Codes correcteurs d'erreurs

Algorithmes avancés - ITI orientation logiciel et systèmes complexes

Paul Albuquerque
HES-SO//Genève, hepia

Table des matières

1 Introduction

Le programme Mariner était une série de missions spatiales de la NASA ayant pour objectif d'envoyer des sondes spatiales afin d'étudier les planètes Mars, Vénus et Mercure. La sonde Mariner 9 fut lancée vers Mars le 30 mai 1971 et atteignit la planète le 14 novembre de la même année. Elle pris de nombreuses photos de la surface et cartographia la planète. Les images étaient en nuances de gris, chaque pixel étant représenté par un entier entre 0 et 63 (soit l'équivalent de 6 bits).

Lors de la transmission des images en provenance de Mars, le brouillage du message durant leur voyage intersidéral fut tel que si on avait envoyé naivement ceci :

FIGURE 1 – Image de la planète Mars envoyée vers le Terre

on aurait reçu cela :

FIGURE 2 – Image brouillée de la planète Mars reçue sur Terre.

Heureusement, grâce aux codes correcteurs d'erreurs, on peut filtrer le bruit des images et obtenir des images nettes. Le principe est de donner une certaine redondance au message transmis pour que l'altération de quelques bits ne remettent pas en cause l'intégrité du message ; on pourrait par exemple répéter plusieurs fois le message, mais ce n'est pas optimal.

2 Quelques remarques sur les codes

Le message d'information (c.-à-d. un pixel sur l'exemple de l'image) est constitué de k bits et on veut le coder sur n bits.

On note $\Omega = \{0, 1\}$ et on définit l'ensemble Ω^l des mots de l bits par

$$\Omega^l = \{\bar{x} = (x_0\ x_1 \dots x_{l-1}) \text{ tels que } x_i \in \{0, 1\} \text{ pour } i = 0, \dots, l-1\}$$

La notation avec le surlignage $\bar{x}$ indique qu'on considère un mot formé de bits. On utilisera aussi les termes de chaîne ou vecteur de bits.

L'ensemble Ω^l est un espace vectoriel. En effet, la famille $(\bar{e}_i)_{i=0,\dots,l-1}$ où

$$\bar{e}_i = (0\ \dots 0\ \underset{\substack{\uparrow \\ i}}{1}\ 0\ \dots 0) \in \Omega^l, \text{ pour } i = 0, \dots, l-1$$

forme une base de Ω^l.

Ainsi tout élément $\bar{x} = (x_0\ x_1 \dots x_{l-1})$ de Ω^l est une combinaison des éléments de la base. Donc

$$\bar{x} = \sum_{i=0}^{l-1} x_i \bar{e}_i$$

Soit la fonction *dec* qui permet d'associer à un $\bar{x} = (x_0\ x_1 \dots x_{l-1}) \in \Omega^l$, un entier entre 0 et $2^l - 1$. Elle est donnée par

$$\begin{aligned} dec \ &: \ \Omega^l \to \{0, 1, \dots, 2^l - 1\} \subset \mathbb{N} \\ &\bar{x} \mapsto x = \sum_{i=0}^{l-1} x_i 2^i \end{aligned}$$

On notera $bin : \{0, 1, \dots, 2^l - 1\} \to \Omega^l$ l'opération inverse de *dec*, et qui produit l'écriture binaire d'un entier. Cette correspondance bijective permet d'écrire que $\Omega^l = \{0, 1, \dots, 2^l - 1\}$ selon qu'on envisage les éléments comme des chaînes de bits ou des entiers.

Le codage d'un élément de Ω^k (par exemple un pixel) sur n bits ($k < n$), c.-à-d. en un élément de Ω^n, nécessite une application

$$\begin{aligned} c \ &: \ \Omega^k \to \Omega^n \\ &\bar{x} \mapsto \bar{y} = c(\bar{x}) \end{aligned}$$

L'ensemble $C = c(\Omega^k) = \{\bar{y} \in \Omega^n \text{ tel que } \bar{y} = c(\bar{x}) \text{ pour un } \bar{x} \in \Omega^k\} \subset \Omega^n$ constitue le code associé à c. Un code est de bonne qualité si ces éléments

sont éloignés les uns des autres dans Ω^n et s'il utilise relativement peu de bits. Ainsi, d'une part un élément de C légèrement perturbé, restera proche de son état original, et d'autre part le temps de transmission du message ne sera pas trop rallongé par l'ajout de bits de redondance.

2.1 Distance de Hamming

Pour mesurer la proximité de deux éléments de Ω^l selon le nombre de bits qu'ils ont en commun, on définit la distance de Hamming $d_H(\bar{u}, \bar{v})$ entre les éléments $\bar{u} = (u_0\ u_1 \dots u_{l-1}) \in \Omega^l$ et $\bar{v} = (v_0\ v_1 \dots v_{l-1}) \in \Omega^l$ par

$$d_H(\bar{u}, \bar{v}) = |\{i \in \{0, 1, \dots, l-1\} \text{ tel que } u_i \neq v_i\}|$$

A noter que

$$d_H(\bar{u}, \bar{v}) = d_H(\bar{u} + \bar{v}, \bar{0}) = \sum_{i=0}^{l-1} (\bar{u} + \bar{v})_i$$

où l'opération "+" effectue un *xor* bit par bit et $\bar{0} = (0 \dots 0)$.

Par exemple, pour $\bar{u} = (0\ 1\ 0\ 0\ 1\ 1), \bar{v} = (0\ 1\ 1\ 0\ 1\ 0) \in \Omega^6$, on a $d_H(\bar{u}, \bar{v}) = 2$.

La distance minimale d'un code C dans Ω^n est définie par

$$d_H(C) = \min\{d_H(\bar{u}, \bar{v}) \text{ tel que } \bar{u} \neq \bar{v} \in C\}$$

Le code C est dit linéaire si C est un sous-espace vectoriel de Ω^n (ceci équivaut à $\bar{u} + \bar{v} \in C$, si $\bar{u}, \bar{v} \in C$, et $\bar{0} \in C$).

Si le code C est linéaire, alors $d_H(C) = \min\{d_H(\bar{u}, \bar{0}) \text{ tel que } \bar{0} \neq \bar{u} \in C\}$.

2.2 Principe du débruitage

Si le message original $\bar{x} \in \Omega^k$ est codé en $\bar{y} = c(\bar{x}) \in C \subset \Omega^n$ et que $\bar{y}$ est transmis à travers un canal bruité, alors $\bar{y}$ peut être brouillé. Ainsi, le message codé $\bar{y} \in C$ à l'entrée du canal peut ne pas correspondre au message codé et brouillé $\bar{z} \in \Omega^n$ reçu à la sortie du canal. Le canal peut donc introduire un certain nombre d'erreurs dans $\bar{y}$ (c.-à-d. modifier des bits de $\bar{y}$) pour produire un $\bar{z} \notin C$.

Pour enlever le bruit responsable de la transformation de $\bar{y} \in C$ en $\bar{z} \notin C$ et retrouver $\bar{y}$, on recherche l'élément de C le plus proche de $\bar{z}$. Si le code est

adapté au canal, c.-à-d. s'il est capable de corriger toutes les erreurs dues au bruit, alors on retrouve le $\bar{y}$ introduit dans le canal :

$$d_H(\bar{z}, \bar{y}) = \min\{d_H(\bar{z}, \bar{u}) \text{ avec } \bar{u} \in C\}$$

Il ne reste plus qu'à décoder $\bar{y}$ pour obtenir le message original (c.-à-d. avant codage et brouillage) $\bar{x} = c^{-1}(\bar{y}) \in \Omega^k$. On constate qu'il est important d'avoir à disposition une méthode rapide pour filtrer le bruit.

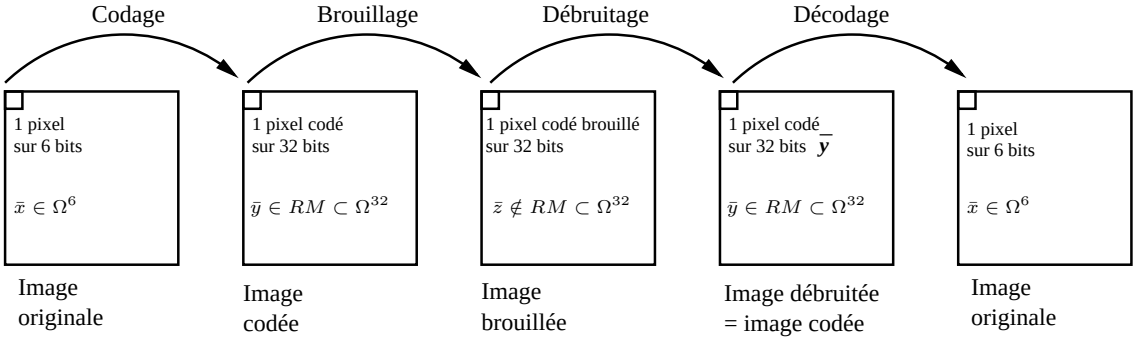

FIGURE 3 – Illustration pour la transmission dans un canal bruité d'une image dont les pixels sont stockés sur 6 bits et codés sur 32 bits à l'aide d'un code de Reed-Muller $RM(1, 5)$.

Deux critères interviennent dans l'efficacité d'un code C :

- d'un côté, $d_H(C)$ doit être le plus grand possible (pour que les éléments soient le plus loin possible les uns des autres) ;
- d'un autre côté, le rapport k/n doit être le plus proche de 1 possible (car on voudrait coder un maximum d'information avec le moins de bits possible).

3 Codes de Reed-Muller du 1er ordre de longueur r

Les codes de Reed-Muller du 1er ordre de longueur r, noté $RM(1, r)$ ou plus simplement RM, sont des codes linéaires de dimension $r + 1$ dans Ω^{2^r}. On note $rm : \Omega^{r+1} \to \Omega^{2^r}$ la fonction de codage.

Ces codes sont très utiles pour la transmission sur des canaux très brouillés. Le code $RM(1, r)$ a en particulier une distance minimale de 2^{r-1} et peut donc

corriger jusqu'à $2^{r-2}-1$ erreurs. C'est un code $RM(1,5)$ qui a été employé pour la transmission d'images de Mars, via le satellite Mariner-9. Chaque pixel était stocké sur 64 niveaux de gris (c.-à-d. sur 6 bits) et transmis au moyen d'un mot de code de longueur $2^5 = 32$ dont on pouvait corriger jusqu'à 7 erreurs.

On peut définir le code $rm : \Omega^{r+1} \to \Omega^{2^r}$ en donnant l'image des éléments de la base canonique de Ω^{r+1}.

Le codage $rm : \Omega^{r+1} \to \Omega^{2^r}$ et une base $(\bar{b}_i)_{i=0,\dots,r}$ du sous-espace vectoriel RM de dimension $r+1$, sont obtenus par

$$\begin{aligned}
\bar{b}_0 &= rm(\underbrace{1\,0\dots0}_{r+1}) = (\underbrace{0\,1\,0\,1\dots0\,1}_{2^r}) \\
\bar{b}_1 &= rm(0\,1\,0\dots0) \\
&= (0\,0\,1\,1\,0\dots1\,0\,0\,1\,1) \\
\bar{b}_2 &= rm(0\,0\,1\,0\dots0) \\
&= (0\,0\,0\,0\,1\,1\,1\,1\,0\dots1\,0\,0\,0\,0\,1\,1\,1\,1) \\
&\vdots \\
\bar{b}_{r-1} &= rm(0\dots0\,1\,0) \\
&= (0\,0\dots0\,1\,1\dots1) \\
\bar{b}_r &= rm(0\dots0\,1) = (1\,1\dots1)
\end{aligned}$$

Pour $k = 0, \dots, r-1$, le i-ème bit de $\bar{b}_k$ est égal au k-ème bit de i. L'élément $\bar{b}_r$ a tous ses bits à 1. Le code $RM = rm(\Omega^{r+1})$ est constitué de 2^{r+1} éléments obtenus par combinaison linéaire des éléments de la base.

3.1 Codage

Si $\bar{x} = (x_0 \dots x_r) \in \Omega^{r+1}$, alors $\bar{y} = rm(\bar{x}) = \sum_{i=0}^{r} x_i \bar{b}_i \in \Omega^{2^r}$.

Donc

$$RM = \left\{ \bar{y} = \sum_{i=0}^{r} x_i \bar{b}_i \text{ avec } \bar{x} = (x_0 \dots x_r) \in \Omega^{r+1} \right\}$$

On peut définir la matrice génératrice G du code RM comme étant la matrice $(r+1) \times 2^r$ dont les lignes sont les $\bar{b}_i, i = 0, \dots, r$.

En écriture matricielle, on a

$$RM = \left\{ \bar{y} = \bar{x}G \text{ avec } \bar{x} \in \Omega^{r+1} \right\}$$

Exemple avec $r = 3$ et $rm : \Omega^4 \to \Omega^{2^3}$

La base de $RM = rm(\Omega^4)$ est

$$\begin{aligned}
\bar{b}_0 &= (0\,1\,0\,1\,0\,1\,0\,1)\\
\bar{b}_1 &= (0\,0\,1\,1\,0\,0\,1\,1)\\
\bar{b}_2 &= (0\,0\,0\,0\,1\,1\,1\,1)\\
\bar{b}_3 &= (1\,1\,1\,1\,1\,1\,1\,1)
\end{aligned}$$

Un message $\bar{x} = (x_0\ x_1\ x_2\ x_3) \in \Omega^4$ est codé en

$$\bar{y} = rm(\bar{x}) = x_0\bar{b}_0 + x_1\bar{b}_1 + x_2\bar{b}_2 + x_3\bar{b}_3 \in \Omega^8$$

Le nombre d'éléments du code est $|RM(1,3)| = 2^4 = 16$.

La matrice génératrice G de $RM(1,3)$ est donnée par

$$G = \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \left(\begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right) \end{array}$$

On peut remarquer que chaque colonne de G contient l'écriture binaire du numéro de la colonne (en omettant la dernière ligne).

Sous forme matricielle $\bar{y} = \bar{x}G$ ou plus explicitement

$$(y_0\ y_1\ y_2\ y_3\ y_4\ y_5\ y_6\ y_7) = (x_0\ x_1\ x_2\ x_3) \left(\begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right)$$

On a listé les éléments du code $RM(1,3)$ dans la table 3.1.

La première colonne de la table 3.1 donne $\bar{x}$ en écriture décimale et les suivantes $\bar{x}$ en écriture binaire. On voit que les éléments du code $\bar{y}$ de la première moitié de la table ont $y_0 = 0$, et tous leurs bits sont inversés par rapport aux éléments de la seconde moitié. Par exemple, pour $\bar{x} = (0\ 1\ 1\ 0)$, on a $\bar{y} = rm(\bar{x}) = (0\ 0\ 1\ 1\ 1\ 1\ 0\ 0)$. L'élément correspondant dans la seconde moitié de la table est $\bar{x} = (0\ 1\ 1\ 1)$, pour lequel $\bar{y} = rm(\bar{x}) = (1\ 1\ 0\ 0\ 0\ 0\ 1\ 1)$.

$\bar{x}$	x_0	x_1	x_2	x_3	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	1	0	1	0	1
2	0	1	0	0	0	0	1	1	0	0	1	1
3	1	1	0	0	0	1	1	0	0	1	1	0
4	0	0	1	0	0	0	0	0	1	1	1	1
5	1	0	1	0	0	1	0	1	1	0	1	0
6	0	1	1	0	0	0	1	1	1	1	0	0
7	1	1	1	0	0	1	1	0	1	0	0	1
8	0	0	0	1	1	1	1	1	1	1	1	1
9	1	0	0	1	1	0	1	0	1	0	1	0
10	0	1	0	1	1	1	0	0	1	1	0	0
11	1	1	0	1	1	0	0	1	1	0	0	1
12	0	0	1	1	1	1	1	1	0	0	0	0
13	1	0	1	1	1	0	1	0	0	1	0	1
14	0	1	1	1	1	1	0	0	0	0	1	1
15	1	1	1	1	1	0	0	1	0	1	1	0

TABLE 1 – Table des éléments du code $RM(1,3)$

Exemple avec $r = 5$ et $rm : \Omega^6 \to \Omega^{2^5}$

La base de $RM = rm(\Omega^6)$ est

$$\begin{aligned}
\bar{b}_0 &= (0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1) \\
\bar{b}_1 &= (0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1) \\
\bar{b}_2 &= (0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1) \\
\bar{b}_3 &= (0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1) \\
\bar{b}_4 &= (0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1) \\
\bar{b}_5 &= (1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1)
\end{aligned}$$

Un message $\bar{x} = (x_0\ x_1\ x_2\ x_3\ x_4\ x_5) \in \Omega^6$ est codé en

$$\bar{y} = rm(\bar{x}) = x_0\bar{b}_0 + x_1\bar{b}_1 + x_2\bar{b}_2 + x_3\bar{b}_3 + x_4\bar{b}_4 + x_5\bar{b}_5 \in \Omega^{32}$$

Le nombre d'éléments du code est $|RM(1,5)| = 2^6 = 64$.

La matrice génératrice de $RM(1,5)$.

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

Sous forme matricielle $\bar{y} = \bar{x}G$ ou plus explicitement

$(y_0\ y_1 \dots y_{31}) =$

$$= (x_0\ x_1\ x_2\ x_3\ x_4\ x_5) \begin{pmatrix}
0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1 \\
0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\
0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \\
1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1
\end{pmatrix}$$

3.2 Décodage

Pour décoder un $\bar{y} \in RM \subset \Omega^{2^r}$, c.-à-d. retrouver sa préimage

$$\bar{x} = rm^{-1}(\bar{y}) \in \Omega^{r+1},$$

il faut déterminer l'écriture de $\bar{y}$ dans la base des $\bar{b}_i, i = 0, \dots, r$:

$$\bar{y} = rm(\bar{x}) = \sum_{i=0}^{r} x_i \bar{b}_i \in \Omega^{2^r}$$

On cherche donc la valeur des $x_i\,(i = 0, \dots, r)$:

— Observons tout d'abord que l'élément de la base $\bar{b}_r$ est le seul à avoir un 1 en position 0. Ainsi, $x_r = y_0$.
Il reste à déterminer $x_0, \dots, x_{r-1}$.

— On considère à présent

$$\bar{w} = \bar{y} + x_r \bar{b}_r = \sum_{i=0}^{r-1} x_i \bar{b}_i$$

Pour $i = 0, \dots, r-1$, l'élément de la base $\bar{b}_i$ est le seul à avoir un 1 en position 2^i. On en déduit $x_i = w_{2^i}$

Le décodage se fait donc en $\mathcal{O}(r)$ opérations. A noter qu'une recherche exhaustive prendrait $\mathcal{O}(2^{2r})$ opérations.

Exemple avec $r = 3$ et $rm : \Omega^4 \to \Omega^8$

Prenons $\bar{y} = (1\ 0\ 1\ 0\ 0\ 1\ 0\ 1)$. On cherche $\bar{x}$ tel que $\bar{y} = rm(\bar{x})$.

$$\begin{aligned}\bar{y} = (1\ 0\ 1\ 0\ 0\ 1\ 0\ 1) &= x_0(0\ 1\ 0\ 1\ 0\ 1\ 0\ 1)\\ &+ x_1(0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)\\ &+ x_2(0\ 0\ 0\ 0\ 1\ 1\ 1\ 1)\\ &+ x_3(1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)\end{aligned}$$

Comme $y_0 = 1$, on a forcément $x_3 = 1$ (c.-à-d. $x_3 = y_0$). On pose $\bar{w} = \bar{y} + x_3\bar{b}_3$.

$$\begin{aligned}\bar{w} &= (1\ 0\ 1\ 0\ 0\ 1\ 0\ 1)\\ &+(1\ 1\ 1\ 1\ 1\ 1\ 1\ 1) \qquad \text{0 1 2 3 4 5 6 7}\\ &= (0\ 1\ 0\ 1\ 1\ 0\ 1\ 0) = x_0(0\ 1\ 0\ 1\ 0\ 1\ 0\ 1)\\ &\qquad + x_1(0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)\\ &\qquad + x_2(0\ 0\ 0\ 0\ 1\ 1\ 1\ 1)\end{aligned}$$

En examinant les bits en position $1 = 2^0, 2 = 2^1$ et $4 = 2^2$, on voit qu'on doit avoir $x_0 = 1, x_1 = 0$ et $x_2 = 1$ (c.-à-d. $x_i = w_{2^i}$ pour $i = 0, 1, 2$). Ainsi, on a trouvé que $\bar{x} = (1\ 0\ 1\ 1)$ est la préimage de $\bar{y} = (1\ 0\ 1\ 0\ 0\ 1\ 0\ 1)$. On peut d'ailleurs vérifier que $\bar{y} = rm(\bar{x})$.

3.3 Débruitage d'un élément brouillé

Pour le code de $RM = RM(1, r)$, on peut montrer que la distance minimale $d_H(RM) = 2^{r-1}$.

Ceci signifie que les boules de rayon $e = 2^{r-2} - 1$ centrées sur les mots du codes sont 2 à 2 disjointes :

— pour $\bar{y} \neq \bar{\xi} \in RM$, $B_e(\bar{y}) \cap B_e(\bar{\xi}) = \emptyset$.

Ainsi le code $RM(1, r)$ permet de corriger jusqu'à $e = 2^{r-2} - 1$ erreurs. Par exemple, pour le cas $r = 5$, on a $e = 7$.

Pour rappel, si le message codé bruité qui est reçu à la sortie du canal de transmission est $\bar{z} \notin RM$, alors on cherche $\bar{y} \in RM$ tel que

$$d_H(\bar{z}, \bar{y}) = \min\{d_H(\bar{z}, \bar{\xi}) \text{ tel que } \bar{\xi} \in RM\}$$

On propose trois méthodes pour corriger les erreurs produites par le canal bruité.

3.3.1 Recherche exhaustive

On parcourt tous les éléments de RM et on retient celui qui est le plus proche du message codé bruité reçu.

La recherche exhaustive nécessite donc de calculer la distance aux 2^{r+1} éléments de RM.

Le calcul de la distance de Hamming entre deux éléments nécessite $\mathcal{O}(2^{r+1})$ opérations. Donc cette méthode est de complexité $\mathcal{O}(2^{2r+2})$.

3.3.2 Recherche semi-exhaustive

D'après les observations faites concernant la table 3.1 du code $RM(1,3)$, on peut réduire la recherche exhaustive de moitié. En effet, un élément $\bar{y}$ de la 1ère moitié de la table peut être associé à un élément $\bar{y}+\bar{b}_r$ de la 2ème moitié. Tous les bits de $\bar{y}+\bar{b}_r$ sont inversés par rapport à ceux de $\bar{y}$.

Ainsi, si $\bar{z} \in \Omega^{2^r}$ est un message codé bruité, alors $d_H(\bar{y}+\bar{b}_r, \bar{z}) = 2^r - d_H(\bar{y}, \bar{z})$.

Prenons par exemple

$$\bar{z} = (1\,0\,1\,1\,1\,1\,1\,1) \in \Omega^8 \text{ et } \bar{y} = (0\,1\,1\,0\,1\,0\,0\,1) \in RM(1,3).$$

On a $d_H(\bar{y}, \bar{z}) = 5$ et comme

$$\bar{y} + \bar{b}_r = (0\,1\,1\,0\,1\,0\,0\,1) + (1\,1\,1\,1\,1\,1\,1\,1) = (1\,0\,0\,1\,0\,1\,1\,0),$$

on peut calculer $d_H(\bar{y} + \bar{b}_r, \bar{z}) = 3$.
On voit qu'on a bien $d_H(\bar{y} + \bar{b}_r, \bar{z}) = 8 - d_H(\bar{y}, \bar{z})$.

Donc, le calcul des distances de Hamming pour les éléments de la 1ère moitié de la table, permet d'obtenir directement celles de la 2ème moitié. Ceci nécessite au total $\mathcal{O}(2^{2r+1})$. En comparaison avec les $\mathcal{O}(2^{2r+2})$ obtenu dans la section précédente, on a presque rien gagné.

3.3.3 Une reformulation de la distance de Hamming

L'ensemble des mots originaux Ω^{r+1} peut être partitionné en deux sous-ensembles disjoints : $\Omega^{r+1} = \Omega^r \times \{0\} \cup \Omega^r \times \{1\}$. Exprimé dans le système décimal, on a : $\{0, 1, \ldots, 2^{r+1}-1\} = \{0, 1, \ldots, 2^r-1\} \cup \{2^r, 2^r+1, \ldots, 2^{r+1}-1\}$.

On peut dès lors faire de même pour le code de Reed-Muller :

$$RM = RM_0 \cup RM_1$$

$$\begin{aligned}
\text{où} \quad RM_0 &= \{v_0\bar{b}_0 + \cdots + v_{r-1}\bar{b}_{r-1} \text{ tel que } \bar{v} = (v_0 \dots v_{r-1}) \in \Omega^r\} \\
&= rm(\Omega^r \times \{0\}) \\
\text{et} \quad RM_1 &= \{v_0\bar{b}_0 + \cdots + v_{r-1}\bar{b}_{r-1} + \bar{b}_r \text{ tel que } \bar{v} = (v_0 \dots v_{r-1}) \in \Omega^r\} \\
&= rm(\Omega^r \times \{1\})
\end{aligned}$$

On a $RM_1 = RM_0 + \bar{b}_r$ (rappelons que $\bar{b}_r = (1\ 1 \dots 1)$ est le dernier élément de la base de RM) ou similairement

$$\bar{y} \in RM_0 \quad \text{si } y_0 = 0 \quad \text{et} \quad \bar{y} \in RM_1 \quad \text{si } y_0 = 1$$

A noter que RM_0 désigne la 1ère moitié de la table du code (voir table 3.1) et RM_1 la 2ème moitié.

On définit la transformation $\hat{F}$ associée à un élément $z \in \Omega^{2^r}$ par

$$\begin{aligned}
&\hat{F} : RM \to \mathbb{Z} \\
&\bar{y} \mapsto \sum_{i=0}^{2^r-1} (-1)^{y_i}(-1)^{z_i} = \sum_{i=0}^{2^r-1} (-1)^{y_i+z_i}
\end{aligned}$$

$$\begin{aligned}
\text{On a} \quad \hat{F}(\bar{y}) &= \sum_{i=0}^{2^r-1} (-1)^{y_i+z_i} = |\{i \text{ tel que } y_i = z_i\}| - |\{i \text{ tel que } y_i \neq z_i\}| \\
&= d_H(\bar{y} + \bar{b}_r, \bar{z}) - d_H(\bar{y}, \bar{z}) = 2^r - d_H(\bar{y}, \bar{z}) - d_H(\bar{y}, \bar{z}) \\
&= 2^r - 2d_H(\bar{y}, \bar{z})
\end{aligned}$$

Donc minimiser la distance de Hamming $d_H(\bar{y}, \bar{z})$ pour débruiter l'élément z équivaut à maximiser la valeur de $\hat{F}(\bar{y})$.

D'autre part,

$$\begin{aligned}
d_H(\bar{y}, \bar{z}) &= \frac{1}{2}(2^r - \hat{F}(\bar{y})) \\
d_H(\bar{y} + \bar{b}_r, \bar{z}) = 2^r - d_H(\bar{y}, \bar{z}) &= 2^r - \frac{1}{2}(2^r - \hat{F}(\bar{y})) = \frac{1}{2}(2^r + \hat{F}(\bar{y}))
\end{aligned}$$

Reprenons l'exemple $\bar{z} = (1\ 0\ 1\ 1\ 1\ 1\ 1\ 1) \in \Omega^8$ et $\bar{y} = (0\ 1\ 1\ 0\ 1\ 0\ 0\ 1) \in RM(1,3)$. On a d'une part $d_H(\bar{y}, \bar{z}) = 5$, et d'autre part

$$\sum_{i=0}^{7} (-1)^{y_i+z_i} = -1 - 1 + 1 - 1 + 1 - 1 - 1 + 1 = -2 = 8 - 2d_H(\bar{y}, \bar{z})$$

Par ailleurs, $\bar{y} + \bar{b}_r = (1\,0\,0\,1\,0\,1\,1\,0)$ et $d_H(\bar{y} + \bar{b}_r, \bar{z}) = 8 - d_H(\bar{y}, \bar{z}) = 3$.
On a donc bien $3 = d_H(\bar{y} + \bar{b}_r, \bar{z}) = \frac{1}{2}(8 + \sum_{i=0}^{7}(-1)^{y_i+z_i}) = \frac{1}{2}(8 + (-2))$.

Il y a ainsi une symétrie au signe près entre les valeurs de $\hat{F}$ sur les 2^r éléments de la 1ère moitié du code RM (noté RM_0) et les 2^r éléments de la 2ème moitié (noté RM_1). On peut donc travailler sur la moitié du code et ainsi maximiser $|\hat{F}|$ au lieu de $\hat{F}$.

Plus précisément, si $|\hat{F}(\bar{\nu})|$ est maximum pour un élément $\bar{\nu} \in RM_0$, on a

$$d_H(\bar{z}, RM) = \begin{cases} \frac{1}{2}(2^r - \hat{F}(\bar{\nu})) & \text{si } \hat{F}(\bar{\nu}) \geq 0 \\ \\ \frac{1}{2}(2^r + \hat{F}(\bar{\nu})) & \text{si } \hat{F}(\bar{\nu}) < 0 \end{cases}$$

Le plus proche voisin de $\bar{z}$ est alors $\bar{\nu}$, si $\hat{F}(\bar{\nu}) \geq 0$, ou $\bar{y} = \bar{\nu} + \bar{b}_r$, si $\hat{F}(\bar{\nu}) < 0$.
Reste à trouver $\bar{\nu}$.

On peut à nouveau procéder par recherche semi-exhaustive en calculant $|\hat{F}(\bar{y})|$ pour tous les $\bar{y} \in RM_0$, mais on n'aurait rien gagné. L'intérêt de cette reformulation avec les $\hat{F}(\bar{u})$ est qu'il existe un algorithme rapide pour faire ce calcul.

3.3.4 Recherche rapide

Il s'agit d'exprimer sous forme matricielle la transformation :

$$\begin{aligned} \hat{F} : \{0, 1, \ldots, 2^r - 1\} &\to \mathbb{Z} \\ u &\mapsto \sum_{i=0}^{2^r-1} (-1)^{y_i} F(i) \end{aligned}$$

où $\bar{y}$ est le u-ème élément dans RM_0 et $F(i) = (-1)^{z_i}$.

On aimerait écrire sous forme matricielle :

$$\big(\hat{F}(0)\,\hat{F}(1)\,\hat{F}(2)\ldots\hat{F}(2^r-1)\big) = \big(F(0)\,F(1)\,F(2)\ldots F(2^r-1)\big)H_r$$

où H_r est une matrice de dimension $2^r \times 2^r$ et où F et $\hat{F}$ permettent de former les vecteurs à 2^r composantes $(F(i))_{i=0\ldots2^r-1}$ et $(\hat{F}(u))_{u=0\ldots2^r-1}$.

On procède récursivement.

- **Cas** $r = 1$
  Matrice d'encodage :
  $$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$
  Avec $u \in \{0, 1\}$, on a :
  $$\bar{y} = (0\ 0) \Rightarrow \hat{F}(0) = F(0)(-1)^0 + (-1)^0 F(1) = F(0) + F(1)$$
  $$\bar{y} = (0\ 1) \Rightarrow \hat{F}(1) = F(0)(-1)^0 + (-1)^1 F(1) = F(0) - F(1)$$
  Donc
  $$(\hat{F}(0)\, \hat{F}(1)) = (F(0)\, F(1)) \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$
  et ainsi
  $$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

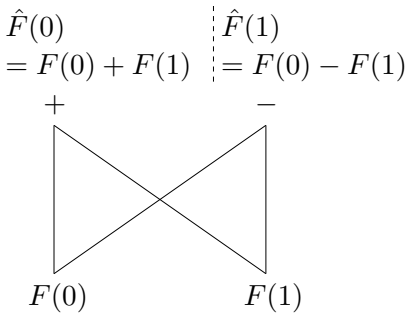

FIGURE 4 – Illustration du calcul pour le cas $r = 1$

- **Cas** $r = 2$
  Matrice d'encodage :
  $$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$
  Avec $u \in \{0, 1, 2, 3\}$, on a :
  $$\bar{y} = (0\ 0\ 0\ 0) \Rightarrow \hat{F}(0) = F(0)(-1)^0 + F(1)(-1)^0 + F(2)(-1)^0 + F(3)(-1)^0$$
  $$= F(0) + F(1) + F(2) + F(3)$$
  $$\bar{y} = (0\ 1\ 0\ 1) \Rightarrow \hat{F}(1) = F(0)(-1)^0 + F(1)(-1)^1 + F(2)(-1)^0 + F(3)(-1)^1$$

$$= F(0) - F(1) + F(2) - F(3)$$

$$\bar{y} = (0\ 0\ 1\ 1) \Rightarrow \hat{F}(2) = F(0)(-1)^0 + F(1)(-1)^0 + F(2)(-1)^1 + F(3)(-1)^1$$
$$= F(0) + F(1) - F(2) - F(3)$$

$$\bar{y} = (0\ 1\ 1\ 0) \Rightarrow \hat{F}(3) = F(0)(-1)^0 + F(1)(-1)^1 + F(2)(-1)^1 + F(3)(-1)^0$$
$$= F(0) - F(1) - F(2) + F(3)$$

Donc

$$(\hat{F}(0)\,\hat{F}(1)\,\hat{F}(2)\,\hat{F}(3)) = (F(0)\,F(1)\,F(2)\,F(3)) \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

et ainsi

$$H_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{pmatrix}$$

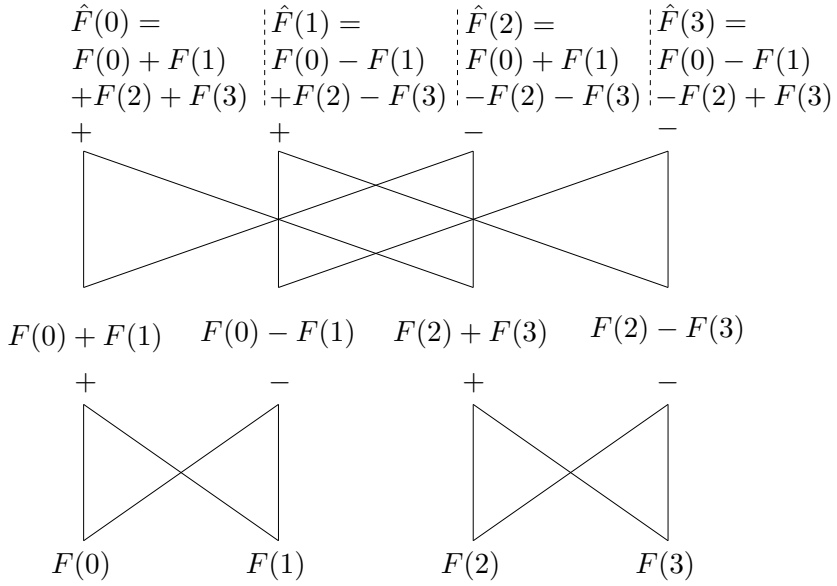

FIGURE 5 – Illustration du calcul pour le cas $r = 2$

- **Cas r quelconque**
  Cela se généralise en

$$H_r = \begin{pmatrix} H_{r-1} & H_{r-1} \\ H_{r-1} & -H_{r-1} \end{pmatrix}$$

Le calcul de $\left(\hat{F}(0)\,\hat{F}(1)\,\hat{F}(2)\ldots\hat{F}(2^r-1)\right)$ se fait en r étapes. A chaque étape, on effectue 2^r additions/soustractions. Ainsi, la complexité de ce calcul est $\mathcal{O}(r2^r)$. On peut lire l'opération à effectuer sur les bits de $u \in \{0,\ldots,2^r-1\}$.
Voici le pseudo-code du calcul des $\hat{F}$:

```
for i in 0..r-1 do
   for k in 0..2^r-1 do
      q := bitInverse(k,i)
      if getBit(k,i) = 0 then
         newF[k] := F[k] + F[q]
      else
         newF[k] := F[q] - F[k]
      end if
   end for
   F := newF
end for
```

FIGURE 6 – Algorithme de calcul de $\left(\hat{F}(0)\,\hat{F}(1)\,\hat{F}(2)\ldots\hat{F}(2^r-1)\right)$

La fonction `bitInverse(k,i)` inverse le i-ème bit de k, tandis que `getBit(k,i)` retourne le i-ème bit de k.

Forme des H_n
Soit I_n la matrice identité $n \times n$. On observe que

$$H_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} = \begin{pmatrix} I_2 & I_2 \\ I_2 & -I_2 \end{pmatrix} \begin{pmatrix} H_1 & 0 \\ 0 & H_1 \end{pmatrix}$$

De même

$$\begin{aligned} H_3 &= \begin{pmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{pmatrix} \begin{pmatrix} H_2 & 0 \\ 0 & H_2 \end{pmatrix} \\ &= \underbrace{\left(\begin{array}{c|c} I_4 & I_4 \\ \hline I_4 & -I_4 \end{array}\right)}_{K_3} \underbrace{\left(\begin{array}{cc|cc} I_2 & I_2 & 0 & 0 \\ I_2 & -I_2 & 0 & 0 \\ \hline 0 & 0 & I_2 & I_2 \\ 0 & 0 & I_2 & -I_2 \end{array}\right)}_{K_2} \underbrace{\left(\begin{array}{cc|cc} H_1 & 0 & 0 & 0 \\ 0 & H_1 & 0 & 0 \\ \hline 0 & 0 & H_1 & 0 \\ 0 & 0 & 0 & H_1 \end{array}\right)}_{K_1} \end{aligned}$$

Pour r quelconque,

$$H_r = \underbrace{\begin{pmatrix} I_{2^{r-1}} & I_{2^{r-1}} \\ I_{2^{r-1}} & -I_{2^{r-1}} \end{pmatrix}}_{K_r} \begin{pmatrix} H_{r-1} & 0 \\ 0 & H_{r-1} \end{pmatrix}$$

Finalement,

$$H_r = K_r \dots K_2 K_1$$

Donc

$$\begin{aligned} \big(\hat{F}(0)\,\hat{F}(1)\,\hat{F}(2)\dots\hat{F}(2^r-1)\big) &= \\ &= \big(F(0)\,F(1)\,F(2)\dots F(2^r-1)\big)H_r \\ &= \big(F(0)\,F(1)\,F(2)\,\dots F(2^r-1)\big)K_r\dots K_2K_1 \end{aligned}$$

Forme des matrices K_n

La matrice K_n de dimension $2^r \times 2^r$ s'écrit sous la forme

$$K_n = \begin{pmatrix} I_{2^{n-1}} & I_{2^{n-1}} & 0 & 0 & \dots & 0 & 0 \\ I_{2^{n-1}} & -I_{2^{n-1}} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & I_{2^{n-1}} & I_{2^{n-1}} & \dots & 0 & 0 \\ 0 & 0 & I_{2^{n-1}} & -I_{2^{n-1}} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & I_{2^{n-1}} & I_{2^{n-1}} \\ 0 & 0 & 0 & 0 & \dots & I_{2^{n-1}} & -I_{2^{n-1}} \end{pmatrix}$$

Il n'y a donc que deux éléments non nuls par colonne $i = 0, \dots, 2^{r-1}$:

— si le $n^{\text{ème}}$ bit de i vaut 0, alors

$$(K_n)_{i,i} = 1 \quad \text{et} \quad (K_n)_{i+2^{n-1},i} = 1$$

— si le $n^{\text{ème}}$ bit de i vaut 1, alors

$$(K_n)_{i-2^{n-1},i} = 1 \quad \text{et} \quad (K_n)_{i,i} = -1$$

La complexité du calcul total est $\mathcal{O}(r2^r)$.

En effet, la multiplication d'un vecteur par K_n nécessite $\mathcal{O}(2^{r+1})$ opérations (2 opérations par colonne) et pour chacune des r matrices K_n.