Miyakawa (H.), Imai (H.) et Fukuda (A.). – A theory of two-dimensional linear recurring arrays. *IEEE Transaction on Information Theory*, vol. 18, 1972, pp. 775–785.

[74] Patel (A.M.) et Hong (S.J.). – Optimal rectangular code for high density magnetic tapes. *IBM J. Res. Devel.*, vol. 18, 1974, pp. 579–588.

[75] Peterson (W. W.). – *Error-Correcting Codes.* – MIT Press, 1961.

[76] Rao (V.V.) et Reddy (S.M.). – A (48,31,8) linear code. *IEEE Transaction on Information Theory*, vol. 19, 1973, pp. 709–711.

[77] Reddy (S.M.). – On decoding iterated codes. *IEEE Transaction on Information Theory*, vol. 16, 1970, pp. 624–627.

[78] Reddy (S.M.). – Further results on decoders for q-ary output channels. *IEEE Transaction on Information Theory*, vol. 20, 1974, pp. 552–554.

[79] Reddy (S.M.) et Robinson (J.P.). – Random error and burst correction by iterated codes. *IEEE Transaction on Information Theory*, vol. 18, n° 1, janvier 1972, pp. 182–185.

[80] Reed (I.S.) et Stewart (R.M.). – Note on the existence of perfect maps. *IEEE Transaction on Information Theory*, vol. 8, 1962, pp. 10–12.

[81] Remijn (J.C.C.M.) et Tiersma (H.J.). – A duality theorem for the weight distribution of some cyclic codes. *IEEE Transaction on Information Theory*, vol. 34, n° 5, 1988, pp. 1348–1351.

[82] Roth (R.M.) et Seroussi (G.). – Encoding and decoding of BCH codes using light and short codewords. *IEEE Transaction on Information Theory*, vol. 34, n° 3, mai 1988, pp. 593–596.

[83] Schroeder (M.R.). – Sound diffusion by maximum-length sequences. *J. Acoustical Soc. Amer.*, vol. 57, 1975, pp. 149–150.

[84] Sendrier (N.). – *Product of Linear Codes.* – Rapport de Recherche n° 1286, INRIA, octobre 1990.

[85] Sloane (N.J.A.). – A simple description of an error-correcting code for high density magnetic tape. *Bell Syst. Tech. J.*, vol. 55, 1976, pp. 157–165.

[86] Sloane (N.J.A.), Fine (T.) et Phillips (P.G.). – New method for grating spectrometers. *Optical Spectra*, vol. 4, 1970, pp. 50–53.

[87] Sloane (N.J.A.), Fine (T.), Philllips (P.G.) et Harwit (M.). – Codes for multislit spectrometry. *Optical Spectra*, vol. 8, 1969, pp. 2103–2106.

[88] Sloane (N.J.A.) et Harwit (M.). – Masks for hadamard transform optics and weighing designs. *Applied Optics*, vol. 15, 1976, pp. 107–114.

[89] Spann (R.). – A two-dimensional correlation property of pseudorandom maximal-length sequences. *Proceedings IEEE*, vol. 53, 1965, p. 2137.

[90] Stenbit (J.P.). – Table of generators for Bose-Chaudhuri codes. *IEEE Transaction on Information Theory*, octobre 1964, pp. 390–391.

[91] Tang (D.T.) et Chien (R.T.). – Cyclic product codes and their implementation. *Information and Control*, vol. 9, 1966, pp. 196–209.

[92] Tolhuizen (L.M.) et Baggen (C.P.). – On the correcting capabilities of product codes. *IEEE Conference*, San-Diego 1989.

[93] Trager (B.M.). – Algebraic factoring and rational function integration. *In : 1976 ACM Symposium on Symbolic and Algebraic Computation*, éd. par Jenks (R.D.), pp. 219–226. – 1976.

[94] van Lint (J.H.). – *Coding Theory*. – Springer-Verlag, 1971.

[95] van Lint (J.H.) et Wilson (R.M.). – Binary cyclic codes generated by $m_1 m_7$. *IEEE Transaction on Information Theory*, vol. 32, n° 2, mars 1986, p. 283.

[96] van Lint (J.H.) et Wilson (R.M.). – On the minimum distance of cyclic codes. *IEEE Transaction on Information Theory*, vol. 32, n° 1, janvier 1986, pp. 23–40.

[97] Wainberg (S.). – Error-erasure decoding of product codes. *IEEE Transaction on Information Theory*, vol. 18, 1972, pp. 821–823.

[98] Wei (V.K.). – Generalized hamming weights for linear codes. to appear in *IEEE Transaction on Information Theory*, 1991.

[99] Weldon (E.J.). – Decoding binary block code on q-ary output channels. *IEEE Transaction on Information Theory*, vol. 17, 1971, pp. 713–718.

[100] Weng (L.J.). – Decomposition of m-sequences and its applications. *IEEE Transaction on Information Theory*, vol. 17, 1971, pp. 457–463.

[101] Weng (L.J.) et Sollman (G.H.). – Variable redundancy product codes. *IEEE Transaction on Communication*, vol. 15, 1967, pp. 835–838.

[102] Wolf (J.K.) et Elspas (B.). – Error-locating codes — a new concept in error control. *IEEE Transaction on Information Theory*, vol. 19, 1963, pp. 113–117.

[103] Zinoviev (V.). – Generalized concatenated codes. *Problemy Peredachi Informatsii*, vol. 12, 1976, pp. 5–15.

[104] Zinoviev (V.). – Generalized concatenated codes for channels with error burst and independant errors. *Problemy Peredachi Informatsii*, vol. 17, n° 4, décembre 1981, pp. 53–56.

Index