République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Akli Mohand Oulhadj – Bouira

Faculté des sciences et des sciences appliquées

Département de Génie Electrique

Mémoire de Master

Option : Technologies des Télécommunications

Réalisé Par :

- AYADI Hanane

- RATNI Souhila

Thème

La correction des erreurs dans les transmissions à accés aléatoire

Date de soutenance : 23/09/2017

Devant le jury composé de :

- Mr.NOURINE Mourad	Président
-Mr. BOUCENNA Mohammed Lamine	Rapporteur
-Mr.BENSEGUENI Skander	Examinateur
- Mr.ARABI Abd erazzek	Examinateur

Année universitaire : **2016- 2017**

Remerciement

Nous remercions Allah le tout puissant, qui nous a donné la force et la patience pour l'accomplissement de ce travail.

Nous remercions les chers parents qui nous ont donné la volonté pour la réussite de ce travail.

Nous exprimons toutes nos gratitudes au ***Dr BOUCENNA Mohamed Lamine****, pour l'effort fourni, les conseils prodigués, sa patience et sa persévérance dans le suivi. Cela a été un plaisir et un honneur de travailler avec lui.*

Nous adressons également nos remerciements à tous nos enseignants, pour leurs aides inestimables, qui nous ont donné les bases de la science, et en perticulier ***Dr BENSEGUENI skander*** *pour son aide et son effort.*

Nous remercions très sincèrement, les membres de jury d'avoir bien voulu accepter de faire partie de la commission d'examinateurs.

Nous tenons à remercier aussi l'ensemble du personnel de la faculté des sciences et sciences appliqués surtout le département Génie Electrique.

Nous remercions aussi toute personne qui a participé de près ou de loin pour l'accomplissement de ce modeste travail.

Dédicace

J'ai le grand honneur de dédier ce travail A l'être le plus cher à mon cœur, Maman rahemah Allah.

A celui qui a fait de moi une femme, Mon père.

A mon mari : Abd Erazzek.

Mes chers frères : Mohamed, Yassine, fouaz.

Mes chères Sœurs : Naima, Karima, Nouara, Massouda, Fatouma.

A mes cousins : Sidali, Zahra, Hassna

A tous mes amies : Zahra, Wafa, Baya, Hanane, Amina, Nana, Chahrazed, Houda, Manar, Meriem, Assma, Khalissa, sabah, Hadjer, Hassina, Sihem.

Mes très chers amis « Houria »

Et mon binôme « Hanane »

A tous mes collègues de pénalité Master Télécom qui me donnent le courage.

A tous ceux et celles qui m'ont aidé de loin ou de prés.

Je dédie ce modeste travail.

Souhila

Dédicace

A Mamère, symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.

Et en souvenir de mon père, avec amour et gratitude.

A tous mes frères et sœurs : Samih, Kamel, Abd el Raouf, Amine, Fatima, Soumai, Kami, Faiza, Ahlem.

A tous ma famille, en particulier, mes grandes mères, Houria et Fatima RahemahomAllah,Bachira,Ratiba,Nadia,Ratiba,Amel,Sihem,Lina ,Hassiba ,Hanai,Hayate

Atous mes amies : Wafa, Baya, Shahrazade, Maryouma, Samira, Khalissa, Assma, Manar, Houda.

Ames voisines Nacira, Aldja, Naima, Zahra, Zahia et mon binôme Souhila.

Atous les personnes de promotion Master Telecom qui me donnent le courage.

Atous ceux et celles qui m'ont aidé de loin de près, Je dédie ce modeste travail.

Hanane

Sommaire

Chapitre 3 : Les performances du code RS

Liste des figures

Liste des tableaux

Liste des abréviations

La signification d'une abréviation ou d'un acronyme n'est souvent indiquée qu'à sa première apparition dans le texte. Il existe dans la plupart des cas une abréviation en français et une abréviation en anglais. Toutes les deux sont indiquées une première fois puis nous employons l'abréviation la plus usuelle, qui est le plus souvent l'abréviation en anglais.

A **ADSL:** Asymmetric Digital Sbscribre Line.

B **BBAG :** Bruit Blanc Additif Gaussien.

BCH: Bose-Chaudhuri –Hocqenghem.

BER: Bit Error Rate.

C **CBC :** Canal Binaire Symétrique.

CAE : Canal à Effacement.

CD : Compact Disc.

CDMA: Code Local Area Network.

CRC: Cyclic Redundancy Checksum.

D **DVD:** Digital Versatile Disc.

DVB: Digital Video Broadcasting.

G **GF:** Galois Field.

L **LDPC:** Lowe Density Parity Check.

M **MCD:** Modèle Conceptuel de Données.

O **OFDM:** Orthogonale Frequency Division Multiplexing.

P **PSNR:** Peak Signal to Noise Ratio.

R **RS:** Reed-Solomon.

V **VDSL:** Very High Bit-rate Digital Subscriber Line.

W **WLAN:** Wireless Local Area Network.

X **XOR :** Ou exclusif ou « exclusive OR ».

Problématique

Le code correcteur d'erreurs, dont l'origine remonte à la fin des années 40, permet de transmettre de façon fiable de l'information codée au moyen de mots binaires d'une longueur donnée, sur des lignes plus ou moins bruitées. Tel que la transmission de l'information binaire sur des lignes bruitées présente un risque d'erreurs variable selon les cas, il s'agit de trouver un moyen de les corriger à la réception de l'information, au prix d'une certaine redondance, tout en minimisant dans chaque situation le temps d'occupation de la ligne.

Les codes correcteurs d'erreurs ont leur source dans un problème très concret lié à la transmission de données. Dans la grande majorité des cas, une transmission de données se fait en utilisant une voie de communication, soit un canal de communication qui n'est pas toujours fiable. Alors, les données au cours de leurs transports dans le canal sont susceptibles d'être altérées. Tel que lors d'une communication radio, la présence de parasites sur la ligne va perturber le signal portant le son et la voix. Pour cela, on se demande comment peuvons nous récupérer le signal bruité et donc nos informations originales. Aussi, quel type d'algorithmes on utilisera pour rendre cette communication plus fiable avec un taux d'erreurs plus faible. Pour répondre à ces questions ; nous proposons dans ce travail une étude détaillée sur les codes correcteur d'erreurs qui sont enmployés justement pour limiter les erreurs comises pendants les transmissions.

Introduction générale

Le monde des télécommunications a vécu des développements rapides et en continu en répondant aux éxigences énormes de la société. Actuelllement, le monde en entier est lié aux moyens de communication qui font avancer et améliorer nos modes de vie. Lorsque l'être humain se trouve à présent lié dans toutes ses activités aux moyens de communiactions par lesquels il achève avec facilité et plus de rapidité la plupart de ses missions. Cependant, tous les moyens de communication sont soumis à des perturbations, lorsque par exemple, un expéditeur envoie un message « m » à une distance quelconque, et durant la transmission de ce message des perturbations peuvent se produire et le récepteur va reçevoir un message « m'» qui comporte alors des erreurs. Nous précisons que les supports de communication sont multiples et chacun d'eux présente de différentes performances, aussi, les techniques d'accès à ces supports sont divers et sont intégrées pour gérer l'usage du canal de transmission à travers un algorithme étudié et en particuliers les techniques à accès aléatoire qui présentent jusqu'à nos jours des taux d'erreurs importants. D'où la nécessité de « corriger » cette transmission : c'est le rôle des codes correcteurs d'erreurs. Cette sécurisation qui fait le critère essentiel pour corriger les erreurs et réussir la transmission. Cela se voit plus important particuliérement dans des transmissions à base des techniques à accès aléatoire.

Les codes correcteurs d'erreurs sont des mécanismes ou des procéssus qui permettent de corriger les erreurs dans un mot de code en ajoutant aux informations des symboles redondants qu'on appel des symboles de contrôle. Ce code est utilisé pour accroitre la fiabilité de système de transmission et augmenter leur débit [10]. Alors, nous allons au début de ce rapport présenter une étude détaillée sur les différents codes correcteurs d'erreurs notamment les codes de Hamming et les codes de Reed-Solomon. Ces derniers seront bien détaillés dans leurs opérations de codage et décodage en donnant des exemples pour mieux éclaircir leurs stratégies et principe de codage et de décodage.

Ce mémoire et composé de trois chapitres : dans le premier chapitre nous introduisons les concepts généraux des codes correcteurs d'erreurs tel que les codes en blocs et les codes convolutifs ensuite dans le deuxieme chapitre nous étudions en particulier les codes de Hamming et les codes de Reed-Solomon. Et à la fin le chapitre trois, qui traite nos exprémentations d'evaluer les performances de code de Reed-Solomon. Nous résumons notre travail par une conclusion générale.

Chapitre 1: généralités sur les codes correcteurs d’erreurs

1.1 INTRODUCTION

Le développement de la théorie de l'information par Claude Shannon en 1948 a donné naissance à la théorie des codes. Poussés par les avancées scientifiques et technologiques d'un côté, et les besoins industriels d'un autre côté ; les codes n'ont eu de cesse de progresser. Lorsqu'ils sont utilisés pour corriger des erreurs intervenant sur des informations codées sous forme numérique ; ces codes sont appelés codes correcteurs d'erreurs [3].

Les codes correcteurs sont largement utilisés dans le domaine des télécommunications, où ils permettent des transmissions fiables sur des médiums de communication bruités comme les canaux sans fil. On les retrouve également dans le domaine du stockage, pour protéger l'information enregistrée face à la détérioration du support [10].

Ce chapitre représente les deux grandes familles des codes correcteurs d'erreurs qui sont les codes convolutifs et les codes en bloc de différente architecture.

1.2 Chaîne de transmission numérique

Les systèmes de transmission numérique véhiculent de l'information entre une source et un destinataire en utilisant un support physique comme le câble, la fibre optique ou encore, la propagation sur un canal radioélectrique. Les signaux transportés peuvent être soient directement d'origine numérique comme dans les réseaux de données, soient d'origine analogique (parole, image...) mais convertis sous une forme numérique. Le principe du système de transmission est alors d'acheminer l'information de la source vers le destinataire avec le plus de fiabilité possible [1].

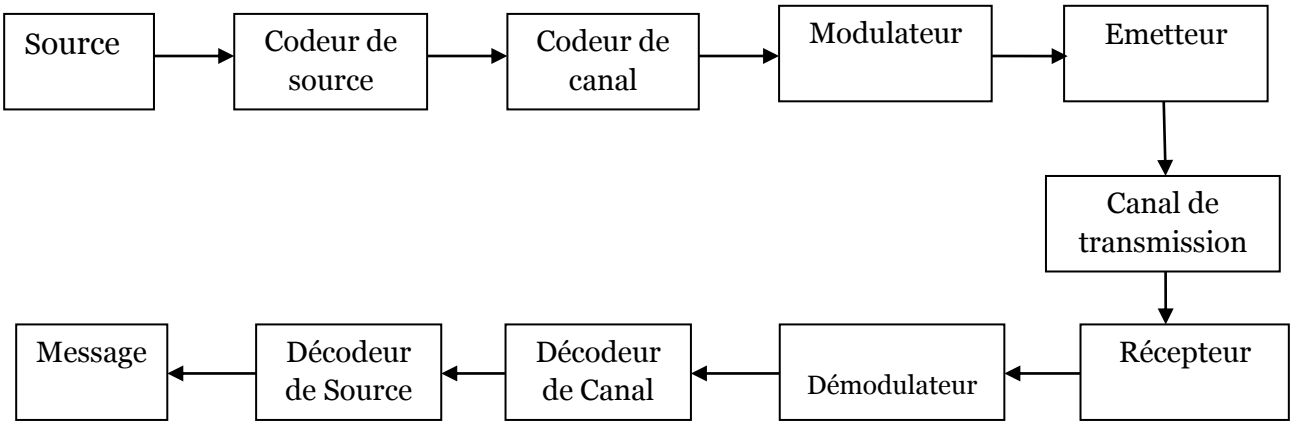


Figure 1.1: *Schéma bloc d'une chaîne de transmission numérique* [2].

Une chaine de transmission numérique peut être représentée par différents blocs modélisant les traitements successifs apportés à l'information. Les blocs peuvent être énumères comme suit [2] :

- **La source :** émet un message numérique sous forme d'une suite d'éléments binaires (Bits).
- **Le codage de source :** supprime les redondances contenues dans le message afin de rendre les éléments binaires mutuellement indépendants.
- **Le codage de canal :** insère des éléments binaires pour améliorer la qualité de la transmission.
- **Le modulateur :** traduit le message binaire en signal permettant son transport dans les milieux tel que l'air, l'eau, les câbles etc.
- **L'émetteur :** permet au signal de se propager dans le canal de transmission.
- **Le canal de transmission :** est un support physique qui est utilisé pour la transmission d'un signal à partir d'un émetteur jusqu'au récepteur. Mais, des perturbations aléatoires non prévisibles affectent le signal transmis avant sa réception.
- **Le récepteur :** capte le signal émis.
- **Le démodulateur :** traduit le message reçu en signal binaire.
- **Le décodeur de canal :** détecte et/ou corrige les erreurs de transmission grâce aux éléments binaires ajoutés lors du codage.
- **Le décodeur de source :** régénère le message binaire.

1.3 Les modèles des canaux

Les systèmes de communication utilisent un support de transmission aussi appelé canal de transmission afin d'échanger de l'information d'un point à un autre. N'étant pas parfait, il introduit des perturbations, affaiblissements, écho, bruit qui détériorent l'information émise et créent des erreurs dans le message. Afin de fiabiliser le message (empêcher la perte de due aux perturbations, bruit), les systèmes intègrent un processus de protection du message émis. Le principe général de cette protection est l'ajout de redondance, c'est-à -dire d'information supplémentaire la plus optimale possible en termes de cout, de volume et de contraintes qui dépendent largement du canal [1].
Nous présentons les trois types de canal :

1.3.1 Le canal binaire symétrique (CBS)

C'est un canal binaire caractérisé par la probabilité d'erreur $\boldsymbol{p}$ qu'au cours de la transmission un bit (**0** ou **1**) soit modifiée son opposé. Ces modifications se produisent indépendamment sur chacun des bits transmis. Le canal CBS peut être représenté par le schéma suivant [1] :

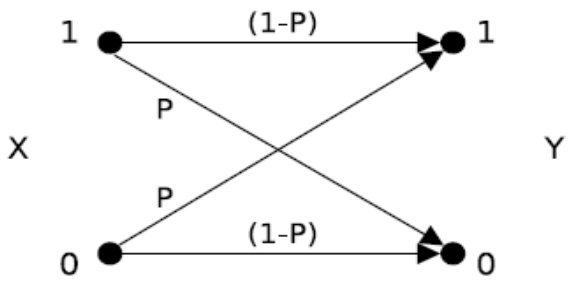


Figure 1.2** : Canal binaire symétrique **[4].

1.3.2 Le canal à effacement (CAE)

Tout comme le **CBS**, le Canal à effacement (**CAE**) est un canal discret, stationnaire et sans effet mémoire [3]. Les erreurs qui interviennent sur ce type de canal sont des effacements des informations. Contrairement au canal binaire symétrique, l'information transmise sur ce canal n'est pas altérée, mais une partie de celle-ci est tout simplement perdue. Sur un canal à effacements de paramètre ***p*** ; la probabilité qu'un symbole transmis soit effacé est égale à ***p***. On modélise souvent ce canal en ajoutant à l'ensemble des valeurs que peut prendre la sortie du canal, un symbole E représentant l'effacement (voir figure 1.3).

La loi de transition du canal binaire à effacements est :

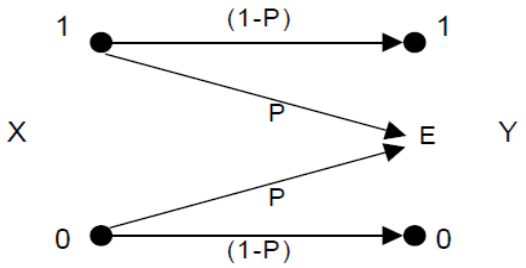


***Figure1.3** : Représentation d'un canal binaire à effacements [4].*

1.3.3Canal à bruit additif blanc gaussien

Le modèle de canal le plus fréquemment utilisé pour la simulation des transmissions numériques, et qui est aussi un des plus faciles à générer et à analyser ; est le canal à bruit blanc additif gaussien (BBAG). Ce bruit modélise à la fois les bruits d'origine interne (bruit thermique dû aux imperfections des équipements…) et le bruit d'origine externe (bruit d'antenne…). Ce modèle

est souvent associé à une transmission filaire, puisqu'il représente une transmission quasi-parfaite de l'émetteur au récepteur. Le signal reçu s'écrit alors :

$$r(t)= s(t)+v(t) \quad \textbf{(1.1)}$$

Où v(t) représente le BBAG est un bruit dont la densité spectrale de puissance est la même pour toutes les fréquences (bruit blanc). Il est dit additif car il est simplement ajouté au signal entrant. Enfin, il est dit gaussien du fait de sa densité de probabilité de transmission définie comme suit [1,3] :

$$P\,(r/s)= \frac{1}{\sigma_v\sqrt{2\pi}} e^{-\frac{(r-s)^2}{2\sigma_v^2}} \quad \textbf{(1.2)}$$

Où σ_V : La variance.

r : Le signal reçu.

s : Le signal transmis.

1.4 Les codes correcteurs d'erreurs

Les codes correcteurs d'erreurs ont pour objectif de permettre la transmission d'information malgré l'ajout éventuel d'erreurs lors de la transmission. Les codes ce sont des techniques de codage de l'information basée sur la redondance au message à transmettre [5].

Figure1.4 : *Schéma général des Codes Correcteurs.*

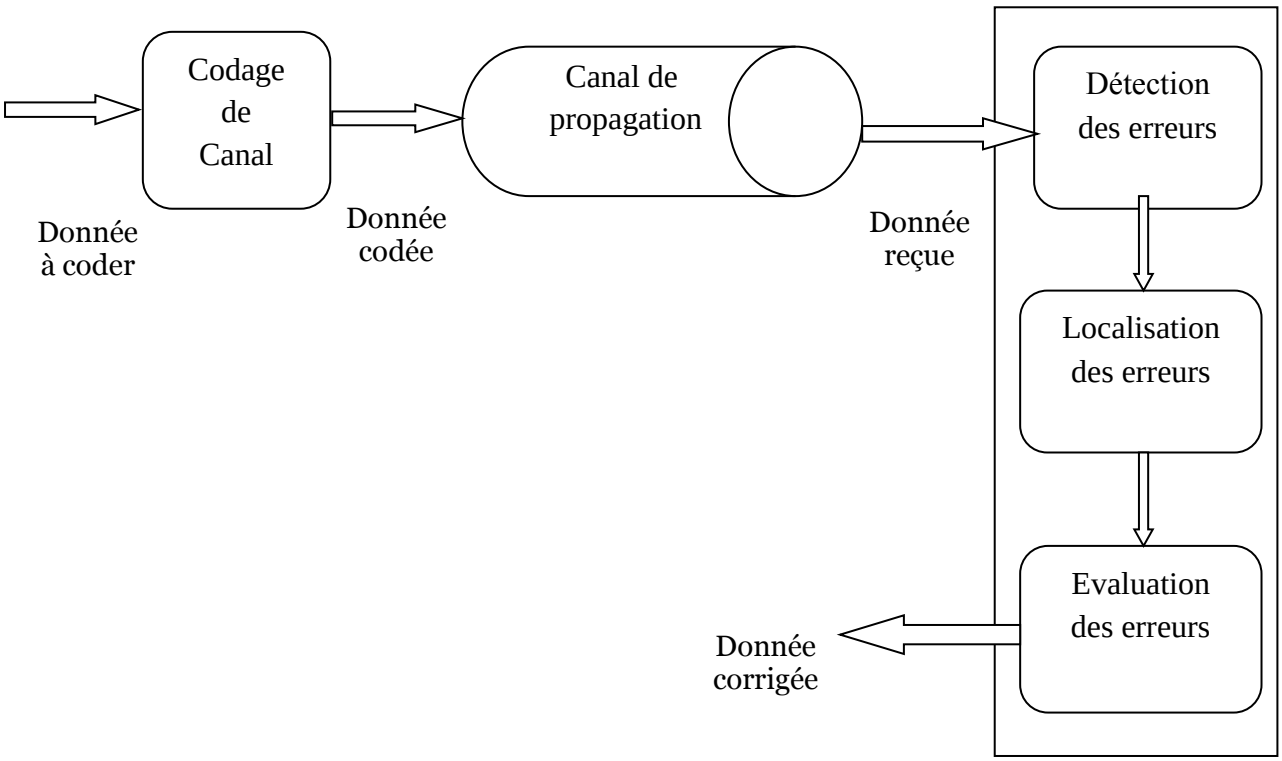


***Figure1.5* :** *l'utilisation d'un code correcteur d'erreurs [21].*

On considère généralement qu'il existe deux classes de codes correcteurs : les codes en blocs et les codes convolutifs. A l'intérieur d'une classe nous distinguerons plusieurs familles de codes, chaque famille se distingue des autres par ses propriétés structurelles (construction, algorithme de décodage). Nous donnons quelques exemples des familles de codes [1].

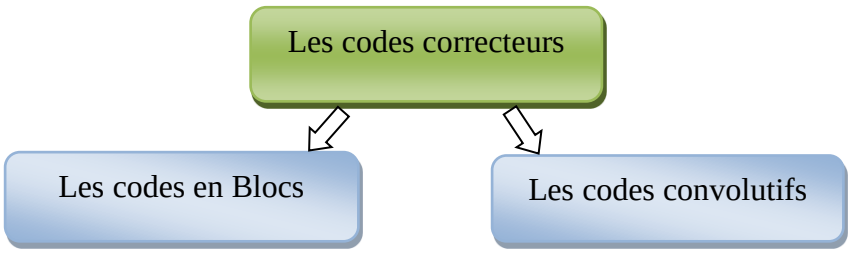


***Figure1.6* :** *les familles des codes correcteurs d'erreur*

1.4.1 Les codes en blocs

Le codeur employé dans un code en bloc divise la séquence d'information en bloc de message de taille fixe *k* bit, pour transformée chacun de message *Di* en un mot de code *Ci* de taille *n* en appliquant une loi linaire. La redondance associée à chaque bloc est de taille *r*, où $k + r = n$.
Le rendement d'un code bloc est défini par la formule suivante :

$$R = \frac{k}{n} \quad \textbf{(1.3)}$$

Où, *k* et *n* représentent respectivement les nombres de bits en entrée et en sortie du codeur, et le taux de codage est *(n-k)*.

Les codes en blocs linéaires constituent un faible pourcentage de l'ensemble des codes en blocs. Les codes en blocs sont les plus utilisés dans la pratique [7,10].

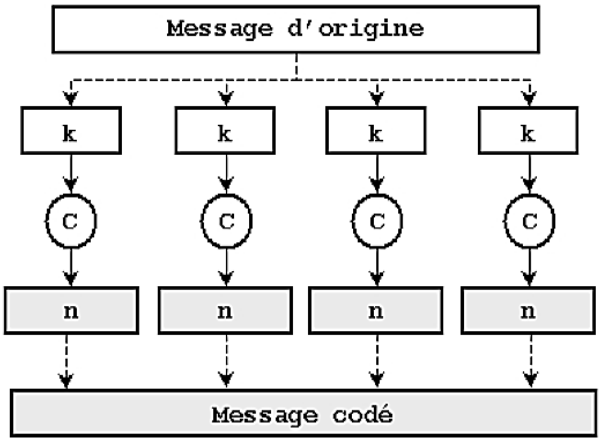


Figure1.7: *Codage en bloc* [11].

Nous donnons ci-dessous une liste de quelques familles de codes en bloc linéaires.

1.4.1.1 Les codes à répétition

Le code à répétition consiste à répéter *n* fois le bit d'information. C'est un code [*n ;* 1 *; d*], de dimension 1, de longueur *n* et de distance *d*. Pour *n* impair, ce code est parfait et optimal lorsque l'on protège un seul bit de donné. Par exemple, pour *n* = 3.

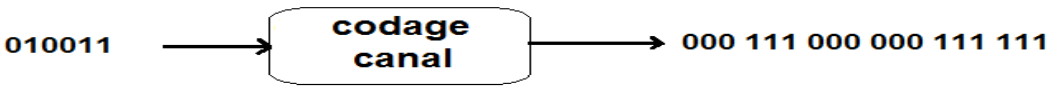


Figure1.8 : *codage de code à répétition.*

Ce code est de dimension 1 et de longueur *n*, chaque bit étant pris un par un et transformé en *n* bits. Le rendement est donc de $\frac{1}{n}$, ce qui est très faible. La distance minimale entre deux mots de code est *n*. En effet, il faut changer les *n* répétitions d'un bit d'un mot de code pour obtenir un autre mot de code. Ce code permet de corriger jusqu'à $[\frac{n}{2}]$ erreur et dispose d'un algorithme de décodage très simple : il suffit de prendre le bit majoritaire de chaque bloc de *n* bits. Si l'on reprend l'exemple *n* = 4 et que le mot reçu est 0100, le mot de code le plus proche est 0000 et le mot d'information qui a

sûrement été envoyé est 0. On remarque que dans cet exemple si le mot reçu est 0110 alors il existe deux mots de code à distance 2 ; on détecte toujours la présence d'une erreur mais le décodage n'est plus unique, alors on ne peut pas savoir si le mot envoyé est 0 ou 1.

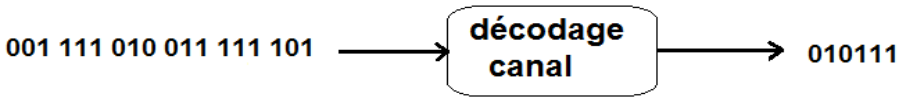


Figure1.9 : *décodage de code à répétition.*

Le décodage s'effectue par vote majoritaire et la capacité de correction est donnée pas $\frac{n-1}{2}$

Ce code est surtout utilisé en télécommunication pour reconnaître une modulation de signal [1,5].

1.4.1.2 Les codes de parité

Dans ces types des codes, intéressons-nous à la transmission des caractères de texte ; un caractère (une lettre ou un signe) est conventionnellement une suite de 7 chiffres binaire. Un texte exprimé sous forme binaire est découpé en mots d'information qui sont ici des caractères. Le codage appelé par bits de parité consiste à ajouter à la suite de chaque mot d'information, un nouveau bit, de telle sorte que le nombre total de chiffres 1 soit alors de parité fixé (paire ou impaire). Il s'agit d'un code systématique dont la clé de contrôle ne possède qu'un seul bit [6].

Le bit de parité est calculé de telle sorte que le nombre total de 1 soit toujours pair (par exemple).

- Mot d'information : 100 1101 → Mot de code : **0**100 1101
- Mot d'information : 110 0111 → Mot de code : **1**110 0111

1.4.1.3 Le code de Hamming

Le code de Hamming est un code correcteur d'erreur linéaire, Permet de détecter et corriger automatiquement l'erreur (une erreur pas deux et le procédé est donc valable pour des taux d'erreurs faibles) sur une lettre de message.

Un code de Hamming est un code parfait, ce qui signifie que pour une longueur de code donnée, il n'existe pas d'autre code plus compact ayant la même capacité de correction. En ce sens, son rendement est maximal. Un code de Hamming est de paramètres :

- n : la longueur totale de message.
- k : les bits de message à transmettre.
- n -k : les bites de contrôle.

1.4.1.4 Les codes cycliques

Les codes cycliques sont des codes linéaires stables par permutation circulaire des coordonnées. Ils ont une forte structure mathématique : si on les voit comme des espaces vectoriels de polynômes (les coordonnées des mots deviennent les coefficients des polynômes) [3]. Ces codes ont la propriété d'être stable par permutation circulaire des mots.

T : $F_{2^n} \rightarrow F_{2^n}$

$(X_1, X_2, \ldots\ldots, X_n) \rightarrow (X_2, \ldots\ldots, X_n, X_1)$

On identifie F^n à l'algèbre F_2 [X]/($X^n - 1$) par $(x_1, x_2, \ldots\ldots, x_n) \rightarrow x_1 X^{n-1} + \cdots + x_{n-1} X + x_n$

Ce sont des codes polynomiaux dont le polynôme générateur g(x) divise (x^n+1) où *n* est la longueur du code. Cette particularité permet la construction immédiate d'une matrice de contrôle caractéristique de ce type de code et une simplification de la méthode de correction automatique. Pour la détermination des codes cycliques de longueur *n*, la connaissance du diviseurs ($X^n + 1$) est essentielle [6].

1.4.1.5 Les codes BCH

Les codes **BCH** (Bose-Chaudhuri-Hocqenghem) sont des codes cycliques. Ils ont été découverts à la fin des années 50 et l'acronyme est composé des lettres des noms de ses inventeurs, Raj Bose, D. K. Ray-Chaudhuriet Alexis Hocquenghem. Ces codes sont construits sur un alphabet composé d'un grand ensemble de symboles basés sur les propriétés des corps finis. Ils définissent une méthode systématique pour construire des codes cycliques capables de corriger un nombre *t* d'erreurs arbitrairement fixés dans un bloc de N éléments binaires.

Les performances de décodage de ces codes sont directement liées à la distance de Hamming du code en bloc utilisé ; plus cette distance est grande meilleure seront les performances de décodage. Les codes BCH sont utilisés dans des applications telles que pour les communications satellitaires [3].

1.4.1.6 Les codes Reed-Solomon

Ces codes appartiennent à la classe des codes correcteurs d'erreurs cycliques non-binaires. Ils sont formés de *n* symboles, avec $n = q - 1$ et $q = 2^s$, chaque symbole appartenant à GF(q) qui est le corps de Galois (Galois Field) à *q* éléments, s représente donc le nombre de bits par symbole. Pour distance minimale *d* et pour dimension $k = n - d + 1$. Nous avons donc affaire à des codes [$n, k, n - k + 1$]. Le nombre t est égal à $(n-k)/2$ représente le nombre de symboles d'erreurs que ce

code sera capable de corriger. Ils sont basés sur des polynômes générateurs. Ces codes sont donc capables de détecter et corriger plusieurs erreurs symboles par paquet du fait de leur structure.
Les codes Reed Solomon sont utilisés dans beaucoup d'applications courantes telles que les Compact Disc (**CD**), Digital Versatile Disc (**DVD**) etc. [1,3,7].

1.4.1.7 Les codes LDPC

Les codes Lowe Density Parity Check (**LDPC**) ont été découverts dans les années 60 par Gallager [3]. Mais il a proposé seulement une méthode générale pour construire des codes LDPC pseudo aléatoires ; les bons codes LDPC sont générés par ordinateur (en particulier les codes longs) [9]. Ces codes demandent une grande complexité calculatoire qui n'était pas disponible au moment de leur découverte. Ils sont donc restés discrets jusqu'au 1996. MacKay, travaillant sur les Turbo codes à ce moment-là, a donné une deuxième naissance aux codes LDPC MacKay (1999). En effet, cela fut possible du fait de l'avancée de la technologie en électronique qui a permis de pouvoir implémenter les algorithmes de décodage de ces codes [3].
Les codes LDPC ont par définition une matrice de parité creuse H (le poids de chaque ligne vaut quelques unités, alors que la longueur peut atteindre plusieurs milliers). La performance de l'algorithme de décodage tient essentiellement à la faible densité de la matrice [1]. Elle contient très peu de 1 (< 3%). Elle peut être représentée sous la forme d'un graphe de Tanner. Ce graphe, dont un exemple est présenté en **figure 1.10** avec une matrice quelconque (non creuse), est composé de N nœuds de variable et de N -K nœuds de parités. Un nœud de variable *i* est connecté à un nœud de parité j si H (i ; j) = 1 [4].

Matrice quelconque Graphe correspondant

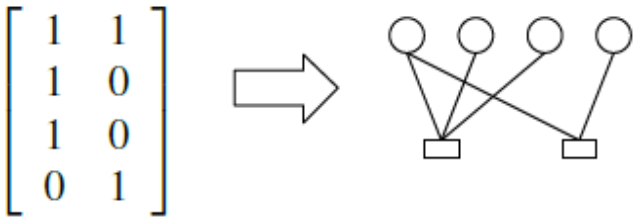


Figure 1.10: *Représentation d'une matrice de parité H d'un code LDPC et de son graphe de Tanner [4].*

1.4.2 Les codes convolutifs

Les codes convolutifs ont été introduits par Elias en 1955. Bien qu'apparus plus tard, ils offrent des performances égales, voir supérieures dans beaucoup d'application pratiques aux codes

en blocs, les codes convolutifs plus utilisés dans les systèmes de télécommunications fixes et mobiles. Ils sont généralement plus faciles à implémenter et utilisables en temps réel [7].

Un code convolutif diffère d'un code bloc par le fait que chaque bloc de *n* éléments en sortie ne dépend pas seulement des k entrées à un instant donné (**figure 1.11**) [10].

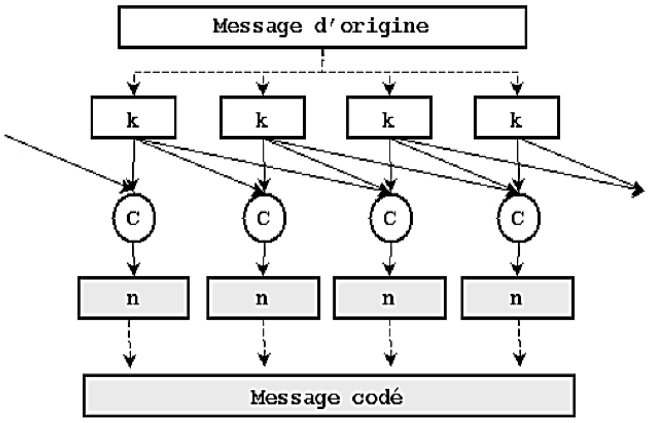


***Figure1.11** : Codage convolutif* [10].

La structure des codes convolutifs est riche et permet plusieurs approches.

1.4.2.1 Principes

Le codeur d'un code convolutif accepte lui aussi en entrée une séquence d'information constituée de blocs de K bits et génère une séquence de blocs de N bits. En codage convolutif un bloc de sortie n'est pas relié qu'à un seul bloc d'entrée mais à plusieurs [7].

Un codeur convolutif est défini à l'aide de trois paramètres (n ; k ; m), *n* le nombre de sorties, *k* le nombre d'entrées et *m* la taille de la mémoire. À chaque unité de temps, le codeur lit *k* bits d'information et produit *n* bits codés (nous avons $n > k$) [1]. Un codeur est composé de (m+1) registres de K bits pour stocker les m entrées précédentes et l'entrée actuelle. Cette quantité, (m+1), est appelé *la longueur de contrainte du code.* Le rapport $\frac{k}{N}$ est appelé le rendement du code [3]. Les paramètres K et N prennent en général des valeurs inférieures à celles prises par des codes en blocs équivalents. On choisira une taille de mémoire suffisamment grande pour que la probabilité d'erreur de décodage soit très faible [7].

1.5 Etude comparatifs des déférant codes

Comme nous avons expliqué lors de ce chapitre ; il existe plusieurs types de code dont chacun d'eux possède des caractéristiques spécifiques et différentes par rapport à l'autre. Dans l'article [11] l'auteur présente des comparaisons entre quelques codes comme : le code Reed

Solomon (RS) et les blocs de circuits turbo et par des simulations explicites il démontre que les codes turbo offrent un BER bien plus petit que celui du code RS que ce dernier présente une capacité de correction meilleure que celle des codes CRC qui ne font que détecter l'erreur sans la corriger. Aussi, dans un travail similaire dans [12] ; le chercheur montre que les meilleurs rapports PSNR sont obtenus par le codage turbo à base du code convolutif et qu'il présente un taux du PSNR plus important que celui de Reed Solomon. Nous présentons ci-dessous un tableau récapitulatif des caractéristiques de quelques codes afin d'avoir une idée plus claire sur les différences existantes entre les différents codes.

Tableau 1.1 : *Comparaison entre quelques types de codes.*

Les codes	***Domaines d'application***	***Avantages***	***Inconvénients***
Code de répétition	- En télécommunication pour reconnaître une modulation du signal.	- Protéger simultanément plusieurs bits d'information. - Permet de corriger n/2 erreurs.	- Rendement faible.
Code de parité	- Dans les transmissions séries	- Détection d'une seule erreur.	- Ne permet pas de corriger l'erreur.
Code de Hamming	- En informatique dans les traitements de signal et En télécommunications.	- Permet de détecter et corriger automatiquement les erreurs. - Un rendement important.	- Permet de corrigé une seule erreur.
Code de Cyclique	- Une représentation polynomiale des bits à transmettre.	- Détecter toutes les erreurs simples et doubles. - Plus performant que les simples checksums, surtout pour les paquets / rafales d'erreurs.	- Moins coûteux en taille.
Code BCH (Bose Chaudhuri-Hocqenghem)	- En communications Satellitaires.	- Permet de corriger un nombre t d'erreur arbitrairement fixés dans un bloc de N éléments binaires. - Une grande capacité de correction d'erreur. - Bon rapport signal sur bruit. - Faible complexité de codage et de décodage.	- Rendement faible.
Code Reed Solomon	- Dans les communications mobile et réseaux sans fils, Dans les communications	- Détecter et corriger plusieurs erreurs symboles.	- Une grande complexité de calcules.

	satellitaires, Dans la télévision numérique et la ggggradio diffusion numérique DVB. - Dans les modems ADSL et VDSL, Dans la sauvegarde de données (sauvegarde magnétique, optique, etc…). - Les codes concaténés. CD et DVD.	- Efficace dans la correction d'erreurs par paquet.	
Code LDPC (Low Density Parity Check)	- Application temps réel comme le stockage magnétique. - Les réseaux locaux sans fil avec un débit élevé (WLAN). - Systèmes CDMA et OFDM.	- Ne nécessite pas d'entrelacereurs pour réaliser une bonne performance d'erreur. - Une meilleure performance par trame. - Marge d'erreur se produit à un niveau de BER plus faible. - Le décodage n'est pas serial et peut être réalisé par un processus parallèle.	- Une grande complexité de calcules.

Tableau 1.1 : *Comparaison entre quelques types de codes.*

1.6 Conclusion

Dans ce premier chapitre nous avons défini les différents étages composant une chaîne de transmission numérique lorsque l'étage du codage fait une phase primordiale et déterminante en face des différentes perturbations rencontrées pendant la transmission et qui peuvent dégrader amplement la qualité du signal par la création d'éventuelles erreurs au niveau des informations envoyées. Et c'est là que vient l'importance des codes correcteurs des erreurs pour récupérer les pertes des données et restituer le contenu du signal transmis. Pour cet effet, nous avons consacré la grande partie de ce chapitre à introduire les codes correcteurs d'erreurs en décrivant leurs différents types et les caractéristiques de chaque type de code. Alors, nous avons présenté les deux grandes familles des codes correcteurs d'erreurs et les relations qui existent entre eux ainsi que la technique de correction adoptée par chaque type de code. En fait, afin de conclure cette recherche ; nous avons rassemblé les caractéristiques de quelques types de code dans un tableau récapitulatif qui permet d'avoir une idée rapide sur le domaine d'application et les performances positives et négatives de chaque type de codage.

Chapitre 2 : Les codes Hamming et Reed-Solomon

2.1 Introduction

Dans ce chapitre nous nous intéresserons à deux codes en bloc ; les codes de Hamming et les codes RS. Nous commençons par l'étude des caractéristiques et principe de fonctionnement du code Hamming et nous présentons des exemples explicites pour mettre en claire la capacité du codage Hamming dans la détection et la correction des erreurs. Notre principal intérêt est d'étudier les codes RS qui sont largement appliqués suite à leur force importante envers la correction des erreurs et effacements. Pour cet effet ; nous étudions dans la deuxième partie de ce chapitre les codes RS où nous définissons l'opération de codage et de décodage pas à pas en détaillant le jeu de calcul et d'équation formant le système de codage/décodage RS.

2.2 Code de Hamming

Le code de Hamming est un code binaire défini par sa matrice de parité plutôt que par sa matrice génératrice. C'est la matrice de dimension $r \times (2r-1)$ qui contient toutes les colonnes non nulles distinctes que l'on peut écrire sur $\boldsymbol{r}$ bits. Le code de Hamming est donc l'ensemble des mots de longueur $2^r - 1$ dont le noyau est *H*. C'est donc un espace de dimension $2^r - 1 - r$. De plus, la distance minimale de ce code est 3 car il n'existe aucun mot de code de poids 1 ou 2 puisque cela signifierait qu'il y a une colonne nulle ou deux colonnes égales dans *H*. On a donc un code $[2^r - 1, 2^r - 1 - r, 3]$ dans lequel on doit donc pouvoir décoder une erreur [15].

2.2.1 La première méthode de Hamming

2.2.1.1La distance de Hamming

Le code de Hamming se base également sur un checksum, sous la forme de bits de parité répartis dans le message à transmettre. On peut définir le bit de parité comme étant égal à zéro si la somme des autres bits est paire et à un dans le cas contraire (impaire) [13].

Exemple 2.1 : 1010001 (7 bits) devient 11010001 (8 bits).

Un code de Hamming se compose alors de m bits du message à transmettre et de n bits de contrôle de parité. La longueur totale du message est de 2^n - 1, et donc m = (2^n- 1) - n. Les bits de contrôle de parité C*i* sont en position 2^i, i ≥ 0, et les bits du message D*j* occupent le reste du code. On parle souvent d'un code de Hamming (n + m) - m.

- Ci sont en position : 2^i et Dj occupe le reste du message [13].

7	6	5	4	3	2	1
D_3	D_2	D_1	C_2	D_0	C_1	C_0

Tableau 2.1 : *Structure d'un code de Hamming 7−4.*

La structure d'un code 7-4 est $D_3D_2D_1C_2D_0C_1C_0$. Tel que pour retrouver l'erreur dans un code de Hamming, on regarde les bits de parité. Dans le cas précédent, si C_2;C_1;C_0 ont tous la bonne valeur, il n'y a pas d'erreur. Dans le cas contraire, la valeur des bits de contrôle indique la position de l'erreur entre 1 et 7. Le code de Hamming présenté ici ne permet de retrouver et corriger qu'une erreur.

Pour connaître quels bits sont vérifiés par chacun des bits de contrôle de parité, il est utile de construire le tableau suivant : les lignes sont numérotées à partir de la dernière colonne de 1 à 2^n - 1(dans l'exemple précédent 2^n - 1 = 7), chaque nombre est converti en binaire et l'on écrit chaque bit dans les colonnes de gauche. On met à côté droit des nombre (1,2,..) des points coloriés de la même couleur de C_i lorsqu'il y a un 1 dans la colonne C_i. Par exemple, 5 sera noté en vert et en rouge, car sur la ligne du 5, il ya un 1 dans les colonnes C_2 et C_0. Les bits de contrôle d'une couleur donnée vérifient les bits du message qui portent la même couleur. Chaque bit de donnée est coloré d'une manière différente, ce qui permet de retrouver la position d'une erreur [13].

C_2	C_1	C_0	décimal	
0	0	1	1 •	← C0
0	1	0	2 •	← C1
0	1	1	3 • •	
1	0	0	4 •	← C2
1	0	1	5 • •	
1	1	0	6 • •	
1	1	1	7 • • •	

Tableau 2.2 : *Méthode pour calculer les bits de contrôle.*

Exemple 2.2 : d'un code de Hamming 7-4 : On souhaite envoyer le message 1010. Complétons le mot de Hamming correspondant :

7	6	5	4	3	2	1
1	0	1		0		

Tableau 2.3 : *la position de message 1010 dans le code de Hamming 7-4.*

- C0 vaut 0 pour pouvoir rendre pair 1+1+0 (les bits d'indices 7,5, 3).
- C1 vaut 1 pour pouvoir rendre impair 1+0+0 (les bits d'indices 7, 6, 3).
- C2 vaut 0 pour pouvoir rendre pair 1+0+1 (les bits d'indices 7, 6, 5).

7	6	5	4	3	2	1
1	0	1	**0**	0	**1**	**0**

Tableau 2.4 : *le mot de code (message envoyé) de code de Hamming 7-4.*

Nous supposons recevoir le mot 0010010 (le bit de poids fort est altéré). Alors ;

- C0 a la mauvaise valeur, car 0+1+0+0=1 est impair, donc il y a une erreur dans les positions 7, 5, 3 ou 1.
- C1 a la mauvaise valeur, car 0+0+0+1=1 est impair, donc il y a une erreur dans les positions 7, 6, 3 ou 2.
- C2 a la mauvaise valeur, car 0+0+1+0=1 est impair, donc il y a une erreur dans les positions 7, 6, 5 ou 4.

Nous écrivons le nombre binaire C2C1C0, dans **l'exemple 2.2** le C2C1C0 = 111, ce qui correspond à 7 en décimal. La position de bit erroné est le numéro 7.

2.2.2 Le codage / décodage de Hamming

2.2.2.1 Codage de Hamming

Ce processus suit les étapes suivantes ;

1. Toutes les positions de bits dans le mot de code qui sont des puissances de 2 (c'est-à-dire des positions 1, 2, 4, 8…) sont pour les bits de parité μ_1, μ_2, μ_3, μ_4, ….

2. Tout le reste des positions (c'est-à-dire les positions 3, 5, 7, 9…) sont pour bits de message m_1, m_2, m_3, m_4, ….

3. Les bits de parité sont régis par les règles suivantes :

- μ_1 vérifie tous les autres bits à partir de laquelle μ_1 est localisé (position 1).
- μ_2 vérifie tous les 2 bits à partir de l'emplacement où se trouve le μ_2 (position 2).
- μ_3 vérifie tous les 4 bits à partir d'où μ_3 est situé (position 4).

- μ_4 vérifie tous les 8 bits à partir d'où μ_4 se trouve (position 8).

4. Définissez un bit de parité sur 1 si le nombre total de 1 dans les bits qu'il vérifie (excluant lui-même) est impair, et à 0 si pair.

La méthode est distinguée au tableau 2.5 Le × signe signifie que le bit de parité vérifie le bit dans cette position. Par exemple, la parité μ_2 vérifie les bits dans les positions 2, 3, 6. Donc

$\mu_1 = m_1 \oplus m_2 \oplus m_4 \oplus \ldots$, La zone dans le tableau correspond au code de (7,4) Hamming [14].

2.2.2.2 Décodage de Hamming

Pour expliquer le processus du décodage Hamming ; nous donnons l'exemple suivant :

1. Selon le tableau 2.5, calculer les bits de parité à partir des bits reçus.
2. Ajouter un bit à partir de XOR de bit à bit.

$$\mu_1' = \mu_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11} \quad \textbf{(2.1)}$$

$$\mu_2' = \mu_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11} \quad \textbf{(2.2)}$$

$$\mu_3' = \mu_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8 \oplus m_9 \oplus m_{10} \oplus m_{11} \quad \textbf{(2.3)}$$

$$\mu_4' = \mu_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 \oplus m_8 \oplus m_{10} \oplus m_{11} \quad \textbf{(2.4)}$$

Position de bit		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu du mot de code		μ_1	μ_2	m_1	μ_3	m_2	m_3	m_4	μ_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}
Bit de parité	μ_1'	×		×		×		×		×		×		×		×
	μ_2'		×	×			×	×			×	×			×	×
	μ_3'				×	×	×	×					×	×	×	×
	μ_4'								×	×	×	×	×	×	×	×

Tableau 2.5 : *Le choix des bits pour calculer les bits de parité* [14].

3. Additionner le résultat, ce qui donne la position où se trouve le bit reçu erroné.
4. Compléter le bit dans la position du bit erroné [14].

Exemple 2.3 :

Supposons le code Hamming (7,4) avec quatre mots en transmission comme suit:

- Les mots à envoyer : (1011), (1001), (0011) et (1011). En se basant sur le tableau 2.6 ; les mots-clés Hamming correspondants seront (0110011), (0011001), (1000011) et (1010011).
- Les quatre mots reçus sont (0110011), (1011001), (1001010) et (0011001), qui ont 0, 1, 2 et 3 erreurs respectivement.

Nous avons résumé les quatre mots selon le tableau 2.6 :

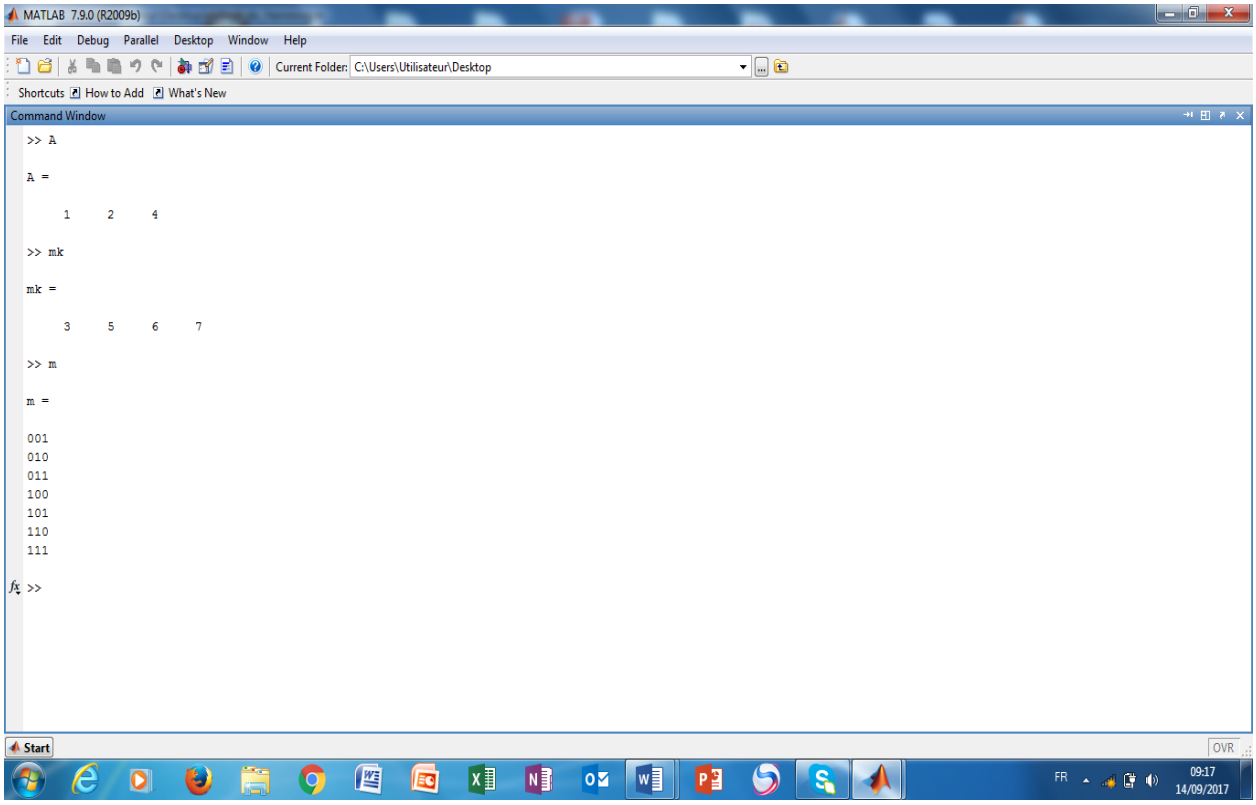
Le mot de code Transmis	**0110011**	**0011001**	**1000011**	**1010011**
Cas	0 erreur	1 erreur	2 erreurs	3 erreurs
Mot reçu	0110011	**1**011001	100**1**01**0**	**0**01**1**0**0**1
Parité reçue ($u_1\ u_2\ u_{3)}$	(010)	(101)	(101)	(001)
Parité calculée (u_1'u_2'u_3')	(010)	(001)	(011)	(001)
($u_1u_2u_3$) $\oplus$ (u_1'u_2'u_3')	(000)	(100)	(110)	(000)
Position d'erreur	–	$=1+0.2^1+0.2^2=1$	$=1+1.2^1+0.2^2=3$	–
Mot décodé	(0110011)	(0011001)	(101100)	(0011001)
La détection d'erreur ?	–	Oui	Oui	Non
La correction d'erreur ?	–	Oui	Non	Non

Tableau 2.6 : *le processus et le résultat du décodage.*

Comme on peut le voir, le code Hamming 7-4 est capable de corriger une erreur ou détecter jusqu'à deux erreurs.

❖ Pratiques sur MATLAB de codage Hamming

Code de Hamming (7-4) : On souhaite envoyer le message1010



2.2.3 Résumé de code Hamming

Le code Hamming présente un mécanisme efficace pour la détection et la correction de l'erreur. Cependant, sa capacité est très limitée en détection et plus encore en correction. Nous avons pu démontrer que le code Hamming arrive à détecter jusqu'à deux erreurs en maximum et qu'il peut seulement corriger une seul erreur. En fait, ce code de Hamming n'est pas tellement performant et ne peut être employé aux environnements ayant un taux d'erreur important ; le cas des transmissions dans les réseaux de la téléphonie mobile, la télévision numérique et toutes transmissions spatiales caractérisées par un taux d'erreurs considérable en rajoutant à cela le phénomène d'effacement qui fait un défi supplémentaire des codes correcteurs d'erreurs.

Les codes de Reed-Solomon sont des codes correcteurs d'erreurs utilisés dans tous les domaines requérant des données fiables. Typiquement, dans les communications spatiales, télévision numérique et stockage de données. Les codes de Reed-Solomon permettent de corriger des erreurs et des effacements grâce à des symboles de contrôle ajoutés après l'information [16]. Aussi, la nature de ces codes a permis d'aboutir à des algorithmes de codage et de décodage très performants y'a compris dans le cas de codes en blocs de grande taille. Nous allons par la suite entamer l'étude sur les codes RS.

2.3 Les codes de Reed-Solomon (RS)

Les codes de Reed-Solomon (RS) sont développés par ***Irving Reed*** et ***Gustave Solomon*** en 1958 [7]. Ces codes sont certainement les codes par blocs les plus utilisés pour la correction d'erreurs dans les CD, les DVD et la plupart des supports de données numériques, Ils sont très utilisés car ils sont puissants du point de vue de la capacité de protection [18]. Il s'agit de codes adaptés à la correction des paquets d'erreurs [10].

Les messages sont divisés en blocs et on a ajouté des informations redondantes à chaque bloc permettant ainsi de diminuer la possibilité de la retransmission. La longueur du bloc dépend de la capacité du codeur. Avant de procéder aux mécanismes de codage/ décodage, on précède par introduire quelques notions élémentaires.

2.3.1 Les champs de Galois

La dénomination « champ de Galois » provient du mathématicien français Galois qui a fondé les lois et les propriétés fondamentales de ces champs mathématiques. Ils sont très utilisés dans la cryptographie ainsi que pour la reconstruction des données.

Nous distinguons deux types de champs ; les champs finis et les champs infinis. Les « champs de Galois » finis sont des ensembles d'éléments fermés sur eux-mêmes où toutes opérations d'addition et de multiplication de deux éléments appartiennent d'un champ de Galois résultent toujours un élément du même champ fini [17,18].

2.3.1.1 Eléments du champ de Galois

Un « champ de Galois » consiste en un ensemble de nombres, ces nombres sont constitués à l'aide de l'élément base « α » donné comme suit :

$0,1,\alpha,\alpha^2,\alpha^3,\ldots\ldots\ldots\alpha^{N-1}$

En prenant N =2^m-1, on forme un ensemble de 2^m éléments. Le champ est alors noté GF (2^m). Ce dernier est formé à partir du champ de base GF (2) et contiendra des multiples des éléments simples de GF (2).

En additionnant les puissances de α, chaque élément du champ peut être représenté par une expression polynomiale de type :

$$\alpha^{m-1}x^{m-1} + \alpha^{m-2}x^{m-2} + \alpha^{m-3}x^{m-3} + \cdots + \alpha x + \alpha^0 \quad \textbf{(2.5)}$$

Avec

$\alpha^{m-1}, \alpha^{m-2}, \alpha^{m-3}, \ldots\ldots, \alpha^0$: les éléments de base du GF (2)

Sur les « champs de Galois », on peut effectuer toutes les opérations de base. L'addition dans un champ fini GF (2) correspond à faire une addition modulo 2, donc l'addition de tous les éléments d'un « champ de Galois » dérivés du champ de base sera une addition modulo 2 (XOR). La soustraction effectuera la même opération qu'une addition donc, la fonction logique « XOR ». La multiplication et la division seront des opérations modulo « grandeur du champ », donc mod (2^m-1) [19].

Les éléments d'un « champ de Galois » de *GF* (2^4) sont :

Eléments	Formes polynomiales	Formes binaires	Formes décimales
0	0	0000	0
1	1	0001	1
α	α	0010	2
α^2	α^2	0100	4
α^3	α^3	1000	8
α^4	$\alpha+1$	0011	3
α^5	$\alpha^2+\alpha$	0110	6
α^6	$\alpha^3+\alpha^2$	1100	12
α^7	$\alpha^3+\alpha+1$	1011	11
α^8	α^2+1	0101	5
α^9	$\alpha^3+\alpha$	1010	10
α^{10}	$\alpha^2+\alpha+1$	0111	7
α^{11}	$\alpha^3+\alpha^2+\alpha$	1110	14
α^{12}	$\alpha^3+\alpha^2+\alpha+1$	1111	15
α^{13}	$\alpha^3+\alpha^2+1$	1101	13
α^{14}	α^3+1	1001	9

Tableau 2.7 : éléments de *GF* (2^4) [16].

2.3.1.2 L'addition

Les additions sont définies dans le champ de Galois GF (2^m), donc pour additionner deux éléments, on prendra la notation binaire de chaque élément et on fait l'addition modulo2. L'addition modulo2 est une opération logique définie par l'opérateur logique « XOR » bit à bit [20].

$$(\alpha_{m-1}x^{m-1} + \cdots + \alpha_1 x^1 + \alpha_0 x^0) + (b_{m-1}x^{m-1} + \cdots b_1 x^1 + b_0 x^0) = c_{m-1}x^{m-1} + \cdots c_1 x^1 + c_0 x^0 \quad \textbf{(2.6)}$$

Avec :

$$c_i = \alpha_i + b_i \qquad 0 \leq i \leq m-1$$

Si $\alpha_i = b_i$ $\quad c_i = 0$

Si $\alpha_i \neq b_i$ $\quad c_i = 1$

Exemple 2.4:

- 2+2=0 0010+0010=0000
- 10+13=7 1010+1101=0111

2.3.1.3 La soustraction

La soustraction dans un *GF* (2^m) s'effectue de la même manière que l'addition et dans le même champ, c.-à-d. une opération logique « XOR » [16].

$$(\alpha_{m-1}x^{m-1} + \cdots + \alpha_1 x^1 + \alpha_0 x^0) - (b_{m-1}x^{m-1} + \cdots b_1 x^1 + b_0 x^0) = c_{m-1}x^{m-1} + \cdots c_1 x^1 + c_0 x^0 \quad \textbf{(2.7)}$$

Avec :

$$c_i = \alpha_i - b_i \qquad 0 \leq i \leq m-1$$

Si $\alpha_i = b_i$ $\quad c_i = 0$

Si $\alpha_i \neq b_i$ $\quad c_i = 1$

Exemple 2.5:

- 1-1=0 0001-0001=000
- 8-6=14 1000-0110=1110

2.3.1.4 La Multiplication

Les multiplications utilisées dans les codes de Reed-Solomon sont des multiplications dans le champ de Galois *GF* (2^m). La multiplication dans le champ de Galois est une opération modulaire, c'est-à-dire que la multiplication entre deux éléments d'un champ fini donnera toujours un élément dans le même champ [20].

Exemple 2.6: $10 \times 13 = 11$

Tel que : $10 = \alpha^9$ et $13 = \alpha^{13}$, alors : $10 \times 13 = \alpha^9 \times \alpha^{13} = \alpha^{(9+13)\bmod(15)}$

$= \alpha^{(22)\bmod(15)} = \alpha^7$ et $\alpha^7 = 11$.

2.3.2 Propriétés des codes Reed-Solomon

Ces codes ont une propriété importante, ils sont linéaires et font partie des codes **BCH**. Le codeur prend k symboles de donnée (chaque symbole contenant s bits) et calcule les informations de contrôle pour construire n symboles, ce qui donne n-k symboles de contrôle. Le décodeur peut corriger au maximum t symboles, où **2t=n-k.** Le diagramme ci-dessous montre une trame constituée par le codeur Reed-Solomon :

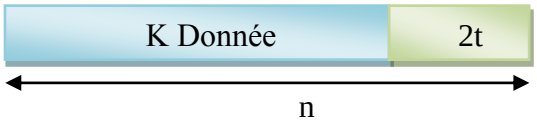


Figure 2.1: *mot-code de Reed-Solomon.*

La longueur maximale d'un code de Reed-Solomon est définie comme :

$$n = k+2t=2^s-1 \quad \textbf{(2.8)}$$

Avec :

k: nombre de symboles de données

$2t$: nombre de symboles de contrôle

s: nombre de bits par symbole

La distance minimale d'un code Reed-Solomon est :

$$d_{min}= 2t+1 \quad \textbf{(2.9)}$$

Les codes de Reed-Solomon sont des codes non-binaires et les codes sont représentés sur des champs de Galois *GF* (2^m) non sur *GF* (2) [16].

Exemple 2.7 :

Prenons un code de Reed-Solomon RS (15,9), que l'on utilisera par la suite pour tous les autres exemples. L'objectif est de découvrir combien de bits sont utilisés pour chaque symbole et combien d'erreurs peut-on corriger.

RS (n, k) =RS (15,9) : n indique la longueur totale d'un bloc de Reed-Solomon ; 15 symboles dans ce cas, et k indique la longueur du bloc d'information ; 9 symboles dans cet exemple.

La capacité de correction des erreurs du système est :

$$2t=n-k=15-9=6$$

Donc :

$$t=\frac{n-k}{2}=3$$

Ce code permettra de corriger 3 symboles. Le nombre de bits s par symbole est :

$$n=2^s-1$$

$$S=\frac{\ln(n+1)}{\ln(2)}=\frac{\ln(16)}{\ln(2)}$$

Le nombre de bits utilisés pour coder les symboles est donc 4 bits. Ce qui nous amène à utiliser un « champ de Galois » de *GF* (2^4).

2.3.3 Le codage de Reed-Solomon

Le codage avec les codes de Reed-Solomon est effectué de la même façon que le codage à l'aide du CRC. La seule différence est que les codes de Reed-Solomon sont non-binaires (Reed-Solomon *GF* (2^m)), alors que le CRC est binaire, (*GF* (2)) [16].

2.3.3.1 La théorie du codage

L'équation clé définissant le codage systématique de Reed-Solomon (n, k) est

$$C(x) = i(x)\, x^{n-k} + [i(x)\, x^{n-k}] \bmod g(x) \qquad \textbf{(2.10)}$$

Avec:

C(x): polynôme du mot-code, degré $n-1$

i (x) : polynôme d'information, degré $k-1$

$[i(x)\, x^{n-k}] \bmod g(x)$: polynôme de contrôle, degré $n-k-1$

g(x) : polynôme générateur, degré $n-k$

Le codage systématique signifie que l'information est codée dans le degré élevé du mot de code et que les symboles de contrôle sont introduits après les mots d'information [16].

2.3.3.1.1 Polynôme générateur

Soit *m* un entier positif strictement supérieur à 2. On pose *t* un entier positif inférieur à $2^m - 1$ et c un élément primitif de GF (2^m). On appelle g(x) le polynôme de degré le plus faible ayant les valeurs $\{\alpha, \alpha^2, \alpha^3 \ldots, \alpha^{2t}\}$ comme racines [16,21].

$$g(x) = \prod_{i=1}^{2t}(x - \alpha^i) \quad \textbf{(2.11)}$$

$$g(x) = (x- \alpha^1)(x-\alpha^2)(x - \alpha^3) \ldots\ldots(x - \alpha^{2t}) \quad \textbf{(2.12)}$$

Exemple 2.8: Calcul des coefficients du polynôme générateur pour RS (15,9) :

On veut calculer les coefficients du polynôme générateur pour le calcul des symboles de contrôle d'un code de RS (15,9) qui peut corriger 3 erreurs, comme nous avons vu précédemment dans l'exemple **(2.7).**

La forme générale du polynôme générateur est donnée dans [16] :

$$g(x) = (x - \alpha^1)(x -\alpha^2)\ (x - \alpha^3) \ldots\ldots(x - \alpha^{2t}) \quad \textbf{(2.13)}$$

En développant l'équation (**2.13)**, on trouve :

$$g(x) = (x-\alpha^1)\ (x-\alpha^2)\ (x-\alpha^3)\ (x-\alpha^4)\ (x-\alpha^5)\ (x-\alpha^6) \quad \textbf{(2.14)}$$

$$= (x^2+\alpha^5 x+\alpha^3)\ (x^2+\alpha^7 x+\alpha^7)\ (x^2+\alpha^9 x+\alpha^{11})$$

$$= (x^4+\alpha^{13}x^3+\alpha^6 x^2+\alpha^3 x+\alpha^{10})\ (x^2+\alpha^9 x+\alpha^{11})$$

$$= x^6+x^5(\alpha^{13}+\alpha^9) +x^4(\alpha^6+\alpha^7+\alpha^{11}) +x^3(\alpha^3 + 1 + \alpha^9) +x^2(\alpha^{12}+\alpha^{10}+\alpha^2) +x\ (\alpha^4+\alpha^{14}) +\alpha^6$$

$$= x^6+\alpha^{10}x^5+\alpha^{14}x^4+\alpha^4 x^3+\alpha^6 x^2+\alpha^9 x+\alpha^6$$

❖ Pratiques sur MATLAB

```
% Polynôme générateur de RS (15,9)
n=15;
k=9;
m=4;
g=rspoly(n,k,m)
g =
    6    9    6    4   14   10    0
```

Le résultat des racines de Polynôme générateur de RS (15,9) et :**(α^6 α^9 α^6 α^4 α^{14} α ^{10}1)**.

2.3.3.1.2 Le mot de code

Le polynôme *C* de degré n-l dont les coefficients appartiennent à *GF(2)*. Si $\{\alpha, \alpha^2..., \alpha^{2t}\}$ sont des racines de C(x), alors, de manière évidente, C(x) est aussi divisible par g(x) [21]. Le code RS (n, k) peut être codé comme un code binaire BCH. La seule différence est que les multiplications et les ajouts doivent maintenant être réalisés dans GF(2^m). Pour le codage systématique d'un polynôme de message *m(x)* avec *k* coefficients ayant le *k* symboles de message sur *GF(2^m)* ; le polynôme de vérification de parité *m(x)* est le reste du polynôme de messages décalés $X^{n-k}m(x)$ divisé par le polynôme générateur *g(x)* [14], alors ;

$$\mu(x) = X^{n-k}m(x) \bmod g(x) \quad \textbf{(2.15)}$$

Le mot de code est :

$$C(x) = \mu(x) + X^{n-k}m(x) \quad \textbf{(2.16)}$$

Exemple 2.9:

On considère le code linéaire cyclique Reed-Solomon défini dans GF (2^3) pour corriger 2t (t=2) erreurs. Les paramètres de ce code sont :

$n=2^p-1=2^3-1=7$. Et $k=2^p-1-2t=8-1-4=3$.

Le polynôme générateur de ce code vérifie la formule :

$G(x) = \prod_0^3(x-\alpha^i) = x^4+\alpha^2x^3+\alpha^5x^2+\alpha^5x+\alpha^6$ **(2.17)**

Supposons maintenant qu'on veut coder le mot $m(x)= x^2+ \alpha x+\alpha^2$. Alors, pour les codes à logique majoritaire, on calcule :

$\mu(x) = X^{n-k}m(x) \bmod g(x)$

$$\mu(x) = X^4(x^2+\alpha x+\alpha^2) mod (x^4+\alpha^2x^3+\alpha^5x^2+\alpha^5x+\alpha^6) \quad \textbf{(2.18)}$$

$$
\begin{array}{l|l}
x^6 + ax^5 + a^2x^4 & x^4 + a^2x^3 + a^5x^2 + a^5x + a^6 \\
-x^6 - a^2x^5 - a^5x^4 - a^5x^3 - a^6x^2 & \hline \\
\hline & x^2 + a^4x + a^4 \\
a^4x^5 + a^3x^4 + a^5x^3 + a^6 & \\
-a^4x^5 - a^6x^4 - a^9x^3 - a^9x^2 - a^{10}x & \\
\hline & \\
a^4x^4 + a^3x^3 + a^0x^2 + a^3x & \\
-a^4x^4 - a^6x^3 - a^9x^2 - a^9x^1 - a^{10} & \\
\hline & \\
a^4x^3 + a^6x^2 + a^5x + a^3 &
\end{array}
$$

Donc : $\mu(x) = \alpha^3 + \alpha^4x^3 + \alpha^5x + \alpha^6x^2$

Le mot de code est :

$$C(x) = \mu(x) + X^{n-k}m(x)$$

$$C(x) = \alpha^3 + \alpha^4x^3 + \alpha^5x + \alpha^6x^2 + x^2 + \alpha x + \alpha^2$$

Le mot de code C(x) est alors : $[\boldsymbol{\alpha^3\ \alpha^4\ \alpha^5\ \alpha^6\ 1\ \alpha\ \alpha^2}]$

❖ Application sur MATLAB

```
% Le mot de code de Reed-Solomon (7,3)
n=7;
k=3;
m= [0 1 2];
c=rsenco(m,n,k,'power')
c =
   3   4   5   6   0   1   2
```

Le résultat des racines de mot de code RS(7,3) **et :** $(\boldsymbol{\alpha^3\ \alpha^4\ \alpha^5\ \alpha^6\ 1\ \alpha\ \alpha^2})$

2.3.4 Le décodage de Reed-Solomon

L'idée de base du décodeur RS est de détecter une séquence erronée avec peu de termes, qui est sommée aux données reçues, donnera lieu à un mot-code valable.

Plusieurs étapes sont nécessaires pour le décodage de ces codes :

- Calcul du syndrome,
- Calcul des polynômes de localisation des erreurs et d'amplitudes,
- Calcul des racines, évaluations des deux polynômes, et sommation du polynôme constitué et du polynôme reçu pour reconstituer l'information de départ sans erreur [16].

Le diagramme ci-dessous (Figure 2.2) explique les étapes de décodage

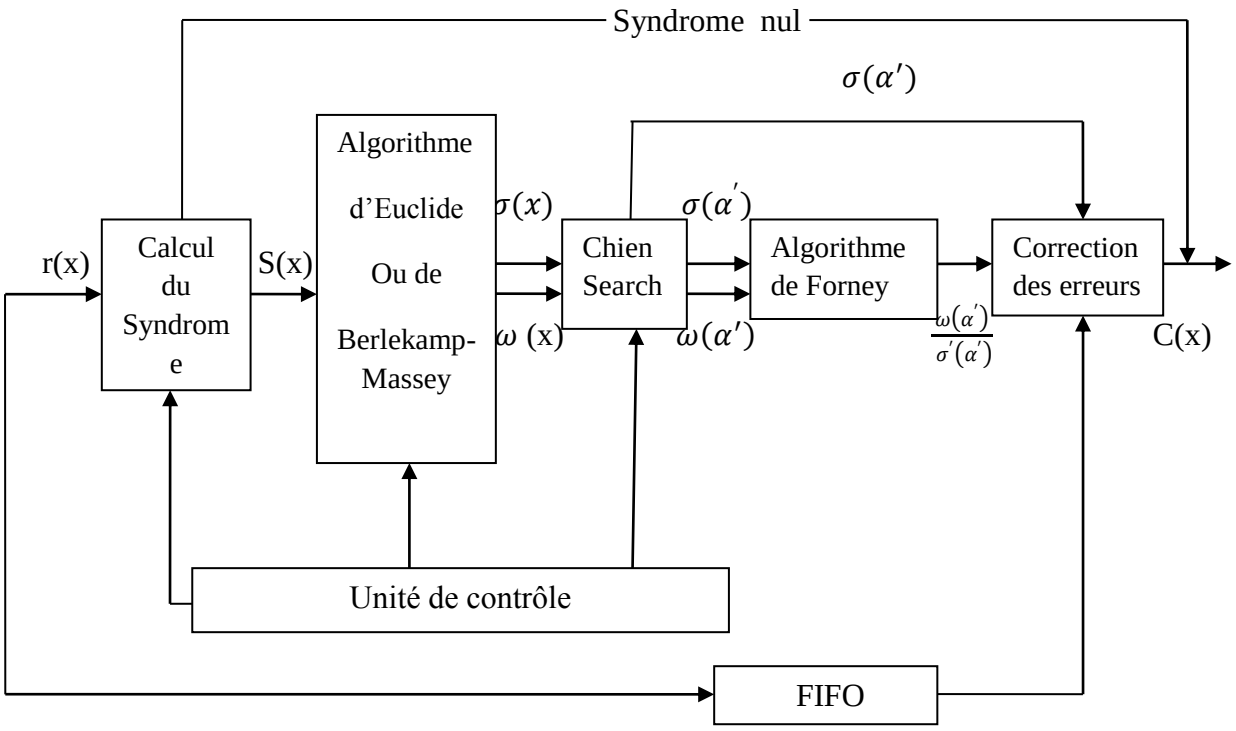


Figure 2.2 : *Schéma du décodage*.

r(x) : Mot de code reçu

S(x) : Syndrome calculé

ω (x) : Polynôme d'amplitude des erreurs

$\omega(\alpha')$: Polynôme d'amplitude des erreurs, pour tous les éléments compris dans *GF* (2^m)

$\sigma(x)$: Polynôme de localisation des erreurs

$\sigma(\alpha')$: Polynôme de localisation des erreurs, pour tous les éléments compris dans *GF* (2^m)

$\sigma'(\alpha')$: Dérivée du $\sigma(\alpha')$

$\frac{\omega(\alpha')}{\sigma'(\alpha')}$ **:** Division entre le polynôme d'amplitude et $\omega(\alpha')$

c(x) : Mot de code reconstitué.

2.3.4.1 Les phases du décodage

Considérons un code de Reed-Solomon *c(x)* correspond au code transmis et soit *r* (*x)* le code que l'on reçoit. Le polynôme d'erreurs introduit par le canal est défini comme :

$$\begin{aligned} e(x) &= r(x) - c(x) = r(x) + c(x) \\ &= e_0 + e_1 x + \cdots + e_{n-1} x^{n-1} \end{aligned} \quad \textbf{(2.19)}$$

Supposant que le polynôme des erreurs contient des erreurs dans les positions $x^{j_1}, x^{j_2}, \ldots, x^{j_v}$ avec $0 \leq j_1 < j_2 < \cdots < j_v \leq n-1$. On peut donc définir le polynôme des erreurs comme suit:

$$e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + \cdots + e_{j_v} x^{j_v} \quad \textbf{(2.20)}$$

Avec :

$e_{j_1}, e_{j_2}, \ldots\ e_{j_v}$: Valeurs d'amplitude des erreurs

$x^{j_1}, x^{j_2}, \ldots\ x^{j_v}$: La position des erreurs

À partir du polynôme *r* (*x)* reçu, on peut calculer le polynôme du syndrome *S* (*x)* qui nous indiquera la présence d'éventuelles erreurs. Si tous les coefficients du syndrome sont nuls, alors les étapes suivantes du décodage n'ont pas lieu d'êtres, car le mot de code reçu ne contiendra pas d'erreurs. Par contre, si le syndrome est non nul, on devra calculer le polynôme de localisation des erreurs et le polynôme d'amplitude des erreurs. Il y a plusieurs méthodes de calcul de ces deux polynômes, dans le cadre de ce projet on n'étudiera qu'une seule méthode ; le décodage selon l'algorithme d'Euclide. Une fois les polynômes calculés en utilisant l'algorithme de Forney, on calculera les valeurs à soustraire pour obtenir le mot de code sans erreurs [18].

2.3.4.2 Calcul du Syndrome

Le calcul du syndrome est défini comme le reste de la division entre le polynôme reçu *r(x)* et le polynôme générateur *g(x)*. Le reste indiquera la présence d'erreurs. Comme l'opération division est toujours une opération complexe par rapport à des sommes et des additions, on est amené à rechercher une autre méthode pour le calcul du syndrome [19]. Le calcul du syndrome peut aussi être effectué par un processus itératif. Avant de pouvoir calculer le polynôme du syndrome, on doit attendre que l'on ait reçu tous les éléments du polynôme *r* (*x)*. Comme :

$$S_i = r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i) \quad \textbf{(2.21)}$$

À partir de cette relation on peut définir les différentes équations :

$S_1 = e_{j_1}\alpha^{j_1} + e_{j_2}\alpha^{j_2} + \cdots + e_{j_v}\alpha^{j_v}$

$S_2 = e_{j_1}\alpha^{2j_1} + e_{j_2}\alpha^{2j_2} + \cdots + e_{j_v}\alpha^{2j_v}$

….

$S_{2t} = e_{j_1}\alpha^{2tj_1} + e_{j_2}\alpha^{2tj_2} + \cdots + e_{j_v}\alpha^{2tj_v}$

Le syndrome sous forme polynomiale est de forme :

$$S(x) = \ldots + S_{2t+1}x^{2t} + S_{2t}x^{2t-1} + \cdots + S_2x + S_1 \quad \textbf{(2.22)}$$

Seuls les premiers 2*t* symboles du syndrome sont connus. Si le code *r* (*x*) n'est pas affecté par des erreurs alors tous les coefficients du syndrome seront nuls (*r*(*x*) = *c*(*x*)).

Le schéma ci-dessous montre le calcul du syndrome de façon itérative :

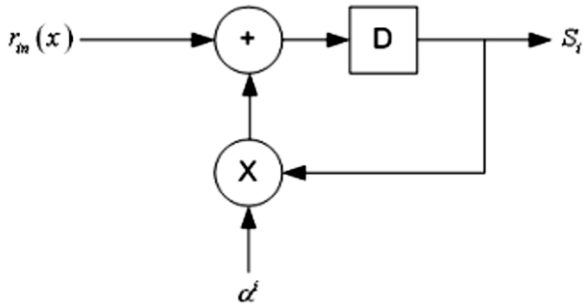


Figure 2.3 : *Schéma pour le calcul du syndrome*.

2.3.4.3 Algorithme d'Euclide

2.3.4.3.1 Généralités sur le théorème d'Euclide

L'algorithme d'Euclide [22] est un algorithme récursif qui permet de trouver le plus grand diviseur commun de deux polynômes $r_0(x)\ et\ r_1(x)$ dans le champ de Galois *GF(q)*. Il existe deux polynômes α(x) et b(x) en *GF(q)* tel que :

$$\text{MCD}\ (r_0(x), r_1(x)) = \alpha(x)\, r_0(x) + b(x)\, r_1(x) \quad \textbf{(2.23)}$$

Où α(x) et b(x) peuvent être calculés selon l'algorithme d'Euclide en donnant deux polynômes non nuls α(x) et b(x) en Gf(q).

L'algorithme d'Euclide fonctionne de la façon suivante :

$$\deg(r_1(x)) \leq \deg(r_0(x))$$
$$\alpha_0(x) = 1, b_0(x) = 0 \quad \textbf{(2.24)}$$
$$\alpha_1(x) = 0, b_1(x) = 1$$

Avec :

$\deg(r_1(x))$: Degré du polynôme $r_1(x)$

$\deg(r_0(x))$: Degré du polynôme $r_0(x)$

Pour i $\geq$ 2, on calcule le quotient et le polynôme restant et la division est effectuée par $r_{i-2}(x)$ et $r_{i-1}(x)$.

$$r_{i-2}(x) = q_i(x)r_{i-1}(x) + r_i \quad \textbf{(2.25)}$$

Avec

$$0 \leq \deg(r_i(x)) < \deg(r_{i-1}(x))$$

$$\alpha_i(x) = \alpha_{i-2}(x) - q_i(x)\alpha_{i-1}(x) \quad \textbf{(2.26)}$$

$$b_i(x) = b_{i-2}(x) - q_i(\mathrm{x})b_{i-1}(x)$$

Les calculs se terminent lorsque :$\deg(r_i) = 0$, et le dernier polynôme non nul indique le plus grand diviseur commun [20].

2.3.4.3.2 Correction des erreurs avec Euclide

Le polynôme de localisation d'erreurs est défini comme dans [19,22] par :

$$\sigma(x) = \prod_{k=1}^{v}(1 - \alpha^{i_k}x)$$
$$= \sigma_v x^v + \sigma_{v-1}x^{v-1} + \cdots + \sigma_1 x + 1 \quad \textbf{(2.27)}$$

Le polynôme d'amplitude des erreurs est calculé de la manière suivante :

$$\omega(x) = \mathrm{S(x)}\,\sigma(x) \quad \textbf{(2.28)}$$

$\sigma(x)$: Polynôme de localisation des erreurs, inconnu à ce stade

S(x) : Polynôme syndrome,

$\omega(x)$: Polynôme d'amplitude, " inconnu à ce stade ".

Comme on connaît seulement 2*t* symboles du polynôme du syndrome ($x^0 \ldots x^{2t-1}$), on devrait limiter le résultat à 2*t* tel que:

$$S(x)\sigma(x) = \omega(x) mod(x^{2t}) \quad \textbf{(2.29)}$$

Cette expression est l'équation clé pour les codes de Reed-Solomon. Si le nombre d'erreurs *v* dans le mot de code transmis C(x) est plus petit ou égale à *t* ; l'équation clé a une seule paire de solution $\sigma(x)$ et $\omega(x)$. Les deux degrés des polynômes doivent respecter la contrainte :

$$\deg(\omega(x)) < \deg(\sigma(x)) \leq t \quad \textbf{(2.30)}$$

L'équation clé peut être résolue selon l'algorithme d'Euclide en appliquant $r_0(x) = x^{2t}$ et $r_1(x) = S(x)$. Le calcul du théorème d'Euclide nous donnera comme solution le polynôme de localisation des erreurs et le polynôme [18].

Le dernier reste de la division nous donnera le polynôme d'amplitude. Le polynôme de localisation des erreurs est donné selon la relation :

$$\sigma_i(x) = \sigma_{i-2}(x) + \sigma_{i-1}(x)Q_i(x) \quad \textbf{(2.31)}$$

Où : $\sigma_i(x) = B_i(x)$

Note : La théorie montre que l'on est obligé d'avoir deux blocs dans l'implémentation hardware. Un bloc qui effectue la division et qui donnera le polynôme d'amplitude des erreurs, et également un bloc de multiplication qui donnera le polynôme de localisation des erreurs [20].

2.3.4.4 Le chien search

Une fois le polynôme de localisation des erreurs calculé, on doit évaluer ses racines et sa dérivée. L'évaluation des racines est effectuée avec l'algorithme appelé « Chien Search » qui est du type « brute force », c'est-à-dire, qu'il évalue toutes les solutions possibles. À la sortie de ce bloc, on obtiendra une séquence de symboles, et lorsque ces derniers seront nuls ; ceci nous indique qu'une racine a été détectée [20].

2.3.4.5 L'Algorithme de Forney

Cet algorithme permet de construire le polynôme d'erreurs *e*(*x*) et l'additionner avec le polynôme reçu *r* (*x*) pour reconstruire le polynôme *c*(*x*). Pour le calcul du polynôme *e*(*x*), les polynômes $\sigma(\alpha')$, $\sigma'(\alpha')$, $\omega(\alpha')$ sont nécessaires. Le polynôme de localisation des erreurs et sa dérivée sont déjà évalués pour les différentes valeurs de α, il nous reste qu'à évaluer $\omega(\alpha')$ .Une fois les différentes valeurs de $\omega(\alpha')$ calculées, on applique l'algorithme de Forney [18]. Cet algorithme est défini comme suit :

$$e_i = \frac{\omega(\alpha^i)}{\sigma'(\alpha^i)} \quad \textbf{(2.32)}$$

Où

$\omega(\alpha^i)$: Polynôme d'amplitude évalué pour les valeurs de *GF* (2^4).

$\sigma'(\alpha^i)$: Dérivée du polynôme de localisation des erreurs pour les valeurs de *GF* (2^4).

2.3.5 La correction des effacements

Les codes de Reed-Solomon sont non seulement utilisés pour la correction des erreurs, mais également pour corriger les effacements. Un effacement suit le même principe que lorsqu'on efface une lettre dans un mot à l'aide d'un effaceur. La lettre effacée dans le mot n'est pas connue, mais la position de celle-ci l'est. Les codes de Reed-Solomon permettent de corriger deux fois plus d'effacements que d'erreurs. La séquence de décodage est presque la même que celle utilisée pour la correction des erreurs, la seule différence est qu'avant de calculer les syndromes, on doit substituer dans le polynôme reçu $r(x)$ les effacements avec des '0' avant de procéder au calcul du syndrome lui-même. La première opération à effectuer pour le décodage des erreurs et des effacements est l'évaluation du polynôme de localisation des effacements [18]. On doit noter ici que la capacité de la correction sera plus grande que dans l'absence des effacements 'erasures' lorsqu'on devrait ajouter f effacements aux v erreurs.

2.3.6 La probabilité d'erreur et le taux d'erreur

Il existe deux mesur23es importantes pour un code de canal [23]:

2.3.6.1 Le taux d'erreur binaire (BER) : Le taux d'erreur binaire constitue le paramètre primaire décrivant la qualité de la transmission numérique. II est définit comme le rapport entre les bits erronés et le nombre total de bits reçus. Ce taux détermine le nombre d'erreurs apparues avant la modulation et juste après la démodulation, il augmente à cause des perturbations : équipement ou réseau défectueux, pointage incorrect d'une antenne et longueur de canal [24].

2.3.6.2 La probabilité d'erreur : Les codes avec une probabilité d'erreur inférieure sont plus souhaitables. Pour les codes de correction d'erreur, la probabilité d'erreur de bloc est importante. Pour les codes de détection d'erreur, la probabilité d'erreur non détectée est importante [23].

2.3.7 La probabilité d'erreur de bit/bloc

2.3.7.1 La probabilité d'erreur en bit : $P_b(k_n)$ est la probabilité d'erreur de bit entre le message et le message décodé [23]. Ça exprime le taux des bits erronés en référence du message original.

2.3.7.2 La probabilité d'erreur en bloc : $P_e(k_n)$ est la probabilité d'une erreur de décodage du mot reçu en fonction de bloc. Alors, il est probable que le décodeur choisit le mot de code non correct lors de l'application de l'algorithme de décodage [23].

Exemple 2.10 :

On considérer le message 000 est transmis sous forme de mot de code 000000. Des erreurs de deux bits se produisent dans les deux derniers bits et le mot reçu est 000011. Le décodeur de canal sélectionnera ensuite le mot de code le plus proche qui est 100011. Ainsi, Une erreur de bloc s'est produite. Si cela se produit toujours, alors on est à faire à une $P_e(k_n)$. Cependant, le mot de code 100011 est décodé en tant que message 100 qui n'a que le premier bit de pois le plus faible et les restes deux bits sont corrects selon le message original. Si cela se produit souvent alors :

$$P_b(k_n) = 1/3 \tag{2.33}$$

$$P_b(k_n) = 0.33$$

2.4 Conclusion

Dans la première partie de ce chapitre nous avons étudié en détails le principe de fonctionnement du codage Hamming où on a décrit toutes les étapes essentielles pour les opérations de codage et de décodage. Pour le décodage, nous avons utilisé deux méthodes différentes pour corriger et restituer l'information erronée. Seulement, nous avons constaté que le code Hamming présente de faibles performances en termes de capacité de détection et de correction. Cependant, malgré la complexité des calculs que présentent ; les codes RS offrent une grande capacité dans la détection et la correction des erreurs ainsi que les effacements. Chose qui priorise d'avantage ce code pour l'employer actuellement dans presque toutes les applications multimédia, transmissions numériques et spatiales en précisant en particulier les communications à base de protocoles à accès aléatoire où les taux d'erreurs sont plus importants.

Chapitre 3 : Les performances du code RS

3.1 Introduction

Le codage RS et comme les autres familles des codes correcteurs d'erreurs, varie d'un milieu à un autre selon les caractéristiques que présente chaque milieu ou environnement. En effet, les paramètres du codage et les conditions du support de transmission rapportent sur les performances du codage RS employé. Dans les deux chapitres précédents, nous avons exposé le principe de fonctionnement du codage (RS) et ses caractéristiques vis à vis les différents types de canal. Et dans ce chapitre nous allons examiner le comportement du codage (RS) par rapport à la variation de ses àdifférents paramètres.

3.2 Probabilité d'erreur en bloc pour un code de correction d'erreur t-bit

La probabilité d'erreurs en bloc peut s'exprimée par l'équation suivant [23]:

$$P_e(k_n) = 1 - \sum_{i=0}^{t} \binom{n}{i} q^i (1-q)^{n-i} \quad \textbf{(3.1)}$$

Où *Kn* présente un codeur de n bits. Tel que la probabilité d'erreurs exactes de *i* bits erronés entre n bits du mot de code est égale à :

$$\binom{n}{i} q^i (1-q)^{n-i} \quad \textbf{(3.2)}$$

Aussi, un code correcteur d'erreurs t-bit est capable de gérer des erreurs de t bit. Donc, la probabilité d'aucune erreur de décodage se produit est [23] :

$$P_c(k_n) = \sum_{i=0}^{t} \binom{n}{i} q^i (1-q)^{n-i} \quad \textbf{(3.3)}$$

Et la probabilité d'erreur est alors:

$$p_e(k_n) = 1 - p_c(k_n) \quad \textbf{(3.4)}$$

Supposons un canal CBS avec q=0.001. Alors, dans le cas d'un canal sans code correcteur ; nous avons :

$$p_b(k_1) = 0{,}001 = 1 \times 10^{-3} \quad \textbf{(3.5)}$$

Et, le rendement :

$$R=1 \quad \textbf{(3.6)}$$

Pour démontrer la validité des équations précédentes, nous estimons utiliser un mot de code de n=3. Alors, par un code correcteur de 1 bit, nous aurons :

$$P_e(k_3) = 1 - \sum_{i=0}^{t} \binom{n}{i} q^i (1-q)^{n-i} \quad \textbf{(3.7)}$$

$$= 1 - \binom{3}{0}(0.999)^3 - \binom{3}{1}(0.999)^2(0.001)^1$$

$$= 3 \times 10^{-6}$$

À partir de k=1, on aura :

$$p_b(k_3) = p_e(k_3) = 3 \times 10^{-6} \quad \textbf{(3.8)}$$

Nous remarquons que la probabilité est diminuée de 1×10^{-3} à 3×10^{-6} , ainsi que le taux de codage qui a diminué de 1 à $1/3$ [23].

Nous allons maintenant évaluer la variation de la probabilité d'erreurs en fonction de nombre de bits erronés d'une façon plus généralisée en faisant des simulations par Matlab ; afin de voir encore le comportement du codage RS.

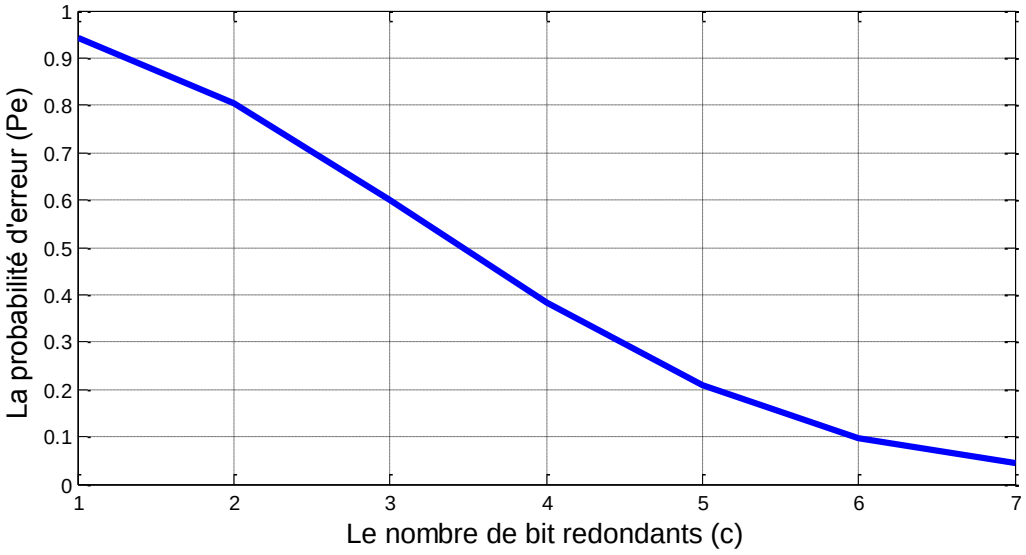


Figure 3.1: *La variation de la probabilité d'erreur* $\boldsymbol{P_e}$ *en fonction de nombre de bit redondance* $\boldsymbol{c}$.

La figure 3.1 représente la variation de la probabilité d'erreur en fonction des bits redondance $\boldsymbol{c}$, tel que la probabilité d'erreur varie inversement proportionnel par rapport à la variation du nombre de bits redondance. Nous remarquons que la probabilité d'erreurs ($\boldsymbol{p_e}$) atteint son valeur maximale (presque 0.98) lorsque le nombre de bits redondants égale à 1, et prend en diminution pour atteindre son valeur minimale (jusqu'à 0.05) lorsque le nombre de bits redondant égale à 7. Nous concluons que la capacité de correction chez le codage RS devienne plus importante avec un nombre de bits de contrôle plus important aussi. Alors, plus le nombre des bits redondants est

supérieur plus la capacité est meilleure. Cependant, l'augmentation du nombre de bits redondants affaiblie en contrepartie la vitesse de transmission lorsque la quantité des bits transmis sera importante. En ajoutant à cela la complexité des opérations de codage et décodage qui peut nuire aux performances du codeur.

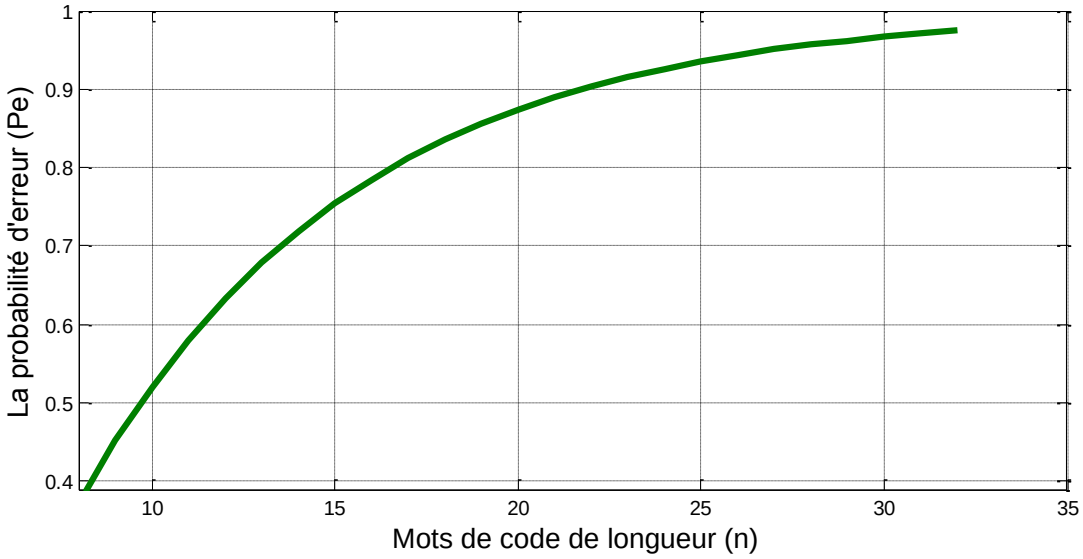


Figure 3.2 : *La variation de la probabilité d'erreur* $\boldsymbol{P_e}$ *en fonction de la longueur* ***n*** *du mot de code.*

La figure 3.2 représente la variation de la probabilité d'erreur en fonction de mots de code **(n).** On remarque qu'il y a une relation proportionnelle entre $\boldsymbol{P_e}$ et ***n*.** Tel que pour des longueurs petites des mots de code la probabilité d'erreurs est aussi faible. Tandis que pour des longueurs plus grandes la probabilité devient maximale et prend sa stabilité malgré l'augmentation de ***n***. Nous pouvons conclure qu'avec des mots de code de longueurs élevées le codage porte alors de faibles performances, ce qui aggrave par conséquence la transmission des données lorsque la capacité de correction devient plus faible.

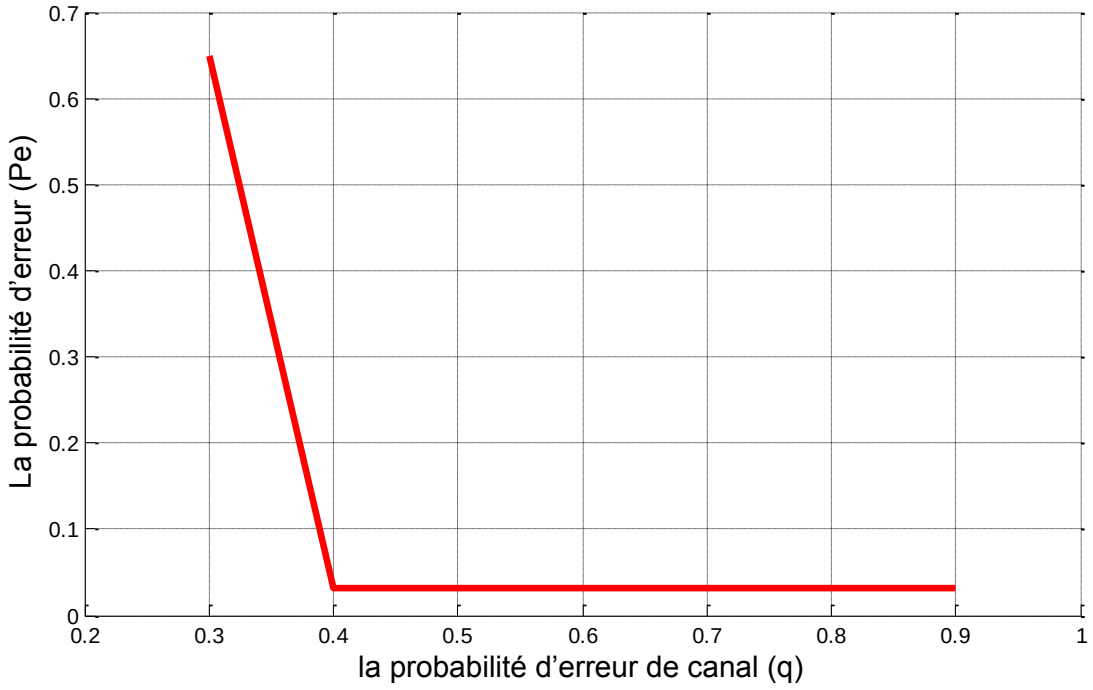


Figure 3.3 : *La variation de la probabilité d'erreur* $\boldsymbol{P_e}$ *en fonction de la probabilité d'erreur de canal* $\boldsymbol{q}$.

La figure 3.3 représente la variation de la probabilité d'erreur en fonction de la probabilité d'erreurs du canal. Elle décrit une variation inversement proportionnelle. Tel que à des valeurs faibles de $\boldsymbol{q}$ nous avons des $\boldsymbol{P_e}$ maximales. Et à partir de q=0.4 nous enregistrons une certaine stabilité de la variation de $\boldsymbol{P_e}$. Cela confirme la validité de l'équation (3.1) qui explique la relation non proportionnelle entre le **q** et le $\boldsymbol{P_e}$. Alors, le code RS présente une forte immunité contre les milieux bruités, tel que pour des canaux présentant un taux d'erreurs élevé le code RS présente offre alors une probabilité d'erreurs plus faible et donc une probabilité de succès plus élevée.

L'auteur dans [25] présente la probabilité d'erreurs en bloc comme suit :

$$p_e \approx \frac{1}{2^m-1}\sum_{i=t+1}^{2^m-1} i\binom{2^m-1}{i} q^i(1-q)^{2^m-1-i} \qquad \textbf{(3.9)}$$

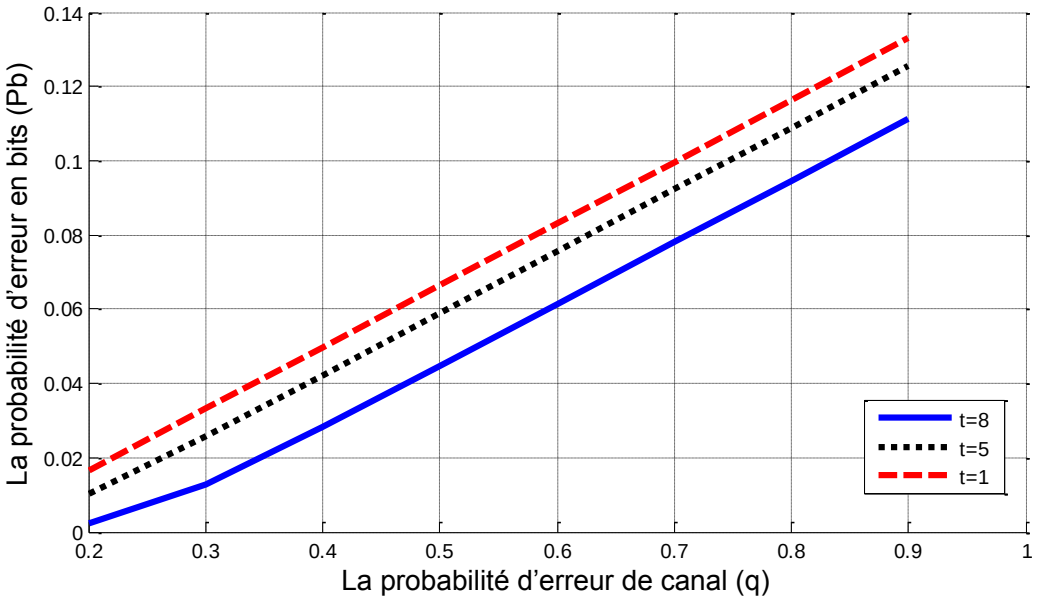
Tel que:$n = 2^m - 1$, alors :

$$p_e \approx \frac{1}{n}\sum_{i=t+1}^{n} i\binom{n}{i} q^i(1-q)^{n-i} \qquad \textbf{(3.10)}$$

Où, Le **t** est la capacité de correction d'erreur, et les symboles sont constitué de **m** bits chacun.

La probabilité d'erreur de bit peut être délimitée par la probabilité d'erreur de symbole pour un type de modulation spécifique. Pour la modulation MFSK avec $M = 2^m$, la relation entre p_b et p_e est la suivante [25] :

$$\frac{p_b}{p_e} = \frac{2^{m-1}}{2^m - 1} \qquad \textbf{(3.11)}$$



La figure 3.4 : *La variation de la probabilité d'erreur en bit* P_b *en fonction de la probabilité d'erreur de canal* q.

La figure 3.4 représente la variation de la probabilité d'erreur en bits P_b en fonction de la probabilité d'erreur du canal q. Nous observons que la variation de P_b augmente parallèlement avec l'augmentation de q, tel que pour une valeur minimum de **q** la probabilité d'erreur est faible et augmente relativement à l'augmentation de q. La simulation montre aussi que la probabilité d'erreurs augmente lorsque le nombre de bits erronés par mot est augmenté. Alors pour avoir de bonnes performances de la transmission ; il est préférable de choisir un canal de transmission plus adéquat et qui offre une faible q.

3.3 Conclusion

Les performances du codage (décodage) RS sont analysées dans ce chapitre. Nous avons entamé des études par des simulations multiples pour pouvoir décrire les caractéristiques d'un codeur/décodeur RS, afin d'évaluer à la fin son comportement, sa capacité et son efficacité dans des milieux de propagation plus au moins bruités. En effet, l'ensemble des simulations faites au sein de ce chapitre nous a permet de savoir comment le codage RS devient plus efficace par un choix plus convenable de nombre de bits redondants et donc la longueur du mot de code suivant la qualité du canal de transmission. Et cela exprime également la capacité de correction chez le décodeur RS. Cette capacité qui est mesurée par la probabilité d'erreurs binaire notée (p_b), et représentée en pratique par le taux d'erreurs binaire.

En fait, nous découvrons une efficacité importante du codage (décodage) RS que présente en particulier lors des canaux de transmission qui ont un taux d'erreurs élevé. Cependant la formulation du mot code est assez délicate et fait créer une sensibilité considérable du codeur (décodeur) RS. Chose qui exige une certaine mesure avant de prendre décision sur la sélection du mot de code en prenant en considération le type de milieu dans lequel la transmission est déroulée pour éviter l'affaiblissement des performances de la transmission lorsque la capacité et le débit serons affaiblies à cause des retransmissions multiples pour réussir la transmission.

Conclusion générale

Les codes de Reed-Solomon présentent de bonnes performances offrant aux systèmes de transmission modernes une bonne qualité de transmission et un rendement acceptable qui avantage ces codes d'êtres leaders et très utilisables dans presque toutes les transmissions et technologies actuelles. Ces codes sont basés sur des algorithmes puissants de codage et de décodage que nous avons traité dans ce mémoire. Pour cela, et pour avoir une référence d'évaluation ; nous avons commencé au premier temps d'étudier le codage (décodage) Hamming dont nous avons présenté dans le premier chapitre des exemples concrets expliquant le principe de fonctionnement de ce codage (décodage) avec des simulations faites sur Matlab ,qui élaborent l'opération de codage et de décodage Hamming. Le programme « codeur Hamming » exécute le codage par l'établissement des bits redondants à travers les bits du mot original pour formuler le mot de code. On doit signaler ici que le programme n'est pas entièrement accompli et qu'une seule partie est exécutable, et qu'on a travaillé jusqu'à l'écriture de ce mémoire pour l'accomplir. Alors, nous avons réussi de bien maitriser le codage (décodage) Hamming et découvrir ses différentes caractéristiques et sa limite en capacité de correction. Pour conclure ce chapitre, nous avons présenté une comparaison entre les codes RS et les codes Hamming afin de récapituler rapidement leurs différentes caractéristiques, avantages et inconvénients.

Une étude approfondie sur les codes RS est présentée dans le deuxième et le troisième chapitre. Des développements mathématiques et plusieurs simulations sont présentés au cours du troisième chapitre. Contrairement aux codes Hamming, les codes RS présentent de bonnes performances et un taux de réussite considérable. Tel qu'en jouant sur ses paramètres de codage (décodage) les codes RS peuvent donner une satisfaction meilleure en termes de qualité et capacité de correction malgré les perturbations et le taux d'erreur que présentent les supports de communication. En effet, cela était l'occasion pour nous dans ce travail de maitriser les secrets du codage RS et de savoir ses points faibles et ses points forts qui nous guident au futur, et nous donnent l'espoir de renforcer ces codes par des idées et techniques supplémentaires, ou de faire modifier quelques détails avec un contrôle adéquat des bits redondants et longueur du mot ou de chercher encore à intégrer des fonctions internes dans les algorithmes de décodage comme l'algorithme d'Euclide pour diminuer les erreurs commises au niveau de ces algorithmes et accélérer par conséquence l'opération de décodage avec plus d'efficacité afin de réussir en mieux notre transmission et d'acquérir des taux de réussite plus élevés avec des performances meilleures.

Les références

[1] M. Côte,"Reconnaissance de codes correcteurs d'erreurs," Thèse de Doctorat, Ecole Polytechnique en Informatique, 2010

[2] J.G. Proakis, M. Salehi, Digital Communications, 5th ed. McGraw-Hill, 2008

[3] G. BERHAULT, "Exploration architecturale pour le décodage de codes polaires," Thèse de Doctorat, université de Bordeaux, France, 2015

[4] M. CUNCHE, "Codes AL-FEC hautes performances pour les canaux à effacements : variations autour des codes LDPC," Thèse de Doctorat, Ecole Doctorale Mathématiques, Sciences et Technologies de L'information(Grenoble), 2010

[5] G. LANDAIS, "Mise en œuvre de crypto systèmes basés sur les codes correcteurs et de leurs cryptanalyses," Thèse de Doctorale, Université Pierre et Marie Curie -Paris VI, 2014

[6] M. Demazure, Cours d'algèbre, Cassini 1997

[7] X. HENOCQ," Contrôle d'erreur pour transmission de flux vidéo temps réels sur réseaux de paquets hétérogènes et variant dans le temps," Thèse de Doctorat, Ecole Mathématique, Télécommunication, Informatique, Système Electroniques (Matisse) ,2002

[8] C. CHRISTOPHE, "Reconnaissance de codes structure des codes quasi-cycliques," Thèse de Doctorat, École Doctorale Science –Technologie-Sante Faculté des Sciences et Techniques XLIM– DMI Equipe PI2C (Limoges), 2009

[9] A. IRINA, "Code correcteurs d'erreurs LDPC structurés," Thèse de Doctorat, Université Laval, France, 2010

[10] J. HOUSSEIN, "Conception architecturale haut débit et sûre de fonctionnement pour les codes correcteurs d'erreurs," Thèse Doctoral, École Doctorale IAEM - Lorraine, 2009

[11] R. Tragi, P.C. Srivastava, M. Kumar, R.K. Singh, "Performance Comparison of RS code and Turbo Codes for Optical Communication," Serials Publications, ISSN: 0973-7383.vol.3, no.2, pp.251-256,2011

[12] I. DIOP, A. DIOP, I. DIOUM, S.M. FARSSI, K. TALL, S. OUYA, "Etude de la Performance des Codes de Reed Solomon et Turbo Code dans la Compression JPEG 2000 en Environnement Bruite, " J.sci.vol.10, N°. 2, pp.5-12, 2010

[13] R. w. Hamming, "Error detecting and Error correcting codes, "Bell system Tech.pp.147-160, 1950

[14] Y. Jiang," A Practical Guide to Error-Control Coding Using MATLAB", U.S. Library of congress, Artech house Boston/London artechhouse.com, 293p.ISBN 13: 978-1-60807-088-6, 2010

[15] O.M. Mohamed Bouye," Application des codes correcteurs d'erreurs en Sténographie," Thèse de Doctorat, Université Mohammed V, Rabat, 2009

[16] S. Dietler, " Implémentation de codes de Reed-Solomon sur FPGA pour communications spatiales," Thèse de Doctorat, Université de Larraine, 2006

[17] W. Stalling, "Cryptography and Network Security", Third edition, Prentice Hall, 2003

[18] J. A. Buchmann, "Introduction to Cryptography, "Second edition, October 2003

[19] C. S. Mondlin, "Error control coding in DSL system, CRC Press LCC, " 2004

[20] M. L. Boucenna, "Protocoles de Communications Satellitaires avec Sécurisation de la transmission et Récupération des Données, "Thèse de Magister, Université de Constantine Faculté des Sciences de l'Ingénieur, Algérie, 2008

[21] V. Thierry, "Recherche des performances dans la mise en œuvre des codes linéaires cycliques en ASIC à haut débit," Thèse de Doctorat, Université de Metz et Supelec, 1999

[22] C. Lee, "Error-Control Block Codes," Artech House Publishers, 2000

[23] R. Togneri, Ch. J.S. de Silva, "Fundamentals of information theory and coding design", Thèse publiée dans Taylor & Francis e-Library, 398p.ISBN 1-58488-310-3, 2006

[24] F. NABILA, H. SOUHAILA, "Etude et simulation d'une transmission de type OFDM pour les communications sans fil, " Thèse de Doctorat, Université Larbi Tebessi, Tebessa, Algérie, 2016

[25] B. SKLAR "Digital communications fundamentals and Applications ", Second Edition, Tarzana, California and University of California, Los Angeles, 2001.1032p.ISBN 0-13-084788-7

Résumé

Le développement vécu dans l'usage de l'informatique et du numérique dans nos sociétés pose le problème de la transmission d'informations numériques, où le critère principal d'une bonne transmission étant la conservation de l'intégrité de l'information initiale au bout du processus de transmission de celle-ci. Le rôle fondamental des codes correcteurs d'erreurs est de s'assurer de l'exactitude de l'information à laquelle nous accédons. L'augmentation constante de l'utilisation du numérique dans notre société rend ces derniers d'autant plus importants. Plusieurs exemples courants illustrent cette problématique : un premier est la transmission d'un message via internet, où le message peut s'altérer à cause du bruit sur les lignes téléphoniques. Cela entraîne une complexité croissante des systèmes de transmission et un débit de plus en plus élevé. À la réception, le système doit pouvoir détecter et corriger rapidement les éventuelles erreurs dues au bruit de canal (diminution du taux d'erreurs) et d'autres sources de perturbations. Les codes correcteurs d'erreurs doivent donc nous permettre d'obtenir les données originales transmises sans erreurs.

Mots clé : code correcteur d'erreur, codage, décodage, code Hamming, code Reed-Solomon, canal, taux d'erreurs.

Abstract

The development lived in the use of data processing and numerical in our companies poses the problem of the transmission of numerical information, where the principal criterion of a good transmission being conservation of the integrity of initial information at the end of the process of transmission of this one. The fundamental role of the error correcting codes is to ensure it self of the exactitude of the information which we reach. The constant increase in the use of numerical in our company makes the latter all the more significant. Several current examples illustrate these problems: a first is the transmission of a message via Internet, where the message can deteriorate because of the noise on the telephone lines. That involves an increasing complexity of the systems of transmission and an increasingly high flow. On the reception, the system must be able to detect and correct quickly the possible errors due to the noise of channel (reduction of the error rate) and other sources of disturbances. The error correcting codes must thus enable us to obtain the original data transmitted without errors.

Keywords: error correcting code, coding, decoding, Hamming code, Reed-Solomon code, channel, rate errors.

الملخص

إن التطور الذي شهده استخدام الحواسيب والتكنولوجيات الرقمية في مجتمعاتنا يثير مشكلة نقل المعلومات الرقمية، حيث يكون المعيار الرئيسي للإرسال الجيد هو الحفاظ على سلامة المعلومات الأولية في نهاية عملية الإرسال. والدور الأساسي لشفرات تصحيح الأخطاء يضمن دقة المعلومات التي نصل إليها. الزيادة المستمرة في استخدام التكنولوجيا الرقمية في مجتمعنا يجعل هذا الأخير أكثر أهمية. توضح عدة أمثلة مشتركة هذه المشكلة: الأولى هي إرسال رسالة عبر الإنترنت، حيث يمكن تغيير الرسالة بسبب الضوضاء على خطوط الهاتف. وهذا يؤدي إلى تعقيد متزايد لنظم الإرسال وزيادة الإنتاجية على نحو متزايد. عند الاستلام، يجب أن يكون النظام قادرا على كشف أي أخطاء أو تصحيحها بسرعة بسبب ضوضاء القناة (انخفاض معدل الخطأ) وغيرها من مصادر و .، يجب أن تمكننا رموز تصحيح الأخطاء من الحصول على البيانات الأصلية المرسلة دون أخطاء لذلك الاضطراب. و

الكلمات المفتاحية: خطأ تصحيح التعليمات البرمجية، ترميز، فك، مرموز هامينغ، مرموز ريد سولومان، قناة، معدل الخطأ.