N° d'ordre : .....

N° de série : .....

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Mémoire de Fin d'Étude

Présenté à

L'Université Echahid Hamma Lakhdar d'El Oued

Faculté de Technologie

Département de Génie Electrique

En vue de l'obtention du diplôme de

MASTER ACADEMIQUE

En Télécommunications

Présenté par

Heniat Nour Elhouda

Ghenadra Imane

Thème

Les codes de Reed Solomon: étude et simulation

Soutenu le 28/05/2016. Devant le jury composé de :

Melle. Boukaous Chahra.	Maitre de conférences	Président
Mr. Chemsa Ali.	Maitre de conférences	Rapporteur
Mr. Touhami Ridha.	Maitre Assistant A	Examinateur

Année Universitaire 2015/2016


بسم الله الرحمن الرحيم

REMERCIEMENTS

Avant tout nous remercions DIEU pour Tout
et pour la volonté pour reprendre nos études.
Nous tenons à remercier nos parents d'avoir sacrifies leur
vie pour notre bien.
Notre profonde gratitude s'adresse tout particulièrement à
notre rapporteur de mémoire, Monsieur ***CHEMSA ALI***
pour avoir accepté de suivre la réalisation de notre
mémoire, aussi pour ses conseils et son aide qu'il nous
apporté tout au long de ce travail.
Nos remerciements vont à tous les membres de jury qui ont
accepté de débattre notre sujet de mémoire.
En fin, nous adressons nos vifs remerciements aux
enseignants de notre spécialité.
Ainsi que tous ceux qui ont aidé de prés ou loin pour
réalisation de nos études et de la réalisation de ce travail

Nour & Imane

Dédicace

Je dédie ce travail ;

A mes très chers parents et bien aimée mère symbole d'amour et patience pour ses sacrifices inestimables.

A ma sœur Ichrak et mes frères.

A mes chers grands parents.

A mes tantes et mes oncles, cousines et cousins

A mon chéri "imane"

A tous mes enseignants pour le savoir et les connaissances qu'ils m'ont inculqué.

A tous mes amis.

Sara, amina, souhir, zineb, hadjer, belkiss, ines, imane…..

A tous mes collègues de la promotion TELECOMM 2015/2016

Nour

Dédicace

Je dédie ce travail ;

A ma très chère mère pour son amour inconditionnel et sa présence à mes cotés dans les moments difficiles.

A la mémoire de mon très cher père symbole de courage, de tendresse et que Dieu le puissant l'accorde sa clémence et l'accueille en son vaste paradis.

A mes soeurs et frères et leurs enfants pour leur encouragement.

A mes chers grands parents.

A mes tantes et mes oncles, cousines et cousins

A mon chéri "Nour"

A tous mes enseignants pour le savoir et les connaissances qu'ils m'ont inculqué.

A tous mes amis.

A tous mes collègues de la promotion

TELECOMM 2015/2016

Imane

Sommaire

Résumé

Ce travail est dédié à l'étude et à la simulation des codes de **Reed Solomon**. Les codes de **Reed Solomon** notés $RS(n,k)$sont des codes de détection et de correction des erreurs. Se sont des codes particuliers des codes BCH.Les messages sont divisés en blocs de k symboles chacun dont on a ajouté $n-k$ symboles de redondance ou de contrôle à chaque bloc permettant ainsi de diminuer la possibilité de retransmission. La longueur des blocs dépend de la capacité du codeur.Le décodeur traite chaque bloc de n symboles et corrige les éventuelles erreurs. A la fin de ce traitement, les données originelles seront restaurées.

Mots clefs : Code Correcteur, Code de **Reed Solomon**, Corps de **Galois**, AWGN, BER, BPSK, SNR.

Abstract

This work is dedicated to the study and simulation of the **Reed Solomon** codes. The **Reed Solomon** codes as noted $RS(n,k)$ are detection codes and error correction. They are a particular codes of the BCH codes. The message to be transmitted are divided into blocks of k symbols each of which was added $n-k$ redundancy symbols or control each block, which allows reduce the possibility of the transmission again. The length of the blocks depends on the encoder's capacity. The decoder processes each block of n symbols and corrects any errors. At the end of this treatment, the original data will be restored.

Key words : Correction Code, **Reed Solomon** code, **Galois** Field, AWGN, BER, BPSK, SNR.

ملخص

يهدف هذا العمل إلى دراسة ومحاكاة مشفر ريد سولومون. مشفر ريد سولومون الذي يرمز له بالرمز $RS(n,k)$هو عبارة عن كاشف ومصحح للأخطاء في آن واحد، وهو حالة خاصة من مشفراتBCH. الرسائل التي يراد ارسالها عن طريق هذا المشفر تقسم إلى كتل مشكلة من k حرف حيث يضيف المشفر لكل كتلة $n-k$ حرف مراقبة للحصول على كتلة ذات n حرف تسمى بكلمة المشفر وهذا كله من أجل حماية الرسالة من الاخطاء ومنه تفادي بعثها مرات أخرى.ننبه بأن طول الكتل يعتمد على قدرة المشفر. على مستوى المستقبل يقوم المفكك بمعالجة كل كتلة ذات n حرف ويصحح الاخطاء الواردة. في نهاية هذه المعالجة يتم استعادة الرسائل الأصلية.

المصطلحات الأساسية : المشفر المصحح، مشفر **ريد سولومون**، حقل **جالوى**، AWGN، BER، BPSK، SNR.

Liste des figures

Glossaire

BCH	**B**ose, Ray-**C**haudhuri et **H**ocquenghem
RS	**R**eed-**S**olomon
MDS	**M**aximum **D**istance **S**eparable
LFSR	**L**inear **F**eedback **S**hif **R**egister
BPSK	**B**inary **P**hase **S**hift **K**eying
AWGN	**A**dditive **W**hite **G**aussian **N**oise
BER	**B**inary **E**rror **R**ate
MAP	**M**aximum **A** **P**osteriori
SNR	**S**ignal-to-**N**oise **R**atio
PSK	**P**hase **S**hift **K**eying
M-PSK	**M**odulator -**P**hase **S**hift **K**eying
M-QAM	**M**-ary **Q**uadrature **A**mplitude **M**odulation

Introduction générale

De nos jours, on vive dans un monde où les communications jouent un rôle primordial tant par la place qu'elles occupent dans le quotidien de chacun, que par les enjeux économiques et technologiques dont elles font l'objet. On a sans cesse besoin d'augmenter les débits de transmission tout en gardant ou en améliorant la qualité de ceux-ci. Mais sans un souci de fiabilité, tous les efforts d'amélioration seraient vains car cela impliquerait forcément à ce que certaines données soient retransmises. C'est dans la course au débit et à la fiabilité que les codes correcteurs entrent en jeu…[1].

Un code correcteur d'erreur permet de corriger une ou plusieurs erreurs dans un mot-code en ajoutant aux informations des symboles redondants, autrement dit, des symboles de contrôle. Différents codes possibles existent mais dans ce travail on traitera seulement les codes de Reed Solomon car pour le moment, ils représentent le meilleur compromis entre efficacité (symboles de parité ajoutés aux informations) et complexité (difficulté de codage)[2].

Les codes de **Reed Solomon** sont de codes correcteurs basés sur les corps ou les champs de **Galois** GFdont le principe est de construire un polynôme formel à partir des symboles à transmettre et de le sur échantillonner. Le résultat est alors envoyé, au lieu des symboles originaux. La redondance de ce sur échantillonnage permet au récepteur du message encodé de reconstruire le polynôme même s'il y a eu des erreurs pendant la transmission [3].

Les codes **Reed Solomon** sont des codes par bloc. En effet ils prennent en entrée un bloc de données de taille fixéek, qui est transformé en un bloc de sortie de taille fixée $n > k$. Ces codes travaillent sur un corps fini noté $GF(2^m)$, qui possède le plus souvent 2^méléments. Grâce à un ajout de redondance, ces codes permettent de corriger deux types d'erreurs

- les erreurs induisant une modification des données, ou certains bits passent de la valeur 0 à la valeur 1 et vice versa comme sur le canal binaire symétrique ;

- les erreurs provoquant des pertes d'informations aussi appelées effacements, lorsque des paquets d'informations sont perdus ou effacés comme sur le canal binaire à effacement).

On note un code de **Reed Solomon** $RS(n,k)$ou $RS(n,k,t)$ avec $n = 2^m - 1$et $2t = n - k$[4].

L'objectif de notre mémoire est de faire une étude détaillée, claire et complète des différents concepts principaux des codes **Reed Solomon** et d'enrichir notre étude par une simulation numérique.

Ce mémoire est structuré en trois chapitres.

Dans le premier chapitre, on va exposer 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, on va regrouper toutes les propriétés essentielles qui peuvent être, plus tard, exploitées dans la suite de ce travail.

Le second chapitre est consacré principalement au fond du travail qui est les codes de **Reed Solomon**. Dans ce chapitre on présentera à une généralité sur les mathématiques des corps finis corps finis dits les « champs de **Galois** » qui sont très utilisés dans la théorie de codage et de décodage des codes de **Reed Solomon**. Les opérations de base dans ces corps, comme l'addition et la multiplication, sont bien présentées dans ce chapitre avec des exemples didactiques.

La théorie des codes de **Reed Solomon** présentera deux méthodes de décodage. La première solution présentée sera la méthode de la division Euclidienne, quand à la deuxième méthode, elle mettra en évidence l'algorithme de **Reed Solomon**[5].

Dans le troisième chapitre on fera la réalisation et la simulation des codes **Reed Solomon**, en présentant l'évaluation des performances de ces codes. Nous verrons que la simulation numérique montre que ces codes sont très puissants et améliore d'une façon certaine et considérable les performances d'un système de communication numérique en en termes de taux d'erreurs binaire BER.

Enfin, une conclusion générale dresse un bilan de ce travail et propose quelques perspectives de recherche.

Chapitre 1

Généralités sur les codes correcteurs

Chapitre 1

Généralités sur les codes correcteurs

1.1 Introduction

Dans le canal de transmission l'information est transmise de l'émetteur au récepteur. A cause du bruit qui est introduit dans la canal, le récepteur reçoit un message qui diffère du message originale. Pour éviter l'effet de bruit on a utilisé la technique de détection et correction des erreurs [6]. Un code correcteur est une technique de codage basée sur la redondance ou l'ajout de symboles de contrôle au message original. Cette technique montre son efficacité dans la pratique, et pour cette raison on la trouve dans la transmission par satellite, téléphonie, disque laser, TV haute définition [7].

Dans ce chapitre, on va présenter une généralité sur les codes correcteurs d'erreurs où on va donner les notions de base et les fondements de ces codes.

1.2 Les codes correcteurs d'erreurs

1.2.1 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 original 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.

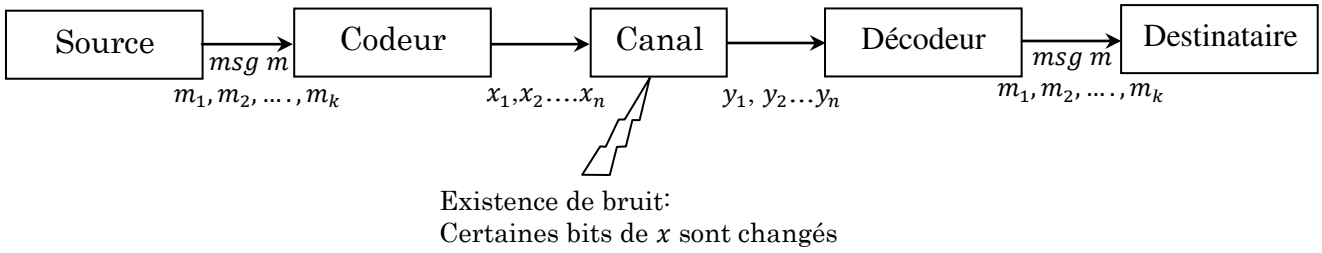


Fig. 1.1 Principe de la chaîne de transmission de l'information.

Un message $m = (m_1, m_2, \ldots m_k)$ 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 y en 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 [8].

1.2.2 Définition mathématique des codes

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 $\sum^n$ l'ensemble de toutes les séquences de taille n, on a donc

$$|\sum^n| = q^n \qquad (1.1)$$

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

- ✓ **Définition 1 :** Un code C défini sur un alphabet $\sum$ est un sous-ensemble de $\sum^n$. Le code ainsi obtenu est appelé un code q −aire de longueur n.
- ✓ **Définition 2 :** Une fonction d'encodage φ est une fonction injective qui, à chaque élément de l'espace du message, associe un mot-code. On a donc

$$\varphi: Espace\ des\ messages \rightarrow \sum^{n}$$

Un code est donc l'image obtenue par cette fonction

$$C = Im\{\varphi\} \tag{1.2}$$

Notation : On note (n, k, d), un code qui transforme des blocs de message de k éléments en mots de code de n éléments, avec une distance minimale égale à d. Le nombre des mots-code est alors $M = q^k$ [3].

1.2.3 Caractéristiques d'un code

Les codes correcteurs sont caractérisés par cinq données importantes.

1.2.3.1 Dimension et longueur d'un code

Un code est une application injective $\varphi: \{0,1\}^k \rightarrow \{0,1\}^n$, Le paramètre k est appelé la dimension du code φ et le paramètre n est appelé la longueur du code: on dit que φ est un code de paramètres (n, k).

L'ensemble $C = \{\varphi(m), m \in \{0,1\}\}$ est appelé l'image du code φ. Les éléments de C sont appelés les mots de code de φ (en opposition aux éléments « originels » de $\{0,1\}^k$qui sont appelés mots de source). Deux codes ayant la même image sont dits équivalents.

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

Correspondant au nombre 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.

1.2.3.3 La probabilité d'erreurs du code

Cette probabilité va dépendre de la distance minimale entre les mots de code. 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.

1.2.3.4 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.

1.2.3.5 Distance minimale d'un code

Soit φ un code d'image C.

On appelle capacité de détection de φ le plus grand entier e_d tel qu'on soit toujours capable de détecter e_d erreurs ou moins.

On appelle capacité de correction de φ le plus grand entier e_c tel qu'on soit toujours capable de corriger e_c erreurs ou moins.

On appelle distance minimale de φ et on note d_φ $(ou\ d_c)$ la plus petite distance non nulle entre deux mots de code. On a

$$e_d = d_\varphi - 1 \ et \ e_c = \left\lfloor \frac{d_\varphi - 1}{2} \right\rfloor \qquad (1.3)$$

La distance minimale d'un code quantifie donc sa qualité vis à vis que l'information ne doit pas être trop diluée.

C'est un paramètre important. En abrégé, « un code de dimension k, de longueur n et de distance minimale d » se dira « un code de paramètres (n, k, d) » ou même « un code (n, k, d) » [9].

1.2.4 Classification des codes Correcteurs

C'est le problème du codage de canal. A côté des premiers codes empiriques (bit de parité, répétition,…) deux grandes catégories de codes ont été développées et sont actuellement utilisées en faisant l'objet permanent de perfectionnements

- ✓ **Les codes en blocs:** (linéaires, cycliques ou non) : le codage/décodage d'un bloc dépend uniquement des informations de ce bloc.
- ✓ **Les codes en treillis :** (convolutifs, récursifs ou non) : le codage/décodage d'un bloc dépend des informations d'autres blocs (généralement de blocs précédemment transmis).

La figure ci-dessous donne un simple résumé de la grande famille de codage. Dans la première classe, citons les codes les plus célèbres comme les codes BCH, Reed Muller, **Reed Solomon** et **Goppa**, **Golay** et **Hamming**. La deuxième classe est moins riche en variété mais présente beaucoup plus de souplesse surtout dans le choix des paramètres et des algorithmes de décodage disponibles. Citons par exemple, les codes convolutifs binaires systématiques récursifs très utilisés dans les modulations codées (TCM) et les codes concaténés parallèles (Turbo Codes) [10].

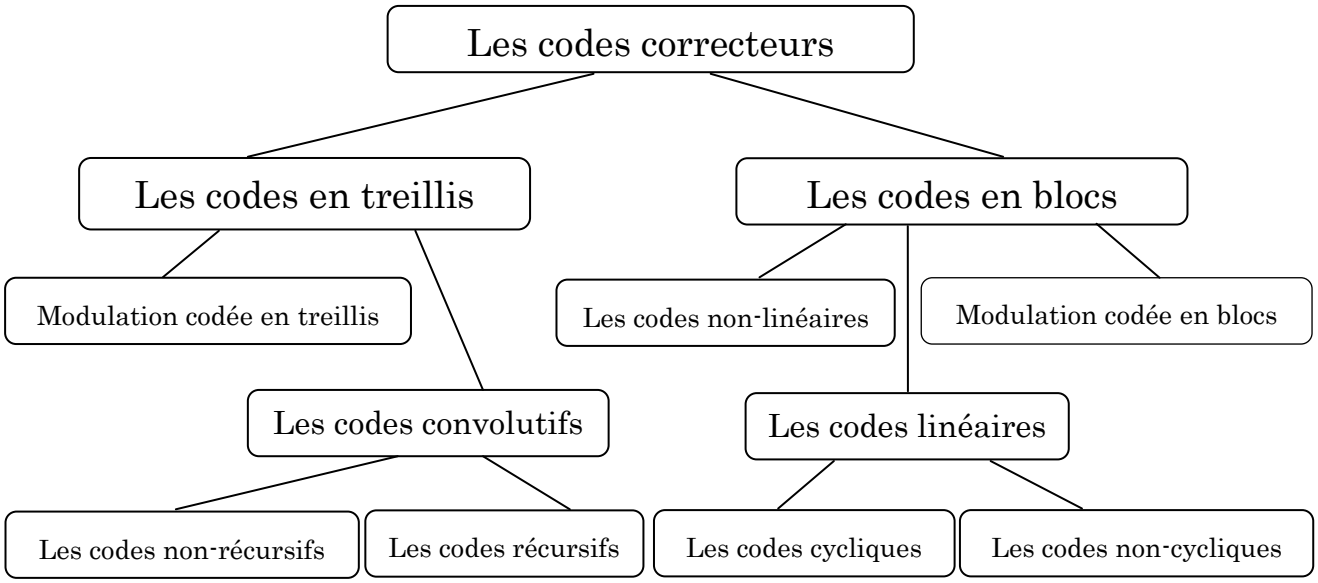


Fig. 1.2 La hiérarchie des codes correcteurs [10].

1.2.5 Exemples de codes simples

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.

1.2.5.1 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

$$C = \{(00\cdots0);(11\cdots1)\} \tag{1.4}$$

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 $\{(00\ldots0);(11\ldots1)\}$ 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 $(n-1)/2$ 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 λ fois.

1.2.5.2 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 de1. 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

$$C = \left\{ \frac{(c_1, \cdots, c_n)}{\sum c_i} = 0 \right\} \tag{1.5}$$

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 [11].

1.3 Les codes linéaires

1.3.1 Matrice Génératrice

Un code φ de paramètres (n,k) est dit linéaire s'il existe une matrice $G \in M_{n,k}(F_2)$ (c'est-à-dire avec n lignes, k colonnes, à coefficients dans $\{0,1\}$), de rang k, telle que

$$\forall m \in \{0,1\}^K, \varphi(m) = G \times m \tag{1.6}$$

La multiplication matricielle est à comprendre dans F_2 (c'est à dire modulo- 2 et le mot m est ici considéré comme un vecteur colonne. La condition sur le rang traduit l'injectivité de φ. La matrice G est appelée matrice génératrice de φ.

Enfin, dire que le code φ est systématique, c'est dire que la matrice G est le la forme $\binom{I_k}{G'}$, où I_k est la matrice identité d'ordre k.

Exemple: a) Considérons φ, code linéaire de matrice génératrice $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$

C'est un code de dimension 2, de longueur 4, donné par le tableau

x	00	01	10	11
$\varphi(x)$	0000	0101	1011	1110

b) Pour le code de bit de parité $(8,9)$, la matrice génératrice est $\begin{pmatrix} I_8 \\ 1 \cdots 1 \end{pmatrix}$ [12].

1.3.2 Matrice de contrôle d'un code linéaire

L'intérêt principal de ne considérer que des codes linéaires est qu'ils disposent de meilleurs algorithmes de décodage. On utilise pour cela les matrices de contrôle.

✓ **Définition**

Soit φ un code linéaire (n,k) de matrice génératrice G. On appelle matrice de contrôle de φ toute matrice $H \in M_{n-k,n}$ (c'est-à-dire avec n colonnes et $n-k$ lignes) telle que

$$H.m = \vec{0} \Leftrightarrow m \in C \tag{1.7}$$

L'existence de matrices de contrôle pour n'importe quel code linéaire φ n'étonnera pas les spécialistes de l'algèbre linéaire: H n'est rien d'autre qu'une matrice dont le noyau est l'image de G. Il est clair que de telles matrices existent toujours.

Au passage, comme le noyau de H est C, H contient assez d'information pour reconstituer φ à équivalence prés.

En revanche, on remarque que H n'est pas unique en général (par exemple, en permutant deux lignes d'une matrice de contrôle, on obtient encore une matrice de contrôle).

Etant donné la matrice génératrice G d'un code φ, se donner une matrice de contrôle H, c'est se donner une base de l'orthogonal de C dans $\{0,1\}^n$, ce qui n'est pas évident. L'opération inverse est plus facile : étant donné une matrice de contrôle H, Se donner la matrice génératrice G d'un code φ correspondant revient à se donner une base du noyau de H.

Le théorème suivant donne un critère très simple pour déterminer une matrice de contrôle d'un code systématique[13].

✓ **Théorème**

Soit φ un code systématique de matrice génératrice $\binom{I_k}{G'}$.

Alors la matrice $H = (G'\ I_{n-k})$ est une matrice de contrôle de G.

✓ **Exemple**

Une matrice de contrôle du code de bit de parité $(8,9)$ est $(1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$[13].

1.3.3 Distance minimale d'un code linéaire

Les codes linéaires sont également intéressants car on dispose d'informations sur leur distance minimale. Tout d'abord, par définition, dire qu'un code φ de paramètres (n,k) est linéaire, c'est exactement dire que φ est une application linéaire injective de $\{0,1\}^k$ dans $\{0,1\}^n$.

✓ **Proposition** (distance minimale d'un code linéaire)

Soit φ un code linéaire d'image C. Alors la distance minimale de d_φ est égale au plus petit poids non nul d'un mot de C [14].

✓ **Calcul de la distance minimale d'un code**

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

$$d_{min} = min\{d(x,y)\ /x\ , y \in C, x \neq\ y\} \quad (1.8)$$

$$d_{min} = min\{w(x+y)\ /\ x\ , y \in C, x \neq y\} \quad (1.9)$$

$$d_{min} = min\{(x)\ /\ x \in C, x\ \neq 0\} =\ w_{min} \quad (1.10)$$

✓ **Distance minimale et matrice de contrôle**

On peut encore avoir un autre critère sur la distance minimale d'un code linéaire il est donné par la matrice de contrôle du code.

✓ **Proposition**

Soit φ un code linéaire de matrice de contrôle H.

Alors d_φ est le nombre minimal de colonnes de H linéairement dépendantes.

- Si H a une colonne nulle, on a $d_\varphi = 1$.
- Sinon, et si H a deux colonnes identiques, on a $d_\varphi = 2$.
- Sinon, et si une colonne de H est égale à la somme de deux autres, on a $d_\varphi = 3$, etc.

1.3.4 Décodage d'un code linéaire

✓ **Définition** (Mot erreur et syndrome)

Soit φ un code de paramètres (n,k), de matrice génératrice G et de matrice de contrôle H. On se fixe un mot de source x (mot de longueur k). Le mot de code correspondant sera $\varphi\ (x) = Y$. Il y a éventuellement des erreurs durant la transmission et on reçoit le mot Z.

On appelle mot erreur associé à Z le mot $E = Z + Y$ .On appelle syndrome de Z le mot $H \times Z$. On note E_i le mot ne contenant que des 0, sauf en position i.

Quelques commentaires sur cette définition

Comme $E = Y + Z$, on a $Z = Y + E$, et donc E quantifie bien les erreurs survenues sur Y : par exemple, $w(E)$ est le ? d'erreurs survenues. Le problème du décodage peut se reformuler : « trouver E », car alors on déduit Y .

Bien sûr, ce « trouver E » est à comprendre dans le sens « trouver E le plus probable », c'est à dire « trouver E de poids minimal ».

Dire que le syndrome de Z est nul, c'est dire que Z est un mot du code, dans ce cas le mot erreur le plus probable est le mot nul[15].

✓ **Remarque**

Soit Z un mot de syndrome S.

Alors E est aussi de syndrome S car $H \times E = H \times (Z + Y) = H \times Z + H \times Y$ et Y est un mot du code.

L'ensemble des mots de syndrome S est appellé classe latérale de Z.

Le problème de décodage se reformule donc : « trouver le mot de plus petit poids dans la classe latérale de Z » ... s'il existe !

Par exemple

- Dire que le mot nul est dans la classe latérale de Z, c'est dire que le syndrome $S = H \times Z$ est nul. Dans ce cas, le mot de poids minimal dans la classe lattérale de Z est le mot nul, il faudra donc corriger Z par $\varphi^{-1}(Z)$.
- Supposons donc que le syndrome de $S = H \times Z$ est non nul. Dire qu'il existe un mot de poids 1 dans la classe latérale de Z, c'est dire que le syndrome S est égal à une colonne C_i de H. Dans ce cas, s'il existe une seule colonne C_i de$H$ égale au syndrome, le mot de poids minimal dans la classe lattérale de Z est E_i, il faudra donc corriger Z par $\varphi^{-1}(Z + E_i)$, ou E_i est le mot ne contenant que des zéros, sauf en position i.
- Supposons donc que le syndrome de $S = H \times Z$ est non nul, et distinct de toutes les colonnes de H. Dire qu'il existe un mot de poids 2 dans la classe latérale de Z, c'est dire que le syndrome de Z est égal à la somme de deux colonnes de H. Dans ce cas, et si c'est la seule façon d'obtenir le syndrome comme somme de deux colonnes de H, le mot de poids minimal dans la classe latérale de Z est $E_{i1} + E_{i2}$, il faudra donc corriger Z par $\varphi^{-1}(E_{i1} + E_{i2})$. ... un algorithme prend forme !

Dans ce qui précède, le point délicat est l'unicité « du » mot de poids minimal dans la classe latérale de Z. Par exemple, si deux colonnes C_i et C_j de H sont égales à $S = H \times Z$, on ne sait pas corriger. Pour commencer, faisons l'hypothèse que ce cas de figure ne se produira jamais. Il suffit pour cela de renoncer à corriger au-delà de la capacité de correction, car un mot de poids $p \leq e_c$ peut s'écrire au plus d'une seule façon comme somme de colonnes de H [16].

Algorithme de de décodage des codes linéaires, 1^{ere} version

C'est un algorithme de décodage en deçà de la capacité de correction, donc non optimal (sauf si le code est parfait).

Etape 1 : on calcule $S = H \times Z$.

Etape 2 : si $S = \vec{0}$, on décode Z par $\varphi^{-1}(Z)$, sinon, on initialise p à 1.

Etape 3 : s'il existe p colonnes de H : $C_{i1}, C_{i2}, \cdots C_{ip}$ telles que

$S = C_{i1} + C_{i2} + \cdots + C_{ip}$, on décode par

$$\varphi^{-1}(Z + E_{i1} + E_{i2} + \cdots + E_{ip}) \qquad (1.11)$$

- Sinon, et qu'on a $p < e_c$, on incrémente p et on recommence l'étape 3.
- Sinon, et donc qu'on a $p = e_c$, on passe à l'étape 4.

Etape 4 : échec du décodage

Finalement, la complexité de cet algorithme est en $O(e_c \times n^{e_c+1} \times (n-k))$, ce qui est loin d'être terrible, même si c'est déjà beaucoup mieux que la complexité exponentielle de l'algorithme naïf (si k est grand).

Comment souvent, on peut faire mieux en convertissant une partie de la complexité temporelle en complexité spatiale. En effet, on peut une bonne fois pour toutes calculer, pour tout $p \leq e_c$, toutes les sommes possibles de p colonnes de H, les trier par ordre lexicographique et leur associer le mot erreur E correspondant.

Le calcul de toutes les sommes dans l'étape 3 est alors remplacé par une recherche dichotomique, de coût $O(p)$.

Algorithme de décodage des codes linéaires, $2^{éme}$ version

Le même, mais avec un prétraitement consistant à calculer pour tout $p \leq e_c$ toutes les sommes possibles de p colonnes de H, à les trier par ordre lexicographique et leur associer le mot erreur E correspondant (par exemple à la somme de C_1 et C_3 est associé $E = 10100\ldots$).

Le coût du prétraitement est en $O(e_c \times n^{e_c+1} \times (n-k))$. L'algorithme a de plus une complexité en espace en $O(e_c \times n^{e_c+1} \times (n-k))$

La complexité temporelle de l'algorithme lui-même est en $O(n \times (n-k))$

Le défaut non réglé est le caractère non optimal de ces algorithmes. Une troisième version, dont la complexité en espace est en $O(2^{n-k})$, règle ce problème[14].

Algorithme de décodage des codes linéaires, $3^{émé}$ version

Pré calcul : pour tous les syndromes possibles (les 2^{n-k} mots de longueur $n-k$), on calcule le mot de poids minimal de la classe latérale associée, ce qui nous donne un tableau de taille (2^{n-k}) (le i^e élément est le mot de poids minimal de la classe latérale dont le syndrome est i écrit en base 2, s'il existe ; et un symbole d'échec sinon).

Etape 1 : on calcul $S = H \times Z$ et i dont S est l'écriture en base 2

Etape 2 : on sélectionne le i^e élément E du tableau

Etape 4 : on corrige par $\varphi^{-1}(\mathrm{Z} + \mathrm{E})$

Le coût de cet algorithme est en $O(n \times (n-k))$, mais la complexité en espace est très importante, et surtout le pré calcul n'est pas réalisable si n est vraiment grand, car il est en $O(n \cdot 2^n)$.

Cet algorithme peut être optimisé, en proposant par exemple que s'il y a plusieurs mots de poids minimal dans une classe latérale, on choisisse celui, s'il existe, pour lequel les erreurs sont le mieux groupées [14].

1.3.5 Exemple de codage d'un mot m

Soit le message à transmettre : $m = (1\ 1\ 0\ 1)$.

On envoie le mot de code c défini par $C = m$

$$C = (1\ 1\ 0\ 1) . \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} = (0\ 0\ 0\ 1\ 1\ 0\ 1)$$

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

1.3.6 Exemple du décodage d'un mot reçu z

Imaginons qu'à la sortie du canal, on reçoive le mot $Z = (1\ 0\ 0\ 1\ 1\ 0\ 1)$ 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

$$S = Z.H^T \tag{1.12}$$

$$S = (1\,0\,0\,1\,1\,0\,1).\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}^T$$

$$S = (1\,0\,0)$$

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 Z.

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

$$Z' = (0\,0\,0\,1\,1\,0\,1) \tag{1.13}$$

Le message envoyé étant contenu dans les quatre derniers bits, on retrouve simplement le mot envoyé qui était bien $m = (1\,1\,0\,1)$ [11].

1.4 Codes parfaits

1.4.1. Bornes caractérisant les codes

Le Singleton Bound et la Sphère **Packing Bound**, appelée également borne de **Hamming**.

- ✓ **Définition 1:** (borne de Singleton)

La distance minimale d d'un code linéaire de dimension k et de longueur n vérifie $d \leq n + 1 - k$.

Un code pour lequel on a égalité est dit MDS (Maximum Distance Separable).

- ✓ **Définition 2:** (Sphères, boules)

Soit $m \in \{0,1\}^n$ et $k \in N$. On appelle sphère de centre m et de rayon k, et on note $S(m\,,k)$, l'ensemble des mots de $\{0,1\}^n$ à distance k de m

$$S\,(m,k) = \{r \in \{0,1\}^n, d(r,m) = k\} \tag{1.14}$$

On appelle boule de centre m et de rayon k, et on note $B(m,k)$, l'ensemble des mots de $\{0,1\}^n$ à distance inférieure ou égale à k de m

$$B(m,k) = \{r \in \{0,1\}^n, d(r,m) \leq k\} \tag{1.15}$$

✓ **Proposition** (Inégalité de **Hamming**)

Soit φ un code de paramétres (k, n), d'image C, de capacité de correction e_c. Alors on a

$$\sum_{i=0}^{e_c} C_n^i \leq 2^{n-k} \quad (1.16)$$

Et on appelle cette propriété l'inégalité de Hamming.

Cette borne nous permet de définir la notion de code parfait [17].

1..4.2 Définition Codes parfaits

Soit φ un code de paramètres (n, k), de capacité de correction e_c. On dit que φ est un code parfait s'il vérifie une des trois caractérisations équivalentes suivantes

- Les boules de rayon e_c de centre les mots du code forment une partition de$\{0,1\}^n$.
- φ vérifie le cas d'égalité de l'inégalité de **Hamming** $\sum_{i=0}^{e_c} C_n^i \leq 2^{n-k}$.
- Pour tout mot $r \in \{0,1\}^n$, il existe un unique mot de code m qui réalise le minimum de $d(r, m)$ [14].

1.4.3 Caractérisation des codes parfaits linéaires

Comme promis, montrons que les codes parfaits sont rares. Commençons par un exemple.

✓ **Définition :** (Codes de **Hamming**)

Soit $m \leq 2$ un entier.

Un code de **Hamming** est un code de paramètres $(2^m - m - 1,\ 2^m - 1)$.

✓ **Proposition** (Distance minimale d'une code de Hamming)

Un code de **Hamming** a toujours pour distance minimale 3.Un code linéaire de Paramétres $(2^m - 1,\ 2^m - m - 1,3)$ est nécessairement un code de **Hamming** , 2^m lignes et $m - 1$ colonnes.

Posons-nous maintenant la question de trouver tous les codes parfaits de capacité de correction 1. Comme les codes parfaits vérifient l'égalité de **Hamming**, les paramètres d'un code parfait de capacité de correction 1 doivent vérifier $\sum_{i=0}^{1} C_n^i \leq 2^{n-k}$ c'est-à-dire $1 + n = 2^{n-k}$. Posons $m = n - k$, on a alors $1 + n = 2^m$, donc $n = 2^m - 1$.

$$k = n - m\ 2^m - m - 1 \quad (1.17)$$

En définitive un code parfait de capacité de correction 1 a nécessairement les paramétres d'un code de **Hamming** ; et un code parfait linéaire de capacité de correction 1 est toujours un code de **Hamming** [14].

✓ **Théorème** (Codes parfaits linéaires)

Les seuls codes parfaits linéaires (binaires) sont

- Les codes de répétition pure $(1, 2e_c + 1)$.
- Les codes de **Hamming**.
- Le code de **Golay** $G_{23}(12, 23, 7)$.

1.5 Les codes cycliques

1.5.1 Rappels sur les polynômes

✓ **Définition** (Polynômes à coefficients dans F_2)

Un polynôme à coefficients dans F_2 est une fonction de la forme

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \quad avec \quad \forall_i \in \{0, \cdots n\}, a_i \in F_2 \qquad (1.18)$$

Si $a_n \neq 0$, l'entier n est appelé le degré du polynôme P, les entiers a_i sont appelés les coefficients de P, par convention le polynôme nul est considéré comme étant de degré $-\infty$.

✓ **Remarque**

Le fait de travailler dans F_2 nous simplifie grandement la vie. Par exemple, on a toujours $(a + b)^2 = a^2 + b^2$

✓ **Proposition**

Soit P un polynôme à coefficients dans F_2. Alors

$$P(x^2)\ P(x)^2 \qquad (1.19)$$

✓ **Définition** (racines)

Soit P un polynôme à coefficients dans F_2 et $a \in F_2$, On dit que a est une racine de P lorsque $P(a) = 0$.

Exemple :

- Le polynôme $x^2 + x$ a pour racines 0 et 1
- Le polynôme $x^2 + 1$ a pour racine 1 .

- Le polynôme $x^2 + x + 1$ n'a pas de racine dans F_2. Si l'on veut à tout prix qu'il ait des racines, il faudra « imaginer » de nouveaux éléments qui ne sont pas dans F_2, de la même façon qu'on construit le corps des nombres complexes en « imaginant » un nouveau nombre i tel que $i^2 = -1$.

✓ **Définition** (factorisation, irréductibilité)

Soit P un polynôme à coefficients dans F_2.

- S'il existe deux polynômes P_1 et P_2 tels que $P = P_1 P_2$, on dira que P_1 et P_2 divisent P, ou encore que P_1et P_2 sont des diviseurs de P. Dans ce cas on a nécessairement

$$deg(P_1) + deg(P_2) = deg(P) \tag{1.20}$$

- S'il existe deux polynômes P_1 et P_2 tels que

$$deg\,(P_1) \geq 1, deg\,(P_2) \geq 1 \;\; et \;\; P = P_1 P_2 \tag{1.21}$$

alors on dit que P_1 P_2 est une factorisation de P.

- Si P n'a pas de factorisation, P est dit irréductible

✓ **Proposition** (division euclidienne)

Soient P_1 et P_2 deux polynômes à coefficients dans F_2. Alors il existe deux polynômes à coefficients dans $F_2\, Q\, et\, R$, uniques, tels que

$$P_1 = P_2 \times Q + R et\; deg(R) < deg(P_2) \tag{1.22}$$

Q est appelé le qutient de la division euclidienne de P_1 par P_2 et R le reste[14].

1.5.2 Définitions d'un code cyclique

Soit C l'image d'un code de paramètres (n, k). Le code est dit cyclique si l'ensemble des mots du code est stable par décalage circulaire. En d'autres termes, notons

$$\sigma(\; m_1\, m_2 \cdots \; m_{n-2}\, m_{n-1}\, m_n\,) = m_n\, m_1\, m_2 \cdots m_{n-2}\, m_{n-1} \tag{1.23}$$

Un code (d'image) C est cyclique si $\forall m \in C, \sigma\,(m) \in C$.

Exemple

- Les codes de répétition pure (n, k) sont cycliques.
- Le code par bit de parité est cyclique.
- Le code de **Hamming** systématique $(7, 4)$ est cyclique.

- Plus généralement il existe des codes de **Hamming** $(2^m - m - 1, 2^m - 1)$ cycliques pour tout $m \leq 2$.
- Le code de **Hamming** étendu $(4, 8)$ n'est pas cyclique !

✓ **Définition** (code cyclique engendré par un mot)

Soit $m \in \{0,1\}^n$. Notons C le plus petit sous-espace vectoriel de $\{0,1\}^n$ stable par décalage circulaire. Alors C est un code (ou l'image d'un code) cyclique qu'on appelle code cyclique engendré par m. Le code cyclique engendré par m est l'ensemble des mots qu'on peut obtenir en faisant des sommes finies de décalés circulaires itérés de m, auquel il faut ajouter le mot nul[14].

Exemple

- Le code cyclique engendré par 111 est (équivalent à) notre code de répétition pure $(1, 3)$.
- Le code cyclique engendré par 110000000 est (équivalent à) notre code de bit de parité $(8, 9)$.

✓ **Théorème** (Théorème fondamental : générateur d'un code cyclique)

Soit C (l'image d') un code cyclique (n, k).

Alors il existe un unique polynôme

$$g(x) = a_0 + a_1\, x + \cdots + a_{n-k}\, x^{n-k} (a_{n-k} = 1) \qquad (1.24)$$

tel que

- $g(x)$ est un diviseur de $x^n + 1$
- C est le code cyclique engendré par $m = a_0 a_1 \cdots a_{n-k} 0 \cdots 0$ $(k - 1$ zéros$)$
- Les mots

$$m = a_0 \cdots a_{n-k} 0 \cdots 00;\ \sigma(m) = 0\, a_0 \cdots a_{n-k} 0 \cdots 0; \cdots; \sigma^{k-1}(m) = 0 \cdots 0 a_0 \cdots a_{n-k} \qquad (1.25)$$

forment une base de C. Ce qui signifie qu'une matrice génératrice de C est donnée par

$$\begin{pmatrix} a_0 & 0 & & 0 \\ a_1 & a_0 & & \vdots \\ \vdots & \vdots & & 0 \\ a_{n-k} & a_{n-k-1} & \cdots & 0 \\ 0 & a_{n-k} & & a_0 \\ 0 & 0 & & a_1 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & a_{n-k} \end{pmatrix}$$

✓ **Définition** (représentation polynomiale)

Soit $m = a_0 a_1 \dots a_n$, un mot de longueur n. On appelle représentation polynomiale de m le polynôme $a_0 + a_1 x + \cdots + a_n x^n$. Dorénavant, on identifiera systématiquement un mot avec sa représentation polynomiale.

Soit C un code cyclique (n, k). On appelle polynôme générateur de C le polynôme $g(x)$ défini par le théorème fondamental. Dorénavant, on identifiera un code cyclique avec son polynôme générateur.

Exemple

- Le polynôme générateur du code de répétition pure $(1, n)$ est $1 + x + x^2 + \cdots + x^{n-1}$
- Le polynôme générateur du code par bit de parité $(n-1, n)$ est $1 + x$
- Le polynôme $1 + x + x^3$ *est* le polynôme générateur d'un c de **Hamming** $(4, 7)$ [14].

1.5.3 Codes cycliques vs codes systématiques

Donnons-nous un code cyclique par son polynôme générateur $g(x)$. Dans cette section, on montre comment coder et décoder sans avoir à utiliser de matrice génératrice ni retrouver de matrice de contrôle.

✓ **Proposition** (Algorithme de codage pour un code cyclique)

Soit C (l'image *d'*) un code cyclique de polynôme générateur $g(X)$. Soit m un mot de source, de représentation polynomiale $P_m(x)$. Alors le mot image correspondant pour représentation polynomiale $g(x) \times P_m(x)$.

Pour un code cyclique, il existe donc des algorithmes de codage en $O(n \times log(n))$.

✓ **Remarque**

Soit C (l'image *d'*) un code cyclique (n, k). Alors un mot de longueur n est un mot de code si et seulement si sa représentation polynomiale est un multiple de $g(x)$, ce qui

revient à dire que le reste de la division euclidienne de sa représentation polynomiale par $g(x)$ est nul.

✓ **Proposition**: (Matrice de contrôle d'un code systématique)

Soit C un code cyclique de polynôme générateur

$$g(x) = a_0 + a_1 x + \cdots + a_{n-k} x^{n-k} \tag{1.26}$$

Alors (théorème fondamental) une matrice génératrice de ce code est

$$\begin{pmatrix} a_0 & 0 & & 0 \\ a_1 & a_0 & & \vdots \\ \vdots & \vdots & & 0 \\ a_{n-k} & a_{n-k-1} & \cdots & 0 \\ 0 & a_{n-k} & & a_0 \\ 0 & 0 & & a_1 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & a_{n-k} \end{pmatrix}$$

Et une matrice de contrôle associée à cette matrice génératrice est

$$\begin{pmatrix} b_k & b_{k-1} & \cdots & b_0 & 0 & \cdots & 0 \\ 0 & b_k & b_{k-1} & \cdots & b_0 & \cdots & 0 \\ & & & \vdots & & & \\ 0 & \cdots & 0 & b_k & b_{k-1} & \cdots & b_0 \end{pmatrix}$$

On peut donc décoder un code cyclique aussi bien qu'un code linéaire systématique. En fait, dans le cas d'un code cyclique quelconque, il existe des algorithmes de décodage meilleurs [14].

1.6 Codes BCH

1.6.1 Détermination des codes cycliques de longueur impaire

Si $n = 2p$ est pair, l'identité remarquable des maternelles donne

$$x^n + 1 = (x^p + 1)^2 \tag{1.27}$$

Il y a alors deux types de diviseurs de $x^n + 1$ les diviseurs de $x^p + 1$, et les produits de deux de ces précédents.

- **Soit n entier impair**

Pour déterminer tous les diviseurs de $x^n + 1$, il suffit de trouver la factorisation

complète de $x^n + 1$, c'est à dire d'écrire $x^n + 1 = \varphi_1 \varphi_2 + \cdots + \varphi_s$ avec les φ_i irréductibles. Comme on l'a vu, la seule racine de $x^n + 1$ dans F_2 est 1, et le polynôme $x^n + 1$ se factorise $x^n + 1 = (x + 1)(x^{n-1} + x^{n-2} + \cdots + x + 1)$, mais parfois le polynôme $x^{n-1} + \ldots + x + 1$ est lui-même factorisable bien qu'il n'ait pas de racines.

Au passage, remarquons que si $x^{n-1} + \cdots + x + 1$ avait des racines, la situation infiniment moins complexe ; mais $x^{n-1} + \cdots x + 1$ n'a pas de racine. Pour remédier à ce soucis, on va « imaginer » de telles racines.

✓ **Définition** (racine primitive ne de l'unité)

On appelle racine primitive ne de l'unité un nouveau nombre, α, tel que $\alpha \neq 1,\ \alpha^2 \neq 1, \cdots, \alpha^{n-1} \neq 1 \ et \ \alpha^n = 1$.

Ce nouveau nombre n'est $n_{i0}\ n_{i1}$, il n'appartient donc pas à F_2 .

✓ **Remarque** Les constatations suivantes s'imposent :

- α est une racine (imaginée) de $x^n + 1$.
- Mais alors, $\alpha^2, \alpha^3, \cdots, \alpha^n = 1$ sont aussi des racines de $x^n + 1$!

✓ **Proposition** (factorisation de $x^n + 1$ dans $F_2(\alpha)[x]$)

D'après la remarque qui précède, $x^n + 1 = (x + 1)(x + \alpha)(x + \alpha^2)\ldots(x + \alpha^{n-1})$ Finalement, on a bien une factorisation extéme de $x^n + 1$, mais le problème c'est que les coefficients des polynômes obtenus ne sont pas dans F_2 .

✓ **Théorème**

Soit $P(x)$ un polynôme à coefficients dans $F_2(\alpha)$, avec α une racine primitive n^e de l'unité. Si $P(x^2) = P(x)^2$, alors $P(x)$ est en rèalitè à coefficients dans F_2 !

D'où le corollaire suivant

✓ **Théorème**

Soit $P(x) = \prod_{i \in \Sigma}(x - \alpha^i)$, avec α une racine primitive n^e de l'unité.

Alors $P(x) \in F_2[x]$ si et seulement si Σ est stable par multiplication par 2 modulo n.

✓ **Définition** (Classes cyclotomiques)

On appelle classe cyclotomique de n une partie de $\{0,..,n-1\}$ stable par multiplication par 2 et minimale pour l'inclusion , donc un ensemble de la forme $\{j, 2j, 4j, \ldots, 2^{s-1}j\}$ $avec\ 2^s j \equiv j\ mod\ n$.

La classe cyclotomique $\{j, 2j, 4j, \ldots, 2^{s-1}j\}$ ($avec\ 2^s\ j \equiv j\ mod\ n$) sera appelée classe cyclotomique engendrée par j.

Exemple

- $x^7 + 1 = (x\ + 1)(\ x^3 + \ x^2 + 1\ )(\ x^3 + x + 1)$ et donc les codes cycliques de longueur 7 sont $(1,7), (3,7), (4,7)\ et\ (6,7)$.
- $x^9 + 1 = (x + 1)(\ x^2 + x + 1)(\ x^6 + \ x^3 + 1)$ et donc les codes cycliques de longueur 9 sont $(1,9), (2,9), (3,9), (6,9), (7,9)\ et\ (8,9)$.
- $x^1 + 1 = (x + 1)(\ x^{10} + \ x^9 + \ldots + x + 1\ )$ et donc les seuls codes cycliques de longueur 11 sont triviaux [18].

1.6.2 Les codes BCH primitifs stricts

Dorénavant, on se donne un entier r et on pose $n\ =\ 2^r - 1$ (en particulier n est impair). On se donne aussi une racine primitive ne de l'unité qu'on note α.

Commençons avec un théorème qui tue tout, à savoir enfin une minoration de la distance minimale d'un code !

✓ **Théorème BCH**

Notons C le code cyclique de associé à la partie stable $\sum = \{i_1\ , \ldots, i_{n-k}\}$ (c'est à dire de polynôme générateur $\prod_{i \in \sum} (x\ -\ \alpha^i)$).

S'il existe des entiers a et s tels que $\sum$ contienne $a\ +\ 1, a\ +\ 2, \ldots, a + s$, alors la distance minimale de C est supérieure ou égale à $s + 1$.

On continuera à se limiter à $n\ =\ 2^r - 1$ et on prendra $a\ =\ 0$.

✓ **Définition**: (Code BCH)

Soit t un nombre entier et $\delta =\ 2t\ +\ 1$.

On appelle code BCH primitif strict associé à δ, le code C associé à la plus petite partie stable contenant $1, 2, \ldots, \delta$. L'entier δ vérifie $\delta \leq$ d (inégalité parfois stricte) et on l'appelle la distance assignée à C [19].

Exemple: (Codes BCH de longueur 15)

Prenons $r = 4$, donc $n = 15$.

Les classes cyclotomiques sont $\{0\},\{1,2,4,8\},\{3,6,9,12\},\{5,10\} et \{7,11,13,14\}$.

On obtient donc quatre codes BCH qui sont les suivants

t	$\sum$	k	δ	d	$g(x)$
1	$\{1,2,4,8\}$	11	3	3	$g_1(x)$
2	$\{1,2,3,4,6,8,9,12\}$	7	5	5	$g_1(x)\ g_3\ (x)$
3	$\{1,2,3,4,5,6,8,9,10,12\}$	5	7	7	$g_1(x)\ g_3\ (x)\ g_5\ (x)$
4 *à* 7	$\{1,\ldots,14\}$	1	9 *à* 15	15	$1+\cdots+\ x^{14}$

Avec $g_1(x) = 1 + x + x^4$.

$g_3(x) = 1 + x + x^2 + x^3 + x^4$ donc $g_1(x)\ g_3\ (x) = 1 + x^4 + x^6 + x^7 + x^8$.

$g_5\ (x) = 1 + x + x^2$ donc $g_1\ (x)\ g_3\ (x)\ g_5\ (x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$.

1.6.3 Un algorithme de décodage des codes BCH

Pour les codes BCH, on dispose d'algorithmes de décodage algébriques. En général, ces algorithmes permettent la correction de t erreurs ou moins. Dans le cas où $\delta = d$, c'est donc un algorithme de décodage en deçà de la capacité de correction. Si $\delta < d$, on renonce même à corriger des mots contenant moins d'erreurs que la capacité de correction. La contrepartie est que ces algorithmes sont trés efficaces (de plus, t est donné par construction alors que e_c est en général difficile à déterminer), présentons un tel algorithme de décodage, dû à Peterson, **Gorenstein** et **Zierler** [14].

1.7 Conclusion

Pour n'est pas être exhaustif, dans ce chapitre on a essayé de présenter l'essentiel sur la théorie des codes correcteurs d'erreurs, et particulièrement les codes linéaires. Dans cette dernière classe on a exposé les codes correcteurs parfaits et les codes correcteurs cycliques. Parmi les codes parfaits et cycliques il existe ce qu'on appelle les codes de **Reed Solomon** RS qui font l'objet du chapitre suivant.

Chapitre 2

Les codes de Reed Solomon RS

Chapitre 2

Les codes de Reed Solomon RS

2.1 Introduction

Le code de **Reed Solomon** est un code non binaire qui travaille dans les « champs de **Galois** », et il est un code correcteur, dit « code parfait » inventé par les mathématiciens **Irving S. Reed** et **Gustave Solomon**. Il est très utilisé et permet de corriger des erreurs dues à la transition imparfaite d'un message. Pour cette raison il est très employé dans tous les domaines requérant des données fiables. Typiquement, dans les communications spatiales, télévision numérique et stockage de données [20].

En plus qu'il peut corriger des erreurs, le code de **Reed Solomon** permet aussi de corriger des effacements grâce à des symboles de contrôle ajoutés après l'information [21].

2.2 Problème principal du codage

Tout code C de paramètres (n, k, d)vérifie $d \leq n - k + 1$. Un code $C(n, k, d)$ vérifiant $d = n - k + 1$ est dit MDS (Maximum Distance Separable). [22]

Le but du codage est de trouver le code ayant le plus grand choix de mots codes de longueur déterminé avec une distance minimale donnée maximisée pour un alphabet donnée $GF(p^m)$. Notons que dans notre cas, les codes de **Reed Solomon**, nous avons des codes MDS (qui sont des codes parfait, ou des codes maximisant la distance entre les mots codes et ainsi abaissant la probabilité de confusion lors de décision ou lors de

décodage), soit des codes où la distance minimale est maximisée. C'est d'ailleurs pourcela, entre autre, que ces codes sont appréciés[23].

2.3 Généralités sur les Corps de Galois GF

Avant d'exposer les notions et les techniques de codage et de décodage des codes de **Reed Solomon**, il est important de parler même avec une manière rudimentaire sur les corps de **Galois** abrégé en GF.La notation pour la suite, sera la suivante

F représente un corps fini (Field).

F_p représente le corps fini à p éléments.

F^* représente le groupe multiplicatif du corps F.

2.3.1 Propriétés d'un corps fini [24]

On rappelle ici, quelques propriétés utiles des corps finis. Soit F un corps fini de caractéristique p ayant $q = n + 1$ éléments. Alors,

a) F est un F_p −espace vectoriel de dimension m, avec $q = p^m$;

b) Le groupe additif de F est isomorphe au groupe $(Z/pZ, +)^m$;

c) F^* est cyclique d'ordre $n = q - 1 = p^m - 1$;

d) Tout élément x de F^* vérifie $x^n = 1$.

Commentaire : Le point $c)$ nous affirme qu'il existe un élément α, dit primitif, qui engendre le groupe multiplicatif F^*. Le point $a)$ nous dit que F est un espace vectoriel de dimension m sur le corps F_p, le choix d'une base sera généralement les puissances entières de l'élément primitif, soit $1, \alpha, \alpha^2, \alpha^3, \cdots, \alpha^{m-1}$. Un élément c de F s'écrira alors de la manière suivante :

$$C = c_{m-1} \cdot \alpha^{m-1} + \cdots + c_1 \cdot \alpha^1 + c_0 \cdot 1 \qquad (2.1)$$

avec c_i appartenant à F_p pour $i = 0, \cdots, m - 1$.

2.3.2 Construction d'un corps fini

La construction d'un corps fini est basée sur le théorème suivant

Théorème 2.1[25]

Soit α un nombre algébrique sur F. Soit m le degré de son polynôme irréductible sur F. L'espace vectoriel engendré sur F par $1, \alpha, \alpha^2, \cdots, \alpha^{m-1}$ est alors un corps, et la dimension de cet espace vectoriel est m.

Ce théorème nous donne une méthode pour construire un corps avec une base donnée, mais ne nous donne pas de méthode pour trouver un élément primitif, α n'est pas forcément un élément primitif.

2.3.2.1 Représentation des éléments

Il existe principalement deux méthodes pour représenterles éléments d'un corps fini qui sont

1. F_q est un F_p −espace vectoriel de dimension m, on choisit une certaine base $\{b_1, \cdots, b_m\}$ de cet espace et on repère chaque élément de F_q par ses composantes sur cette base.
2. On prend un élément α de F^* qui engendre ce groupe et tout élément non nul de F_q s'écrit d'une manière, et d'une seule, sous la forme α^l, avec $l = 0, \cdots, n$.

Pour décrire un élément du corps, on utilisera exclusivement la première représentation, car elle permet d'associer un nombre à chaque élément du corps. Mais on prendra soin de tabuler ces deux représentations, la raison en est que la représentation (1) se prête très bien pour l'addition et que la représentation (2) est bien adaptée pour la multiplication. Comme on a besoin de ces deux opérations, on travaillera avec ces deux représentations qui seront tabulées une fois pour toute.

Avertissement : On ne s'intéressera, par la suite, que de corps finis de caractéristique 2, qui sont les seuls utilisés pour les codes de **Reed Solomon**.

2.3.2.2 Exemple illustratif [26]

Soit$F_2 = \{0,1\}$ lecorps à deux éléments. On vérifieque le polynôme $P(x) = x^2 + x + 1$ est bien irréductible sur F_2. Soit α une racine de $P(x)$,on aura alors $P(\alpha) = \alpha^2 + \alpha + 1 = 0$ donc $\alpha^2 = \alpha + 1$ ce qui veut dire que l'espace vectoriel engendré sur F_2 par $1, \alpha$ est alors un corps,et la dimension de cet espace vectoriel est de 2.

Le corps à 4 éléments sera alors $F_4 = \{0,1,\alpha,\alpha+1\}$. Si l'on choisit $\{1,\alpha\}$ comme base, la représentation 1) sera la suivante

$$0 = 0.\alpha + 0.1 = (0,0) = 0$$
$$1 = 0.\alpha + 1.1 = (0,1) = 1$$
$$\alpha = 1.\alpha + 0.1 = (1,0) = 2$$
$$\alpha + 1 = 1.\alpha + 1.1 = (1,1) = 3$$

Cette représentation est très pratique, car elle permet d'associer un nombre à chaque élément. Si l'on avait choisi $\{1,\alpha+1\}$ comme base, on aurait alors la représentation suivante

$$0 = 0.(\alpha+1) + 0.1 = (0,0) = 0$$
$$1 = 0.(\alpha+1) + 1.1 = (0,1) = 1$$
$$\alpha + 1 = 1.(\alpha+1) + 0.1 = (1,0) = 2$$
$$\alpha = 1.(\alpha+1) + 1.1 = (1,1) = 3$$

On comprend facilement que le choix de la base est fondamental pour savoir de quoi l'on parle.

Pour effectuer une addition sur ce corps, il faut voir que l'on additionne en fait deux vecteurs, et donc que l'on effectue l'addition composante par composante, et ceci sur F_2. Soit par exemple,

$$(0,1) + (0,1) = (0+0,1+1) = (0,0) \Longrightarrow 1+1 = 0$$
$$(1,1) + (1,0) = (1+1,1+0) = (0,1) \Longrightarrow 3+2 = 1$$
$$(1,1) + (0,1) = (1+0,1+1) = (1,0) \Longrightarrow 3+1 = 2$$
$$(0,1) + (1,0) = (0+1,1+0) = (1,1) \Longrightarrow 1+2 = 3$$

Sur un corps de caractéristique 2, l'addition revient à faire un OU EXCLUSIF (notation $\oplus$)entre deux nombres. Soit par exemple

$$3 + 2 = 11 \oplus 10 = 01 = 1$$
$$3 + 1 = 11 \oplus 01 = 10 = 2$$

Cela nous donne une opération assez simple à réaliser et à implémenter dans un programme.

Pour la multiplication, il nous faut trouver un élément primitif quigénère le groupe F_4^*. Dans notre exemple, α est un élément primitif, en effet

$$\alpha^0 = 1$$
$$\alpha^1 = \alpha$$

$$\alpha^2 = \alpha + 1$$

ce qui nous donne les trois éléments de notre groupe.

Si l'on prend $\{1, \alpha\}$ comme base, on aura alors la table de multiplication suivante

$1 \times 1 = 1$

$1 \times 2 = 1 \times \alpha = \alpha = 2$

$1 \times 3 = 1 \times (\alpha + 1) = \alpha + 1 = 3$

$2 \times 1 = 1 \times 2 = 2$

$2 \times 2 = \alpha \times \alpha = \alpha^2 = \alpha + 1 = 3$

$2 \times 3 = \alpha \times (\alpha + 1) = \alpha^2 + \alpha = \alpha + 1 + \alpha = 1$

$3 \times 1 = 1 \times 3 = 3$

$3 \times 2 = 2 \times 3 = 1$

$3 \times 3 = (\alpha + 1) \times (\alpha + 1) = \alpha^2 + \alpha + \alpha + 1 = \alpha^2 + 1 = \alpha = 2$

ce qui termine notre exemple.

2.4 Définition des codes de Reed Solomon

Les codes de **Reed Solomon** RS sont des codes linéaires cycliques non binaires. Les codes RS sont non binaires car les symboles n'appartiennent pas à $GF(2)$ mais à $GF(2^m)$. Un code RS est noté $RS(n,k)$ou $RS(n,k,t)$, où k est le nombre de symboles du mot d'information et n est le nombre de symboles du mot-code et$t$ représente le nombre maximum de symboles d'erreurs que ce code sera capable de corriger (voir la figure 2.1). Pour un code $RS(n,k)$ on définit la classe du code m, qui est le nombre de bits nécessaires pour représenter les symboles. n est relié à la classe du code par $n = 2^m - 1$. De plus, le nombre de symboles redondants $n - k$ est lié au nombre maximum d'erreurs corrigibles t par $n - k = 2t$[27].

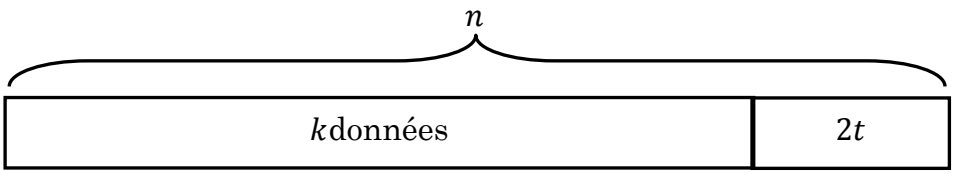


Fig. 2.1 Mot-code d'un code de Reed Salomon$RS(n,k)$.

Exemple 2.1

Soit le code $RS(15,9)$ qui est pour $k = 9$ symboles donne $n = 15$ symboles, chaque symbole étant contenu dans $GF(16)$, donc 4 bits par symbole. Un mot-code c qui contient $n = 15$ symboles s'écrit sous forme polynômiale comme suit

$$c(x) = c_{14} \cdot x^{14} + c_{13} \cdot x^{13} + \cdots + c_1 \cdot x^1 + c_0 \quad (2.3)$$

avec $c_i$contenu dans $GF(16)$pour $i = 0, \cdots, 14$. Les $n - k = 6$ coefficients $c_0, c_1, \ldots, c_5$ sont les symboles du contrôle de parité (parity check), et les coefficients$c_6, c_7, \cdots, c_{14}$représentent les symboles d'information à transmettre.

2.5 Avantages des codes de Reed Solomon

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

  $$n = 2^m - 1,\ n - k = 2t \text{ et } d_{min} = 2t + 1 = n - k + 1 \quad (2.4)$$

  oùd_{min} correspond à la distance minimale.
- Pour des corps de **Galois** $GF(q)$, lorsque l'on a besoin de coder des messages de longueur inférieur à q, l'encodage de RS est facilement utilisable. Puisque ce sont des codes MDS, la distance minimale est maximisée.
- Ils peuvent être combiné ou ajouté à autres codes afin de réaliser des codes plus efficaces.
- L'encodage est assez facile.
- Ils sont très efficace pour la correction d'erreurs consécutives, qu'ils soient utilisés seuls ou en conjonction avec d'autres codes. Bien entendu, cela est valide pour les cas où le nombre d'erreur demeure en dessous de la capacité de correction du code employé.
- Leurs algorithmes de décodages sont très développés.
- Pour des corps de **Galois** $GF(q = 2^m)$, on peut les représenter par des codes binaires puisque chacun des éléments de ce corps de **Galois** peuvent être représenté par une séquence binaire de longueur m [28][29].

2.6 Technique de codage[29]

Considérons un code de **Reed Solomon** $RS\ (n,k)$avec ses symboles dans $GF(2^m)$, où m est le nombre de bits par symbole. Soit $i(x) = c_{k-1} \cdot x^{k-1} + \cdots + c_1 \cdot x^1 + c_0$ le polynôme d'information, et soit $p(x) = c_{2t-1} \cdot x^{2t-1} + \cdots + c_1 \cdot x^1 + c_0$le polynôme de contrôle, le code de **Reed Solomon** sous sa forme polynômiale sera alors

$$c(x) = i(x) \cdot x^{2t} + p(x) = c_{n-1} \cdot x^{n-1} + \cdots + c_1 \cdot x^1 + c_0 \qquad (2.5)$$

avec c_i appartenant à $GF(2^m)$, pour $i = 0, \cdots, n-1$.

Un vecteur à n symboles, $(c_{n-1}, \cdots, c_1, c_0)$ est un code de **Reed Solomon** si et seulement si son polynôme correspondant $c(x)$ est un multiple du polynôme générateur $g(x)$. La méthode courante pour construire un tel polynôme, est de diviser $i(x) \cdot x^{2t}$ par $g(x)$ et d'additionner le reste à $c(x)$. En effet

$$i(x) \cdot x^{2t} = q(x) \cdot g(x) + r(x) \qquad (2.6)$$

où $r(x)$ est le reste de la division de $i(x)$ par $g(x)$.

$$c(x) = i(x) \cdot x^{2t} + p(x) = q(x) \cdot g(x) + r(x) + p(x) = q(x) \cdot g(x) \qquad (2.7)$$

Pour que $c(x)$ soit un multiple de $g(x)$, soit $c(x) = q(x) \cdot g(x)$, il faut que $p(x) = r(x)$. Comme on travaille toujours sur des corps de caractéristique 2, l'opération de soustraction sera toujours égale à l'opération d'addition, soit de manière algébrique $-1 = +1$.

Cela nous donne une méthode pour construire le polynôme de contrôle, il suffit de prendre le reste de la division du polynôme $i(x) \cdot x^{2t}$ par $g(x)$.

Il nous reste encore à construire le polynôme générateur $g(x)$. Il est définit comme 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 $g(x)$. On a alors

$$g(x) = (x + \alpha)(x + \alpha^2) \cdots (x + \alpha^{2t}) = x^{2t} + g_{2t-1} \cdot x^{2t-1} + \cdots + g_1 \cdot x^1 + g_0 \qquad (2.8)$$

où les coefficients g_i appartiennent à $GF(2^m)$, et α est un élément primitif de $GF(2^m)$.

Exemple 2.2 (exemple de codage)

Soit le code $RS(15,9)$, avec $n = 15$, et soit un polynôme $P(x)$ irréductible et primitif sur F_2, avec $P(x) = x^4 + x^3 + 1$. Soit α une racine de $P(x)$, qui est également un élément

primitif, et sera utilisé pour construire $GF(16)$. La base utilisée sera alors $\{\alpha^3, \alpha^2, \alpha, 1\} = \{8,4,2,1\}$, il y aura donc 16 symboles, qui vont de 0 à 15.

Maintenant, on désire coder les 9 symboles d'informations suivants $i = [9,8,7,6,5,4,3,2,1]$ (matrice 1×9). Soit sous forme polynômiale

$$i(x) = 9x^8 + 8x^7 + 7x^6 + 6x^5 + 5x^4 + 4x^3 + 3x^2 + 2x + 1 \qquad (2.9)$$

Le polynôme générateur est

$$g(x) = (x+2)(x+4)(x+8)(x+9)(x+11)(x+15) \qquad (2.10)$$

$$= x^6 + 3x^5 + x^4 + 4x^3 + 7x^2 + 13x + 15$$

$$c(x) = i(x).x^{2t} + r(x) \qquad (2.11)$$

$$= i(x).x^{2t} + 6x^5 + 15x^4 + 15x^3 + 15x^2 + 11x + 14$$

$$= (9x^8 + 10x^7 + 9x^6 + x^5 + x^4 + 12x^3 + 3x^2 + 11x + 4) \cdot g(x)$$

$$= q(x) \cdot g(x)$$

2.7 Décodage de codes de Reed Solomon

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

Plusieurs étapes sont nécessaires pour le décodage de ces codes, à savoir

- Calcul du syndrome
- Calcul des polynômes de localisation des erreurs et de d'amplitude
- Calcul des racines et évaluation des deux polynômes
- Sommation du polynôme constitué et du polynôme reçu pour reconstituer l'information de départ sans erreur.

Considérons un code de **Reed Solomon**$c(x)$ correspondant au code transmis, et soit $d(x)$ le code que l'on reçoit. Le code d'erreur est définit par

$$e(x) = d(x) - c(x) = d(x) + c(x) \qquad (2.12)$$

car le $(-)$ et le $(+)$ sont équivalentsdans $GF(2^m)$.

2.7.1 Calcul des syndromes [30]

La première opération à faire est de calculer le syndrome$S(x)$. Le syndrome est défini comme le reste de la division du polynôme reçu $d(x)$par le polynôme générateur $g(x)$.

$$d(x) = q_d(x)g(x) + S(x) \tag{2.13}$$

avec

$$S(x) = S_{2t} \cdot x^{2t-1} + \cdots + S_3 \cdot x^2 + S_2 \cdot x \ + S_1 \tag{2.14}$$

En tenant compte que le mot-code $c(x)$ est un multiple de $g(x)$, c'est-à-dire

$$c(x) = q_c(x)g(x) \tag{2.15}$$

l'équation (2.7) entraine que

$$e(x) = (q_d(x) + q_c(x))g(x) + S(x) = d(x) = q_e(x)g(x) + S(x) \tag{2.16}$$

Par conséquent, si le syndrome est nul, le code que l'on reçoit ne contient pas d'erreurs sinon il y a d'erreurs.

Le calcul du syndrome ce fait alors en prenant le reste de la division euclidienne de $d(x)$par $g(x)$, et comme l'opération division est toujours une opération complexe, on peut calculer ce syndrome en résolvant dans $GF(2^m)$le système d'équations

$$S(\alpha^i) = S_{2t} \cdot \alpha^{i\cdot(2t-1)} + \cdots + S_3 \cdot \alpha^{i\cdot 2} + S_2 \cdot \alpha^i + S_1 \tag{2.17}$$

avec$i = 1, \cdots, 2t$

$$\text{car} g(\alpha^i) = 0 \text{et} d(\alpha^i) = q_d(\alpha^i)g(\alpha^i) + S(\alpha^i) = S(\alpha^i) \tag{2.18}$$

Exemple 2.3

On a un code de $RS(15,9)$, donc $15 - 9 = 6$ symboles de contrôle $(2t = 6)$. On reçoit le polynôme suivant $d(x)$et on veut calculer le syndrome $S(x)$

$$d(x) = \alpha x^{14} + \alpha^2 x^{12} + \alpha^{13} x^4 \tag{2.19}$$

On a

$S(\alpha) = d(\alpha) = \alpha\alpha^{14} + \alpha^2\alpha^{12} + \alpha^{13}\alpha^4 = \alpha^{1+14} + \alpha^{2+12} + \alpha^{13+4} = \alpha^6$

$S(\alpha^2) = \alpha(\alpha^2)^{14} + \alpha^2(\alpha^2)^{12} + \alpha^{13}(\alpha^2)^4 = \alpha^7$

$S(\alpha^3) = \alpha(\alpha^3)^{14} + \alpha^2(\alpha^3)^{12} + \alpha^{13}(\alpha^3)^4 = \alpha^{12}$

$S(\alpha^4) = \alpha(\alpha^4)^{14} + \alpha^2(\alpha^4)^{12} + \alpha^{13}(\alpha^4)^4 = 0$

$S(\alpha^5) = \alpha(\alpha^5)^{14} + \alpha^2(\alpha^5)^{12} + \alpha^{13}(\alpha^5)^4 = \alpha$

$S(\alpha^6) = \alpha(\alpha^6)^{14} + \alpha^2(\alpha^6)^{12} + \alpha^{13}(\alpha^6)^4 = \alpha^8$

2.7.2Évaluation du polynôme de locations d'erreurs

Soit M l'ensemble des positions d'erreurs, on définit le polynôme de localisation (locator polynomial) par

$$l(x) = (1 - \alpha^{p_1} \cdot x)(1 - \alpha^{p_2} \cdot x) \cdots (1 - \alpha^{p_j} \cdot x) \text{ avec} p_1, \cdots, p_j \text{ dans } M \qquad (2.20)$$

Si $b = \alpha^i$ est une racine de $l(x)$, donc $l(b) = 0$, on aura alors un des monômes qui s'annule, soit

$$(1 - \alpha^{p_i} \cdot b) = 0 \qquad (2.21)$$

ce qui nous donne la relation suivante

$$b = \alpha^{-p_i} = \alpha^{q-1}.\alpha^{-p_i} = \alpha^{q-p_i-1} = \alpha^i \text{car} \alpha^{q-1} = 1 \qquad (2.22)$$

La position de l'erreur sera alors donné par $p_i = q - 1 - i$ qui est positif et que l'on connait, puisque l'on connait $b = \alpha^i$, racine de$l(x)$. Il nous faudra donc chercher toutes les racines de$l(x)$. Il ne faut pasoublier que α est un élément primitif, et que tout élément de $GF(2^m)^*$ s'exprime comme une puissance de l'élément primitif, qui est un générateur du groupe $GF(2^m)^*$.

2.7.3 Equation fondamentale

Il nous reste encore à calculer $l(x)$. Pour cela, on fait usage de la relation fondamentale suivante

$$l(x) \cdot S(x) = w(x) + u(x) \cdot x^{2\,t} \qquad (2.23)$$

Comme l'on travaille dans $GF(2^m)$, on peut l'écrire sous la forme

$$l(x) \cdot S(x) + u(x) \cdot x^{2\,t} = w(x) \qquad (2.24)$$

On peut voir cette équation comme la relation de **Bézout** avec des nombres, soit $v \cdot b + u \cdot a = w$ où a et b sont des nombres donnés, et w est le plus grand diviseurcommun des nombres $a$et b. $u$et$v$ sont deux nombres qui satisfont cette relation. Notre équation ci-dessus est identique, sauf qu'elle s'applique à des polynômes.

On pose $b = S(x)$, $a = x^{2t}$et on applique l'algorithme d'**Euclide** pour trouver $u(x)$, $l(x)$, $w(x)$, et les racines de $l(x)$ sont utilisées pour trouver la position de l'erreur.

Soit b_i une racine de $l(x)$, la valeur ou l'amplitude de l'erreur à la position p_i se calcule de la manière suivante

$$e_i = w(b_i)/l'(b_i) \qquad (2.25)$$

où $l'(x)$ est la dérivée du polynôme $l(x)$.

En fait, l'algorithme d'**Euclide** ne donne pas $l(x)$, $v(x)$, $w(x)$ mais $Kl(x)$, $Kv(x)$, $Kw(x)$, c'est-à-dire à une constante K près. Mais pour trouver les racines de $l(x)$, qu'on ait $l(x)$ ou $Kl(x)$ cela nous donne le même résultat, et comme la valeur ou l'amplitude de l'erreur est calculée par le rapport $Kw(b_i) / Kl'(b_i)$, on obtient le bon résultat.Il y a plusieurs méthodes de calcul des polynômes de localisation des erreurs et de d'amplitude, dans le cadre de ce mémoire on ne traitera que deux méthodes, le décodage selon l'algorithme d'**Euclide** et le décodage selon l'algorithme de **Reed Solomon**. Une fois les polynômes calculés en utilisant l'algorithme de **Horner**, on calculera les valeurs à soustraire pour obtenir le mot-code sans erreur[31][32].

2.7.4 Algorithme d'Euclide

On rappelle ici, le principe de l'algorithme d'Euclide appliqué à l'équation

$$v \cdot b + u \cdot a = w \tag{2.26}$$

1) Initialisation

$$q_0 = - \quad ; \quad r_0 = a \quad ; \quad u_0 = 1 \quad ; v_0 = 0$$
$$q_1 = - \quad ; \quad r_1 = b \quad ; \quad u_1 = 0 \quad ; \quad v_1 = 1$$

2) Calcul de q

On divise r_{i-1}par r_i, ce qui nous donne un quotient q et un reste r, soit $r_{i-1} = qr_i + r$ et on pose $q_{i+1} = q$

3) Calculs

$$r_{i+1} = r_{i-1} - q_{i+1} \cdot r_i$$
$$u_{i+1} = u_{i-1} - q_{i+1} \cdot u_i$$
$$v_{i+1} = v_{i-1} - q_{i+1} \cdot v_i$$

4) Test

Si r_{i+1} est différent de 0, on incrémente i et on retourne au point 2), et si $r_{i+1} = 0$ l'algorithme est terminé. On a alors,

a)r_i est le plus grand diviseur commun entre $a$et b.

b) $r_i = v_i.b + u_i.a$.

En fait, la relation b) est valable pour tout i[33].

2.7.5Algorithme de Reed Solomon

L'algorithme de décodage des codes de **Reed Solomon** sera le suivant

1) Calcul du polynôme syndrome $S(x)$.

Si $S(x) = 0$, il n'y a pas d'erreurs : **STOP**.

2) On applique l'algorithme d'**Euclide** avec

$$a(x) = x^{2t} \text{et } b(x) = S(x) \tag{2.27}$$

L'algorithme se termine dès que le degré de $r_i(x)$ est inférieur à t.

Si $r_i(x) = 0$, il y a plus que t erreurs : **STOP**.

3) On pose $L(x) = Kl(x) = v_i(x)$.

On cherche toutes les racines de $L(x)$:$b_1, \cdots, b_r$.

4) Pour chaque racine $b_i = \alpha^j$, la position p_i de l'erreur est

$$p_i = q - j - 1 = 2^m - j - 1 \tag{2.28}$$

5) On pose $W(x) = Kw(x) = r_i(x)$.

La valeur de l'erreur à la position p_i est donnée par

$$e_i = \mathrm{W}(b_i) / \mathrm{L'}(b_i) = \mathrm{K}.\mathrm{w}(b_i) / \mathrm{K}.\mathrm{l'}(b_i) = \mathrm{w}(b_i) / \mathrm{l'}(b_i) \tag{2.29}$$

La nouvelle valeur à la position p_i sera alors $c_i = d_i + e_i$.On évalue toujours un polynôme ainsi que sa dérivée à l'aide d'unschéma d'**Horner**, et ceci afin de minimiser le nombre de calculs[34].

2.7.6 Schéma d'Horner

On donne ici, le schéma d'**Horner** pour évaluer un polynôme et sa dérivée à une valeur donnée.

$$\text{Soit } p(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \cdots + a_1 \cdot x + a_0 \tag{2.30}$$

On désire évaluer notre polynôme et sa dérivée à $x = \beta$, soit $p(\beta)$ et $p'(\beta)$.

I) Initialisation

$$b_n = \mathrm{a}_n$$
$$\mathrm{a}_n = b_n$$
$$k = n$$

II) Itération

$$b_k = a_k + \beta \cdot b_{k+1}$$

Si $k > 0$ alors $c_k = b_k + \beta c_{k+1}$

III) Contrôle

Si k est différent de 0, on décrémente k et on retourne aupoint II, sinon l'algorithme est terminé.

On aura alors $p(\beta) = b_0 \text{et} p'(\beta) = c_1$[35].

Exemple 2.4 (exemple de décodage)

Reprenons notre exemple 2.2 ci-dessus, soit notre code de **Reed Solomon**

$$c(x) = 9x^{14} + 8x^{13} + 7x^{12} + 6x^{11} + 5x^{10} + 4x^9 + 3x^8 + 2x^7$$
$$+1x^6 + 6x^5 + 5x^4 + 15x^3 + 15x^2 + 11x + 14 \quad (2.31)$$

Choisissons un polynôme d'erreur, soit $e(x) = 7x^{11} + 10x^2$.Notre code reçu sera alors,

$$d(x) = c(x) + e(x) \quad (2.32)$$
$$d(x) = 9x^{14} + 8x^{13} + 7x^{12} + 1x^{11} + 5x^{10} + 4x^9 + 3x^8 + 2x^7$$
$$+1x^6 + 6x^5 + 5x^4 + 15x^3 + 5x^2 + 11x + 14$$

On calcule le syndrome en $\alpha^i : S(\alpha^i) = d(\alpha^i) = d(2^i)$ avec $i = 1, \cdots, 6$, cequi nous donne le syndrome sous forme polynômiale

$$S(x) = 1x^5 + 15x^4 + 7x^3 + 8x^2 + 11 \quad (2.33)$$

On pose $a(x) = x^6$et $b(x) = s(x)$, et on applique l'algorithme d'**Euclide**, cela nous donne

$L(x) = 6x^2 + 9x + 1$

$W(x) = 5x + 11$

$U(x) = 6x$

On cherche les 2 racines pour $l(x)$, on trouve

$$b_1 = 9 = \alpha^4 \text{ et } b_2 = 6 = \alpha^{13}$$

d'où l'on tire

$$p_1 = 15 - 4 = 11 \text{ et } p_2 = 15 - 13 = 2.$$

Il nous reste à trouver la valeur de l'erreur à ces positions. Pour cela, on calcule

$$e_1 = \text{W(9) / L'(9)} = 13 / 9 = 7 \quad (2.34)$$
$$e_2 = W(6)/L'(6) = 12/9 = 10 \quad (2.35)$$

Ce qui nous donne un polynôme de correction

$$v(x) = 7x^{11} + 10x^2$$

On remarquera que l'on a $v(x) = e(x)$, et que si l'on effectue la correction, cela nous donne $c'(x) = d(x) + v(x) = c(x) + e(x) + v(x) = c(x)$ et que l'on retrouve notre code de départ, ce qui bien le but recherché[36].

2.7.7 Détection d'erreurs dont le nombre est supérieur à t

Voici les principaux contrôles qu'ils faut effectuer afin de détecter un nombre d'erreurs supérieur à t.

Après avoir terminé l'algorithme d'**Euclide**, deux cas peuvent se présenter

- x^tdevise $S(x) \Longrightarrow w(x) = 0$.
- $S(x) = w(x) \Longrightarrow l(x) = 1$ et donc pas de racine.

Lors de la recherche des racines de $l(x)$, trois cas peuvent se présenter

- 0 est racine de $l(x)$.
- $l(x)$a des racines multiples.
- $l(x)$ ne se décompose pas en produit de facteur linéaire.

On trouve donc moins de racine qu'il ne devrait y en avoir.

Lors de la recherche de la position de l'erreur, un cas peu se présenter

- $p < 0$ ou $p \geq n$.

Il est clair que lorsque l'on a un nombre d'erreurs supérieur à t, tous ces cas peuvent se présenter. Il faut donc absolument faire ces contrôles afin de détecter cette situation qui peut dès lors se présenter.

2.8 Conclusion

Dans ce chapitre on a exposé l'aspect mathématique d'un type très important des codes correcteurs d'erreurs à savoir les codes de **Reed Solomon** RS. Ces codes sont utilisés dans 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.

Comme on a vu, les codes envisagés fondent leur théorie sur les corps finis dits corps de **Galois**. En mathématiques et plus précisément en algèbre, un corps fini est un

corps commutatif qui est par ailleurs fini. Aisomorphisme près, un corps fini est entièrement déterminé par son cardinal, qui est toujours une puissance d'un nombre premier, ce nombre premier étant sa caractéristique. Pour tout nombre premier p et tout entier non nul m, il existe un corps de cardinal p^m, qui se présente comme l'unique extension de degré m du corps premier Z/pZ.

On a présenté dans ce chapitre avec une façon rudimentaire la théorie des corps de **Galois** notés GF pour une caractéristique $p = 2$, où on a expliqué à l'aide des exemples détaillés les opérations d'addition et de multiplication dans ces corps finis.

La technique de codage des codes RS ainsi que les techniques de décodage sont bien exposées. Pour n'est pas être exhaustif, on a présenté dans ce chapitre deux algorithmes de décodage des codes RS à savoir, l'algorithme d'**Euclid** et l'algorithme de **Reed Solomon**.

Pour mieux comprendre l'effet d'un code RS sur un système de communication numérique on doit étudier son pouvoir de correction d'erreurs dans ce système, soit sur le champs réel ou soit à l'aide d'une simulation. La simulation de ces codes fait l'objet du chapitre suivant.

Chapitre 3

Etude des performances de codes RS par simulation numérique

Chapitre 3

Etude des performances de codes RS par simulation numérique

3.1Introduction

La simulation permet l'évaluation des performances des systèmes de communication numériques, et constitue aussi un outil puissant qui aide à l'étude et la conception de tels systèmes. Nous citons quelques raisons montrant l'importance de cet outil

- Quelques niveaux d'analyse ne sont pas faisables avec les outils mathématiques traditionnels, surtout qu'on est en face d'un modèle de canal et d'un système complexe.
- Il est très facile de passer d'un modèle de simulation d'un système à son implémentation matérielle, et pour cela la distinction entre un modèle de simulation et un prototype d'un système donné devient minimale, d'où l'intérêt économique de la simulation qui réduit considérablement le temps et le coût de réalisation d'un système, et qui se prête plus facilement à la correction et à l'amélioration qu'un prototype.

L'évaluation des performances se fait en calculant la probabilité d'erreur pour un système de communication numérique donné et un bruit donné[37].

3.2 Modèle de simulation

Le modèle utilisé lors de notre simulation est représenté par la figure 3.1. Il comporte

- Une source d'information binaire ;

- Un codeur de RS ;
- Un modulateur BPSK ;
- Un canal AWGN supposé idéal (sans distorsions) ;
- Une source de bruit gaussien ;
- Un décodeur de RS ;
- Un comparateur pour calculer le nombre de bits erronés après le décodage afin d'estimer la probabilité d'erreurs binaire BER.

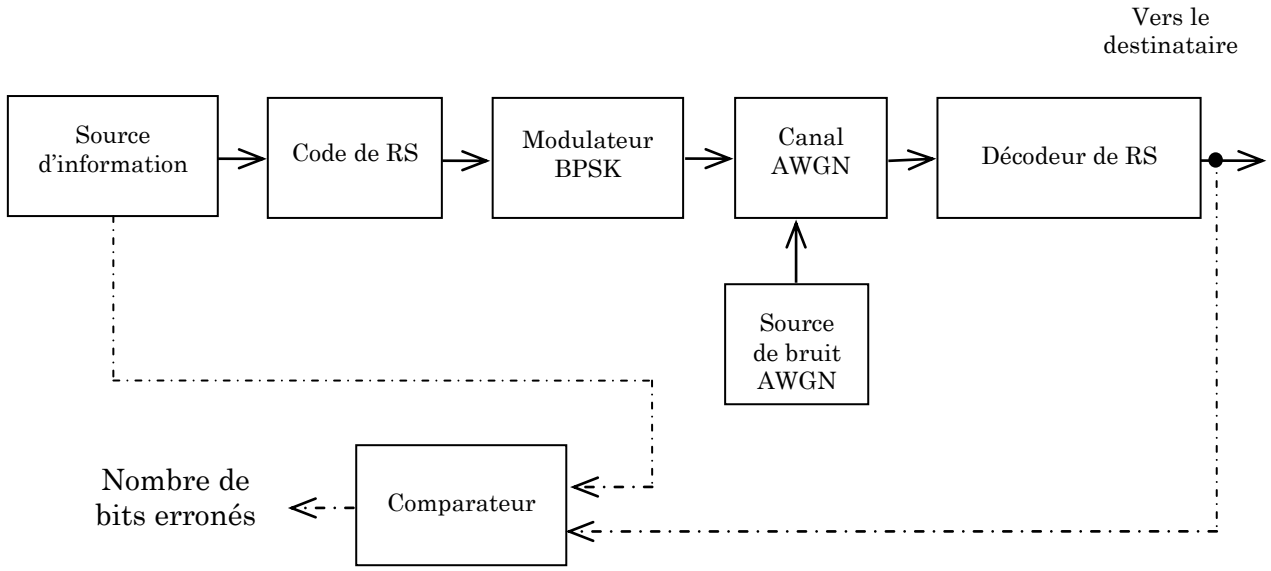


Fig. 3.1 Modèle de simulation.

3.2.1 Source d'information

Pour que l'évaluation soit effectuée dans des conditions proches de ce qui est rencontre en exploitation, on doit choisir une séquence d'information qui simule le mieux possible le trafic réel tout en permettant une mesure simple.

La méthode universellement utilisée dans toutes les simulations, consiste à utiliser une séquence dite pseudo-aléatoire, séquence périodique dont les propriétés statistiques sont «voisines » de celles d'un trafic réel «aléatoire» qui est généralement le résultat d'un brassage. Le générateur d'une telle séquence est constitué d'un registre à décalage comportant m étages, et des additionneurs modulo-2[38].Le choix de m détermine la période de la séquence, qui est égale à $2^m - 1$.

Les connexions aux additionneurs modulo-2 sont déterminées par des polynômes primitifs de degré m ayant la forme [39]

$$h(x) = \sum_{j=0}^{m} h_j x^j \quad \text{avec} \quad \begin{cases} h_0 = h_m = 1 \\ h_j = 0 \text{ ou } 1 \end{cases} \text{ pour } j = 1, \ldots, m-1.$$

Un polynôme primitif de degré m est un polynôme irréductible qui devise$x^{2m-1} +$ 1et ne devise pas $x^n + 1$ pour$n < 2^m - 1$.

Chaque coefficient h_j égal à 1 correspond à une connexion.

La séquence générée comporte 2^{m-1} éléments égaux à 1 et$2^{m-1} - 1$ à égaux à 0 ce qui donne une répartition presque uniforme des 1 et des 0.

Les principales propriétés de ces séquences sont bien détaillées dans [40][41].

- **Choix de la longueur de la séquence utilisée**

Le choix de la longueur de la séquence à utiliser dans la simulation, c'est-à-dire la longueur du registre qui l'engendre, est basé sur le fait qu'avec de faibles longueurs de registre (3 ou 4) on risque d'obtenir des résultats différents de la réalité, car une séquence courte ne représente pas un nombre statistiquement suffisant de configurations, donc il est recommandé d'utiliser des séquences relativement longues,dontleurs nombres des registres soient supérieurs à 15 [42]. A cause du facteur de temps et de la mémoire limitée de l'ordinateur utilisé, nous avons choisi$m = 16$qui nous donne une séquence de longueur $2^m - 1 = 65535$.

3.2.2 Codes de RS utilisés

Comme on le sait, deux quantités sont nécessaires pour la caractérisation d'un RS : n et k, sachant que n s'écrit sous la forme $2^m - 1$. Dans cette simulation on a adopté le **Matlab**. La difficulté du codage et du décodage RS réside dans le fait que les instructions de codage et de décodage RS de **Matlab** ont des entrées et des sorties au format d'éléments de $GF\ (2^m)$. Il faut donc

- convertir les bits de toutes séquence binaire en éléments de $GF\ (2^m)$ avant les étapes de codage et de décodage RS ;
- convertir les éléments de $GF\ (2^m)$ en bits après les étapes de codage et de décodage RS.

Par ailleurs, la conversion ne peut pas se faire directement et nécessite une étape intermédiaire où les bits et les éléments de $GF\ (2^m)$ sont convertis en entiers.

Dans cette simulation on a choisi quatre codes RS, à savoir

- $RS(15,3)$ qui est capable de corriger $t = 6$ erreurs ;
- $RS(15,5)$ qu'il peut corriger $t = 5$ erreurs ;
- $RS(7,3)$ qui possède un nombre d'erreurs corrigibles $t = 2$ erreurs ;
- $RS(31,23)$ où le nombre d'erreurs corrigibles est $t = 4$ erreurs.

3.2.3 Modulation utilisée

Pour la modulation utilisée dans la simulation, notre choix reste limité à la modulation BPSK pour tous les codes RS employés. La modulation des bits de sortie du turbo code est alors [43].

$$Y = \begin{cases} +\sqrt{E_b} & pour\ un\ bit\ de\ sortie\ X = 0 \\ -\sqrt{E_b} & pour\ un\ bit\ de\ sortie\ X = 1 \end{cases} \tag{3.1}$$

Pour simplifier les choses, on a pris $\sqrt{E_b} = 1$. La modulation devient tout simplement

$$Y = \begin{cases} +1\ pour\ un\ bit\ X = 0 \\ -1\ pour\ un\ bit\ X = 1 \end{cases} \quad \text{ou} \quad Y = 1 - 2X \tag{3.2}$$

Si la densité spectrale de puissance bilatérale du bruit est $N_0/2 = \sigma^2$ où σ^2est la variance du bruit, alors le rapport signal sur bruit est

$$SNR = \frac{E_b}{N_0} = \frac{1}{2\sigma^2} \tag{3.3}$$

soit en dB

$$SNR = 10\log\left(\frac{1}{2\sigma^2}\right)\ (dB) \tag{3.4}$$

Cette relation nous permet d'exprimer la variance du bruit en fonction du SNR comme suit

$$\sigma = \frac{1}{\sqrt{2}} \cdot 10^{-SNR(dB)/20} \tag{3.5}$$

3.2.4 Source de bruit

Le bruit envisagé est AWGN

- blanc ;
- additif ;
- gaussien.

On n'a pas pris en compte de l'effet du filtre de réception, c'est-à-dire que le bruit reste AWGN à la réception (car le filtrage rend ce bruit coloré).

Dans le **Matlab**, on peut générer un bruit blanc gaussien

- de variance égale à l'unité
- de moyenne nulle, à l'aide de l'instruction **randn**

Pour avoir un bruit de varianceσ^2, on multiplie les échantillons générés du bruit parσ^2.

Soient $p$la probabilité à calculer et $\hat{p}$ la valeur estimée par la simulation. Dans la méthode de Monte-Carlo, Si on désire connaître la précision relative $r$pour un nombre d'essais N_e, et une confiance$1 - c$, on applique la formule [44]

$$r^2 \geq 2\left(erfc^{-1}(c)\right)^2 \frac{1}{N_e}\left(\frac{1}{p} - 1\right) \tag{3.6}$$

L'inconvénient majeur de cette méthode réside dans le fait que le nombre d'essais Ne, croît rapidement lorsque les probabilités d'erreur sont faibles. Malheureusement, on n'a pas pu calculer les précisions r dans notre simulation, à cause du facteur de temps !

3.2.5Décodeur de RS

Pour le décodage des codes de RS, il existe plusieurs algorithmes [43], mais dans notre simulation on a utilisé l'algorithme de décodage par calcul du syndrome exposé dans le chapitre 2, à savoir l'algorithme de **Reed Solomon**.

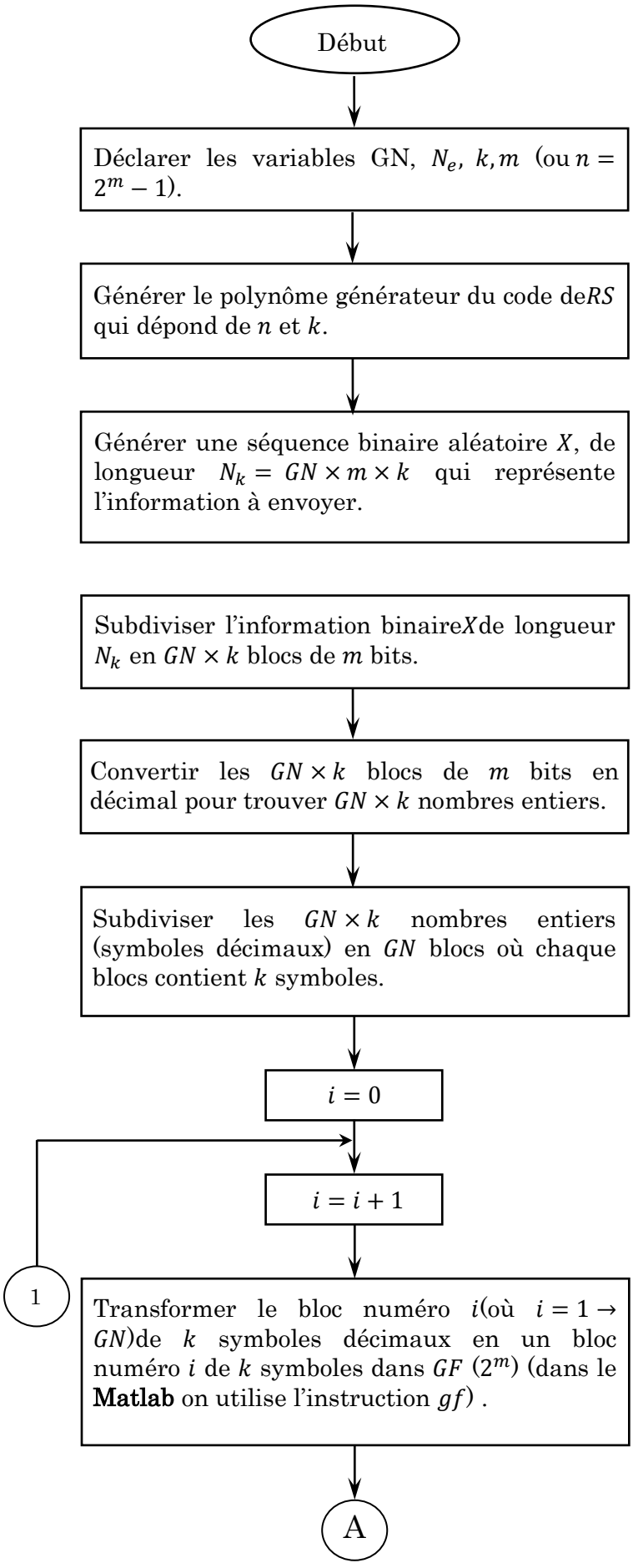
3.2.6 Calcul du taux d'erreurs binaire BER

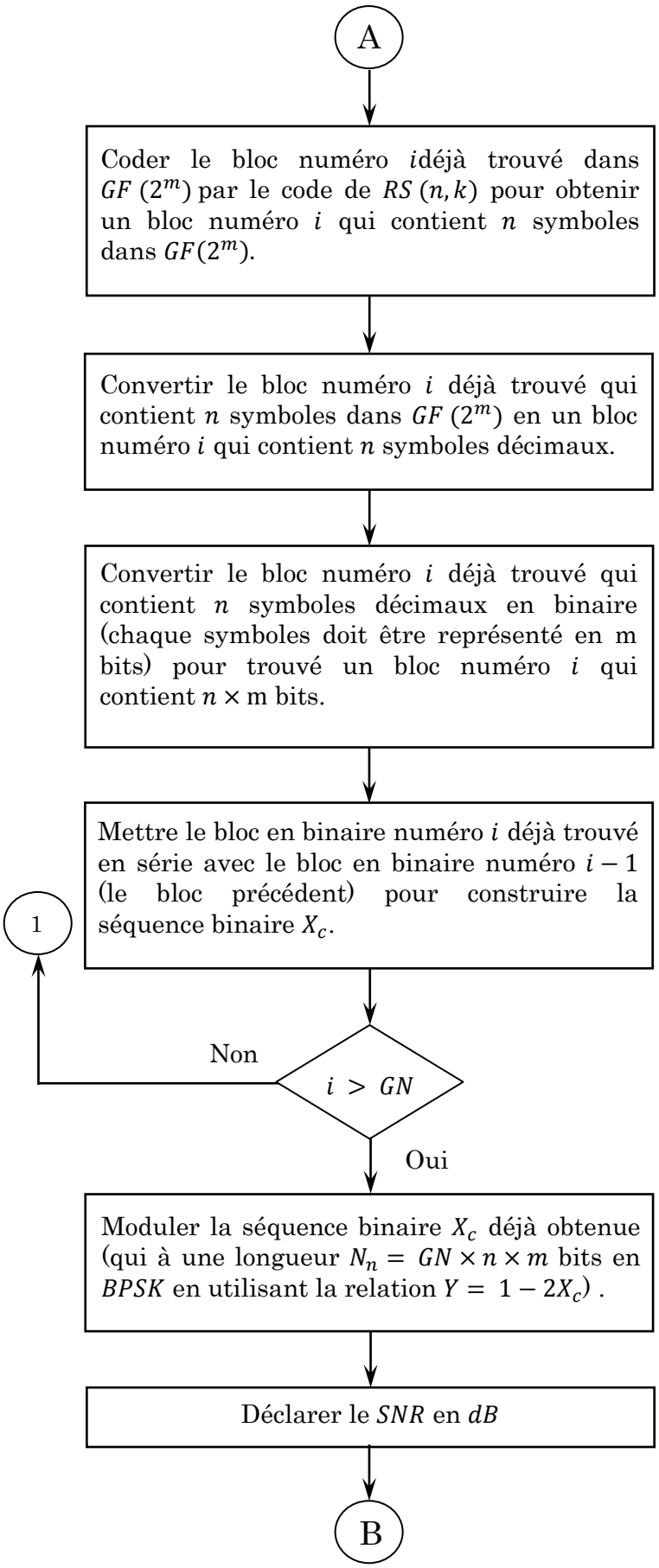
La séquence d'information pseudo-aléatoire générée par la source est de longueur $2^{16} - 1 = 65535$ bits, qu'on doit le rendre un multiple de $m \cdot k$ (le nombre de bits par symbole × le nombre de symboles d'entrée du codeur RS), en ajoutant ou éliminant quelques bits.

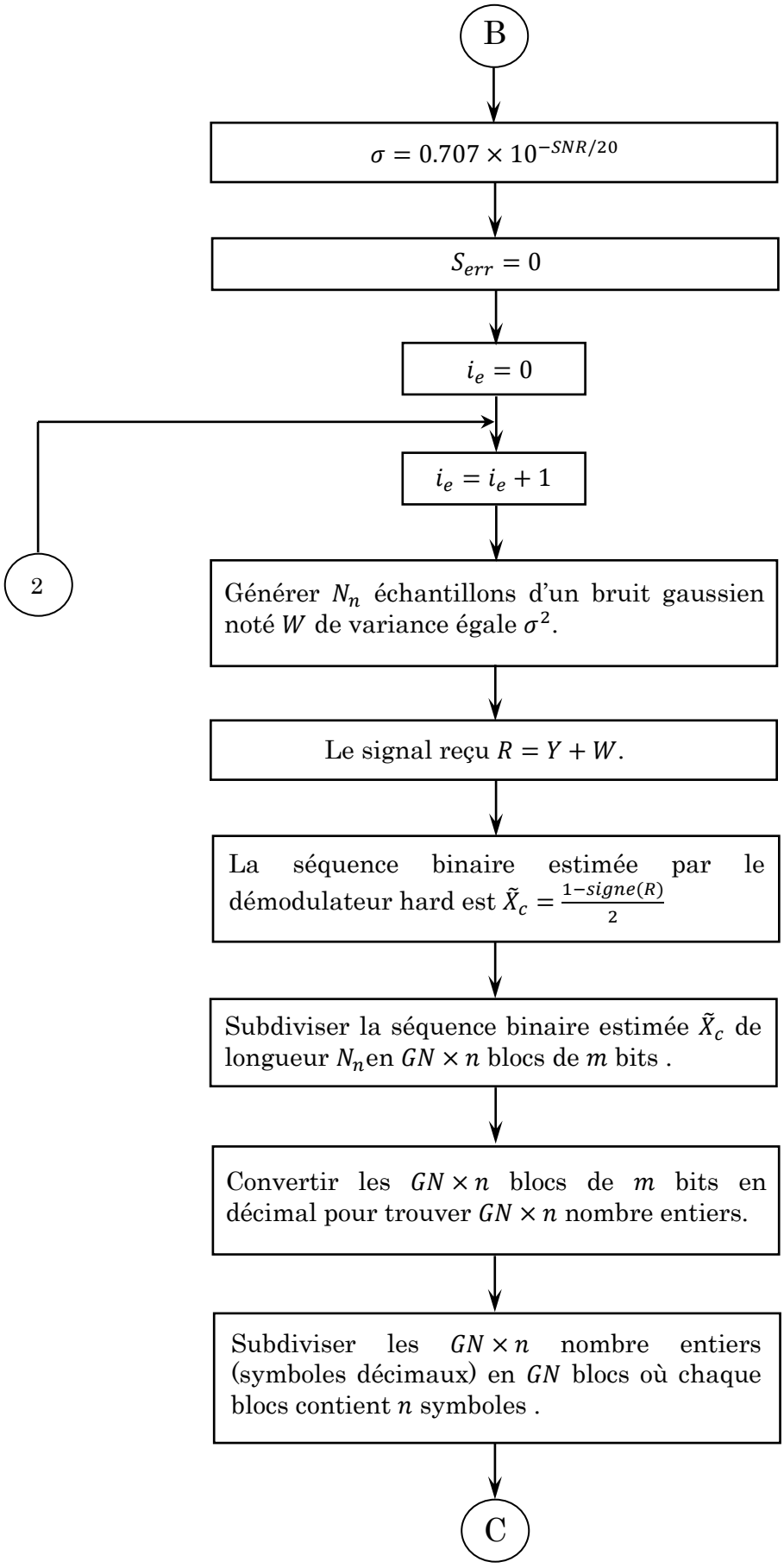
Pour un SNR donné, après le décodage de RS on compte le nombre des bits erronés noté $nber$ par le truchement d'une comparaison entre la vraie séquence x et la séquence estimée $\hat{x}$ délivrée par le bloc de décodage. Le taux d'erreurs binaire BER est le rapport entre ce nombre des bits erronés et la longueur totale de la séquence notée $lseq$,

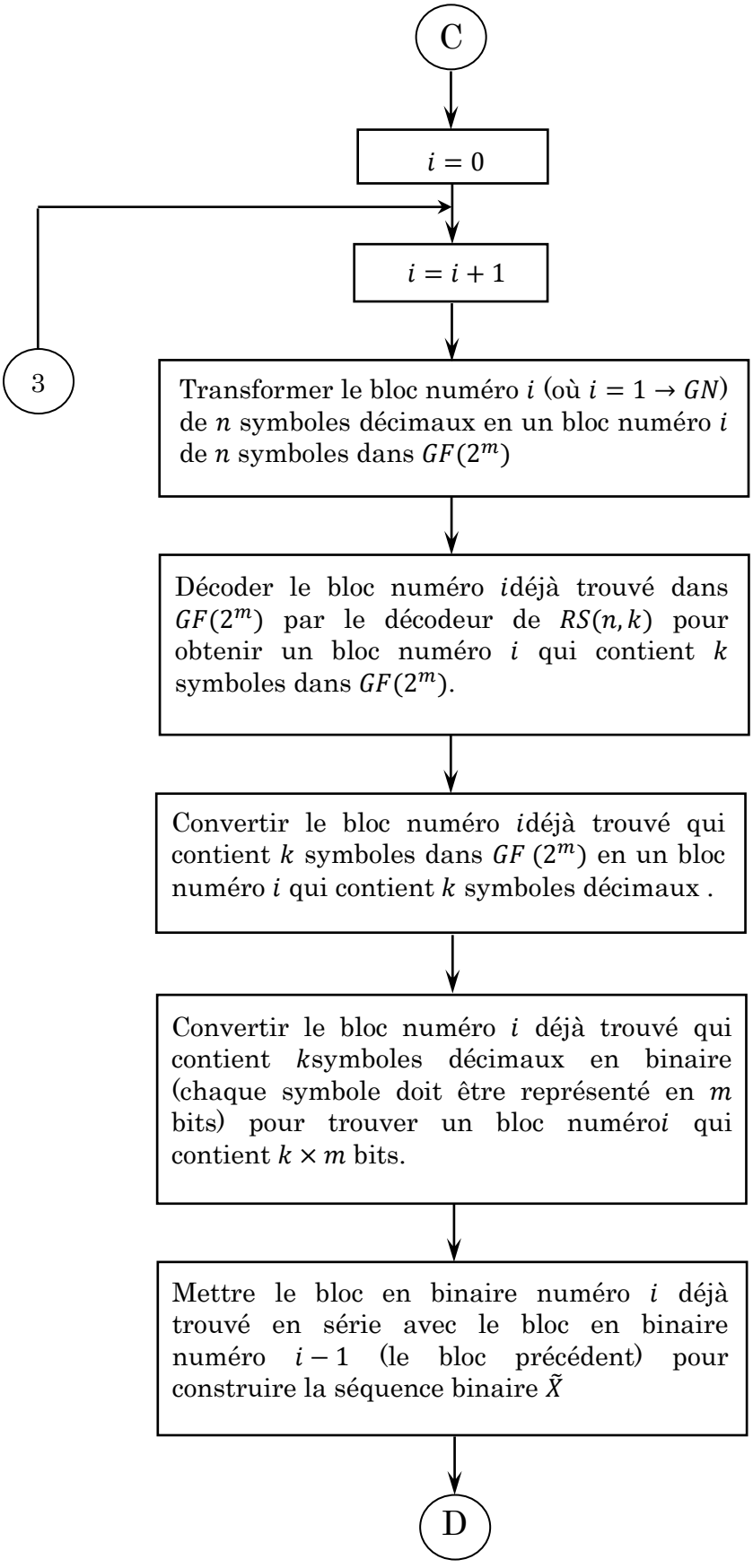
$$BER = \frac{nber}{lseq} \tag{3.7}$$

La simulation numérique de la chaine de transmission selon le modèle de simulation de la figure 3.1 est basée sur l'organigramme de la figure 3.2.


Début
Déclarer les variables GN, N_e, k, m (ou $n = 2^m - 1$).
Générer le polynôme générateur du code deRS qui dépond de n et k.
Générer une séquence binaire aléatoire X, de longueur $N_k = GN \times m \times k$ qui représente l'information à envoyer.
Subdiviser l'information binaire$X$de longueur N_k en $GN \times k$ blocs de m bits.
Convertir les $GN \times k$ blocs de m bits en décimal pour trouver $GN \times k$ nombres entiers.
Subdiviser les $GN \times k$ nombres entiers (symboles décimaux) en GN blocs où chaque blocs contient k symboles.
$i = 0$
$i = i + 1$
1
Transformer le bloc numéro i(où $i = 1 \rightarrow GN$)de k symboles décimaux en un bloc numéro i de k symboles dans GF (2^m) (dans le **Matlab** on utilise l'instruction gf) .
A


A
Coder le bloc numéro $i$déjà trouvé dans $GF\ (2^m)$ par le code de $RS\ (n,k)$ pour obtenir un bloc numéro i qui contient n symboles dans $GF(2^m)$.
Convertir le bloc numéro i déjà trouvé qui contient n symboles dans $GF\ (2^m)$ en un bloc numéro i qui contient n symboles décimaux.
Convertir le bloc numéro i déjà trouvé qui contient n symboles décimaux en binaire (chaque symboles doit être représenté en m bits) pour trouvé un bloc numéro i qui contient $n \times$ m bits.
Mettre le bloc en binaire numéro i déjà trouvé en série avec le bloc en binaire numéro $i-1$ (le bloc précédent) pour construire la séquence binaire X_c.
1
Non
$i > GN$
Oui
Moduler la séquence binaire X_c déjà obtenue (qui à une longueur $N_n = GN \times n \times m$ bits en $BPSK$ en utilisant la relation $Y = 1 - 2X_c$) .
Déclarer le SNR en dB
B


B
$\sigma = 0.707 \times 10^{-SNR/20}$
$S_{err} = 0$
$i_e = 0$
$i_e = i_e + 1$
2
Générer N_n échantillons d'un bruit gaussien noté W de variance égale σ^2.
Le signal reçu $R = Y + W$.
La séquence binaire estimée par le démodulateur hard est $\tilde{X}_c = \frac{1-signe(R)}{2}$
Subdiviser la séquence binaire estimée $\tilde{X}_c$ de longueur N_n en $GN \times n$ blocs de m bits .
Convertir les $GN \times n$ blocs de m bits en décimal pour trouver $GN \times n$ nombre entiers.
Subdiviser les $GN \times n$ nombre entiers (symboles décimaux) en GN blocs où chaque blocs contient n symboles .
C


C
$i = 0$
$i = i + 1$
3
Transformer le bloc numéro i (où $i = 1 \rightarrow GN$) de n symboles décimaux en un bloc numéro i de n symboles dans $GF(2^m)$
Décoder le bloc numéro $i$déjà trouvé dans $GF(2^m)$ par le décodeur de $RS(n,k)$ pour obtenir un bloc numéro i qui contient k symboles dans $GF(2^m)$.
Convertir le bloc numéro $i$déjà trouvé qui contient k symboles dans $GF\ (2^m)$ en un bloc numéro i qui contient k symboles décimaux .
Convertir le bloc numéro i déjà trouvé qui contient $k$symboles décimaux en binaire (chaque symbole doit être représenté en m bits) pour trouver un bloc numéro$i$ qui contient $k \times m$ bits.
Mettre le bloc en binaire numéro i déjà trouvé en série avec le bloc en binaire numéro $i-1$ (le bloc précédent) pour construire la séquence binaire $\tilde{X}$
D

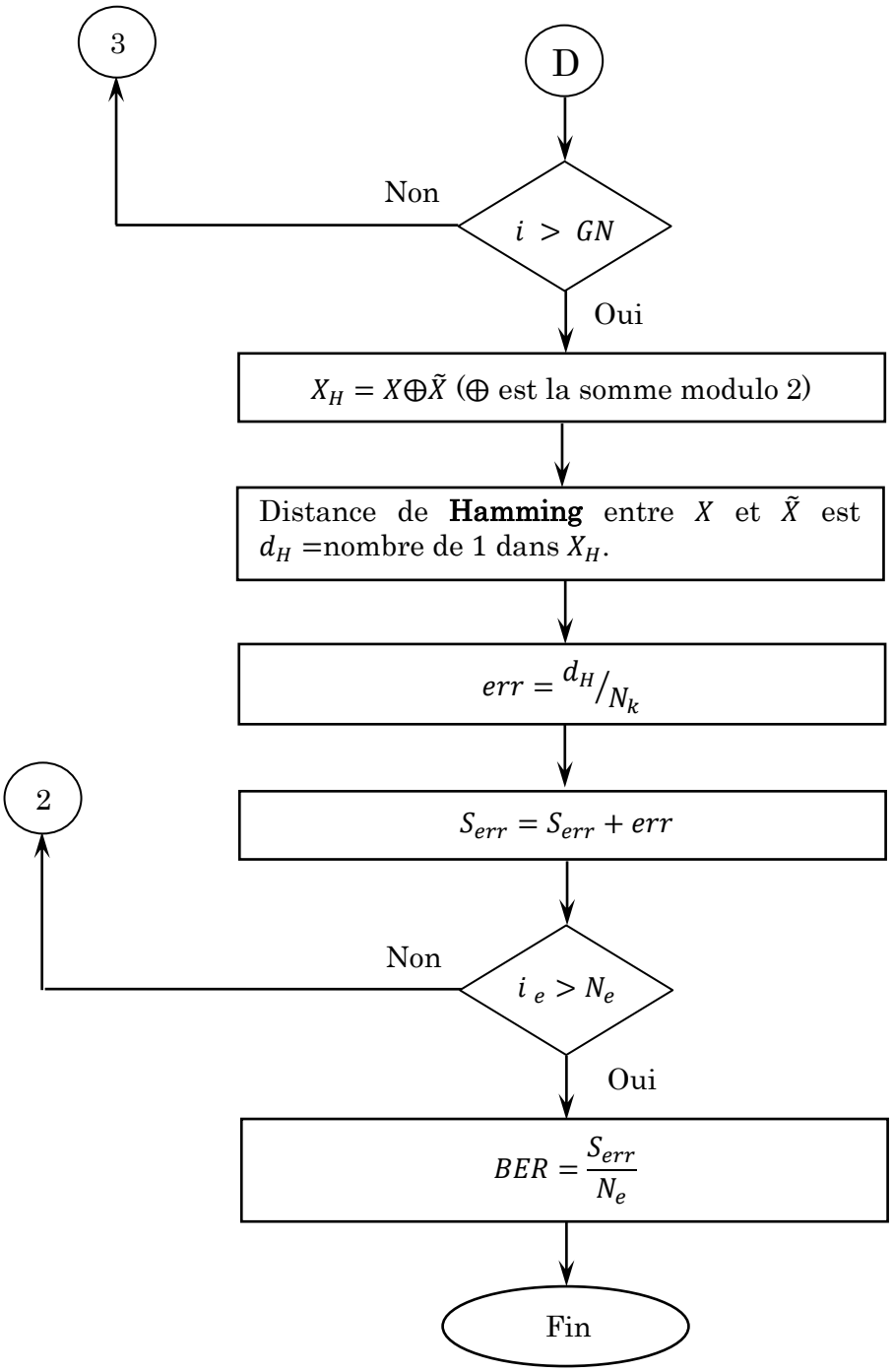


Fig. 3.2 L'organigramme de la simulation selon le modèle de figure 3.1.

On définit les notations utilisées dans cet organigramme.

N_e : nombre d'essais pour la méthode de Monte Carlo ;

k : nombre des symboles d'entrée du codeur $RS(n,k)$;

m : nombre des bits par symbole

n : nombre des symboles de sortie du codeur $RS(n,k)$;

X : la séquence binaire qui représente l'information ;

N_k : la taille de la séquence X (nombre des bits de X) ;

GN : un grand nombre choisit librement, 10 000 par exemple ;

X_c : la sortie du codeur $RS(n,k)$ écrit en binaire, qui sera injectée dans le modulateur BPSK ;

Y : la sortie du modulateur BPSK ($Y = 1 - 2X_c$) ;

N_n : la taille de X_c ;

SNR : le rapport signal sur bruit en dB ;

σ : l'écart type du bruit AWGN ;

$Serr$: la somme d'erreurs ou erreur cumulée dans chaque essais numéro i_e ;

i_e : compteur d'essais ;

W : le bruit AWGN de taille N_n ;

R : signal reçu qui égal à $Y + W$;

$\tilde{X}_c$: la séquence binaire estimée par le démodulateur hard qui est égale à $\tilde{X}_c = \frac{1-signe(R)}{2}$;

$\tilde{X}$: la séquence binaire estimée de taille N_k qui doit être égale à X s'il n'y a pas d'erreurs ;

X_H : la somme exclusive de X et $\tilde{X}$;

d_H : la distance de **Hamming** entre X et $\tilde{X}$;

err : l'erreur qui est égale à la distance d_H divisée par la taille de X ;

BER : la moyenne des erreurs calculées dans chaque essais.

3.3. Résultats et Commentaires

3.3.1. Programmation

Pour la mise en œuvre de notre simulation, nous avons opté pour la réalisation des programmes le **Matlab** pour les raisons suivantes

- Le **Matlab** contient des fonctions prêtes à utiliser, telles que la génération du bruit, la fonction d'erreur, fonction d'autocorrection, etc. Ces fonctions permettent une réduction des programmes.
- Le **Matlab** contient un graphisme qui permet une visualisation et un traçage simple des courbes.

Nous avons réalisé donc, à l'aide du **Matlab** des programmes de simulation pour les codes de **Reed Solomon** $RS(n,k)$ quelques soient les valeurs de n et k.

Nous signalons que la partie décodage par l'utilisation de l'algorithme de **Reed Solomon** était la partie difficile à mettre en œuvre.

3.3.2. Taux d'erreurs binaire BER

Le taux d'erreurs binaire BER mesure la performance d'un système de communication numérique. Notre simulation a permis de calculer ce paramètre pour un système de communication numérique avec code de RS pour différentes valeurs de bruit.

Les figures3.3 jusqu'à 3.7 montrent les BER pour deux systèmes, l'un est sans codage et l'autre avec codage de RS, en fonction du rapport signal à bruit SNR

- Pour le système BPSK sans codage son BERest bien approximée par la relation [45].

$$BER = P_s(e) \cong \frac{1}{2}\operatorname{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{3.8}$$

et représentée par la courbe supérieure dans toutes les figures ci-dessous.

- Pour le système BPSK codé avec les codes de RS, son BER est représenté par les courbes inférieures.

L'analyse des figures 3.3jusqu'à 3.7permet de faire les constatations et les remarques suivantes

- Un système avec codage RS est plus performant que celui sans codage. En effet, dans la figure 3.3 et pour un $BER = 10^{-2}$ on gagne 3 dB en termes du SNR si on utilise un code $RS(7,3)$ qui corrige au maximum 2 symboles sur 7 symboles ;
- Le BER est inversement proportionnelle au SNR, ce qui confirme davantage la théorie, puisque pour un SNR grand la séquence émise n'est affectée que peu par le bruit. Donc, le décodeur de vraisemblance maximale aura à son entrée une séquence avec un nombre faible de bits erronés, d'où une probabilité faible, et vice versa ;
- Pour atteindre un $BER = 10^{-5}$il faut : avec le code le $RS(15,3)$ un $SNR = 2.78\ dB$, avec le code $RS(15,5)$ un $SNR = 3.75\ dB$, avec le code $RS(7,3)$un $SNR = 4.85\ dB$ et avec le code $RS(31,27)$ un $SNR = 5.6\ dB$;
- On sait qu'un code de **Reed Solomon** $RS(n,k)$corrige au maximum $t = (n-k)/2$ symboles sur n symboles. Par conséquent, théoriquement plus le rapport t/n est grand plus le code est performant. En effet, la figure 3.7 confirme cette affirmation car le $RS(15,3)$qui est le plus performant possède un rapport $\frac{t}{n} = \frac{6}{15} = 0.4$, et vient après le code $RS(15,5)$ qui a un rapport $\frac{t}{n} = \frac{5}{15} = 0.33$, puis le code $RS(7,3)$ où son

rapport est $\frac{t}{n} = \frac{2}{7} = 0.28$ et enfin le code $RS(31,23)$ avec un rapport $\frac{t}{n} = \frac{4}{31} = 0.13$. Mais il est important de remarquer dans les quatre codes de RS que plus le code est performant plus son rendement est faible. En effet, le $RS(15,3)$ qui est le plus performant possède un rendement $\frac{k}{n} = \frac{3}{15} = 0.2$,et vient après le code $RS(15,5)$ qui a un rendement $\frac{k}{n} = \frac{5}{15} = 0.33$,puis le code $RS(7,3)$ où son rendement est $\frac{k}{n} = \frac{3}{7} = 0.43$ et enfin le code $RS(31,23)$ avec un rendement $\frac{k}{n} = \frac{23}{31} = 0.74$. Ce résultat est prévisible car le rapport $\frac{t}{n} = 0.5 - 0.5 \times rendement$, et plus le rendement est petit plus le rapport t/n est grand et vice versa.

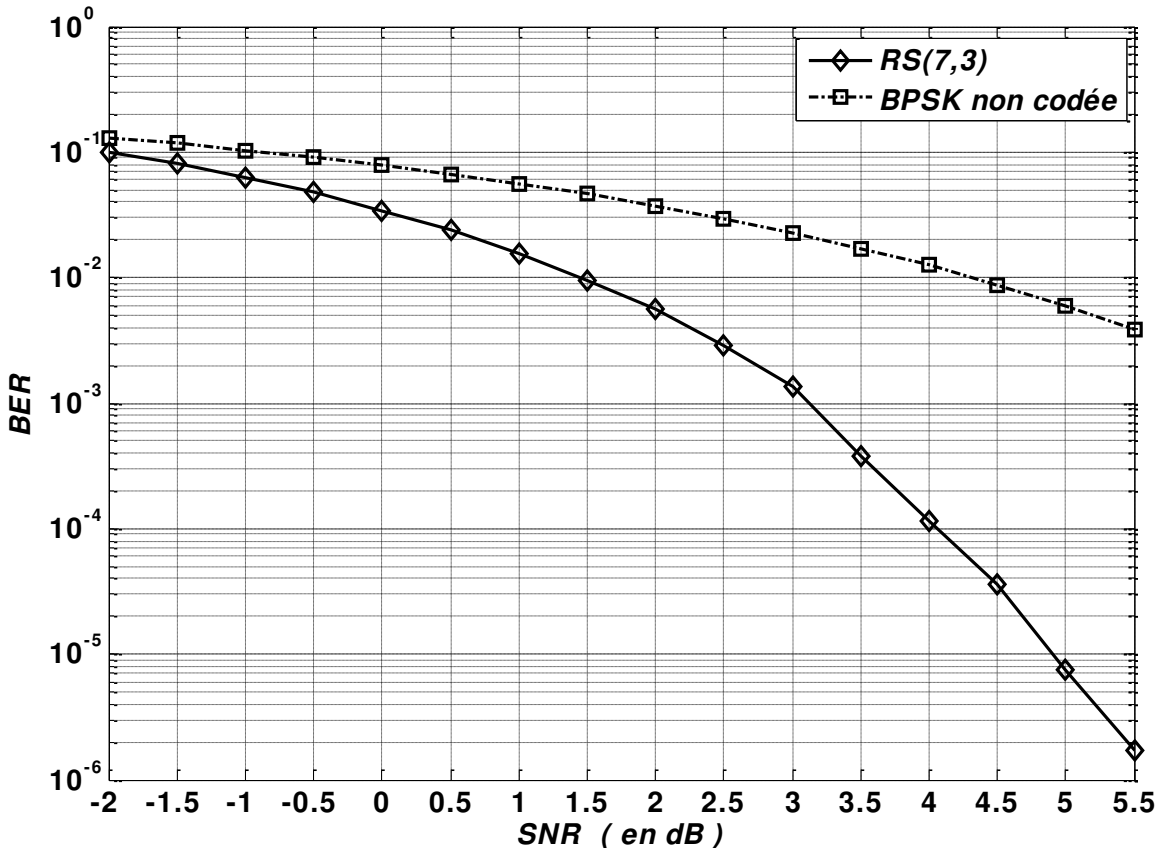


Fig.3.3 Le BER en fonction du SNR du code RS(7,3).

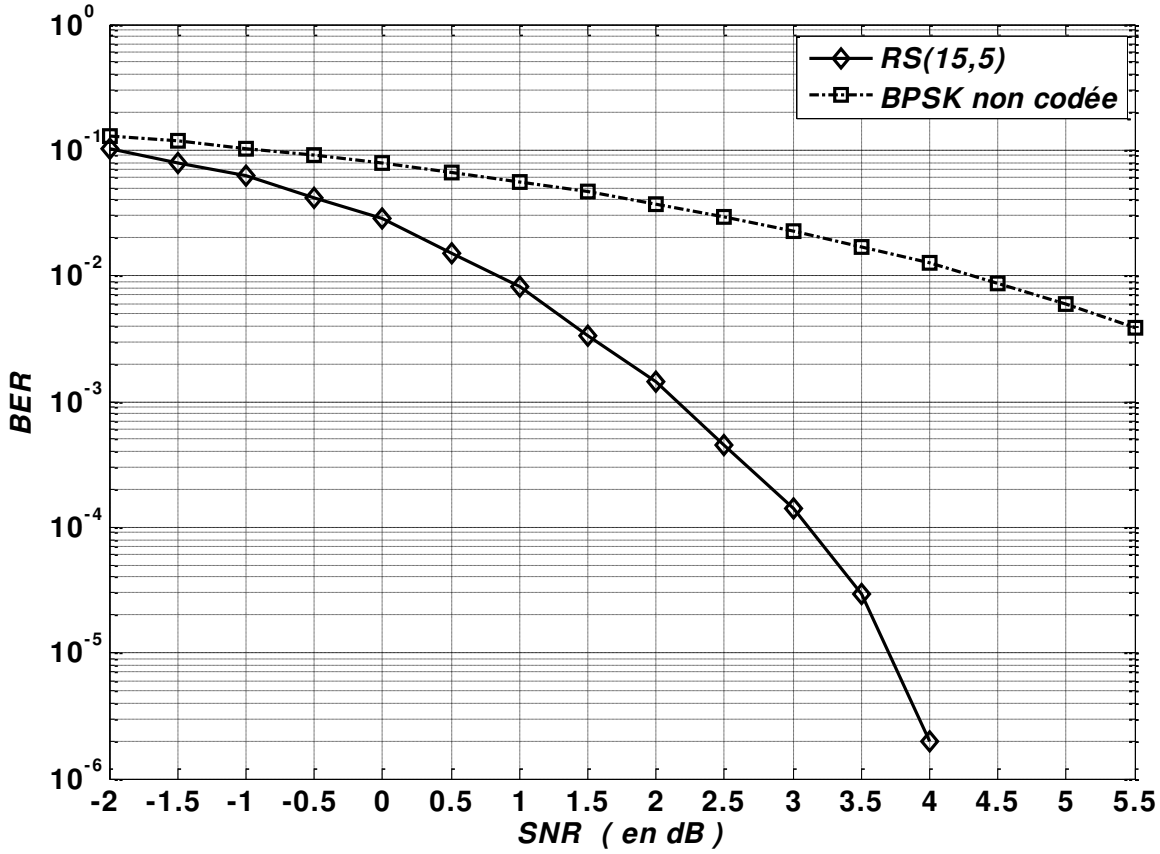


Fig.3.4 Le BER en fonction du SNR du code RS (15,5).

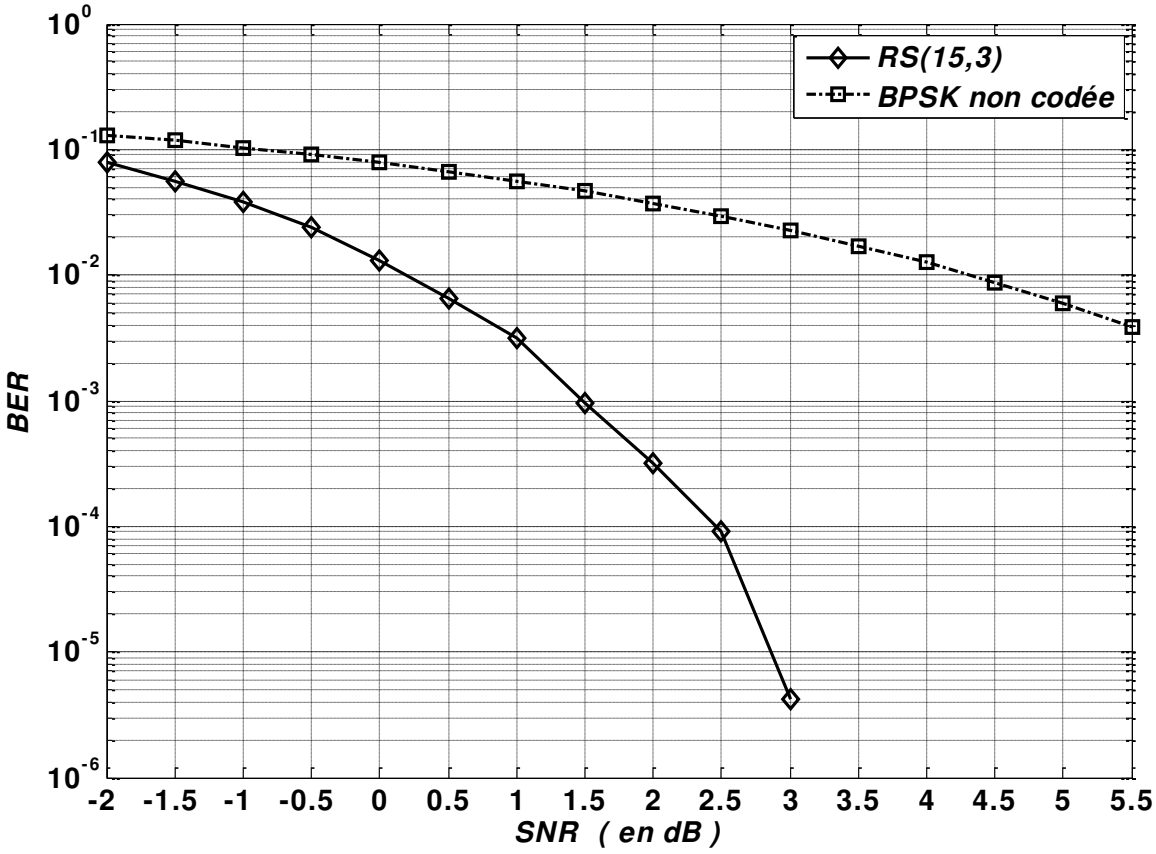


Fig.3.5 Le BER en fonction du SNR du code RS (15,3).

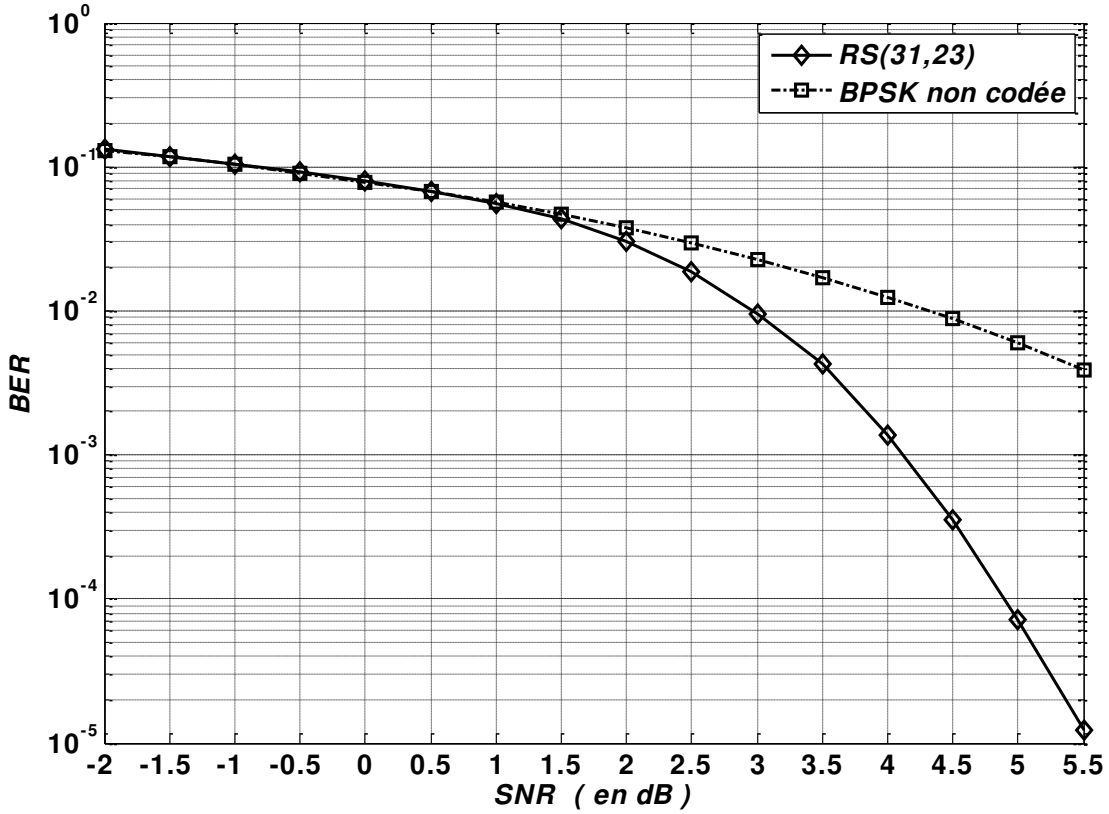


Fig.3.6 Le BER en fonction du SNR du code RS (31,23).

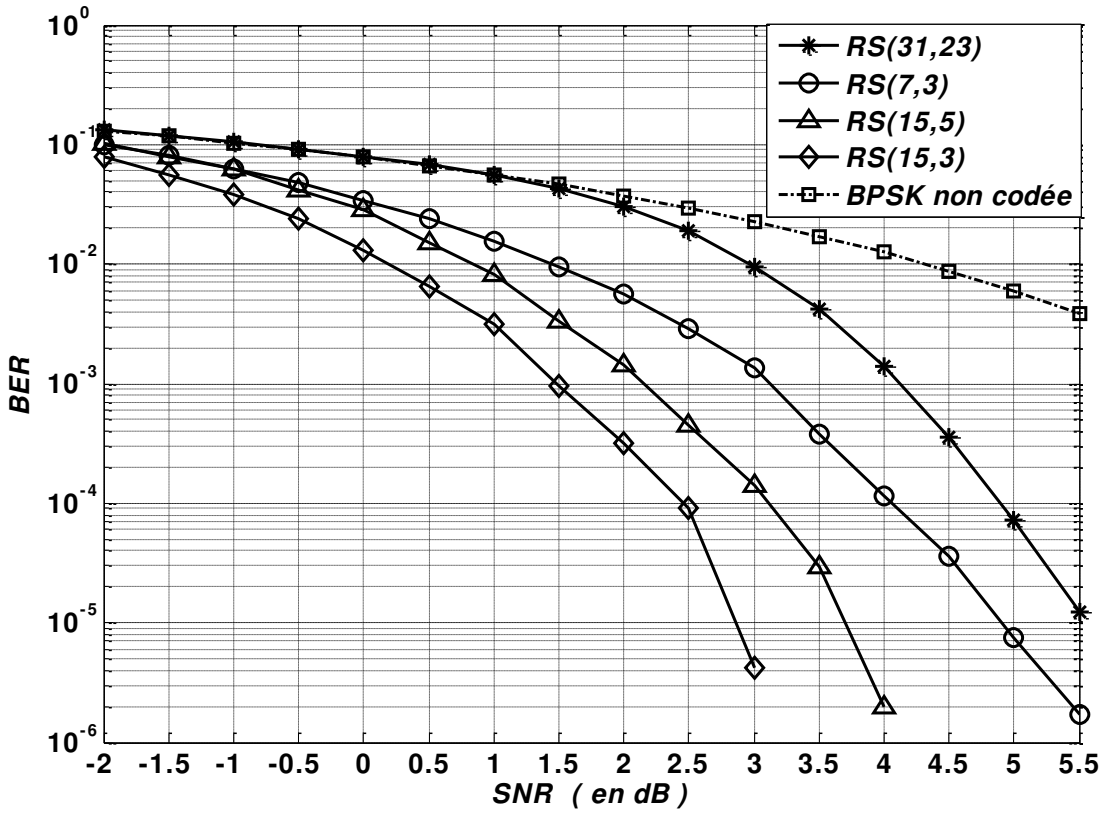


Fig.3.7 Le BER en fonction du SNR du code RS (7,3), RS (15,5), RS (15,3), RS (31,23).

3.4 Conclusion

Notre simulation a permis de calculer les performances des systèmes avec code RS. Les résultats obtenus sont en conformité avec la théorie, et montre que les codes RS sont parmi les meilleurs codes existants.

La simulation montre aussi qu'un code RS possède un grand pouvoir de correction des erreurs si son rendement est faible et vice versa. Mais dans la pratique on cherchait toujours de trouver un compromis entre le pouvoir de correction des erreurs et le rendement. Dans notre simulation le code $RS(15,5)$ et $RS(7,3)$ font des bons compromis.

Notre travail s'est limite bien à la modulation BPSK, les plus utilisées dans le domaine et au bruit AWGN. on peut bien entendu étudier les codes de **Reed Solomon** pour un canal non gaussien et avec modulation différente de la BPSK.

Conclusion générale et perspectives

Selon le cahier de charge exposé, notre travail concernant l'étude des codes de **Reed Solomon** RS et leur simulation, a englobé deux parties : une partie théorique et l'autre une simulation numérique.

La partie théorique a pour but de dégager les fondements théoriques des codes correcteurs d'erreurs en général, à savoir : les théorèmes fondamentaux des codes correcteurs, la classification des codes, les codes linéaires, la matrice génératrice d'un code linéaire, la matrice de contrôle d'un code linéaire, la distance minimale d'un code linéaire, le décodage d'un code linéaire, les codes parfaits et les codes cycliques. Une classe très importante des codes parfaits et cycliques a été présenté dans cette partie qui est la classe des codes de **Reed Solomon**. Dans cette classe on a présenté le problème principal du codage, une généralité sur les corps finis de **Galois**, la définition d'un code de **Reed solomon** et ses avantages, la technique de codage, les algorithmes de décodage. Il y a plusieurs algorithmes de décodage des codes de **Reed Solomon**, mais dans cette partie on a exposé que deux algorithmes qui sont : l'algorithme d'**Euclid** et l'algorithme de **Reed Solomon**. Les deux premiers chapitres ont été consacrés à cette partie.

La deuxième partie ne contient que le troisième chapitre seulement. Ce chapitre était le plus intéressant dans ce mémoire. Il présente une simulation numérique d'un système de communication avec un code RS en considérant un canal à bruit blanc gaussien additif et avec modulation BPSK. Dans cette simulation on a étudié quatre codes RS qui sont : $RS(15,3)$ qui est capable de corriger $t = 6$ erreurs, $RS(15,5)$ qu'il peut

corriger $t = 5$ erreurs, $RS(7,3)$ qui possède un nombre d'erreurs corrigibles $t = 2$ erreurs et $RS(31,23)$ où le nombre d'erreurs corrigibles est $t = 4$ erreurs.

On a montré par simulation qu'un système BPSK codé avec un code RS est plus performant que celui sans codage. Aussi, on a vu que plus le rapport t/n (le nombre maximum d'erreurs corrigibles sur le nombre des symboles de sortie du code) est grand plus le code possède un grand pouvoir de correction, mais ceci est au détriment du rendement du code, car le rendement devient petit dans ce cas comme montre la relation $\frac{t}{n} = 0.5 - 0.5 \times rendement$. Un meilleur code RS doit présenter un compromis entre le pouvoir de correction et le rendement.

A l'issue de ce mémoire, plusieurs perspectives peuvent être envisagées. Elles se résument en certains points restent à approfondir et d'autres à étudier, on peut en évoquer quelques-uns

- Etendre notre étude avec un code RS pour un canal avec modulation différente de la BPSK ;
- Etendre notre étude avec un code RS pour un canal avec bruit non gaussien ;
- Etendre notre étude avec un code $RS(n,k)$ pour un canal à bruit non gaussien et avec une modulation $(n+1)-$aire.

Bibliographie

[1] Samuele Dietler, Implémentation de codes de Reed-Solomon sur FPGA pour communications Spatial (Code correcteur d'erreurs), 2005.

[2] Jean. Baptiste Campesato, Les codes correcteurs d'erreurs, 2009.

[3] J.G. Dumas, Théorie des codes (Compression, cryptage, correction), 2007.

[4] A. Julien, V.janne et J.Pierre Cances**,** Codage de canal pour les communications optiques, 2009.

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

[7] P.V. Koseleff, Codes correcteurs d'erreur, 2004.

[8] Badrikian Josèphe, Mathématique pour téléinformatique, Codes correcteurs, principes et exemple.

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

[10] M^{lle}. Khadija SERIR, Mémoire de Master, thème « Application des codes correcteurs d'erreurs Reed Muller », Université Abou Bakr Belkaid–Tlemcen 2011.

[11] S.F Amouri, Mémoire de Licence académique, thème « Les codes de Reed-Solomon et le problème de leur décodage », 2014.

[12] Jessie Mac, Williams et Neil Sloane, The Theory of Error-Correcting Codes, North-Holland, 1977.

[13] A. Spătaru, Fondements de la théorie de la transmission de l'information, PPUR, 1987 .

[14] Pierre Abbrugiati, Introduction aux codes correcteurs d'erreurs,23 janvier 2006.

[15] Matthieu Finiasz, Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef Publique, 2004.

[16] Arnaud Labourel, codes détecteur correcteur , 2011.

[17] Dr. Hitta Amara, Information binaire Codes correcteurs et Compression, 5.07.2008.

[18] Bose. R.C, Ray. Chaudhuri, "On A Class of Error Correcting Binary Group Codes", control, March 1960.

[19] Hocquenghem, A, Codes correcteurs d'erreurs, Chiffres (in French Paris) Septembre 1959.

[20] Stephen B. Wicker, Vijay K. Bhargava, Reed-Solomon Codes and Their Applications, IEEE Press, New York 1999.

[21] Fatma Abdelkefi, Les Codes Reed-Solomon pour la correction des erreurs impulsives dans les systèmes multiporteuses, Thèse de doctorat en Traitement de signal et images, Soutenue en 2002 à Paris ENST.

[22] Les codes algébriques principaux et leur décodage, Daniel Augot, INRIA Saclay-Île de France et Ecole polytechnique, Journées nationales du calcul formel Luminy, Mai 2010.

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

[24] M. Reversat et Benoît Zhang, Cours de théorie des corps, université Paul Sabatier de Toulouse, 2003.

[25] Romain Basson, Preparation a l'agregation (Corps finis), 2014-2015.

[26] Marfc. Le large, Théorie de l'information et codage (Cours10-3 Mai), 2010-2011.

[27] Nicolas BARBOT, Codage de canal pour les communications optiques sans fil, Université DE LIMOGES, le 22 novembre 2013.

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

[29] Wicker .SB, Bhargava. VK, An Introduction to Reed-Solomon Codes.

[30] F. Arlotto et M Candau, Décodage des codes de Reed–Solomon, Master 1 Cryptologie et Sécurité Informatque, Université Bordeaux 1, 12 Avril 2010.

[31] Jen-Louis, Une nouvelle méthode de décodage des Reed-Solomon, (A new méthode for décodage Reed-Solomon codes), 1982.

[32] Samuele Dietler, Implémentation de codes de Reed-Solomon sur FPGA pour communications Spatial (Code correcteur d'erreurs), 2005.

[33] Pierre lescanne, L'algorithme d'Euclide, 18 Janvier 2006.

[34] Daniel Augot, Les codes algépriques principaux et leur décodage, INRIA Saclay-Île de France et Ecole polytechnique, Mai 2010.

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

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

[37] Oliver Pretzel. Clarendon Press, Oxford, Error-Correcting Codes and Finite Fields.

[38] S.Lin & D.Costello, Error control coding.

[39] B.Ferhat et A. Guemari, Les turbocodes convolutifs avec modulation BPSK pour un canal AWGN, Projet de fin d'étude, Université de Kasdi Merbah, département de mécanique et d'électronique, Ouaregla, Juin 2008.

[40] F.J. Mac Williams and N.J.A. Sloane, « Pseudo-Random Sequences and Arrays », Proceedings of the IEEE, Vol, IT-64, PP. 1715-1728, Decem. 1989.

[41] A.Spataru, Fondements de la théorie de la Transmission de l'information, Presse Polytechnique Romandes. Lausanne, Suisse, 1987.

[42] Z.Thabet, L.Dalal, Turbo décodage à bruit impulsif $\alpha-$stable symétrique,Université Hamma Lakhder d'El-Oued Institut des Sciences et Technologie Département D'électronique, 2015.

[43] CHEMSA Ali, Thèse "Modulation avec codage itératif pour un canal à bruits non-gaussiens", Avril 2016.

[44] J.C. Bic. Duponteil, J.C.Lmbeaux, Elements de Communications Numériques : Transmission sur Fréquence Porteuse, Tome II, Bordas et C.E.N.T.E.N.S.T, Paris, 1986.

[45] S. Haykin, Digital Communication, John Wiley & Sons, New York.1988.