Codes Correcteurs

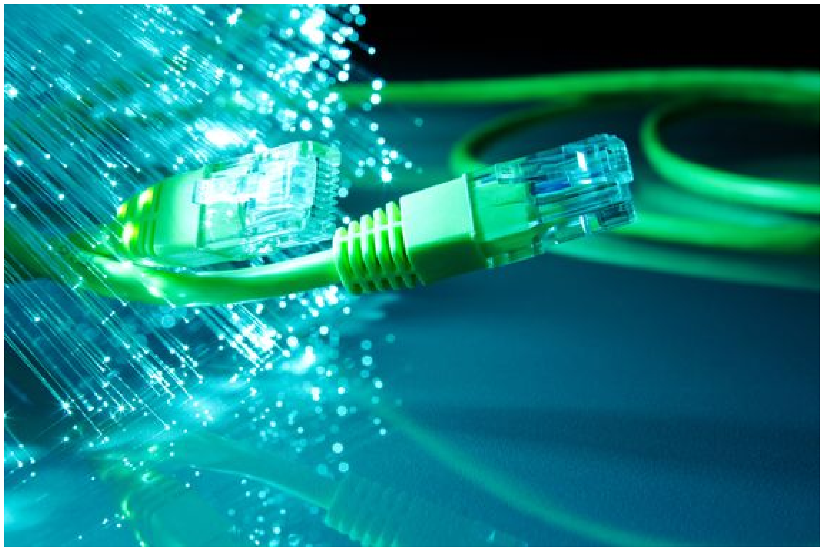
Théorie et exemple :
étude du code de Hamming

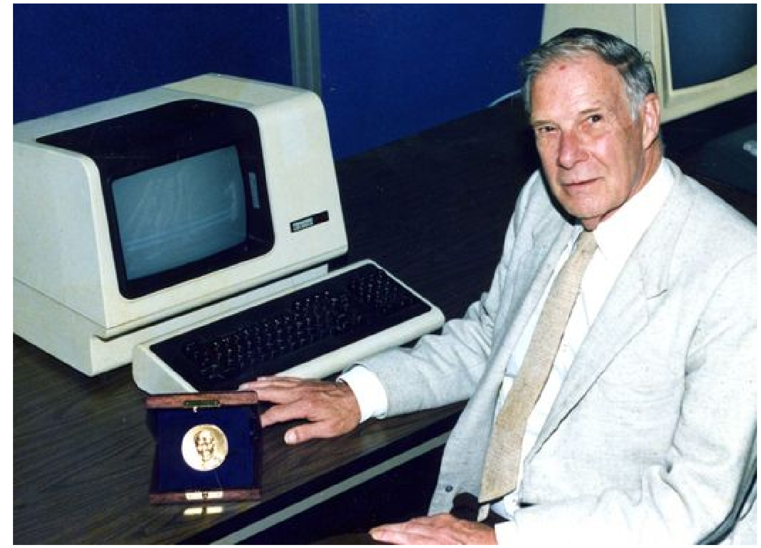
Sommaire

- Introduction
- Principe des codes correcteurs
- Codes de Hamming
    - Détection d’une erreur
    - Correction d’une erreur
    - Contrôles de parité
    - Exemple : Code de Hamming(7,4)
    - Code étendu

- Aspect code linéaire
  - Matrices utiles
  - Correction d’une erreur
  - Code parfait
- Algorithme rapide
- Implémentation
  - Exécution standard
  - Comparaisons
- Évolutions
- Conclusion
- Bibliographie

Introduction





Richard Hamming (1915 - 1998)

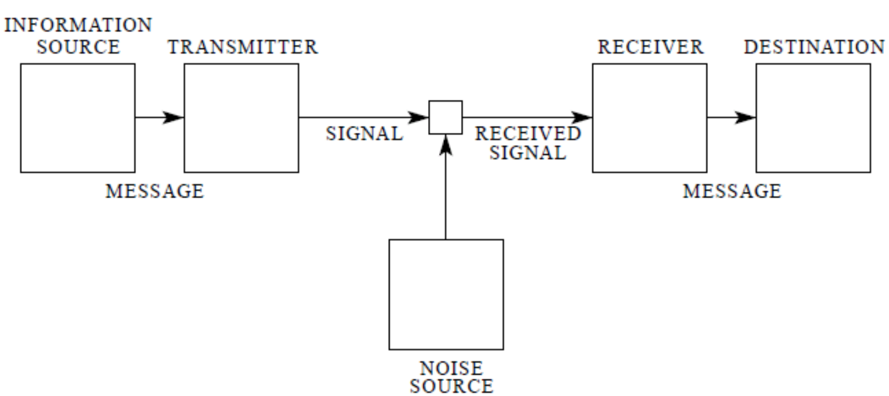
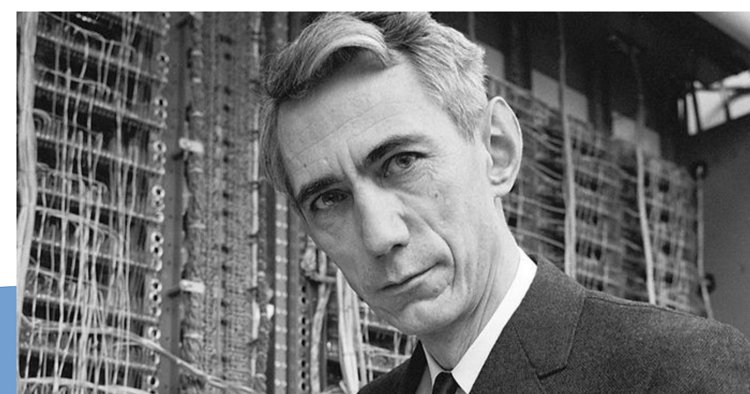


Fig. 1—Schematic diagram of a general communication system.



Claude Shannon (1916 - 2001)

Principe des codes correcteurs

- m $\rightarrow$ nombre de bits du message initial à transmettre
- n $\rightarrow$ taille du message final transmis
- Complexité en temps et en taille
- Redondance : R = n/m
- Capacité de détection et de corrections des erreurs

Détection d'une erreur

Introduction d'une somme de contrôle :

$M = \overline{x_0 x_1 \ldots x_{m-1}}^2 \quad N = M + x_m$

$x_0 + x_1 + \ldots + x_{m-1} + x_m = 0$

- Si un nombre impair de bit est changé :

$x_0 + x_1 + \ldots + x_m + x_{m+1} \neq 0$

- Si un nombre pair de bit est changé :

$x_0 + x_1 + \ldots + x_m + x_{m+1} = 0$

- Redondance : $R = \frac{m+1}{m}$

Correction d’une erreur

Principe généralisable :
k = nombre de contrôle de parité
= nombre de bits de parité ajoutés au message original

S = nombre de contrôle, représente la position de l’erreur (aussi appelé syndrome)
= concaténation des résultats des k contrôles de parité

Relations : $2^k \geq m+k+1$

Sachant que n=m+k , on a : $2^m \leq \frac{2^n}{n+1}$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
m	0	0	1	1	2	3	4	4	5	6	7	8	9	10	11	11
k	1	2	2	3	3	3	3	4	4	4	4	4	4	4	4	5
R	/	/	3	4	2,5	2	1,75	2	1,8	1,67	1,57	1,5	1,44	1,4	1,36	1,45

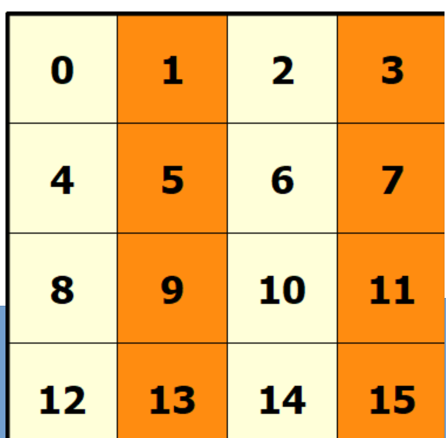
Description d’un code de Hamming par les paramètres (n,m,δ)

Contrôles de parité

$$N=\overline{x_0 x_1 \dots x_n}^{2}$$

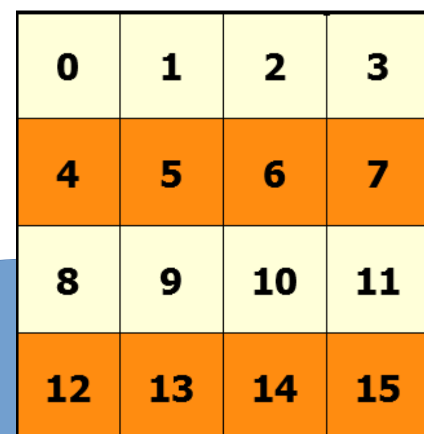
Principe : La valeur du i-ème contrôle indique la valeur du i-ème bit de l'écriture binaire de la position de l'erreur

Numéro du contrôle	Position du contrôle	Positions contrôlées
1	1	1,3,5,7,9,11,13,15,17 …
2	2	2,3,6,7,10,11,14,15,18 …
3	4	4,5,6,7,12,13,14,15,20 …
4	8	8,9,10,11,12,13,14,15,24 …
…	…	…

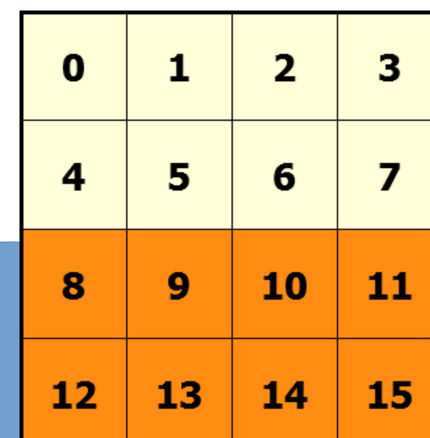


$x_1 + x_3 + x_5 + \dots = 0$

$x_2 + x_3 + x_6 + \dots = 0$

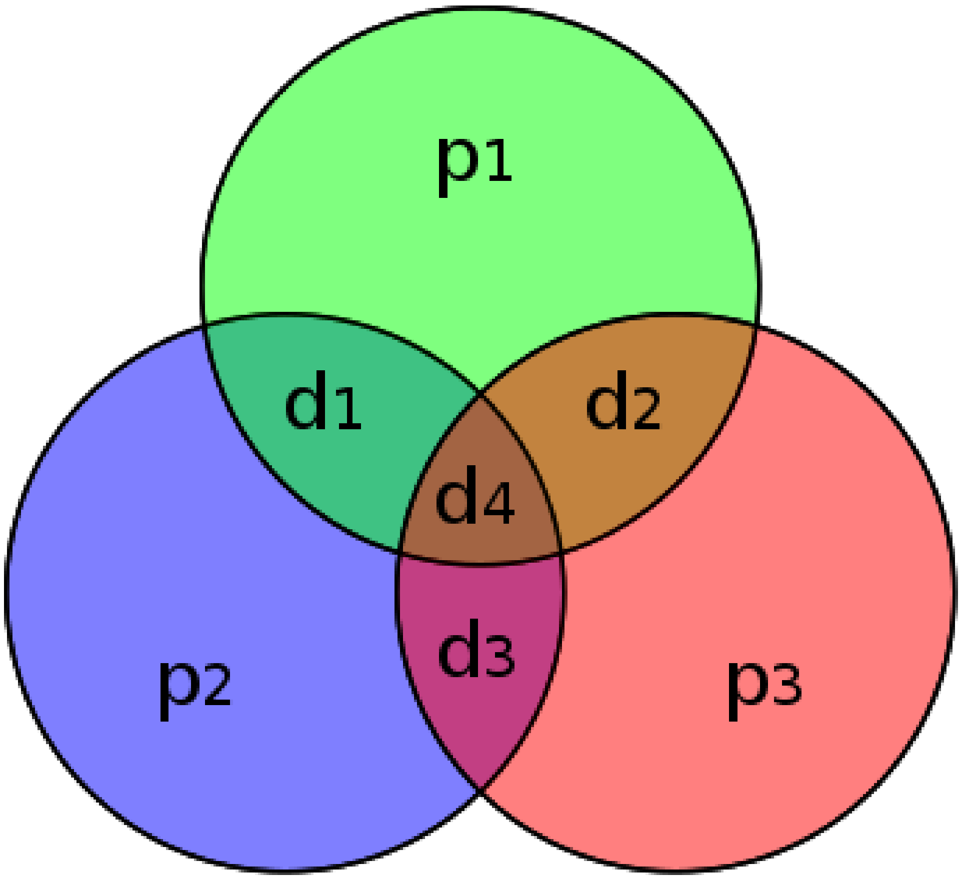


$x_4 + x_5 + x_6 + \dots = 0$



$x_8 + x_9 + x_{10} + \dots = 0$

Exemple : Code de Hamming (7,4)



Codage du message :

- M = 0110 ; $N = \overline{p_1 p_2 d_1 p_3 d_2 d_3 d_4}^2$

p1 + d1 + d2 + d4 = 0 => p1 = 1

p2 + d1 + d3 + d4 = 0 =>p2 = 1

p3 + d2 + d3 + d4 = 0 => p3 = 0

N = 1100110

Introduction d'une erreur :

- e = 0000001 ; N' = N + e = 1100111

p1 + d1 + d2 + d4 = 1 + 0 + 1 + 1 =1

p2 + d1 + d3 + d4 = 1 + 0 + 1 + 1 = 1

p3 + d2 + d3 + d4 = 0 + 1 + 1 + 1 = 1

=> Erreur en 7ème position (111 en binaire)

Crédits : Wikipédia

Code étendu

Ajout du contrôle global de parité : $x_0 + x_1 + \ldots + x_n = 0$

Pour n = 8 : $M = \overline{p_0\, p_1\, p_2\, d_1\, p_3\, d_2\, d_3\, d_4}^2$

Exemple avec deux erreurs : N = 01100110
e1 = 00010000, e2 = 00001000
N’ = N + e1 + e2 = 01111110

Code classique:

p1 + d1 + d2 + d4 = 1 + 1 + 1 + 0 =1

p2 + d1 + d3 + d4 = 1 + 1 + 1 + 0 = 1

p3 + d2 + d3 + d4 = 1 + 1 + 1 + 0 = 1

=> Erreur en position 7 (111 en binaire)

Code étendu:

p0 + p1 + p2 + p3 + d1 + d2 + d3 + d4
= 0 + 1 + 1 + 1 + 1 + 1 + 1 + 0 = 0

=> deux erreurs détectées !

Influence de la taille des messages

k	1	2	3	4	5	6	7	8	9
n	1	3	7	15	31	63	127	255	511
R	/	3	1,75	1,36	1,19	1,11	1,06	1,03	1,02

=> Meilleur rapport de redondance

Le taux d’erreurs varie en pratique de 10-4 (ligne téléphonique) à 10-9 (réseaux locaux), voire jusqu’à 10-12 pour des fibres optiques standards.

Aspect code linéaire

Espace de départ : $E=F_2^m$ Espace d'arrivée : $F=F_2^n$ $\varphi : E \to F$

Espace des mots du code : $C=\varphi(E)$

- φ est une application linéaire injective
- F est muni de la distance de Hamming
  - Dérive d'une pseudo-norme ω, le poids de Hamming
  - Symétrie : $\forall (x,y)\in F^2 \ d(x,y)=d(y,x)$ $\forall (x,y)\in C^2 \ d(x,y)=\omega(x-y)$
  - Séparation: $\forall (x,y)\in F^2 \ d(x,y)=0 \Leftrightarrow x=y$
  - Inégalité triangulaire : $\forall (x,y,z)\in F^3 \ d(x,z) \le d(x,y)+d(y,z)$

Soient x, y, z dans F.

On note $x=x_1 \dots x_n$, $y=y_1 \dots y_n$ et $z=z_1 \dots z_n$.

$U=\{x_i \neq z_i\}, S=\{(x_i\neq z_i)\wedge(x_i=y_i)\}, T=\{(x_i\neq z_i)\wedge(x_i\neq y_i)\}$.

$S\cup T=U$ et $S\cap T=\emptyset$.

$d(x,z)=|U|=|T|+|S|$ et $|T|\le d(x,y)$ et $|S|\le d(y,z)$

Exemples : ω(01010101) = 4 ; ω(0000) = 0 ;
d(1111,0101) = 2

Matrices utiles :

- Matrice génératrice :

$$\forall x \in E \quad G.x = \varphi(x) \in C \subset F$$

$$\varphi(1000) = 11110000$$
$$\varphi(0100) = 11001100$$
$$\varphi(0010) = 10101010$$
$$\varphi(0001) = 01101001$$

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

- Matrice de contrôle :

$$x = \overline{x_0 x_1 \ldots x_{n-1}}^2$$

$$\forall x \in F \quad H.x^T = 0 \Leftrightarrow \ x \in \mathrm{Ker}(H) \ \Leftrightarrow x \in C$$

$$\forall x \in F \quad H.x^T = S$$

$$x_0 + x_1 + x_2 + \ldots = 0$$
$$x_1 + x_3 + x_5 + \ldots = 0$$
$$x_2 + x_3 + x_6 + \ldots = 0$$
$$\ldots$$

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

On remarque que les colonnes de H sont les inverses des représentations binaires des chiffres de 0 à 7 avec un 1 devant
→ Permet de la construire facilement

On a la relation : H.G = 0

Correction d’une erreur

M’ = M + e
H.M’ = H.M + H.e = H.e, ce qui donne une colonne de H.
Le numéro de cette colonne donne la position de l’erreur que l’on peut alors corriger.

Par exemple avec M = 01100110:
e = 00001000 donne M’ = 01101110

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

H.M’ = H.e = $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

5ème colonne de H => erreur en 5ème position

Avec deux erreurs: M’ = M + e1 + e2
H.M’ = H.M + H.e1 + H.e2 = H.e1 + H.e2 = $\begin{pmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$ avec $(x_1, x_2, x_3) \neq (0,0,0)$
=> détection de deux erreurs

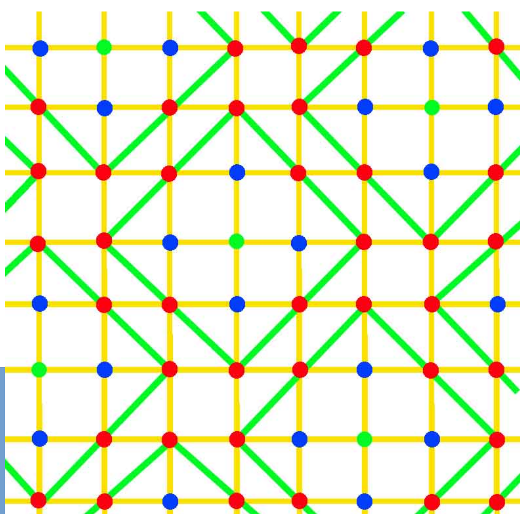
Code Parfait

- δ = distance minimale entre 2 mots du code
- Capacité de correction t = $\lfloor \delta/2 \rfloor$
- Borne de Hamming = $M \leqslant \frac{q^n}{V_t}$ où $V_t = \sum_{k=0}^{t} \binom{n}{k} (q-1)^k$ est le nombre d'éléments dans chaque boule fermée de rayon t centrée sur un mot du code (q = nombre d'éléments du corps fini, ici : q= 2)
- Code Parfait : La borne de Hamming est atteinte

<=> Les boules précédentes forment une partition de l'espace d'arrivée

c'est à dire: Le code est le plus compact possible

Exemple : Code de Hamming (7,4,3)



Représentation d'un code parfait avec δ = 5:
-Vert : Messages corrects
-Bleu : Messages erronés (1 erreur)
-Rouge : Messages erronés (2 erreurs)

Crédits : Wikipédia

Algorithme rapide

- $C = \overline{p_k \cdots p_1}^2$ ou-exclusif (XOR) des positions du message où il y a un coefficient 1
  - Exemple : N = 0110, M' = 00000110 $M = \overline{p_0 p_1 p_2 d_1 p_3 d_2 d_3 d_4}^2$

    p3p2p1 = 101 $\oplus$ 110 = 011 d'où M'' = 01100110

    puis après calcul du bit de parité général: M= 01100110

Correction : Même processus permet de trouver le syndrome C, qui donne la position de l'erreur

- Exemple : E = 00000001, M' = 01100111
  p3p2p1 = 001 $\oplus$ 010 $\oplus$101 $\oplus$ 110 $\oplus$ 111 = 111
  d'où M = 01100110

Implémentation

- Structure :
  - Classe de base pour implémenter les fonctions communes
  - Sous-classes pour implémenter les différentes méthodes

```
class HammingCodeBase
    def __init__
    def getBinaryRepresentation
    def parityCheck
    def cutDataInChunks
    def calculateGeneratorMatrix
    def calculateParityBitsValue
    def getEncryptedDataChunk
    def getEncryptedDataChunkWithGeneratorMatrix
    def getDecryptedHammingBlock
    def encodeData
    def encodeDataWithGeneratorMatrix
    def decodeBlocks
```

```
class HammingCode
    def __init__
    def calculateParityMatrix
    def insertParityBits
    def removeParityBits
    def correctParityBits
    def ensureValidityAndCorrect
```

```
class ExtendedHammingCode
    def __init__
    def calculateParityMatrix
    def insertParityBits
    def removeParityBits
    def correctParityBits
    def ensureValidityAndCorrect
```

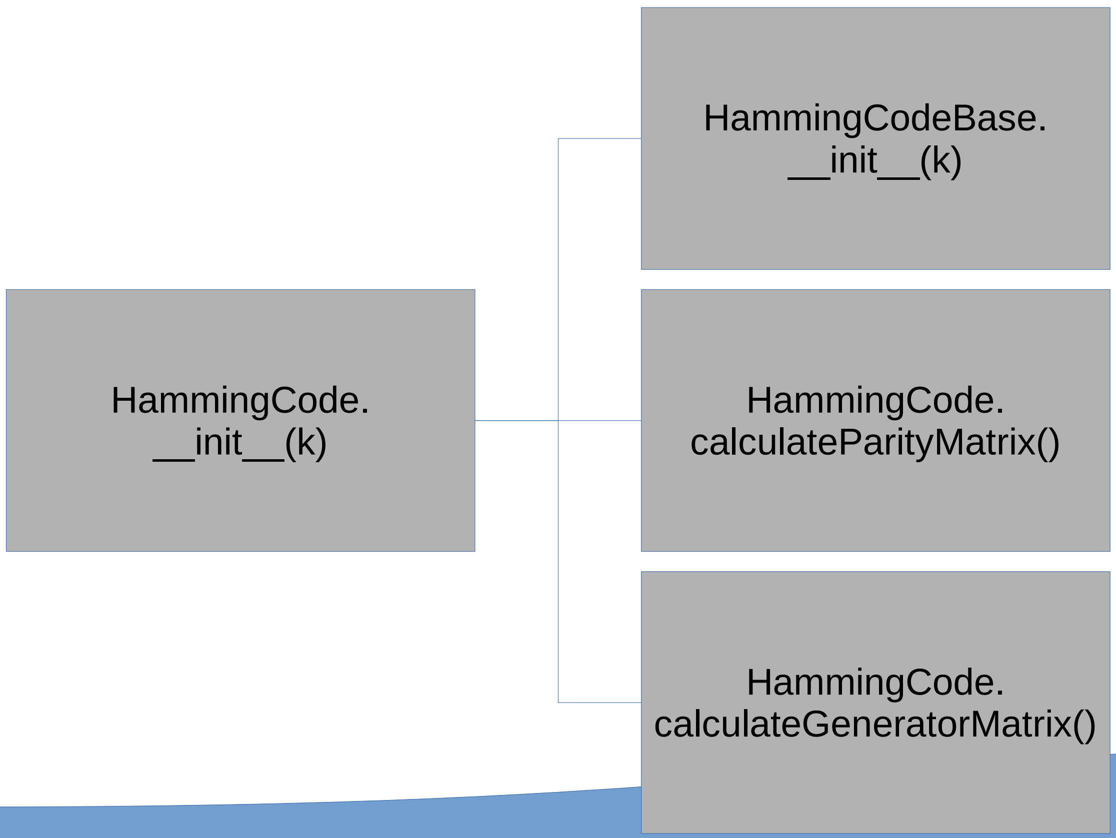
```
class HammingCodeXOR
    def __init__
    def insertParityBits
    def removeParityBits
    def correctParityBits
    def ensureValidityAndCorrect
    def binaryListXOR
    def getEncryptedDataChunk
    def encodeData
    def getDecryptedHammingBlock
    def decodeBlocks
```

  - Classe utilitaire de matrice binaires :

```
class Matrix
    def __init__
    def fromArray
    def __getitem__
    def __str__
    def __eq__
    def setToNull
    def getTransposed
    def transpose
    def getHeight
    def getWidth
    def multiply
    def toColumn
    def changeLine
    def searchColumn
    def multiplyLine
    def multiplyLineWithColumn
```

Exécution standard

- Initialisation :



Exécution standard

- Encodage d’un bloc :

HammingCode.
encodeData(data)

HammingCodeBase.
cutDataInChunks(data)

HammingCodeBase.
getEncryptedDataChunk(data_chunk)

HammingCode.
insertParityBits(data_chunk)

HammingCode.
correctParityBits(data_chunk)

- Décodage d’un bloc :

HammingCode.
decodeBlocks(blocks)

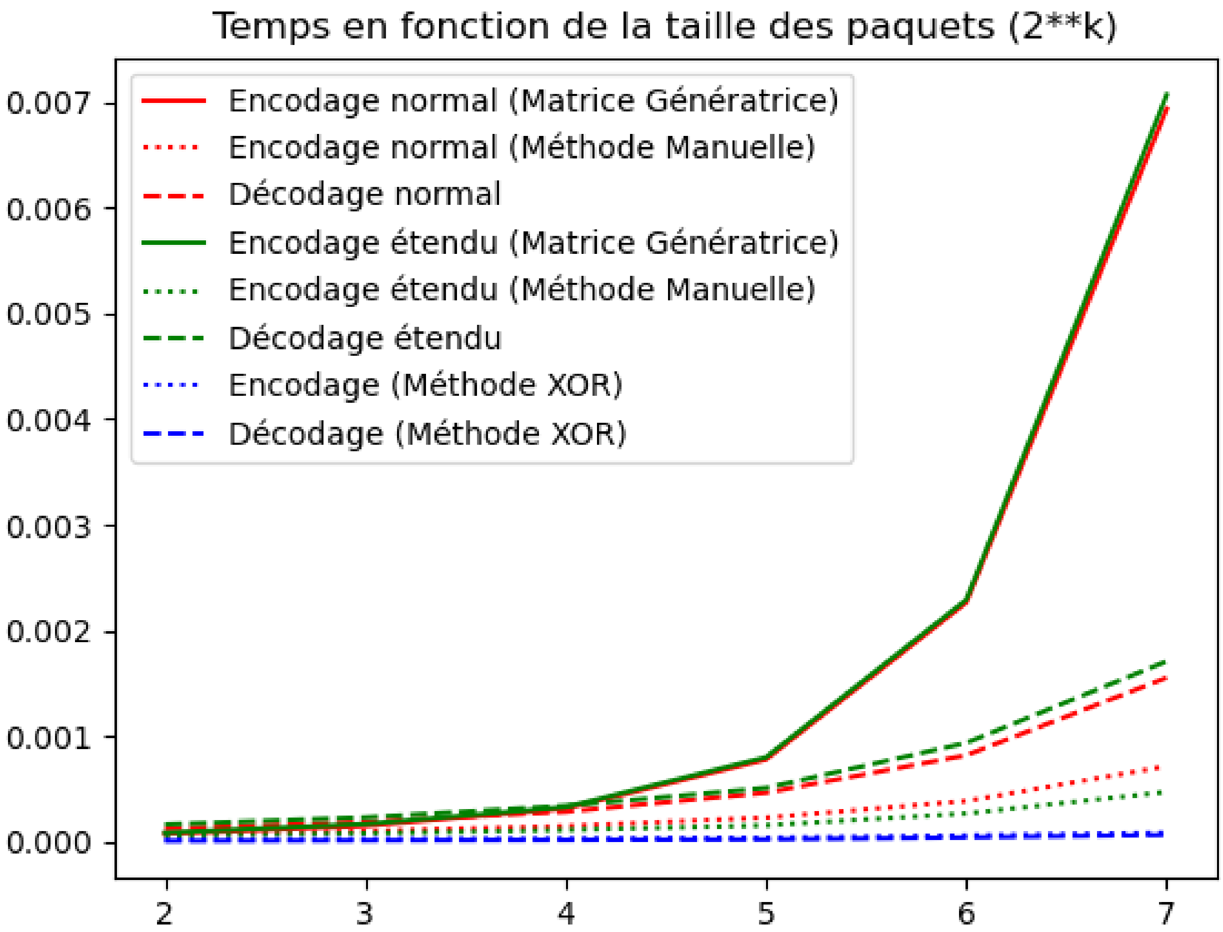
HammingCodeBase.
getDecryptedHammingBlock(block)

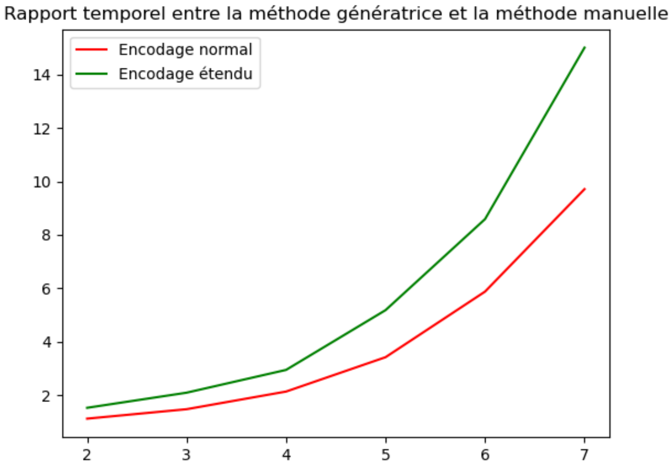
HammingCode.
ensureValidityAndCorrect(block)

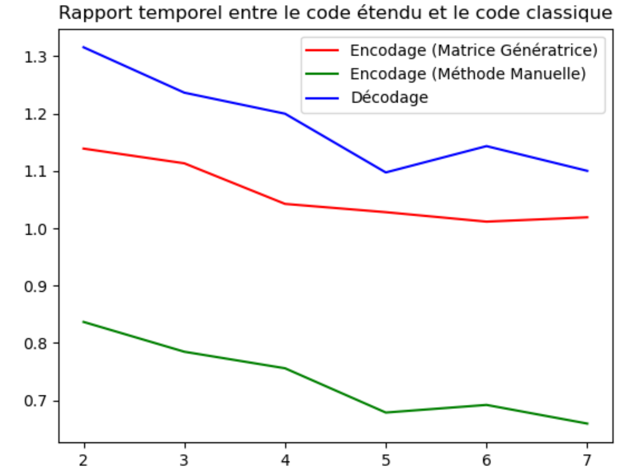
HammingCode.
calculateParityBitsValue(block)

HammingCode.
removeParityBits(block)

Comparaisons :

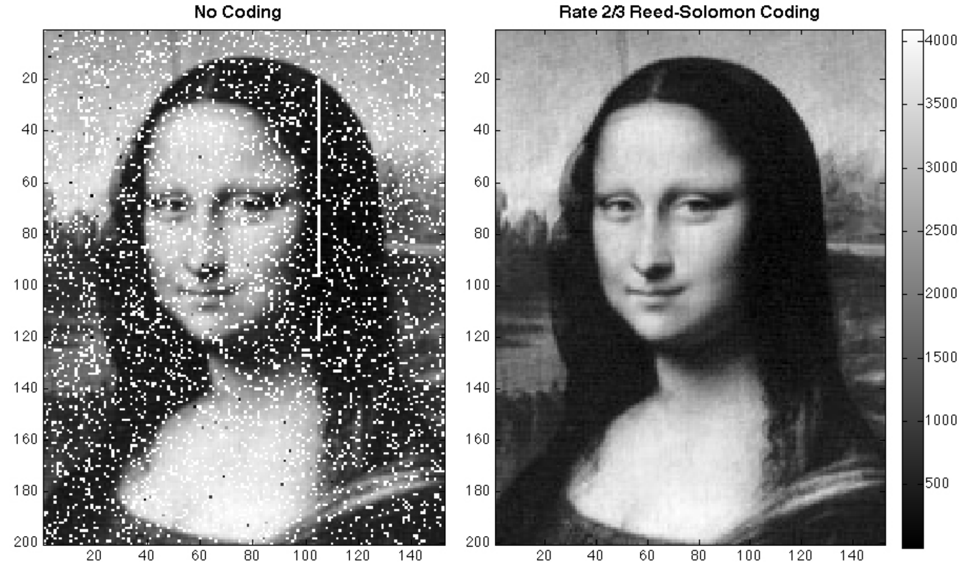






Évolutions :

- Code Polynomial : Ajout d’une structure d’algèbre à l’espace vectoriel F
  - Principe : Voir les bits comme les coefficients d’un polynôme de $F_2^n[X]$
- Exemple : Reed-Solomon
  - Utilisations :
    - DVD, CD,Transmission satellite, ADSL
  - Capacité:
    - DVD : 2 erreurs/32 bits;
      4 effacements/32 bits



Crédits : Xiaoli Sun, NASA Goddard — NASA Lunar Science page

- Codes à contrôle de redondance cyclique (CRC) : Détection d'erreurs
  - Utilisation : vérification de la validité d'un téléchargement (utilisé sur Linux)

Conclusion

Bibliographie

- Claude Shannon, A mathematical theory of communication, Bell System Technical Journal, vol. 27, pp. 379-423 and 623-656, July and October 1948
- Richard Hamming, Error-detecting and error-correcting codes, Bell System Technical Journal vol. 29 , pp. 147-160, April 1950
- J. H. van Lint, Introduction to Coding Theory, Springer, 1982