N° d'ordre : 017/L3/TCO Année Universitaire : 2006 / 2007

UNIVERSITE D'ANTANANARIVO

ECOLE SUPERIEURE POLYTECHNIQUE

DEPARTEMENT TELECOMMUNICATION

MEMOIRE DE FIN D'ETUDES

en vue de l'obtention

du DIPLOME de

LICENCE ES SCIENCES et TECHNIQUES

en Télécommunication

par : **ANDRIANARISON RAKOTONIRAINY Toky**

CONCEPTION D'UN LOGICIEL DE CODAGE ET CORRECTEUR D'ERREURS : Code MC Donald

Soutenu le 20 Février 2008 devant la Commission d'Examen composée de :

Président :

Monsieur RANDRIAMITANTSOA Paul Auguste

Examinateurs :

Monsieur ANDRIAMIASY Zidora

Monsieur RANDRIARIJAONA Lucien Elino

Monsieur RATSIHOARANA Constant

Directeur de mémoire :

Monsieur RASAMOELINA Jacques Nirina

REMERCIEMENTS

Je rends grâce à Dieu pour sa bonté, de m'avoir donné la force et la santé durant la réalisation de ce mémoire.

Je tiens également à adresser mes vifs remerciements aux personnes suivantes sans qui ce travail de mémoire n'aurait pas pu être réalisé :

Monsieur RAMANANTSIZEHENA Pascal, Professeur, Directeur de l'Ecole Supérieure Polytechnique d'Antananarivo (ESPA), pour mes trois années d'études ;

Monsieur RANDRIAMITANTSOA Paul Auguste, Professeur, Chef du Département Télécommunication à l'ESPA, qui me fait l'honneur de présider le Jury de ce mémoire ;

Monsieur RASAMOELINA Jacques Nirina, Enseignant-chercheur au sein du Département Télécommunication à l'ESPA, Directeur de ce mémoire qui malgré ses lourdes responsabilités m'a toujours prodigué ses conseils. Je tiens à lui adresser toute ma gratitude ;

Monsieur, ANDRIAMIASY Zidora, Enseignant-chercheur au sein du Département Télécommunication à l'ESPA, membre du Jury ;

Monsieur, RANDRIARIJAONA Lucien Elino, Enseignant-chercheur au sein du Département Télécommunication à l'ESPA, membre du Jury ;

Monsieur, RATSIHOARANA Constant, Enseignant-chercheur au sein du Département Télécommunication à l'ESPA, membre du Jury ;

Mes vifs remerciements s'adressent également à tous les enseignants et personnels de l'Ecole Supérieure Polytechnique d'Antananarivo en général et ceux du Département Télécommunication en particulier, sans leurs efforts notre formation n'aurait pas pu atteindre ce stade.

Je n'oublierai pas ma famille pour leurs soutiens bienveillants et leurs encouragements, pour ce mémoire, comme en toutes circonstances.

Plus particulièrement, à mes parents pour leurs sacrifices durant ces longues années afin que je puisse arriver à ce niveau et pour tout ceux qui ont contribué de près ou de loin a l'élaboration de ce mémoire.

TABLE DES MATIERES

NOTATIONS ET SYMBOLES

a	: message à codé
d	: distance
d_{min}	: distance minimale
e	: capacité de correction
e_D	: erreur détectable
e’	: efficacité de code
k	: dimension du code
n	: longueur de code
s	: syndrome
t	: nombre d’erreur
x	: mot envoyé
y	: mot reçu
C	: code
G	: matrice génératrice
GF (q)	: corps fini
H	: matrice de contrôle
K	: corps
N	: cardinal du mot de code
R	: taux de transmission
ε	: Vecteur erreur
$\mathcal{H}$	: Matrice de contrôle d’un code de HAMMING étendu
η	: rendement de code
τ	: taux d’erreurs brute

INTRODUCTION

La théorie des communications s'intéresse aux moyens de transmettre une information depuis une source jusqu'à un destinataire. La nature de la source peut être très variée. Il peut s'agir par exemple d'une voix, d'un signal électromagnétique ou d'une séquence de symboles binaires. Le canal peut être une ligne téléphonique, une liaison radio ou encore un support magnétique ou optique : bande magnétique ou disque compact. Le canal sera généralement perturbé par un bruit qui dépendra de l'environnement et de la nature canal : perturbations électriques, rayures, … . Le codeur représente l'ensemble des opérations effectuées sur la sortie de la source avant la transmission. Ces opérations peuvent être, par exemple, la modulation, la compression ou encore l'ajout d'une redondance pour combattre les effets du bruit. Elles ont pour but de rendre la sortie de la source compatible avec le canal. Enfin, le décodeur devra être capable, à partir de la sortie du canal de restituer de façon acceptable l'information fournie par la source.

La technique de codage est utilisée depuis longtemps. De nos jours, le codage est une technique employée pour la communication, à coté d'autres techniques, matérielles ou logicielles, comme les protocoles de communications ou algorithme distribué. Il évoque toujours un procédé de transformation d'un objet, avec le procédé inverse, appelé le décodage qui permet de restituer l'objet initial.

Le codage s'applique à une suite de symbole émis par une source d'information. Cette suite peut être une suite discrète d'élément pris dans un ensemble fini ou bien il peut s'agit d'une fonction continue caractérisée par son amplitude en fonction du temps. Une source continue peut être transformée en source discrète par des techniques d'échantillonnage. Le codage consiste en la transformation de la suite émise de la source en une nouvelle suite qui est alors envoyée dans un canal de transmission.

Les codages que nous étudierons consistent toujours à la transformation d'une suite de symboles prise dans un ensemble fini. Cet ensemble est appelé alphabet. Souvent l'alphabet sera l'ensemble {0, 1}, c'est un alphabet binaire ou code binaire qui utilise deux symboles.

Dans ce travail, le code qui nous intéresse c'est le code linéaire et en particulier le code de MC Donald.

Dans les années 40, C. E. Shannon a développé une théorie mathématique appelée théorie de l'information qui décrit les aspects les plus fondamentaux des systèmes de communication. Cette théorie s'intéresse à la construction et à l'étude de modèles mathématiques à l'aide essentiellement de la théorie des probabilités. La théorie de l'information s'est faite de plus en plus précise et est devenue aujourd'hui incontournable dans la conception de tout système de communication.

L'évolution de l'informatique a permis la conception de nombreux logiciel. Le logiciel MATLAB de Mathworks Inc est spécialement conçu pour les scientifiques. MATLAB s'impose dans le monde universitaire comme un outil puissant de simulation.

Chapitre 1 GENERALITE

La transmission des informations à travers un système de communication risque d'introduire des erreurs sur les suites de symboles qu'on transmet du fait du bruit. Alors le message 'Y', à la sortie du canal de transmission, peut être différent au message émis de l'entrée. Dans notre cas, alphabet à deux symboles, l'erreur peut se traduire par des changements de '0' en '1', ou de '1' en '0'.

Le problème de la transmission est la suivante : un expéditeur *A* envoie un message *m* à *B*; durant la transmission de ce message, des erreurs se produisent éventuellement, et *B* reçoit un message *m'* qui comporte peut-être des erreurs. Il s'agit de trouver comment faire pour que *B*, d'une part détecte l'existence d'erreurs, d'autre part, si elles ne sont pas trop nombreuses, sache les corriger.

Pour lutter contre le bruit du au canal, on utilise des blocs de n symboles pour transmettre k symboles d'information. Un tel bloc de longueur n est appelé un mot de code, avec $k \leq n$.

1.1 Codage [1] [2]

Considérons un ensemble fini S de message de k symboles d'information à transmettre, les symboles prenant leur valeur dans un alphabet K. dans le cas de codage binaire, utilisons un alphabet à deux symboles que nous pouvons noter :{0,1}=K. supposons que le nombre d'éléments de S est une puissance de entière de 2 (2^k) et S est identifié à ensemble K^k. Cette identification correspond à un codage sans redondance conduit à l'impossibilité de détecter la moindre erreur de transmission dans le cas de perturbation.

Pour pouvoir détecter les erreurs, il faut plonger S dans un ensemble plus grande de telle manière que les erreurs les plus courantes fassent passer d'une suite correspondant à un message à une autre suite qui n'est pas un message. Il faut utiliser des suite de n symboles de longueur supérieure à k, c'est-à-dire ajouter n-k symboles supplémentaires dits : redondance

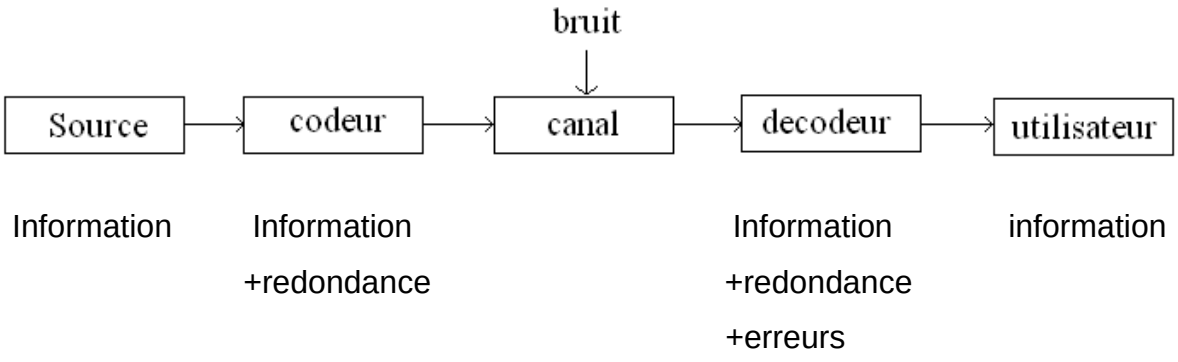

Figure 1.01 : Schéma d'un système de communication

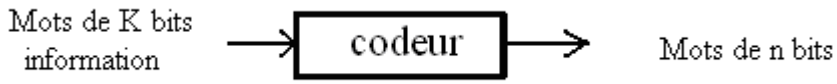

Figure1.02 : Schéma bloc du codeur

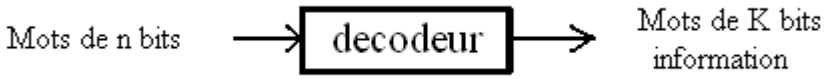

Figure 1.03 : Schéma bloc du décodeur

Pour détecter et corriger les erreurs pendant la transmission, il faut donc ajouter des symboles (symbole de contrôle) aux symboles correspondant à l'information que l'on veut transmettre. Cet ajout se fait selon une règle de codage connu par l'émetteur et par le récepteur. Ainsi, le décodeur vérifiera si la suite qu'il vient de recevoir satisfait ou non, à la règle C choisie. Si elle ne la satisfait pas alors il est sûr qu'au moins une erreur est advenue.

Nous pouvons choisir un code en fonction des paramètres suivants :

- la distance minimale
- la capacité de correction d'erreur
- le temps d'attente pour avoir le premier mot décodé

Le codage d'un canal aussi appelé codage détecteur et/ou correcteur est une fonction spécifique des transmissions numériques, qui n'a pas son équivalent en transmission analogique. Cette opération permet d'améliorer la qualité la transmission.

1.2 Quelques définitions et notions de bases [2] [3] [4]

1.2.1 L'efficacité d'un code C

L'efficacité *e* d'un code est la probabilité de reconnaître comme faux un message faux. C'est une probabilité conditionnelle.

$$e' = P\ (F/F) \tag{1.01}$$

Ou

$$e' = \frac{\text{nombres d'erreurs répétées}}{\text{nombres d'erreurs totales}} \tag{1.02}$$

1.2.2 Taux de transmission R d'un code

Le taux de transmission R d'un code C est définie par :

$$R = \frac{\log_2 N}{n} \tag{1.03}$$

1.2.3 Performances

Un algorithme de décodage d'un code C est dit borne par t si pour tout x

$$2\ C\_d_H\ (x; y) \leq t \Rightarrow \varphi(y) = x \tag{1.04}$$

Si la réciproque est vraie, l'algorithme est dit strictement borné. Tout code de distance minimale d possède un algorithme de décodage borné par b (d -1) = 2c. C'est aussi la meilleure valeur possible.

Proposition La probabilité de transmission correcte d'un mot à travers un canal de probabilité d'erreur p décodé à l'aide d'un algorithme strictement borne par t vaut

1.2.4 Taux d'erreurs brutes

Le taux d'erreur brute est la probabilité pour qu'un message reçu soit faux. Ce taux caractérise le canal de transmission seul.

$$\tau = 1 - (1-p)\ n \cong n\ .\ p \quad \text{Pour } p<<1,\ 1.\ p<<1 \tag{1.05}$$

1.2.5 Rendement d'un code

Le rendement d'un code η est le rapport quantité d'informations effectives / quantité d'informations maximales susceptibles d'être véhiculé par le même nombre de bit.

Pour la procédure de détection et de répétition, ce rendement est une moyenne.
η = k/n (1-ד) + (k/2n) ד e' (1-ד) + (k/3n)(ד e')²(1-ד)+ …

$$\eta = k/n\ (1-ד)\ \text{ si } ד << 1 \qquad (1.06)$$

k ≡ nombres d'informations.

1.3 Distance de HAMMING [3] [4] [5]

1.3.1 Définition

Les messages transmis sont supposés découpés en blocs (*ou mots*) de longueur *n* écrits avec l'alphabet *{0, 1}*. Un code est un sous-ensemble *C* de l'ensemble $\{0, 1\}^n$ de tous les mots possibles. On dit que *n* est la *longueur* de *C*.
La *distance de Hamming* entre deux mots $x = (x1, \ldots, xn)$ et $y = (y1, \ldots, yn)$, que l'on notera $d(x,y)$, est le nombre d'indices *i* tels que $xi \neq yi$. C'est bien une distance sur $\{0, 1\}^n$. La *distance minimum* du code *C* est le minimum des $d(x, y)$ pour *x* et *y* des mots différents de *C* (on suppose que *C* a au moins 2 mots). On la notera toujours *d*.

1.3.2 Exemple

a =11001000 et b = 10011010 ont une distance de 3.
➔d (a, b) = 3.

1.3.3 La distance minimale d'un code

La distance minimale du code est le minimum de l'ensemble des distances entre codes :

$$dmin = \min \{dij\} \qquad (1.07)$$

1.4 Condition de décodage d'ordre e [3], [9]

Un code C de longueur *n* sur C vérifie la condition de décodage d'ordre *e* si:
$\forall$ x' $\in$ A^n, il y a au plus un mot x $\in$ tel que d (x, x') $\leq$ *e* ou bien x' $\in$ B(x, *e*).

1.4.1 Définition de la capacité de correction d'erreur

Soit C un code, sa capacité de correction est le nombre maximal d'erreur qu'il peut corriger à la réception pendant la transmission.

1.4.2 Erreurs détectables

Le nombre minimal de bits changeant entre chaque code est d_{min} la distance minimale du code. e_D, le nombre maximal d'erreurs détectables est donc :

$$e_D = d_{min} - 1 \tag{1.08}$$

1.4.3 Théorème

Soit D la distance minimale d'un code C, si d $\geq$ 2*e* + 1, C vérifie la condition de décodage d'ordre *e*, c'est à dire que le code C peut corriger *e* erreur à la réception.

Chapitre 2 CODES LINEAIRES

2.1 Définitions [2] [5] [6]

➢ Lorsque l'alphabet est un corps fini (par exemple A = F_2 = {0; 1} l'espace de Hamming A^n est un espace vectoriel.

Un code en bloc linéaire de longueur n sur Fq (le corps fini à q élément) est un sous-espace vectoriel de $F^n q$.

Nous parlerons de code [n; k] q si le code est de dimension k et de code [n; k; d]q si sa distance minimale est d. Un code C [n; k] q peut se caractériser par :

- sa matrice génératrice G (de taille k _ n sur F_q) :

  C = {(u1, ….. uk) G | (u1,…… uk) ∈ F_q^n }

Les lignes de G forment une base de C.

- ou sa matrice de contrôle de parité H (de taille (n - k) * n sur F_q) :

  C = {(x1,……, xn) ∈ F_2^n , | H(x1,….., xn)^T = 0 }

Les lignes de H forment une base de l'orthogonal de C.

➢ On note F2 le corps à deux éléments 0 et 1. Les mots de longueurs *n* sont les éléments de F_2^n , que l'on écrira comme des vecteurs lignes. Un *code linéaire de longueur n* est un sous-espace vectoriel $C \subset F_2^n$. La lettre *k* désignera toujours la *dimension* de *C* (comme espace vectoriel). Le nombre de mots du code *C* est 2^k .

2.2 Propriété de base [3] [6] [9]

2.2.1 Propriété de la distance de HAMMING

C'est la distance entre deux suites de longueur n qui est égale au nombre de position où elles sont différentes.

Soit x = (x_1, x_2 ,…, x_n)

y = (y_1, y_2 ,…, y_n)

x-y = (x_1-y_1, x_2 -y_2 ,…, x_n-y_n)

D'où d(x, y) = nombre de composante non nulle de x-y.

$$d_H(x,y) = Card\{i : x_i \neq y_i, 1 \leq i \leq n\} \quad (2.01)$$

La distance de Hamming est l'outil de base de la correction des erreurs.

2.2.2 Poids d'un vecteur

Le poids d'un mot de code est par définition le nombre de caractères non nuls que contient ce mot.

2.2.2.1 Exemples

- X = 11001000 est de poids 3 ; notation :(w (X) = 3)
- Y = 10011010 est de poids 4 ; notation :(w (Y) = 4)
- Z = 00000000 est de poids nul ; notation :(w (Z) = 0)

2.2.2.2 Remarque

A l'ensemble des mots d'un code on associe l'ensemble des poids qui est la distribution de poids du code. Un code est dit de poids fixe (ou poids constant) si tous les mots du code ont le même poids. Si ω désigne le poids de Hamming pour un code linéaire C, alors la distance de Hamming d est définie par la formule suivante:

$$\forall x, y \in C \quad d(x, y) = w(x - y) \tag{2.02}$$

2.2.2.3 Propriété

- La distance de HAMMING de deux vecteurs, d (u, v), est égale au poids de la différence des deux vecteurs, w (u-v) ;
- w (u) = 0 ⇔ u = 0;
- $\lambda \in$ A, w (λ u) = λ w (u);
- w (u) = d (u, 0)
- w (u1+z2) ≤ w (u1) + w (u2) ;

2.2.3 Propriété de la distance minimale pour un code linéaire

Si C est un code linéaire, l'ensemble distance entre mots de code est l'ensemble des poids des mots non nuls du code.

Par conséquent : la distance minimale de ce code linéaire est égale à :

$$\text{Int } w(x), \ \forall \ x \in C, x \neq 0 \tag{2.03}$$

2.2.4 Condition de décodage

$$\text{Int } w(x) \geq 2e + 1,\ x \in C,\ x \neq 0 \tag{2.04}$$

2.2.5 Capacité de correction d'erreur

$$e = (\text{int } [w(x)] - 1) / 2 = (\text{int } [d_{min} - 1]) / 2 \tag{2.05}$$

2.3 Matrice génératrice et matrice de contrôle [5] [6] [8]

2.3.1 Matrice génératrice

2.3.1.1 Définition

Soit *C* un code linéaire. Une *matrice génératrice* de *C* est une matrice dont les lignes forment une base de *C*. Une matrice génératrice *G* est donc de taille $k \times n$ et de rang *k*. Si *m* est un vecteur ligne de F_2^k, le produit *mG* est un mot du code *C* et l'application $m \rightarrow mG$ est un isomorphisme de F_2^k sur *C* (que l'on peut voir comme une opération de codage). Si la matrice *G* est de la forme (I_k , P), on dit que le codage est *systématique*. Les *k* premiers bits d'un mot de code portent l'information (on y recopie le vecteur de F_2^k), les $n - k$ suivants sont de la redondance.

2.3.1.2 Remarques

- Un code C possède plusieurs matrices génératrices.
- Toutes les combinaisons linéaires des lignes d'une matrice génératrice sont les mots de codes C.
- On dit qu'une matrice génératrice d'un code (n, k) est normalisée ou canonique si la matrice forme par les k premières colonnes est la une matrice unité. Le code correspondant est dit *systématique*.

2.3.1.3 Exemple

Soit une matrice génératrice G = $\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$

G est de rang 3 car il y a une matrice d'ordre 3 tel que son déterminant est différent de 0.

$$\text{En effet} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 1 \neq 0$$

⇨ c'est une matrice 3 * 5 de rang 3 sur F_2

Alors G est une matrice génératrice d'un code linéaire C (5,3) dont les mots sont les combinaisons linéaires des lignes de G.

On a alors :

A (information)	C = {v = a .G} (Mot codé)	w(v) (Poids)
0 0 0	0 0 0 0 0	0
0 0 1	0 0 1 0 1	2
0 1 0	0 1 0 1 1	3
0 1 1	0 1 1 1 0	4
1 0 0	1 0 0 1 0	2
1 0 1	1 0 1 1 1	4
1 1 0	1 1 0 0 1	3
1 1 1	1 1 1 0 0	3

Tableau 2.01: tableau de déchiffrement complet

2.3.2 Matrice de contrôle

2.3.2.1 Définition

On peut aussi se donner un sous-espace vectoriel par un système d'équations indépendantes. Soit *C* un code linéaire. Une *matrice de contrôle* de *C* est la matrice d'un système d'équations linéaires homogènes indépendantes dont l'espace des solutions est *C*. Autrement dit, une matrice de contrôle *H* est de taille $(n-k) \times n$ et de rang $n-k$, et $C = \{x \in F_2^n \ ; \ H^t x = 0\}$.

La théorie des codes correcteurs affirme l'existence d'une application linéaire surjective de F dans un espace de dimension n - k dont la matrice H est appelée matrice de contrôle et dont le noyau est le code. En conséquence, on dispose des égalités suivantes :

$$H^t x = H^t(c+e) = H^t c + H^t e = 0 + H^t e = H^t e \qquad (2.06)$$

2.3.2.2 Remarques

Si G est une matrice génératrice de C et H est une matrice de contrôle de C alors
$H^t . G = 0$ ou $G^t . H = 0$

2.3.2.3 Exemple de recherche de matrice de contrôle

G est normalisé ⇔ $G = (I_k, M)$

Nous avons H par la formule suivante :

$$\mathbf{H} = {}^t\begin{pmatrix} M \\ I_{n-k} \end{pmatrix} \qquad (2.07)$$

Soit $G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$

G est la matrice génératrice d'un code linéaire C (5, 3) de rang = 3 car il y a une matrice d'ordre 3 tel que son déterminant est différent de 0.

En effet $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 1 \neq 0$ $\qquad I_k = I_3$

et $\qquad M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$ $\qquad$ et $\qquad \begin{pmatrix} M \\ I_{n-k} = I_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$

D'où

$$H = {}^t\begin{pmatrix} M \\ I_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

2.3.3 Code de HAMMING

2.3.3.1 Définition :

Les codes de Hamming sont des codes binaires qui ont la propriété :
(n, k) = (2m-1, 2m-1-m) avec m entier.
Exemple : - si m = 2 code (3, 1) peu d'intérêt, k = 1 on ne transmet qu'un bit de message.
- si m = 3 code (7, 4)
- si m = 4 code (15, 11)

Soit H la matrice qui a m lignes et 2^m-1 colonnes constituées de tous les m-uples (0, 0, …, 0) ; on appelle code de HAMMING de longueur 2^m-1, tout code admettant comme matrice de contrôle H, une matrice dont les 2^m-1 colonnes sont tous les vecteurs de $F_2 \setminus \{0\}$.

2.3.3.2 Propriétés

Un code HAMMING a pour :
- longueur 2^m-1,
- dimension 2^m-1 – m,
- capacité de correction e = 1

2.3.3.3 Exemple :

Le code de Hamming de paramètre m = 3 est un code linéaire de longueur 7 et de dimension k = 4.
Sa matrice de contrôle est :

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

- longueur 2^m-1 = 7
- dimension 2^m-1 – m = 4
- capacité de correction e = 1.

2.3.4 Code de HAMMING étendu

2.3.4.1 Définition

Le Code de Hamming étendu est un code linéaire de longueur $n = 2^m$ et de dimension $k = 2^m - m$; la matrice de contrôle d' un code de Hamming étendu est $\tilde{H}$ de paramètre m est donnée par :

$$\tilde{H} = \begin{vmatrix} & & 0 \\ & H & 0 \\ & & 0 \\ 1\,1\,1\,1\,1\,1\,1 & & 1 \end{vmatrix}$$

Avec H est la matrice de contrôle de code de HAMMING.

3.4.2. Exemple

Pour m = 3, la matrice de contrôle du code de Hamming étendu est :

$$\tilde{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

2.3.5 Décodage d'un code linéaire

2.3.5.1 Définition

On appelle décodage tout moyen qui permet de corriger toutes les erreurs qui effectuent les mots du code C, c'est-à-dire retrouver le mot x émis du code à partir du mot reçu y.
Il existe 3 méthodes de décodage. La linéarité simplifie grandement le décodage.

Le nombre t d'erreurs maximales assurément corrigibles est le plus grand entier strictement inférieur à 3/2 c'est-à-dire t = 1.

2.3.5.2 Décodage de voisin le plus proche

Elle consiste à décoder le mot reçu y comme le mot de code x le plus proche dans la distance de Hamming, c'est-à-dire rechercher le vecteur d'erreur e qui est le plus faible poids. Ainsi nous pouvons comparer y avec tous les q^k mots de code, il faut calculer toutes les distances d'un mot donné à chacun des mots de code. Lorsque la taille des paramètres augmente, cette méthode devient impraticable, alors ceci n'est pas valable que pour les petits codes (k petit).

2.3.5.3 Décodage par syndrome utilisant le tableau de décodage

Coder chaque vecteur reçu par le mot de code qui est dans le même colonne du tableau, en utilisant la matrice de contrôle.

2.3.5.3.1 Classe latérales

D'après une propriété immédiate du noyau d'une application linéaire, deux éléments de K^n ont la même image par H si et seulement si leur différence est dans le noyau de H. Ceci détermine une relation d'équivalence, donc une partition de K^n où chaque classe d'équivalence est formée par les vecteurs de K^n qui ont une même image par h.
Chacune de ces classes, les classes latérales (modulo C) sont translatés de C, soit u + C avec u Є K^n.

- Il y en a 2^{n-k} et chacune d'elle contient 2^n vecteurs.
- Tout éléments de K^n appartient à une classe est une seule.

2.3.5.3.2 Syndrome

- Définition

Le syndrome d'un élément x de K^n (par rapport à H) est le vecteur h(x) de K^{n-k}.
Si x= (x1, x2, …, xn) est considéré comme une matrice ligne, et si x^t est la matrice transposée alors le syndrome de x est H. x^t.

- Méthode de calcul de syndrome

Soit H la matrice de contrôle de code C (n, k) sur k et h l'application linéaire de K^n dans K^{n-k} dont H est la matrice par rapport aux bases naturelles de ces espaces.

x Є C si et seulement si h(x) = 0, c'est-à-dire que C est le noyau de h.

Si x est le mot envoyé et y le mot reçu, on appelle le mot erreur le mot différence de y avec x c'est-à-dire y = x + ε.
H(y) = h(x) + h (ε), et puisque x est un mot de code on obtient h(y) = h (ε).
Le mot reçu et le mot erreur ont la même image par h.

2.3.5.3.3 Principe de décodage

Nous devons savoir que le mot envoyé x et le mot erreur ε ont la même syndrome. Alors ils sont dans une même classe.

Si l'on connaît, pour chaque classe latérale, le syndrome correspondant, on calcul le syndrome de *y*, et ε est dans la classe associée.

Les opérations sont les suivantes :

- Calcul de syndrome du mot reçu *y*
- Détermination la classe latérale associée
- Recherche dans cette classe du mot erreur ε
- Calcul de $x = y - \varepsilon$

2.3.5.3.4 Construction du tableau de décodage

Les opérations pratiques utilisent le tableau suivant :

- on écrit sur une ligne tous les mots de code C en commentant à gauche par le mot nul : 0, v1, v2,……
- on écrit, en dessous, du mot non nul, un mot non écrit sur la ligne précédente, soit u1puis les autres mots de la classe de u1, soit u1, u1 + v1, u1 + v2, u1 + v3,……
- on recommence avec un mot u2 non écrit dans les ligne précédentes et ainsi de suite.
- Enfin, on écrit à droite de chaque ligne, le syndrome associé à la classe correspondante (h (0)) = 0 pour la première ligne, h (u1) pour la deuxième ligne, h(u2) pour la troisième ligne et ainsi de suite.

C								h (U)
	0	V1	V2	……	Vj	……	Vn	h (0)
U1 + C	U1	U1 + V1	U1 + V2	…..	U1 + Vj	…..	U1 + V n	h(U1)
….	…..	…..	….	…..	…..	….	……	…..
Ui + C	Ui	Ui + V1	Ui +V2	…..	Ui + Vj	…..	Ui + Vn	h(Ui)
…..	…..	…..	….	……	……	……	……	…..
Um + C	Um	Um + V2	Um + V3	…..	Um + Vj	….	Um + Vn	h(Um)

Tableau 2.02: tableau de déchiffrement complet

2.3.5.3.5 Application

Soit une application linéaire dont les matrices génératrices et de contrôle sont données ci-dessous :

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad \text{et} \quad H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

	000000	001011	010111	011100	100101	101110	100011	111001	000
U1+C	100000	101011	110111	111100	000101	001110	000011	011001	101
U2+C	010000	011011	000111	001100	110101	111110	110011	101001	111
U3+C	001000	000011	011111	010100	101101	100110	101011	110001	011
U4+C	000100	001111	010011	011000	100001	101010	100111	111101	100
U5+C	000010	001000	010101	011110	100111	101100	100000	111011	010
U6+C	000001	001010	010110	011101	100100	101111	100010	111000	001

Tableau 2.03: exemple de tableau de déchiffrement complet

Supposons que lors de la transmission du mot de code (101110), il arrive une erreur, par à la quatrième coordonnée.

On recevra donc le vecteur (101010).

Le décodeur fait les opérations suivantes :

- Calcul du syndrome S :

$$\mathbf{S = H * y}^t \qquad (2.08)$$

$$S = H * y^{t} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

(Il y a détection d'erreur car le syndrome n'est pas le vecteur zéro).

- Détermination du mot u qui a le même syndrome (100) :

U4 = (000100)
(101010) – (000100) = (101110)

- Conclusion :

Le décodage a été correct car le code est 1 erreur.

L'avantage de cette méthode est la minimisation du temps perdu à cause de l'utilisation du tableau de décodage.

Mais l'inconvénient est de ne trouver qu'une seule erreur, c'est-à-dire ne détecter qu'une seule erreur et une seule lors de la réception de l'information. Alors s'il y a une détection de k erreurs, on emploie une autre méthode.

2.3.5.4 Méthode en utilisant le tableau réduit

Nombre d'erreur	Mots erreurs	Syndromes
1 erreur	(1,0,…,0) (0,1,…,0) … (0,0,…,1)	Colonne de la matrice H
2 erreurs	(1, 1,0,…,0) (1, 0,1,…,0) … (0,0,…, 1,1)	Somme 2 à 2 des colonnes de la matrice H
…	…	…
e erreurs	… (1, 1,0,…, 1,0,…,0)	Sommes *e* à *e* des colonnes de H

Tableau 2.04 : tableau de déchiffrement réduit

Application

Soit un code qui a pour matrice génératrice et contrôle :

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \quad \text{et} \quad H = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Les colonnes sont non nulles et distinctes et la $4^{ème}$ est la somme de la première et le deuxième. Le code et donc 1-correcteur.

Mots erreurs	Syndromes
(0 0 0 0 0)	0, 0, 0
(1 0 0 0 0)	1, 0, 0
(0 1 0 0 0)	0, 1, 0
(0 0 1 0 0)	0, 0, 1
(0 0 01 0)	0, 1, 1
(0 0 0 0 1)	1, 0, 1

Tableau 2.05 : tableau des syndromes pour une erreur

Mots erreurs	Syndromes
1 1 0 0 0	1, 1, 0
1 0 1 0 0	1, 0, 1
1 0 0 1 0	1, 1, 1
1 0 0 0 1	0, 0, 1
0 1 1 0 0	0, 1, 1
0 1 0 1 0	0, 0, 1
0 1 0 0 1	1, 1, 1
0 0 1 1 0	0, 1, 0
0 0 1 0 1	1, 0, 0
0 0 0 1 1	1, 1, 0

Tableau 2.06 : tableau des syndromes pour deux erreurs

Mots erreurs	Syndromes
11100	1, 1, 1
11010	1, 0 ,1
11001	0, 1, 1
10110	1, 1, 0
10101	0, 0, 0
10011	0, 1, 0
01110	0, 0,0
01101	1, 1, 0
01011	1, 0, 0
00111	1, 1, 1

Tableau 2.07 : tableau des syndromes pour trois erreurs

Le mot reçu est y = (1, 1, 0, 1, 1)

Son syndrome $s = H \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

- Dans l'hypothèse d'une erreur au plus le mot reçu est dans le code : il n'y a pas d'erreur à la réception.

- Dans l'hypothèse deux erreurs au plus le mot reçu est dans le code : il n'y a pas d'erreur à la réception.

- Dans l'hypothèse trois erreurs au plus, le mot erreur ε est l'un des suivant :

  (0, 1, 1, 1, 0) ou (1, 0, 1, 0, 1)

  Et le mot envoyé x peut être :

  (1, 0, 1, 0,1) ou (1, 0, 1, 0,1)

Chapitre 3 CODE MC-DONALD

3.1 Définition [11] [13]

Un code MC Donald est un code linéaire spécifié par le vecteur P et la représentation modulaire de la matrice génératrice, notée par C (n, k) avec k défini la longueur du message et n définie la longueur du mot de code ou la dimension.

3.2 Représentation modulaire d'une matrice [11], [13]

Soit un code linéaire C (n, k) et soit G sa matrice génératrice.

3.2.1 Remarque

La matrice génératrice G a (q^k - 1) colonnes qui peuvent être placées dans n'importe quel ordre.

3.2.2 Définition

La représentation modulaire d'une matrice génératrice est la liste de nombre de colonnes différentes composant cette matrice et elle est représenté par :

$$L = (n_1, n_2, \ldots, nj, \ldots, nq^{(k-1)}) \qquad (3.01)$$

Où n jetant le nombre de colonnes de type j que possède la matrice G

3.2.3 Exemple

Pour k = 3 et n = 5

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

$$L = (1, 1, 1, 1, 1, 0, 0)$$

3.3 Distance minimale d'un code de MC-DONALD [13]

MC-Donald a recherché des codes linéaire des de groupe (n, k) ayant pour distance minimale la limite supérieure établie par Plotkin.

Limite de Plotkin :
C'est le poids minimale d'un mot de code dans un code linéaire (n, k) sur un corps K(q) qui est au plus égal à :

$$\text{Lim de Plotkin} = [nq^{(k-1)}(q-1)]/(q^k - 1) \qquad (3.02)$$

Le calcul de la distance minimale d'un code de MC-Donald est indiqué dans le tableau suivant :

L	Distance minimale d
$n = 2^k - 1$ et h = 0 (1 1 1 1 …1 1 1)	$2^{(k-1)}$
$n = 2^k - 2$ et h = 1 (1 1 1 1 …1 1 0)	$2^{(k-1)}$-1
$n = 2^k - 3$ et h = 2 (1 1 1 1 …1 0 0)	$2^{(k-1)}$-2
…	…
$n = 2^k - 2^u$ et $h^u = 1$ (1 1 1 1 0 0…0 0 0) Pour u=2, 3, 4,.., k-1	$2^{(k-1)}$-2^u +1

Tableau 3.01 : tableau donnant la distance minimale

3.4 Le vecteur P [13]

3.4.1 Définition

Le vecteur P est un vecteur ayant pour composantes le poids de chaque mot de code dans l'ordre de la matrice ligne génératrice.

3.4.2 Calcul de P

Soit la matrice M une matrice de dimension k $(q^k - 1)$ ayant comme colonnes tous les vecteur possible sur le corps K (q), excepté le vecteur nul. La j-ième colonne de cette matrice représente une colonne de type j

D'après MC-Donald, la liste des poids de tous les mots des codes, non nul, d'un code binaire peut être obtenu sous forme de composantes d'un vecteur résultant de la multiplication matricielle sur les corps de réels du vecteur L de sa représentation modulaire par la matrice N :

$$P = L \,.\, N \tag{3.03}$$

Tel que :

$$N = M^t \,.\, M \tag{3.04}$$

3.4.3 Exemple

Pour k = 3

En classant les colonnes de la matrice M suivant l'ordre des nombres binaire qu'elles représentent, nous obtenons :

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

Avec $N = M^t \,.\, M$ donc =>

$$N = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

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

Comme P = L . N

$$P = (1, 1, 1, 1, 1, 0, 0) * \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

P = (3, 2, 3, 2, 3, 4, 3)

3.4.4 Mots du code

La multiplication matricielle de M^t et G nous permet de donner les mots de codes. Pour le code ayant la matrice génératrice précédente, ses mots de codes sont :

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

3.5 Codage et décodage d'un code de MC-DONALD [15]

3.5.1 Codage

Pour avoir le mot codé, la méthode consiste à multiplier le message avec la matrice génératrice du code.

$$\mathbf{X = a . G} \qquad (3.05)$$

Exemple

Soit ‘a’ un message à envoyer, et soit G la matrice génératrice : a = (1 0 0)

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

Apres le codage, le mot codé x = a . G = (1 0 0 0 1)

3.5.2 Décodage

Soit Y le mot reçu au récepteur. Pour savoir le mot erreur et le mot envoyé, nous employons la méthode de décodage utilisant le tableau de déchiffrement réduit.

Exemple

Soit le reçu au récepteur y = (1 0 1 1 1)

Calculons le syndrome de ‘y’ par la multiplication de y avec H^t.

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \text{ et } H = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$S = y * H^t = (1\ 0\ 1\ 1\ 1) * \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Il faut construire le tableau de déchiffrement réduit

Nombre d'erreurs	Mots erreurs	Syndrome
1 erreur	(1 0 0 0 0)	(1,0)*
	(0 1 0 0 0)	(0,1)
	(0 0 1 0 0)	(1,1)
	(0 0 0 1 0)	(0,1)
	(0 0 0 0 1)	(1,0)*
2 erreurs	(1 1 0 0 0)	(1,1)
	(1 0 1 0 0)	(0,1)
	(1 0 0 1 0)	(1,1)
	(1 0 0 01)	(0,0)
	(0 1 1 0 0)	(1,0)*
	(0 1 0 1 0)	(0,0)
	(0 1 0 0 1)	(1,1)
	(0 0 1 1 0)	(1,0)*
	(0 0 1 0 1)	(0,1)
	(0 0 0 1 1)	(1,1)
3 erreurs	(1 1 1 0 0)	(0,0)
	(1 1 0 1 0)	(1,0)*
	(1 1 0 0 1)	(0,1)
	(1 0 1 1 0)	(0,0)
	(1 0 1 0 1)	(1,1)
	(1 0 0 1 1)	(0,1)
	(0 1 1 1 0)	(1,1)
	(0 1 1 0 1)	(0,0)
	(0 1 0 1 1)	(1,0)*
	(0 0 1 1 1)	(0,0)

Tableau 3.02: tableau de déchiffrement réduit

- Dans l'hypothèse d'une erreur au plus, le mot erreur est l'un des suivants :
  (1, 0, 0, 0, 0) ou (0, 0, 0, 0, 1)

⇨ Le mot envoyé peut être donc :
  (0, 0, 1, 1, 1) ou (1, 0, 1, 1, 0)

- Dans l'hypothèse de deux erreurs au plus, le mot erreur est l'un des suivants :
  (0, 1, 1, 0, 0) ou (0, 0, 1, 1, 0)

⇨ Le mot envoyé peut être donc :
  (1, 1, 0, 1, 1) ou (1, 0, 0, 0, 1)

- Dans l'hypothèse de trois erreurs au plus, le mot erreur est l'un des suivants :
  (1, 1, 0, 1,0) ou (0, 1, 0, 1,1)

⇨ Le mot envoyé peut être donc :
  (0, 1, 1, 0, 1) ou (1, 1, 1, 0, 0)

3.6 Comparaison entre code de HAMMING et code de MC-DONALD [3] [8] [13]

Voici un tableau de comparaison entre le code de HAMMING et code de MC-DONALD :

CODE DE HAMMING	CODE DE MC-DONALD
Matrice génératrice	Matrice génératrice
Corrige une seule erreur	Corrige jusqu'à t erreurs ($t \leq 3$)
Pas utilisation de représentation modulaire	Utilisation de représentation modulaire

Tableau 3.03 : tableau de comparaison entre les codes : Hamming et MC-Donald

3.7 Application

Soit un code de MC-Donald ayant pour matrice génératrice :

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

Soit y = (1 0 1 0 1 0 1) le mot reçu

Voici les étapes à suivre pour trouver le mot envoyé après la transmission :

- Chercher la représentation modulaire L de ce code.
- Donner la distance minimale de ce code
- Donner la capacité de correction d'erreur.
- Calculer les composantes du vecteur P.
- Chercher le mot envoyé

3.7.1 La représentation modulaire L de ce code

L = (1, 1, 1, 1, 1, 1, 1)

Car la longueur du mot de code est égale à $7 = 2^3 - 1$, donc h = 0

3.7.2 Distance minimale de ce code

D'après le tableau 3 qui donne la distance minimale du code de MC-Donald, nous obtenons :

$d = 2^{(k-1)}$ d'où $d = 2^{(3-1)} = 4$

d = 4

3.7.3 Capacité de correction d'erreur

Car d est paire, la capacité de correction d'erreur t peut être obtenue par la formule :

$$\mathbf{t = d / 2} \qquad (3.06)$$

Donc t = 4/2 = 2

3.7.4 Les composantes du vecteur P

3.7.4.1 Calcul de la matrice N

$$M = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad \text{et} \quad M' = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$N = M'.\ M$$

$$N = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

D’après les calculs on a :

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

3.7.4.2 Calcul du vecteur P

$$P = L * N$$

$$P = (1, 1, 1, 1, 1, 1, 1) * \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$P = (4, 4, 4, 4, 4, 4, 4)$$

3.7.5 Le mot envoyé

3.7.5.1 Calcul du syndrome

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

$$\boldsymbol{s = y * H^t}$$

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

3.7.5.2 Construction de tableau de déchiffrement réduit

Nombre d'erreurs	Mots erreurs	Syndrome
1 erreur	1000000	(0011)
	0100000	(0101)
	0010000	(1111)
	0001000	(1000)
	0000100	(0100)
	0000010	(0010)
	0000001	(0001)
2 erreurs	1100000	(0110)
	1010000	(1100)
	1001000	(1011)
	1000100	(0111)
	1000010	(0001)
	1000001	(0010)
	0110000	(1010
	0101000	(1101)
	0100100	(0001)
	0100010	(0111)
	0100001	(0100)
	0011000	(0111)
	0010100	(1011)
	0010010	(1101
	0010001	(1110)
	0001100	(1100)
	0001010	(1010)
	0001001	(1001)*
	0000110	(1110)
	0000101	(0101)
	0000011	(0010)

Tableau 3.04 : tableau de déchiffrement réduit

D’après ce tableau, le mot erreur est e = (0 0 0 1 0 0 1)

Donc le mot envoyé est :

$$x = y - e$$

x = (0 0 0 1 0 0 1) – (1 0 1 0 1 0 1) =

x = (1 0 1 1 1 0 0)

Pour conclure, on constate, après le décodage, qu’il existe deux erreurs, pendant la transmission de l’information.

Chapitre 4 PRESENTATION DU LOGICIEL MATLAB

4.1 Introduction sur MATLAB [15]

Matlab (ou MATrix LABoratory) est un logiciel très puissant qui permet de faire des calculs tel que les matrices et les opérations vectorielles. MATLAB a été développé par Math Works Inc (USA).

C'est un logiciel scientifique destiné aux ingénieurs, aux chercheurs, aux industriels…. Il possède un noyau dans lequel est intégré les outils, mathématiques classiques, fonctions élémentaires, calcul matriciel, graphisme D, 2D, 3D et quelque fonction de base de mathématiques, traitement du signal. Matlab est un langage interprété qui permet de développer des algorithmes rapidement, de visualiser des données (sous la forme de graphiques 2D ou 3D et d'images, voir de séquences d'images), et de réaliser des interfaces graphiques conviviales.

Il est un logiciel additionnel pour la simulation des circuits. Il est une puissante collection d'outil pour le développement d'algorithme, d'information, de visualisation. Il crée plus de contrôles et de flexibilité comparé à un langage de programmation traditionnel de haut niveau.

Dans Matlab, il existe un seul type de donnée : le type matrice. Il n'y a donc pas de déclaration de type. Les matrices et les vecteurs peuvent être redimensionnés.

Communication toolbox :
La communication Toolbox est une collection des fonctions d'information et des blocs de simulation pour la recherche, le développement' l'analyse et la simulation dans le champ de communication. La communication Toolbox est créée pour utiliser le Matlab et SIMULINK
On peut utiliser directement Toolbox à partir du Matlab.
Les fonctions disponibles :

- Source de donnée
- codage/décodage de source
- code détecteur d'erreur
- modulation/démodulation
- filtre de réception
- canal de transmission
- accès multiples
- synchronisation

4.2 Prise en main [15]

4.2.1 Démarrage

Matlab va interpréter les commandes que vous aurez sauvegardé dans un fichier texte avec l'extension ".m".

Il vous faut donc créer un répertoire de travail ("C:\xxxx\yyyyyyy" par exemple) pour y sauvegarder vos programmes, lancer le Matlab et lui demander d'exécuter les commandes sauvegardées dans votre fichier.

4.2.2 Lancement

Pour démarrer Matlab, il suffit de cliquer dans l'icône "Matlab" si vous êtes sous Windows, ou de taper la commande Matlab si vous êtes sous Unix.

L'espace de travail de Matlab se présente alors sous la forme d'une fenêtre affichant un prompt ">>" à la suite duquel vous pouvez taper une commande qui sera exécutée après avoir tapé sur la touche "return".

En haut de cette fenêtre se trouve une barre de menu qui vous permet d'ouvrir un fichier texte, de définir certaines variables de travail et surtout d'accéder à l'ensemble des fichiers d'aides.

Il faut indiquer à Matlab le répertoire dans lequel vous voulez travailler (celui qui contient vos programmes).

4.3 Exemple de simulation de circulation de données

On donne un exemple pour expliquer la conception de simulation de circulation de données. Il montre un codage linéaire correcteur d'erreur. La simulation utilise la fonction matlab de codage linéaire correcteur d'erreur. Le message source utilisé dans la simulation est une séquence d'entiers aléatoires ;

Sur les lignes de commandes, les lignes commençant avec '%' sont des lignes de commentaires.

```
% Matrice génératrice du code
G = [1 0 0 1 0 0 ; 1 1 1 0 1 0 ; 0 1 1 0 0 1]
G' = [];
G'
[K, N]= size (G)
```

```
Pause
% Matrice de controle
H=Hammgen (G)
Pause
%Message
Msg= rangint (k, 1, 2)
%Codage du message
G'
Code = msg*G'
Pause
% mot reçu
Code_noise= rem (code+rand (N, 1)>.70.2)
Pause
% Décodage
Rcv=decode (code_noise, N, K,'linear', G)
Pause
% Taux d'erreur
Disp (['Error rate after decode :'…
        Num2str (symerr (code, code_noise) / max (size (code)))])
Pause
Disp (['Erreur rate after decode:'…
        Num2str (symerr (msg, rcv) / max (size (msg)))])
```

4.4 Modélisation d'un système de communication

Un système de communication typique inclut:

- Un signal source
- Un signal reçu
- Canal de transmission
- Réception

La section dans ce chapitre suit le schéma d'émission/réception de la figure suivante :

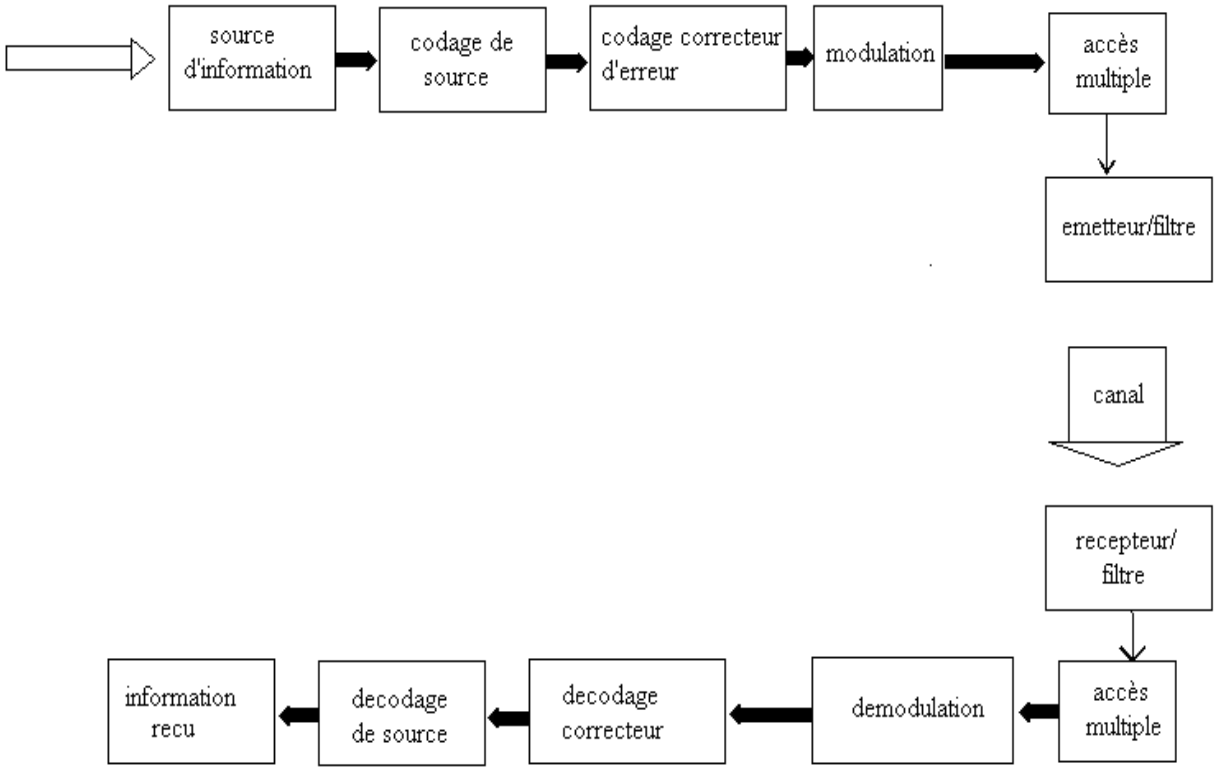

Figure 4.01 : schéma de l'émission/réception des informations

4.4.1 Générateurs de signal et périphérique d'affichage

Cette boite à utile inclut certain nombres de générateurs de signal, des procédures d'affichage de signal et des fonctions. Les générateurs de signal introduit dans cette section qui, tous génèrent des signaux au hasard.

4.4.1.1 Générateur de signal au hasard

Cette boite à outil contient : Gaussien, Rayleigh, Rician, Poisson et générateur de bruits uniforme dans le librairie de Simulink bloc.

On trouver dans les fonctions MATLAB pour générer les variables au hasard dans Statistiques Toulbox.

Le bruit Gaussien est plus utilisé dans beaucoup de modèles dans les systèmes de communication, le AWGN (Additive White Gaussien Noise) canal.

Le bruit de Rayleigh est très utilisé les canaux de radio, comme les canaux de radio cellulaire.

4.4.1.2 Périphérique d'affichage

La communication Toolbox a un périphérique d'affichage qui lui rang facile à analyser la performance du système de communication.

4.4.2 Codage de source

La communication Toolbox inclut certaines fonctions de base pour le codage de source. Coder la source qui est une quantification et formatage du signal, inclut des concepts de conversation analogique en numérique et compression de données

Le codage source convertit le signal source en code numérique utilisant la méthode de quantification.

Le décodage de source recouvre la séquence du signal d'information originale utilisant le signal source codé.

4.4.3 Codage correcteur d'erreurs

Les techniques du codage de correcteur d'erreurs sont utilisées pour détecter et/ou corriger les erreurs dans le message émis dans le système de communication numérique.

Le codage correcteur d’erreur dans la transmission consiste à inclure des bits de redondances ou symboles de redondances dans le signal information original. Le codage de correcteur d’erreurs dans la réception utilise ces redondances pour détecter et/ou corriger les erreurs qui se sont introduites pendant l’émission.

Le processus de code d’émission est appelé codage et celui de réception décodage.
On peut exécuter tous les calculs de codage en utilisant la fonction ‘‘ encode’’. Le format de encode est :

Code = encode (msg, N, K, ‘method’, opt) ;

Et pour le décodage, la fonction ‘‘decode’’

msg = decode (code, N, K, ‘method’, opt1, opt2) ;

Chapitre 5 CONCEPTION D'UN LOGICIEL DE CODAGE "CODE MC DONALD"

5.1 Présentation du logiciel [15]

Ce logiciel traite le codage et le décodage d'un message quelconque. Mais c'est le code linéaire en particulier le code MC-Donald que nous avons employé dans ce logiciel.

Ce logiciel fonctionne avec le MATLAB version 7.0.
Il permet de voir des résultats pour :

- le message émis codé
- la distance minimale
- la capacité de correction d'un code
- la matrice génératrice
- la matrice de contrôle
- le mot erreur pendant la transmission
- le mot codé
- le message reçu à la réception
- les nombres d'erreur dans les mots
- les positions d'erreur dans les mots reçus

5.2 Quelques commandes utilisées dans ce logiciel

- ***matgen (n,k):*** donne la matrice génératrice d'un code MC-Donald

```
function aa=matgen (n,k)
%Matrice génératrice d'un code
   gen=[];
   for i=1:k;
      gen (:,i)=dec2bin(2^(k-i),k)';
   end
   if k+1<=n
      gen (:,( k+1)) = (dec2bin (3,k))';
   end
   if k+2<=n
      gen(:,(k+2))=(dec2bin(4,k))';
   end
```

```
if k+3<=n
   gen(:,(k+3))=(dec2bin(5,k))';
end
if k+4<=n
   gen(:,(k+4))=(dec2bin(6,k))';
end
if k+5<=n
   gen(:,(k+5))=(dec2bin(9,k))';
end
if k+6<=n
   for i=(k+6):n
      gen(:,i)=(dec2bin(i,k))';
   end
end
aa=mod(gen,2);
```

- ***matcont (gen):*** donne la matrice de contrôle d'un code.

```
function aa=matcont(gen)
%Matrice de contrôle
   a=size (gen);
   k=a(1);
   n=a(2);
   m=gen (:,k+1:n);
   el=[];
   for i=1:n-k;
      el(:,i)=dec2bin(2^(n-k-i),n-k)';
   end
   el = mod(el,2);
   res = cat(1,m,el)';
   aa=res;
```

- ***mescod (msg,g1):*** donne la représentation modulaire d'une la matrice génératrice

```
function aa=mescod(msg,g1)
gen=g1;
mes=rem(msg*gen,2);
aa=mes;
```

- ***detect (msgrec, h, nberr)*** : Détecte les erreurs et les corrige

```
function aa=detect (msgrec, h, nberr)
% DETECT (msgrèçu, h, nberr)= [msgcorrigé, place des erreurs]
   Syndrome=mod (msgrec*h',2);
   if syndrome==max(zeros(length(syndrome)))
         res=msgrecu;
         nberr1=[];
   else
      rg=existe(syndrome',h);
      if rg>0
         res=correct(msgrec,errmot(max(zeros(length(msgrec))),[rg]));
         nberr1=[];
      else
         materr=errorgen(length(msgrec),nberr);
         matsynd=syndgen(materr,h);
         rang=existe(syndrome',matsynd');
         if rang==0
            disp('Le code n''a pas d''erreur');
            res=msgrec;
            nberr=[];
         else
            err=materr(rang,:);
            res1=correct(msgrecu,err);
            nberr1=res1(length(msgrec)+1:length(res1));
            res=res1(1:length(msgrec));
         end
      end
   end
   res=[res,nberr1];
   while length(res)<nberr+length(msgrec)
      res(length(res)+1)=0;
   end
   aa=res;
```

- ***dismin (n,k) :*** donne la distance minimale

```
function aa=dismin(n,k)
%Distance minimale
%DISMIN(n,k)=[distance minimale,capacite de correction]
dis=1;
   for i=1:2^k-1
      if n==2^k-i
            dis=2^(k-1)-(i-1);
      end
   end
   res1=(dis-1)/2;
   res=ceil(res1);
   aa=[dis,res];
```

- ***correct (msgrec,err):*** Corrige les erreurs dans le message

```
function aa=correct(msgrecu,err)
%Corrige les erreurs dans le message et renvoie
%les places de ces erreurs
   nberr=0;
   for i=1:length(err)
      if err(i)==1
         nberr=nberr+1;
         plerr(nberr)=i;
         msgrec(i)=mod(msgrec(i)+1,2);
      end
   end
   aa=[msgrec,plerr];
```

- ***errorgen (n,t) :*** donne la longueur des mots reçus

```
function aa=errorgen(n,t)
%n:longueur des mots reçus
%t:nbre d'erreur
```

```
err=[];
ref=zeros(1,t);
ref=errmot(ref,1);
mmax=0;
for a=1:t
   mmax=mmax+fact(n)/(fact(a)*fact(n-a));
end
for i=1:mmax
   res=max(zeros(n));
   res=errmot(res,ref);
   err=cat(1,err,res);
   ref=incr(ref,n);
end
aa=err;
```

- ***existe (rech,mat)***

```
function aa=existe(rech,mat)
   if length(mat)==0
      aa=0;
      return
   else
      lg=size(mat);
      for i=1:lg(2)
         if mat(:,i)==rech
            aa=i;
            return
         end
      end
      aa=0;
      return
   end
```

5.3 Fonctionnement du logiciel

- ***Fenêtre d'accueil***

Figure 5.01 : Fenêtre d'accueil du logiciel

- Bouton <FERME> : fermeture de la fenêtre
  On peut aussi employer le raccourci « contrôle Q »
- Bouton <SUIVANT> : passage à la fenêtre suivante

- ***Fenêtre de la matrice génératrice***

Cette fenêtre donne la distance minimale, la capacité de correction du code, longueur du code et la dimension du code.

Exemple : C (12,4).

Cette fenêtre donne aussi, automatiquement la matrice génératrice *G* d'un code donnée.

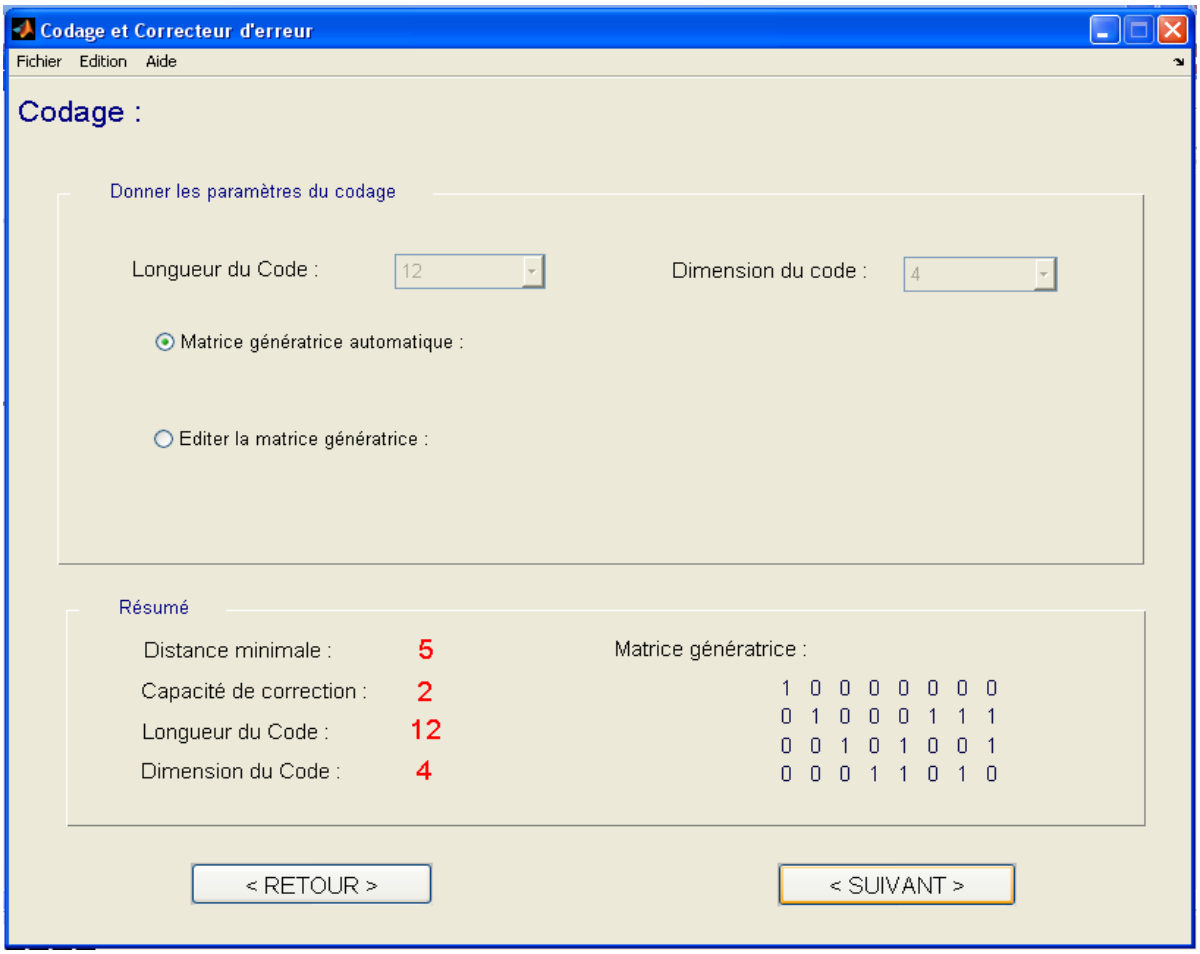

Figure 5.02 : Fenêtre de la matrice génératrice

- Bouton <RETOUR> : revient à la fenêtre précédente (fenêtre d'accueil)
- Bouton <SUIVANT> : passage à la fenêtre suivante

Remarque : La longueur du code doit être supérieure à la dimension du code, sinon cette fenêtre de signalisation apparaît.

- Si le code est non utilisable cette fenêtre de signalisation apparaît.

Figure 5.03 : Fenêtres de signalisations

- ***Fenêtre du mot codé***

Cette fenêtre donne les mots codés avec la matrice génératrice.

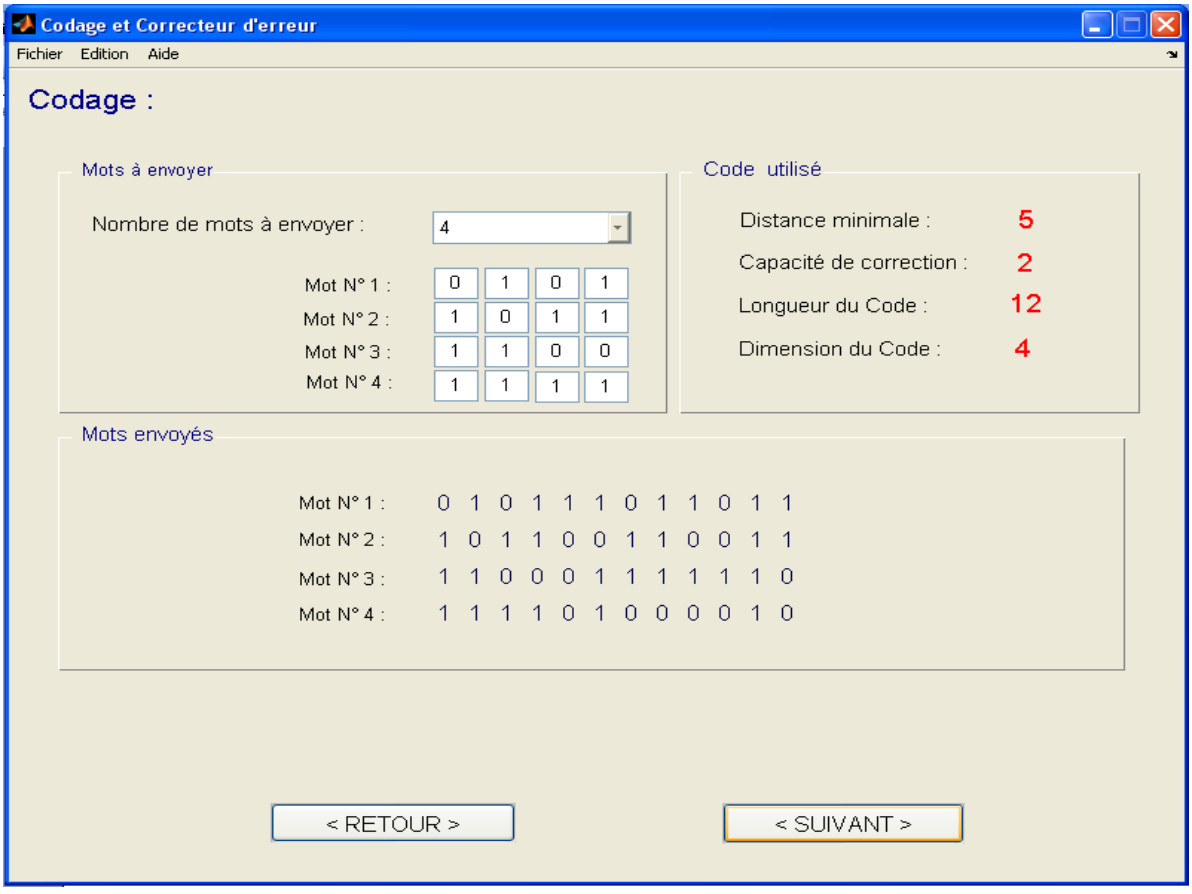

Figure 5.04 : Fenêtre de mot codé

- ***Fenêtre de visualisation des erreurs***

Les mots qu'on désire envoyés (de 1 à 4 mots au maximum) s'affiche dans le panneau Mots envoyés. Comme le logicielle créé ne simule pas un canal de transmission provoquant des erreurs sur les informations transmises ; nous l'avons conçue de façon à ce qu'on puisse cocher les bits que l'on désire erronés. Le second panneau affiche les mots reçus avec les positions et les nombres des erreurs de chaque mot.

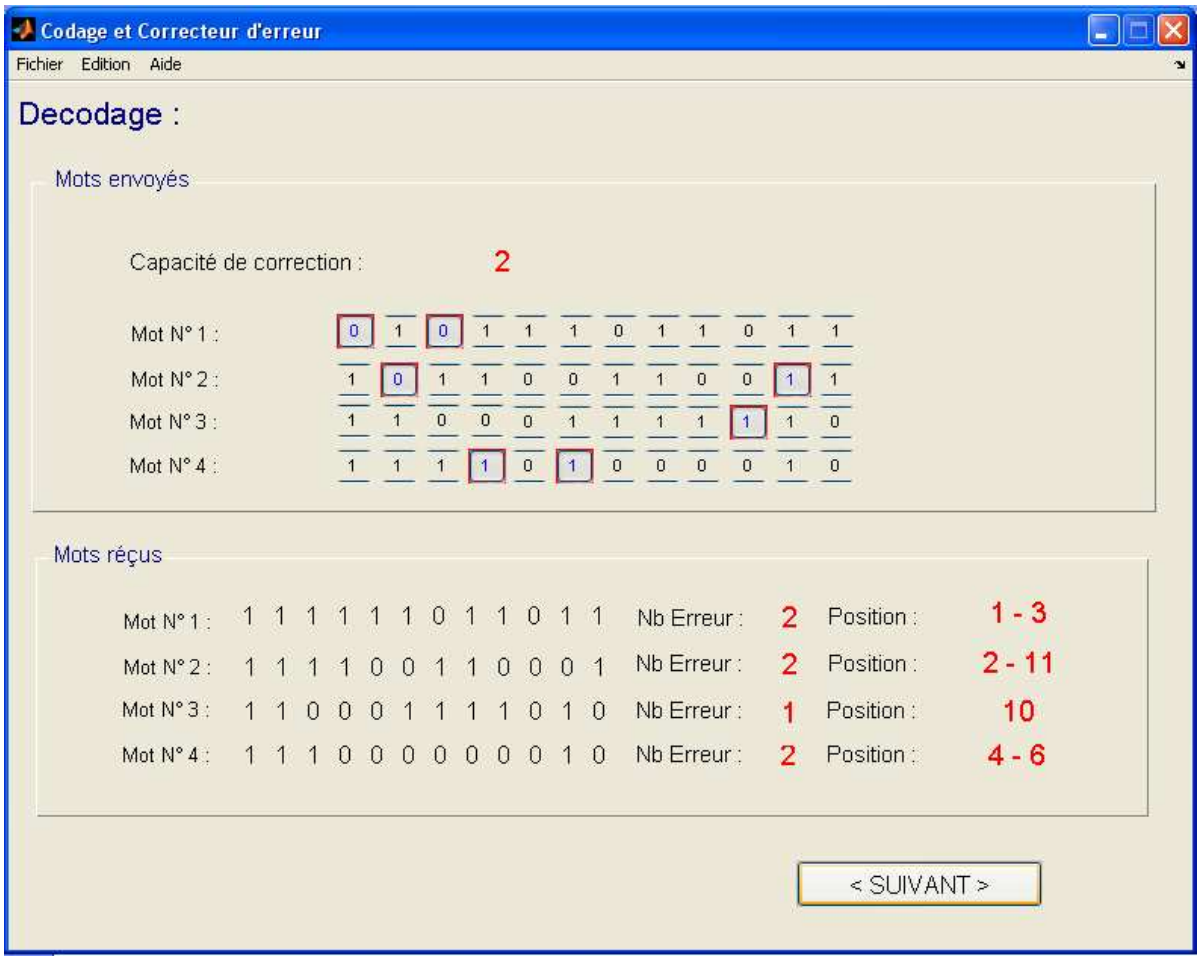

Figure 5.05 : Fenêtre de visualisation des erreurs

- ***Fenêtre de la matrice de contrôle et du syndrome***

Cette fenêtre nous donne les syndromes de chaque mot et la matrice de contrôle H.

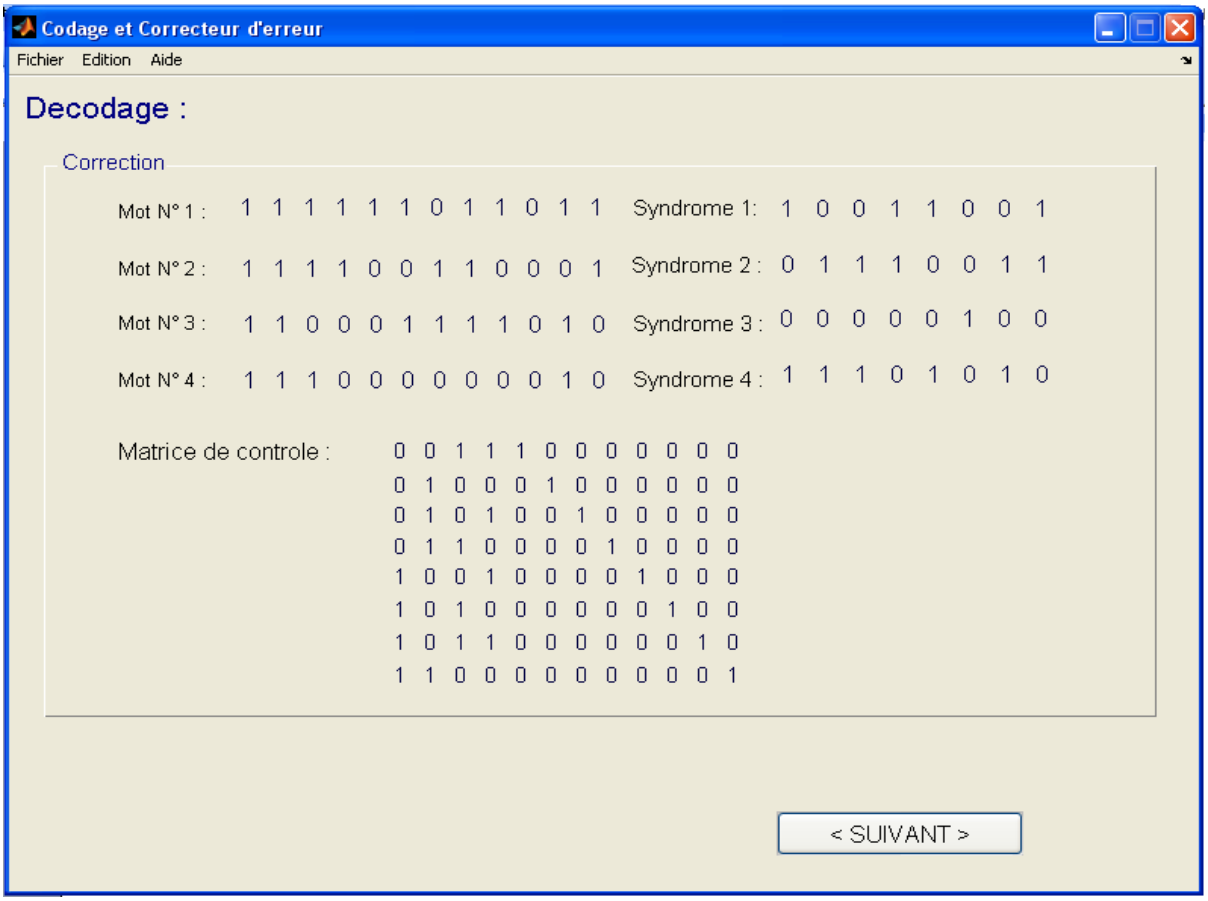

Figure 5.06 : Fenêtre de la matrice de contrôle et du syndrome

- ***Fenêtre du mot corrigé***

Cette fenêtre donne les mots corrigés, le décode pour ensuite obtenir le message.

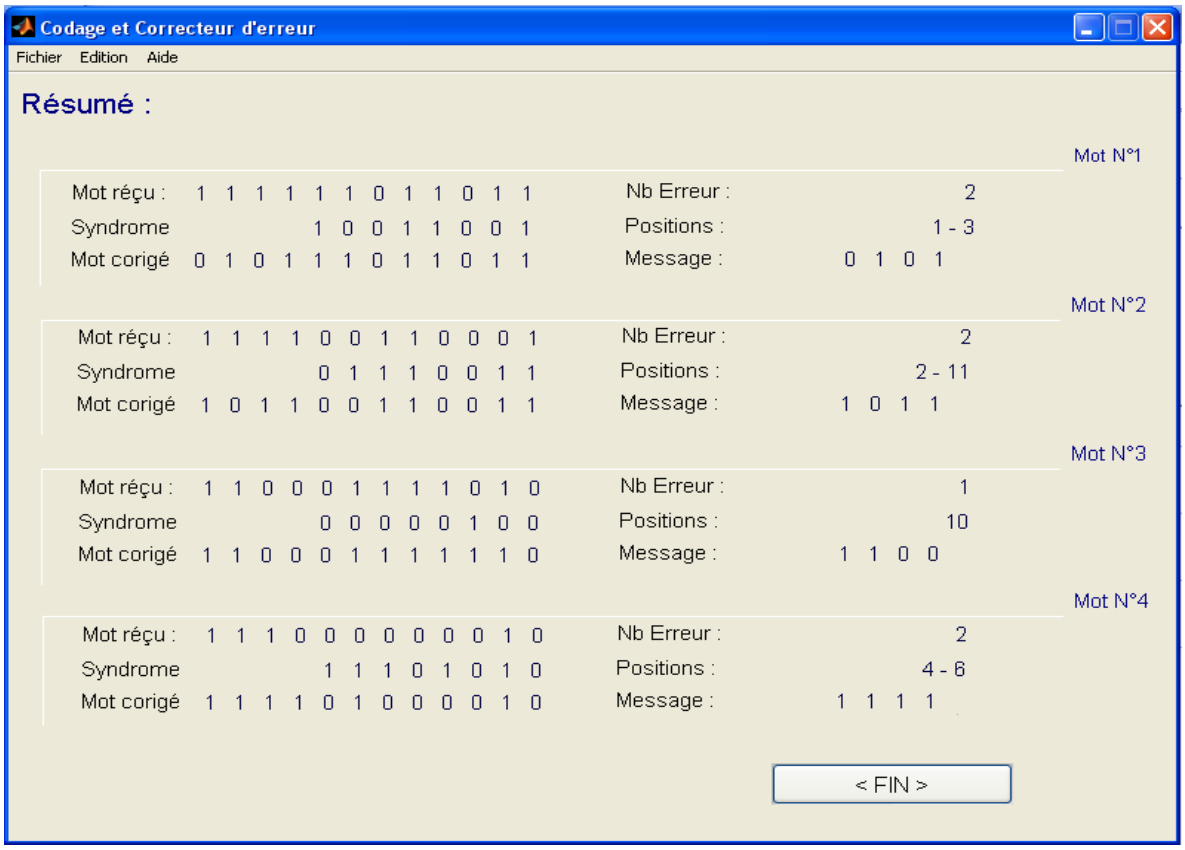

Figure 5.07 : Fenêtre du mot corrigé

- Bouton <FIN> : fin de la simulation

- Dans l'onglet Fichier, pour faire une nouvelle application, il faut cliquer sur Nouveau : pour faire une nouvelle application. On peut aussi employer le raccourci « contrôle N ».

CONCLUSION

Le codage et la détection d'erreur sont des techniques très importantes et incontournable dans le domaine des télécommunications. Il existe trois techniques de détection d'erreurs qui sont : la détection par écho, la détection par répétition et la détection et correction d'erreurs par code. Notons que ce dernier est la technique utilisée dans le présent ouvrage.
La progression du système de communication est due au développement de l'information où le principal objet est d'avoir une transmission de l'information dans un minimum de temps avec un minimum d'erreur.

L'amélioration de la qualité dans une transmission est obtenue en utilisant la technique de codage et correcteur d'erreurs.

Les caractéristiques du code détecteur et correcteur d'erreurs présentées et étudiées dans cet ouvrage nous ont conduit à proposer l'utilisation d'un code de MC-DONALD dans le système de communication.

La conception du logiciel de codage et correcteur d'erreurs, code MC-Donald consiste en l'étude théorique ainsi que pratique de la théorie de l'information et du codage. L'environnement du logiciel présente une interface simple et lucide pour la compréhension ; car le logiciel n'a pas été conçu que pour les experts en Matlab, il vise surtout les initiés pour l'étude et la compréhension de la théorie de l'information et du codage.

Pour conclure, on peut dire qu'il est plus particulièrement, un outil pédagogique pour les étudiants dans le domaine des télécommunications qui s'intéressent au traitement de codage ainsi qu'a exploitation du logiciel MATLAB.

ANNEXE I : ESPACE VECTORIEL

I.1. Structure d'un espace vectoriel [8] [12] [14]

Soient A et B des ensembles. On appelle **loi externe** sur B une application $A \times B \to B$.

Considérons un ensemble E des suites des ordonnées x1 x2 x3 … xi … xn de n éléments xi d'un élément de K muni d'une structure de corps. Un élément X de l'ensemble E

X= (x1, x2, x3, … xi, … xn) et xi ∈ K est appelé un n-uple

Sur l'ensemble E ainsi précisé, on définit une structure d'espace vectoriel par les deux lois suivantes :

- une loi composition interne ou somme de deux vecteurs

  X= (x1, x2, x3, … xi, … xn) et Y= (y1, y2, y3, … yi, … yn)

  X + Y = (x1+ y1, x2+ y2, x3+ y3, … xi+yi, … xn+ yn)

- une loi externe ou produit d'un vecteur X par un élément z de K :

  z .X= (z.x1, z.x2, z.x3, … z.xi, … z.xn)

On constate qu'à partir de ces deux définitions, la somme est une loi de groupe abélien. Elle est associative, commutative, il existe qui est le vecteur nul (0, 0, 0, 0, ….), 0 étant l'élément neutre de la loi additive de K.

Tout vecteur X possède un opposé : - X = (-x1, -x2, -x3, …-xi, … -xn)

I.2. Structure d'un sous-espace vectoriel [12] [14]

F est dit sous espace vectoriel de E si: Xi, Xj ∈ F entraîne Xi + Xj ∈ F et a.Xi ∈ F pour a∈ K.

Dans ces conditions : aXi + aXj ∈ F ∀ les scalaires a, b dans K. en particulier

Xi + (-Xj) = 0 ∈ F et les opérations définies sur E donne bien F d'une structure d'espace vectoriel.

ANNEXE II : LA MATRICE

I.1. Définition [8] [14]

Une matrice à m lignes et n colonne est un tableau rectangulaire de *m n* nombres, rangés ligne par ligne. Il y a m lignes, et dans chaque ligne n nombres.

Soient A un ensemble et (m, n) un couple d'entiers positifs. Le plus souvent, l'ensemble A est muni d'une structure de corps commutatif mais on utilise aussi fréquemment des matrices à coefficients dans un anneau.
On appelle matrice à coefficients dans A, de dimension (ou taille) (m,n) c'est à dire à m lignes et n colonnes-, une famille (ai,j)d'éléments de A indexée par le produit cartésien des ensembles de nombres entiers [1,m] et [1,n].

La matrice M pourra être notée par, $M = (a_{i,j})1 \leq i \leq m, 1 \leq j \leq n$

On représente généralement une matrice sous la forme d'un tableau rectangulaire. Par exemple, est représentée ci-dessous une matrice M, à coefficients entiers, et de dimension (3,4) :

$$M = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{pmatrix}$$

Dans cette représentation, le premier coefficient de la dimension est le nombre de lignes, et le deuxième, le nombre de colonnes du tableau. Une matrice pour laquelle le nombre *m* de lignes est égal au nombre *n* de colonnes sera dite matrice carrée de taille *n*. Une matrice ne comportant qu'une seule ligne et *n* colonnes est appelée matrice ligne de taille *n*. Une matrice ne comportant m lignes et une seule colonne est appelée matrice colonne de taille *m*.

Pour repérer un coefficient d'une matrice, on indique son indice de ligne puis son indice de colonne, les lignes se comptant du haut vers le bas et les colonnes de la gauche vers la droite. Par exemple, on notera ai, j, les coefficients de la matrice M, pour $1 \leq i \leq 3$ désignant le numéro de la ligne sur laquelle figure le coefficient envisagé, et $1 \leq j \leq 4$ désignant son numéro de colonne ; ainsi $a_{2,4} = 7$.

La disposition générale des coefficients d'une matrice M de taille (m,n) est donc la suivante :

$$M = \begin{bmatrix} a_{1,1} & a_{1,2} & ... & a_{1,n} \\ a_{2,1} & a_{2,2} & ... & a_{2,n} \\ ... & ... & ... & ... \\ a_{m,1} & a_{m,2} & ... & a_{m,n} \end{bmatrix}$$

Pour effectuer certaines opérations, il peut être utile de travailler sur le système des lignes ou des colonnes d'une matrice. On pourra alors l'écrire sous une des formes suivantes :

$$M = \begin{pmatrix} L_1 \\ L_2 \\ ... \\ L_3 \end{pmatrix} \qquad \text{ou} \qquad M = (C_1 C_2 ... C_n)$$

L'ensemble des matrices à coefficients dans A possédant m lignes et n colonnes est noté $M_{m,n}(A)$. Lorsque m=n on note plus simplement $M_n(A)$.

Soit : $M = (a_{i,j}) 1 \le i \le m, 1 \le j \le n \in M_{m;n}(A)$

On appelle transposée de A la matrice : ${}^tM = (a_{j,i}) 1 \le j \le n, 1 \le i \le m$

Par exemple, avec la matrice M des exemples précédents, on a :

$${}^tM = \begin{pmatrix} 0 & 4 & 8 \\ 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \end{pmatrix}$$

L'opération de transposition est involutive, c'est-à-dire que : ${}^t({}^tM) = M$

II.2. Produit matriciel [14]

On commence par définir le produit d'une matrice ligne par une matrice colonne. Soit n un nombre entier, L une matrice ligne, x_i ses coefficients, C une matrice colonne, y_i ses coefficients. On les suppose toutes deux de taille *n*. On définit alors le produit, considéré comme un scalaire ou une matrice de dimension *(1,1)* :

$$LC = (x_1.....x_n)\begin{pmatrix} y_1 \\ ... \\ y_n \end{pmatrix} = \sum_{1\le i\le n} x_i y_j$$

Ou :

$$\begin{pmatrix} L_1 \\ ... \\ L_m \end{pmatrix}(C_1.....C_p) = \begin{pmatrix} L_1C_1 & L_1C_2 & ... & L_1C_p \\ L_2C_1 & L_2C_2 & ... & L_2C_p \\ ... & ... & ... & ... \\ L_mC_1 & L_mC_2 & ... & L_mC_p \end{pmatrix}$$

II.3. Matrice identité et inverse d'une matrice [14]

Pour chaque nombre entier n, on note I_n la matrice carrée de taille n dont les coefficients diagonaux sont égaux à 1 et dont les autres coefficients sont nuls ; elle est appelée matrice identité de taille *n.*

$$I_1 = 1; I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}; I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; I_n = (\delta_{i,j})1 \le i \le n, 1 \le j \le n$$

Où $\delta_{i,j}$ désigne le symbole de Kronecker.

ANNEXE III : LES COMMANDES DU LOGICIEL MATLAB

III.1. les commandes [15]

ABS :

ABS (x) est la valeur absolue de la matrice x ;

ADDPAPH :

ADDPAPH ajoute un répertoire pour chercher un chemin

ADDPAPH DIRNAME ajoute un répertoire spécifié vers un chemin courant du Matlab.

CEIL :

CEIL(x) arrondit les éléments de x au entier supérieur à ces éléments.

CLOSE :

CLOSE ferme une fenêtre ouvert

CLOSE (‘name’) : ferme la fenêtre nommée ‘name’

CLOSE ALL ferme toutes les fenêtres

DECODE :

DECODE calcul de décodage.

Msg = DECODE (code, n, k) décode le mot de code binaire sur CODE en utilisant le code de HAMMING. La longueur du mot de code est n et la longueur du message à envoyer est k.

Msg = DECODE (CODE, N, K, METHOD, OPT1, OPT2,) decode le mot de code CODE en utilisant le code correcteur d’erreur dont la longueur du mot de code est N et la longueur du message à envoyer est K.

METHOD :

‘Hamming’ : code de Hamming

‘Linear’ : code lineaire

‘Cycl’ : code cyclique

‘bch’ : code BCH

‘rs’ : code Reed solomon

‘convol’ : code convolutif

DE2BI : convertit un nombre décimal positif en un nombre binaire

ENCODE : coder un message

FIGURE : crée une figure ou fenêtre graphique.

FLIPLR :

FLIPLR(x) donne la matrice x avec une direction changée de gauche à droite

FLOOR :

FLOOR(x) arrondi les éléments de x aux entiers inférieurs à ces éléments

LENGTH :

LENGTH(x) donne la longueur du vecteur x.

NUM2STR :

NUM2STR convertit le nombre scalaire x en une représentation d'une chaîne de caractères.

ONES :

ONES (N) est une matrice de N par N dont les éléments ont tous la valeur 1.

ONES (N, M) est une matrice de N par M dont les éléments ont tous la valeur 1.

PAUSE

PAUSE est une procédure d'arrêter le programme et attend que l'utilisateur tape une touche

PAUSE (n) pause d'une durée de n secondes

SIZE : donne la dimension d'une matrice

III.2 Environnement de Matlab

Matlab est un environnement de calcul numérique matriciel. Après le lancement de Matlab, une fenêtre de commande apparaît qui permet à l'utilisateur de taper une commande quelconque obéissant à la syntaxe de Matlab.

Matlab permet d'ouvrir, de créer, de modifier,… des fichiers. Matlab sauvegarde tous les fichiers créés dans le répertoire par défaut qu'il est possible de modifier à l'aide de la commande "cd" ou en lançant le "path browser".

Le symbole % introduit un commentaire, celui-ci n'est pas évalué par l'interpréteur.

III.3. Données de Matlab

- **Vecteurs**

Vecteur ligne

```
>> v = [1 2 3 4 5] ; % vecteur 1*5
>> v = [1:5]; % résultat identique à la ligne précédente (incrément de 1 par défaut)
```

>> v = [1:1:5]; % idem incrément spécifié

Vecteur transposé

>> v = [1 2 3 4 5] '; % vecteur 5*1

>> v = [1:5]';

>> v= [1:1:5]';

>> w = v';

>> v1 = [borne inf:incrément:borne sup];

>> v1 (i) ; % ième élément ATTENTION le premier est à i = 1

>> Length (v) %dimension du vecteur

- **Matrices**

>>A = [1 2 3; 4 5 6;7 8 9; 10 11 12]; %matrice rectangle 4*3

>>B = [1 2 3] %matrice 3*1

>> Size (A); %dimension de la matrice

>>A (i,j); % élément ligne i colonne j

>>A (:,n); % nième colonne

>>A (p,:); % pième ligne

>>A (i:j,:); % sous matrice des lignes i à j

>>A (i:j,k:l); % sous matrice des lignes i à j colonnes k à l

- **Suppression des données**

>> clear % supprime toutes les variables
>> clear A % supprime la variable A

- **Sauvegarde des données**

save nom_fichier A B C % sauve les variables A B et C dans le fichier nom_fichier
load nom_fichier % récupération des données

BIBLIOGRAPHIE

[1] J. C. Dany, M.C.Dumas, *codes correcteurs d'erreurs : Ecole Supérieure d'Electricité.*

[2] J. N. Rasamoelina, *Théorie de l'Information et du Codage,* cours 2ème année, Dép. Tél. - E.S.P.A., A.U. : 2004 - 2005.

[3] D. Forney, *On the Hamming distance property of group codes*, IEEE Trans. Information Theory, Vol. IT-38, No. 6, pp.

[4] P. Csillas, *Introduction aux Codes Correcteurs*, Ed Ellipses, 1996.

[5] G. Lachaud, S.Vladut, *Les codes correcteurs d'erreur*, col. 26, n°278, pp. 778 782, juillet- août 1995.

[6] O. Huguet, *Codes correcteurs – Théorie et application*, Masson ,1989.

[7] J. N. Rasamoelina, *Codage et Trafic,* cours 3ème année, Dép. Tél. - E.S.P.A., A.U. : 2006 - 2007.

[8] http://www.math.u-psud.fr.

[9] G. Cohen, J. L. Dornstetter et P.Godlewski, *Codes correcteurs d'erreurs : Une introduction au codage algébrique*, Masson.1984.

[10] R. T. Andriatsiferana, *Le code convolutif en utilisant le décodeur de Viterbi,* mémoire de fin d'étude, dép. Tél., A.U. :2003- 2004.

[11] A. M. Zanotti *Codage et correcteur d'erreurs* Dunod, 1992.

[12] http://www.ugm.univ-mlv.fr.

[13] B. R. McDonald, *Finite Rings with Identity*, Marcel-Dekker, 1974.

[14] A. Ramasy, *Algèbre,* cours 1ère année Département Télécommunications E.S.P.A., 2006 et 2007.

[15] V. Heidelberg, *Apprendre et maîtriser MATLAB,* Springer 1997.

RENSEIGNEMENTS

Nom : ANDRIANARISON RAKOTONIRAINY
Prénom : Toky
Adresse de l'auteur : Cité 67 ha Nord-Est Lot 2087 bis Antananarivo 101 Madagascar
Tél. : (261) 33.12.613.03 / (261) 33.12.141.89
E-mail : tokytelecom@yahoo.fr

Titre de mémoire : CONCEPTION D'UN LOGICIEL DE CODAGE ET CORRECTEUR D'ERREURS : CODE MC DONALD

Nombres de pages : 62
Nombres de figures : 11
Nombres de tableaux : 09

Mots clés : Information, message, codage, MC Donald, simulation et correction d'erreurs

Directeur de mémoire : RASAMOELINA Jacques Nirina

RESUME

Ce mémoire est consacré à l'étude théorique ainsi que pratique, la technique de codage et correcteur d'erreur. Il donne des explications et des méthodes de calculs du codage et du décodage. La seconde partie est constituée par un guide d'utilisation de ce logiciel, qui est entièrement réalisé sous MATLAB. On y trouve aussi des explications ainsi que des programmations de chaque fonction qui a été utilisée pour la réalisation du logiciel. Ce livre s'adresse à tous les étudiants dans le domaine télécommunication et surtout ceux qui ont une connaissance de base en théorie de l'information et du codage.

ABSTRACT

This dissertation is devoted to the theoretical study and practical, the technical coding and correcting. It provides explanations and methods of calculations encoding and decoding. The second part is consisting of a teacher's guide to using this software, which is written entirely in MATLAB. There are also explications as well as programs for each function that was used for the implementation of software. This book is intended for all students in the field of telecommunication and especially those who have a basic knowledge in information theory and coding.