

**Projet de Recherche (PRe)**

**Spécialité : STIC LECS**

**Année scolaire : 2021-2022**

**Codes de Reed-Solomon**

**Equivalences et décodage souple**

**Non Confidentiel**

**Auteur : Joris Gallene Promotion : 2023**

**Tuteur ENSTA Paris : Benoit Geller Tuteur organisme d’accueil : Tarak Arbi**

**Stage effectué du 16 / 05 / 2022 au 05 / 08 / 2022**

**Nom de l’organisme d’accueil : ENSTA Paris**

**Adresse : 828 boulevard des Maréchaux 91120 Palaiseau**

# Note de non-confidentialité

Ce document est non confidentiel et peur donc être consulté librement par tous.

# Remerciements

Merci à Benoit Geller pour son encadrement et ses conseils.

# Résumé

Ce rapport présente les résultats d’un stage de trois mois à ENSTA Paris. Ce stage part des travaux effectués par Nicolas Drouin lors de son stage l’année précédente et les prolonge en présentant des preuves à certains éléments présentés puis en s’intéressant au décodage des codes de Reed-Solomon.

La première partie de ce stage a été consacrée à l’étude des codes de Reed-Solomon et à leur généralisation. Tout d’abord le but était de trouver une condition pour qu’un code de Reed-Solomon généralisé soit un code de Reed-Solomon classique puis d’étudier les équivalences entre différents codes. Des pistes de réflexion sur l’équivalence des codes ont été proposées mais ces travaux n’ont pas pu être continués car trop compliqués pour un stage de 3 mois.

La deuxième partie de ce stage a été consacré à l’étude d’un algorithme de décodage des codes de Reed-Solomon généralisés. Cet algorithme a été présenté par Arnaud Dagnelies dans sa thèse *Algebraic soft-decoding of Reed-Solomon Codes*. Cette partie s’articule en deux axes. Tout d’abord ce rapport présente cet algorithme dans le but de le comprendre en apportant des exemples. Ensuite, une étude des performances de l’algorithme sur un canal avec un bruit gaussien est proposée.

Table des matières

[Note de non-confidentialité 1](#_Toc111988826)

[Remerciements 2](#_Toc111988827)

[Résumé 3](#_Toc111988828)

[Introduction 7](#_Toc111988829)

[Chapitre 1. – Equivalences et égalité entre codes de Reed Solomon et codes de Reed Solomon généralisés 8](#_Toc111988830)

[I. Généralités sur les codes de Reed Solomon 8](#_Toc111988831)

[II. Génération des codes 8](#_Toc111988832)

[1. Corps de Galois 8](#_Toc111988833)

[2. Codes de Reed Solomon 10](#_Toc111988834)

[3. Codes de Reed-Solomon généralisés 10](#_Toc111988835)

[4. Exemples 11](#_Toc111988836)

[5. Notes théoriques sur les RS et GRS 11](#_Toc111988837)

[III. Comparaisons pratiques des codes RS et GRS 12](#_Toc111988838)

[1. Condition pour la cyclicité des codes GRS 12](#_Toc111988839)

[2. Répartitions des poids des mots de codes [5,7] 13](#_Toc111988840)

[3. Classes d’équivalences des codes [5;7] [5;15] et [7;15] 15](#_Toc111988841)

[4. Comparaison des matrices des codes 15](#_Toc111988842)

[5. Comparaison des polynômes générateurs 17](#_Toc111988843)

[IV. Egalité entre RS et GRS 18](#_Toc111988844)

[1. Condition d’égalité entre codes GRS et RS 18](#_Toc111988845)

[Chapitre 2. Décodage des codes GRS 20](#_Toc111988846)

[I. Introduction 20](#_Toc111988847)

[1. Principe de l’algorithme 20](#_Toc111988848)

[2. Degré pondéré et énumération monomiale 20](#_Toc111988849)

[3. Zéros de multiplicité m 20](#_Toc111988850)

[4. Matrice de multiplicité 21](#_Toc111988851)

[II. Construction de la matrice de multiplicité 21](#_Toc111988852)

[1. Matrice de fiabilité 21](#_Toc111988853)

[2. Exemple pour un canal dur 22](#_Toc111988854)

[3. Remarques et approximations 22](#_Toc111988855)

[4. Algorithme glouton 23](#_Toc111988856)

[5. Exemple 23](#_Toc111988857)

[6. Algorithme proportionnel 25](#_Toc111988858)

[III. Interpolation de Kötter 25](#_Toc111988859)

[1. Contraintes sur Q 25](#_Toc111988860)

[2. Relation d’ordre sur 26](#_Toc111988861)

[3. Algorithme 26](#_Toc111988862)

[4. Preuve de l’algorithme 27](#_Toc111988863)

[IV. Factorisation 28](#_Toc111988864)

[1. Introduction 28](#_Toc111988865)

[2. Algorithme de Roth-Ruckenstein 29](#_Toc111988866)

[3. Exemple 29](#_Toc111988867)

[V. Performance de l’algorithme. 30](#_Toc111988868)

[1. Comparaison avec le décodage classique 31](#_Toc111988869)

[Chapitre 3. Exemple d’encodage et décodage sur un petit code 32](#_Toc111988870)

[I. Description du corps 32](#_Toc111988871)

[II. Encodage 32](#_Toc111988872)

[III. Décodage 34](#_Toc111988873)

[IV. Décodage 35](#_Toc111988874)

[Annexes 36](#_Toc111988875)

[Annexe A : Preuve du Lemme IV.1.1 36](#_Toc111988876)

[Annexe B : CryptoExplorer 38](#_Toc111988877)

# Introduction

Ce stage a duré trois mois et s’est articulé en deux parties d’étude des codes de Reed-Solomon.

La première partie était consacrée à l’étude de la généralisation des codes de Reed-Solomon. Cette étude a nécessité un travail préliminaire important pour pouvoir implémenter les opérations dans un corps de Galois en langage C. Ensuite un travail bibliographique a été nécessaire afin de comprendre la généralisation des codes de Reed-Solomon. Après ces travaux préliminaires, il a été possible d’implémenter en C les codes de Reed-Solomon classiques et généralisés. Finalement, j’ai pu, en m’appuyant sur la bibliographie, comparer les codes de Reed-Solomon classiques et généralisés. Dans cette première partie, des résultats pratiques sont présentés puis suivis d’explications théoriques extraites de la bibliographie. Les conditions sur un code généralisé pour qu’il soit égal à un code classique sont données dans cette partie. Des résultats sur l’équivalence des codes de Reed-Solomon sur le poids en bits sont présentés mais l’étude théorique n’a pas été poursuivi sur les conseils de M. Geller car il n’y a pas de bibliographie existante et le sujet ne pouvait pas être étudié en seulement 3 mois.

La deuxième partie du stage a été dédiée à l’étude d’un algorithme de décodage des codes de Reed-Solomon généralisés. La première étape a été d’implémenter l’algorithme en reprenant certains codes déjà développés dans la première partie. Cette implémentation a permis de comprendre le fonctionnement de l’algorithme qui est présenté dans ce rapport avec des preuves et des exemples pour aider la compréhension. Enfin une étude des performances de correction dans le cadre d’un décodage souple avec un bruit gaussien est présentée.

Le troisième chapitre de ce rapport est constituée d’exemples d’encodage et de décodage dans le but de résumer le contenu de ce rapport de manière pratique.

# – Equivalences et égalité entre codes de Reed Solomon et codes de Reed Solomon généralisés

## Généralités sur les codes de Reed Solomon

Les codes correcteurs sont une manière d’encoder une information avant de la transmettre. Le but de ces codes est de rendre possible la détection et la correction d’erreurs de transmission. On introduit de la redondance en transmettant des bits d’informations supplémentaires qui permettent de corriger plus ou moins d’erreur ; plus on veut pouvoir corriger d’erreur, plus il faut introduire de redondance et donc de bits supplémentaires.

Les codes de Reed Solomon sont une catégorie particulière de codes correcteurs adaptés à la correction de paquets d’erreurs. En effet dans un canal, les erreurs arrivent souvent par paquet sur plusieurs bits consécutifs à cause d’une perturbation. Les codes de Reed Solomon permettent de traiter ce problème mais ne permettraient pas de corriger le même nombre d’erreur réparties aléatoirement. Cependant, en pratique, les erreurs arrivent toujours par paquet.

Les codes de Reed Solomon classiques sont générés par division polynomiale dans un corps de Gallois. M. Arnaud Dagnelies présente dans sa thèse une méthode de généraliser les codes de Reed Solomon en encodant cette fois ci avec des évaluations polynomiales.

Le but de cette étude est de comprendre à quelles conditions on a équivalence ou égalité entre les codes de Reed-Solomon (RS) et les codes de Reed-Solomon généralisés (GRS).

## Génération des codes

### Corps de Galois

#### Corps à 8 éléments

Encoder des données à l’aide d’un code de Reed-Solomon consiste à faire des opérations dans un cors fini. Un mot de code est une suite de symboles qui sont des éléments de ce corps. On va donc choisir des corps de Galois.

Dans un premier temps, pour simplifier, on va s’intéresser à des codes de petite taille. On prend donc le corps de Galois à 8 engendré par le polynôme à coefficients dans {0, 1}

|  |  |
| --- | --- |
| Polynôme générateur : | |
| Forme Polaire | Forme Cartésienne |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Avec ce corps de Galois, on a les tables d’addition et de multiplication suivantes :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

#### Corps à 16 éléments

On s’intéressera aussi à des codes dans un corps à 16 éléments. On prendra le corps généré par le polynôme

|  |  |
| --- | --- |
| Polynôme générateur : | |
| Forme Polaire | Forme Cartésienne |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### Codes de Reed Solomon

Les mots de code sont écrits sur n=7 symboles, chaque symbole étant un élément du corps de Galois choisi.

On veut ici un code 1-correcteur (t=1), donc les mots à encoder sont des mots de   
 symboles. Il y a donc mots de codes.

#### Méthode d’encodage systématique

Pour avoir un code 1-correcteur, on doit choisir un polynôme à deux racines consécutives. On choisit dans un premier temps

On écrit le mot à encoder sous forme polynômiale  de degré au plus 4 : chaque symbole du mot est un coefficient de . Par exemple le mot sera noté

On divise ensuite par et on note le reste.

Le mot de code à transmettre est

### Codes de Reed-Solomon généralisés

La méthode d’encodage présentée ici est expliquée dans la thèse de Arnaud Dagnelies. Ici l’encodage est réalisé par évaluation polynomiale.

#### Méthode d’encodage

Soit le vecteur des points où l’on va évaluer le code GRS, avec

Soit le vecteur de normalisation non nul.

On note comme précédemment le mot à encoder sous forme polynomiale.

Alors le mot encodé est {}

### Exemples

Soit le mot de code . On a donc

#### Reed-Solomon

Soit

On fait la division euclidienne :

Donc

Donc

Donc le mot encodé est

#### Reed-Solomon généralisé

Soit et

Donc ici le mot encodé est

### Notes théoriques sur les RS et GRS

Les codes correcteurs sont caractérisés par leurs matrices de génération G et de contrôle G. Encoder un mot, revient à le multiplier par G, et la matrice G permet de détecter les erreurs.

On se place dans un corps de Galois à éléments (n éléments non nuls). On note le code de Rees-Solomon obtenu par division polynômiale par , et le code de Rees-Solomon généralisé obtenu avec de .

#### Codes RS (sous forme non systématique)

Le code où a pour matrice G :

Soit tel que , Alors à pour matrice de contrôle H :

#### Codes GRS

On note la longueur du message et la longueur des mots de codes.

Soit et , le code a pour matrice génératrice G :

La matrice de contrôle H est de la forme :

Où les sont obtenus en résolvant le système :

avec

C’est-à-dire

## Comparaisons pratiques des codes RS et GRS

### Condition pour la cyclicité des codes GRS

Pour pouvoir cmparer les codes RS et GRS, on doit savoir à quelles conditions un codes GRS est en fait un code RS cyclique. La proposition suivante, tirée du livre Notes en Coding theory de J.I Hall, donne ces conditions.

Notation :

Pour une racine primitive de un corps de Galois, on note :

Remarques :

* est le vecteur dont toutes les composantes sont **1.**
* est la liste des puissances de .
* . C’est-à-dire qu’un décalage cyclique de est un multiple de .

Proposition :

est cyclique. C’est-à-dire est cyclique si et avec

Preuve :

On a vu que pour , le terme de la matrice génératrice est

Donc pour , le terme de la matrice génératrice est  
, avec . Donc les lignes de la matrice sont les pour .

On a vu que les sont des décalages de , donc le code est cyclique.

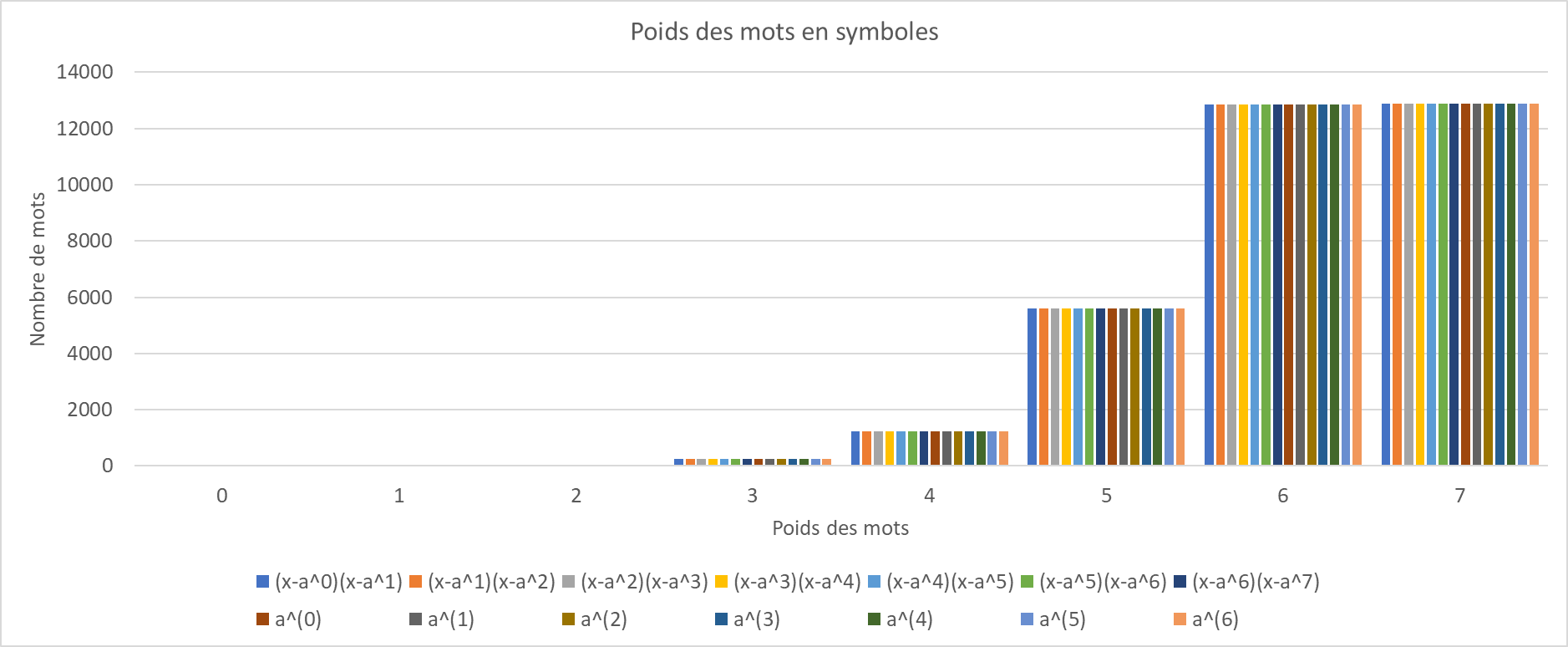
### Répartitions des poids des mots de codes [5,7]

Dans un corps de Galois à 8 éléments, il y a 7 codes de Reed-Solomon [5,7] :

Il existe également 7 codes de Reed-Solomon généralisés qui respectent la condition de cyclicité donnée par la proposition précédente :

#### Equivalence en poids en symboles

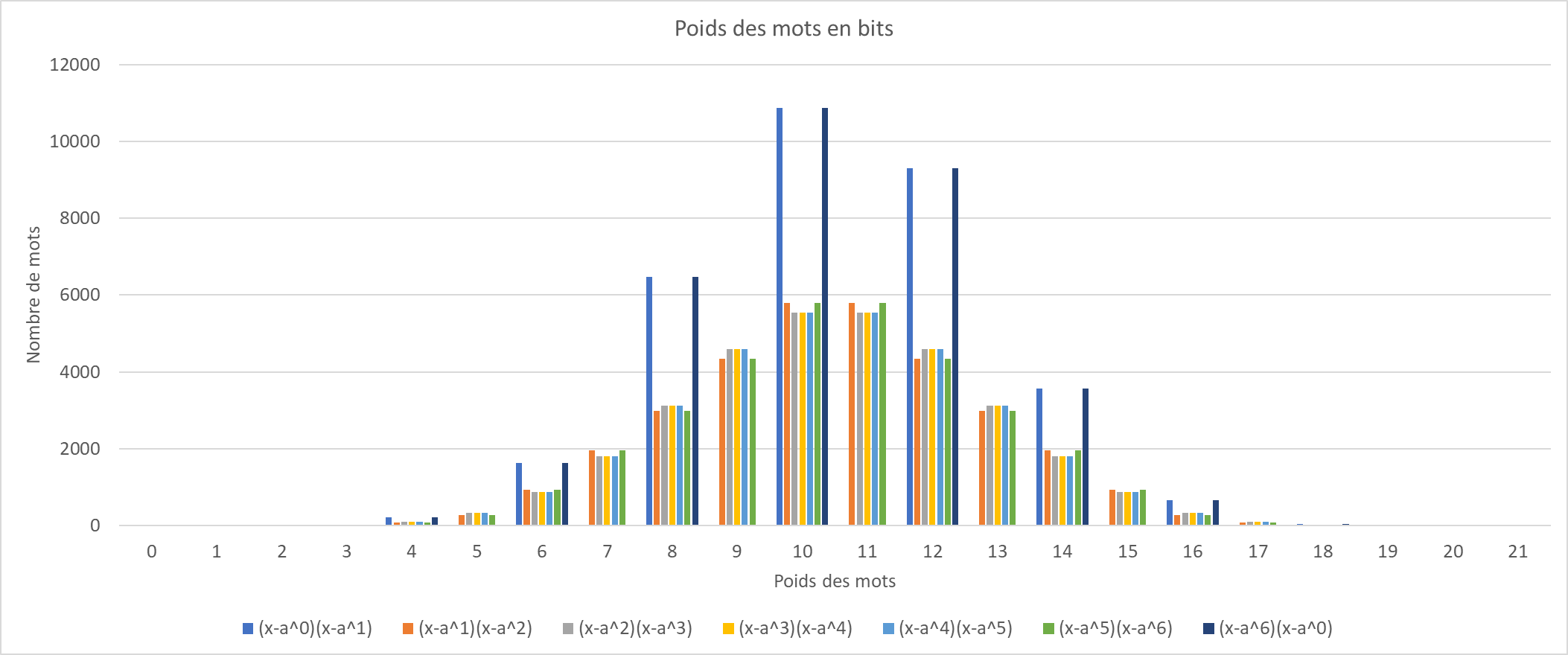
Tout d’abord, on regarde si les codes sont équivalents en poids en symboles. C’est-à-dire qu’on compte le nombre de symboles non nuls dans chaque mot des codes.



On voit que les 14 codes [5,7] sont parfaitement équivalents. Si on veut voir une différence il faut regarder l’équivalence en poids en bits.

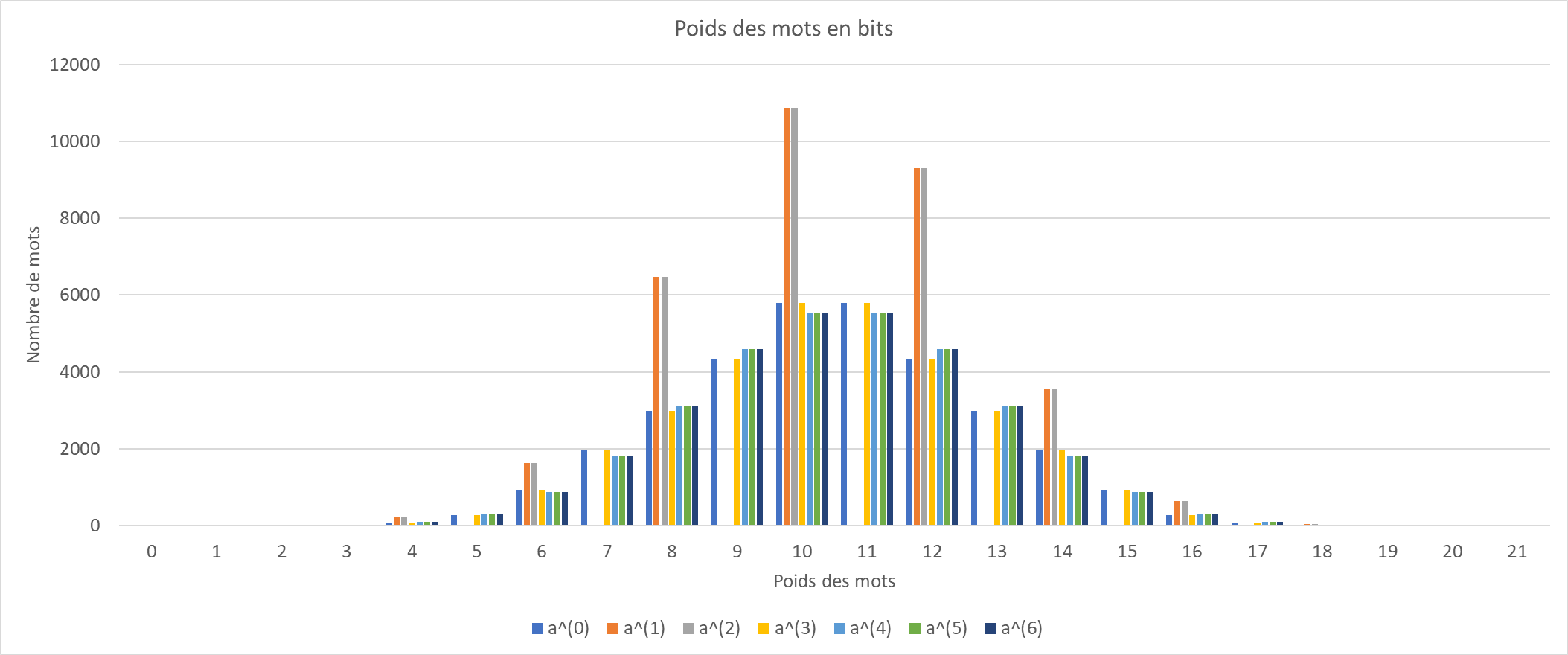
#### Equivalence en poids en bits

Tout d’abord on regarde la répartition des poids des mots des 7 codes de Reed Solomon.



On voit l’apparition de 3 familles d’équivalence :

On regarde maintenant la répartition des poids des 7 codes GRS.



On remarque de nouveau les 3 même familles d’équivalence.

Finalement, on a donc 3 familles d’équivalence :



### Classes d’équivalences des codes [5;7] [5;15] et [7;15]

On note

#### Codes [5 ; 7]

#### Codes [5 ; 15]

#### Codes [7 ; 15]

#### Remarques

On remarque que dans les 3 cas, on a un nombre tel que . De plus, on a pour les deux codes où et pour le code où .

### Comparaison des matrices des codes

#### Rappels théoriques :

Codage sous forme systématique :

Un code sous forme systématique est un code dont la matrice génératrice est de la forme :

Propriété :

Tout code est équivalent à un code mis sous forme systématique que l’on trouve par l’algorithme du Pivot de Gauss.

#### Comparaisons des matrices

Comparons les matrices des codes d’une des classes d’équivalence, par exemple les codes :

Les formes des matrices sont données à la partie II.5

On note (resp. ) les matrices génératrices du code (resp. ) et (resp. ) les matrices des codes sous forme systématique équivalents.

Code  :

Code :

Code :

Code :

La première chose que l’on remarque, c’est que et . Ces codes sont en fait égaux comme on le démontrera dans la partie suivante. C’est-à-dire que leurs mots de code sont les mêmes.

Si on compare toutes les matrices, on a :

On a en fait : , c’est-à-dire qu’une matrice d’un code cyclique est égale à une matrice d’un code généralisé.

### Une image contenant texte Description générée automatiquementComparaison des polynômes générateurs

Voici les polynômes des codes [7 ; 15] :

Si on prend deux codes équivalents avec leurs polynômes notés et .

On remarque qu’on a

C’est-à-dire que le polynôme est le polynôme renversé et normalisé.

Cela se vérifie pour tous les codes testés.

Ce critère n’est pour l’instant pas prouvé, mais on pourra l’utiliser pour vérifier les classes d’équivalences de codes plus gros et donc trop long à calculer comme les codes . En effet calculer les polynômes est très simple pour toutes les tailles de codes. On ne pourra pas comparer aux codes GRS, mais on va donner dans la partie IV un critère d’égalité entre RS et GRS.

#### Application du critère

Ici, on va simplement appliquer le critère à d’autres familles de codes en supposant qu’il est vrai.

Codes [9 ; 15]

## 

Ici on a avec

Tableau des valeurs de en fonction des codes.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 3 | 7 | 4 | 2 |
| 3 | 15 | 12 | 2 |
| 4 | 7 | 3 | 6 |
| 4 | 15 | 11 | 10 |
| 5 | 7 | 2 | 3 |
| 5 | 15 | 10 | 3 |
| 6 | 7 | 1 | 0 |
| 6 | 15 | 9 | 11 |
| 7 | 15 | 8 | 4 |
| 7 | 31 | 24 | 4 |
| 8 | 15 | 7 | 12 |
| 9 | 15 | 6 | 5 |
| 10 | 15 | 5 | 13 |
| 11 | 15 | 4 | 6 |

## Egalité entre RS et GRS

Les résultats exhibés dans cette partie sont présentés par Jonathan I Hall dans son livre *Notes on coding theory*

### Condition d’égalité entre codes GRS et RS

Lemme IV.1.1 :

Preuve en Annexe.

Lemme IV.1.2 :

Pour un polynôme dans , un corps, et , tel que

Preuve :

Par division polynômiale, on a l’existence de et tels que avec , donc est une constante. On trouve que en calculant .

Conséquence :

Théorème :

Ce critère confirme bien les résultats de la partie III.4 où on avait trouvé des matrices sous forme systématiques identiques pour certains codes.

Preuve :

Soit . Par le **Lemme IV.1.1**, on a .

Donc comme vu précédemment, les lignes de la matrice génératrice de sont les pour .

Ainsi, pour et un mot de code et sa forme polynômiale,

Ainsi, par la conséquence du **Lemme IV.1.2**, on a :

Comme est unitaire et de degré , il s’agit du polynôme générateur de C.

De plus les sont des puissances consécutives de , donc .

# Décodage des codes GRS

## Introduction

### Principe de l’algorithme

Soit le mot envoyé tel que et le mot reçu. On prend tel que , .

Par exemple, si le polynôme utilisé pour encoder est sur , l’ensemble des points envoyés est :

.

On suppose que le mot reçu est tel que l’ensemble des points est :

.

Il faut maintenant trouver les polynômes de degré inférieur à 2 qui passent par le maximum de points. Par exemple ici les polynômes et sont des candidats équiprobables sans informations supplémentaires.

On voit que le problème est donc un problème d’interpolation polynomiale.

Cet algorithme, plus simple que celui qui est présenté dans la suite, est l’algorithme de Sudan. Cet algorithme ne permet que du décodage dur, c’est-à-dire qu’on assigne à chacun des points reçus une valeur donnée. L’algorithme suivant permet du décodage « souple ». Cela permet de prendre en compte plus d’informations présente dans le canal de transmission. C’est-à-dire qu’on assigne à une probabilité qu’il soit égal à chaque élément de , c’est ce que l’on appellera dans la suite la matrice de fiabilité. Cependant, le principe de l’algorithme présenté ici reste le même que l’algorithme de Sudan : il s’agit de trouver les polynômes correspondants aux mots les plus probables.

### Degré pondéré et énumération monomiale

Définition : Le (a,b) degré pondéré d’un monôme est :

Definition : Le (a,b) degré pondéré d’un polynôme est le plus grands degré pondéré de ses monômes.

L’algorithme est basé sur un polynôme bivarié . Si on remplace par , on a : . De plus, est de degré au plus . Donc le degré de est majoré par le degré pondéré de .

On note

Définition : L’ensemble est l’ensemble des monomes de (a,b) degré pondéré inférieur ou égal à

### Zéros de multiplicité m

Définition : un polynôme a un zéro de multiplicité en si   
.

Théorème : Si a un zéro de multiplicité en et , alors

### Matrice de multiplicité

On note les points où le code GRS est évalué et le vecteur contenant tous les éléments de

Définition : La matrice de multiplicité est la matrice telle que pour tout , le polynôme a un zero de multiplicité au point .

Définition : Le coût de la matrice est défini par :

Où est la matrice dont toutes les composantes sont égales à .

Le coût d’une matrice est égal au nombre de contraintes sur les coefficients de qu’il faudra satisfaire lors de l’interpolation.

Notation : On note le plus petit entier tel que

Le but de cet entier est de savoir le degré minimum que doit avoir le polynôme afin d’avoir suffisamment de coefficients pour pouvoir satisfaire toutes les contraintes imposées par . Si , le polynôme devra satisfaire contraintes sur ses coefficients, et donc devra avoir au moins coefficients. Donc le degré du polynôme doit être tel que **.**

Définition : Le score du mot est :

Théorème de décodage : Soit le polynôme qui satisfait les multiplicités de la matrice , , et soit un polynôme d’encodage évalué en et qui donne le mot . ()

Si et

Alors

Ce théorème signifie que si , alors le décodage à fonctionné.

## Construction de la matrice de multiplicité

### Matrice de fiabilité

La matrice de fiabilité est la matrice qui concentre toutes les informations du canal de communication. Elle est construite à partir des informations que l’on a, c’est-à-dire ce que l’on reçoit, et d’autres informations comme la probabilité qu’une erreur ait lieu si le canal transmet les bits un par un. L’avantage de cette modélisation par rapport à l’algorithme de Sudan, est que l’on peut prendre en compte des informations plus complètes. Par exemple :

Dans un canal, la tension -1.5V correspond à 0, -0.5V à 1, 0.5V à 2 et 1.5V à 3. On reçoit . Avec l’algorithme de Sudan, on devrait choisir ou et appliquer l’algorithme ensuite. Ici on peut dire par exemple que . Et donc utiliser plus d’informations que l’on a à disposition. Dans les faits on utilisera des lois normales pour calculer ces probabilités.

Définition : La matrice de fiabilité est une matrice telle que

est une variable aléatoire dans qui représente le mot envoyé, et est une variable aléatoire qui prend ses valeurs dans ce que l’ont reçoit du canal.

Dans un premier temps, on prendra également à valeurs dans ce qui permettra de traiter le cas particulier d’un canal dur, mais l’algorithme qui suit fonctionne exactement de la même manière pour un canal souple. Seule la construction de la matrice est différente.

### Exemple pour un canal dur

On reprend le corps de Galois à 8 éléments présenté au début.

|  |  |
| --- | --- |
| Polynôme générateur : | |
| Forme Polaire | Forme Binaire |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Les symboles sont transmis via le canal bit à bit, et la probabilité qu’un bit soit inversé lors de la transmission est .

On note la distance entre deux symboles. C’est le nombre de bits différents entre les deux. Par exemple, .

On a donc

Par la formule de Bayes,

### Remarques et approximations

On note et vecteurs de variables aléatoires.

On remarque que la probabilité n’est pas égale au produit . En effet, les vecteurs sont des éléments de , mais on n’a pas l’égalité entre et . On n’a que .

La matrice de multiplicité idéale, serait celle qui maximise la probabilité de décoder. C’est-à-dire telle que soit maximale. Cependant ce problème est extrêmement difficile car la matrice affecte les deux côtés de l’inéquation et les probabilités sont difficiles à calculer. Ainsi, on va faire deux approximations.

La première est que l’on ne va pas chercher la matrice la plus optimale parmi toutes, mais simplement la matrice qui maximise le score moyen parmi les matrices de même coût.

La seconde est que l’on va utiliser l’approximation :

Ainsi le problème revient à trouver parmi les matrices de coût qui maximise

### Algorithme glouton

**Initialisation :** On pose la matrice nulle.

Itération : Choisir tel que le ratio soit maximal, puis augmenter de .

Lorsque l’on augmente de , on augmente le coût de et le score moyen de . Maximiser le ratio ci-dessus revient donc à maximiser la probabilité de décoder le mot.

Fin : On s’arrête juste avant que dépasse un seuil fixé. On verra que la complexité de l’interpolation dépend fortement de , on veut donc la matrice la plus optimale à fixé.

Comme on l’a vu dans le théorème de décodage, pour que le décodage fonctionne, c’est-à-dire pour qu’on retrouve le mot envoyé avant corruption, on doit avoir   
, où est le mot envoyé. Dans cet algorithme, on cherche donc à maximiser le score moyen tout en minimisant . A chaque itération on augmente les chances que le mot soit décodé, mais on augmente aussi le coût. Il s’agit donc d’optimiser la balance performance/temps d’execution.

### Exemple

On se place dans un corps de Galois à 4 éléments , de polynôme générateur   
. On a

On suppose que la matrice de fiabilité est :

On démarre avec matrice nulle.

1. Pour , on a maximal. On augmente donc de 1. On a donc :
2. maximal, On a donc :
3. maximal, donc :
4. , . On prend donc , donc :
5. De même :
6. De même :
7. , . On prend donc , donc :
8. De même :
9. De même :
10. , 05. On prend donc , donc :

Fin : On s’arrête juste avant que dépasse un seuil que l’on choisit.

Pour calculer on calcule tout d’abord le coût de :

On doit ensuite trouver tel que et

Pour , et . Donc

### Algorithme proportionnel

L’algorithme gourmand permet de bien comprendre comment on construit la matrice en optimisant la balance performance/coût mais on peut obtenir le même résultat plus rapidement.

Propriété : tel que, après itérations,

Preuve :

On note la valeur initiale des . On a évidemment .

On note la suite des valeurs du ratio . est donc le nombre de fois qu’on a incrémenté lors de l’algorithme glouton. Donc .

Donc

Si alors est strictement décroissante.

Donc

Donc si alors

Ce qui se réécrit :

Donc en posant on a c’est-à-dire .

Cette propriété nous permet d’éviter les premières itérations de l’algorithme en choisissant et en initialisant à . Ensuite, on continue l’algorithme glouton et on s’arrête juste avant que ne change de valeur. Pour mieux contrôler le coût, on peut choisir une multiplicité maximale , qui correspondra à la valeur maximale des avant de passer à la partie gloutonne de l’algorithme. On pose ensuite .

Cette partie de l’algorithme n’est pas critique et l’optimiser ne fait donc pas gagner beaucoup de temps, mais c’est une optimisation facile.

## Interpolation de Kötter

### Contraintes sur Q

Soit . Le but est de savoir comment choisir les afin que ait effectivement des 0 de multiplicités choisies en des points donnés.

En , le polynôme s’exprime . Avoir un 0 de multiplicité en signifie que lorsque .

Exprimons maintenant les en fonction des .

En isolant les termes de degré on peut en déduire l’expression des .

On notera . On remarque que cet opérateur est un opérateur linéaire, et donc que .

Avec cet opérateur, «  a un zéro de multiplicité en » s’écrit . On retrouve bien ici que pour un zéro de multiplicité , on a contraintes linéaires ce qui correspond au calcul du coût de la matrice de multiplicité.

Le problème d’interpolation revient donc à résoudre un système linéaire à équations.

### Relation d’ordre sur

est l’ensemble des monômes a deux variables .

Définition : Soit , On dit que si et seulement si :

On peut définir la même relation d’ordre sur les polynômes en regardant l’ordre du monôme dominant.

### Algorithme

Pour plus de simplicité, quand on parlera du degré de on fera en fait référence au degré pondéré .

En revenant à l’algorithme de décodage, on voit que pour que le décodage fonctionne, on doit avoir . Le but est donc de trouver le de plus petit degré , qui respecte les contraintes définies par la matrice de multiplicité.

Les contraintes forment en fait un système linéaire à équations. Un polynôme de degré a au plus coefficients non nuls. C’est-à-dire qu’il a degrés de liberté. Pour résoudre le système, il suffit donc que (la condition n’est pas nécessaire si les contraintes ne sont pas indépendantes). Or le plus petit tel que est . Il existe donc au moins une solution de degré . Si l’algorithme trouve le de plus petit degré solution du problème, on sera sûr d’avoir et donc on pourra décoder si . Si les contraintes sont indépendantes, on a .

On note l’ensemble des polynômes tels que Le principe de l’algorithme est de trouver le polynôme solution du problème de degré minimal dans chacun de ces ensembles, puis de sélectionner celui de plus petit degré parmi ces solutions qui sera la solution optimale. Ce polynôme est bien la solution car l’union des contient l’ensemble des polynômes de degré inférieur ou égal à .

On va renommer les en . On verra plus loin qu’il y a une contrainte sur l’ordre des .

Initialisation :

Soit le degré en maximum de la solution . .

On initialise polynome :

On a donc

Le but à chaque itération va être de modifier les afin qu’il respecte une contrainte de plus, tout en restant dans , et en restant le polynôme de plus petit degré possible.

Itération :

A la ieme itération, est le polynôme de de plus petit degré respectant pour

* Pour chaque , on calcule , la divergence .
* Parmi les de divergence non nulle, on note l’indice du de plus petit rang.
* Pour chaque , on met à jour le polynôme :
* On met à jour  :. (Le est tel que )

Fin :

Après la dernière itération, il suffit de choisir le polynôme de plus petit rang parmi les . Il s’agit du polynôme de plus petit rang solution du système.

Complexité :

On note . On a donc itérations en tout.

A chaque itération, le calcul limitant de cet algorithme est le calcul des divergences .

On doit à chaque itération, calculer divergences.

dépend linéairement de .

Le nombre de divergences à calculer est donc .

Le calcul d’une divergence est un calcul en .

Donc l’algorithme est en .

### Preuve de l’algorithme

On note

Théorème :

Si est le plus petit polynôme dans

Alors est le polynôme de plus petit rang dans

Preuve :

* Si , alors on a bien . De plus, comme et que est le polynôme de plus petit rang dans alors il est aussi le polynôme de plus petit rang dans .
* Si et ,  
  De plus , Donc   
  De plus , donc est de même rang que .  
  Donc est le polynôme de plus petit rang de .
* Si , on va voir apparaître la condition dans l’ordre des contraintes mentionnées plus tôt.  
  On a donc facilement par la linéarité des contraintes.
* A FINIR

## Factorisation

### Introduction

Le but ici est de trouver les facteurs de de degré inférieur à .

On note . Si est une racine de telle que , alors on a . C’est-à-dire que est une -racine de .

Définition : Polynôme normalisé

Le polynôme normalisé est défini par

Où est le plus grand entier tel que .

Propriété :

En normalisant , on évite que soit le polynôme nul. Ainsi, en calculant   
 pour où les sont les éléments de , on peut connaitre toutes les racines de .

Le principe de l’algorithme est de construire terme par terme. C’est-à-dire calculer à partir de .

### Algorithme de Roth-Ruckenstein

Théorème :

On pose :

Si

et

Alors

### Exemple

On se place dans corps de Galois à 4 éléments, avec . On cherche donc les sous la forme .

Le polynôme suivant a été calculé en décodant le mot erroné .

Les racines de sont :

1. On s’intéresse tout d’abord à :

a pour racine .

On a donc racine de .

a pour racines et

On a donc et racines de .

a pour racine

On a donc racine de .

On a maintenant une liste de 4 mots candidats. Il faut maintenant choisir le meilleur en réencodant les mots avec le code de Reed Solomon et en choisissant celui qui a le meilleur score avec la matrice de multiplicité.

## Performance de l’algorithme.

Le but dans cette partie sera de tracer le pourcentage de réussite de décodage en fonction de caractéristique du canal.

On prend ici l’exemple d’un canal souple où les symboles sont transmis à l’aide d’une valeur analogique entre 0 et 7. On se place dans le corps de Galois à 8 éléments étudié dans la première partie. On considère que la valeur 0 du canal correspond à 0 dans le corps et 7 correspond à . Lors de la transmission, les erreurs sont caractérisées par un bruit gaussien d’écart type .

Pour tester les capacités de correction de l’algorithme, on va choisir des mots de codes au hasard auxquels on va ajouter le bruit gaussien caractéristique du canal, puis essayer de décoder. On calcule ensuite le taux de correction pour plusieurs valeurs de et on trace le taux de correction en fonction de sigma. On peut ensuite recommencer ce procédé pour plusieurs valeurs de multiplicité maximale lors de la génération de la matrice de multiplicité.

L’algorithme peut être incapable de décoder pour deux raisons différentes :

* Il peut se tromper de mot et donc décoder un mot qui n’est pas le bon. C’est ce qu’on observe lorsque sigma augmente, le mot reçu est trop éloigné de celui envoyé pour corriger.
* L’algorithme peut être incapable de trouver une racine lors de la factorisation. Lorsqu’il n’y a pas assez d’informations, aucun mot n’est trouvé. C’est ce qu’il se passe pour max\_mult=3 et sigma entre 0.1 et 0.4. L’algorithme est souvent incapable de trouver un mot -même faux- car il n’y a pas assez d’informations à disposition dans la matrice de multiplicité.

Je n’ai pas testé pour des valeurs plus importantes de multiplicité maximale pour des raisons de performance. La complexité étant cubique, le temps d’exécution augmente énormément avec la multiplicité maximale dans la matrice de multiplicité.

### Comparaison avec le décodage classique

Le code étudié ici est un code 1-correcteur. On considère alors le canal comme dur, puis en appliquant l’algorithme on peut corriger une erreur. A la réception, on décide d’attribuer la valeur la plus proche de celle reçue à chaque symbole. Par exemple si on reçoit 1.2, on décide qu’on a reçu . Si on a reçu 3.7, on décide qu’on a reçu .

Un symbole est donc mal transcrit avant le décodage si la valeur est éloignée de plus de 0.5 de celle envoyée. Cette probabilité d’erreur dépend de . Ensuite, on sera capable de décoder si il n’y a qu’une seule ou aucune erreur. C’est-à-dire que la probabilité de décoder est . On trace donc en fonction de sigma pour comparer les méthodes de décodage.

Dans le cas d’un code , on constate donc que dès max\_mult=4, le décodage souple est plus efficace que le décodage classique.

# Exemple d’encodage et décodage sur un petit code

Le but de ce chapitre est de montrer un exemple complet de tout ce qui a été étudié dans ce rapport. On va se placer dans le corps de Galois à 4 éléments de polynôme générateur   
.

## Description du corps

|  |  |
| --- | --- |
| Polynôme générateur : | |
| Forme Polaire | Forme Cartésienne |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Encodage

On va s’intéresser aux codes correcteurs de Reed-Solomon afin d’avoir des codes qui permettent de corriger 2 erreurs.

Il existe donc 3 codes de Reed-Solomon :

|  |  |
| --- | --- |
|  | |
| Message | Mot de code |
|  |  |
|  |  |
|  |  |
|  |  |
|  | |
| Message | Mot de code |
|  | 000 |
|  | 111 |
|  | 222 |
|  | 333 |
|  | |
| Message | Mot de code |
|  | 000 |
|  | 321 |
|  | 132 |
|  | 213 |

Parmi les codes de Reed-Solomon généralisés avec k=1 et n=3, il y en a donc 3 qui sont cycliques et égaux aux codes précédents. Pour ces trois codes, .

|  |  |
| --- | --- |
|  | |
| Message | Mot de code |
|  |  |
|  |  |
|  |  |
|  |  |
|  | |
| Message | Mot de code |
|  | 000 |
|  |  |
|  |  |
|  |  |
|  | |
| Message | Mot de code |
|  | 000 |
|  |  |
|  |  |
|  |  |

On avait vu la condition d’égalité entre les codes de RS et GRS :

On voit qu’on a effectivement :

## Décodage

On se place dans le cas où . C’est le seul que l’on peut décoder.

L’algorithme présenté ici ne permet pas de décoder avec . On va donc s’intéresser au code . Ce code n’a pas d’intérêt pratique car son taux de correction est très faible   
(< 1 erreur en moyenne avec un décodage classique) mais le décodage fonctionne de la même manière et peut être présenté à la main. On peut utiliser ce code pour décoder dans certains cas le code précédent. En effet, lors de la sélection du polynôme parmi les racines après la factorisation, on peut choisir d’ignorer tous les polynômes de degré 1 et de ne choisir que les polynômes de degré 0. En effet le code est un sous code du code correspondant au sous ensemble des polynômes de degré 0 pour encoder.

On va supposer ici que . On a reçu le mot .

La matrice de fiabilité est donc :

Avec max\_mult=4, on a la matrice de multiplicité :

On a donc .

L’interpolation donne : . Ce polynôme est le polynôme de le plus faible ayant des zéros correspondant à la matrice M. C’est-à-dire :

* Multiplicité 4 en
* Multiplicité 4 en
* Multiplicité 3 en

On cherche ensuite les facteurs avec de degré maximum 1 de .

Il y a 4 polynômes qui correspondent :

On calcule ensuite le score de chacun des mots candidats en encodant les avec le code de Reed-Solomon puis en calculant le score du mot de code correspondant.

Les mots de codes correspondant aux sont :



Le mot le plus probable est donc

# Bibliographie

1. Nicolas Drouin, Cryptographie et Codage, 2021, U2IS ENSTA Paris
2. Arnaud Dagnelies, Algebraix Soft-Decoding of Reed-Solomon Codes, 2007, Université Catholique de Louvain
3. J. I. HALL, Notes on Coding Theory, 9 Septembre 2010, Department of Mathematics Michigan State University

# Annexes

## Annexe A : Preuve du Lemme IV.1.1

Notations :

et

Lemme A.1 :

Si sont les n racines de , alors

Lemme A.2 : Code dual

où

*Preuve :*

Soient

est de degré inférieur à et est de degré inférieur à , donc est de degré inférieur à .

Par interpolation de Lagrange, on a :

On regarde la valeur des coefficients de des deux côtés de l’égalité :

Le produit scalaire des mots de codes est toujours égal à 0 ce qui démontre le Lemme.

Lemme A.3 :

Soient des éléments non nuls de , et un vecteur de dont toutes les composantes sont égales, alors

Lemme IV.1.1 :

*Preuve :*

Par le **Lemme A.2**, on a où pour   
 en notant , on a :

De plus par le **Lemme A.1**: Donc :

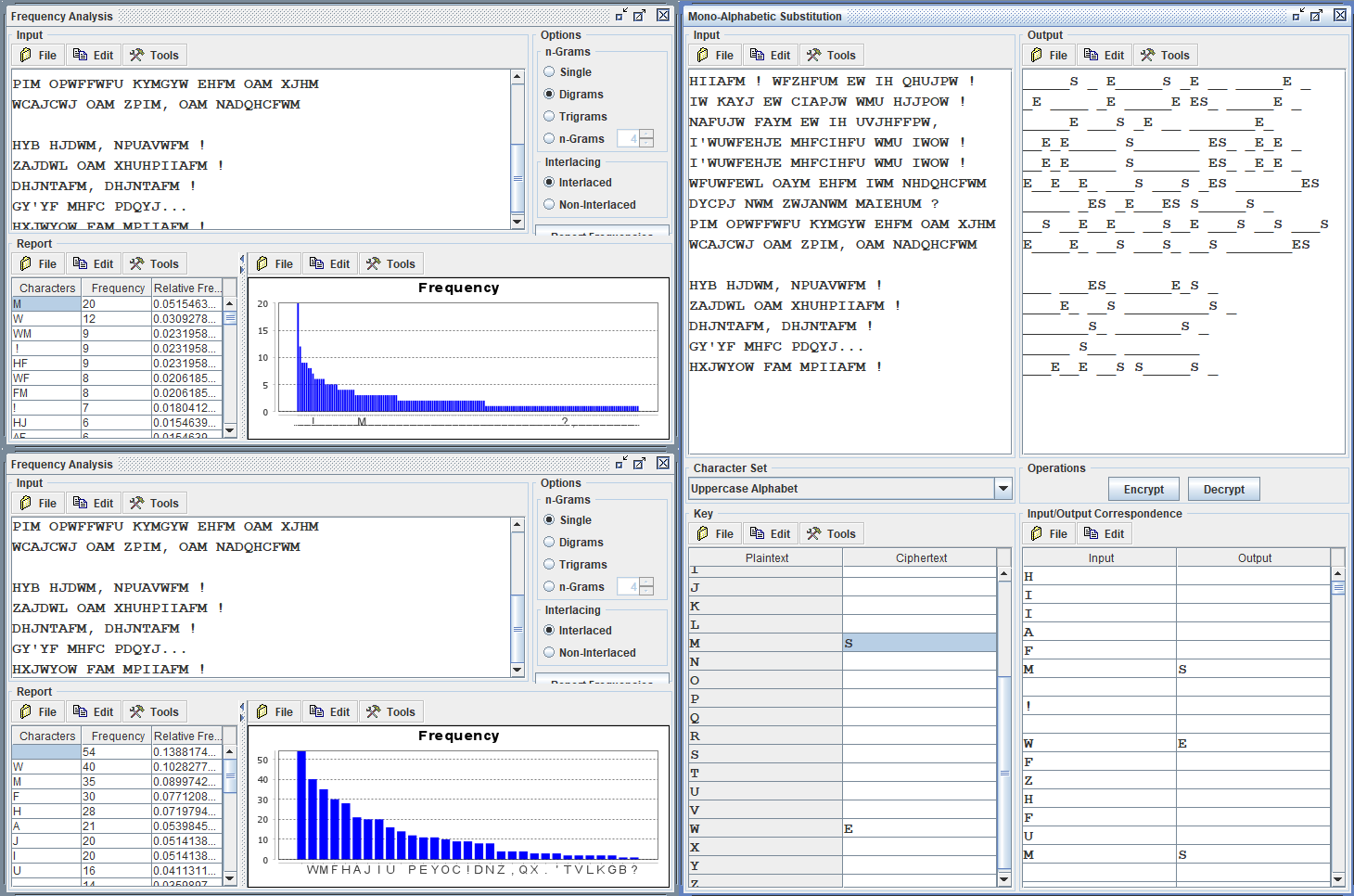
Donc

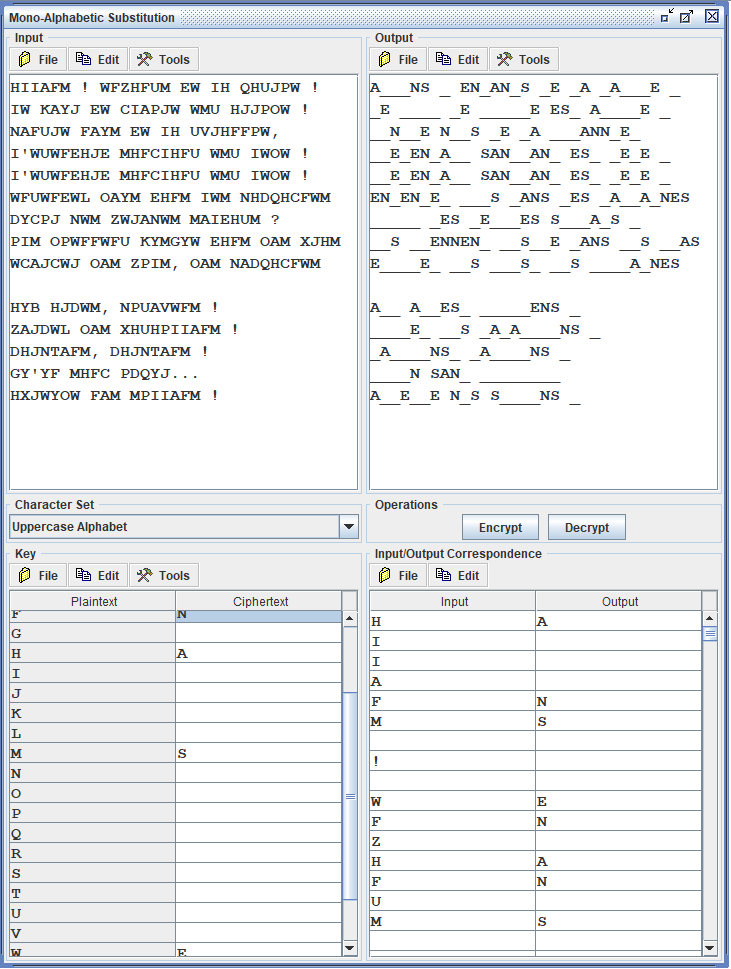
Donc par le **Lemme A.3 :**

## Annexe B : CryptoExplorer

Lien du programme : [The Home Page of Dr. Donald E. Spickler Jr. (salisbury.edu)](http://faculty.salisbury.edu/~despickler/personal/CryptographyExplorer.html)

(http://faculty.salisbury.edu/~despickler/personal/CryptographyExplorer.html)



* On voit que les lettres les plus courantes sont, dans l’ordre : W,M,F,H
  + On peut remplacer W et M par E et S.
* Les digrammes les plus courants sont WM, HF, WF, FM
  + Le digramme le plus courant en français est ES : il correspond à WM.
  + Le deuxième digramme le plus courant commençant par E est EN. Le F étant très présent dans le message chiffré et souvent après un W, on peut remplacer F par N.
* On a donc W->E, M->S, F->N, H->A.
* Ici on commence à reconnaitre des mots : par exemple le mot FAM correspond à N\_S. Il n’y a qu’une seule solution en français : NOS. Donc on a A->O.
* Le mot MHFC correspond à SAN\_. Il n’y a également qu’une seule solution en français : SANG. Donc C-> G
* Le mot WMU correspond à ES\_. On a donc U->T.

Une image contenant texte

Description générée automatiquement

* On peut également remarquer les lettres doubles. Le mot HIIAFM correspond à A\_\_ONS. En français, les consonnes qui peuvent se doubler sont : c,j,l,m,n,p,r,s,t,b,d,g,k,z. Seul L a du sens. On a donc I->L.
* Quelques lettres sont encore facilement reconnaissables comme le F de EN\_ANT, le D de SOL\_ATS, le I de \_LS et F\_LS puis le R de GLOI\_E.
* Ici le texte commence à être reconnaissable facilement.

Une image contenant texte

Description générée automatiquement

Remarques :

* Le A et le N ne sont pas dans le bon ordre de fréquence. Il est possible de le remarquer car le digramme AN est beaucoup plus fréquent que le digramme NA et le digramme EA n’existe presque pas alors que EN est l’un des plus fréquent.
* Les digrammes ne sont pas dans l’ordre, mais si on ne s’intéresse que au digrammes contenant une lettre donnée à une position donnée on peut retrouver des lettres.

