TIPE : Code correcteur d'erreurs

Melvyn EL KAMEL–MEYRIGNE

Sous la direction de Benoit Fabrèges

Table des matières

Introduction

Lorsque deux personnes communiquent entre elles, il arrive souvent que le message reçu diffère de celui émis ; que ce soit à cause de fautes d'orthographes, d'une diction trop rapide ou bien de bruits perturbateurs. Toutefois, il n'est pas toujours nécessaire d'obtenir l'intégralité du message pour en deviner le sens. Ainsi, même si on mélange l'ordre des lettres dans un mot tout en laissant les lettres aux extrémités à leur place, on puet qnaud mmêe cmopnrde la prshae.

On constate que le transfert d'informations numériques est très important dans de nombreux domaines, par exemple pour les sondes spatiales qui reçoivent des signaux parcourant une énorme distance. Il est donc primordial de s'assurer que la perte d'informations soit minimale : c'est là qu'interviennent les codes correcteurs.

L'objectif est de rajouter une couche d'informations au message initial qui ne rajoute pas de sens mais qui permettront de détecter les erreurs et de les corriger. Bien évidemment, l'ajout d'informations a un prix : l'efficacité d'un code sera donc déterminée par la quantité d'informations ajoutée et par sa capacité à conserver l'information d'un message. Nous allons nous pencher sur les principaux codes correcteurs binaires et voir comment se déroule le codage et la correction d'erreurs, notamment pour les codes cycliques.

1 Théorie des codes correcteurs

1.1 Définition

On appellera une *lettre* la plus petite information transmissible et un *mot* comme un ensemble de lettres. L'ensemble des lettres est l'*alphabet*, noté Ω ; nous travaillerons avec $\Omega = \{0, 1\}$.

Ainsi, on considérera qu'un mot m est une suite de lettres appartenant au corps fini à deux éléments que l'on notera $\mathbb{F}_2$ (ou encore $\frac{\mathbb{Z}}{2\mathbb{Z}}$ ou $GF(2)$). Un mot de longueur n appartient donc à $\mathbb{F}_2^n$.

Introduisons quelques définitions afin de pouvoir travailler avec ces mots.

Définition 1.1.1. *On définit le* ***poids*** *d'un mot m de $\mathbb{F}_2^n$ comme le nombre de composantes non nulles du mot. On le note $w(m)$.*

Ainsi, si on note $\mathbf{0}$ le mot nul et $\mathbf{1}$ le mot plein (composé uniquement de 1), on a $w(\mathbf{0}) = 0$ et $w(\mathbf{1}) = n$.

Définition 1.1.2. *La distance de Hamming entre deux mots x, y de $\mathbb{F}_2^n$ est le nombre de composantes distinctes de x et y. On la note $d(x, y)$.*

Proposition 1.1.3. *La distance de Hamming est une distance.*

Démonstration. Montrons qu'il s'agit bien d'une distance. Il est clair que pour tout x, y de $\mathbb{F}_2^n$, on a,

$$d(x, y) = d(y, x); \quad d(x, y) \geq 0; \quad d(x, y) = 0 \iff x = y.$$

Vérifions l'inégalité triangulaire : Soient x, y, z dans $\mathbb{F}_2^n$; On note $x = x_1 \dots x_n$, $y = y_1 \dots y_n$ et $z = z_1 \dots z_n$. On définit les ensembles U, S et T suivants :

$$\begin{aligned} U &:= \{x_i \neq z_i\}, \\ S &:= \{(x_i \neq z_i) \wedge (x_i = y_i)\}, \\ T &:= \{(x_i \neq z_i) \wedge (x_i \neq y_i)\}. \end{aligned}$$

Ainsi, $S \cup T = U$ et $S \cap T = \emptyset$. Donc, $d(x, y) = |U| = |T| + |S|$ et $|T| \leqslant d(x, z)$ et $|S| \leqslant d(y, z)$. □

Puisque l'on est dans $\mathbb{F}_2^n$, $d(x, y) = w(x - y) = w(x + y)$.

Définition 1.1.4. *Une boule de Hamming de centre x et de rayon r est définie comme,*

$$B_H(x, r) = \{y \in \mathbb{F}_2^n \mid d(x, y) \leq r\}.$$

On peut maintenant s'intéresser aux codes.

1.2 Codes binaires

Définition 1.2.1. *Un* ***code binaire*** *de longueur n et de dimension k est une partie C de dimension k de $\mathbb{F}_2^n$; on dit que C est un code de paramètres (n, k).*

En associant à l'action d'encoder une application injective φ (l'injectivité assure que le décodage soit possible), on a $C = \mathrm{Im}(\varphi)$.

Un exemple particulièrement simple est le bit de parité : on ajoute à la fin du message un bit correspondant à la parité de la somme des éléments du message. Si la somme est paire, le bit vaut 0 sinon il vaut 1. Pour un mot de longueur k, le code associé a donc pour paramètres $(k+1, k)$. Exemple :

$$0011010 \to 00110101$$

Lorsqu'il y a au plus une erreur, ce code permet de détecter s'il y a eu une erreur ou non ; mais il ne permet pas de trouver sa position et donc de la corriger. De plus, s'il y a un nombre pair d'erreurs, elles se compensent et sont donc indétectables. On peut pallier partiellement ce dernier problème en divisant le mot en plusieurs petits mots de longueur 2 que l'on encodera séparément.

$$0011010 \to 000110101$$

Ainsi, tant que les erreurs ne se produisent pas sur des bits appartenant à la même paire, on peut les détecter.

Une autre idée qui pourrait venir à l'esprit serait de répéter plusieurs fois chaque bit. En répétant n fois chaque bit d'un mot de longueur k, le code associé a pour paramètres (nk, k). Exemple : On répète deux fois chaque bit,

$$010 \to 000111000$$

Si une erreur se produit, il est possible de la localiser et de la corriger facilement :

$$000111000 \to 000111001 \quad \text{(erreur sur la dernière lettre).}$$

On constate que le dernier trio de bits n'est pas composé de lettres identiques, donc l'erreur provient probablement de la lettre différentes des deux autres. Malheureusement, ce code reste lui aussi assez limité : s'il y a deux erreurs dans le même trio de bits, cela donnera lieu à une mauvaise correction ; s'il y en a 3, on ne peut même pas détecter l'erreur. On constate qu'en répétant chaque bit n fois, le code permet de retrouver le bon mot tant qu'il y a au plus $\lfloor \frac{n-1}{2} \rfloor$ erreurs ($\lfloor \cdot \rfloor$ désignant la partie entière). Cela peut sembler correct, mais la quantité d'informations ajoutée est particulièrement élevée.

Définition 1.2.2. *La distance minimale d_C d'un code C est définie comme,*

$$d_C = \min\{d(x,y) | x \in C, y \in C, x \neq y\}.$$

Définition 1.2.3. *La capacité de correction e_C d'un code est le plus grand entier tel qu'il soit toujours possible de corriger e_C erreurs ou moins.*

Un mot erroné peut donc être corrigé s'il existe un unique mot du code le plus proche ; de ce fait, les boules de Hamming centrées en un mot du code et de rayon e_C sont disjointes. Il en découle que $e_C = \lfloor \frac{d_C - 1}{2} \rfloor$ (le -1 permet de lever l'incertitude lorsqu'il y a $\frac{d_C}{2}$ erreurs).

La capacité de détection d'erreurs d'un code est naturellement plus élevée : on peut détecter toute erreur e vérifiant $w(e) < d_C$. En effet, si le mot du code m est perturbé par l'erreur e, $m' = m + e \notin C$ car sinon $e = m - m' \in C$ ce qui contredit la définition de d_C.

Définition 1.2.4. *Le taux d'information d'un code C de paramètres (n,k) est le rapport $\frac{k}{n}$.*

On veut maximiser ce taux d'informations (qu'il soit le plus proche possible de 1) afin de ne pas rajouter des données inutilement ; le taux d'information d'un code de répétition vaut $\frac{1}{n}$ ce qui est loin d'être désirable.

Intéressons nous à un type de code particulier qui offre des algorithmes de décodage efficaces : les codes linéaires.

1.3 Codes linéaires

Définition 1.3.1. *Un code C de paramètres (n,k) est dit* linéaire *si pour tout m, m' de C, $m + m'$ est dans C. C est un sous-espace vectoriel de dimension k de $\mathbb{F}_2^n$*

Cette nouvelle condition permet d'établir un lien entre le poids et la distance minimale :

Proposition 1.3.2. *Soit C un code linéaire,*

$$d_C = \min\{w(m) | m \in C, m \neq 0\}.$$

Démonstration. Soient x, y dans C tels que $d_C = d(x,y)$. On sait que $d(x,y) = w(x+y)$. Comme C est linéaire, il existe m dans C tel que $x+y = m$ donc $w(x+y) = w(m)$.

□

Désormais, nous inclurons la distance minimale du code dans ses paramètres ; un code C a pour paramètres (n, k, d_C). Comment déterminer la distance minimale d'un code ? On pourrait calculer le poids de tous les éléments mais ce serait laborieux. Il est possible d'obtenir une majoration de d_C en fonction de n et de k.

Proposition 1.3.3. *Soit C un code linéaire de paramètres (n, k, d_C). Alors,*

$$d_C \leq n - k + 1.$$

C'est la borne de Singleton.

Démonstration. Il suffit de montrer qu'il existe un mot du code dont les $k-1$ dernières composantes sont nulles. Soit E le sous-espace vectoriel de $\mathbb{F}_2^n$ dont les $k-1$ dernières composantes sont nulles, on a $\dim(E) = n - (k-1) = n - k + 1$. Donc, $\dim(C) + \dim(E) = n + 1 > n$ donc $C \cap E \neq \{\emptyset\}$. Il existe bien un mot du code dont les $k-1$ dernières composantes sont nulles ce qui prouve que $d_C \leq n - k + 1$.

□

1.4 Codage

Soit C un code linéaire de paramètres (n, k, d_C). Il existe φ une application linéaire injective de ${\mathbb{F}_2}^k$ dans ${\mathbb{F}_2}^n$ telle que $Im(\varphi) = C$. On peut le représenter matriciellement par $\varphi(x) = xG$; G étant la transposée de la matrice représentative de φ par rapport aux bases canoniques de ${\mathbb{F}_2}^k$ et de ${\mathbb{F}_2}^n$. Les mots sont donc considérés comme des vecteurs lignes. On dit que G est la matrice génératrice de C et que C est le code engendré par G.

Exemple : $x = (101)$, $G = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$

$$\varphi(x) = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} = (1010)$$

Il s'agit du codage par bit de parité vu précédemment.

Définition 1.4.1. *Un codeC linéaire de paramètres (n, k, d_C) est dit **systématique** si l'encodage consiste à rajouter $n - k$ bits à la fin du mot. Pour un code linéaire, sa matrice génératrice G est de la forme $(I_k|B)$, I_k étant la matrice unité à k lignes et k colonnes et B une matrice à k lignes et $n - k$ colonnes.*

Travailler avec un code systématique nous simplifiera quelques calculs par la suite.

1.5 Décodage

Définition 1.5.1. *On appelle* ***matrice de contrôle*** *d'un code linéaire C de paramètres (n, k, d_C) la matrice $H \in M_{n-k,n}$ à coefficients dans $\mathbb{F}_2$ telle que $ker(H) = C$. Ainsi, $\forall m \in C, H^t m = 0$ et $H^t G = 0$.*

Proposition 1.5.2. *: S'il existe dans C un mot de poids r, il existe r colonnes de H linéairement dépendantes.*

Démonstration. Soit $m = m_0 m_1 ... m_{n-1}$ un mot du code de poids r, on note C_i la i-ème colonne de H et $m_{i_1}, m_{i_2}, ... m_{i_r}$ les composantes non nulles de m.

$0 = H^t m = H \sum_{i=0}^{n-1} \begin{pmatrix} 0 \\ \vdots \\ m_i \\ \vdots \\ 0 \end{pmatrix} = H \sum_{p=1}^{r} \begin{pmatrix} 0 \\ \vdots \\ m_{i_p} \\ \vdots \\ 0 \end{pmatrix} = \sum_{p=1}^{r} m_{i_p} C_{i_p}$ donc il existe bien r colonnes de H étant linéairement dépendantes. □

Proposition 1.5.3. *S'il existe r colonnes linéairement dépendantes de H, alors il existe un mot du code de poids $r' \leq r$.*

Démonstration. Il existe r scalaires non tous nuls tels que $\sum_{p}^{r} m_{i_p} C_{i_p} = 0$. Soit $m \in F_2^n$ dont les composantes de rang $i_1, i_2, ... i_r$ sont $m_{i_1}, m_{i_2}, ... m_{i_r}$ et les autres sont nulles. On a $w(m) \leq r$. □

On en déduit que la distance minimale d_C (c'est-à-dire le poids minimal) est égale au nombre minimum de colonnes linéairement dépendantes de H.

Définition 1.5.4. *Soit $m \in {\mathbb{F}_2}^n$, on appelle* ***syndrome*** *s du mot m le vecteur $s = H^t m$.*

Supposons que w soit un mot erroné, $w = m + e$ avec m un mot du code e l'erreur. Ainsi, $H^t w = H^t(m+e) = H^t e$; le mot erroné et l'erreur ont donc le même syndrome s. L'ensemble des mots de syndrome s est appelé *classe latérale* (ou *coset*) de w. Corriger w revient donc à trouver le mot dont le poids $w(m)$ vérifie $w(m) \leq e_C$ dans la classe latérale de w ; cela n'est possible que si ce mot est unique, on suppose donc qu'il y a moins d'erreurs que la capacité de correction du code.

Proposition 1.5.5. *Chaque syndrome s est associé à un unique mot $m \in {\mathbb{F}_2}^n$ dont le poids $w(m)$ vérifie $w(m) \leq e_C$.*

Démonstration. Supposons qu'il existe $m' \in {\mathbb{F}_2}^n$ tel que $w(m') \leq e_C$ et $H^t m = H^t m'$ donc $H^t(m - m') = 0$ et donc $m - m'$ est un mot du code. $w(m-m') \leq w(m)+w(m') \leq 2e_C \leq d_C - 1$. Le seul mot du code x qui vérifie $w(x) \leq d_C - 1$ est le mot nul donc $m - m = 0$ et $m = m'$. □

Il suffit maintenant de dresser une liste des syndromes de chaque élément de ${\mathbb{F}_2}^n$ pour pouvoir retrouver l'erreur et à fortiori le mot codé.

S'il n'y a qu'une erreur : soit e_i le mot contenant un 1 à la i-ème place et des 0 ailleurs. Alors, il existe i tel que $w = m + e_i$. Le syndrome de e_i est donc la i-ème colonne de H, on peut donc facilement localiser l'erreur.

Proposition 1.5.6. *Si C est un code systématique et $G = (I_k|B)$, alors $H = (-{}^tB|I_{n-k})$.*

Démonstration. $H^t G = {}^t B - {}^t B = 0$. □

Exemple : Prenons G = $\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$.

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

Codons le mot $m = (01)$, $w = (01)\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} = (0111)$.

Ajoutons lui d'abord une erreur en troisième position : $z = (0101)$.

Calculons le syndrome de z : $s = H^t z = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

Cela correspond à la troisième colonne de H, l'erreur provient donc du troisième bit.

1.6 Codes parfaits

Intéressons-nous à C en tant qu'espace vectoriel.

Proposition 1.6.1. $\#B_H(x, r) = \sum_{i=0}^{r} C_n^i$.

Démonstration. Pour tout $i \in \{0, 1, ..., r\}$, il existe C_n^i mots $y \in {\mathbb{F}_2}^n$ tels que $d(x, y) = i$. □

Proposition 1.6.2. *Inégalité de Hamming :* $\sum_{i=0}^{e_c} C_n^i \leq 2^{n-k}$.

Démonstration. Les boules $B_H(m, e_C)$ centrées en les mots du codes de rayon e_C sont deux à deux disjointes donc :

$$\sum_{m \in C} \#B_H(m, e_C) = \sum_{m \in C} \sum_{i=0}^{e_C} C_n^i \leq |\{0,1\}^n| = 2^n.$$

Il y a 2^k boules de Hamming dans le code donc :

$$2^k \sum_{i=0}^{e_C} C_n^i \leq 2^n \text{ et donc } \sum_{i=0}^{e_C} C_n^i \leq 2^{n-k}.$$

□

La situation optimale serait donc que les boules de Hamming forment une partition de ${\mathbb{F}_2}^n$, ce qui est le cas lorsque l'inégalité de Hamming est une égalité.

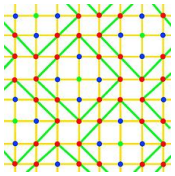

FIGURE 1 – Situation où chaque mot est associé à un unique mot du code.

Définition 1.6.3. *Un code C de paramètres (n, k, d_C) est dit* ***parfait*** *lorsque les boules de Hamming centrées en les mots du code de rayon e_C forment une partition de ${\mathbb{F}_2}^n$.*

Ainsi, $\sum_{i=0}^{e_C} C_n^i = 2^{n-k}$ et $\forall\ m \in {\mathbb{F}_2}^n$, il existe m qui minimise $d(r, m)$. On remarquera que d_C est forcément impair.

2 Code de Hamming

Créons un code qui satisfait l'égalité de Hamming et qui soit capable de corriger une erreur ; on prend donc la distance minimale la plus petite possible, $d_C = 3$.

$$\text{Posons } r = n - k.$$
$$\sum_{i=0}^{1} C_n^i = 2^{n-k} \Rightarrow 1 + n = 2^k$$
$$n = 2^r - 1 \text{ et } k = 2^r - r - 1.$$

Les codes ayant ces paramètres sont appelés codes de Hamming.

Définition 2.0.1. *Un* ***code de Hamming*** *est un code de paramètres* $(2^r - 1, 2^r - r - 1, d_C)$ *avec* $r \geq 2$ *; c'est un code parfait.*

Les codes parfaits ayant une capacité de correction 1 sont donc nécessairement des codes de Hamming. Pour $r = 2$, le code parfait de paramètres $(3, 1, 3)$ est le code de répétition.

Exemple : Étudions le cas $r = 3$.
Considérons un code linéaire systématique $C(7, 4, 3)$. Soit $x \in \mathbb{F}_2^4$, $x = d_1d_2d_3d_4$ et soit $\varphi : {\mathbb{F}_2}^4 \to {\mathbb{F}_2}^7$ telle que $\varphi(x) = d_1d_2d_3d_4p_1p_2p_3$ avec $p_1 = d_1 + d_2 + d_4$, $p_2 = d_1 + d_3 + d_4$ et $p_3 = d_2 + d_3 + d_4$. On obtient la matrice génératrice

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

Essayons d'encoder un mot, de modifier un de ses bits et de le décoder. Prenons $m = (1010)$.

$$mG = (1010) \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = (1010101).$$

Modifions le 4ème bit $(1010101) \longrightarrow (1011101)$ Puisque le code est systématique, on peut rapidement trouver la matrice de contrôle H.

$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$ On a bien $H^tG = 0$.

Calculons le syndrome du message erroné :

$$(1011101)^t H = (1011101) \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (111).$$

On obtient trois 1, il y a donc une erreur sur les trois bits de parité ; l'erreur provient donc du bit qui intervient dans la construction des trois bits de parité : le quatrième. On aurait bien sûr reprendre l'algorithme vu précédemment et retrouver le même résultat.

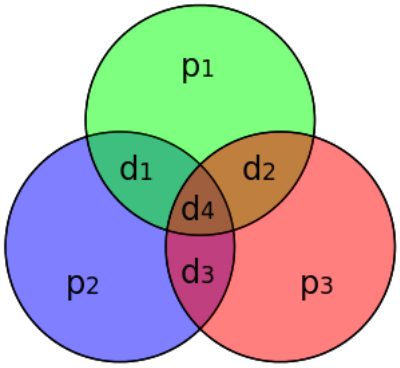

FIGURE 2 – Illustration montrant comment repérer facilement la provenance d'une erreur.

Ainsi, ce code est plus efficace qu'un simple code linéaire. Voilà une comparaison en image (avec un taux d'erreur de 0.01) :

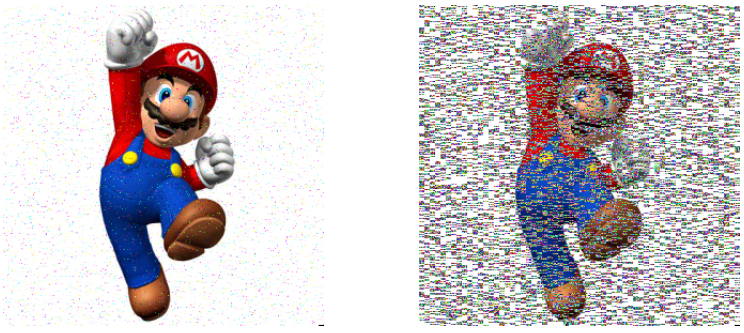

FIGURE 3 – Code de Hamming/ Code linéaire généré aléatoirement

Ces codes ont été réalisés sur Cocalc `https://cocalc.com/` (ancienne-

ment nommé Sage) et sont disponibles dans l'annexe. J'ai utilisé le Jupiter Notebook et le kernel le plus récent (SageMath Dev). Cocalc permet de travailler facilement avec plusieurs types de codes correcteurs (linéaires, de Hamming, cycliques) et permet de simuler l'apparition d'erreurs.

3 Codes cycliques

3.1 Fonctionnement des codes cycliques

Définition 3.1.1. *Un code linéaire est dit* ***cyclique*** *s'il est stable par décalage circulaire.*

Plus formellement, soit C un code de paramètres (n, k, d_C) et $m \in {\mathbb{F}_2}^n$, $m = m_0 m_1 ... m_{n-1}$. On note $\sigma(m) = m_{n-1} m_0 ... m_{n-2}$.
Alors, C est cyclique si $\forall m \in C, \sigma(m) \in C$.

Ainsi, $\forall k \in \mathbb{N}$, $\sigma^k(m) \in C$ et $\sigma^n(m) = m$.

Exemple Le code $C = \{000, 110, 011, 101\}$ est cyclique.

Définition 3.1.2. *Soit $m = m_0 m_1 ... m_{n-1}$ un mot, $m(X) = m_0 + m_1 X + ... + m_{n-1} X^{n-1}$ est la représentation polynomiale associée à m. On notera P_C la représentation polynômiale d'un code (on pourra aussi utiliser C s'il n'y a pas de risque de confusion).*

En reprenant le code précédent, on a $P_C = \{0, 1 + X, X + X^2, 1 + X^2\}$.

Regardons à quoi correspond un décalage circulaire vers la droite en termes de polynôme.

$$\begin{aligned} \sigma(m)(X) &= m_{n-1} + m_0 X + ... + m_{n-2} X^{n-1} \\ &= X m(X) + m_{n-1}(1 - X^n). \end{aligned}$$

Donc, $\sigma(m)(X) = X(m)X \qquad (\text{mod } X^n - 1)$.
C'est ce qui nous motive à travailler dans l'anneau des polynômes à coefficients dans $\mathbb{F}_2$ modulo $(X^n - 1)$: $\frac{\mathbb{F}_2[X]}{X^n - 1}$.

Rappelons la définition d'un idéal :

Définition 3.1.3. *Un sous-ensemble non-vide I d'un anneau commutatif A est un* ***idéal*** *si $\forall a, b \in I, a + b \in I$ et $\forall x \in A, x \cdot a \in I$.*

Proposition 3.1.4. *C est un code cyclique si P_C est un idéal de $\frac{\mathbb{F}_2[X]}{X^n - 1}$.*

Démonstration. Si P_C est un idéal de $\frac{\mathbb{F}_2[X]}{X^n - 1}$ alors $\forall m \in C, Xm(X) \in P_C$ et$Xm(X) = \sigma(m)(X) \Rightarrow \sigma(m) \in C$.
Inversement, si C est un code cyclique, $\sigma^k(m) \in C \forall k \in \mathbb{N}$ donc $X^k m(X) \in P_C$. Puisque C est un code linéaire, $P(X)m(X) \in P_C \ \ \forall P[X] \in \frac{\mathbb{F}_2[X]}{X^n - 1}$ donc P_C est un idéal. □

Définition 3.1.5. *Le* ***polynôme générateur*** $g(X)$ *d'un code cyclique* C *est le polynôme non nul unitaire de plus bas degré de* C.

Proposition 3.1.6. *Le polynôme générateur est unique.*

Démonstration. Supposons que g_1 et g_2 soient deux polynômes générateurs distincts et différents de 0 de C. Alors, g_1 - g_2 est un polynôme de C de degré strictement inférieur à celui de g_1 et g_2, ce qui entraîne une contradiction. □

Proposition 3.1.7. *Tout mot du code est un multiple de* g.

Démonstration. Soit $m \in C$. Par division euclidienne, on a :
$m(X) = q(X)g(X)+r(X)$ avec $q(X), r(X) \in F_2[X]$ et $deg(r(X)) < deg(g(X))$.
$r(X) = q(X)g(X) - m(X)$ donc r est un mot du code d'après les propriétés précédentes (multiple de $g(X)$ et linéarité). Si $r(X) \neq 0$, cela contredit l'hypothèse que $g(X)$ soit le polynôme générateur. Donc, $r(X) = 0$ et $m(X) = q(X)g(X)$. □

Proposition 3.1.8. $g(X)$ *divise* $X^n - 1$.

Démonstration. On sait que $X^k g(X) = g^k(X)$ où $g^k(X)$ est le polynôme obtenu en décalant les composantes de $g(X)k$ fois vers la droite, c'est donc un mot du code. Ce qui implique qu'il existe $a(X)$ tel que $X^k g(X) = a(X)g(X)$ et donc $(X^k + a(X))g(X) = X^n - 1$. □

Définition 3.1.9. *Un idéal* I *de* A *est dit* ***principal*** *s'il existe* $i \in I$ *tel que* $I = \{i * a | a \in A\}$.

C est donc un *idéal principal* de $\frac{\mathbb{F}_2[\mathbf{X}]}{\mathbf{X}^n-1}$ et $C = \{m(X)g(X) | m(X) \in \frac{\mathbb{F}_2[\mathbf{X}]}{\mathbf{X}^n-1}\}$.

Proposition 3.1.10. $dim(C) = n - deg(g(X))$ *et* $(g(X), Xg(X), ..., X^{n-1-deg(g(X))}g(X))$ *est une base de* C.

Démonstration. Soit $m \in C, deg(m(X)) \leq n - 1$ par définition. Si$m(X) = q(X)g(X)$ alors $deg(q(X)) \leq n - 1 - deg(g(X))$. Cela montre que $m(X)$ est une combinaison linéaire des polynômes $g(X), Xg(X), ..., X^{n-1-deg(g(X))}g(X))$. Donc, $dim(C) = n - deg(g(X))$ et comme chaque combinaison linéaire est de degré au plus $n - 1$, ces polynômes forment une famille libre de $\frac{\mathbb{F}_2[X]}{X^n-1}.(X, Xg(X), ..., X^{n-1-deg(g(X))})$ est bien une base de C. □

De ce fait, on en déduit que la matrice génératrice G liée à C est de la forme $\begin{pmatrix} a_0 & a_1 & \cdots & a_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & a_0 & a_1 & \cdots & a_{n-k} & 0 & \cdots & 0 \\ & & & \vdots & & & & \\ 0 & \cdots & \cdots & 0 & a_0 & a_1 & \cdots & a_{n-k} \end{pmatrix} \in M_{k,n}(\mathbb{F}_2)$, $\deg(g(X)) =$

$n-k$.

Proposition 3.1.11. *Soit C un code cyclique et $g(X)$ son polynôme générateur de degré r, $g(X) = a_0 + a_1X + ...a_{r-1}X^{r-1} + X^r$. Alors, $a_0 = 1$.*

Démonstration. Supposons que $a_0 = 0$.
$g(X) = a_1X + ...a_{n-1}X^{n-1} + X^r$
$g(X) = X(a_1 + ...a_{n-1}X^{n-2} + X^{r-1})$
Si on décale $g(X)n-1$ fois vers la droite (ou 1 fois vers la gauche), on obtient le polynôme ci-dessus de degré $< r$ ce qui génère une contradiction. □

Définition 3.1.12. *On appelle **polynôme de contrôle** le polynôme $h(X)$ qui vérifie $g(X)h(X) = X^n - 1$.*

Son existence est garantie par $g(X) \mid X^n - 1$.
En outre, si $m \in C$ et $m(X) = q(X)g(X)$,

$$\begin{aligned} h(X)m(X) &= h(X)g(X)q(X) \\ &= (X^n - 1)q(X) \\ &= 0 \text{ dans } \tfrac{\mathbb{F}_2[\mathbf{X}]}{\mathbf{X}^n-1}. \end{aligned}$$

$h(X)$ remplit bien un rôle similaire à celui de la matrice de contrôle ; on considérera d'ailleurs la matrice de contrôle H associée. Si $h(X) = b_0+b_1X+ ... + b_kX^k$, $H = \begin{pmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & h_k & \cdots & h_1 & h_0 & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & 0 & \cdots & 0 & h_k & \cdots & h_0 \end{pmatrix} \in M_{n-k,n}(\mathbb{F}_2)$.

On a bien $H^tG = 0$.

3.2 Recherche des codes cycliques

Soit C un code cyclique. Puisque le polynôme générateur divise $X^n - 1$, rechercher les polynômes générateurs possibles pour un code de longueur n revient à trouver les facteurs de $X^n - 1$ à coefficients dans $\mathbb{F}_2$.

Soit α une racine primitive n-ième de l'unité, on a $X^n - 1 = \prod_{i=0}^{n-1}(X - \alpha^i)$ dans $\mathbb{C}[X]$.

Si $g(X)$ est un polynôme générateur, il est de la forme $g(X) = \prod_{i\in\Sigma}(X-\alpha^i)$ où Σ est une partie convenable de $\mathbb{F}_n$.

Proposition 3.2.1. *Les coefficients de* $g(X) = \prod_{i\in\Sigma}(X-\alpha^i)$ *sont dans* $\mathbb{F}_2$ *ssi* Σ *est stable par multiplication par* 2 *modulo* n.

Démonstration. Si g est à coefficients dans $\mathbb{F}_2$, chaque coefficient de g est stable par élévation au carré. Comme cette opération respecte aussi l'addition, on voit que $g(X^2) = (g(X))^2$. Ainsi, l'ensemble des racines de g est stable par élévation au carré, ce qui signifie que Σ est stable par multiplication par 2. Inversement, si Σ est stable par multiplication par 2 :

$g(X^2) = \prod_{i\in\Sigma}(X^2-\alpha^i) = \prod_{i\in\Sigma}(X^2-\alpha^{2i}) = \prod_{i\in\Sigma}(X-\alpha^i)^2 = (g(X))^2$
et donc les coefficients de g sont dans $\mathbb{F}_2$. □

La recherche de polynômes générateurs revient donc à trouver les parties stables par multiplication par 2 de F_n. Soit $j \in \mathbb{F}_n$, on note Σ_j la plus petite partie stable contenant j. Alors, il existe un plus petit entier $s > 0$ tel que $2^s j \equiv j[n]$, on peut ainsi prendre Σ_j=$\{j, 2j, ..., 2^{s-1}j\}$ qui est bien stable par multiplication par 2 modulo n.

Définition 3.2.2. *La* ***classe cyclotomique*** *de* j *(relative à* 2 *modulo* n*) est* Σ_j=$\{j2^k \ mod\ n | k \in \mathbb{N}\}$.

Le degré du polynôme irréductible associé $g_j = \prod_{i\in\Sigma_j}(X-\alpha^i)$ est égal à $\text{card}(\Sigma_j) = s$.

Cherchons les codes cycliques de longueur 7 afin d'illustrer tout cela.

Exemple : $2^s j \equiv j[7]$. Pour $j = 1$, $s = 3$ convient.
$\Sigma_1 = \{1, 2, 4\}$. De même, pour $j = 3$, $s = 3$ convient à nouveau : $\Sigma_3 = \{3, 5, 6\}$
Enfin pour $j = 0$, $\Sigma_0 = \{0\}$.

$X^7 - 1$ peut donc être décomposé comme produit de 3 facteurs irréductibles :

$$\begin{cases} g_0 = X - 1 \\ g_1 = (X-\alpha)(X-\alpha^2)(X-\alpha^4) \\ g_3 = (X-\alpha^3)(X-\alpha^5)(X-\alpha^6) \end{cases}$$

α étant une racine primitive 7-ième de l'unité.

On peut tester manuellement les polynômes de degré 3 se terminant par 1 (sinon 0 est une racine) à coefficients dans $\mathbb{F}_2$ qui conviennent : X^3+X^2+1 ; X^3+X+1 ; X^3+1 et X^3+X^2+X+1.
Les deux derniers admettent une racine en 1 et $X^3+1=(X-1)(X^2+X+1)$ et $X^3+X^2+X+1=(X-1)^3$, il n'y a donc que deux choix possibles.
Supposons que α soit racine primitive de $P(X)=X^3+X+1$, on obtient plusieurs informations concernant α :
$\alpha^3=\alpha+1$
$\alpha^4=\alpha^2+\alpha$
$\alpha^5=\alpha^4+1$
$\alpha^6=\alpha^2+1$

$$\begin{aligned} g_1 &= (X-\alpha)(X-\alpha^2)(X-\alpha^4) \\ &= X^3+X^2(\alpha+\alpha^2+\alpha^4)+X(\alpha^3+\alpha^5+\alpha^6) \\ &= X^3+X+1. \end{aligned}$$

De même, on trouve que $g_3=X^3+X^2+1$ et $X^7-1=(X-1)(X^3+X+1)(X^3+X^2+1)$.

On obtient ainsi $2^3=8$ codes cycliques différents :

$$\begin{cases} C_0=\langle X^7+1\rangle & \text{Code nul.} \\ C_1=\langle X+1\rangle \\ C_2=\langle X^3+X+1\rangle \\ C_3=\langle X^3+X^2+1\rangle \\ C_4=\langle (X^3+X+1)(X+1)\rangle=\langle X^4+X^3+X^2+1\rangle \\ C_5=\langle (X^3+X^2+1)(X+1)\rangle=\langle X^4+X^2+X+1\rangle \\ C_6=\langle (X^3+X+1)(X^3+X^2+1)\rangle=\langle X^6+X^5+X^4+X^3+X^2+X+1\rangle \\ C_7=\langle 1\rangle & \text{Ne change pas le mot.} \end{cases}$$

Les code C_2 et C_3 ont pour paramètres $(7,4,1)$, ce sont des codes de Hamming. Le code C_6 correspond au bit de répétition.

On peut déterminer les polynômes générateurs d'une autre façon :

Définition 3.2.3. *On appelle* ***polynôme cyclotomique*** *le polynôme* Φ_n *défini par* $\Phi_n(X)=\prod_{\substack{k=1\\ k\wedge n=1}}^{n}(X-exp(\frac{2ki\pi}{n}))$.

Proposition 3.2.4. $X^n-1=\prod_{d|n}\Phi_d(X)$.

Démonstration. On sait que $X^n - 1 = \prod_{i=0}^{n-1}(X - \alpha^i)$. De plus, $\{1, ...n\} = \bigsqcup_{d|n}\{k \in \{1, ...n\} | k \wedge n = d\}$. □

$$X^n - 1 = \prod_{d|n} \prod_{\substack{k=1 \\ k\wedge n=1}}^{n} (X - \alpha^k) = \prod_{d|n} \prod_{\substack{k=1 \\ k\wedge(\frac{n}{d})=1}}^{\frac{n}{d}} (X - \alpha^{kd})$$
$$= \prod_{d|n} \prod_{\substack{k=1 \\ k\wedge d=1}}^{d} (X - \alpha^{\frac{nk}{d}}) = \prod_{d|n} \Phi_d(X).$$

La dernière égalité venant du fait que si α est une racine primitive n-ième de l'unité, $\alpha^{\frac{n}{d}}$ est une racine primitive d-ième de l'unité.

□

Proposition 3.2.5. *Les polynômes cyclotomiques sont à coefficients dans $\mathbb{Z}$.*

Démonstration. On montre qu'ils sont à coefficients entier par récurrence. C'est vrai pour $\phi_1(X) = X - 1$. On suppose que c'est vrai jusqu'au rang $n - 1$. On a $X^n - 1 = \prod_{d|n} \phi_d(X) = \phi_n(X)h(X)$ dans $\mathbb{C}[X]$ où $h(X)$ est un polynôme à coefficients entier par hypothèse de récurrence. En faisant la division euclidienne dans $\mathbb{Z}$[X] de $X^n - 1$ par $h(X)$ on a : $X^n - 1 = g(X)h(X) + r(X)$ avec $deg(r) < deg(h)$ et $g(X)$ dans $\mathbb{Z}[X]$. Par unicité de la division euclidienne dans $\mathbb{C}[X]$, on a $\phi_n = g$ et donc ϕ_n est à coefficient entier. □

Le calcul des polynômes cyclotomiques permet donc aussi de retrouver les polynômes générateurs.

Nous allons désormais introduire une minoration de la distance minimale d'un code cyclique.

Proposition 3.2.6. *S'il existe des entiers a et $s > 0$ tels que $\sum$ contienne $a + 1, a + 2, ..., a + s$, la distance minimale du code construit est $\geq s + 1$.*

Démonstration. Il faut montrer qu'un polynôme $R \in \mathbb{F}_2[X]$ de degré $< n$ qui a au plus s coefficients non nuls (et donc de poids $< s+1$) est identiquement nul. Cela revient donc à montrer le lemme suivant : □

Lemme Soient $d_1, ...d_s$ des entiers avec $0 \leq d_1 < \cdots < d_s < n$, et $\lambda_1, ..., \lambda_s$ des éléments de F_2. Posons $R(X) = \sum \lambda_j X^{d_j}$. Si $R(\alpha^i) = 0$ pour $i = a + 1$,...,$a + s$, alors tous les λ_j sont nuls.

Démonstration. Le déterminant des α^{id_j} est un déterminant de Vandermonde. Mais comme α est une racine primitive n-ième de l'unité, les α^{d_j} sont deux à deux distincts. □

Pour le code cyclique de longueur 7 vu précédemment, $\sum$ contient $\{1, 2\}$ (ou $\{5, 6\}$), la distance minimale de ce code est donc ≥ 3.

Les différents codes cycliques désormais déterminés, on peut passer à l'étape du codage : il suffit de multiplier la représentation polynomiale d'un mot par le polynôme générateur modulo $X^n - 1$.

Exemple : On prend $g(X) = X^3 + X + 1$ et $m(X) = 1 + X + X^2$.
$g(X)m(X) = (X^3 + X + 1)(1 + X + X^2) = 1 + X + X^5$.

3.3 Algorithme de décodage

De façon analogue aux codes linéaires, le décodage se fait par la détermination du syndrome.

Définition 3.3.1. *Le syndrome d'un mot $S(m)(X)$ est le reste de la division euclidienne de ce mot par le polynôme générateur modulo $X^n - 1$.*

Cette définition vérifie bien que si un mot appartient au code, son syndrome est nul.

Proposition 3.3.2. *Soit C un code cyclique de longueur n et de dimension k. Soit $w(X) = q(X)g(X) + s(X), s(X)$ étant le syndrome de $w(X)$. Alors, le syndrome de $Xw(X)$ est :*
$Xs(X)$ si $deg(s(X)) < n - k - 1$.
$Xs(X) - g(X)$ si $deg(s(X)) = n - k - 1$.

Démonstration. Si $\deg(s(X)) < n - k - 1$ alors $\deg(Xs(X)) < n - k = deg(g(X))$ et on a bien $Xw(X) = Xq(X)g(X) + Xs(X)$ par unicité de la division euclidienne.
Si $\deg(s(X)) = n-k-1, s(X) = s_0+s_1X+...+X^{n-k-1}$. On note $\widehat{s}$(X)$=s_0+s_1X + ... + X^{n-k-2}$. De même, $g(X) = g_0 + g_1X + ... + X^{n-k}$ et $\widehat{g}(X) = g_0 + g_1X + ... + X^{n-k-1}$.

$$Xs(X) = X\widehat{s}(X)+X^{n-k} = Xs(X)+g(X)-\widehat{g}(X) = g(X)+(X\widehat{s}(X)-\widehat{g}(X))$$

et $\deg(X\widehat{s}(X) - \widehat{g}(X)) < n-k-1$. Comme $X\widehat{s}(X) = X(X) - X^{n-k}$ et $\widehat{g}(X) = g(X) - X^{n-k}$, le syndrome de $w(X)$ vaut bien $Xs(X) - g(X)$. $\square$

Soit C un code cyclique de paramètres(n, k, d_C), g(X) son polynôme générateur et w un mot erroné. Ainsi, $w(X) = m(X) + e(X)$ où m est un mot du code et e l'erreur associée à w. On rappelle que le mot peut être corrigé si $w(e) \leq e_C = \lfloor \frac{d_C - 1}{2} \rfloor$, w désignant le poids du mot (le nombre de composantes non nulles). L'objectif du décodage est donc de déterminer le polynôme $e(X)$.

Déterminons la forme de l'erreur. Soit $e' = ({}^t s, \mathbf{0})$ où $\mathbf{0}$ désigne le vecteur à k 0 et s le syndrome de w. Le degré de $e'(X)$ étant strictement inférieur à celui du polynôme générateur par définition, son syndrome est le même que celui de e. De plus, si on suppose que $w(s) \leq e_C$ alors $w(e') \leq e_C$ cela implique que $e' = e$ car il n'existe qu'un seul vecteur de poids $\leq e_C$ associé à chaque syndrome. L'erreur est donc de la forme $({}^t s, \mathbf{0})$.

On veut maintenant se ramener au cas où $w(s) \leq e_C$ afin de pouvoir déterminer l'erreur. Supposons que e possède k 0 à la suite et que $w(e) \leq e_C$. Alors, il existe $i \in \{0, 1, .., n-1\}$ tel que les k 0 de $\sigma^i(e)$ soient situés à la fin du mot. Ainsi, $\sigma^i(e)$ est de la forme$(\beta, \mathbf{0})$, β étant un vecteur de longueur n-k ; donc $\deg(\beta(\mathrm{X})) \leq n-k-1 = \deg(g(X)) - 1$. Le syndrome de $\beta(X)$ est donc $\beta(X)$ lui-même. Comme $\mathrm{w}(\beta(X)) \leq e_C, w(s_i(X)) \leq e_C$ où s_i(X) est le syndrome de $X^i e(X)$ modulo $X^n - 1$. D'après l'argument précédent, $e(X)X^i = ({}^t s_i, \mathbf{0})$ donc $e(X) = X^{n-i}({}^t s_i, \mathbf{0})$.

On peut donc expliciter l'algorithme de décodage :

1. On détermine le syndrome de $w(X)$ par division euclidienne.
2. On prend $i = 0$ et si $w(s_i(X)) \leq e_C, e(X) = X^{n-i}(s_i, \mathbf{0})$ et
$m(X) = w(X) - e(X)$.
3. Sinon, $i = i + 1$ et on calcule $s_i(X)$;on recommence jusqu'à ce que $i = n$.

Si $i = n$ la correction est impossible.

Exemple Soit C un code cyclique de paramètres $(15, 7, 5)$ de polynôme générateur $g(X) = X^8 + X^7 + X^6 + X^4 + 1$; sa capacité de correction est donc de 2.
Prenons $m(X) = 1 + X + X^3$ et $w(X) = m(X)g(X)$.
$w(X) = (1 + X + X^3)(1 + X^4 + X^6 + X^7 + X^8) = 1 + X + X^3 + X^4 + X^5 + X^6 + X^7 + X^{10} + X^{11}$.

Modifions deux de ses composantes, disons la quatrième et la huitième et notons ce nouveau polynôme $z(X) = 1 + X + X^3 + X^5 + X^6 + X^7 + X^8 + X^{10} + X^{11}$.
Par division euclidienne, on obtient $z(X) = (X^3 + X)(1 + X^4 + X^6 + X^7 + X^8) + X^7 + X^6 + 1$.
Le syndrome est donc $s(X) = X^7 + X^6 + 1$ de poids $3 > 2$.

$\deg(s(X)) = 7 = \deg(g(X)) - 1$ donc $s_1(X) = Xs(X) - g(X)$.

$s_1(X) = X^8 + X^7 + X - (X^8 + X^7 + X^6 + X^4 + 1) = X^6 + X^4 + X + 1$.
$s_2(X) = X(X^6 + X^4 + X + 1) = X^7 + X^5 + X^2 + X$.
$s_3(X) = X^8+X^6+X^3+X^2-(X^8+X^7+X^6+X^4+1) = X^7+X^4+X^3+X^2+1$.
$\vdots$
$s_{10}(X) = X^7 + X^6 + X^5$.
$s_{11}(X) = X^8 + X^7 + X^6 - (X^8 + X^7 + X^6 + X^4 + 1) = X^4 + 1$.

$w(s_{11}(X)) = 2$, c'est ce que l'on recherchait.
$e(X) = X^{n-11}s_{11}(X) = X^4(X^4 + 1) = X^8 + X^4$.

Les erreurs sont bien en position 4 et 8, là où on les avait placées. Il suffit ensuite de diviser le mot obtenu par le polynôme générateur afin de retrouver le mot initial.
Toutefois, l'algorithme de décodage n'aurait pas fonctionné si $e(X)$ ne comportait pas au moins k zéros à la suite (ce qui arrive forcément pour un code de longueur 15 et de dimension 7 lorsqu'il n'y a que deux erreurs).

Définition 3.3.3. *Un "**burst**" de longueur λ est un mot de $\mathbb{F}_2^n$ dont les composantes non nulles sont confinées dans λ positions consécutives.*

Exemple $m = (00100110000)$ est un "burst"de longueur 5 dans $\mathbb{F}_2^{11}$.

Un code est dit λ-burst-correcteur s'il peut corriger toutes les erreurs de burst$< \lambda$.

Proposition 3.3.4. *Tout code cyclique peut détecter les erreurs "burst" de longueur $\leq$ deg$(g(X))$.*

Démonstration. Montrons qu'il n'existe pas d'erreur "burst" de longueur $\leq \deg(g(X))$ appartenant au code.

Par définition, ce "burst" est de la forme $e(X) = X^i\beta(X)$ avec $\deg(\beta(X)) \leq \deg(g(X))$ et $\beta(X) \neq 0$. Le coefficient constant de $g(X)$ est 1 donc $g(X) \nmid X^i$. Si $e(X)$ est un mot du code alors $g(X) \nmid X^i\beta(X)$ donc $g(X) \mid \beta(X)$ mais $\deg(\beta(X)) \leq \deg(g(X))$ d'où la contradiction. □

Les codes cycliques permettent donc non seulement de corriger e_C erreurs mais aussi d'en détecter $\deg(g(X))$ pour peu qu'elles apparaissent au bon endroit ; c'est ce qui explique leur popularité ; ils sont notamment utilisés dans certains codes barres, dans les disques durs afin de corriger les erreurs provoquées par les rayures ou bien encore lors du vol spatial de *Voyager* 1 et 2

Conclusion

Nous avons pu voir quels étaient les mécanismes qui régissaient le codage et le décodage d'informations binaires, mais ce n'était qu'un bref aperçu du monde des codes correcteurs d'erreurs : les codes BCH , Reed-Solomon, de Golay sont tous d'autres codes fréquemment utilisés. De plus, nous ne sommes pas penchés sur le cas de l'effacement d'un bit qui aurait ajouté une nouvelle dimension à notre sujet.
La protection de l'information reste un secteur en pleine expansion dont l'avancement offre des possibilités insoupçonnées dans de multiples domaines et qui continue de donner naissance à de nouvelles théories.

4 Annexes

4.1 Bibliographie

Ce travail se base principalement sur le livre de Michel Demazure, Cassini, *Cours d'algèbre. Primalité. Divisibilité. Codes.*

Et quelques cours disponibles en ligne :

1. Massoud Malek , *Linear Cyclic Codes*, `http://www.mcs.csueastbay.edu/~malek/Class/Cyclic.pdf`
2. A.A. Pantchichkine, *Mathématiques des codes correcteurs d'erreurs* `https://www-fourier.ujf-grenoble.fr/~panchish/04cc`
3. Pierre Abbrugati, *Introduction aux codes correcteurs d'erreurs*, `http://www.lirmm.fr/~chaumont/download/cours/codescorrecteur/Cours_Pierre_Abbrugiati.pdf`
4. Pierre Wassef, *Cours d'arithmétique* `http://www.edu.upmc.fr/maths/wassef_LM220/documents/arithmetique.pdf`

4.2 Code SAGE

Hamming :

```
In [1]: from PIL import Image, ImageFilter;
        import scipy.ndimage
        import matplotlib.image
        import matplotlib.pyplot
        import numpy as np

In [3]: im = Image.open('Mario.png')

In [5]: data = list(im.getdata());

In [6]: data3 = np.array(data, dtype=np.uint8

In [7]: data[4]=[0]*90000

In [8]: data4=np.unpackbits(data3)

In [10]: data5=list(data4)

In [26]: len(data5)

Out[26]: 2160000
```

La première étape consiste à extraire l'information contenue dans l'image (de taille 300*300) qui est sous la forme de triplet de nombres compris entre 0 et 255, chaque triplet correspondant à la couleur d'un pixel. Il faut ensuite la convertir dans un format où elle pourra être encodée par un code binaire, chaque valeur est donc convertie en un vecteur de 8 bits. On obtient donc 8*3*300*300=2160000 éléments à coder que l'on place dans une liste

```
In [2]: C=codes.HammingCode(GF(2),7); C

Out[2]: [127, 120] Hamming Code over GF(2)

In [12]: Z=GF(2)^120
         Y=GF(2)^127

In [11]: datac=[0]*2160000*7
```

Ensuite, on crée un code de Hamming de paramètres (120,127) à l'aide du logiciel ; on encodera donc les bits par groupe de 120 (2160000/120=18000) et on obtiendra des mots de longueur 127. On prépare un tableau qui contiendra des vecteurs à 127 éléments qui appartiennent à $\mathbb{F}_2$.

```
In [13]: p=0.01;
         Chan=channels.QarySymmetricChannel(GF(2)^127,p);
```

On introduit 1% d'erreurs, ce qui est considérablement élevé mais met bien en évidence la différence d'efficacité entre les deux codes.

```
In [14]: for i in range(0,17999):
          P=Z(data5[120*i:120*(i+1)])
          X=C.encode(P)
          datac[127*i:127*(i+1)]=X
```

On peut maintenant passer à l'encodage des vecteurs.

```
In [15]: datae=[0]*2160000*7
```

```
In [16]: for i in range(0,17999):
          U=Y(datac[127*i:127*(i+1)])
          K=Chan.transmit(U)
          datae[127*i:127*(i+1)]=K
```

On ajoute des erreurs sur chaque mot de longueur 127.

```
In [17]: datam=[0]*2160000
```

```
In [18]: for i in range(0,17999):
          P=Y(datae[127*i:127*(i+1)])
          X=C.decode_to_message(P)
          datam[120*i:120*(i+1)]=X
```

La fonction de décodage présente dans le logiciel permet de retrouver nos vecteurs à 120 éléments pour peu qu'ils n'aient pas été trop modifiés.

```
In [19]: datap=np.array(datam,dtype=np.uint8)
```

```
In [20]: datar=np.packbits(datap)
```

```
In [21]: datar=datar.reshape(90000,3)
```

```
In [22]: datah=tuple(map(tuple,datar))
```

On reconvertit chaque vecteur de 8 éléments en un nombre entre 0 et 255 et on regroupe ces nombres par triplet (il y a donc 90000 triplets de nombres). On remet enfin tous ces triplets dans une liste.

```
In [23]: imf = Image.new("RGB",(300,300),"white")
```

```
In [24]: imf.putdata(datah)
```

On crée une image vierge et l'on y insère les données obtenues.

```
In [25]: imf
```

```
Out[25]:
```

On obtient une image quelque peu différente de la version originale mais parfaitement reconnaissable.

Code linéaire aléatoire :

```
In [1]: from PIL import Image, ImageFilter;
        import scipy.ndimage
        import matplotlib.image
        import matplotlib.pyplot
        import numpy as np

In [2]: C=codes.random_linear_code(GF(2),127,120); C
```

On utilise cette fois un code linéaire généré aléatoirement

```
Out[2]: [127, 120] linear code over GF(2)

In [3]: im = Image.open('Mario.png')

In [4]: data = list(im.getdata());

In [5]: data3 = np.array(data, dtype=np.uint8)

In [6]: data[4]=[0]*90000
        data4=np.unpackbits(data3)

In [7]: data5=list(data4)
        datac=[0]*2160000*7
        datae=[0]*2160000*7
        datam=[0]*2160000

In [8]: Z=GF(2)^120
        Y=GF(2)^127

In [9]: p=0.01;
        Chan=channels.QarySymmetricChannel(GF(2)^127,p);
        Chan

Out[9]: q-ary symmetric channel with error probability 0.0100000000000000, of in

In [10]: for i in range(0,17999):
           P=Z(data5[120*i:120*(i+1)])
           X=C.encode(P)
           datac[127*i:127*(i+1)]=X
```

```
In [11]: for i in range(0,17999):
             U=Y(datac[127*i:127*(i+1)])
             K=Chan.transmit(U)
             datae[127*i:127*(i+1)]=K

In [12]: for i in range(0,17999):
             P=Y(datae[127*i:127*(i+1)])
             X=C.decode_to_message(P)
             datam[120*i:120*(i+1)]=X

In [13]: datap=np.array(datam,dtype=np.uint8)
         datar=np.packbits(datap)
         datar=datar.reshape(90000,3)

In [ ]: datah=tuple(map(tuple,datar))

In [15]: imf = Image.new("RGB",(300,300),"white")
         imf.putdata(datah)

In [16]: imf

Out[16]:
```

On obtient une image qui bien que reconnaissable est d'une très mauvaise qualité.