La théorie des codes correcteurs et leur application à la transmission efficace d'informations

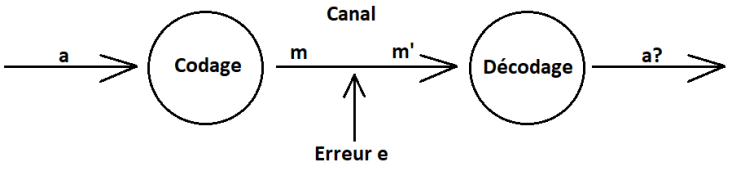
Monika Šostak / Auguste Paoli / Alexis Gadonneix

Contents

1 Introduction

L'omniprésence du numérique pour la transmission d'informations crée depuis quelques décennies l'apparition de nouveaux défis mathématiques liés à la correction des erreurs qui peuvent se produire durant ces transmissions.

Pour cela, la technique usuelle est de rajouter des éléments au message, qui ne modifient pas son contenu mais permettent de corriger une éventuelle perturbation. C'est le principe d'un code correcteur.



Ainsi, il s'agit de déterminer les propriétés sur le code en question pour que celui-ci soit performant, tant en théorie qu'en pratique.

Nous allons donc commencer par nous intéresser aux codes correcteurs globalement, en voyant les caractéristiques et définitions principales, ainsi qu'en étudiant l'exemple du code de Hamming. Ensuite, nous nous concentrerons sur la théorie des corps finis, qui sert de base à la contruction de codes correcteurs, et plus particulièrement celle des codes cycliques et des codes de résidus quadratiques. Enfin, nous aboutirons par une application de l'étude menée précédemment, le code de Golay.

2 Définitions et généralités

2.1 Définitions

Définition :

On définit un **alphabet** comme un ensemble A, fini non vide et composé d'éléments appelés **caractères**.

Définition :

Soit A et A' deux alphabets. Soit k et n deux entiers positifs.
On pose $E = A^k$ et $F = A'^n$
Une **application d'encodage** est une application injective

$$\varphi : E \longrightarrow F$$

Et $\varphi(E)$ est appelé **code**.

Enfin, n est appelé **longueur** du code.

De manière qualitative, il s'agit ici de transformer le message de longueur k, en un autre message de longueur n, qui sera le message transmis.
L'application φ doit donc être injective pour qu'à un mot corresponde un seul élément du code.

Dans la suite, cette application φ sera négligée, et c'est directement sur les éléments du code que l'on travaillera.

Définition :

On reprend les notations de la définitions précédente.
Le **taux d'information** d'un code est alors défini comme le rapport

$$\tau_x = \frac{k}{n}$$

Remarque :

On cherche en général à obtenir un taux d'information le plus grand possible : en effet, on cherche à éviter que la taille du message augmente trop, afin de ne pas avoir des messages trop grands à transmettre; ce qui augmente le risque d'erreurs et les rend trop lourds à transporter.

2.2 Introduction aux codes linéaires

Définition :

Un code est dit **linéaire** lorsque l'application φ d'encodage est linéaire.

Dans les faits, seuls les codes linéaires ont un intérêt mathématique. C'est donc ceux-ci que l'on traitera.

On pose pour la suite C un code linéaire.
Remarquons que puisque φ est linéaire, C est un sous-espace vectoriel de F.

Définition :

Soit m et m' deux mots de C.

On définit le **poids** de m, noté $\omega(m)$, par le nombre de caractères non nuls de m.

On définit la **distance de Hamming** entre m et m', notée $d(m, m')$, par :

$$d(m, m') = \omega(m - m').$$

C'est donc le nombre de différences entre les mots m et m'.

On peut alors définir la **distance minimale** d_{min} du code :

$$d_{min} = \min\{d(m, m'), (m, m') \in \mathrm{C}^2\}$$

On peut donc caractériser un code par trois paramètres (n, k, d),
où

- n est la longueur des mots de C,
- k est la dimension de C,
- d est la distance minimale du code.

Proposition 1.

Soit C un code linéaire. On pose d la distance minimale de C.
Alors :

- On peut détecter $d - 1$ erreurs.
- Soit t le plus grand entier strictement infèrieur à $\frac{d}{2}$.
  Alors C peut corriger t erreurs.

Démonstration :

- Soit m un mot du code et m' un mot obtenu après k erreurs sur m, avec $k \leq d-1$.
  Alors $d(m, m') = k < d$.
  On peut donc détecter que m' n'appartient pas au code.
- Soit m un mot du code et m' un mot obtenu après k erreurs sur m, avec $k \leq \frac{d}{2}$.
  Alors d'une part $d(m, m') < \frac{d}{2}$ et d'autre part, pour tout autre mot m_1 du code :

$$\begin{aligned} d(m_1, m') &= \omega(m_1 - m') \\ &= \omega(m_1 - m + m - m') \\ &\geq \omega(m_1 - m) - \omega(m - m') \\ &\geq d - \frac{d}{2} \\ &\geq \frac{d}{2} \end{aligned}$$

  On peut donc montrer que c'était le mot m à l'origine.

❑

Cela peut se représenter géométriquement en imaginant l'ensemble d'arrivée du code comme un ensemble de points, les mots du code étant au milieu de ces points.
La distance minimale se visualise alors comme des cercles autour des mots du code :

On peut le visualiser comme suit :

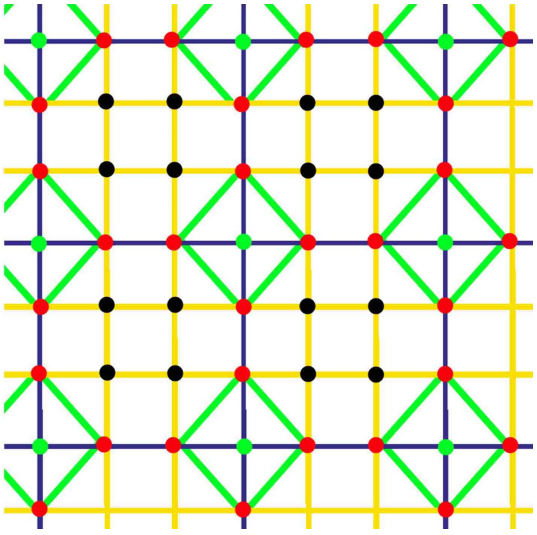


Figure 1: Représentation schématique d'un code correcteur

2.3 Codes parfaits et principe de décodage

Définition :

Un code C est dit **parfait** lorsqu'à tout élément de l'ensemble qui contient le code, on peut associer un et un seul mot du code.

Définition :

Soit C un code de paramètres (n,k,d), dans un ensemble à valeurs dans un alphabet à q caractères, qui peut corriger t erreurs.
La **borne de Hamming** est définie par:

$$\sum_{i=0}^{t} \binom{n}{i} (q-1)^i \leq q^{n-k}$$

La somme $\sum_{i=0}^{t} \binom{n}{i} (q-1)^i$ correspond au nombre de vecteurs erreurs distincts, dont le poids va de 0 à t.

Remarque : D'où vient cette borne ?

En fait, pour un mot donné, on peut lui appliquer $\sum_{i=0}^{t} \binom{n}{i} (q-1)^i$ erreurs correctibles (c'est simplement du dénombrement).

Donc on peut créer à partir des q^k mots du code, $q^k \times \sum_{i=0}^{t} \binom{n}{i} (q-1)^i$ mots qui peuvent être corrigés.

Ce nombre étant forcément inférieur au nombre de mots q^n de l'ensemble qui contient le code, on obtient alors notre borne.

Proposition 2.

Un code est parfait si et seulement si la borne de Hamming est atteinte.

Définition :

On pose $S_r(m) = \left\{m' \in \mathbb{F}_q^n / d(m, m') \leq r\right\}$

On dit alors que $S_r(m)$ est une **sphère de Hamming** de centre m et de rayon r.

Proposition 3.

Soit C un code t-correcteur, i.e. il corrige t erreurs. On note F l'ensemble qui contient le code. Alors C est un code parfait si et seulement si les sphères de Hamming de rayon t recouvrent F entièrement.

Ces deux propositions sont en fait deux manières, l'une mathématique et l'autre géométrique, de voir la même chose : si le code est parfait, alors tout mot de l'ensemble F qui contient le code peut s'écrire comme "un mot de C + une erreur correctible".
Les boules qu'on définissait précédemment sont donc disjointes (par définition de t) et en même temps elles recouvrent F entièrement : elles forment une sorte de **partition** de F.

Pour reprendre notre exemple visuel :

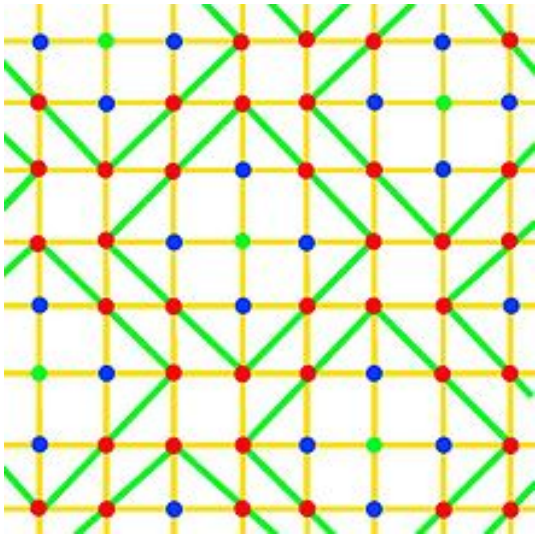


Figure 2: Illustration d'un code correcteur parfait

Remarque :
Les seuls codes parfaits binaires connus sont:

- les codes de répétition,
- les codes de Hamming,
- le code de Golay G_{23}.

Les codes de répétition, qui consistent simplement à répéter un grand nombre de fois le mot de départ, n'ont que très peu d'intérêt en raison de leur taux d'information très faible.
Quant aux deux autres, on les traitera ultérieurement dans notre étude.

Principe de décodage :

Explicitons un peu le principe de décodage dans ses grandes lignes.
On s'intéresse ici au cas du code parfait.

- Puisque le code est parfait, on peut pour chaque mot du message obtenu lui associer le mot d'origine.
- On obtient donc le message tel qu'avant sa transmission. Puis par injectivité de l'application d'encodage, on remonte alors au mot de départ.

Evidemment, tout cela est traité informatiquement, puisque ces opérations peuvent être réalisées de manière algorithmique, en traitant tous les cas d'erreurs possibles et en stockant les résultats obtenus

2.4 Un exemple : le code de Hamming (7,4,3)

On considère quatres vecteurs dans l'espace vectoriel $\mathbb{F}_2^7$:

$$e_0 = [1101000]$$
$$e_1 = [0110100]$$
$$e_2 = [0011010]$$
$$e_3 = [0001101]$$

les trois derniers étant obtenus par décalage vers la droite du précédant.

Soit $\psi : \mathbb{F}_2^4 \to \mathbb{F}_2^7$ une application linéaire injective, qui à un mot m=$[m_0m_1m_2m_3]$ associe le vecteur $m_0 * e_0 + m_1 * e_1 + m_2 * e_2 + m_3 * e_3$.

L'espace vectoriel $\mathbb{F}_2^4$ étant constitué de 16 éléments, on peut expliciter l'image de ψ :

$$\begin{array}{ll} \psi[0000]=[0000000] & \psi[1000]=[1101000] \\ \psi[0100]=[0110100] & \psi[0010]=[0011010] \\ \psi[0001]=[0001101] & \psi[1100]=[1011100] \\ \psi[1010]=[1110010] & \psi[1001]=[1100101] \\ \psi[0110]=[0101110] & \psi[0101]=[0111001] \\ \psi[0011]=[0010111] & \psi[1110]=[1000110] \\ \psi[1101]=[1010001] & \psi[1011]=[1111111] \\ \psi[0111]=[0100011] & \psi[1111]=[1001011] \end{array}$$

Définition :

Le code de Hamming est le sous-espace vectoriel Im(ψ), que l'on notera H, de $\mathbb{F}_2^7$.

A partir des caractéristiques de cette image, on peut retrouver les paramètres du code de Hamming.

Proposition 4.

Le code de Hamming a pour paramètres (7,4,3).

Démonstration :

- H est un sous-espace vectoriel de l'espace vectoriel $\mathbb{F}_2^7$, qui est constitué de mots de 7 bits. Donc la longueur du code de Hamming est 7.
- Chaque élément de H est une combinaison linéaire d'éléments de la famille (e_0, e_1, e_2, e_3), dont les coefficients appartiennent sont égaux à 0 ou 1. Donc (e_0, e_1, e_2, e_3) est une famille génératrice de H.
  De plus, c'est une famille libre.
  Donc (e_0, e_1, e_2, e_3) est une base de H.

  $$\dim(\mathrm{H})=\#(e_0, e_1, e_2, e_3)=4$$

  Donc k=4
- En parcourant les éléments de H, on peut voir que la plus petite distance entre deux éléments distincts de H est 3. Donc la distance minimale du code de Hamming est égale à 3.

❑

Proposition 5.

Le code de Hamming peut détecter 2 erreurs et en corriger 1 seule.
De plus, le taux d'information du code de Hamming est $\frac{4}{7}$.

Démonstration :

- La distance minimale du code de Hamming est égale à 3: $d_{min} = 3$.
  Or, le nombre d'erreurs que peut détecter un code est égal à $d_{min} - 1$.
  Donc le code de Hamming peut détecter 2 erreurs.
- Le nombre d'erreurs que peut corriger un code est le plus grand entier strictement inférieur à $\frac{d_{min}}{2}$, pour le code de Hamming, $\frac{d_{min}}{2}$=1,5.
  Donc le code de Hamming ne peut corriger qu'une seule erreur.

❑

Proposition 6.

Le code de Hamming est un code parfait.

Démonstration :

Le code de Hamming est 1-correcteur et de paramètres (7,4,3).

$$1 + 7 * (2 - 1) = 8 = 2^3 = 2^{7-4}$$

On a bien $1 + n(2 - 1) = 2^{n-k}$ (le 2 provient du fait que c'est un code binaire).

La borne de Hamming est atteinte, donc le code de Hamming est un code parfait.

❑

Proposition 7.

Le code de Hamming est un code cyclique.

Démonstration :

(Cette notion sera approfondie par la suite)
Montrons que H est stable par décalage circulaire.

Soit $m \in$ H.
Alors il existe $(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6) \in \{0, 1\}^7$ tels que m= $[\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6]$.

Montrons que m'=$[\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6] \in$ H
m' est constitué d'une suite de 0 et/ou 1, de longueur 7. Donc $m' \in$ H.
Conclusion: H est stable par décalage circulaire.

On en déduit que le code de Hamming est un code cyclique.

❑

3 Théorie des corps finis et application à la construction d'espaces adaptés

Il s'agit ici de déterminer les propriétés sur l'ensemble qui contient le code, pour que celui-ci soit performant.
On doit donc construire un corps (fini) qui puisse répondre à certaines propriétés.
En particulier, en notant $\mathbb{K}$ ce corps, on doit avoir :

- $\mathbb{K}$ doit être un sous-corps de l'ensemble d'arrivée de la fonction φ définie précédemment (souvent, cet ensemble sera $\mathbb{F}_p^n$, avec p un nombre premier qui sera, dans les faits, souvent égal à 2).
- On doit disposr, dans ce corps, de diviseurs de $X^n - 1$.
- Ce corps doit même être obtenu à partir d'un ces diviseurs, afin de construire à partir de ce diviseur une base du corps conidéré.

On notera également $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$
avec, par abus de language, $\mathbb{Z}/p\mathbb{Z}$ qui est en fait $\mathbb{Z}$ quotienté par la relation d'équivalence qu'est la congruence modulo p.
Et tout élément de cet ensemble sera noté $\tilde{n}$ au lieu de cl(n).

3.1 Quelques préliminaires

Définition :

Soit $\mathbb{K}$ un corps. On pose

$$\varphi : \begin{cases} \mathbb{Z} \longrightarrow \mathbb{K} \\ k \mapsto k \cdot 1_{\mathbb{K}} \end{cases}$$

On remarque que φ est un morphisme de groupes. $Ker(\varphi)$ est donc un sous-groupe de $\mathbb{Z}$.
Par caractérisation des sous-groupes de $\mathbb{Z}$, il existe $p \in \mathbb{N}$ tel que $Ker(\varphi) = p\mathbb{Z}$

On définit alors p comme la **caractéristique** de $\mathbb{K}$.

Remarque :

Si $p \neq 0$, p est nécessairement premier.
De plus, si $p = 0$, $\mathbb{K}$, l'ensemble $\{n \cdot 1_{\mathbb{K}},\ n \in \mathbb{Z}\}$ est un sous-anneau de $\mathbb{K}$, infini car décrit sans redondance lorsque n décrit $\mathbb{Z}$.

Il vient donc que : **Pour tout corps fini $\mathbb{K}$, $p \neq 0$ et donc p est premier**

Définition :

Soit $\mathbb{K}$ un corps. On dit que $\mathbb{K}$ est **cyclique** lorsqu'il existe $a \in \mathbb{K}$ tel que :

$$\forall g \in \mathbb{K}, \exists k \in \mathbb{N} \ / \ g = a^n$$

De plus, pour tout élément g de $\mathbb{K}$, il existe alors $r \in \mathbb{N}$ tel que $a^r = 1_{\mathbb{K}}$
Et on définit l'**ordre** de g, noté q comme :

$$q = min\{r \in \mathbb{N} \ / \ g^r = 1\}$$

Définition : .

Soit $\mathbb{K}$ un corps et $\mathbb{L}$ un sous-corps de $\mathbb{K}$. Soit $\alpha \in \mathbb{K}$.

- On dit que α est **algébrique** sur $\mathbb{L}$ lorsque : $\exists Q \in \mathbb{L}[X] \ / \ Q(\alpha) = 0$

  On pose P le polynome non nul de degré minimal qui vérifie cette propriété. P est appelé **polynôme minimal** de α sur $\mathbb{L}$.
  Et le degré d de P est appelé **degré** de α sur $\mathbb{L}$

- On définit $\mathbb{L}(\alpha) = \{Q(\alpha), Q \in \mathbb{L}[X]\}$.

Proposition 8.

On reprend les notations de la définition précédente. On a alors :

- $\forall Q \in \mathbb{L}[X]$,
$$Q(\alpha) = 0 \ \Leftrightarrow \ P \text{ divise } Q$$
De plus, P est irréductible sur $\mathbb{L}$.
- $\mathbb{L}(\alpha)$ est un sous-corps de $\mathbb{K}$.
- $\mathbb{L}(\alpha)$ est un espace vectoriel sur $\mathbb{L}$, et une base de cet espace vectoriel est $(1_{\mathbb{K}},\ \alpha, ...,\ \alpha^{d-1})$.

Démonstration :

- Par l'absurde, supposons qu'il existe $Q \in \mathbb{L}[X]$ tel que $Q(\alpha) = 0$ et P divise pas Q.
En écrivant la division euclidienne de Q par P, il vient qu'il existe $R \in \mathbb{L}[X]$, non nul et de degré $< d$, tel que :
$$Q(X) \ \equiv \ R(X)\ mod(P)$$

Puis en évaluant en α, il vient :
$R(\alpha) = 0$, ce qui contredit la minimalité de P.

En écrivant P comme produit de deux polynômes, puis en évaluant en α, il vient que l'un des deux au moins admet α comme racine.
Par minimalité de P, il vient alors que l'un des deux est associé à P, tandis que l'autre est constant.
P est donc irréductible.

- La stabilité de $\mathbb{L}(\alpha)$ par addition et multiplication est immédiate. De plus, on a bien $1_{\mathbb{K}} \in \mathbb{L}(\alpha)$. Montrons que tout élément de $\mathbb{L}(\alpha)$ est inversible.

  Soit $g \in \mathbb{L}(\alpha) \setminus \{0\}$. Il existe $Q \in \mathbb{L}[\alpha]$ tel que $g = Q(\alpha)$.
  P est irréductible et Q n'est pas un multiple de P donc P et Q sont premiers entre eux.
  Donc d'après le théorème de Bezout, il existe $(U, V) \in \mathbb{L}[X]^2$ tel que :

  $$U(X)Q(X) \ + \ V(X)P(X) \ = \ 1$$

  Puis en évaluant en α :

  $$U(\alpha)Q(\alpha) = 0$$

  Donc

  $$U(\alpha) \cdot g = 0$$

  Et $U(X) \in \mathbb{L}[X]$ donc $U(\alpha) \in \mathbb{L}(\alpha)$.
  g est donc inversible.

  $\mathbb{L}(\alpha)$ est donc un sous-corps de $\mathbb{K}$.

- En écrivant, pour tout $Q \in \mathbb{L}[X]$, la division euclidienne de Q par P, le caractère générateur de cette famille devient immédiat.
  De plus, toute combinaison linéaire nulle de cette famille est de la forme $Q(\alpha)$, avec $Q \in \mathbb{L}[X]$ et $\deg(Q) <$ d. Il vient donc par minimalité de P que $Q = 0$, donc cette combinaison linéaire est triviale. La famille est libre.

  $(1_{\mathbb{K}}, \ \alpha, ..., \ \alpha^{d-1})$ est donc une base de $\mathbb{L}(\alpha)$ en tant que $\mathbb{L}$-espace vectoriel.

❑

Définition :

On reprend les notations précédentes.

On dit que α est un **élément primitif** de $\mathbb{K}$ lorsque :

$$\mathbb{K} = \mathbb{L}(\alpha)$$

3.2 Généralités sur les anneaux quotients et les corps finis

Définition :

Soit $\mathbb{A}$ un anneau. Soit $\mathbb{I}$ un ensemble inclus dans $\mathbb{A}$.
On dit que $\mathbb{I}$ est un **idéal** de $\mathbb{A}$ lorsque :

$$\begin{cases} (\mathbb{I}, +) \text{ est un sous-groupe de } (\mathbb{A}, +) \\ \forall x \in \mathbb{I}, \ \forall a \in \mathbb{A}, x \cdot a \in \mathbb{I} \text{ et } a \cdot x \in \mathbb{I} \end{cases}$$

Exemple :

- Pour tout $n \in \mathbb{Z}$, l'ensemble $n\mathbb{Z}$ des multiples de n est un idéal.
- Pour tout $P \in \mathbb{K}[X]$, l'ensemble (P) des multiples de P est un idéal également.

Proposition 9.

Soit $\mathbb{A}$ un anneau et I un idéal de $\mathbb{A}$.
Alors la relation $\sim$ définie sur $\mathbb{A}$ par :

$$\forall (x,y) \in \mathbb{A}^2,\ x \sim y \Leftrightarrow x - y \in \mathbb{I}$$

est une relation d'équivalence sur A.

Et on note

$$\mathbb{A}/\mathbb{I} \ = \ \mathbb{A}/\sim$$

Reprenons nos deux exemples précédents :

Alors on montre grâce au théorème de Bezout que si p est premier (respectivement si P est irréductible), l'anneau quotient $\mathbb{Z}/p\mathbb{Z}$ (respectivement $\mathbb{K}[X]/(P)$) est un corps.

Proposition 10.

Soit $\mathbb{K}$ un corps fini de caractéristique p.

- Il existe $n \in \mathbb{N}$ tel que $\#\mathbb{K} = p^n$. Plus précisément, n est la dimension de $\mathbb{K}$ en tant que $\mathbb{F}_p$-espace vectoriel.
- $\mathbb{K}^*$ est cyclique, de cardinal $p^n - 1$.

Démonstration :

On va démontrer la première propriété.

Commençons par poser sur $\mathbb{K}$ une structure de $\mathbb{F}_p$-espace vectoriel. Pour cela, on définit la multiplication externe par tout élément $\tilde{n}$ élément de $\mathbb{F}_p$:

$$\forall g \in \mathbb{K},\ \tilde{n} \cdot g = n \cdot g (\text{au sens de l'itérée})$$

.
Remarquons qu'on n'a pas de problème de bonne définition.
En effet, soit $(k,l) \in \mathbb{Z}^2$ tel que $\tilde{k} = \tilde{l}$ et $g \in \mathbb{K}$. On doit donc montrer que $\tilde{k} \cdot g = \tilde{l} \cdot g$.
Or, il existe $q \in \mathbb{Z}$ tel que $k - l = q \times p$.
Il vient alors :

$$\begin{aligned} \tilde{k} \cdot g - \tilde{l} \cdot g &= (k-l) \cdot g \\ &= qp \cdot g \\ &= q(p \cdot 1_{\mathbb{K}}) \cdot g \\ &= q \cdot 0_{\mathbb{K}} \cdot g \\ &= 0_{\mathbb{K}} \end{aligned}$$

D'où finalement l'égalité qu'on voulait démontrer.
Et la qualité de $\mathbb{F}_p$-espace vectoriel vient enuite assez naturellement.

De plus la famille $(x)_{x \in \mathbb{K}}$ est une famille génératrice **finie** de $\mathbb{K}$ (car $\mathbb{K}$ est un corps fini).
Donc $\mathbb{K}$ est un $\mathbb{F}_p$-espace vectoriel de dimension finie. Posons n sa dimension, et posons $(e_1, ..., e_n)$ une base de $\mathbb{K}$

Alors la fonction

$$\varphi \ : \begin{cases} \mathbb{F}_p^n \longrightarrow \mathbb{K} \\ (\lambda_1, ..., \lambda_n) \mapsto \sum_{i=1}^{n} \lambda_i \cdot e_i \end{cases}$$

est, par définition, bijective.

Et puisque p est premier, on sait que $\mathbb{F}_p$ est un corps de cardinal p.
Par égalité des cardinaux des ensembles de départ et d'arrivée de φ, il vient finalement :

$$\#\mathbb{K} = p^n$$

❑

Remarque :
Puisque $\mathbb{K}*$ est cyclique, il existe α un générateur de $\mathbb{K}*$.
Et α est alors également un élément primitif de $\mathbb{K}$.
Il vient donc que :

Tout corps fini admet un élément primitif

Proposition 11.

Soit $\mathbb{K}$ un corps fini de caractéristique p. Soit P un polynôme irréductible de $\mathbb{K}[X]$, de degré r.

Alors le corps $\mathbb{K}[X]/(P)$ est fini, de cardinal p^r.

Démonstration :

$\mathbb{K}$ est de caractéristique p, donc il vient naturellement que $\mathbb{K}[X]/(P)$ est fini également, de caractéristique p.
De plus, on montre que $(X^0, X, ..., X^{r-1})$ est une base de $\mathbb{K}[X]/(P)$.

On conclut d'après la proposition précédente.

❑

Remarque :
D'après cette démonstration, et ce qu'on a fait précédemment, il vient que pour tout élément primitif α de $\mathbb{K}$, et avec $\mathbb{L}$ le sous-corps de $\mathbb{K}$ qui convient :

On peut identifier les corps $\mathbb{L}(\alpha)$ et $\mathbb{L}[X]/(P)$, avec P le polynôme minimal de α.

3.3 Polynômes cyclotomiques et racines n-ièmes de l'unité

Définition :

Soit $n > 0$.
On pose $(\mathbb{Z}/n\mathbb{Z})^*$ l'ensemble des inversibles de $\mathbb{Z}/n\mathbb{Z}$, et on note $\varphi(n)$ le cardinal de ce corps.

On définit alors le n-**ième polynôme cyclotomique** :

$$\Phi_n(X) = \prod_{k \in (\mathbb{Z}/n\mathbb{Z})^*} (X - e^{\frac{2ik\pi}{n}})$$

Proposition 12.

Soit $n > 0$.

- $\Phi_n(X) = \prod_{d|n} \Phi_d(X)$
- $\Phi_n(X)$ est à coefficients entiers.

Démonstration :

La première proposition est assez naturelle.
Et on peut déduire la seconde de la première par récurrence forte sur n.

❑

Remarque :

Cette proposition permet pour tout corps $\mathbb{K}$ d'écrire $\Phi_n(X)$ comme polynôme de $\mathbb{K}[X]$, en remplaçant chacun de ses coefficients par son image par la fonction $\varphi : \begin{cases} \mathbb{Z} \longrightarrow \mathbb{K} \\ k \mapsto k \cdot 1_{\mathbb{K}} \end{cases}$

Théorème 1.

Soit $\mathbb{K}$ un corps fini de cardinal q. Soit n un entier naturel premier à q.
On pose r l'ordre de q dans $(\mathbb{Z}/n\mathbb{Z})^*$.

Vient alors ce théorème puissant qu'on admettra :

Φ_n se décompose dans $\mathbb{K}[X]$ en polynômes irréductibles de degré r, tous différents.

On pose maintenant $\mathbb{K}$ un corps fini, à q éléments. On pose également n un entier positif premier à q. On va à présent pouvoir construire ce corps dont on parlait en début de partie.

On pose α une racine de l'unité. On va maintenant expliciter le corps $\mathbb{K}(\alpha)$.

Proposition 13.

On reprend les notations de la proposition précédente.

- Le polynôme minimal de α est un des diviseurs de degré r de $\Phi_n(X)$
- $\mathbb{K}(\alpha)$ peut donc s'identifier à $\mathbb{K}[X]/(P)$, avec P le polynôme minimal de α.

Finalement, on a donc construit un corps dans lequel se plonge $\mathbb{K}$ et pour lequel on possède bien des racines de l'unité.

On sait que ce corps est de cardinal q^r.
A isomorphisme près, ce corps ne dépend donc pas du choix de α.

On le note alors $R_n(\mathbb{K})$

3.4 Diviseurs de $X^n - 1$ dans $\mathbb{F}_q[X]$

On se place maintenant dans le cas qui nous intéresse vraiment pour la suite.
On pose donc q un nombre premier et n un entier naturel premier à q.

On cherche donc à expliciter un algorithme permettant de déterminer un diviseur non trivial (ie. différent de $X-1$) de X^n-1 dans $\mathbb{F}_q[X]$.

On part de la propriété suivante, qui découle du travail effectué précédemment sur les polynômes cyclotomiques :

Proposition 14.

Il existe un diviseur différent de $X-1$ de X^n-1 dans $\mathbb{F}_q[X]$.
Ce diviseur est de degré r, avec r l'ordre de q dans $(\mathbb{Z}/n\mathbb{Z})^*$.

Démonstration :

Puisque n est premier,

$$X^n - 1 = (X-1)\Phi_n(X)$$

D'où le résultat.

❑

On se place à présent dans le fameux corps $R_n(\mathbb{F}_q)$ obtenu précédemment.
Dans ce corps, il existe n racines n-ièmes de l'unité. Soit α l'une d'entre elles.

On sait alors que : $X^n - 1 = \prod_{k=1}^{n}(X - \alpha^i)$

On cherche donc une partie de $\mathbb{Z}/n\mathbb{Z}$, notée Σ, telle que $g_\Sigma := \prod_{k\in\Sigma}(X-\alpha^i)$ soit à coefficients dans $\mathbb{F}_q$.

Proposition 15.

On a l'équivalence suivante :

$$g_\Sigma \text{ est à coefficients dans } \mathbb{F}_q \;\Leftrightarrow\; \Sigma \text{ est stable par multiplication par q}$$

Démonstration :

On donnera ici les grandes étapes de la démonstration.

On remarque que d'après le petit théorème de Fermat, pour tout k dans $\mathbb{F}_q$, $k^q = k$.
De plus, $\forall (x,y) \in \mathbb{F}_q^2,\ (x+y)^p = x^p + y^p$

Soit maintenant un polynôme Q à coefficients dans $\mathbb{F}_q$.
Il vient alors que $Q(X)^p = Q(X^p)$.
Et donc pour toute racine α de Q, α^p est également racine.
Donc Σ est stable par multiplication par p.

Le sens réciproque se fait de manière équivalente.

❑

Ayant obtenu une telle partie stable, il faut encore vérifier qu'on ait un diviseur de X^n-1.
Pour cela, il faut calculer le PGCD de X^n-1 et du polynôme g_Σ obtenu.

Alors ce PGCD convient !

4 Codes cycliques et codes de résidus quadratiques

On pose q un nombre premier.
On se place maintenant dans l'espace $\mathbb{F}_q^n$

4.1 Codes cycliques

Définition :

Un code linéaire C est dit **cyclique** lorsque C est stable par décalage circulaire.

Autrement dit:

Définition :

Soit C un code linéaire de longueur n.
Soit σ une permutation circulaire telle que

$$\sigma((m_0, m_1, ..., m_{n-2}, m_{n-1})) = (m_{n-1}, m_0, ..., m_{n-2})$$

Alors C est dit cyclique lorsque:

$$\forall m \in \mathrm{C}, \sigma(m) \in \mathrm{C}.$$

Définition :

Soit $\mathbb{F}_q^n$ un espace-vectoriel constitué de mots de longueur n, à partir d'un alphabet de q éléments.
Alors pour tout mot m=$[m_0 m_1 ... m_{n-k}]$ de $\mathbb{F}_q^n$, on peut associer un polynôme m(X) tel que:

$$m(X) = m_0 + m_1 * X + m_2 * X^2 + ... + m_{n-k} * X^{n-1}$$

Dans la suite, pour tout code cyclique C et tout mot m de C, on pose $m(X)$ le polynôme associé à m.

Proposition 16.

Soit C un code cyclique et m un mot de ce code.
Soit σ la permutation circulaire associée.

Alors

$$(\sigma(m))(X) \equiv Xm(X)\ [X^n - 1]$$

Démonstration :

$(\sigma(m))(X) = m_{n-1} + m_0 X + ... + m_{n-2} X^{n-1}$.
Il vient donc que :

$$Xm(X) = m_{n-1}(X^n - 1) + (\sigma(m))(X)$$

D'où l'égalité modulo $X^n - 1$ déterminée précédemment.

❑

Ainsi, en assimilant, comme on peut le faire, $\mathbb{F}_q^n$ et $\mathbb{F}_q[X]/(X^n - 1)$, il vient que les codes cycliques de longueur n sont assimilés aux sous-espaces de $\mathbb{F}_q[X]/(X^n - 1)$ stables par multiplication par X.

Or un tel espace est un idéal de $\mathbb{F}_q[X]/(X^n - 1)$.
Par image réciproque de l'anneau quotient, on doit donc rechercher un idéal de $\mathbb{F}_q[X]$, qui contient $X^n - 1$.

Il s'agit donc de rechercher un diviseur de $X^n - 1$ dans $\mathbb{F}_q$.

Tout cela est explicité dans la proposition suivante :

Proposition 17.

Soit $n, q, k \in \mathbb{N}^3$, avec $k \leq n$.

Soit $H(X) \in \mathbb{F}_q[X]$, que l'on écrit H(X)=$m_0 + m_1 * X + m_2 * X^2 + ... + m_{n-k} * X^{n-k}$, tel que H(X) divise $X^n - 1$.

Il existe alors m$\in \mathbb{F}_q^n$, tel que m=$[m_0 m_1 ... m_{n-k} 00..00]$ (k-1 zéros), et un code cyclique C de $\mathbb{F}_q^n$ de dimension k, dont une base est $(m, \sigma(m), ..., \sigma^{k-1}(m))$,
avec σ la permutation circulaire définie précédemment.
On dit que C est **engendré** par m.

H(X) est alors nommé le polynôme générateur du code C.

Démonstration :

- Soit $n, q, k \in \mathbb{N}^3$, avec $k \leq n$. Soit $H(X) \in \mathbb{F}_q[X]$, que l'on écrit H(X)=$m_0 + m_1 * X + m_2 * X^2 + ... + m_{n-k} * X^{n-k}$, tel que H(X) divise $X^n - 1$.

- Les coefficients de H(X) appartiennent alors à $\mathbb{F}_q$. Donc on peut lui faire correspondre un mot m de $\mathbb{F}_q^n$, les premiers caractères du mot étant les coefficients du polynôme H(X), et en complétant le reste par des 0. Comme $0 \in \mathbb{F}_q$, on a bien m$\in \mathbb{F}_q^n$, et m=$[m_0 m_1 ... m_{n-k} 00..00]$.

- On a:

$$\begin{aligned}
m &= [m_0\ m_1\ ...\ m_{n-k}\ 0\ 0\ ...\ 0\ 0] \\
\sigma(m) &= [0\ m_0\ m_1\ ...\ m_{n-k}\ 0\ 0\ ...\ 0] \\
\sigma^2(m) &= [0\ 0\ m_0\ m_1\ ...\ m_{n-k}\ 0\ ...\ 0] \\
&... \\
\sigma^{k-2}(m) &= [0\ ...\ 0\ 0\ m_0\ m_1\ ...\ m_{n-k}\ 0] \\
\sigma^{k-1}(m) &= [0\ 0\ ...\ 0\ 0\ m_0\ m_1\ ...\ m_{n-k}]
\end{aligned}$$

Donc $(m, \sigma(m), \sigma^2(m), ..., \sigma^{k-2}(m), \sigma^{k-1}(m))$ est une famille libre.

Montrons que le code C engendré par m, c'est-à-dire engendré par la famille libre $(m, \sigma(m), \sigma^2(m), ..., \sigma^{k-2}(m), \sigma^{k-1}$ est un code cyclique.
Cela revient à montrer que C est stable par permutation circulaire σ.
Soit a un mot du code C. Alors a=$a_0 * m + a_1 * \sigma(m) + ... + a_{k-1} * \sigma^{k-1}(m)$.
Il s'agit de montrer que $\sigma(a) \in$ C.
Or,

$$\begin{aligned}
\sigma(a) &= \sigma(a_0 * m + a_1 * \sigma(m) + ... + a_{k-1} * \sigma^{k-1}(m)) \\
&= a_0 * \sigma(m) + a_1 * \sigma^2(m) + ... + a_{k-1} * \sigma^k(m)
\end{aligned}$$

Il suffit donc de montrer que $\sigma^k(m) \in$ C.
On sait que H(X) divise $X^n - 1$, donc il existe $Q(X) \in \mathbb{F}_q[X]$ tel que $X^n - 1$=H(X)*Q(X).
On peut poser $Q(X) = q_0 + q_1 * X + ... + q_{k-1} * X^{k-1} + X^k$.

Donc,

$$\begin{aligned} X^n - 1 &= H(X) * (q_0 + q_1 * X + ... + q_{k-1} * X^{k-1} + X^k) \\ &= q_0 * H(X) + q_1 * X * H(X) + ... + q_{k-1} * X^{k-1} * H(X) + X^k * H(X) \end{aligned}$$

D'où,

$X^k * H(X) \equiv -q_0 * H(X) - q_1 * X * H(X) - ... - q_{k-1} * X^{k-1} * H(X)\ [X^n - 1]$

Donc

$\sigma^k(m) = -q_0 * m - q_1 * \sigma(m) - ... - q_{k-1} * \sigma^{k-1}(m)$

Donc $\sigma^k(m) \in$ C.
Ainsi, C est un code cyclique.

On peut donc conclure que la famille libre $(m, \sigma(m), \sigma^2(m), ..., \sigma^{k-2}(m), \sigma^{k-1}(m))$ engendre un code C, donc elle est une base du code C.

On en déduit également dim(C)=#b=k.

❑

Définition :

La matrice génératrice de C est alors définie par

$$\begin{pmatrix} m_0 & m_1 & ... & m_{n-k} & 0 & ... & ... & 0 \\ 0 & m_0 & m_1 & ... & m_{n-k} & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & ... & 0 & m_0 & m_1 & ... & m_{n-k} & 0 \\ 0 & ... & 0 & m_0 & m_1 & ... & ... & m_{n-k} \end{pmatrix}$$

Proposition 18.

Tout code C de $\mathbb{F}_q^n$ admet un polynôme générateur dans $\mathbb{F}_q[X]$.

Principe de codage
Soit C un code de $\mathbb{F}_q^n$ et H(X) son polynôme générateur.
Soit m le mot que l'on veut coder, avec M(X) sa représentation polynômiale.
Alors le mot obtenu après le codage a pour représentation polynômiale $M(X) * H(X)$

Proposition 19.

Soit C un code de $\mathbb{F}_q^n$ et H(X) son polynôme générateur.
Soit m$\in \mathbb{F}_q^n$, avec m=$[m_0 m_1 ... m_n]$.
On pose M(X)=$m_0 + m_1 * X + m_2 * X^2 + ... + m_n * X^n$

$$\text{m} \in \text{C} \Leftrightarrow H(X) | M(X)$$

Proposition 20.

Minoration de la distance minimale d'un code cyclique

Soit C un code de $\mathbb{F}_q^n$, engendré par le mot m=$[m_0 m_1 ... m_n]$, et d_{min} la distance minimale du code.
On pose a le nombre de coefficients non nuls du polynôme associé à m.

Alors $a \leq d_{min}$.

4.2 Codes de résidus quadratiques

Définition :

Soit p un nombre premier strictement supérieur à 2.
Soit n$\in \mathbb{Z}$.

n est un résidu quadratique modulo p $\Leftrightarrow \exists m \in \mathbb{Z}$, $m^2 \equiv n\,[p]$.

Définition :

Un code C de $\mathbb{F}_q^p$ est un code de résidus quadratiques lorsque p est un nombre premier différent de q (on dit alors que q est résidu quadratique modulo p).

Un code de résidus quadratiques est un cas particulier de codes cycliques, dont l'avantage est une distance minimale élevée.

Etant donnés les objectifs de ce TIPE, on va seulement s'intéresser aux codes de résidus quadratiques binaires, i.e. les sous-espaces vectoriels de $\mathbb{F}_2^p$, avec p un nombre premier strictement supérieur à 2.

Proposition 21.

Soit un p un nombre premier strictement supérieur à 2.
Un code de résidus quadratiques associé à p est de dimension $\frac{n+1}{2}$.

Proposition 22.

Soit C un code de résidus quadratiques associé à p.
Le polynôme générateur de C est sous la forme:

H(X)=$\prod_{i \in Q}(X - \alpha^i)$

où α est la racine p-ième primitive de l'unité de $\mathbb{F}_q^p$,
Q est l'ensemble des résidus quadratiques non nuls modulo p.

Définition :

Soit C un code de résidus quadratiques associé à p.
Alors C est forme de tous les polynômes de $\mathbb{F}_q[X]$, de degré inférieur ou égal à p-1, qui admettent tous les α^i comme racines.

Proposition 23.

Soit p un nombre premier tel que p$\equiv -1\,[8]$.
Alors la distance minimale du code binaire de résidus quadratiques associé à p vérifie:

$$p \leq d_{min}^2 - d_{min} + 1$$

5 Application : le code de Golay

On traite ici du cas assez commode du code binaire de Golay G_{23}. Ce code possède un bon taux d'information et a notamment été utilisé pour la transmission des images en couleurs envoyées par les sondes Voyager durant les années 1980.

Définition :

Le code G_{23} est le code de résidus quadratiques de longueur 23.

5.1 Construction

On cherche d'abord un polynôme générateur de notre code, c'est à dire un polynome unitaire irréductible divisant $X^{23}+1$ (qui est égal à $X^{23}-1$ ici)

Proposition 24.

Le polynôme $P(X) = X^{11}+X^{10}+X^6+X^5+X^4+X^2+1$ est générateur de G_{23}

Démonstration :

L'ordre de 2 dans $(\mathbb{Z}/23\mathbb{Z})^*$ est 11,
on a donc 2 classes cyclotomiques de longueur 11 stables par multiplication par 2 :

$C_1 = \{1,2,4,8,16,9,18,13,3,6,12\}$
$C_2 = \{5,7,10,11,14,15,17,19,20,21,22\}$

On construit ainsi deux polynômes,
$P_1(X) = X^{18}+X^{16}+X^{13}+X^{12}+X^9+X^8+X^6+X^4+X^3+X^2+X$
$P_2(X) = X^{22}+X^{21}+X^{20}+X^{19}+X^{17}+X^{15}+X^{14}+X^{11}+X^{10}+X^7+X^5$

En réalisant les pgcd de ces deux polynômes avec $X^{23}+1$, on obtient deux polynômes de degré 11 diviseurs de $X^{23}+1$,
$Q_1(X) = X^{11}+X^9+X^7+X^6+X^5+X+1$
$Q_2(X) = X^{11}+X^{10}+X^6+X^5+X^4+X^2+1$

Ces deux polynômes sont générateurs de notre code. On conserve généralement Q_2.

❑

Proposition 25.

- Le code de Golay de longueur 23 est un code cyclique parfait.
- Il s'agit d'un code de paramètres (23,12,7). Il est donc 3-correcteur.
- Le taux d'information de ce code est $\frac{12}{23}$
- On peut ajouter un bit de parité à ce code pour obtenir le code de Golay étendu de paramètres (24,12,8)

5.2 Matrices génératrice et de contrôle de parité

Le polynôme générateur permet de définir une base de ce code composée du mot

$$\mathrm{v}_1 = [11000111010100000000000]$$

et de ses décalages successifs vers la droite v_2, ..., v_{12}.

On peut alors écrire une matrice dite génératrice du code :

$$\mathrm{M} = \begin{pmatrix} \mathrm{v}_1 \\ \mathrm{v}_2 \\ \dots \\ \mathrm{v}_{12} \end{pmatrix}$$

Définition :

- Un codage est dit systématique lorsque l'on retrouve dans le mot codé les k symboles d'information dans k positions déterminées.
- Une matrice sous forme systématique est composée de 2 sous-matrices : la matrice identité I_k et une matrice $k * (n-k)$ dite de parité (notée P) telles que :

$$G_{sys} = (\ I_k \mid \mathrm{P}\ )$$

Lorsqu'un codage existe sous forme systématique, on peut mettre sa matrice sous forme systématique en lui appliquant des opérations élémentaires sur les lignes et/ou des permutations sur les colonnes.

Proposition 26.

La matrice génératrice sous forme systématique du code G_{23} est :

$$G_{sys} = (\ I_{12} \mid \mathrm{P}\ )$$

$$\text{où } P = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Ce résultat peut être vérifié algorithmiquement.
L'avantage de cette forme est que le message d'origine est intact et se trouve au début du mot de code.

Définition :

Soit C un code linéaire.
Soit H une matrice de dimension $(n-k) * n$.
H est une matrice de contrôle lorsque :

$$m \in \mathrm{C} \Leftrightarrow m.H^t = 0$$

Ce qui est équivalent à $G.H^t = 0$

Cette matrice peut également se mettre sous forme systématique.

Proposition 27.

En notant P la matrice de parité définie précédemment, on a

$$H_{sys} = (\ \text{-P}^t \mid I_{n-k}\ ) = (\ \text{P}^t \mid I_{n-k}\ )$$

avec $(n-k) = 11$ dans notre cas.

Démonstration :

$G_{sys}.H_{sys}^t = (\ I_k \mid \text{P}\ ).(\ \text{P}^t \mid I_{n-k}\ )^t = -\text{P} + \text{P} = 0$

❑

5.3 Procédures de codage et décodage

Définition :

On nomme syndrôme d'un vecteur x de $\mathbb{F}_2^n$ le vecteur y de longueur $(n-k)$ tel que $y = m.H^t$On nomme syndrôme d'un vecteur x de $\mathbb{F}_2^n$ le vecteur y de longueur $(n-k)$ tel que $y = m.H^t$

- On commence par créer une Look-Up Table (LUT), c'est à dire une table mettant en correspondance tous les syndrômes avec les vecteurs d'erreur correctibles correspondants (pour cela, on calcule le syndrôme des vecteurs d'erreur).
  Dans le cas des codes parfaits, il y a autant de syndromes que de vecteurs d'erreur, on a la belle identité :

$$\sum_{i=0}^{3} \binom{23}{i} = 2^{11}$$

- On sectionne ensuite l'information binaire à transmettre en blocs de k éléments.
- On code chacun de ces vecteurs en le multipliant par la matrice génératrice G_{sys}.
- On fait passer l'information ainsi codée par un canal bruité.
- On récupère le syndrôme de chaque vecteur de longueur 23 en le multipliant par la matrice de contrôle de parité H_{sys}.
- On trouve l'erreur associée au syndrôme à l'aide de la table.
- On additionne le vecteur bruité et l'erreur probable modulo 2.
- On récupère pour chaque bloc de 23 caractères les 12 premiers qui correspondent à notre mot d'origine (si il y a eu moins de 3 erreurs)

5.4 Mise en application sur des images avec Python

On a ici commencé par mettre sous la forme d'une matrice à 2 dimensions en noir et blanc toutes les images pour ensuite les coder/bruiter/décoder.
On a ici choisi de modifier aléatoirement la couleur de chaque pixel avec une probabilité de 0.03

Remarque :
On aurait pu traiter également des images en couleur en exploitant les écritures binaires des entiers codant les couleurs dans le système RGB.

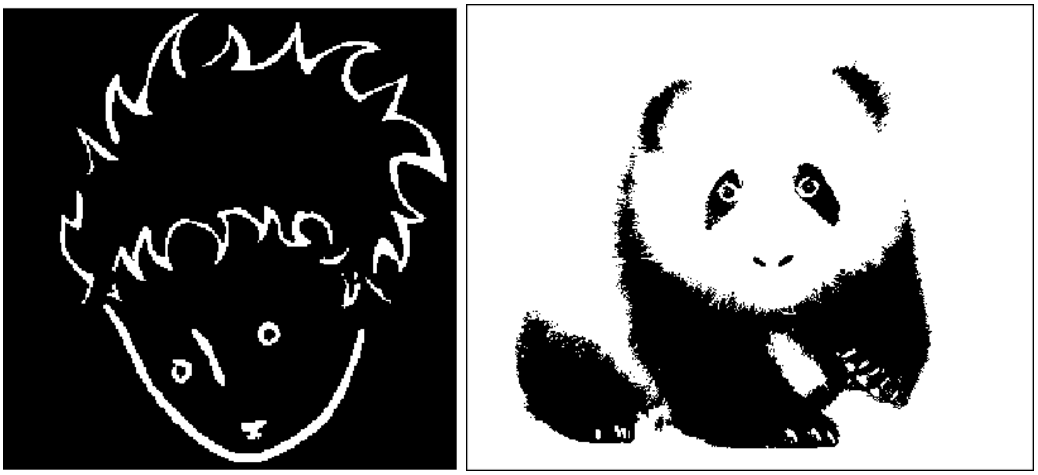


Figure 3: Images originales

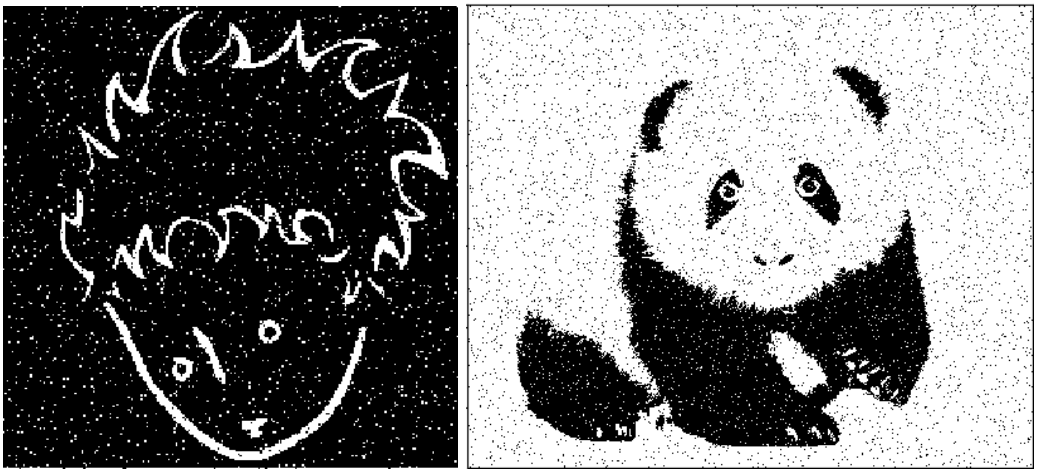


Figure 4: Images bruitées avec une probabilité de 0.03

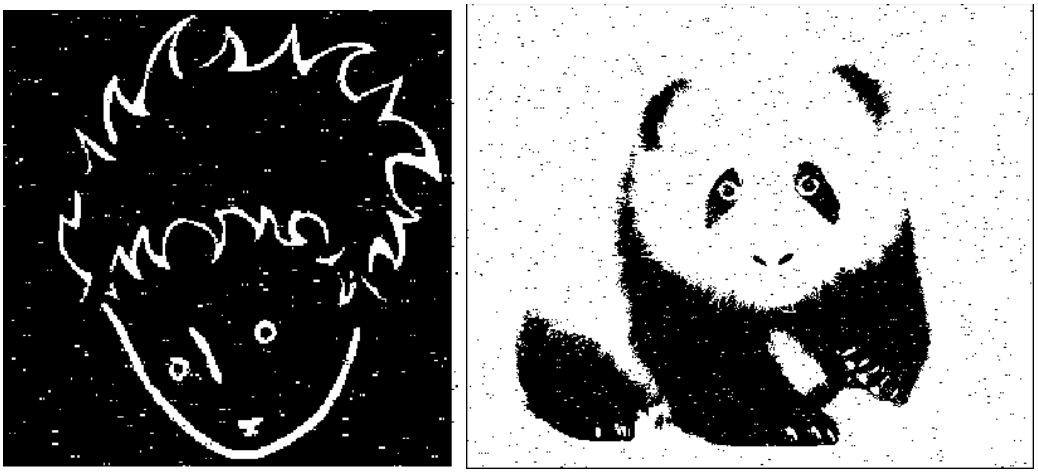


Figure 5: Images bruitées codées avec le code de Hamming (7,4,3)

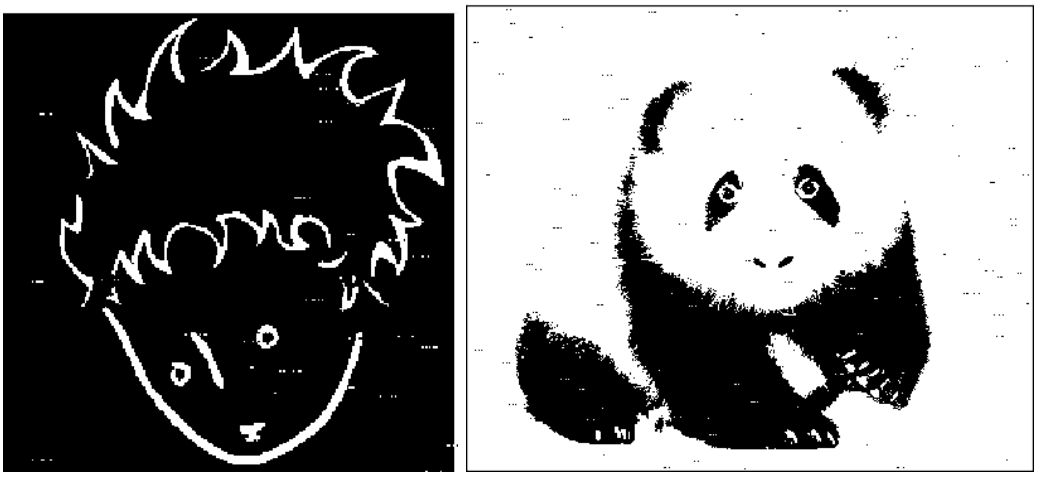


Figure 6: Images bruitées codées avec le code de Golay

On remarque que les erreurs qui persistent touchent des blocs entiers (de longueur 12 pixels pour le code de Golay). On aurait pu répartir les erreurs de manière plus homogène en créant les blocs de manière moins linéaire.

6 Conclusion

Ainsi, la théorie des codes correcteurs, et en particulier celle des codes linéaires, implique la connaissance de certains espaces mathématiques poussés : cependant, en pratique, la construction d'un code ne demande de connaître que certaines propriétés, bien reconnaissables et qui permettent de caractériser tel ou tel code.
Par exemple, Golay a eu l'idée de son code simplement en se rendant compte de l'identité $\sum_{i=0}^{3} \binom{23}{i} = 2^{11}$!

Ainsi, ces techniques nous ont permis de reproduire nous-mêmes l'algorithme de codage/décodage du code de Goley; et de le mettre en application : plus que la partie théorique, c'est cette application que nous voudrions mettre en valeur, car elle a une vraie utilité expérimentale, comme nous l'avons prouvé.

Cependant, nous devons reconnaitre que bien que poussé, ce code n'est déjà plus utilisé, dans les faits, depuis plusieurs décennies.
En effet, bien que non parfaits, d'autres codes permettent de plus grande performances au niveau de la correction, ainsi que plus d'efficacité en limitant le poids informatique des opérations de codage et de décodage.

7 Bibliographie - Sitographie

Cours d'algèbre, Michel Demazure
http://www.lirmm.fr/~chaumont/CorrectingCodes.html
https://fr.wikipedia.org/wiki/Code_correcteur
http://www-igm.univ-mlv.fr/~desar/Cours/M1-1_Codage_Cryptogrphie/chap2.pdf
http://library.iugaza.edu.ps/thesis/91817.pdf

8 Annexe (programmes)

```
import pylab as pl
import time
import random

tot = time.time()

def modif(m,i):
    m[i] = (m[i]+1)%2


# CODE DE GOLAY

A = pl.matrix([[1,1,1,1,1,1,1,1,1,1,1],
        [1,1,0,1,1,1,0,0,0,1,0],
        [1,0,1,1,1,0,0,0,1,0,1],
        [0,1,1,1,0,0,0,1,0,1,1],
        [1,1,1,0,0,0,1,0,1,1,0],
        [1,1,0,0,0,1,0,1,1,0,1],
        [1,0,0,0,1,0,1,1,0,1,1],
        [0,0,0,1,0,1,1,0,1,1,1],
        [0,0,1,0,1,1,0,1,1,1,0],
        [0,1,0,1,1,0,1,1,1,0,0],
        [1,0,1,1,0,1,1,1,0,0,0],
        [0,1,1,0,1,1,1,0,0,0,1]])


gene = pl.block([pl.eye(12),A])
contro = pl.block([[A],[pl.eye(11)]])

print(gene)
# print("")
# print(contro)
# print("")

def code(m):
    res = pl.dot(m,gene)
    for i in range(len(res)):
        res[i]= res[i]%2
    return pl.ravel(res)


def syndrome(m):
    res = pl.dot(m,contro)
    for i in range(len(res)):
        res[i]= res[i]%2
    return pl.ravel(res)


```

```
49  def creaTable() :
50      LUTs = []
51      LUTs.append([pl.zeros((23)),pl.zeros((11))])
52      for i in range(23):
53          l=pl.zeros((23))
54          modif(l,i)
55          LUTs.append([l,syndrome(l)])
56
57      for i in range(23):
58          for j in range(i+1,23):
59              l=pl.zeros((23))
60              modif(l,i)
61              modif(l,j)
62              LUTs.append([l,syndrome(l)])
63
64      for i in range(23):
65          for j in range(i+1,23):
66              for k in range(j+1,23):
67                  l=pl.zeros((23))
68                  modif(l,i)
69                  modif(l,j)
70                  modif(l,k)
71                  LUTs.append([l,syndrome(l)])
72      return pl.array(LUTs)
73
74
75  t = time.time()
76  Table = creaTable()
77  print("temps table golay :")
78  print(time.time()-t)
79  print("")
80
81  def decode(m):
82      synd = syndrome(m)
83      for e in Table:
84
85          if (synd == e[1]).all():
86              err = e[0]
87              corr  = err + m
88              for i in range(len(corr)):
89                  corr[i]= corr[i]%2
90              return corr[:12]
91      print("erreur decodage golay")
92
93
94  # CODE DE HAMMING
95
96  B = pl.matrix([[0,1,1],[1,0,1],[1,1,0],[1,1,1]])
97
98  geneh = pl.block([pl.eye(4),B])
99  controh = pl.block([[B],[pl.eye(3)]])
100
101
102 def codeh(m):
103     res = pl.dot(m,geneh)
104     for i in range(len(res)):
105         res[i]= res[i]%2
106     return pl.ravel(res)
```

```
107
108
109 def syndromeh(m):
110     res = pl.dot(m,controh)
111     for i in range(len(res)):
112         res[i]= res[i]%2
113     return pl.ravel(res)
114
115
116 def creaTableh() :
117     LUTs = []
118     LUTs.append([pl.zeros((7)),pl.zeros((3))])
119     for i in range(7):
120         l=pl.zeros((7))
121         modif(l,i)
122         LUTs.append([l,syndromeh(l)])
123
124     return pl.array(LUTs)
125
126
127 t = time.time()
128 Tableh = creaTableh()
129
130 print("temps table hamming :")
131 print(time.time()-t)
132 print("")
133
134 def decodeh(m):
135     synd = syndromeh(m)
136     for e in Tableh:
137
138         if (synd == e[1]).all():
139             err = e[0]
140             corr  = err + m
141             for i in range(len(corr)):
142                 corr[i]= corr[i]%2
143             return corr[:4]
144     print("erreur decodage hamming")
145
146
147
148 # IMAGES / MODIFICATIONS
149
150 def binim(im):
151     p,q = im.shape[0],im.shape[1]
152     nveau = pl.zeros((p,q))
153     for i in range(p):
154         for j in range(q):
155
156             if pl.sum(im[i,j])> 1.5:
157                 nveau[i,j] = 1
158             else :
159                 nveau[i,j] = 0
160     return nveau
161
162 def imaff(bin):
163     p,q = bin.shape[0],bin.shape[1]
164     nveau = pl.zeros((p,q,3))
```

```
    for i in range(p):
        for j in range(q):
            a = bin[i,j]
            nveau[i,j]= pl.array([a,a,a])

    return nveau


pingu = pl.imread('/Users/alexgado/Documents/INFO/IPT/pinguin.jpg')
auguste = pl.imread('/Users/alexgado/Documents/INFO/IPT/auguste.JPG')
axiome = pl.imread('/Users/alexgado/Documents/INFO/IPT/axiome.png')
ellipse = pl.imread('/Users/alexgado/Documents/INFO/IPT/ellipse.png')
panda = pl.imread('/Users/alexgado/Documents/INFO/IPT/panda.png')



def modifvect(m,p):
    for i in range(len(m)):
        r = random.random()
        if r < p:
            m[i] = (m[i]+1)%2


def modifim(im,pr):
    bin = binim(im)
    (p,q) = bin.shape
    ligne = pl.ravel(pl.reshape(bin, (1,p*q)))
    modifvect(ligne,pr)
    finalbin = pl.reshape(ligne,(p,q))
    return imaff(finalbin)


# TESTS GOLAY

def modifimcorr(im,pr):
    bin = binim(im)
    (p,q) = bin.shape
    ligne = pl.ravel(pl.reshape(bin, (1,p*q)))
    nveau = []
    for i in range(0,int(p*q)-11,12):

        bloc = ligne[i:i+12]
        codeb = code(bloc)

        modifvect(codeb,pr)
        deco = decode(codeb)
        nveau.append(deco)
    nveau.append(ligne[p*q-1:p*q-1-p*q%12:-1])

    lis = []
    for m in nveau:
        for i in range(len(m)):
            lis.append(m[i])
    finalbin = pl.reshape(pl.array(lis),(p,q))
    return imaff(finalbin)
```

```
223 # TESTS HAMMING
224
225 def modifimcorrh(im,pr):
226     bin = binim(im)
227     (p,q) = bin.shape
228     ligne = pl.ravel(pl.reshape(bin, (1,p*q)))
229     nveau = []
230     for i in range(0,int(p*q)-3,4):
231
232         bloc = ligne[i:i+4]
233         codeb = codeh(bloc)
234
235         modifvect(codeb,pr)
236         deco = decodeh(codeb)
237         nveau.append(deco)
238     nveau.append(ligne[p*q-1:p*q-1-p*q%4:-1])
239
240     lis = []
241     for m in nveau:
242         for i in range(len(m)):
243             lis.append(m[i])
244     finalbin = pl.reshape(pl.array(lis),(p,q))
245     return imaff(finalbin)
246
247
248
249 # COMPARAISON
250
251 def compare(im,pr):
252     pl.figure()
253     pl.imshow(imaff(binim(im)))
254     pl.show()
255     pl.figure()
256     pl.imshow(modifim(im,pr))
257     pl.show()
258     pl.figure()
259     pl.imshow(modifimcorr(im,pr))
260     pl.show()
261     pl.figure()
262     pl.imshow(modifimcorrh(im,pr))
263     pl.show()
264
265
266 compare(panda,0.03)
267
268 print("temps total :")
269 print(time.time()-tot)
```