République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université d'El-oued – El-oued

Faculté des Sciences et technologies
Département d'électronique
Spécialité : Télécommunications

Mémoire de Fin d'Etude

En vue de l'obtention du diplôme de Licence académique

Thème

Les codes de Reed-Solomon et le problème de leur décodage

Présenté par :
Amouri Safia
Amouri Fatiha

Encadré par :
Chemsa Ali

2013/2014

Sommaire

Résumé

Le code de Reed-Solomon est un code correcteur basé sur les corps de Galois dont le principe est de construire un polynôme formel à partir des symboles à transmettre et de le suréchantillonner. Le résultat est alors envoyé, au lieu des symboles originaux. La redondance de ce suréchantillonnage permet au récepteur du message encodé de reconstruire le polynôme même s'il y a eu des erreurs pendant la transmission.Les codes Reed-Solomon sont des codes par bloc. En effet ils prennent en entrée un bloc de données de taille fixée, qui est transformé en un bloc de sortie de taille fixée. Ces codes travaillent sur un corps fini qui possède le plus souvent2^m éléments. Le plus souvent on prend $m = 8$ ou $m = 16$.

Mots clefs:

Code de Reed-Solomon, Message, Matrice génératrice, Matrice de parité, Probabilité d'erreur.

Abstract

TheReed-Solomoncodeis acorrector codebased onGalois fieldwhose principle isto construct a formalsymbolsfromtransmitpolynomialandtheoversampling. The result is thensentinstead of theoriginal symbols. The redundancy ofthis oversamplingallows the receiverof the messageencodedrebuild thesame polynomialif therewere errorsduring transmission.Reed-Solomon codes area block codes. Indeedtheyareinputina data blockoffixed size,which is converted intoan output blockoffixed size. These codesworkover a finite fieldhasmost often2^melements.Most oftenwe take$m = 8$and$m = 16$.

Key Words:

Reed-Solomon code, Message, Generator matrix, Parity-Check matrix, Error probability.

ملخص

ترميز ريد سولومون هوترميزتصحيحي يعتمد على حقلقالوى حيث مبدأه يعتمد على تكوين كثير حدود شكلي انطلاقا من الرموز المراد ارسالها ثم تؤخذ عينات منها. الشيء المتحصل عليه يبعث عوضا عن الرموز الأولية. تكرار أخذ العينات يسمح للمستقبل بإعادة إنشاء كثير الحدودحتى ولو كانت هناك أخطاء أثناء الإرسال. ترميزات ريد سولمون هيترميزات بالكتلة. بالفعل، مدخلها عبارة عن كتلة من المعلومات حجمها ثابت ثم تتحول إلى كتلة عند المخرج حجمها ثابت كذلك. هذه الترميزات تعمل في حقل منته يكون فيه غالبا 2^m عنصر حيث يؤخذ $m = 8$ أو $m = 16$.

المصطلحات الأساسية :

ترميز ريد سولومون، رسالة، المصفوفة المولدة، مصفوفة التحقيق، احتمال الخطأ.

Introduction générale

Les codes Reed-Solomon sont considérés comme étant un sous-groupe de codes cycliques de la famille des codes BCH. Ces derniers ont été mis au jour parHocquenghem[7] autour de 1960, et de façon indépendante par Bose et Ray-Chaudhuri [8, 9]. Lescodes BCH furent généralisés à tous les corps fini par Gorenstein et Zierler dans [10].

Approximativement à la même époque, Reed et Solomon ont développé et publié leurstravaux sur une famille de codes qui portera leur noms : les codes de Reed-Solomon [5].Les codes de Reed-Solomon aurait été découvert plutôt dans le contexte des matricesorthogonales par Bush en 1952 dans [4].

Les codes de Reed-Solomon ont trouvés applications dans le monde de la conservation d'informations numérique, ainsi que dans les systèmes de communication. Onretrouve par exemple le code $RS(255,233,33)$ qui est utilisé par la NASAdans lescommunications spatiales. Différentes versions sous le corps de Galois binaire exposanthuit $(GF(2^8))$ se retrouve dans les disques compacts, dvd, transmission HDTV, et puisd'autres versions sous $(GF(2^7))$ sont retrouvés dans les modems câbles entre autres.Malgré l'arrivé des codes Reed-Solomon, le monde pris bien du temps avant son utilisation, le problème de l'époque était de trouver des algorithmes de décodage efficace, ceque Berlekamp fit [11], puis Massey fit quelques améliorations dans [13]. Peterson [15]et Sugiyama [2] ont également contribué avec leurs algorithmes de décodages.

" C'est quoi les codes Reed-Solomon ?" serait probablement la première questionsi vous ne savez pas ce que c'est. Voici un façon simple de comprendre ce que c'est avantd'entamer la version poussé de la réponse à cette question. Notez que cela ne constituequ'un aperçu de réponse, la réponse clair, complète et détaillé est constitué de la totalitédu présent travail.

Lorsque l'on veut transmettre une information d'un point A au point B, il transite par ce que l'on désigne comme étant un canal de communication. Ce canal decommunication affecte, de façon volontaire ou pas, de façon généralement aléatoire etinconnu l'information que l'on tente de transmettre. Il l'affecte de façon à faire quelorsqu'un lecteur lira l'information au point B, il s'y trouvera des erreurs avec une certaine probabilité. Le codage, en général, tente de solutionner les erreurs de décodagede l'information reçu. La façon classique est de rajouter de l'information au messageenvoyé qui se trouve à être le résultat d'une fonction mathématique du contenu dumessage que l'on veut transmettre. Au plus simple, il peut être une simple répétitiondu message, et au plus complexe, bien, il n'y a pas de limite à la complexité du codage,mais souhaitant transmettre un minimum de quantité d'informations, on se créé unelimite.

Les codes de Reed-Solomon peuvent être vu comme étant un polynôme représentantl'information à transmettre qui a subi un suréchantillonnage, ce qui créé de l'information de redondance requise pour la correction d'erreurs. Ainsi, par le suréchantillonnement, on a un surplus d'informations qui permettra de corriger les portions erronées.

Les codes de Reed-Solomon sont basés sur un théorème d'algèbre linéaire disant quel'on peut définir de façon unique k points distinct par un polynôme de degré $k-1$ au plus.

Ce mémoire est structuré de façon à être partagé en trois chapitres :

Dans le premier chapitre, nous avons exposé la théorie des codes correcteurs linéaires. Après un bref rappel des principaux éléments de base sur lesquels reposent les fondements mathématiques de ce type des codes, nous avons regroupé toutes les propriétés essentielles qui peuvent être, plus tard, exploitées dans la suite de ce travail.

Le second chapitreest consacré principalement au fond du travail qui est les codes de Reed-Solomon. Ce type des codes possèdent beaucoup d'avantages par rapport aux autre codes.

Parmi les avantages remarquables de ces codes est que leurs algorithmes de décodage sont très développés. Ces algorithmes font l'objet du troisième chapitre.

Enfin, une conclusion générale est donnée comme couronnement de tout letravail.

Chapitre 1

Les codes correcteurs linéaires

1.1. Introduction sur les codes correcteurs

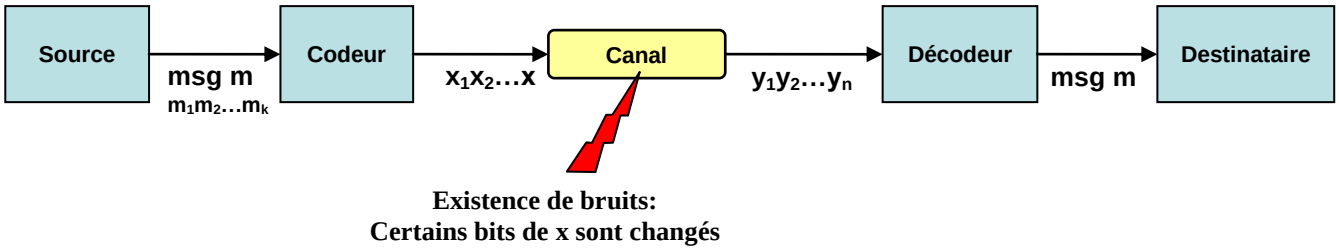
Pour commencer, il nous semble important d'introduire quelques concepts basiques concernant les codes correcteurs d'erreurs, qui seront utilisés dans les paragraphes suivants.

Nous illustrerons ensuite le concept de code correcteur par des exemples très simples.

1.1.1. Intérêt

En communication, une fonction d'encodage est une règle qui permet de convertir de l'information sous une autre forme de représentation, appelée code. Dans le domaine de la théorie de l'information, les codes sont utilisés pour détecter et corriger les erreurs de transmission. En effet, lorsqu'une source souhaite transmettre un message à un destinataire, elle envoie ce message au travers d'un canal dans lequel peuvent apparaître des erreurs dues à des bruits. Ainsi le message que le destinataire reçoit peut avoir été modifié. Afin de protéger l'information de ce genre d'erreurs, on ajoute au message de la redondance avant de le transmettre dans le canal. A la réception, le décodeur tente de récupérer le message originel à partir du message reçu.

Le schéma suivant illustre la chaîne d'actions qui permet à une source de transmettre de l'information à un destinataire [1]:



Un message $m = (m_0, m_1, \ldots, m_k)$ constitué de k symboles est généré par la source et envoyé au codeur. Celui-ci va l'encoder en mots de code x de longueur n, avec $n > k$. On envoie donc des mots de la forme $x = (x_1, x_2, \ldots, x_n)$ au travers du canal.

A la sortie du canal, on obtient un mot qui contient un certain nombre d'erreurs qui sera noté $y = (y_1, y_2, \ldots, y_n)$. Le décodeur récupère le message m à partir de yen appliquant une règle de décodage, qui peut être, par exemple, le principe du maximum de vraisemblance. On peut donc retrouver le message de départ, à condition que le nombre d'erreurs ne soit pas trop grand. En outre, plus on a ajouté de symboles redondants, plus on peut corriger d'erreurs.

Nous nous intéresserons donc aux mécanismes par lesquels l'information va être codée puis décodée, ainsi qu'aux outils mathématiques qui ont été développés pour ce fait.

1.1.2.Définition mathématique des codes [1]

On considère que les mots sont constitués d'éléments appartenant tous à un alphabet $\sum$ contenant q éléments. Le mécanisme de codage fournit des mots de code de n éléments appartenant tous à $\sum$.

Si on appelle $V_n(\sum)$ l'ensemble de toutes les séquences de taille n, on a donc :

$$|V_n(\sum)| = q^n$$

Un code serait donc un sous-ensemble de $V_n(\sum)$.Donnons une définition plus précise de ce terme ainsi que de la fonction d'encodage :

Définition : Un ***code*C** défini sur un alphabet $\sum$est un sous-ensemble de $\sum^n$. Le code ainsi obtenu est appelé un **q-ary code de longueur n.**

Définition : Une **fonction d'encodage** f est une fonction injective qui, à chaque élément de l'espace du message, associe un mot de code. On a donc:

$$f: \{espace\ du\ message\} \rightarrow \sum n$$

Un code est donc l'image obtenue par cette fonction : $C = Im\{f\}$

Dans le cas où $q = 2$, le code est dit **binaire**. On a alors $\Sigma = \{0,1\}$.Le code Cest donc un sous-ensemble de$\{0,1\}^n$ et$|V_n(\sum)| = 2^n$.

Notation : On note (n, M, d), un code qui transforme des blocs de message en mots de code de n éléments, avec une distance minimale égale à d. Mreprésente le nombre de mots de codes.

1.1.3. Caractéristiques d'un code [2]

Les codes correcteurs sont caractérisés par quatre données importantes :

1. La longueur des mots de code n

Dans les codes par blocs, on code le message en mots de longueur n. On souhaite envoyer des blocs de messages relativement grands car dans ce cas, le nombre d'erreurs se rapproche de son espérance, ce qui implique que la probabilité d'erreurs est meilleure. Cependant, au niveau du décodeur, si la complexité de l'algorithme de décodage est en $O(n^2)$, on souhaitera que n soit petit afin que le décodage ne soit pas trop long.

2. Le rendement ou taux de codage (« rate » en anglais)

Correspondant au nombre d'de symboles d'information des blocs du message divisé par le nombre d'éléments des mots de code, c'est-à-dire k/n. Si ce taux est très petit, cela signifie que pour envoyer un petit bloc de message, il va être nécessaire de transmettre un mot de code très grand. Or ceci n'est pas souhaitable car la vitesse de transmission va être fortement réduite, et il faudra beaucoup de temps pour envoyer un court message. Le but étant d'optimiser cette vitesse de transmission, on cherchera à avoir le meilleur rendement possible.

3. La probabilité d'erreurs du code

Cette probabilité va dépendre de la distance minimale entre les mots de code (ce terme sera défini plus clairement plus loin). Il est en effet évident que plus les mots

seront différents les uns des autres, plus la capacité à retrouver le message originel va augmenter.

4.La complexité de l'algorithme d'encodage et de décodage

La complexité de ces algorithmes est une donnée importante car elle décidera du temps de calcul et des ressources nécessaires pour l'exécution des fonctions de codage et de décodage. Il s'agit donc d'une donnée à prendre en compte lors du choix d'un type de code.

1.2.Exemples de codes simples [1]

Pour ces exemples de codes, nous considérerons des codes binaires. Les blocs de messages seront des éléments de $\{0,1\}^k$, et les mots de code générés seront des éléments de $\{0,1\}^n$.

Pour illustrer ces codes, nous utiliserons le bloc de message : 1011001.

1.2.1. Code répétitif

Il s'agit d'un code très simple dans lequel on répète n fois chaque bit du message à envoyer.

On a alors C ={(00…0); (11…1)}.

Illustration: Considérons le cas où $n = 5$, on obtiendrait alors le mot de code :

11111 00000 11111 11111 00000 00000 11111

On remarque qu'il n'y a que deux mots de code possibles qui sont {(00…0); (11…1)}, où les bits sont répétés n fois. On déduit donc que la distance minimale de ce code est de n, c'est-à-dire que les mots diffèrent de n bits au minimum.

De plus, on code des blocs de 1 bit en mots de n bits. Nous verrons plus loin que l'on note ce code $[n, 1, n]$, pour signifier un code de dimension 1, de longueur n et de distance minimale n. Nous verrons également que le fait que ces mots soient distants de n bits implique que l'on pourra détecter $n-1$ **erreurs** et corriger $\lfloor (n-1)/2 \rfloor$erreurs.

Illustration : Le code répétitif où$\lambda = 5$ peut donc détecter jusqu'à 4 erreurs et corriger 2 erreurs. En effet, si le décodeur applique la règle du maximum de vraisemblance,si un

mot a 3 erreurs, le décodeur va retourner le mauvais bit. En revanche, il ne pourra pas retrouver le mot originel car le mot reçu sera plus proche d'un autre mot de code.

Remarque : Si on analyse ce code, on se rend compte qu'il s'agit d'un code qui a un rendement faible puisque pour transmettre un bit il faut le répéter λfois.

1.2.2. Parity Check code

Dans ce code, on ajoute un bit correcteur à la fin du bloc de message pour indiquer si le nombre de 1 est pair ou impair. Il faut que le mot de code possède un nombre pair de 1.

Le codage s'effectue donc de la façon suivante :

Si le nombre de bits 1 dans le bloc est pair, on ajoute 0.
Si le nombre de bits 1 dans le bloc est impair, on ajoute 1.

On a donc $C = \{(c_1, \dots, c_n) / \sum c_i = 0\}$.

Illustration: Etant donné que dans notre exemple, le nombre de bits 1 est égal à 4, on ajoute un 0 à la fin du mot. Le mot de code ainsi généré serait donc : 10110010.

Si l'on code des blocs de $k$bits, on obtient donc des mots de code de $k + 1$ bits.
De plus, si on prend deux blocs qui ne diffèrent que d'un seul bit, on obtient deux mots de code distants de 2 bits. Ainsi la distance minimale du code est de 2.

On va donc noter ce code comme étant $[k + 1, k, 2]$. Si on pose $n = k + 1$, on peut également écrire cette définition de la façon $[n, n - 1, 2]$.
Ce code peut détecter 1 erreur, mais ne peut pas la corriger.

1.3. Codes linéaires

Dans la première section, nous avons introduit quelques généralités sur les codes. Les codes dont nous avons parlés étaient essentiellement des **codes par blocs** (blocks codes), c'est-à-dire des codes qui s'appliquent sur des blocs de symboles. On obtenait ainsi des mots de code qui étaient par la suite transmis dans le canal.

A partir de maintenant, nous allons nous restreindre à une sous-classe des codes par blocs, les **codes linéaires.** Nous étudierons le cas général des q-ary codes, ainsi que le cas plus particulier des codes binaires car ceux-ci sont très couramment utilisés pour la transmission d'informations.

Nous commencerons par en donner une définition, puis nous donnerons une description des matrices génératrices et de vérification, puis nous définirons plus précisément ce qu'est la distance minimum d'un code et nous expliquerons quel est son intérêt. Dans la section 3.6, nous définirons quelques bornes sur le nombre de mots par rapport à la distance minimale du code. Et pour finir, nous illustrerons les codes linéaires par un exemple de code connu, les **codes de Hamming**.

1.3.1. Définition [3]

A partir de maintenant, on considère que l'on code des blocs de symboles de k éléments. Le message émis par la source est donc tout d'abord découpé en blocs qui seront encodés séparément.

1.3.1.1.Définition générale des q-ary codes

Soit un alphabet $\sum = F_q$, avec q une puissance d'un nombre premier, i.e. $q = p^r$ avec p premier.

Définition: Un code par bloc Cest un code linéaire si et seulement si mots de codes de longueur n forment un sous-espace de dimension k de F_q^n.Ce code est noté$[n, k]$.

Ainsi, il existe $k$vecteurs g_0, g_1,...,g_{k-1} qui forment une base de ce sous-espace. Tout mot de code **c** appartenant à C peut s'écrire comme une combinaison linéaire de ces vecteurs :

$$c = u_0 \cdot g_0 + u_1 \cdot g_1 + \ldots + u_{k-1} \cdot g_{k-1}$$

où les $u_i, 0 \leq i \leq k-1$, sont des éléments de l'alphabet$\sum$.

On déduit de cette notation qu'il existe q^k mots de code différents.

1.3.1.2.Cas particulier des codes binaires

Soit $\sum = F_2 = \{0, 1\}$.

Le code par blocs C est dit**linéaire**si et seulement si ses 2^k mots de code forment un sous-espace de dimension k de l'espace vectoriel de tous les vecteurs de longueur n.

Par conséquent C est défini comme étant un sous-espace vectoriel de dimension $k$de F_2^n.

Plus simplement, un code binaire est linéaire si la somme modulo 2 de deux mots de code est aussi un mot de code.

De plus, on a :

$$c = u_0 \cdot g_0 + u_1 \cdot g_1 + \ldots + u_{k-1} \cdot g_{k-1}$$

avec $g_0, g_1, \ldots, g_{k-1}$les k vecteurs formant la base, et$u_i = \{0,1\}$ pour $0 \leq i \leq k-1$.

1.3.2.Matrice génératrice [3][4]

Nous avons vu qu'un mot de code pouvait s'écrire comme la combinaison linéaire des k vecteurs de la base $(g_0, g_1, \ldots, g_{k-1})$.

$$c = u_0 \cdot g_0 + u_1 \cdot g_1 + \ldots + u_{k-1} \cdot g_{k-1}$$

Ces vecteurs $g_i$permettent donc de générer un mot de code. Ils peuvent être regroupés dans une matrice G, $k \times n$, appelée **matrice génératrice**, formée de telle sorte que ses lignes soient composées des vecteurs de la base g_i.

On obtient donc la matrice suivante :

$$\mathrm{G} = \begin{bmatrix} g_o \\ g_1 \\ \square \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \square & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \square & g_{1,n-1} \\ \square & \square & \square & \square \\ g_{k-1,0} & g_{k-1,1} & \square & g_{k-1,n-1} \end{bmatrix}$$

Définition: Une **matrice génératrice** Gd'un code linéaire C est une matrice$k \times n$dont les lignes sont une base de C.

Si on considère le mot à encoder $u \in F2$,avec $u = (u_0, u_1, \ldots, u_{k-1})$,le mot de code est obtenu en calculant :

$$c = u \cdot G = \begin{pmatrix} u_0 & u_1 & \cdots & u_{k-1} \end{pmatrix} \cdot \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix}$$

On a alors $C = \{uG / u \in \Sigma^k\}$.

Définition: On dit que G est dans une **forme standard** si G est de la forme$(\mathrm{I_k}\ \mathrm{P})$ou $(\mathrm{P}\ \mathrm{I_k})$où$\mathrm{I_k}$est la matrice$k \times k$.

Dans ce cas, les k premiers (ou derniers, suivant la façon dont est formée la matrice génératrice) symboles du mot de code sont appelés les **symboles d'information** ("information symboles" en anglais), ils peuvent être choisis de façon arbitraire. Les $n-k$ autres symboles calculés par la fonction d'encodage à partir des symboles d'information sont appelés **symboles de contrôle**("parity-check symboles" en anglais).
Dans le cas d'un code binaire, on dit **bits d'information** et **bits de contrôle**.

1.3.3. Matrice de vérification [3][4]

Une autre matrice très utile est la matrice H, appelée **matrice de vérification**("parity-check matrix" en anglais). On la définit de la façon suivante :
Définition:la matrice de vérification H est une matrice $\mathrm{n-k \times n}$,avec $\mathrm{n-K}$ lignes linéairement indépendantes, telle que chaque valeur de l'espace des lignes de G est orthogonal aux lignes de H, et que tout vecteur orthogonal aux lignes de H est dans l'espace des lignes de G.
Définition: L linéairement indépendantes, telle que chaque vecteur de l'espace des lignes de G est orthogonal aux lignes de H, et que tout vecteur orthogonal aux

lignes de H est dans l'espace des lignes de G.

On peut alors définir un code linéaire de la façon suivante :
Un n-tuple$c$ est un mot du code C généré par G si et seulement si$c \cdot H^T = 0$.

Remarque: Ayant$c \cdot H^T = 0$et sachant que$c = u \cdot G$, avec$c \in C$, on peut écrire:

$$u \cdot G \cdot H^T = 0$$

Ayant$u \neq 0$, on en déduit que :$G \cdot H^T = 0$
Théorème: Si la matrice génératrice G de C est dans la forme standard $(I_k : P)$,alors une matrice de vérification pour C est $H = (-P^T, I_r)$, avec $r = n - k$.

Preuve : Pour cela, on vérifie que $G \cdot H^T = 0$.

$$G \cdot H^T = (I_k \; P) \cdot \left(-P^T \; I_r\right)^T$$

$$G \cdot H^T = \left[\begin{array}{cccc:ccc} 1 & 0 & \cdots & 0 & p_{11} & \cdots & p_{1r} \\ 0 & 1 & \cdots & 0 & p_{21} & \cdots & p_{2r} \\ \vdots & & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & p_{k1} & \cdots & p_{kr} \end{array}\right] \cdot \left[\begin{array}{ccc:cccc} -p_{11} & \cdots & -p_{1r} & 1 & 0 & \cdots & 0 \\ -p_{21} & \cdots & -p_{2r} & 0 & 1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ -p_{k1} & \cdots & -p_{kr} & 0 & 0 & \cdots & 1 \end{array}\right]^T$$

$$G \cdot H^T = \left[\begin{array}{cccc:ccc} 1 & 0 & \cdots & 0 & p_{11} & \cdots & p_{1r} \\ 0 & 1 & \cdots & 0 & p_{21} & \cdots & p_{2r} \\ \vdots & & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & p_{k1} & \cdots & p_{kr} \end{array}\right] \cdot \left[\begin{array}{cccc} -p_{11} & -p_{21} & \cdots & -p_{k1} \\ \vdots & \vdots & & \vdots \\ -p_{1r} & -p_{2r} & \cdots & -p_{kr} \\ \hdashline 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{array}\right]$$

$$G \cdot H^T = \left[\begin{array}{ccc} p_{11} - p_{11} & \cdots & p_{1r} - p_{1r} \\ p_{21} - p_{21} & \cdots & p_{2r} - p_{2r} \\ \vdots & & \vdots \\ p_{k1} - p_{k1} & \cdots & p_{kr} - p_{kr} \end{array}\right] = 0$$

Ainsi, la forme standard de G est $(-P^T \mid I_{n-k})$, avecI_{n-k}lamatrice identité $n-k \times n-k$, et$P$une matrice$n-k \times k$.

1.3.4. Distance minimum [4]

Tout au long de ce paragraphe, nous considèrerons le code linéaire par blocs C ainsi que deux mots de ce code x et y.

1.3.4.1. Définition

Commençons par définir les notions de distance entre deux mots de code et le poids d'un mot.

Définition: La **distance de Hamming** entre deux mots x et y ,notée $d(x,y)$ est le nombre symboles différents entre ces deux mots .En définissant les mots

$x = (x_1, x_2, \ldots, x_n)$ et $y = (y_1, y_2, \ldots, y_n)$,on a:

$$d(x,y) = \{i / x_i \neq y_i\}$$

Exemple : Considérons les mots binaires x = (0 0 1 0 1 1 0) et **y** = (1 0 0 0 1 0 1).

On a alors : $d(x,y) = 4$

Définition: Les **poids** $w(x)$d'un bloc de symboles x est défini par : $w(x) = d(x,0)$où

$$0 = (0,0,\ldots 0)$$

Exemple : Si on reprend les exemples ci-dessus, on a :

$$w(x) = 3 \; et \; w(y) = 3$$

Dans le cas d'un code binaire, si l'on réalise l'addition modulo 2 de deux mots x et y, les bits qui seront égaux à 1 seront placés aux endroits où x et y diffèrent. Ainsi, le poids de Hamming de cette somme sera égale à la distance de Hamming entre x et y, c'est-à-dire:

$$d(x,y) = w\,(x + y)$$

Exemple: Avec les mots x et y définis précédemment, on a :

$$w(x+y) = w(1\,0\,1\,0\,0\,1\,1) = 4 = d(x,y)$$

On peut alors définir la distance minimum du code, notéed_{min} , ou plus simplement d :

Définition: La distance minimum du code C, notée d, est le minimum des distances de Hamming entre les mots du code. On a donc :

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

Notation : Un code C ayant une distance minimale de d est noté $[n,k,d]$. Cette notation signifie que l'on code des mots de k symboles en mots de code de n symboles étant tous distants les uns des autres de d symboles au minimum.

1.3.4.2. Calcul de la distance minimale d'un code

Dans le cas où C est un code linéaire binaire, on a :

$$d_{min} = min\,\{d(x,y) \,/\, x,y \in C, x \neq y\}$$
$$d_{min} = min\,\{w(x+y) \,/\, x,y \in C, x \neq y\}$$
$$d_{min} = min\,\{w(x) \,/\, x \in C, x \neq 0\} = w_{min}$$

On obtient donc le théorème suivant :

Théorème: La distance minimum d'un code linéaire par bloc est égale au poids minimum de ses mots de code non nuls.

Voyons à présent comment calculer la distance minimum d'un code à partir de sa matrice de vérification H.

Théorème : Soit C un code binaire $[n,k]$, et H sa matrice de vérification. Pour chaquemot de code de poids w, il existe w colonnes de H telles que la somme de ces colonnes est

nulle. Inversement, s'il existe w colonnes de H dont la somme est nulle, alors il existe un vecteur de poids w.

Preuve : Pour plus de simplicité, on considère un code binaire C, ayant une matrice de vérification $H = [h_0, h_1, \dots, h_{n-1}]$où $h_i$est la i^{ème} colonne de H.

Soit $r = (r_0, r_1, \dots, r_{n-1})$ un mot reçu au niveau du décodeur de poids w. r a donc w composantes non nulles égales à 1. On a donc $r_{i1} = r_{i2} = \dots = r_{iw} = 1$ avec

$$0 \leq i_w \leq n - 1.$$

r sera un mot de code si$r \cdot H^T = 0$, ce qui équivaut à :

$$r_0 \cdot h_0 + r_1 \cdot h_1 + \dots + r_{n-1} \cdot h_{n-1} = 0$$
$$r_{i1} \cdot h_{i1} + \dots + r_{iw} \cdot h_{iw} = 0$$
$$h_{i1} + \dots + h_{iw} = 0$$

On a alors w colonnes de H dont la somme est nulle.

Pour montrer l'autre partie du théorème, on considère le n-tuple binaire $x = (x_0, x_1, \dots, x_{n-1})$, de poids w, ayant les w composantes x_{i1}, x_{i2}, …, x_{iw}non nulles. On a:

$$x \cdot H^T = x_{i1} \cdot h_{i1} + \dots + x_{iw} \cdot h_{iw} = h_{i1} + \dots + h_{iw}$$

Si $h_{i1} + \dots + h_{iw} = 0$, on en déduit que $x \cdot H^T = 0$.

Donc x est un mot du code C, de poids w.

De ce théorème, on déduit les deux corollaires suivants :

Corollaires:

1. Soient C un code linéaire et H sa matrice de vérification. Si on ne trouve pas $d - 1$ ou moins colonnes telles que leur somme est nulle, alors $d_{min} \geq d$.

2. Soit C un code linéaire et H sa matrice de vérification. Le poids minimum du code w_{min} est le plus petit nombre de colonnes telles que leur
Somme soit nulle.

Ainsi on pourra calculer la distance minimum du code C en trouvant le nombre minimum de colonnes dont la somme est nulle.

1.3.5. Détection et correction des erreurs [1][2]

1.3.5.1. Mécanisme de décodage d'un mot

L'encodage d'un mot u de k symboles s'effectue en calculant le vecteur $c$de longueur$n$tel que :

$$c = u \cdot G$$

Puis on transmet c dans le canal. A la sortie du canal, on reçoit un message r. Le but du décodeur va donc être de tenter de retrouver le message qui a été envoyé à partir de celui qu'il a reçu. Il cherche donc à détecter s'il y a eu une erreur, puis dans un deuxième temps, il cherche à corriger cette ou ces erreurs.

Une méthode pour détecter la présence d'erreurs et de calculer le syndrome $s = (s_0, s_1, \dots, s_{n-1})$, défini par :

$$s = r \cdot H^T$$

Si le vecteur s obtenu est égal au vecteur nul, étant donné que le produit d'un des mots du code Cavec la transposée de la matrice de vérification est nulle, on peut déduire que r est un mot appartenant au code C. Le récepteur va alors considérer qu'il n'y a pas d'erreur, et qu'il s'agit du message envoyé.

Si le syndrome est différent du vecteur nul, cela signifie que le mot reçu rn'est pas un mot appartenant au code C. Par conséquent, on est sûr qu'une erreur est apparue lors de la transmission. Le but du décodeur va donc être de trouver où se situe cette erreur et de la corriger.

Voyons plus précisément le cas des codes binaires

On pose $e = (e_0, e_1, \dots, e_{n-1})$, le vecteur d'erreur, représentant les erreurs ayant été introduite par le canal. Ce vecteur a des zéros partout sauf aux positions où une erreur s'est produite.

On a alors :

$$r = c + e$$

Le syndrome **s** est alors égal à :

$$s = (c + e) \cdot H^T$$

$$s = c \cdot H^T + e \cdot H^T$$

Or$c \cdot H^T = 0$, donc on a:

$$s = e \cdot H^T$$

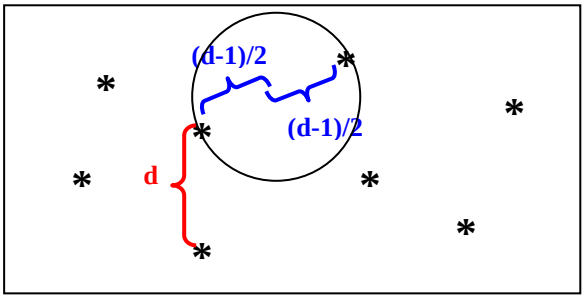
On peut donc calculer les syndromes possibles à partir des vecteurs d'erreurs et constituer une table associant à chaque erreur pouvant apparaître dans le canal le syndrome que l'on obtiendrait. Ainsi lors du décodage, on pourra comparer s avec les entrées de la table et retrouver le vecteur erreur. Lorsque e est connu, on retrouve facilement c en calculant :

$$c = r + e$$

1.3.5.2. Distance minimale et capacité de détection et correction d'erreurs

Voyons à présent comment la distance minimale nous permet de savoir combien d'erreurs pourront être détectées et corrigées.

Afin de comprendre le principe, commençons par réaliser une illustration des mots d'un code binaire C. Les petites étoiles représentent les 2^k mots du code pris dans un ensemble de 2^n éléments.



Un mot n'appartient pas au code C si celui-ci n'est pas un des 2^k mots représentés. Instinctivement, on déduit de ce schéma qu'on pourra détecter qu'un mot n'appartient pas à C s'il a une distance avec un mot de code non nulle et strictement inférieure à d.

De plus, en admettant que l'on utilise une règle de maximum de vraisemblance pour le décodage, on pourra corriger correctement les erreurs et retrouver le mot originel si la distance entre le mot reçu et le mot envoyé est inférieure à $\left\lfloor \frac{d-1}{2} \right\rfloor$.

On obtient donc la propriété suivante :

Propriété: Un code ayant une distance minimale de d peut détecter $(d-1)$ erreurs et peut corriger $\left\lfloor \frac{d-1}{2} \right\rfloor$ erreurs.

Ainsi, si l'on pose $d = 2e + 1$, on pourra détecter $2e$ erreurs et on pourra corriger e erreurs.

1.3.6. Bornes caractérisant les codes [2][3]

Nous avons vu que l'on caractérise un code par la longueur de ses mots de code n, son nombre de mots M (ou la longueur des blocs de message k), et la distance minimum $d$entre les mots de code. On dit alors qu'il s'agit d'un code (n, M, d).

Intuitivement, on comprend que les grandeurs M et d "jouent" l'une contre l'autre. En effet, si on a un très grand nombre de mots, ceux-ci auront une distance faible. Alors que si on a M petit, alors la distance entre les mots pourra être plus grande, permettant ainsi de détecter et de corriger plus d'erreurs.

Pour caractériser cette relation, on introduit la notation $A_q(n, d)$ qui représente le nombre de mots M maximum tels qu'il existe un code q-ary(n, M, d).

Il va donc être nécessaire de trouver un compromis entre ces deux valeurs, afin d'avoir un nombre de mots suffisants et une distance minimale suffisamment grande pour pouvoir détecter et corriger un certain nombre d'erreurs. Pour cela, il existe des bornes qui caractérisent les grandeurs M et d. Nous allons étudier dans ce paragraphe deux bornes : le **Singleton Bound** et la **SphèrePackingBound**, appelée également **borne deHamming**.

1.3.6.1. Singleton Bound

Considérons un code C $[n, k, d]$**.** Cette borne permet de trouver une borne maximale sur la distance minimale d par rapport aux valeurs n et k.

Théorème: Soit un code $[n, k, d]$.On a l'inégalité suivante: $d \leq n - k + 1$

Preuve: Soient C le code $[n, k, d]q$**,**et x un mot de ce code.

Considérons le mot x' formé à partir de x auquel on enlèverait les $(d - 1)$ derniers symboles.

On obtient ainsi un nouveau code C' de taille $n' = n - (d - 1) = n - d + 1$.

Comme C avait une distance minimale de d, tous les mots de C avaient une distance entre eux supérieure ou égale à d, donc si on enlève $(d - 1)$ symboles à ces mots, on obtient des mots qui sont tous différents d'au moins un symbole. Ainsi, tous les mots de C' sont différents les uns des autres, et on en déduit que Cet C' ont le même nombre de mots de code.

On peut donc écrire que :

$$q^k = q^{k'}$$

ce qui implique :

$$k = k'.$$

C'est donc un code $[n', k]$ avec $n' = n - d + 1$. On en déduit qu'il existe un code de dimension k et de distance $d \geq 1$ et de longueur n.

Or, on sait que la longueur des mots de code est toujours supérieure ou égale à la dimension du code, et donc on a la relation suivante :

$$n' \geq k \Leftrightarrow n - d + 1 \geq k$$

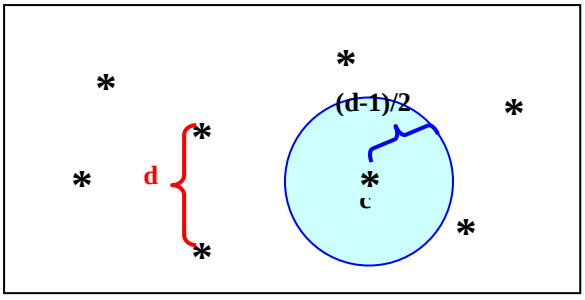
$$d \leq n - k + 1$$

1.3.6.2. SpherePackingBound

Il s'agit d'une borne sur la dimension du code k.

Pour commencer, définissons une "balle" de rayon $r = \lfloor (d-1)/2 \rfloor$ autour de chaque mot de code . Celle-ci englobe tous les éléments de l'ensemble F_q^n qui se trouvent à une distance inférieure ou égale à r d'un mot de code.

Illustrons ce concept pour le cas d'un code binaire, à l'aide d'un schéma. Le cercle bleu représente la balle de rayon r. Pour des raisons de clarté, nous ne représenterons cette balle que pour un seul mot de code, l'idée étant la même pour tous les autres mots de codes.



On s'intéresse alors au nombre de mots se situant dans cette balle, c'est-à-dire au nombre de mots se trouvant à une distance inférieure ou égale à r d'un mot de code **c** donné. On note :

$$S_r(c) = \{x \in F_2^n / d(x,c) \leq r\} \text{ avec } r = \lfloor (d-1)/2 \rfloor$$

On calcule le cardinal de $S_r(c)$. Pour cela, on compte le nombre de mots qui diffèrent d'un nombre de symbole de 0 à $r$du bloc c. On choisit donc i places ($0 \leq i \leq$

r)dans le mot que l'on multiplie par le nombre de possibilités de changements d'un symbole. On obtient alors la formule suivante :

$$\left|S_r(C)\right| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i$$

Pour finir, étant donné que les balles sont disjointes, le nombre de mots du code multipliés par le nombre d'éléments contenus dans cette balle doit donner un nombre inférieur ou égal au nombre total d'éléments de F_q^n. On obtient donc la borne suivante :

$$q^k \cdot \left|S_{\frac{d-1}{2}}\right| \leq q^n$$

Théorème: Soit un code linéaire $[n,\ k, d]$défini sur un alphabet à q éléments. La borde de Hamming du code est relation : $q^k \cdot \sum_{i=0}^{\left[\frac{d-1}{2}\right]} \binom{n}{i} (q-1)^i \leq q^n$

Dans le cas d'un code binaire, on a la relation :

$$2^k \cdot \sum_{i=0}^{\left\lfloor \frac{d-1}{2} \right\rfloor} \binom{n}{i} \leq 2^n$$

Cette borne nous permet de définir la notion de code parfait.

Définition: Un code est dit **parfait** la borne de Hamming est égalité, c'est-à-dire si:

$$q^k \cdot \sum_{i=0}^{\left[\frac{d-1}{2}\right]} \binom{n}{i} (q-1)^i = q^n$$

Plus simplement, un code est parfait si tous les éléments de F_q^n sont inclus dans les balles de rayon r autour des mots du code.

1.3.7. Exemple : les codes de Hamming [1]

Ces codes ont été inventés par Richard Hamming, et sont utilisés dans le domaine des "digital communications" et des systèmes de sauvegarde de données.

Le codage d'un mot de k bits s'effectue en ajoutant m bits de façon à obtenir des mots de codes de n bits. On a donc $m = n - k$. Ces bits sont calculés à l'aide de la matrice génératrice. Le code a une distance minimale de 3, il peut donc détecter 2 erreurs et en corriger 1.

1.3.7.1. Cas général des q-ary codes de Hamming

Voyons tout d'abord comment on forme de façon générale les codes de Hamming définis sur un alphabet contenant q éléments.

On part du "projective espace" $p^n(F_q)$ de dimension n défini sur F_q.Cetensembleest l'ensemble des classes d'équivalence de $F_q^{n+1}\backslash\{0\}$,où deux vecteurs x et y sont dit équivalentssi$y = \lambda x$, avec$\lambda \in F_q^*$. Par conséquent, dans une classe d'équivalence, tous les éléments dont des multiples les uns des autres. Le principe est comparable aux classes d'équivalence définies sur Z/mZ.

1. Montrons que le nombre d'éléments de$p^n(F_q)$ est$N = \frac{q^{n+1}-1}{q-1}$

Le nombre d'éléments de$F_q^{n+1}\backslash\{0\}$est égal au nombre de vecteurs de longueur $n+1$ constitués d'éléments pris dans un ensemble de q éléments auquel on enlève le vecteur nul. On a alors :

$$|F_q^{n+1}\backslash\{0\}| = q^{n+1} - 1$$

Pour trouver le nombre d'élément de $p^n(F_q)$, on applique le principe suivant :

On regroupe tous les vecteurs de $F_q^{n+1}\backslash\{0\}$ en classes d'équivalence et on cherche le nombre d'éléments contenus dans une classe d'équivalence. De là, on pourra déduire le nombre de classe d'équivalence, qui est le nombre recherché.

Sachant que $y = \lambda \cdot x$, si l'on considère un vecteur x, on obtient tous les autres vecteurs de la classe représentée par x en multipliant x par un $\lambda \in F_q^*$. Ona $q-1$ possibilités pour choisir λ(F_q^*contient $q-1$ éléments car on y a exclu l'élément 0), ce qui implique que l'on a $q-1$ éléments dans une classe d'équivalence.

Par conséquent, on en déduit que :

$$|p^n(F_q)| = N = \frac{q^{n+1}-1}{q-1}$$

On considère maintenant la matrice H possédant $n+1$ lignes et N colonnes, où les colonnes de H sont les représentants de chaque classe d'équivalence qui sont les éléments de $p^n(F_q)$.

2. Montrons que le rang de H est égal à $n+1$

Ceci équivaut à montrer qu'il existe $n+1$ colonnes linéairement indépendantes. Prenons les vecteurs $e_1, e_2, \ldots, e_{n+1}$, définis de la façon suivante :

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \square \\ 0 \end{pmatrix},\ e_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \square \\ 0 \end{pmatrix},\ \ldots.,\ e_{n+1} = \begin{pmatrix} 0 \\ 0 \\ \square \\ 0 \\ 1 \end{pmatrix}$$

On remarque que les e_i sont tous dans des classes d'équivalence différentes, et que tout autre vecteur de $n+1$ composantes peut être défini par une combinaison linéaire de $e_1, e_2, \ldots., e_{n+1}$. Par conséquent, $(e_1, e_2, \ldots., e_{n+1})$ peut être considéré comme une base de l'espace des colonnes de H. Le rang de H est donc de $n+1$.

3. Montrons que la distance minimale du code obtenu est égale à 3

Ceci revient à montrer que le poids minimum des mots du code est de 3.

On sait que si x est un mot du code C défini par la matrice de vérification H, alors :

$$x \cdot H^T = 0$$

Supposons que le poids minimum est strictement inférieur à 3, prenons le cas où il serait égal à 2.

Il existerait alors un mot $x = (x_0, x_1, ..., x_{N-1})$ de poids égal à 2, c'est-à-dire que les x_i seraient tous nuls sauf aux deux positions i_α et i_β où on aurait $x_{i_\alpha} = \alpha,\ x_{i_\beta} = \beta$.

On aurait alors :

$$x \cdot H^T = 0 \quad \Leftrightarrow \quad \alpha \cdot col\, i_\alpha + \beta \cdot col\, i_\beta$$

où $col\, i_\alpha$ et $col\, i_\beta$ sont les colonnes de H^T (les lignes de H) se situant aux positions i_α et i_β

$$x \cdot H^T = 0 \quad \Leftrightarrow \quad col\, i_\alpha = -\frac{\beta}{\alpha} col\, i_\beta$$

De fait, les colonnes i_α et i_β seraient dans la même classe d'équivalence. Au moins une de ces deux colonnes n'est donc pas un représentant de cette classe. Ceci remet en cause la définition de la matrice, car celle-ci a été construite de telle sorte que ses colonnes soient les représentants des classes d'équivalence. Il y a donc une contradiction. On en déduit que le poids minimum d'un mot doit être de 3, et donc on a $d_{min} = 3$.

La matrice $H, n+1 \times N$, décrite ci-dessus définie un q-ary code de Hamming. La matrice génératrice G de ce code est une matrice $N - n - 1 \times N$.

On code donc des mots de $N - n - 1$ symboles en mots de code de N symboles, avec $N = \frac{q^{n+1} - 1}{q - 1}$.

4. Montrons que ce code est parfait

On rappelle que le code a une dimension égale à $k = N - n - 1$, et une longueur de N, avec N défini comme auparavant, et une distance minimale $d_{min} = 3$.

Pour montrer que le code est parfait, il nous faut montrer que la borne de Hamming est une égalité. On montre donc que :

$$q^k \cdot \sum_{i=0}^{e} \binom{N}{i} (q-1)^i = q^N \quad avec\ e = \left\lfloor \frac{d-1}{2} \right\rfloor et\ k = N - (n+1)$$

$$q^{N-n-1} \cdot \sum_{i=0}^{1} \binom{N}{i} (q-1)^i = q^{N-n-1} \cdot [1 + N(q-1)]$$

$$= q^{N-n-1} \cdot [1 + q^{n+1} - 1]$$

$$= q^{N-n-1+n+1} = q^N$$

On a en déduit que le code de Hamming défini au point précédent est un code parfait.

1.3.7.2. Cas particulier des codes de Hamming binaires

L'alphabet utilisé est $\{0,1\}$, donc $q = 2$. Le code C est défini de la façon suivante :

Sa longueur est: $N = \frac{q^{n+1}-1}{q-1} = 2^{n+1} - 1 = 2^k - 1$ avec $k = n + 1$.

Sa dimension est: $N - n - 1 = 2^k - k - 1$.

On a donc un code $[2^k - 1, 2^k - k - 1, 3]$. On code des mots de $2^k - k - 1$ bits en mots de $2^k - 1$ bits en leur ajoutant k bits.

Etudions plus précisément l'exemple du code **[7, 4, 3]**. Dans ce code, on encode des mots de 4 bits en ajoutant 3 bits de contrôle. Nous donnons les matrices qui définissent ce code, puis nous illustrerons le fonctionnement de ce code en donnant l'exemple de l'encodage d'un mot, du décodage et de la correction d'un message reçu avec une erreur.

1.Matrices de vérification et génératrice en forme systématique

$$H = \begin{bmatrix} I_3 & Q \end{bmatrix} = \left[\begin{array}{ccc:cccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$$

$$G = \begin{bmatrix} Q^T & I_4 \end{bmatrix} = \left[\begin{array}{cccc:cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

2.Exemple de codage d'un mot u

Soit le message à transmettre : u = (1 1 0 1).

On envoie le mot de code x défini par $x = u \cdot G$:

$$x = \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix} \cdot \left[\begin{array}{ccc:cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

On remarque que comme G est dans une forme systématique, les quatre derniers bits du mot x sont les bits d'information non modifiés. Les trois premiers bits de x sont donc les bits de contrôle.

3.Exemple du décodage d'un mot reçu r

Imaginons qu'à la sortie du canal, on reçoive le mot $r =$ (1 0 0 1 1 0 1) comportant une erreur au premier bit.

Pour décoder ce mot, on commence par calculer le syndrome s qui y est associé en le multipliant par la matrice de vérification H. Si le résultat est nul, ce mot est considéré comme ne comportant pas d'erreur. Si le résultat est non nul, on en déduit qu'une erreur a été introduite dans le canal. Dans le cas des Hamming codes, le vecteur obtenu se retrouve dans les colonnes de H, et la position de la colonne nous indique la position du bit erroné dans le mot reçu.

On calcule donc le syndrome :

$$s = r \cdot H^T$$

$$s = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \cdot \left[\begin{array}{ccc:cccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]^T$$

$$s = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

Ayant $s \neq 0$, on en déduit qu'une erreur a été introduite par le canal. De plus, la colonne $(1\ 0\ 0)$ est en première position dans H, ce qui implique que l'erreur se trouve au premier bit de r.

On peut donc corriger le mot reçu en changeant le bit erroné. On obtient alors :

$$r' = (0\ 0\ 0\ 1\ 1\ 0\ 1)$$

Le message envoyé étant contenu dans les quatre derniers bits, on retrouve simplement le mot envoyé qui était bien $m = (1\ 1\ 0\ 1)$.

1.4. Conclusion

On a essayé dans ce chapitre d'exposer la théorie des codes correcteurs d'erreurs linéaires en général qui fait la base des chapitres suivant. On a vu que ces codes correcteurs utilisant des mathématiques avancées. Cependant, pour trouver des codes correcteurs encore plus performants est donc une nécessité et les mathématiques jouent un grand rôle dans cette recherche.

Chapitre 2

Les codes de Reed-Solomon RS

Le code de Reed-Solomon est un code correcteur, dit "code parfait" inventé par les mathématiciens Irving S.Reed et Gustave Solomon, qui est très utilisé et permet de corriger des erreurs dûes à la transition imparfaite d'un message.

Ce code de Reed-Solomon est basé sur un principe mathématique qui a pour objectif de construire un polynôme à partir des symboles à transmettre (information source) et de le suréchantillonner. C'est à dire d'apporter beaucoup plus d'informations à ce polynôme. Le résultat est alors envoyé, au lieu des symboles originaux. La redondance de ce suréchantillonnage permet au récepteur du message encodé de reconstruire le polynôme source même s'il y a eu des erreurs pendant la transmission.

2.1. Théories, définitions, et propriétés [2][3]

- **Information (message) :** Élément de connaissance susceptible d'être codé pour être conservé, traité ou communiqué.
- **Transmission :** Envoie d'une information d'un point à un autre ; communiquer.– Canal de communication : Voie par laquelle voyage l'information : moyen de communication.
- **Émetteur :** Station d'émission de signaux porteurs d'information.
- **Récepteur :** Appareil recevant un signal et le transformant en information.
- **Encodage :** Transformation d'un message en vue d'une transmission de qualité.

- **Code :** Un langage L sur un alphabet A est un code seulement s'il n'existe pasdeux factorisations différentes dans l'ensemble des mots de A avec des mots de L.
- **Codage :** Soit L et M deux langages ; un codage c de L dans M est un morphismeinjectif. En d'autres termes, c'est une fonction entre les mots de L et ceux M,où à tout mot de L est associé un unique mot de M et tel que le codage de laconcaténée soit égale à la concaténée des codages.
- **Décodage :** Traduire, comprendre un message, une information encodée.
- **Détection d'erreur :** Opération permettant de déterminer les erreurs qu'auraitpu subir une information.
- **Correction d'erreur :** Retrait des erreurs détectées dans l'information.
- **Théorie de l'information :** étude du processus de communication, des diversfacteurs qui régissent la transmission et la réception de signaux.

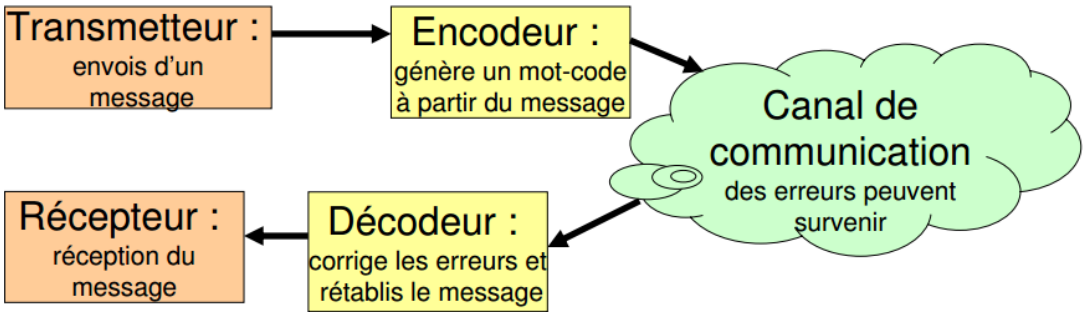


Fig. 2.1 – Modèle de processus de communication

2.2 Modèle de communication avec codage [2][4]

2.2.1 Processus de communication

Le modèle du processus de communication tel que représenté à la figure 2.1 est unevue simplifié d'un modèle globale pouvant atteindre un haut niveau de complexité. Ilest simplifié dans le sens où nous y avons uniquement représenté le strict nécessaireconcernant les communications avec encodage du message à transmettre, puis décodagedu message reçu.

Canal de communication (Canal symétrique binaire)

Le modèle du canal symétrique binaire est une modélisation d'un canal où l'information y circulant sont des bits, ayant une probabilité p d'être mal interprété à laréception. Donc, une probabilité p que le bit en réception, soit l'inverse du bit émis.

2.2.2 Théorème de Shannon

Très brièvement, C.E. Shannon aurait établis, avec son prédécesseur scientifique,R.A. Fisher, que plus une informationest probable, moins elle contiendra d'information.Évidemment, à l'inverse, moins une observation sera probable, plus elle comporterad'information. L'information étant définit dans ce cas comme étant une événement possible parmi toutes lespossibilités d'évènement dans un contexte précis.

2.2.3 Problème principal du codage

Le but du codage est de trouver le code ayant le plus grand choix de mots codesde longueur déterminé avec une distance minimal donné maximisé pour un alphabetdonnée $GF(q^m)$. Notons que dans notre cas, les codes Reed-Solomon, nous avons descodes MDS (Maximum Distance Separable, i.e. : des codes parfait, ou des codes maximisant la distanceentre les mots codes et ainsi abaissant la probabilité de confusion lors de décision lors de décodage.), soit des codes où la distance minimale est maximisé. C'est d'ailleurs pourcela, entre autre, que ces codes sont appréciés.

2.3 Généralités sur les codes de Reed-Solomon

2.3.1 Avantages des codes de Reed-Solomon [5][6]

- Pour des corps de Galois $GF(q)$, lorsque l'on a besoin de coder des messages delongueur inférieur à q, l'encodage de RS est facilement utilisable. Puisque ce sontdes codes MDS, la distance minimum est maximisé.
- Ils peuvent être combiné ou ajouté à autres codes afin de réaliser des codes pluseefficaces.
- L'encodage est assez facile.
- Ils sont très efficace pour la correction d'erreurs consécutives, qu'ils soient utilisésseuls ou en conjonction avec d'autres codes.Bien entendu, cela est valide pourles cas où le nombre d'erreur demeure en dessous de la capacité de correction ducode employé.
- Leurs algorithmes de décodages sont très développés.
- Pour des corps de Galois $GF(q = 2^r)$, on peut les représenter par des codesbinaires puisque chacun des éléments de ce corps de Galois peuvent être représentépar une séquence binaire de longueur r.

2.3.2 Propriétés [7]

Nous poserons α tel qu'étant un élément primitif dans $GF\,(2^m)$. Notez que l'ontravaillera avec des codes RS ayant leurs symboles de $GF\,(2^m)$.

Pour tout entier positif t, tel que $t \leq 2m-1$, il existe un code RS en mesure decorriger t symboles, pour des symboles parmi $GF\,(2^m)$, et ayant les paramètres suivant :

$$\begin{aligned} n &= 2^m - 1 \\ n - k &= 2t \\ k &= 2^m - 1 - 2t \\ d_{min} &= 2t + 1 = n - k + 1 \end{aligned} \tag{2.1}$$

où n correspond au nombre de symbole d'un mot code, k correspond au nombre desymbole du segment de parité, et enfin, où d_{min} correspond à la distance minimale.

2.3.3 Polynômes générateurs

Polynôme générateur de mot code [8]

Générateur de mots code sous forme algébrique $(g(x))$, sous représentation polynômiale, qui une fois multiplié par la représentation polynômiale du message à transmettre$(m(x))$, génèrera le mot code correspondant au message $(c(x))$. Ainsi nous avons doncla formulation suivante :

$$c(x) = g(x)m(x) \tag{2.2}$$

Polynôme générateur pour RS

L'équation (2.3) correspond au polynôme générateur de Reed-Solomon ainsi qu'à la manière de le construire. En effet, il n'y a qu'à prendre les produits des binômes pour$(x + \alpha^i)$, pour les i allant de 1 jusqu'à $2t$. On note que dans cette construction, les α^icorrespondront aux racines du polynôme $g(x)$, et où les $g_i$sont des symboles membrede $GF\,(2^m)$.

$$g(x) = (x+\alpha^1)(x+\alpha^2)\cdots(x+\alpha^{2t}) = g_0 + g_1x + g_2x^2 + \cdots + g_{2t-1}x^{2t-1} + x^{2t} \tag{2.3}$$

2.4 Encodage de codes de Reed-Solomon [7][8]

Pour un message $m(x) = m_0 + m_1x + \cdots + m_{k-1}x^{k-1}$où nous avons la représentationpolynômiale d'un message à encoder. Nous avons les $m_i \in GF\,(2^m)$ ainsi que $k = n - 2t$.

Nous devons d'abords multiplier $m(x)$, le message à transmettre, par x^{2t}procédantainsi à un décalage. Ce décalage permettra d'insérer le polynôme de parité.Une foisque l'on a en main $x^{2t}m(x)$, nous procédons à sa division par le polynôme générateurde mots codes $g(x)$ trouvé préalablement.

Nous avons donc maintenant l'équation (2.4), où $b(x)$ représente le reste.

$$x^{2t}m(x) = a(x)g(x) + b(x)$$

$$b(x) = b_0 + b_1x + b_2x^2 + \cdots + b_{2t-1}x^{2t-1} \quad (2.4)$$

Ce reste que l'on vient de trouver représente en fait le polynôme de parité recherché.

Pour le message $m(x)$, le polynôme de parité est donc représenté par $b(x)$, et le messagecodé, soit le mot de code, $c(x)$ correspondra à :

$$b(x) + x^{2t-1}m(x) \quad (2.5)$$

2.4.1 Circuit d'un encodeur cyclique non binaire

Il vous est possible de consulter [9] afin d'y trouver en page 172, à la figure 6.13,un schéma d'un circuit d'encodage pour un code cyclique non binaire. Des explicationsdétaillés vous sont présentés dans les pages aux alentours de la figure selon le conceptque vous souhaiteriez éclairer.

2.5Conclusion

Nous avons vu dans ce chapitre un type important des codes correcteurs linéaires à savoir les codes de Reed-Solomon. Les définitions et les propriétés principales de ces codes sont exposées ici.

Le champs d'application de ces codes est très vaste car les codes de Reed-Solomon sont utilisées pour corriger des erreurs dans de nombreux systèmes, y compris: les périphériques de stockage (y compris les bandes, disques compacts, DVD, codes à barres, etc), les communications sans fil ou mobiles (y compris les téléphones cellulaires, des liens micro-ondes, etc), les communications par satellite, la télévision numérique / DVB et les Modems haut débit comme l'ADSL, xDSL, etc.

L'opération du codage des codes Reed-Solomon est présentée ici. Donc, il reste l'opération inverse qui est le décodage. Cette dernière fait l'objet du chapitre suivant.

Chapitre 3

Décodage de codes de Reed-Solomon

L'objectif du décodage est de retrouver le mot de code émisà partirdu mot de code reçu,potentiellement erroné. Pour cela, avec des codes deReed-Solomon, on peut procéder de plusieurs manières, la plus évidente estde retrouver le polynôme correspondant au mot de code émis, maisil est aussi possible de seulement déterminer les emplacements des symboleserronés du mot de code reçu, puis de retrouver les bons symboles qui auraientdû se trouver à ces emplacements.

Toutes les versions d'algorithme principal de décodage des codes de Reed-Solomonsont similaire à l'énumération suivante. Ce sera suivis d'une version un peu plus mathématique.

1. Calculer les $2t$ syndromes s_0, s_1,⋯, s_{2t-1}où l'on a que less$_i = r(\alpha_i)$, où $r(x)$ estlepolynôme reçu. Notez que $s_0 = 1$, et que le polynôme des syndromes $s(x) = s_0 + s_1x^1 + \cdots + s_{2t-1}x^{2t-1}$.
2. Le rang de la matrice P nous donne le degré du polynôme localisateur d'erreur$\lambda(x)$.La matrice P étant la matrice des coefficients lorsque l'équation du polynôme dessyndromes est vue matriciellement en relation avec les coefficients λ_i du $\lambda(x)$.
3. Évaluer le polynôme localisateur d'erreur à partir des informations en main.
4. Évaluer les racines du polynôme localisateur d'erreur. Ces racines, lorsque l'on aleurréciproque, nous donnent les positions des erreurs.

5. La matrice des positions d'erreurs par la matrice de leur intensité nous donneralessyndromes, il nous faudra donc résoudre ce système matriciel pour obtenirl'intensité deserreurs.

Considérant le code $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$, où $c_i \in GF(2^m)$ que l'on envoidans un canal de communication, puis le code reçu $r(x) = r_0 + r_1x + \cdots + r_{n-1}x^{n-1}$, où$r_i \in GF(2^m)$.Le polynôme d'erreur est donc alors $e(x) = r(x) - c(x)$ où $e(x) = e_0 + e_1x + \cdots + e_{n-1}x^{n-1}$où $e_i = r_i - c_i \in GF(2^m)$.

Faisant l'hypothèse que $r(x)$ comporte l erreurs, le polynôme $e(x)$ comportera leserreurs aux locations $x^{j_1}, x^{j_2}, \cdots, x^{j_l}$, et donc $e(x) = e_{j_1}x^{j_1}, e_{j_2}x^{j_2}, \cdots, e_{j_l}x^{j_l}$.

Les positions des locations d'erreurs sont alors $Z_{j_i} = \alpha^{j_i}$pour des i allant de 1 àl. Les valeurs de ces erreurs sont de e_{j_i}, encore une fois pour des i allant de 1 à l. Lescodes de Reed-Solomon ont donc en plus des procédures des codes BCH à déterminerla valeur des erreurs trouvées.

S'il y a p erreurs d'effacement et q erreurs dans le polynôme reçu $r(x)$, alors undécodeur $RS(n, k)$ pourra corriger ces erreurs si on a $2q + p \leq d - 1 = n - k$. Pourun polynôme d'erreurs d'effacement $e^*(x)$, nous auront donc en réception le polynômer$(x) = c(x) + e(x) + e^*(x)$.

3.1 Calcul des syndromes

Pour un message reçu $r(x)$, où $r(x) = r_0 + r_1x^1 + \cdots + r_{n-1}x^{n-1}$, sachant quele polynôme générateur, tel que vu plus haut, comporte les racines $\alpha^1, \alpha^2, \cdots, \alpha^{2t}$, etpuisque $c(\alpha^i) = m(\alpha^i)g(\alpha^i)$ pour des i allant de 1 à $2t$, nous avons la relation suivante [5][6] :

$$r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i);$$

$$e(\alpha^i) = \sum_{j=0}^{n-1} ej\ \alpha ij \qquad (2.6)$$

Circuit d'un évaluateur de syndromes pour codes RS

Il vous est possible de consulter [9] afin d'y trouver en page 174 à la figure 6.14,deux schémas servant à la structure d'un circuit évaluateur de syndromes pour codesde Reed-Solomon. Des explications détaillées vous sont présentés dans les pages auxalentours selon le concept que vous souhaiteriez éclairer.

3.2 Évaluation du polynôme de locations d'erreurs

Avant d'aller dans la résolution par les algorithmes possibles décrits ci-dessous, il fautévaluer le nombre d'erreurs détectés par le code. Ceci peut être fait par l'évaluationdu rang de la matrice des coefficients du système d'équation permettant l'obtentiondu polynôme de locations d'erreurs. Cette étape n'est pas toujours présenté dans lalittérature, ce qui occasionne bien des maux.

3.2.1 Algorithme de Peterson (RS binaire)

Tel que mentionné lors de la présentation audio-visuelle, l'algorithme de Petersonpeut servir à trouver le polynôme locateur d'erreur. Toutefois, il est efficace que pourdes petit nombre d'erreurs pour $r(x)$. La complexité de l'algorithme se trouverait à$O(n^3)$.

Il vous est possible de consulter le volume de T.K. Moon, [10] pour en découvrirles détails. Il se trouverait également un exemple pour un cas simple dans les notesde cours [11]. Rien de spécifique par rapport à ces présentations ne pourrait vous êtrerévélé en supplément ici.

3.2.2 Algorithme de Berkelamp-Massey

Connaissant déjà l'algorithme présenté par le volume de [10] et ayant constaté lorsde la présentation audiovisuelle que le comité de correction n'apprécie par sa présentation, l'auteur a essayé de trouver une autre méthodologie pour sa représentation,cette dernière sera tiré de la construction des LFSR, (Linear Feedback Shift Register),communément désigné par registre à décalage (RAD), dans la langue de Molière. Onmettra à profit sa représentation polynômiale, et bien évidemment on notera égalementque ce type de vision de l'algorithme est préféré dans les systèmes à implémentationmatérielle.

$Pour\ i\ =\ 0{:}n$
$d = P(X)\ AND\ s$ % bit à bit
$si\ d\ !=\ 0\ \ alors$

$T(X) = P(X)$

$P(X) = P(X)\ XOR\ Q(X)X\hat{}\hat{}\{i-m\}$

$si\ 2\backslash lambda\ <=\ i\ alors$

$\backslash lambda = i + 1 - \backslash lambda$

$m = i$

$Q(X) = T(X)$

3.2.3 Algorithme de Sugiyama

L'algorithme de Sugiyama, que vous pouvez retrouver à [12], ou, aux pages 182-183de [10] s'est inspiré de l'algorithme Euclidien que l'on retrouve à [12], ainsi qu'en algorithme5.1 de [10]. C'est simplement une représentation algorithmique d'une résolutionmatricielle tel que demandé par le point 2 et 3 des étapes d'un décodage de code deReed-Solomon. La version matricielle, étant donnée les capacités opérationnelles matricielles de Matlab serait à prévilégier avec cet outil de calculs mathématique. Noterqu'un algorithme de Sugiyama est également connu en théorie des graphs, dû à la différence de notations entre les différents domaines de la science, l'auteur n'a pu être enmesure de déterminer, avec preuve, dans le court laps de temps si c'est effectivement lemême algorithme, mais c'est fort probable selon lui.

3.3 Évaluation des racines du polynôme de locations d'erreurs

L'évaluation des racines du polynôme de locations d'erreurs pour un code RS àgrand nombre d'erreur corrigeable possible, c'est à dire lorsque le polynôme de locationd'erreur est de degré élevé, se fait préférablement avec la *Chien Search*. Elle consistesimplement à prendre notre polynôme de location d'erreur $\lambda(x)$ et d'essayé des $x = \alpha^i$pour des i allant de 1 à $q^m - 1$. Lorsque λ(αi)=0, c'est simplement que αicorrespondà une de nos racines. Plus de détails sont disponibles en page 249 de [23].

3.4 Poids des erreurs détectées (Algo de Forney)

L'algorithme de Forner tel que décrit par [10] en page 263, requière toutes lesinformations trouvés à présent, du polynôme localisateur d'erreur ($\lambda(x)$), au polynôme

dessyndromes ($S(x)$), en passant par les racines et leurs réciproques pour $\lambda(x)$ ainsi que ladérivé sous Galois de ce dernier. L'algorithme se déroule grossièrement en les deuxétapesreprésentés par les deux équations suivante dont la première permettant de trouver lepolynôme évaluateur d'erreur, puis la seconde de déterminer les poids d'erreurs.

$$\Omega(x) = S(x)\Lambda(x)(mod x^{2t})$$

$$e_{i_k} = -\frac{\Omega(X_k^{-1})}{\Lambda'(X_k^{-1})} \tag{2.7}$$

3.5 Conclusion

Pour n'est pas être exhaustif, on a essayé dans ce chapitre de présenter les algorithmes de décodage des codes Reed-solomon les plus utilisables dans le domaine du communication.

La découverte d'un algorithme de décodage des codes de Reed-Solomon avecun tel pouvoir de correction a relancél'intérêt pratique de ces codes, puisque leurperformances'en trouve grandementaméliorée. Si la question se pose de savoir en pratique comment choisir unesolution dans la liste des solutions proposées par l'algorithme, force est deconstater que l'algorithme retourne une unique solution avec une probabilitéquasi égale à un, ce qui est fort intéressant. Reste maintenant l'implémentation de ces algorithmes pour êtreutilisables et performants dans les applications.

L'implémentation de ces algorithmes et son amélioration font un bon axe de travail dans le domaine de codage de l'information.

Conclusion générale

Selon le thème abordé, notre travail concernant les codes de Reed-Solomon et le problème de leurs décodage, a englobé trois chapitres :le premier chapitre expose la théorie et les notions des codes correcteurs linéaires, le chapitre suivant traite la théorie des codes de Reed-Solomonet le troisième chapitre présente les algorithmes de décodage, les plus utilisés, des tels codes.

Le chapitre un a pour but de dégager les fondements théoriques des codes correcteurs linéaires. Un code correcteur est une technique de codage basée sur la redondance. Elle est destinée à corriger les erreurs de transmission d'une information (appelée message) sur une voie de communication peu fiable.La théorie des codes correcteurs ne se limite pas qu'aux communications classiques (radio, câble coaxial, fibre optique, etc.) mais également aux supports pour le stockage comme les disques compacts, la mémoire RAM et d'autres applications où l'intégrité des données est importante.

Le deuxième chapitre est consacré à un type important des codes correcteurs linéaires à savoir les codes de Reed-Solomon qui font l'objet de ce travail. On a vu que ces codes sont des codesdétecteurs correcteurs d'erreurs pouvant corriger les erreursindépendantes, les symboles erronés contigus dans la mesure oùle nombre d'erreurs ne dépasse pas la capacité de correction.

Le décodage des codes de Reed-Solomon est l'objet du troisième chapitre. Dans notre travail on a essayé de présenter les algorithmes de décodage les plus utilisés dans le domaine de communication à savoir : Algorithme de Peterson, Algorithme de Berkelamp-Massey, Algorithme de Sugiyama.

Puisque notre domaine est les télécommunications, on peut proposer comme perspectives l'étude d'un système de communication numérique avec un bruit non gaussien comme les bruits à distributions alpha-stables à l'aide des HOS. On trouve

cetype de bruits dans les systèmes de communication PLC (Power Line Communication)qui permettent de transmettre des données via le réseau électrique et de commander des appareils électriques.

Il est nécessaire de signaler les domaines d'applications des codes de Reed-Solomon : **Stockage de données** :Pour le CD, on utilise 2 codages de Reed-Solomon, on code une première fois avec un code C1 = RS (28, 24) puis on entrelace ensuite on code à nouveau les données entrelacées avec un code C2 = RS (32, 28). L'idée est que le premier code permet d'éliminer le bruit ambiant mais s'il ne peut corriger il efface le bloc et ensuite le code est désentrelacé ainsi la perte d'information est diluée sur une grande plage de données ce qui permet au code de corriger ces effacements.Pour le DVD le principe est le même que pour les CD, on a un code PI= RS (182, 172) et un code PO = RS (208, 192). **Transmission par satellite :** Pour le DVB, le codage est RS (204, 188, t=8). **Transmission de données :** En ADSL/ADSL2/ADSL2plus, le codage est souvent RS (240,224,t=8) ou encore RS (255,239,t=8).

La faiblesse de ces codes réside au faible nombre de symboles que le codage Reed-Solomon peut corriger, car le codage ici est très mauvais en cas de bruit impulsif de longue durée, ou de bruit aléatoire régulier : Pour la transmission de données (ADSL, DVB-T), le bruit impulsif peut être dû à des moteurs, relais, lampes à décharge ou tubes d'éclairage, clôture électrique...etc et pour le stockage de données (CD, DVD), le bruit impulsif peut être dû à une rayure sur le support.

Comme perspectives on peut proposer pour l'année prochaine l'implémentation de ces codes sous Matlab, par exemple, pour permettre de bien comprendrele fonctionnement de codage et de décodage.

Bibliographie et Webographie

[01] Badrikian Josèphe mathématique pour téléinformatique, codes correcteurs, principes et exemples.

[02] Christophe Ritzenthaler, codes correcteurs d'erreurs.

[03] F.J. MacWilliams and N.J.A. Sloane. The theory of error- correcting codes, volume 16. North holland mathematical libray, 1977.

[04] Hankerson, DR et al. Théorie du codage et cryptographie: Les Essentiels. Marcel Dekker, New York, 1991.

[05] C. K. P Clarke, Reed – Solomon error correction, British Broadcast Corporation

[06] Wicker SB., Bhargava VK. An Introduction to Reed-Solomon Codes (http://media.wiley.com/product_data/excerpt/19/07803539/0780353919-2.pdf)

[07] Wikipedia Avril 2007, Reed-Solomon error correction, http://en.wikipedia.org/wiki/Reed-Solomon_error_correction

[08] Hall, J.I., Generalized Reed-Solomon, Michigan State University

[09] Lin, S., Costello D.J. Jr., Error Control Coding : Fundamentals and Applications, New Jersey : Prentice Hall Inc., 1983.

[10] Moon, T.K., ERROR CORRECTION CODING, Mathematical Methods and Algorithms, New Jersey : John Wiley & Sons, Inc., 2005.

[11] Chouinard, J.-Y., Notes de Cours - GEL-66680 Théorie et pratique des codes correcteurs, Département de génie Électrique, U Laval, Janvier, 2007.

[12] US NAVY, Euclidean Algorithm, http://web.usna.navy.mil/~wdj/book/node64.html