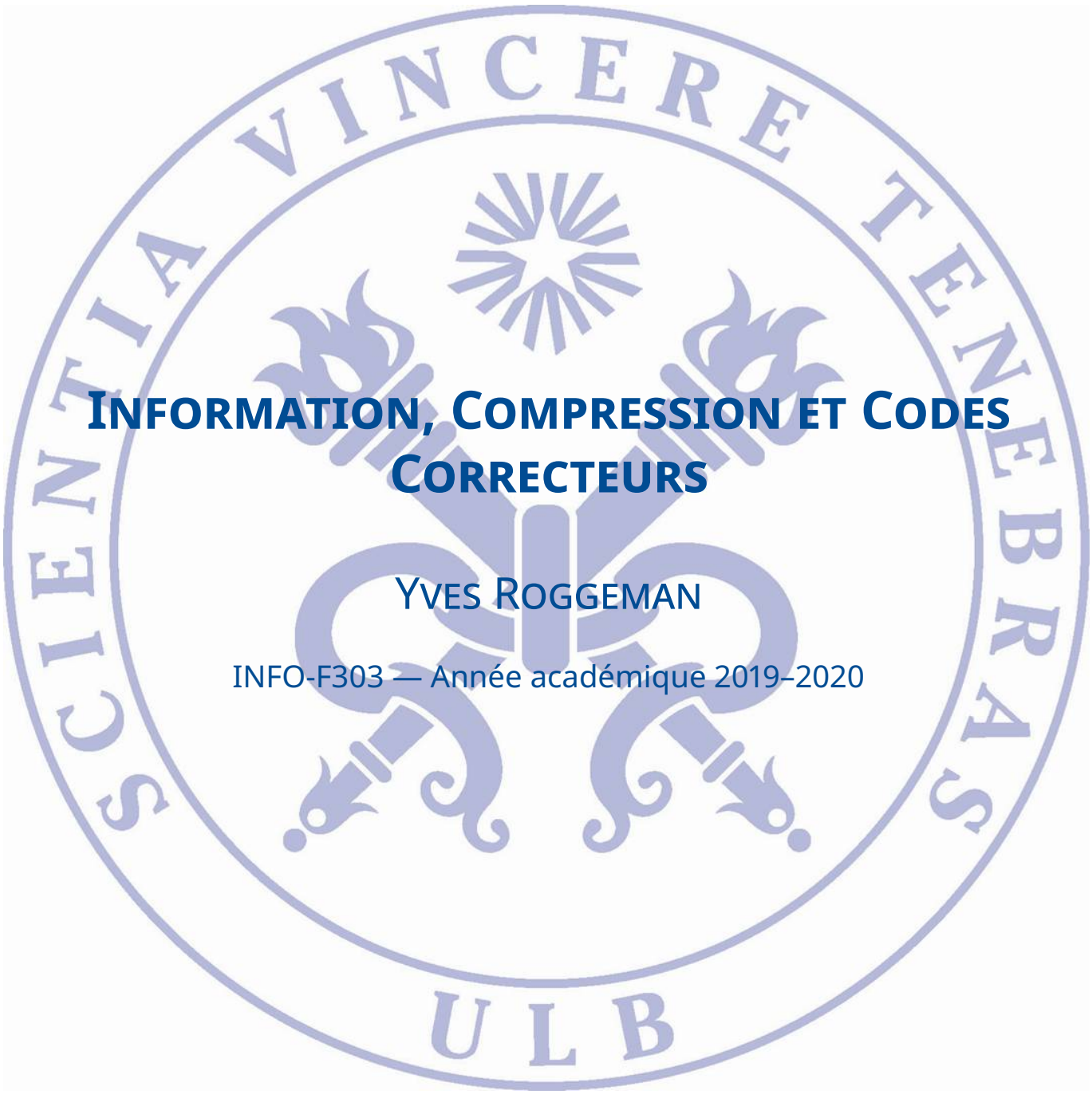
ULB UNIVERSITÉ LIBRE DE BRUXELLES

FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE



Information, Compression et Codes Correcteurs

Yves Roggeman

INFO-F303 — Année académique 2019–2020

Ce document sert de support de cours de l'activité d'enseignement théorique « Théorie de l'information » (1 crédit) qui fait partie de l'unité d'enseignement INFO-F303 « *Réseau, information et communication* ». Il ne se veut pas exhaustif, mais vise à donner un aperçu synthétique d'un sujet relativement vaste, complexe et très utile dans de nombreux domaines de l'informatique.

Le contenu présenté ci-dessous dépasse donc parfois la matière stricte du cours, afin de répondre à cet objectif de référence de base dans le cursus. D'autre part, il doit être complété des exercices associés à l'unité d'enseignement et aux exemples présentés au cours. Les autres supports, références et diaporamas présentés durant les leçons servent d'annexes à ce texte et font donc aussi partie de la matière.

Yves ROGGEMAN
Samedi 22 août 2020
<yves.roggeman@ulb.ac.be>

Chapitre I

Notion de code

On considère généralement que la *théorie de l'information* est véritablement née durant la Seconde Guerre mondiale des travaux de Claude Elwood Shannon, publiés seulement en 1948 (voir [32] et [33]), et de Warren Weaver (voir [34]), même s'il existait déjà des résultats antérieurs, notamment ceux, en 1924, de Harry Nyquist (né Harry Theodor Nyqvist) ou, en 1929, de Ralph Vinton Lyon Hartley.

Ces travaux furent également le point de départ des *algorithmes de compression*, de la *théorie des codes* et de la *cryptographie*.

Cette première période de la théorie de l'information, plutôt probabiliste, s'est complétée depuis 1960 de travaux plus algorithmiques, fondateurs de la *théorie de la complexité*, notamment ceux de Андрей Николаевич Колмогоров (Andreï Nikolaïevitch Kolmogorov), Ray Solomonoff (Raymond J. Solomonoff), Gregory John Chaitin, etc.

Enfin, la théorie moderne des codes correcteurs d'erreur, dont les fondateurs incontestés sont Richard Wesley Hamming (dès 1947) et Marcel Jules Édouard Golay (dès 1949), s'est jointe rapidement à la théorie de l'information (Claude Shannon intègre dès le début les travaux de Richard Hamming). Une introduction à ce sujet est donc également présentée ici.

Les sources principales servant de support à ce document sont [2], [19, 2d ed.], [40, 2d ed.] et [4].

I.1 Fonction de codage et notations

La théorie mathématique de l'information introduite par Claude Shannon repose sur un modèle où « l'information » est représentée par des *symboles*, plus précisément par des éléments d'un ensemble de symboles. Un message portant une information est une suite, une séquence finie de symboles. Pour être traité informatiquement, chaque symbole ou chaque message doit être *codé* par un *mot du code*, une suite finie de valeurs informatiques (des bits, des caractères...).

En informatique, de manière générale, on appelle *alphabet* un ensemble de symboles, appelés plutôt *caractères* ; par définition, il est toujours fini. Par contre, les *mots* écrits dans cet alphabet sont des chaînes de caractères, *a priori* de longueur quelconque ; l'ensemble des mots est donc potentiellement infini (dénombrable), ce qui permet donc de coder tout symbole ou tout message d'une source.

Dans ce contexte, la *fonction de codage*, *stricto sensu*, est une application de l'ensemble des symboles d'information vers les mots de longueur non nulle[1] du code informatique.

Les notations utilisées dans la suite de ce document sont les suivantes.

- S désigne l'ensemble des symboles de la source ; on le suppose fini ayant q symboles ($q = |\mathrm{S}|$) ; $\mathrm{S} = \{s_1, s_2 \ldots s_q\}$. Un message est alors un élément de S^* (notation habituelle pour désigner toutes les chaînes de S).

1. Sinon, la trace du symbole source, donc de l'information qu'il porte, disparaîtrait !

- C désigne l'alphabet du code ; il contient r caractères ($r = |C|$). Souvent $r = 2$ (codage binaire), mais pas toujours : d'autres exemples fréquents sont $r = 2^8$ (octet, UTF-8), $r = 2^{16}$ (dans un système Java) ou $r = 2^{32}$ (point de code UTF-32), etc.
- K désigne la fonction de codage : $K : S \to C^+ : s \mapsto c_1 c_2 \dots c_\ell$ où $\ell \geq 1$ (C^+ est une notation habituelle pour désigner toutes les chaînes non vides de C).
- $K(s)$ est ainsi le codage de s ; c'est un *mot*, une chaîne d'éléments de C.
- $K = K(S) \subset C^*$ est le *code*, l'ensemble des *mots du code* ; c'est l'image de la fonction de codage.
- ℓ_i désigne la longueur *non nulle* du mot $K(s_i)$, le nombre de caractères utilisés pour coder le symbole s_i ($\ell_i = |K(s_i)| > 0$).
- ℓ ou $L_{max}(K)$ désignent la longueur du code K, la longueur maximale des mots : $\ell = \max_{i=1\dots q}\{\ell_i\}$.

On étend généralement cette fonction de codage d'un symbole isolé en une fonction de codage d'un message, d'une suite quelconque de symboles sources.

$$K^* : S^* \to C^* : s_1 s_2 \dots s_n \mapsto K(s_1)K(s_2)\dots K(s_n)$$

La valeur de cette fonction est une chaîne de caractère appelée « *texte codé* ».

I.1.1 Codes univoques

Les *codes univoques* sont ceux qui sont *décodables*, c'est-à-dire ceux dont la fonction de codage K est injective :

$$s_i \neq s_j \Rightarrow K(s_i) \neq K(s_j)$$

Ainsi, on peut définir la *fonction de décodage* qui est sa fonction inverse :

$$(K^*)^{-1} : C^* \to S^* \cup \{\perp\}$$

Par convention, elle vaut $\perp$ si la chaîne de caractères ne correspond à aucun message valide.

On ne considérera ici que des codes univoques, car ils sont les seuls à préserver l'information de la source.

I.1.2 Codes en bloc

Les *codes en bloc* sont un cas particulier où tous les mots ont la même longueur n :

$$K : S \to C^n : s \mapsto c_1 c_2 \dots c_n$$

Dans ce cas, $\forall i : \ell_i = n$ et bien sûr $q \leq r^n$ (si le code est univoque, mais c'est implicite) ; on ne peut donc coder qu'un nombre fini de symboles.

L'avantage d'un tel code est que la découpe d'un texte codé en mots du code est aisée, donc le décodage également.

Les exemples de tels codes sont nombreux : ASCII, UTF-32 (« point de code ») et la plupart des *codes correcteurs d'erreur* (voir plus loin).

I.1.3 Codes sans préfixe et arbres de code

On appelle *préfixe* d'un mot ou d'une chaîne de caractère une sous-chaîne initiale du mot, c'est-à-dire correspondant à ses premiers caractères. Un *code sans préfixe* est un code dans lequel aucun mot du code n'est le préfixe d'un autre.

L'avantage d'un tel code est qu'il permet la découpe du texte codé en mots du code successifs *à la volée* [2]. En effet, il suffit de lire le texte caractère par caractère en s'arrêtant dès qu'un mot du code

2. Il s'agit d'un simple automate fini déterministe.

est reconnu. Évidemment, un texte erroné ne pourra pas toujours être correctement traité ainsi et la possibilité de *resynchronisation* à la limite d'un mot après erreur guide souvent la conception de tels codes.

La plupart des codes efficaces appartiennent à cette famille dont les exemples sont nombreux. Le code UTF-8 d'une chaîne de caractères Unicode est un code sans préfixe sur l'alphabet des octets [3].

Un code en bloc univoque n'est qu'un cas particulier de code sans préfixe.

Définition (Arbre de code)**.** Un code sans préfixe — c'est un ensemble fini de mots — peut également être représenté par un *arbre de code*. Un tel arbre est un arbre de degré r au plus dont chaque arc porte un caractère et dont les feuilles marquent les mots du code ; le mot du code associé à une feuille est formé de la suite des caractères rencontrés le long du chemin — unique, puisque c'est un arbre — de la racine à la feuille.

Pouvoir représenter un code sous forme d'arbre de code est une propriété nécessaire et suffisante au fait qu'il s'agisse d'un code sans préfixe. En effet, si un mot du code était le préfixe d'un autre, il correspondrait à un chemin partiel depuis la racine ; il ne serait donc pas associé à une feuille.

Le décodage d'un code sans préfixe, ainsi que la découpe d'un texte source se font simplement en parcourant l'arbre de code de la racine jusqu'à une feuille en choisissant à chaque nœud le sous-arbre en fonction du caractère lu [4].

La hauteur de l'arbre de code correspond ainsi à la *longueur maximale* des mots du code.

Dans l'arbre d'un code en bloc de longueur n, toutes les feuilles sont donc au même niveau n. Un arbre complet correspond alors à un code en bloc où tous les mots de C^n correspondent à un symbole de S.

I.2 Inégalité de Kraft-McMillan

Cette section concerne la propriété fondamentale des codes univoques, ce qui nous permettra de nous limiter à l'étude des seuls codes sans préfixe (et des codes par bloc qui en sont un cas particulier).

I.2.1 Inégalité de Kraft

Théorème I.2.1 (Leon Gordon Kraft, 1948)**.** *Pour tout ensemble de longueurs de mots du code* $\{\ell_1, \ell_2 \dots \ell_q\}$ *donné, il existe (au moins) un code sans préfixe univoque correspondant si et seulement si l'inégalité suivante — dite « inégalité de Kraft » — est satisfaite.*

$$\sum_{i=1}^{q} r^{-\ell_i} \leq 1 \tag{I.1}$$

Démonstration. Supposons que, dans un code sans préfixe donné, les q symboles de la source ont été réordonnés par longueurs de mots du code non décroissantes : $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_q$. La démonstration se fait alors par induction (ou récurrence) sur le nombre de symboles.

Pour $q = 1$, il y a r^{ℓ_1} choix possibles pour coder s_1 et bien sûr $r^{-\ell_1} \leq 1$ (et on n'a l'égalité que si $r = 1$).

Pour $q > 1$, nous supposons que le code $K(s_i)$ des $q - 1$ premiers symboles est fixé ; il constitue un code sans préfixe. Pour que le code complet soit également sans préfixe, le code de s_q ne peut être en collision préfixe avec aucun d'eux. Comptons les choix possibles pour $K(s_q)$: nous en avons r^{ℓ_q} dont nous devons retirer les choix qui seraient en conflit avec les préfixes déjà utilisés pour les

3. Ce code permet également une détection simple d'erreur avec resynchronisation, puisqu'un *octet de suite* n'est jamais le préfixe (l'octet initial) d'un mot du code.

4. Le code Morse peut également être représenté par un arbre binaire (il utilise deux caractères : point ou trait), mais ses nœuds codent tous des symboles. Pour permettre de décoder, il faut marquer les fins de mot du code par un espace (ou silence). Formellement, il devient alors un code sans préfixe sur un alphabet de trois caractères, donc avec un arbre de code ternaire.

symboles précédents. Chaque K(s_i) s'octroie un préfixe de ℓ_i caractères de C, donc exclut $r^{\ell_q-\ell_i}$ mots de longueur ℓ_q. L'existence de K(s_q) impose que :

$$1 \leq r^{\ell_q} - \sum_{i=1}^{q-1} r^{\ell_q-\ell_i}$$

Ensuite, un peu d'algèbre donne :

$$\begin{aligned} 1 &\leq r^{\ell_q} \cdot \left(1 - \sum_{i=1}^{q-1} r^{-\ell_i}\right) \\ r^{-\ell_q} &\leq 1 - \sum_{i=1}^{q-1} r^{-\ell_i} \\ \sum_{i=1}^{q} r^{-\ell_i} &\leq 1 \end{aligned}$$

Ainsi, s'il existe, le code satisfait l'inégalité. Inversement, comme cette propriété est équivalente à l'existence d'au moins un choix sans collision pour K(s_q), il existe un code sans préfixe pour cet ensemble de longueurs. ∎

Propriété I.2.2. *Dans l'inégalité de Kraft, l'égalité est atteinte si et seulement si l'arbre de code est homogène de degré r (i.e. dont tous les nœuds, c'est-à-dire les sommets qui ne sont pas de feuilles, ont exactement r fils).*

Démonstration. Cette propriété se déduit directement de l'observation d'un arbre de code, en remontant ses niveaux. Dans un arbre de hauteur n, un sommet de niveau $n-1$ intervient dans la somme pour $r^{-(n-1)}$ si c'est une feuille, mais pour $\sum_{i=1}^{\delta} r^{-n}$ si c'est un sous-arbre de degré δ ; on n'a pour un tel sous-arbre l'égalité que s'il est de degré maximum r et il pèse alors également $r^{-(n-1)}$. On applique ensuite ce même raisonnement à chaque sous-arbre en remontant vers la racine. ∎

I.2.2 Théorème de McMillan

Ce théorème est largement plus général que celui de Kraft ; combiné avec ce dernier, il nous permettra de raisonner simplement sur les arbres de code, si nécessaire.

Théorème I.2.3 (Brockway McMillan, 1956)**.** *Tout code univoque — i.e. qui peut être décodé — satisfait l'inégalité de Kraft.*

Démonstration. Considérons un message de longueur n ; son code K($s_{i_1}s_{i_2}\ldots s_{i_n}$) est de longueur notée $j = \ell_{i_1}+\ell_{i_2}+\cdots+\ell_{i_n}$. On notera $\mu_j(n)$ le nombre de tous les messages de longueur n produisant un code de longueur j (il pourrait être nul pour certaines valeurs de j et de n) et par $\mu_j = \sum_{n=1}^{j} \mu_j(n)$ le nombre de tous les messages de longueur quelconque produisant un code de longueur j (c'est une somme finie d'au plus j termes, puisque chaque symbole du message est codé par au moins un caractère). Par définition, $\forall n, \mu_j(n) \leq \mu_j$. De plus, comme le code est univoque, on sait que $\mu_j \leq r^j$.

La suite de la démonstration est fondée sur une astuce : on va considérer l'expression *constante* qui intervient dans l'inégalité de Kraft

$$c = \sum_{i=1}^{q} r^{-\ell_i}$$

et prouver que $\frac{c^n}{n}$ est borné $\forall n$, ce qui impliquera bien $c \leq 1$, car c'est une condition nécessaire et suffisante de $\lim_{n\to\infty} \frac{c^n}{n} < \infty$.

Pour ce faire, calculons les puissances successives de c :

$$\begin{aligned} c &= \sum_{i=1}^{q} r^{-\ell_i} \\ c^2 &= \left(\sum_{i_1=1}^{q} r^{-\ell_{i_1}}\right)\left(\sum_{i_2=1}^{q} r^{-\ell_{i_2}}\right) = \sum_{i_1=1}^{q}\sum_{i_2=1}^{q} r^{-\left(\ell_{i_1}+\ell_{i_2}\right)} \\ &\vdots \\ c^n &= \sum_{i_1=1}^{q}\sum_{i_2=1}^{q}\cdots\sum_{i_n=1}^{q} r^{-\left(\ell_{i_1}+\ell_{i_2}+\cdots+\ell_{i_n}\right)} \end{aligned}$$

Dans cette dernière somme (de q^n termes), on peut regrouper tous les termes de même puissance de r, c'est-à-dire les termes en r^{-j} où $j = \ell_{i_1} + \ell_{i_2} + \cdots + \ell_{i_n}$.

Si l'on note ℓ la longueur maximale des mots du code ($\ell = \max_{i=1\ldots q}\{\ell_i\}$), la plus grande valeur de j dans c^n est au maximum $n \cdot \ell$.

Nous avons donc

$$c^n = \sum_{j=1}^{n\cdot\ell} \mu_j(n) \cdot r^{-j} \leq \sum_{j=1}^{n\cdot\ell} \mu_j \cdot r^{-j} \leq \sum_{j=1}^{n\cdot\ell} r^j \cdot r^{-j} = n \cdot \ell$$

Nous venons donc de prouver que $\frac{c^n}{n} \leq \ell, \forall n$, ce qui achève la démonstration. ∎

Nous en tirons immédiatement une propriété sur la longueur minimale des mots d'un code.

Corollaire I.2.4. *Dans tout code univoque, la longueur maximale des mots de code est bornée inférieurement par*

$$\ell_{\max} \geq \left\lceil \log_r q \right\rceil \geq \log_r q \tag{I.2}$$

Démonstration. De l'inégalité de Kraft (I.1), nous tirons $1 \geq \sum_{i=1}^{q} r^{-\ell_i} \geq q.r^{-\ell_{\max}}$, donc $r^{\ell_{\max}} \geq q$. ∎

Chapitre II

Source aléatoire et codes efficaces

On ajoute maintenant deux hypothèses :

1. L'alphabet source S est fini et muni d'une loi de probabilité d'apparition de chaque symbole $\mathbb{P}[s_i] = p_i > 0$; les symboles de probabilité nulle peuvent donc être « oubliés », puisqu'ils ne seront jamais codés[1].
2. La source est *markovienne* : les messages de S^* sont constitués de symboles *indépendants*, donc

$$\mathbb{P}[s_{i_1} s_{i_2} \dots s_{i_n}] = p_{i_1} \cdot p_{i_2} \cdot \dots \cdot p_{i_n}$$

Dans ce contexte, la *longueur moyenne* d'un code est

$$L(K) = \mathbb{E}[\ell_i] = \sum_{i=1}^{q} \ell_i \cdot p_i$$

Définition (Code efficace)**.** Un code est *efficace* ou *optimal* si sa longueur moyenne est minimale ($L(K) = L_{\min}(\{p_i\})$) parmi celles de tous les codes possibles pour la source S de loi de probabilité $\mathbb{P}$. C'est une propriété qui ne dépend pas de la valeur de chaque mot du code particulier $K(s_i)$, ni même des symboles en soi, mais seulement de la distribution de probabilités de la source (p_i) et des longueurs des mots (ℓ_i).

Puisque seule la distribution de probabilité importe, on supposera également que les symboles de S sont ordonnés, numérotés par probabilités non croissantes : $p_1 \geq p_2 \geq \dots \geq p_q$.

Pour la suite, on supposera que l'alphabet comporte au moins deux symboles de probabilités non nulles (donc $q \geq 2$). Sinon, il n'y a que le code trivial, qui est sans intérêt : $p_1 = 1$, $K(s_1) = c_1$ et $\ell_1 = 1$.

II.1 Code de Shannon

Ce code a été décrit par Claude Elwood Shannon en 1948. Il se définit naturellement pour $C = \{0, 1 \dots r-1\}$ formé de r « chiffres » (les r premiers nombres naturels), mais si l'on veut, on peut bien sûr généraliser à tout autre C en numérotant ses caractères et en raisonnant ensuite sur leur numéro.

Comme $p_i > 0$ et $q > 1$, on a $p_i < 1$ ce qui permet d'attribuer la longueur $\ell_i = \left\lceil -\log_r p_i \right\rceil \geq 1$ aux mots du code ; c'est la définition du code de Shannon. On a $\ell_i \geq -\log_r p_i$, donc un code sans préfixe ayant ces longueurs existe toujours, puisque l'inégalité de Kraft est vérifiée :

$$\sum_{i=1}^{q} r^{-\ell_i} = \sum_{i=1}^{q} r^{-\left\lceil -\log_r p_i \right\rceil} \leq \sum_{i=1}^{q} r^{+\log_r p_i} = \sum_{i=1}^{q} p_i = 1$$

Les longueurs ℓ_i sont non décroissantes : $p_i \geq p_j \Rightarrow \ell_i \leq \ell_j$. Comme l'ensemble des longueurs des mots d'un code de Shannon $\{\ell_i\}$ est déterminé par la seule loi de probabilité d'occurrence des symboles, sa longueur moyenne également. Il peut être construit de la manière suivante.

1. Même si formellement un événement improbable n'est pas impossible, mais sa contribution sera nulle.

Propriété II.1.1. *Le code de Shannon* $\mathrm{K}(s_i)$ *est formé des chiffres en base* r *de* $\left\lfloor r^{\ell_i} \sum_{j=1}^{i-1} p_j \right\rfloor$. *C'est un code sans préfixe.*

Démonstration. Considérons le code ainsi défini. Comme $p_q > 0$ n'intervient dans aucun mot du code et que r^{ℓ_i} est entier, $\mathrm{K}(s_i) < r^{\ell_i}$ et ce nombre peut toujours s'écrire en ℓ_i chiffres en base r, éventuellement en le faisant commencer par des 0. Il a donc bien ainsi la longueur voulue.

Ceci produit un code sans préfixe. En effet, le préfixe strict de $\mathrm{K}(s_{i+1})$ de longueur ℓ_j, $j \leq i$, est $\left\lfloor r^{\ell_j} \sum_{k=1}^{i} p_k \right\rfloor$. Comparons-le à $\mathrm{K}(s_j)$.
De $r^{\ell_j}\left(\sum_{k=1}^{i} p_k - \sum_{k=1}^{j-1} p_k\right) = r^{\ell_j} \sum_{k=j}^{i} p_k \geq r^{\ell_j} p_j \geq r^{-\log_r p_j} p_j = \frac{1}{p_j} p_j = 1$, on déduit que leurs parties entières sont également différentes, donc $\mathrm{K}(s_{i+1})$ n'a aucun préfixe commun avec les codes précédents. ∎

Si les probabilités d'occurrence des symboles sont distinctes, le code de Shannon est ainsi univoquement déterminé. Bien sûr, d'une manière générale, on peut toujours échanger le code de deux symboles de même probabilité sans modifier les propriétés du code.

Comme nous le verrons, ce code est loin d'être optimal, raison pour laquelle il est très peu utilisé. Mais il sert de référence tant théorique — notamment à des fins de démonstration de propriétés des codes — que pratique, car il formule la longueur des mots du code de façon simple et explicite.

II.2 Code de Shannon-Fano

II.2.1 Construction

Ce code a été proposé par Roberto Mario Fano en 1949 comme amélioration du précédent. Nous allons présenter l'algorithme de construction du code dans le cas *binaire* ($r = 2$, $\mathrm{C} = \{0,1\}$).

1. On « scinde » la liste ordonnée des symboles en deux parts de probabilités proches. La probabilité de la première peut être légèrement supérieure ou inférieure à la valeur pivot selon le cas ; un seul test supplémentaire est donc nécessaire.
2. On complète le préfixe — qui est vide au départ — par un bit 0 pour les symboles de la première moitié et par un bit 1 pour ceux de la seconde.
3. On répète le même traitement sur chaque sous-liste en complétant ainsi le préfixe bit à bit ; on arrête bien sûr la division d'une sous-liste lorsqu'elle est constituée d'un seul symbole.

Cette méthode est généralisable pour $r > 2$, mais la division optimale en r sous-listes équilibrées n'est pas aussi efficace du point de vue algorithmique.

Le code de Shannon-Fano est trivialement sans préfixe : il construit l'arbre r-aire du code progressivement à partir de sa racine en veillant à ce que chaque nœud ait des arbres fils de poids quasi égaux. Cette construction est *descendante* (*top-down*).

II.2.2 Longueurs des mots du code

Un code de Shannon-Fano n'est pas univoquement déterminé. Outre l'échange entre symboles de même probabilité, cet algorithme laisse deux degrés de liberté :

- l'attribution de la valeur du bit à l'un et l'autre sous-arbre (on peut échanger le fils gauche et le fils droit de chaque nœud) ;
- la scission en deux sous-arbres lorsque deux positions mènent au même écart par rapport à la valeur pivot.

La première ne change évidemment aucune longueur des mots du code. On considérera dans ce cas qu'il s'agit intrinsèquement du même code. Mais ce n'est évidemment pas le cas de la seconde. En effet, l'ensemble des longueurs peut être différent comme le montre l'exemple $\mathbb{P} = \{3/8, 1/4, 1/4, 1/8\}$ conduisant aux deux codes $\{0, 10, 110, 111\}$ et $\{00, 01, 10, 11\}$. Mais on peut montrer que la longueur moyenne est toujours la même ($L = 2$ dans ce cas-ci).

On peut également montrer que la longueur moyenne d'un code de Shannon-Fano n'est jamais plus grande que celle du code de Shannon [2], même si certains mots du code peuvent être plus longs, comme le montre l'exemple suivant. Soit $q = 12$ et $\mathbb{P} = \{7/16, 1/4, 10 \times (1/4)\}$. Pour Shannon : $\ell_1 = \ell_2 = 2 < \ell_{3\ldots12} = 5$ (et $L = \frac{94}{32}$), mais ici : $\ell_1 = 1 < \ell_2 = \ell_{3\ldots12} = 3$ (et $L = \frac{88}{32}$).

Contrairement à l'intuition [3], ce code n'est pas efficace, car il peut avoir une longueur moyenne strictement supérieure à celle d'un code de Huffman, ce que montreront de simples contre-exemples (voir ci-dessous).

Ce code a été utilisé notamment dans l'algorithme IMPLODE du format ZIP, mais son usage y est tombé en désuétude, vu sa longueur sous-optimale.

II.3 Code de Shannon-Fano-Elias

Nous décrivons ce code uniquement pour ne pas le confondre avec le code de Shannon-Fano, car il est vraiment inefficace, puisqu'il utilise un caractère de plus que le code de Shannon. Par contre, il est relié directement aux travaux de Peter Elias en compression de données.

Ce code sans préfixe se construit de manière très semblable au code de Shannon : pour coder le symbole s_i, on utilise les $1 + \lceil -\log_r p_i \rceil$ chiffres les plus significatifs du développement de $\sum_{j=1}^{i-1} p_j + \frac{1}{2} p_i$ en base r.

Intuitivement, on considère la fonction de répartition de la loi de probabilité

$$F(x) = \mathbb{P}[X \leq x]$$

Dans le cas de la distribution discrète des $\mathbb{P}(s_i)$, c'est une fonction escalier à chaque i correspondant au symbole s_i dont la marche est de hauteur p_i. Elle induit une découpe de l'intervalle réel $[0, 1]$ en sous-intervalles disjoints correspondant à ces marches. Le code de Shannon choisit comme mot du code la valeur inférieure de chaque sous-intervalle, tandis qu'ici, c'est le milieu de la marche. L'avantage du code de Shannon-Fano-Elias est que les mots de code sont construits à partir de valeurs réelles dans l'intervalle ouvert $(0, 1)$, donc évitant les valeurs extrêmes 0 et 1. Voilà pourquoi il sert d'archétype aux *codes arithmétiques* utilisés en compression de données.

II.4 Code de Huffman

Le code de Huffman est, comme nous le verrons, « le » code univoque (et sans préfixe) *optimal*, c'est-à-dire celui dont la longueur moyenne est minimale. Il permet donc de coder un message de la façon la plus compacte possible pour une distribution des symboles et un alphabet de codage donnés. Il est donc très largement utilisé, notamment par la plupart des standards et logiciels ; c'est la méthode suivie par l'algorithme DEFLATE du format ZIP, ainsi que dans les formats multimédias JPEG ou MP3, par exemple.

2. Essentiellement par récurrence sur la longueur maximale des mots : les r sous-arbres de la racine satisfont la propriété par hypothèse et chacun pèse $\frac{1}{r}$ environ, ce qui correspond à accroître la longueur de chaque mot d'exactement d'un caractère. La fin technique de la démonstration consiste à répartir les petites variations autour de cette valeur entre les sous-arbres : l'inégalité découle du fait que cette variation est inférieure à la plus petite probabilité.

3. Et tant Fano que Huffman on tenté de le démontrer pendant des années.

II.4.1 Construction

Contrairement au code de Shannon-Fano, le code de Huffman (inventé par David Albert Huffman en 1952, *cf.* [21]) construit l'arbre r-aire du code de manière *ascendante* (*bottom-up*), par fusions successives d'arbres au sein d'une forêt à priorité où les arbres sont ordonnés par poids non croissants.

1. Au départ, les symboles constituent chacun leur propre arbre au sein de la forêt ayant pour poids la probabilité du symbole, dans leur ordre non croissant.
2. Si nécessaire, on ajoute (virtuellement) à la fin au plus $(r-2)$ arbres vides de poids nul, afin que le nombre d'arbres dans la forêt soit congruent à $1 \pmod{r-1}$; ceci n'est donc jamais nécessaire pour $r = 2$.
3. On fusionne les r derniers arbres, ceux ayant les poids les plus faibles, en un seul arbre ; son poids est la somme de ceux des arbres fusionnés.
4. On attribue *en préfixe* respectivement les valeurs $c_1, c_2 \dots c_r$ à ses différents arbres fils (*i.e.* aux codes de leurs feuilles).
5. On glisse cet arbre *à sa place* dans la forêt, afin d'y préserver l'ordre des arbres en poids non croissants.
6. La forêt a maintenant $(r-1)$ arbres en moins (car r arbres ont été fusionnés en un seul) ; le nombre d'arbres est donc toujours congruent à $1 \pmod{r-1}$. On répète le même traitement (depuis l'étape 3) jusqu'à la réduction de la forêt en un seul arbre.

On note que dans le cas binaire ($r = 2$), le nombre d'arbres décroît d'une unité à chaque étape et tous les nœuds sont de degré 2 ; il n'est donc jamais nécessaire d'ajouter d'arbres vides. Dans les autres cas ($r > 2$, ces arbres vides seront tous regroupés dès la première étape, donc l'algorithme s'achève avec un arbre dont ce seul nœud aura un degré parfois inférieur à r, mais toujours au moins 2. L'algorithme effectue donc exactement $\left\lceil \frac{q-1}{r-1} \right\rceil = \left\lfloor \frac{q+r-3}{r-1} \right\rfloor$ fusions d'arbres et insertions triées au sein de la forêt.

Exemple. Considérons un alphabet de 6 symboles ($q = 6$), notés A, B, C, D, E, F et codés en binaire ($r = 2$) (3 bits suffisent) ; leurs probabilités respectives sont $\mathbb{P} = \{3/8, 3/16, 5/32, 1/8, 3/32, 1/16\}$.

Les étapes de la construction du code de Huffman binaire correspondant sont illustrées au schéma II.1. Nous y notons les probabilités des symboles par leur numérateur sur le commun dénominateur 32.

Cet exemple illustre aussi le fait que le code de Huffman n'est pas unique : à l'étape 2, la place de l'arbre de poids 5 n'est pas univoquement déterminée, ce qui conduit à deux codes intrinsèquement distincts (indépendamment du choix de la valeur du bit pour « gauche » ou « droit »).

II.4.2 Efficacité

Par construction, un code de Huffman est sans préfixe, puisque l'algorithme aboutit à un arbre de code.

Nous allons démontrer qu'un code de Huffman est *efficace*, c'est-à-dire que sa longueur moyenne est la plus petite possible pour tous les codes sans préfixe construits sur S avec sa loi de probabilités. Il est donc le plus efficace (en moyenne) pour coder un message : pour toute suite de symboles de S, la longueur moyenne dans C^* de $\mathsf{K}(s_1 \dots s_n)$ sera minimale, donc K réalise pour cette source le meilleur taux de compression.

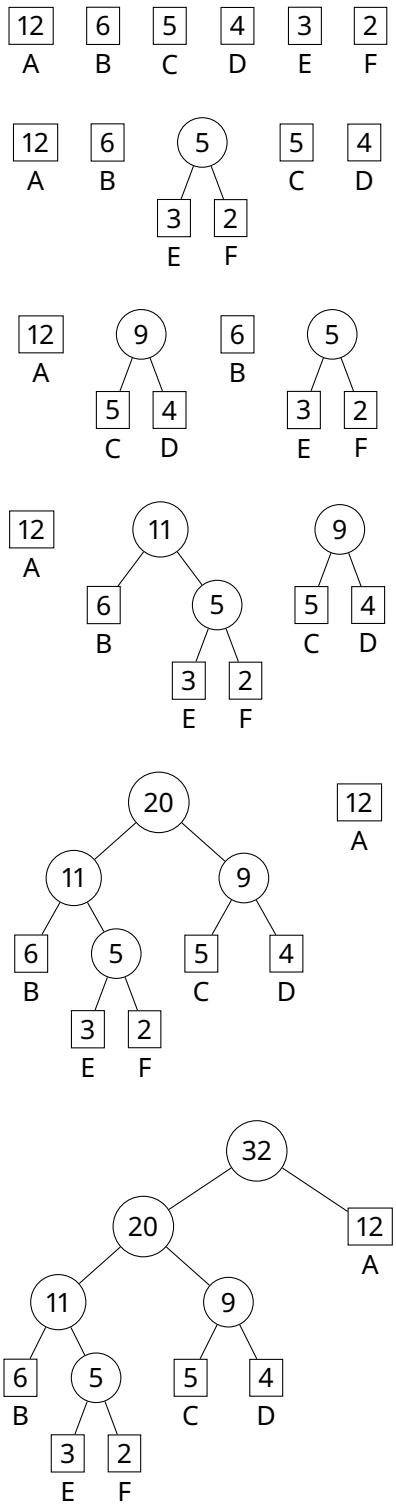
Cette démonstration utilisera les lemmes suivants.

Lemme II.4.1. *Un arbre dont tous les nœuds ont r fils contient n feuilles où* $n \equiv 1 \pmod{r-1}$.

Démonstration. La démonstration est implicitement contenue dans l'algorithme de construction d'un code de Huffman. On peut le démontrer par récurrence sur le nombre de feuilles de l'arbre.

Un arbre réduit à une seule feuille satisfait trivialement la propriété.

SCHÉMA II.1 – Construction d'un code de Huffman



Supposons que nous disposions d'un arbre ayant n feuilles dont tous les nœuds ont r fils ; par hypothèse d'induction, $n \equiv 1 \pmod{r-1}$. Si nous voulons lui ajouter une feuille, il nous faut nécessairement transformer une de ses feuilles en un nœud auquel on attache cette ancienne feuille et la nouvelle. Mais pour préserver la propriété, il faut que ce nouveau nœud ait r feuilles, donc il faut le compléter et lui attacher en fait de $(r-1)$ nouvelles feuilles ($r-1 \geq 1$). Son nouveau nombre de feuilles sera donc $n' = n + r - 1$ et $n' \equiv n \pmod{r-1}$. ∎

Lemme II.4.2. *Dans tout code efficace, $p_i > p_j \Rightarrow \ell_i \leq \ell_j$.*

Démonstration. Par l'absurde : supposons que nous disposions d'un code efficace K de longueur moyenne L minimale, mais pour lequel il existe deux symboles s_i et s_j avec $p_i > p_j$ et $\ell_i > \ell_j$.

Considérons le code K′ obtenu après échange des mots du code de ces deux symboles ; notons L′ sa longueur moyenne. On a :

$$L - L' = (\ell_i p_i + \ell_j p_j) - (\ell_j p_i + \ell_i p_j) = (\ell_i - \ell_j)(p_i - p_j) > 0$$

Or L est minimale, donc $L - L' \leq 0$, ce qui contredit l'hypothèse. ∎

Remarquons qu'on n'a pas utilisé le fait que le code est *sans préfixe*, mais échanger des mots du code entre symboles conserve bien cette propriété.

Par conséquent, dans tout code (sans préfixe) efficace dont les symboles sont rangés par probabilités non croissantes ($i < j \Rightarrow p_i \geq p_j$) et après permutation éventuelle de symboles de même probabilité, on a :

$$\ell_1 \leq \ell_2 \leq \cdots \leq \ell_q$$

Lemme II.4.3. *Pour tout code sans préfixe efficace, après échange éventuel de mots du code entre symboles, les nœuds dans l'arbre du code ont tous exactement r fils, à l'exception éventuelle du seul père des symboles de probabilités les plus faibles. Ces derniers symboles ont des mots du code de même longueur qui ne diffèrent que par leur dernier caractère ; ils sont au nombre de*

$$\delta = 2 + (q - 2) \bmod (r - 1)$$

Démonstration. En vertu du lemme II.4.2 précédent, nous supposons que les symboles sont triés par probabilités non croissantes et que les mots du code correspondants sont également triés par longueurs non décroissantes.
La hauteur de l'arbre du code est, par définition, $\ell = \max(\ell_i) = \ell_q$.

Considérons d'abord les nœuds « intermédiaires », ceux de niveau $h : 0 \leq h < \ell - 1$. Par l'absurde, si l'un de ceux-ci était incomplet, nous pourrions y ajouter une feuille dont le mot du code serait attribué à s_q (en remplacement de son mot du code actuel, et si son ancien frère devient fils unique, on le remonte dans son père ce qui diminue sa longueur également). Ce nouveau mot du code de s_q serait de longueur strictement inférieure à son ancienne et la longueur moyenne du code le serait également, ce qui contredit l'hypothèse d'efficacité.

Considérons maintenant les nœuds de niveau maximal $h = (\ell - 1)$; leurs fils sont tous des feuilles dont les mots du code sont tous de même longueur maximale ℓ. Nous pouvons échanger ces mots du code entre eux sans modifier la longueur moyenne, donc en préservant l'efficacité. Nous pouvons donc « compacter » cet arbre de code afin que tous ces nœuds aient exactement r feuilles, sauf éventuellement le « dernier », celui ayant les feuilles associées aux symboles de probabilités les plus faibles.

Si l'on « oublie » provisoirement ce dernier nœud, en vertu du lemme II.4.1 ci-dessus, cet arbre résiduel contient $(n-1)$ feuilles où $(n-1)$ est un multiple de $(r-1)$. Donc pour l'arbre complet qui compte δ feuilles supplémentaires : $q \equiv \delta \pmod{r-1}$. Mais dans cet arbre complet, le dernier nœud a au moins deux feuilles, sinon on pourrait amputer le seul mot du code de ce nœud de son dernier caractère et faire ainsi décroître strictement la longueur moyenne du code, ce qui contredit l'hypothèse d'efficacité. Au total, on a donc $\delta \geq 2$ et $q - 2 \equiv \delta - 2 \pmod{r-1}$, ce qui donne le résultat attendu. ∎

Une conséquence de ce lemme est que dans le cas binaire ($r = 2$), tous les nœuds ont exactement 2 fils.

Lemme II.4.4. *L'arbre de construction d'un code de Huffman a la même propriété.*

Démonstration. C'est une conséquence directe de l'algorithme de construction : chaque nœud a exactement r fils, sauf le dernier qui contient le solde δ. Le même calcul conduit ici aussi à $\delta = 2 + (q - 2) \bmod (r - 1)$. ∎

Théorème II.4.5. *Les codes de Huffman sont efficaces.*

Démonstration. Celle-ci se fait par récurrence sur le nombre q de symboles dans S.

Pour $q \leq r$, c'est immédiat : dans un code de Huffman, $\ell_i = 1, \forall i$. Il est donc de longueur moyenne $L = 1$ et c'est bien un code efficace, car on ne pourrait faire moins !

Supposons donc $q > r$ et construisons un code de Huffman K_H pour S ; notons L_H sa longueur moyenne. À la première étape, l'algorithme fusionne δ symboles en un sous-arbre, où δ est déterminé par le lemme II.4.4. Les mots du code de ces symboles auront tous la même longueur (la longueur maximale du code) et ne différeront que par leur dernier caractère.

Appelons $\bar{S}$ l'ensemble de $q - \delta + 1$ symboles dérivé de S par fusion de ces δ derniers symboles en un seul : $\forall i \leq q - \delta : \bar{s}_i = s_i$ et $\bar{p}_i = p_i$, mais $\bar{s}_{q-\delta+1} = \{s_{q-\delta+1}, \dots, s_q\}$ et $\bar{p}_{q-\delta+1} = p_{q-\delta+1} + \cdots + p_q$.

Les étapes suivantes de l'algorithme de construction de K_H correspondent exactement à la construction du code de Huffman $\bar{K}_H$ de $\bar{S}$ obtenu par ablation du dernier caractère de symboles fusionnés pour définir $\bar{K}_H(\bar{s}_{q-\delta+1})$ et maintien du code des autres symboles. Ce code est sans préfixe et, par hypothèse d'induction, il est efficace ; sa longueur moyenne vaut $\bar{L} = L_H - p_{q-\delta+1} - \cdots - p_q$. Dans l'arbre du code de $\bar{S}$, tous les nœuds ont exactement r fils.

Par ailleurs, supposons que K_{min} est un code sans préfixe efficace pour S de longueur moyenne L_{min} minimale. Nous pouvons supposer que ce code satisfait le lemme II.4.3. Si nous y fusionnons les δ derniers symboles comme ci-dessus, nous déduisons de la même façon un code $\bar{K}_{min}$ dont la longueur moyenne est $L' = L_{min} - p_{q-\delta+1} - \cdots - p_q$. Mais nous ne savons pas s'il est efficace pour $\bar{S}$; donc $L' \geq \bar{L}$.

On remplaçant L' et $\bar{L}$ par leurs valeurs, on trouve $L_{min} \geq L_H$. Mais comme K_{min} est efficace, on a nécessairement $L_{min} = L_H$, ce qui induit le théorème. ∎

Par conséquent, si l'on veut construire un code efficace, on utilise l'algorithme de Huffman. Cela conduit directement au corollaire suivant.

Corollaire II.4.6. *Pour tout ensemble de symboles* $S = \{s_i\}$ *muni de toute loi de probabilité* $\mathbb{P}(s_i)$*, il existe un code efficace dont la longueur moyenne vaut* $L_{min}(S)$ *qui ne dépend que de ces probabilités.*

II.4.3 Longueurs des mots du code

La construction du code laisse une certaine latitude dans le tri des arbres (et des symboles) au sein de la forêt en cas de poids égaux. Par conséquent, un code de Huffman n'est pas univoquement déterminé, même pas par son ensemble des longueurs $\{\ell_i\}$ qui résulte des fusions successives ! Ce qui est préservé, c'est sa longueur *moyenne* L(K), puisque comme nous l'avons vu, celle-ci vaut toujours la valeur minimale (les codes de Huffman sont efficaces). Le tableau ci-dessous (tableau II.1) illustre la multiplicité des codes de Huffman possibles pour une même loi de probabilité.

Supposons $r = 2$ et $q = 7$ avec la loi de probabilité $\left\{\frac{8}{21}, \frac{5}{21}, \frac{3}{21}, \frac{2}{21}, \frac{1}{21}, \frac{1}{21}, \frac{1}{21}\right\}$. Même sans compter les 3! permutations des symboles de même probabilité $\frac{1}{21}$ ni les choix arbitraires du caractère (ici, un bit) associé à chaque sous-arbre, on trouve 16 arbres de Huffman distincts (*i.e.* ce sont des graphes non isomorphes), correspondant aux 8 ensembles de longueurs reprises dans le tableau II.1. On a indiqué pour chacun le nombre d'arbres de code correspondants [4] ; le nombre exact de codes différents pour le cas binaire a été déterminé par Franklin et Golomb (*cf.* [11]).

Nous y avons chaque fois indiqué la longueur maximale (correspondant à la hauteur de l'arbre) et la somme non pondérée des longueurs (une mesure de sa densité), deux valeurs qui nous serviront pour comparer les codes adaptatifs. Bien sûr, tous ces codes ont la même longueur moyenne valant ici $\frac{52}{21}$.

4. On peut démontrer (*cf.* [15]) que la situation où il y a le plus de variantes possibles correspond comme ici à des probabilités respectant une suite de Fibonacci, généralisée si $r > 2$.

Tableau II.1 – Longueurs des mots d'un code de Huffman

Fréquences :			8	5	3	2	1	1	1
Nbr	Max	Σ	Longueurs						
1	6	27	1	2	3	4	5	6	6
1	5	26	1	2	3	5	5	5	5
2	5	25	1	2	4	4	4	5	5
2	5	24	1	3	3	3	4	5	5
2	5	23	2	2	2	3	4	5	5
2	4	23	1	3	3	4	4	4	4
2	4	22	2	2	2	4	4	4	4
4	4	21	2	2	3	3	3	4	4
Shannon :			2	3	3	4	5	5	5

Comparaison avec les codes précédents

À titre de comparaison, nous avons également indiqué dans le tableau II.1 les longueurs du code de Shannon pour les mêmes fréquences. On voit qu'un code de Huffman peut avoir des mots plus longs ou plus courts que dans un code de Shannon.

D'une part, la longueur maximale d'un code de Shannon peut dépasser autant qu'on veut celle du code de Huffman correspondant. Considérons en effet le cas $q = 2$ où l'un des deux symboles est de probabilité r^{-n}. Le code de Shannon de ce symbole est de longueur $\lceil -\log_r r^{-n} \rceil = n$, aussi grande que l'on veut, mais dans le code de Huffman correspondant, les 2 mots du code sont de longueur 1. On note que la longueur moyenne du code de Shannon $1 + (n-1)r^{-n}$ est toujours strictement supérieure à la longueur optimale, mais s'en approche à la limite si $n \to \infty$ (car ce symbole au long code devient improbable).

Inversement, les longueurs de certains mots et la longueur maximale d'un code de Huffman peuvent également être plus grandes que celles d'un code de Shannon, comme on le voit également dans l'exemple de la table II.1, même si sa longueur moyenne reste évidemment inférieure. Yaser Said Abu-Mostafa (اسر سعيد أبو مصطفى) et Robert James McEliece [1] ont montré que, dans le cas binaire, la longueur maximale de Huffman peut être jusqu'à 44 % supérieure à celle de Shannon.

Exemple. À titre d'exemple, le tableau II.2 contient les différents codes binaires ($r = 2$) correspondant à une source S de 6 symboles ($q = 6$) de probabilités $\mathbb{P} = \{3/8, 3/16, 5/32, 1/8, 3/32, 1/16\}$. Nous indiquons pour chacun la longueur moyenne (plus exactement son numérateur, soit 32 fois cette longueur moyenne), la longueur maximale et la somme non pondérée des longueurs.

Tableau II.2 – Codes de Shannon, Fano et Huffman

Probabilités ×32 :				12	6	5	4	3	2
Code	Moy	Max	Σ	Mots du code					
Shannon-Fano-Elias	121	5	25	001	0111	1010	1100	11101	11111
Shannon	89	4	19	00	011	100	101	1101	1111
Shannon-Fano	78	3	14	00	01	100	101	110	111
		4	17	00	01	10	110	1110	1111
Huffman	77	4	18	1	000	001	011	0100	0101
				0	100	110	111	1010	1011

Dans ce cas-ci, les longueurs des mots du code de Huffman sont univoquement déterminées, mais pas son arbre de code. Par contre, ce n'est pas le cas du code de Shannon-Fano qui peut conduire à des longueurs maximales et totales variées.

Chapitre III

Entropie et codage efficace

Les exemples de codes ci-dessus montrent qu'ils peuvent être définis sur base de la seule loi de probabilité de la source. La probabilité d'occurrence d'un symbole de la source est donc intrinsèquement liée à l'information codable qu'il porte.

III.1 Entropie

III.1.1 Quantité d'information propre

La quantité d'information propre portée par un symbole de probabilité donnée (en anglais, *self-information*) est, par définition, une fonction à valeur réelle qui ne dépend que de la probabilité d'apparition de ce symbole[1]. Par conséquent, elle est définie par $\mathcal{I}^*(s_i) = \mathcal{I}(\mathbb{P}(s_i))$ où $\mathcal{I}$ est une fonction d'une variable réelle

$$\mathcal{I} : \ ]0,1] \subset \mathbb{R} \to \mathbb{R} : p \mapsto \mathcal{I}(p)$$

dite fonction « *quantité d'information* ». Elle satisfait conventionnellement les axiomes suivants.

Définition (Axiomes de la quantité d'information)**.**

1. Elle est **positive** ($\mathcal{I}(p) \geq 0$) et elle n'est pas identiquement nulle ($\exists\, s_i \ : \ \mathcal{I}(\mathbb{P}(s_i)) \neq 0$). C'est l'idée que la source fournit effectivement de l'information.
2. Elle est **additive** : $\mathcal{I}^*\left(\{s_i, s_j\}\right) = \mathcal{I}(p_i \cdot p_j) = \mathcal{I}(p_i) + \mathcal{I}(p_j)$. Cet axiome exprime le fait que les symboles ne portent pas d'information commune ; ils sont « indépendants ».
3. Elle est **continue** (en p). Cette hypothèse est de nature à simplifier la formalisation, sans en limiter l'expressivité, mais elle est totalement arbitraire.

L'information est donc une *mesure*[2] positive sur S, au sens de l'analyse mathématique.

De ces axiomes, on déduit la propriété suivante.

Propriété III.1.1 (Claude E. Shannon, 1948)**.** *L'information d'un symbole de probabilité p non nulle (*$0 < p \leq 1$*) est nécessairement de la forme*

$$\mathcal{I}_b(p) = -\log_b p \quad \textit{avec} \quad b > 1 \tag{III.1}$$

Elle est donc fixée à une constante multiplicative positive près, l'unité de mesure utilisée.

1. L'information ne peut pas être liée au symbole en soi, car il n'est qu'une convention d'écriture que l'on peut librement modifier.

2. À ne pas confondre avec une *distance* mathématique. La distance est un concept géométrique : c'est une fonction définie pour une paire d'éléments d'un ensemble qui généralise la notion d'éloignement entre ces éléments. La mesure est une fonction *réelle* définie sur une famille de sous-ensembles d'un ensemble qui généralise la notion de poids ou de volume d'un sous-ensemble. Une loi de probabilité est un exemple classique de mesure bornée positive de poids total valant 1.

Démonstration. De l'axiome d'additivité appliqué à deux symboles équiprobables, on déduit $\mathcal{I}(p^2) = 2\mathcal{I}(p)$ et, par itération, $\mathcal{I}(p^n) = n\mathcal{I}(p)$ pour tout p et tout n entier positif.

De cela et de $p = \left(p^{\frac{1}{n}}\right)^n$, on déduit également $\mathcal{I}(p) = n\mathcal{I}\left(p^{\frac{1}{n}}\right)$, c'est-à-dire $\mathcal{I}\left(p^{\frac{1}{n}}\right) = \frac{1}{n} \cdot \mathcal{I}(p)$. Donc, $\forall p$ et $\forall \alpha \in \mathbb{Q} : \mathcal{I}(p^\alpha) = \alpha \cdot \mathcal{I}(p)$.

De l'axiome de continuité et, vu la densité de $\mathbb{Q}$ dans $\mathbb{R}$, cette relation s'étend à tout $\alpha \in \mathbb{R}$.

Donc $\mathcal{I}(p) = k \cdot \ln p$ où $k < 0$ en vertu du premier axiome ; on peut ainsi écrire $k = \frac{-1}{\ln b}$ avec $b > 1$. ∎

Corollaire III.1.2. *La fonction $\mathcal{I}(p)$ est strictement monotone décroissante : $p_i > p_j \Rightarrow \mathcal{I}(p_i) < \mathcal{I}(p_j)$; de plus, $\mathcal{I}(1) = 0$.*

En d'autres termes, plus un symbole est rare, plus son occurrence dans un message est porteuse d'information ; inversement, un symbole très fréquent est sans intérêt puisqu'un message reste intelligible s'il est omis (pensons au caractère « e », voire toute voyelle, en français). À la limite, une source d'un seul caractère ($q = 1$) ne transporte aucune information. Voilà pourquoi l'information propre est aussi appelée la « surprise » d'apparition d'un symbole.

III.1.2 Entropie d'une source

Par définition, l'entropie d'une source S est la quantité d'information *moyenne* de ses symboles.

Définition (Entropie)**.**

$$\mathrm{H}(\mathrm{S}) = \mathrm{H}\left(p_1, p_2, \dots, p_q\right) = \mathbb{E}[\mathcal{I}] = -\sum_{i=1}^{q} p_i \log_b p_i = \sum_{i=1}^{q} p_i \log_b \frac{1}{p_i} \tag{III.2}$$

Nous avons considéré que $p_i > 0$, mais cette définition peut s'étendre à des symboles improbables en posant $0.\log_b 0 = 0.\log_b \frac{1}{0} = 0$, puisque $\lim_{x\to 0} x \log_b x = \lim_{x\to 0} x \log_b \frac{1}{x} = 0$. Ces symboles de probabilité nulle n'ont donc aucune influence sur l'entropie de la source. Nous adopterons cette convention lorsque nous aurons besoin de tels symboles.

Propriété III.1.3.

1. H *est positive :* $\mathrm{H}\left(p_1, \dots, p_q\right) \geq 0$.
2. $\mathrm{H}\left(p_1, \dots, p_q\right) = \mathrm{H}\left(p_1, \dots, p_q, 0\right)$.
3. $\mathrm{H}\left(p_1, \dots, p_q\right)$ *est continue et symétrique*[3] *en ses q variables.*
4. $\mathrm{H}\left(p_1, \dots, p_q\right) \leq \mathrm{H}\left(\frac{1}{q}, \dots, \frac{1}{q}\right)$.
5. H *est « cohérente » :* $\mathrm{H}\left(p_1, \dots, p_q\right) = \mathrm{H}\left((p_1+p_2), p_3, \dots, p_q\right) + (p_1+p_2)\mathrm{H}\left(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2}\right)$.

Cette dernière propriété dite de cohérence exprime que l'information moyenne est la somme de celle où l'on confond d'abord deux symboles et de celle apportée par leur distinction ultérieure, pondérée bien sûr par leurs probabilités d'apparition respectives.

Démonstration.

1. Immédiat, puisque $\forall i : -p_i \log_b p_i \geq 0$.
2. Grâce à la convention $-0.\log_b 0 = 0$.
3. Immédiat.

3. On peut permuter ses variables sans en changer la valeur.

4. Pour trouver max(H), il faut tenir compte du fait que les q variables sont liées[4]. On écrit donc $p_q = 1 - \sum_{i=1}^{q-1} p_i$ et on annule les dérivées

$$0 = \frac{\partial \mathrm{H}}{\partial p_i} = -\log_b p_i - p_i \cdot \frac{1}{\ln b} \cdot \frac{1}{p_i} + \log_b p_q - p_q \cdot \frac{1}{\ln b} \cdot \frac{-1}{p_q} = -\log_b p_i + \log_b p_q \Rightarrow p_i = p_q$$

puisque le logarithme est une fonction monotone. Comme ceci est vrai pour tout i, on en déduit $p_i = \frac{1}{q}$.

5. On développe le second membre et

$$\begin{aligned}
\mathrm{H} &= -(p_1+p_2)\log_b(p_1+p_2) \\
&\quad - \sum_{i=3}^{q} p_i \log_b p_i + (p_1+p_2)\left(-\frac{p_1}{p_1+p_2}\log_b \frac{p_1}{p_1+p_2} - \frac{p_2}{p_1+p_2}\log_b \frac{p_2}{p_1+p_2}\right) \\
&= -(p_1+p_2)\log_b(p_1+p_2) \\
&\quad - \sum_{i=3}^{q} p_i \log_b p_i - p_1 \log_b p_1 + p_1 \log_b(p_1+p_2) - p_2 \log_b p_2 + p_2 \log_b(p_1+p_2) \\
&= -\sum_{i=1}^{q} p_i \log_b p_i
\end{aligned}$$

∎

Théorème III.1.4 (Дмитрий Константинович Фаддеев [Dimitri K. Faddéev], 1956)**.** *Les seules fonctions satisfaisant ces propriétés sont celles définies par (III.2) ci-dessus ; elles sont donc toutes égales à une constante multiplicative positive près : la base des logarithmes ou l'unité de mesure utilisée.*

La démonstration est longue et très technique (voir [7]), sans intérêt particulier pour notre sujet.

Corollaire III.1.5. *On a*

$$0 \leq \mathrm{H}_b(\mathrm{S}) \leq \log_b q$$

et les valeurs extrêmes sont atteintes si et seulement si $\exists i : p_i = 1$ *(entropie nulle) ou* $\forall i : p_i = \frac{1}{q}$ *(entropie maximale).*

Pourquoi H ?

Beaucoup se demandent pourquoi l'entropie se note traditionnellement « H ». Cette convention est utilisée systématiquement dès l'émergence de la théorie de l'information, notamment dans les travaux fondateurs de Claude Shannon, sans aucun commentaire ni justification.

Cet usage remonte en fait à la fin du XIX^{e} siècle en thermodynamique ; citons en particulier le célèbre H-théorème de Ludwig Boltzmann. Dans ses premiers écrits, il définit le « *Begriff der Entropie* » (« concept d'entropie » en allemand), terme inspiré du grec ἐντροπία (et repris des travaux antérieurs de Rudolph Clausius vers 1865) qu'il note naturellement « E ». Mais c'est également l'initiale de « *Energie* » (« énergie » en allemand), voilà pourquoi Boltzmann utilisa ensuite la lettre grecque êta majuscule « H » pour l'entropie, et pas un epsilon majuscule trop semblable au « E » latin. Mais pourquoi ce choix ? Il est probable que, vu la similarité des glyphes, ce soit la lettre latine « H » qui s'est imposée pour des raisons mnémoniques, car elle rappelle le mot « *Heat* » ; le H-théorème décrit en effet la distribution d'énergie cinétique des molécules d'un gaz, donc la *chaleur* qu'il renferme. La lettre capitale latine « H » est ainsi devenue la notation habituelle de l'entropie dans tous les contextes[5].

4. On peut aussi le faire par la technique des multiplicateurs de Lagrange : on ajoute une variable λ et on minimise la fonction en $q+1$ variables $\mathrm{H}(p_i) + \lambda(1 - \sum p_i)$

5. Voir par exemple <http://math.stackexchange.com/questions/84719/why-is-h-used-for-entropy> et ses nombreuses références.

III.1.3 Unité d'information et définition du bit

Selon le choix de la base des logarithmes, on obtient des valeurs numériques différentes pour l'entropie d'une source. En fait, il s'agit d'un simple choix de *l'unité de mesure* pour l'information propre ou pour l'entropie (puisque la seconde n'est que la moyenne de la première).

Définition (Unité d'information)**.** L'unité d'information est, par convention, l'information propre d'un symbole lorsque tous sont équiprobables, ce qui induit $b = q$. Exprimées en cette unité, la quantité d'information et l'entropie valent

$$\mathcal{I}(p) = -\log_q p \quad \text{et} \quad \mathrm{H}(\mathrm{S}) = -\sum_{i=1}^{q} p_i \log_q p_i \tag{III.3}$$

Dans ce cas, l'entropie maximale vaut également 1 ($0 \leq \mathrm{H} \leq 1$) en vertu du corollaire III.1.5.

Dorénavant, lorsque nous écrirons simplement $\mathcal{I}$ ou H, ce sera implicitement en base q. Trivialement, pour toute autre base :

$$\mathcal{I}_b(p) = \frac{\mathcal{I}(p)}{\log_q b} \quad \text{et} \quad \mathrm{H}_b(\mathrm{S}) = \frac{\mathrm{H}(\mathrm{S})}{\log_q b}$$

Trois unités sont aussi classiquement utilisées :

- Le « **hartley** »[6] (symbole « Hart ») correspondant au codage décimal ($b = 10$). Cette unité est celle qui est reprise par le système international d'unités (SI).
- Le « **nat** » où l'on choisit d'utiliser les logarithmes népériens ($b = e$) dits « naturels », même si la base n'est pas un nombre entier et ne peut correspondre à q !
- Le « **shannon** » (symbole « Sh »), unité la plus traditionnelle en informatique, celle du système binaire ($b = q = 2$).

En théorie de l'information, par référence aux travaux de Shannon, c'est ce dernier choix qui est le plus fréquent et, dans ce contexte, quand on écrit « log » sans indiquer la base, celle-ci elle vaut implicitement 2. On appelle alors l'information unitaire un « **bit** », même si la confusion entre le shannon et le bit n'est pas tout à fait correcte. En effet, *l'information propre d'un bit ne vaut un shannon que si les deux valeurs du bit sont équiprobables*.

III.1.4 Fonction d'entropie

Considérons le cas particulier $q = 2$; les probabilités sont alors $p_1 = p$ et $p_2 = 1-p$. L'entropie d'une telle source est alors une fonction d'une variable p appelée « fonction d'entropie binaire » :

$$\mathcal{H}_2(p) = -p \log_2 p - (1-p) \log_2 (1-p) \tag{III.4}$$

On a donc, $\mathcal{H}_2(0) = \mathcal{H}_2(1) = 0$, $\max \mathcal{H}_2 = \mathcal{H}_2\left(\frac{1}{2}\right) = 1$ et $\frac{\mathrm{d}\,\mathcal{H}_2}{\mathrm{d}\,p} = -\log_2 \frac{p}{1-p}$, l'opposé de la fonction « logit »[7].

Cette fonction se généralise au cas où les $(q-1)$ premiers symboles sont équiprobables :

$$\mathcal{H}_q(p) = -(q-1)\frac{p}{q-1} \log_q \frac{p}{q-1} - (1-p)\log_q (1-p) = p \log_q (q-1) + \frac{\mathcal{H}_2(p)}{\log_2 q} \tag{III.5}$$

On a $\frac{\mathrm{d}\,\mathcal{H}_q}{\mathrm{d}\,p} = \log_q (q-1) - \log_q \frac{p}{1-p}$ qui s'annule en $p = \frac{q-1}{q}$ et $\max \mathcal{H}_q = 1$ en cette valeur. De plus, $\frac{\mathrm{d}^2\,\mathcal{H}_q}{\mathrm{d}\,p^2} = \frac{-1}{(\ln q) p (1-p)} < 0$; c'est donc une fonction convexe.

6. De Ralph Hartley, électronicien, qui proposa cette unité en 1928.
7. C'est l'inverse de la sigmoïde ou *fonction logistique*, fort utilisée notamment dans les modèles de réseaux neuronaux.

III.1.5 Inégalité de Gibbs

La notion d'entropie en thermodynamique est évidemment antérieure aux travaux de Shannon. Les travaux des physiciens Rudolf Julius Emmanuel Clausius, Ludwig Eduard Boltzmann et Josiah Willard Gibbs, notamment, menés dans les années 1860–70 ont été pour lui et ses prédécesseurs immédiats en théorie de l'information une source directe d'inspiration. En effet, l'entropie au sens physique, introduite par Rudolf Clausius en 1865, est une mesure du « désordre » interne d'un système, de la dégradation de son énergie. Dans un système désordonné, il n'émerge plus aucune structure : tous ses éléments sont indistinguables. L'entropie informatique, maximale lorsque tous les symboles sont équiprobables, exprime donc une idée semblable. *A contrario*, à température absolue nulle, l'entropie est nulle. Tous les états sont figés, ce qui est analogue à la situation ou un seul symbole peut apparaître (donc avec une probabilité 1), ce qui porte une information nulle.

Même sans pousser l'analogie trop loin, plusieurs propriétés « physiques » ont ainsi une transcription immédiate en théorie de l'information. L'inégalité dite « de Gibbs » en est un exemple. Pour la démontrer, nous aurons besoin du lemme suivant.

Lemme III.1.6. *Si* $b > 1$, $\log_b x \leq \frac{1}{\ln b}(x-1)$

Démonstration. La tangente à la courbe plane $y = \log_b x$ au point $(1,0)$ est la droite $y = \frac{1}{\ln b}(x-1)$. Comme elle a une courbure toujours de même signe ($\frac{d^2y}{dx^2} < 0$), elle reste toujours du même côté de cette tangente, plus précisément sous elle. ∎

Théorème III.1.7 (Inégalité de Gibbs)**.** *Pour toute source* S *de q symboles suivant une loi de probabilité* $\{p_i\}$ *et pour toute fonction* $f(s_i) = f_i$ *réelle positive définie sur ces mêmes symboles, on a*

$$\sum_{i=1}^{q} f_i \leq 1 \Rightarrow \sum_{i=1}^{q} p_i \log_b \frac{f_i}{p_i} \leq 0$$

Démonstration. En utilisant le lemme précédent, on a

$$\sum_{i=1}^{q} p_i \log_b \frac{f_i}{p_i} \leq \frac{1}{\ln b} \sum_{i=1}^{q} p_i \left(\frac{f_i}{p_i} - 1\right) = \frac{1}{\ln b} \sum_{i=1}^{q} f_i - \frac{1}{\ln b} \sum_{i=1}^{q} p_i \leq \frac{1}{\ln b} - \frac{1}{\ln b} = 0$$ ∎

L'entropie constitue ainsi un minimum fonctionnel des fonctions bornées, puisque l'inégalité de Gibbs peut se réécrire

$$\sum_{i=1}^{q} f_i \leq 1 \Rightarrow \mathrm{H(S)} \leq -\sum_{i=1}^{q} p_i \log_q f_i \tag{III.6}$$

Elle nous servira régulièrement sous cette forme.

III.2 Codage asymptotique

Puisque l'entropie est la quantité d'information propre moyenne portée par les symboles s_i, elle devrait indiquer également la longueur moyenne d'un codage *idéal*, celui conduisant à des fréquences d'apparition égales des différents caractères de codage c_j. Ceci résulte des propriétés ci-dessous.

III.2.1 Extension de la source

Une *extension d'une source* S de longueur n est l'ensemble S^n (constitué des n-uplets de S) muni de sa loi de probabilité : $\mathbb{P}[s_{i_1} s_{i_2} \ldots s_{i_n}] = p_{i_1} \cdot p_{i_2} \cdot \cdots \cdot p_{i_n}$. Celle-ci découle directement de l'hypothèse d'une *source markovienne*. Plutôt que de coder chaque symbole individuellement, on peut alors définir un mot du code pour chaque n-uplet et calculer l'entropie résultante $\mathrm{H}(\mathrm{S}^n)$.

Théorème III.2.1. *En toute base b,* $\forall \mathrm{S}, \forall n : \mathrm{H}_b(\mathrm{S}^n) = n\,\mathrm{H}_b(\mathrm{S})$

Démonstration. Elle se fait par récurrence sur n. C'est trivial pour $n = 1$.

Pour $n > 1$, on a $S^n = S \cdot S^{n-1}$, c'est-à-dire $\mathbb{P}[s_{i_1} s_{i_2} \dots s_{i_n}] = \mathbb{P}[s_1] \cdot \mathbb{P}[s_{i_2} \dots s_{i_n}]$. Donc

$$\begin{aligned}
H_b(S^n) &= -\sum_{i_1=1}^{q} \sum_{i_2=1,\dots i_n=1}^{q} p_{i_1} \cdot (p_{i_2} \cdots p_{i_n}) \log_b \left(p_{i_1} \cdot (p_{i_2} \cdots p_{i_n})\right) \\
&= -\sum_{i_1=1}^{q} \sum_{i_2=1,\dots i_n=1}^{q} p_{i_1} \cdot (p_{i_2} \cdots p_{i_n}) \left(\log_b p_{i_1} + \log_b (p_{i_2} \cdots p_{i_n})\right) \\
&= -\sum_{i_1=1}^{q} p_{i_1} \log p_{i_1} \sum_{i_2=1,\dots i_n=1}^{q} (p_{i_2} \cdots p_{i_n}) - \sum_{i_2=1,\dots i_n=1}^{q} (p_{i_2} \cdots p_{i_n}) \log_b (p_{i_2} \cdots p_{i_n}) \sum_{i_1=1}^{q} p_{i_1} \\
&= -\sum_{i_1=1}^{q} p_{i_1} \log_b p_{i_1} - \sum_{i_2=1,\dots i_n=1}^{q} (p_{i_2} \dots p_{i_n}) \log_b (p_{i_2} \dots p_{i_n}) \\
&= H_b(S) + H_b\left(S^{n-1}\right) \\
&= H_b(S) + (n-1) H_b(S)
\end{aligned}$$

par hypothèse d'induction. ∎

Ce théorème est conforme à l'intuition : il dit que la quantité d'information moyenne envoyée par paquets de n symboles vaut n fois celle de la même source envoyant ses symboles individuellement. Donc, si l'on mesure la quantité d'information moyenne « par symbole », elle reste la même.

Théorème III.2.2. *Pour toute source* S *et tout code* K *de longueur moyenne* $L_K(S)$ *sur un alphabet de r caractères :*

$$H_r(S) \leq L_K(S)$$

Démonstration. Notons ℓ_i les longueurs de mots de K ; par l'inégalité de Kraft, nous savons que $\sum_{i=1}^{q} r^{-\ell_i} \leq 1$. De $\ell_i = -\log_r r^{-\ell_i}$, on a

$$H_r(S) - L_K(S) = -\sum_{i=1}^{q} p_i \log_r p_i + \sum_{i=1}^{q} p_i \log_r r^{-\ell_i} = \sum_{i=1}^{q} p_i \log_r \frac{r^{-\ell_i}}{p_i} \leq 0$$

par l'inégalité de Gibbs (III.6). ∎

Ce théorème indique donc que l'entropie est une mesure du codage théorique le plus efficace en moyenne :

$$H(S) \leq \frac{L_K(S)}{\log_r q} \tag{III.7}$$

Le dénominateur $\log_r q$ représente en effet le nombre de caractères nécessaires pour coder dans l'alphabet de r caractères — *i.e. en base r* — le numéro d'un symbole. C'est un simple changement d'unité : la conversion entre une mesure faite dans l'espace des symboles (il y en a q) et une autre faite dans celui des caractères (il y en a r) ; l'entropie est calculée dans le premier, la longueur du code dans le second.

III.2.2 Premier théorème de Shannon

Ce théorème est également appelé le « **théorème du codage sans bruit de la source** » (*Shannon's Noiseless Coding Theorem*, en anglais). Il relie entre elles les propriétés précédentes et montre que l'on peut s'approcher asymptotiquement de l'efficacité maximale, celle qui est égale à l'entropie de la source.

Théorème III.2.3 (Claude Shannon, 1948)**.** *Pour toute source* S *de longueur moyenne minimale* $L_{\min}(S)$*,*

$$H_r(S) \leq L_{\min}(S) \leq H_r(S) + 1$$

et l'on n'atteint la borne supérieure que pour une source dégénérée (i.e. $\exists i : p_i = 1$*), pour laquelle* $H = 0$ *et* $L = 1$*. De plus,*

$$\lim_{n \to \infty} \frac{L_{\min}\left(S^n\right)}{n} = H_r(S) = \frac{H(S)}{\log_q r} \tag{III.8}$$

On note que $\frac{L_{min}(S^n)}{n}$ est la longueur moyenne minimale de codage par symbole de S.

Démonstration. Par le théorème III.2.2, nous savons que $\forall K, H_r(S) \leq L_K(S)$; donc $H_r(S) \leq L_{min}(S)$, puisqu'il existe des codes atteignant cette longueur minimale (*cf.* corollaire II.4.6).

Nous devons démontrer que $L_{min}(S) < H_r(S) + 1$ lorsque $0 < p_i < 1, \forall i$.

Choisissons pour K le code de Shannon; par définition $\ell_i = \lceil -\log_r p_i \rceil < -\log_r p_i + 1$. Pour lui,

$$L_K(S) = \sum_{i=1}^{q} p_i \cdot \ell_i < \sum_{i=1}^{q} p_i (-\log_r p_i + 1) = -\sum_{i=1}^{q} p_i \log_r p_i + 1 = H_r(S) + 1$$

Comme il existe un code de longueur inférieure à la borne, la longueur minimale possible $L_{min}(S)$ l'est aussi.

Pour démontrer (III.8), on utilise ce premier résultat et le théorème III.2.1:

$$n\, H_r(S) = H_r\left(S^n\right) \leq L_{min}(S^n) \leq H_r(S^n) + 1 = n\, H_r(S) + 1$$

Donc, $\forall n$:

$$H_r(S) \leq \frac{L_{min}(S^n)}{n} \leq H_r(S) + \frac{1}{n}$$

ce qui fournit le résultat par passage à la limite, une convergence en $O\left(\frac{1}{n}\right)$. ∎

III.2.3 Codage, efficacité et entropie

Le premier théorème de Shannon dit que si l'on dispose d'autant de caractères que de symboles ($\log_q r = 1$) et si l'on choisit un code efficace pour une source étendue, la longueur moyenne *par symbole* est asymptotiquement celle de l'entropie de la source. On pourra donc en être aussi proche que l'on veut, bien sûr, au prix d'un nombre exponentiellement croissant de mots du code différents, un pour chaque n-uplet de symboles.

Illustrons cela sur une source binaire ($q = r = 2$) où $\mathbb{P}(0) = \frac{3}{4}$, donc $\mathbb{P}(1) = \frac{1}{4}$, et examinons l'effet sur la longueur moyenne *par symbole encodé* du code de Huffman (*cf.* tableau III.1). On observe la convergence rapide de cette longueur vers l'entropie de la source.

TABLEAU III.1 – Extension de la source et code de Huffman

$n = 1$		
Symboles:	0	1
Probabiltés:	3⁄4	1⁄4
Mots du code:	0	1
Longueur:	$1/1 \cdot 4/4 = 1$	

$n = 2$				
Symboles:	00	01	10	11
Probabiltés:	9⁄16	3⁄16	3⁄16	1⁄16
Mots du code:	0	11	100	101
Longueur:	$1/2 \cdot 27/16 = 0.84375$			

$n = 3$								
Symboles:	000	001	010	100	011	101	110	111
Probabiltés:	27⁄64	9⁄64	9⁄64	9⁄64	3⁄64	3⁄64	3⁄64	1⁄64
Mots du code:	1	001	010	011	00000	00001	00010	00011
Longueur:	$1/3 \cdot 158/64 = 0.82291666...$							

$n = \infty$	
Entropie:	$2 - \frac{3}{4}\log 3 \approx 0.811278124459...$

De manière plus générale, que peut-on dire d'autre de la longueur moyenne des codes — que nous avons étudiée par comparaison à l'entropie — par rapport à l'information propre des symboles correspondants ?

Nous supposons $q > 1$. Pour atteindre l'entropie comme longueur moyenne, c'est-à-dire la longueur moyenne minimale, l'idéal en première approche serait d'avoir un code K pour lequel la longueur des mots vaut exactement $\ell_i = \mathcal{I}_r(p_i) = -\log_r p_i$, mais cela ne correspond pas nécessairement à un nombre entier de caractères !

Dans le code de Shannon, $\ell_i = \lceil \mathcal{I}_r(p_i) \rceil$. On a donc $\mathcal{I}_r(p_i) \leq \ell_i < \mathcal{I}_r(p_i) + 1$ et, comme on l'a vu dans la démonstration du théorème III.2.3, $H_r \leq L < H_r + 1$. De même, les codes de Huffman étant efficaces, on a également $H_r \leq L < H_r + 1$. La longueur moyenne du code de Shannon-Fano étant entre celles de ces deux codes, elle satisfait également cette inégalité.

Chapitre IV

Compression sans perte

Les *algorithmes de compression sans perte*[1] sont un cas particulier de codes d'une source aléatoire. On se place en fait dans un contexte où les symboles sont déjà des caractères informatiques ; bien souvent, l'ensemble des symboles et l'alphabet du code sont identiques ($S = C$, donc $q = r$), même si pour pouvoir compresser un message, on tentera souvent de coder toute une sous-chaîne de message en un seul mot de code binaire ($S = C^*$ et $r = 2$).

Les algorithmes de codages supposent que l'on connaisse la loi de probabilité d'apparition des différents symboles. Ici, si l'on veut compresser un message donné, on va devoir *estimer* ces probabilités, par exemple par la fréquence d'apparition des caractères au sein du message. Mais cette approche a deux défauts :

- D'une part, il faudrait examiner préalablement tout le message à coder (à compresser), ce qui n'est donc pas adapté à une codification en flux, comme dans le cas d'une transmission que l'on souhaite traiter à la volée.
- D'autre part, cela nécessiterait de stocker ou de transmettre non seulement le message codé (compressé), mais également l'arbre de code utilisé, ce qui peut réduire à néant l'effort de réduction de la longueur du message originel.

Les algorithmes de compression *sans perte* visent à contourner ces deux difficultés, tout en garantissant un *décodage parfait*, une restitution exacte du message original. Ils exploitent le premier théorème de Shannon (théorème III.2.3), donc l'intérêt de l'extension de la source, en transformant des sous-chaînes par un seul mot de code.

Le *taux de compression* est le rapport entre la différence de longueurs des messages original et compressé comparé à la longueur initiale, en supposant une même base de codage, traditionnellement binaire. Les méthodes sont généralement comparées par leur *taux moyen* sous l'hypothèse d'une distribution *a priori* des messages, c'est-à-dire des *chaînes de symboles*. On ne fait donc plus ici d'hypothèse markovienne sur l'apparition isolée de ces symboles.

IV.1 Codage par dictionnaire

Suivant l'idée que l'extension de la source permet d'approcher un bon taux de compression (*cf.* théorème III.2.3), les algorithmes de cette famille gèrent un tableau (*array*) de *chaînes de symboles* sources appelé *dictionnaire* et ces chaînes sont alors codées par leur indice dans ce dictionnaire. La compression est obtenue en y plaçant les chaînes de symboles les plus longues (pour accroître la compression) parmi celles qui sont *utiles*, celles qui apparaissent effectivement (pour limiter la taille du dictionnaire, donc aussi celle des indices). En pratique, le contenu du dictionnaire évolue en fonction des symboles rencontrés à la compression et l'algorithme de décodage doit être à même de le reconstituer progressivement.

1. Par opposition à la *compression avec perte* où le message initial ne peut plus être intégralement restitué par décodage. Ces derniers algorithmes sont utilisés pour comprimer des informations de type image, son ou vidéo où, pour des raisons physiologiques, on peut accepter une légère dégradation de l'original.

Malheureusement, le dictionnaire peut devenir très, voire trop grand, ce qui impose en pratique de limiter arbitrairement sa taille ; on arrête alors toute possibilité de compression de nouvelles chaînes.

IV.1.1 Algorithme de Lempel-Ziv

Deux versions sont connues sous les acronymes LZ77 ([47]) et LZ78 ([48]) en fonction de leur année de publication par Abraham Lempel (אברהם לפמל) et Jacob Ziv (יעקב זיו) ; nous présentons ici la première version, l'autre sera plutôt décrite sous sa forme améliorée (LZW).

L'algorithme **LZ77** travaille avec une *fenêtre glissante* (*sliding window*) contenant les w derniers symboles traités. Pour coder les suivants, il y cherche le plus long préfixe de l'input. L'output, le codage, est constitué de triplets (ℓ, d, s) :

ℓ La *longueur* du préfixe trouvé ; 0, s'il est absent.

d Sa *distance* d'apparition antérieure (son « âge »), *i.e.* sa position dans la fenêtre à partir de la fin ; si $\ell = 0$, elle est notée ici $\perp$, mais cette valeur est indifférente [2], car jamais utilisée.

s Le nouveau *symbole* d'input, celui qui suit ce préfixe reconnu.

C'est donc un code en bloc. Le dictionnaire est en fait constitué de toutes les sous-chaînes du contenu de la fenêtre.

Ensuite, cette fenêtre « glisse » pour s'achever sur ce nouveau symbole (inclus) ; les $(\ell + 1)$ plus anciens sont donc oubliés.

L'algorithme de codage a besoin non seulement de sa fenêtre, son dictionnaire évolutif, mais également d'un *tampon* (*buffer*) de lecture anticipée de longueur b, ce qui est un autre paramètre de l'algorithme. Il limite la longueur maximale des sous-chaînes codables ($\ell \leq b$), mais cette longueur peut être supérieure à celle de la fenêtre. L'astuce consiste en effet à considérer que la fenêtre et le tampon sont concaténés, ce qui permet d'identifier des sous-chaînes dont la fin peut dépasser celle de la fenêtre. L'algorithme de codage utilise donc un espace de travail de $(w + b)$ caractères. Ainsi : $0 \leq \ell \leq b$, $0 \leq d < w$ et $0 \leq s < q$ (en codant le symbole par son numéro), donc $\log_r(b+1) + \log_r w + \log_r q$ caractères par triplet d'output.

Pour décoder, on recrée la fenêtre à la volée, ce qui permet de reconstituer le message symbole par symbole en parcourant la fenêtre à partir du début de la distance codée sur la longueur indiquée, en réinsérant chaque fois le symbole produit en fin de fenêtre. Une chaîne dépassant la fenêtre ne constitue donc pas un cas particulier. Le symbole du triplet est ensuite traité : output et insertion en fin de fenêtre. L'algorithme de décompression n'utilise donc pas de tampon, uniquement la fenêtre de w caractères.

L'efficacité de la compression est donc liée à la présence ou non de nombreuses sous-chaînes identiques *dans un voisinage proche*. En fait, l'algorithme LZ77 ne contient qu'un dictionnaire tronqué (de taille fixe) et partiel (il ne contient que les sous-chaînes contenues dans les w derniers caractères du message), ce qui fait qu'il oublie les anciennes occurrences.

Quelques choix d'implantation ont été suggérés dès le début : en cas d'occurrences multiples dans la fenêtre, choisir l'occurrence la plus récente (d minimum), pour privilégier la localité, et surtout initialiser la fenêtre de la valeur du premier caractère du message (ce qui fait gagner une étape).

L'efficacité de la compression souffre principalement de la présence de caractères absents de la fenêtre (triplets $(0, \perp, c)$). Une amélioration, connue sous le nom **LZSS**, a été suggérée par James Andrew Storer et Thomas Gregory Szymanski ([36]). Elle consiste à insérer un bit en début de code produit permettant de distinguer ce cas et de ne coder alors que le caractère (le surcoût est alors de 1 bit pour tout output, plutôt que $\log(b+1) + \log w$ pour ces cas particuliers) [3]. Le code résultant n'est plus un code en bloc, mais un code sans préfixe.

L'algorithme LZ77 est utilisé dans de nombreux outils ou formats (PKZIP, ARJ, LHA, PNG…) ; sous le nom « deflate », il est utilisé par le système de fichiers NTFS, entre autres.

Exemple. Soit le message "ABRABABABRABABACACABABA" de 23 caractères de l'alphabet $\{A, B, C, R\}$ ($q = 4$). Codons-le par LZ77 avec comme paramètres $w = 8$ et $b = 7$. Le tableau IV.1 présente les différentes étapes du codage.

2. On n'a donc pas besoin d'une valeur distincte, ce qui permet de coder d en $\lceil \log_r |w| \rceil$ caractères.

3. Pour être complet, l'algorithme LZSS utilise également une représentation plus efficace du dictionnaire pour améliorer le temps de recherche de la sous-chaîne au codage. Celle-ci a été reprise dans l'algorithme LZW décrit plus loin.

TABLEAU IV.1 – Codage par LZ77

État														Code		
Win.								Buf.						ℓ	d	s
A	A	A	A	A	A	A	A	A	B	R	A	B	A	B	1	0	B
A	A	A	A	A	A	A	B	R	A	B	A	B	A	B	0	⊥	R
A	A	A	A	A	A	B	R	A	B	A	B	A	B	R	2	2	A
A	A	A	B	R	A	B	A	B	A	B	R	A	B	A	3	1	R
R	A	B	A	B	A	B	R	A	B	A	B	A	C	A	5	6	C
B	R	A	B	A	B	A	C	A	C	A	B	A	B	A	3	1	B
A	B	A	C	A	C	A	B	A	B	A	$^{\mathrm{N}}\mathrm{U}_{\mathrm{L}}$				3	1	$^{\mathrm{N}}\mathrm{U}_{\mathrm{L}}$

Bien sûr, la compression n'est pas efficace sur un exemple si court[4], car la longueur du message, ainsi que les tailles de la fenêtre et du tampon sont vraiment trop petites. Mais, comme l'ont montré ses auteurs (*cf.* [47]), l'efficacité asymptotique de ce code — et de toutes ses améliorations, bien sûr — est optimale : pour une fenêtre et un tampon suffisamment longs, la longueur du message compressé tend vers celle d'un code de Huffman lorsque la longueur du message donné augmente.

Par comparaison, si l'on compresse le message `"ABRABABABRABABABRABACACACBBACACA"` avec $w = 8$ et $b = 15$[5], on passe de $32 \times 2 = 64$ bits à $7 \times (4 + 3 + 2) = 63$ bits, mais on est encore loin des tailles suffisantes pour que la méthode produise ses effets.

Variantes

L'autre version, l'algorithme **LZ78** utilise non une fenêtre glissante, mais un dictionnaire de sous-chaînes *global*, permettant donc de repérer des répétitions plus espacées dans la source. Pour être efficace, ce dictionnaire est représenté comme un *trie*, c'est-à-dire un arbre préfixe dont les feuilles correspondent aux plus longues sous-chaînes trouvées dans l'input. Contrairement aux arbres de code sans préfixe, dans un *trie*, on ne se limite pas aux seules feuilles : tous les nœuds correspondent également à des sous-chaînes du dictionnaire, sans devoir dupliquer les informations (la racine correspond à la chaîne de longueur nulle). Cette variante a été brevetée (aux USA), ce qui fait qu'elle a été moins largement utilisée que la version originelle.

Une autre variante, enfin, appelée **LZMA**, optimise la représentation du dictionnaire en tenant compte de la corrélation observée entre chaque caractère (ou bit) et ceux qui l'y précèdent. Les octets ne sont donc plus encodés comme tels, mais comme outputs d'une chaîne de Markov. LZMA est utilisé notamment dans le format de compression 7z.

IV.1.2 Algorithme de Lempel-Ziv-Welch

Les deux algorithmes LZ77 et LZ78 souffrent de devoir transmettre non seulement le codage d'une sous-chaîne, sa référence dans le dictionnaire, mais également les symboles à y ajouter. Une amélioration consiste à éviter cela en supposant le codage des symboles isolés, des chaînes de longueur 1, fixé *a priori* : ils sont déjà tous présents dans le dictionnaire dès l'initialisation. Cette méthode améliorée est connue sous l'acronyme **LZW** ; elle a été publiée en 1984 par Terry Archer Welch ([45]).

L'algorithme LZW a été initialement breveté par la société Unisys aux USA, mais il a été très vite utilisé dans de nombreux contextes : les transmissions analogiques (protocole V42bis des modems), les formats GIF, TIFF, MOD, PDF, etc.

a) Encodage

Au départ, chaque symbole est donc codé simplement par lui-même, c'est-à-dire par sa représentation numérique qui est sa position dans le dictionnaire ; dans le *trie*, ils sont considérés comme les

4. Le message occupe $23 \times 2 = 46$ bits, le « compressé » en nécessite $7 \times (3 + 3 + 2) = 56$ bits !
5. Ceci est laissé comme exercice.

fils directs de la racine virtuelle représentant la chaîne vide. Typiquement, on considère pour cela des symboles correspondant à un octet et le dictionnaire initial contient ainsi 256 entrées (aux positions 0 à 255)[6]. Cela ne produit ni compression ni allongement du message.

Ensuite, on recherche dans le dictionnaire le plus long préfixe w de l'input ; l'output, le codage produit correspond à l'indice de cette chaîne dans le dictionnaire, celui du nœud (ou feuille) associé à son dernier symbole. On allonge ensuite cette chaîne dans le dictionnaire avec le symbole d'input c qui suit ce préfixe reconnu : on accroche à ce nœud correspondant au préfixe reconnu un fils étiqueté par ce symbole. La position du fils est fixée de manière déterministe, puisqu'elle correspond à la taille du dictionnaire qui croît exactement d'une entrée lors de chaque output correspondant à une occurrence d'un suffixe inconnu. La taille finale du dictionnaire, l'espace de travail de l'algorithme, est donc linéairement proportionnelle à la taille de l'input traité et majorée par celle-ci.

Reste la question de la représentation (binaire, en général) des mots du code, c'est-à-dire des indices des entrées du dictionnaire. Au départ, il suffit de pouvoir représenter un nombre entier inférieur au nombre de symboles ($\lceil \log q \rceil$, précisément). Au moment où la taille du dictionnaire va dépasser la limite codable, l'algorithme accroît *pour la suite* la taille de la représentation d'un incrément fixé (1 bit, 8 bits, parfois plus) ; c'est un paramètre de l'algorithme. Au décodage, cette même opération peut être effectuée de manière synchrone, ce qui garantit la bonne interprétation du flux binaire à décoder. Le code LZW est donc un code en bloc, mais de taille croissante par paliers déterministes.

Exemple. Reprenons le même message "ABRABABABRABABACACABABA" de 23 caractères. Pour coder par l'algorithme LZW (voir tableau IV.2), nous représentons le *trie* par un tableau d'enregistrements à 3 entrées : le contenu du nœud (un seul symbole), l'indice du frère et celui du fils (0 si absent). La racine n'y est pas présente : le *trie* est initialisé par l'alphabet (aux positions 0 à 3).

La colonne i du tableau indique le nombre de caractères lus (la position du caractère c dans l'input), la colonne t la taille du *trie* (*i.e.* le nombre de sous-chaînes dans le dictionnaire en début d'étape), la colonne w la sous-chaîne reconnue (présente dans le *trie*), la colonne c le caractère à lui insérer en suffixe, enfin la colonne o le code produit, l'output de l'algorithme.

La taille du dictionnaire vaut 4 au départ ; le code initial est donc de 2 bits. Ensuite, il passera à 3, puis 4 bits respectivement après les étapes $t = 4$ et $t = 8$. La séquence binaire d'output est donc "00 001 011 100 111 0101 1000 0000 0010 1011 1010" longue de 38 bits[7] (les espaces sont purement typographiques pour indiquer les limites des symboles, mais ils ne sont pas présents dans la séquence binaire).

b) Décodage

L'algorithme de décodage fonctionne de la même façon en reconstruisant localement le dictionnaire à la volée de façon analogue. À chaque étape, il produit la chaîne correspondant au codage reçu et il insère le premier caractère de son output courant comme nouvelle feuille en suffixe de l'output précédent (la feuille sur laquelle il était à l'étape précédente).

Un cas particulier survient lorsque le code reçu est précisément celui qui ne figure pas encore dans le dictionnaire (*i.e.* le code reçu est la taille du dictionnaire). Comme l'avait montré Welch, cela se produit pour un message de la forme $\ldots c\omega c\omega c$ où ω est une chaîne quelconque, c un caractère et $c\omega$ figure déjà au dictionnaire. Dans ce cas, l'output est $c\omega c$ dont le suffixe manquant est précisément le caractère traité à l'étape précédente, le suffixe de la sous-chaîne introduite. Cette situation sera illustrée ci-dessous.

Exemple. Le tableau IV.3 illustre les étapes de décodage de l'exemple précédent. La colonne i est l'input reçu, la colonne t est la taille du dictionnaire (en début d'étape). On insère c, le caractère « courant », comme suffixe de w, l'output à l'étape précédente ; c est le premier caractère de l'output o, chaîne dans le dictionnaire correspondant à l'input. À l'étape 7, l'output n'est pas encore présent dans le dictionnaire, mais on sait que, dans ce cas, c reste le même qu'à l'étape précédente.

6. Pour des raisons d'efficacité si le nombre de symboles est grand, on peut toujours supposer que les fils sont triés par ordre croissant.

7. Le message initial nécessitait $23 \times 2 = 46$ bits, soit une légère compression de $8/46 \approx 17.4$ %.

TABLEAU IV.2 – Codage par LZW

i	*t*	*Trie*											*w*	*c*	*o*
		Car.	A	B	C	R									
1	4	Frère	1	2	3	0							A	B	0
		Fils	0	0	0	0									
		Car.	A	B	C	R	B								
2	5	Frère	1	2	3	0	0						B	R	1
		Fils	4	0	0	0	0								
		Car.	A	B	C	R	B	R							
3	6	Frère	1	2	3	0	0	0					R	A	3
		Fils	4	5	0	0	0	0							
		Car.	A	B	C	R	B	R	A						
5	7	Frère	1	2	3	0	0	0	0				AB	A	4
		Fils	4	5	0	6	0	0	0						
		Car.	A	B	C	R	B	R	A	A					
8	8	Frère	1	2	3	0	0	0	0	0			ABA	B	7
		Fils	4	5	0	6	7	0	0	0					
		Car.	A	B	C	R	B	R	A	A	B				
10	9	Frère	1	2	3	0	0	0	0	0	0		BR	A	5
		Fils	4	5	0	6	7	0	0	8	0				
		Car.	A	B	C	R	B	R	A	A	B	A			
14	10	Frère	1	2	3	0	0	0	0	0	0	0	ABAB	A	8
		Fils	4	5	0	6	7	9	0	8	0	0			
		Car.	A	B	C	R	B	R	A	A	B	A	A		
15	11	Frère	1	2	3	0	0	0	0	0	0	0	0	A	C	0
		Fils	4	5	0	6	7	9	0	8	10	0	0			
		Car.	A	B	C	R	B	R	A	A	B	A	A	C	
16	12	Frère	1	2	3	0	0	0	0	0	0	0	0	4	C	A	2
		Fils	11	5	0	6	7	9	0	8	10	0	0	0			
		Car.	A	B	C	R	B	R	A	A	B	A	A	C	A		
18	13	Frère	1	2	3	0	0	0	0	0	0	0	0	4	0	AC	A	11
		Fils	11	5	12	6	7	9	0	8	10	0	0	0	0			
		Car.	A	B	C	R	B	R	A	A	B	A	A	C	A	A		
23	14	Frère	1	2	3	0	0	0	0	0	0	0	0	4	0	0	ABABA	NUL	10
		Fils	11	5	12	6	7	9	0	8	10	0	0	13	0	0			

TABLEAU IV.3 – Décodage par LZW

i	*t*	*w*	*c*	*o*
0	4	—		A
1	4	A	B	B
3	5	B	R	R
4	6	R	A	AB
7	7	AB	?	$\underline{\text{AB}}\underline{\text{A}}$
5	8	ABA	B	$\overline{\text{BR}}$
8	9	BR	A	ABAB
0	10	ABAB	A	A
2	11	A	C	C
11	12	C	A	AC
10	13	AC	A	ABABA

IV.2 Codage de Huffman adaptatif

Une autre approche consiste à utiliser un code de Huffman — que l'on sait efficace — en modifiant continûment ce code au fur et à mesure que les probabilités des symboles se précisent. Ces probabilités sont estimées à la volée à partir des fréquences d'apparition des symboles déjà rencontrés.

Comme dans l'algorithme LZW, il n'y a aucune transmission de l'arbre du code, puisque le décodage le reconstruit localement pas à pas pour chaque symbole traité. Mais en plus ici, il n'y a pas de risque d'explosion, puisque nombre de feuilles de l'arbre, donc son nombre de nœuds également, est limité par le nombre de symboles présents dans le message. En pratique, on fixe ce paramètre par le choix de l'espace de symboles considéré, vu que pour accroître le taux de compression, il peut être utile de considérer comme symbole des chaînes de plusieurs caractères (des digrammes, des trigrammes...) plutôt que des caractères isolés.

La méthode la plus connue d'un tel codage est celle de Faller-Gallager-Knuth ou algorithme **FGK**; elle est due à Newton Faller en 1973 ([8]), améliorée par Robert Gray Gallager en 1978 ([12]), puis par Donald Ervin Knuth en 1985 ([22]). Mais elle présente quelques inconvénients, quelques opérations internes sur l'arbre de code peu efficaces; voilà pourquoi on lui préfère généralement l'amélioration de Jeffrey Scott Vitter (de 1987 [43]) présentée ci-dessous.

IV.2.1 Invariant de Gallager

Dans son étude des codages de Huffman adaptatifs, Robert G. Gallager a mis à jour une propriété intéressante des arbres de code correspondants.

Définition (Propriété de fratrie)**.** On dit qu'un arbre de code satisfait la propriété de fratrie (*sibling property*) si tous ses sommets ont un frère, à l'exception bien sûr de la racine, et si l'on peut les ordonner totalement par probabilités non décroissantes en plaçant les frères de manière contiguë.

Ainsi, un arbre de code satisfaisant la propriété de fratrie peut être parcouru de bas en haut et de gauche à droite en rencontrant ses sommets par probabilités non décroissantes.

Théorème IV.2.1 (Robert G. Gallager, 1978)**.** *Un arbre de code est de Huffman si et seulement s'il satisfait la propriété de fratrie.*

La démonstration se trouve dans [12].

Les algorithmes de codage de Huffman adaptatif sont donc tous conçus pour garantir l'invariance de cette condition nécessaire et suffisante.

IV.2.2 Invariant de Vitter

L'amélioration de Jeffrey Scott Vitter ([43]) vise à construire, parmi tous les arbres de Huffman possibles, celui qui sera algorithmiquement le plus efficace pour le codage et le décodage. Il s'agit non seulement de minimiser la longueur moyenne ($\sum \ell_i p_i$), ce que Vitter appelle le *coût de communication*, mais également la hauteur de l'arbre ($\max \ell_i$) et la somme des longueurs ($\sum \ell_i$), sa « densité » ou compacité. Elle est fondée sur la propriété suivante.

Théorème IV.2.2 (Jeffrey S. Vitter, 1987)**.** *Si dans le parcours par niveau d'un arbre de Huffman, de bas en haut et de gauche à droite, on rencontre, parmi les sommets de même probabilité, toutes les feuilles avant tous les nœuds, alors cet arbre est de hauteur minimale et minimise le coût de communication.*

La démonstration se trouve dans [43] .

Cet ordre de parcours est appelé par Vitter « énumération implicite », par opposition à l'ordre « explicite » de fusion des sous-arbres durant la construction de l'arbre de code de Huffman.

IV.2.3 Algorithme Λ

L'algorithme « Lambda », noté « Algorithme **Λ** » par Jeffrey Vitter retournant son initiale (donc souvent appelé « algorithme de Vitter »), maintient, comme dans l'algorithme FGK, un arbre de code à partir des symboles déjà codés. C'est un arbre de Huffman *binaire* où chaque sommet (feuille ou nœud) est enrichi de deux informations: son numéro d'ordre implicite et le bloc (voir ci-dessous) auquel il appartient.

- Comme dans tout arbre de Huffman, seules les feuilles sont associées à un symbole et elles contiennent le nombre d'occurrences de ce symbole ; chaque nœud contient la somme des occurrences de ses descendants directs (ce qui indique le nombre de fois qu'un des symboles de son sous-arbre a été rencontré) [8].
- Comme déjà dans l'algorithme FGK, l'arbre contient une feuille supplémentaire d'occurrence nulle représentant le *symbole inconnu*, celui qui n'a pas encore été rencontré.
- Les sommets de l'arbre (ses feuilles et ses nœuds) sont tous identifiés par leur *numéro* d'ordre d'insertion dans l'arbre (correspondant ici peu ou prou à l'inverse de leur ordre implicite). Notons qu'un sommet est toujours créé comme une nouvelle feuille (pour contenir un symbole nouveau d'occurrence 1), mais son statut peut changer plusieurs fois par la suite.
- Ces sommets sont regroupés en blocs : chaque *bloc* est une classe d'équivalence des sommets de même nombre d'occurrences et de même type, *i.e.* feuille ou nœud. Chaque bloc est représenté par son sommet d'ordre implicite le plus élevé, appelé *aïeul* ou chef de file (*leader*).
- Dans l'ordre implicite, les blocs sont ordonnés (et liés entre eux) par nombre d'occurrences croissant, un bloc de feuilles précédant un bloc de nœuds, dit bloc *interne*, de même nombre d'occurrences. Cet ordre fait donc alterner bloc de feuilles et bloc interne, certains blocs pouvant être vides (notamment toujours celui des nœuds de poids nul).

Lors du traitement de chaque symbole, il faut incrémenter son nombre d'occurrences ou bien l'insérer s'il n'avait pas encore été rencontré. À ce moment, l'arbre n'est donc plus nécessairement celui d'un code de Huffman. Plutôt que de reconstruire complètement un nouvel arbre à chaque étape, les algorithmes FGK et Λ effectuent *une simple mise à jour* de l'arbre existant le long du chemin correspondant au mot du code émis. Cette mise à jour consiste en deux opérations :

1. Incrémenter le compteur d'occurrences de chaque sommet du chemin vers la racine.
2. Rebalancer l'arbre en déplaçant si nécessaire ce sommet vers une place conforme aux critères d'un arbre de Huffman par un simple échange du sous-arbre dont il est la racine avec celui de sa destination. Notons qu'on déplace des sous-arbres, pas des sommets.

L'amélioration de l'algorithme Λ sur celui de FGK consiste à utiliser les blocs pour sélectionner un rebalancement satisfaisant les propriétés des théorèmes IV.2.1 et IV.2.2. Après incrément, le sommet doit quitter son bloc pour rejoindre le suivant par occurrences croissantes, celui correspondant à son nouveau nombre d'occurrences. Ceci revient à lui faire « sauter » le bloc qui suit directement le sien, puisque si c'est une feuille, ce bloc est interne correspondant à son ancien nombre d'occurrences et si c'est un nœud, c'est un bloc de feuilles ayant son nouveau nombre d'occurrences. On place alors le sommet en fin de son nouveau bloc, c'est-à-dire juste avant l'aïeul du bloc sauté (s'il existe). Cet aïeul sert ainsi à identifier la destination du sommet. Ce déplacement en fin de son nouveau bloc correspond à une *insertion linéaire* dans la liste des sommets groupés par bloc, opération optimisée par la structure de données et méthode de Vitter.

Pplus précisément, l'arbre au départ est constitué du seul sommet « symbole inconnu » qui constitue à lui seul son propre bloc de poids nul. Ensuite, pour chaque symbole traité, les deux opérations suivantes sont effectuées.

1. **Émission du code.** Le symbole à coder est recherché dans l'arbre [9] :
   - S'il y figure déjà, le mot du code associé est émis.
   - Sinon, le mot du code émis est celui du symbole inconnu, suivi du nouveau symbole « en clair ». Une nouvelle feuille associée à ce dernier symbole est insérée (avec l'occurrence 1) comme fille de suffixe « 1 » (*i.e.* fille droite) du symbole inconnu ; ce sommet « trouvé » est donc devenu un nœud. Une nouvelle feuille « symbole inconnu » est insérée comme fille de suffixe « 0 » (ou fille gauche) de ce nœud.
2. **Mise à jour de l'arbre.** On remonte dans l'arbre depuis le sommet dont le code a été émis (celui trouvé ou l'ex-inconnu) jusqu'à la racine pour mettre les poids de ce chemin à jour. Cela se fait de la façon suivante.
   - On incrémente le nombre d'occurrences du sommet courant.

8. Les arcs ou leur symbole extrémité portent bien sûr un caractère du code (ici, un bit), mais l'on convient simplement que le fils gauche vaut 0 et le droit 1, par exemple.
9. Cela suppose l'existence d'un tableau trié, d'une table de hachage ou d'un arbre de recherche pour ces symboles, structure mise à jour en parallèle avec l'arbre du code.

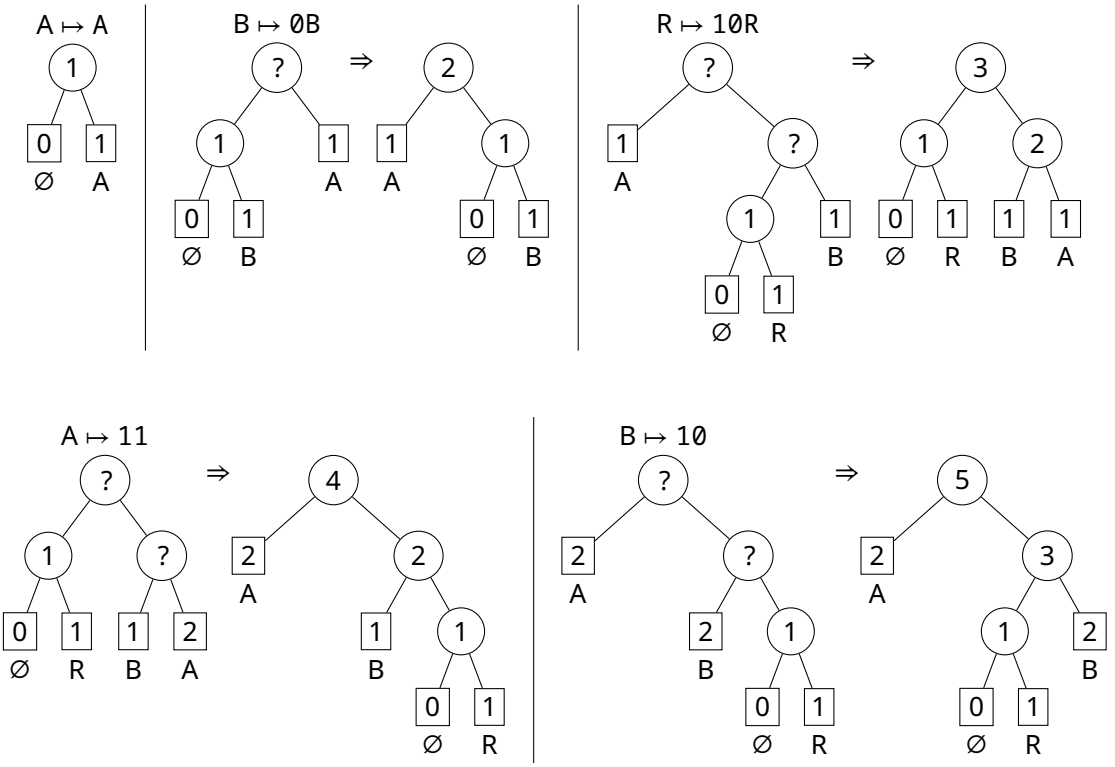
- Si ce sommet doit sauter le bloc suivant (par comparaison des statuts et des nombres d'occurrences), le sommet est inséré avant l'aïeul de ce bloc (les *contenus* sont échangés, donc leurs sous-arbres respectifs aussi). Ainsi, le sommet se retrouve entre ce bloc et le suivant.
- Si nécessaire, cette opération est répétée[10].
- On poursuit par le père du sommet dont le nombre d'occurrences a effectivement crû ; ce n'est pas toujours le sommet *courant*.

Comme Vitter l'a montré, son amélioration réduit le nombre de rebalancements nécessaires en cours de remontée vers la racine par rapport à l'algorithme FGK. En particulier, un sommet ne peut remonter que d'un seul niveau au maximum en cours de mise à jour.

Notons enfin que le décodage se fait en suivant le même algorithme, c'est-à-dire en émettant le symbole correspondant à la feuille trouvée ou celui transmis juste après le code, si cette feuille est celle du symbole inconnu. Dans le flux binaire, on ne peut donc jamais confondre un mot du code et un symbole en clair ; le code résultant à chaque étape est bien un code préfixe[11]. Ensuite, la mise à jour de l'arbre est faite de façon similaire.

Exemple. Reprenons une fois encore le message `"ABRABABABRABABACACABABA"` de 23 caractères. On démarre avec un arbre contenant le seul symbole inconnu (noté ∅) comme racine. Les premières étapes sont illustrées au schéma IV.1.

Schéma IV.1 – Cinq premières étapes de l'algorithme de Vitter

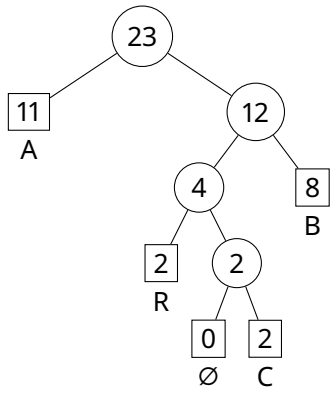


On constate plusieurs rebalancements. Ceux-ci sont plus fréquents en début de traitement, puisqu'ils correspondent essentiellement à un changement de hiérarchie dans la fréquence des symboles déjà rencontrés. Ainsi, après cette 5e étape, un seul rebalancement se produira encore : lors du traitement du 18e symbole (la deuxième occurrence du caractère C). L'arbre obtenu en fin de message est illustré au schéma IV.2.

Le code produit est `"A 0B 10R 11 10 0 11 0 11 101 0 11 0 11 0 100C 0 1001 0 11 0 11 0"`. Il nécessite 46 bits (38 bits de code et 4 symboles codables par 2 bits), soit aucune compression, car le message est un peu trop court. S'il avait été `"ABRABABABRABABABRABACACACBBACACA"`, comme

10. Comme dans la mise à jour d'une file à priorité (*heap*), ces échanges successifs peuvent bien sûr être optimisés en une seule permutation cyclique.
11. Que les caractères soient transmis par leur point de code (un entier) ou codés en UTF-8, par exemple.

SCHÉMA IV.2 – Arbre final du code de Vitter



ci-dessus, message de 32 caractères, on aurait obtenu un message codé en 60 bits, soit un taux de compression de $4/64 = 6.25$ %.

On constate que le code final est le code de Huffman incluant un symbole supplémentaire de fréquence nulle dont la longueur moyenne est $L = \sum \ell_i p_i = 41/23 \approx 1.783$, que la hauteur de l'arbre ou longueur maximale est $\max \ell_i = 4$ et que $\sum \ell_i = 14$. Par comparaison, le code de Huffman obtenu en deux passes, donc sans symbole de poids nul aurait eu pour valeurs respectives $L = 39/23 \approx 1.7$, à comparer à l'entropie $H = \log 23 - (11 \log 11 + 2 \log 3 + 28)/23 \approx 1.514$, $\max \ell_i = 3$ au moins et $\sum \ell_i = 9$ au moins, mais qu'il aurait fallu transmettre l'arbre de code en plus du codé (de 39 bits).

Chapitre V

Canal bruité

Les codes que nous avons étudiés jusqu'à présent réalisent une transcription directe et inversible de la source, mais en cas de détérioration du code produit, le décodage n'est plus possible ou erroné. Nous allons maintenant tenir compte du fait qu'un mot de code puisse être modifié tout en préservant, dans une certaine mesure, la possibilité de recouvrer le message originel. Ainsi, un même symbole peut être codé de plusieurs façons : le « vrai » mot du code et ses versions altérées, néanmoins décodables.

Ce contexte est celui du *canal bruité* introduit par Claude Shannon pour modéliser les erreurs de transmission, par opposition au contexte précédent dit de *canal sans bruit*.

V.1 Modèle

Nous supposons qu'un message est codé par un *émetteur* A (ou « Alice », pour les intimes) dans l'alphabet $\mathrm{X} = \{x_i\}$ avec $|\mathrm{X}| = m$, puis transmis *via* un média non fiable à un *destinataire* B (ou « Bob », le copain d'Alice) qui l'interprète dans un alphabet $\mathrm{Y} = \{y_j\}$ avec $|\mathrm{Y}| = n$. En pratique, dans ce modèle, ces ensembles X et Y sont plutôt les « codes » (les symboles x_i et y_j correspondent aux mots des codes à l'émission et à la réception) et Y est généralement plus vaste que X ($n > m$) pour englober les variantes dues aux erreurs de transmission.

V.1.1 Canal markovien

Pour simplifier le modèle, nous nous plaçons, comme précédemment, dans un contexte markovien : les caractères sont détériorés indépendamment les uns des autres[1]. Le canal de transmission bruité est ainsi défini par les seules lois de probabilité conditionnelle

$$p_{ij} = \mathbb{P}[y_j \,|\, x_i] \tag{V.1}$$

représentant la probabilité de recevoir y_j si x_i a été émis.

On a, bien sûr,

$$\forall i : \sum_{j=1}^{n} \mathbb{P}[y_j \,|\, x_i] = 1$$

V.1.2 Source aléatoire

Supposons, comme précédemment, que les messages émis respectent également une loi de probabilité donnée $p_i = \mathbb{P}[x_i]$ (avec $\sum_{i=1}^{m} p_i = 1$).

1. Cette hypothèse ne tient pas compte des erreurs en rafale (*burst*) qui touchent généralement plusieurs caractères contigus.

En combinant avec les p_{ij} définis par (V.1), cette hypothèse induit une loi de probabilité bidimensionnelle ou *distribution liée* (V.2) de X et Y et une loi de probabilité sur Y, dite *distribution marginale* (V.3) :

$$\mathbb{P}[x_i, y_j] = \mathbb{P}[x_i] \cdot \mathbb{P}[y_j \,|\, x_i] \tag{V.2}$$

$$\mathbb{P}[y_j] = \sum_{i=1}^{m} \mathbb{P}[x_i, y_j] \tag{V.3}$$

On note que la définition (V.2) — qui n'est, en fait, que le théorème de Thomas Bayes (publié en 1763) — est symétrique : $\mathbb{P}[x_i, y_j] = \mathbb{P}[y_j, x_i]$, puisqu'elle représente la probabilité jointe que le caractère émis et le caractère reçu aient simultanément une valeur fixée pour chacun : $\mathrm{X} = x_i$ et $\mathrm{Y} = y_j$.

De là, on déduit les lois de *probabilité bayésienne*, probabilité *estimée* de la source sur base du caractère reçu :

$$\mathbb{P}[x_i \,|\, y_j] = \frac{\mathbb{P}[x_i, y_j]}{\mathbb{P}[y_j]} \tag{V.4}$$

Nous disposons donc maintenant de 3 lois de probabilité $\{\mathbb{P}[x_i]\}$, $\{\mathbb{P}[y_j]\}$, $\{\mathbb{P}[x_i, y_j]\}$, des m lois $\{\mathbb{P}[y_j \,|\, x_i]\}$ et des n lois $\{\mathbb{P}[x_i \,|\, y_j]\}$, reliées entre elles par les relations précédentes.

V.2 Entropie croisée et conditionnelle

V.2.1 Entropie résiduelle

Comme pour tout i fixé, $\{p_{ij} = \mathbb{P}[y_j \,|\, x_i]\}$ est une loi de probabilité sur les symboles de Y, la définition (III.2) de l'entropie de Y selon cette loi donne directement

$$\forall i \,:\, \mathrm{H}(\mathrm{Y} \,|\, x_i) = -\sum_{j=1}^{n} \mathbb{P}[y_j \,|\, x_i] \cdot \log_n \mathbb{P}[y_j \,|\, x_i]$$

Cela représente le degré d'imprévu des symboles reçus *sachant avec certitude que x_i a été émis*. C'est l'entropie *résiduelle* sur Y sur base de cette information : « x_i a été émis » ; elle est indépendante de la probabilité d'émission $\mathbb{P}[x_i]$, donc ne dépend que du canal.

On a également la notion symétrique, plus naturelle,

$$\forall j \,:\, \mathrm{H}(\mathrm{X} \,|\, y_j) = -\sum_{i=1}^{m} \mathbb{P}[x_i \,|\, y_j] \cdot \log_m \mathbb{P}[x_i \,|\, y_j]$$

mesurant l'incertitude sur le symbole émis sachant que y_j a été effectivement reçu.

V.2.2 Entropie conditionnelle

Définition (Entropie conditionnelle)**.** Par définition, l'entropie conditionnelle est la moyenne des entropies résiduelles, c'est-à-dire

$$\mathrm{H}(\mathrm{X} \,|\, \mathrm{Y}) = \sum_{j=1}^{n} \mathbb{P}[y_j] \cdot \mathrm{H}(\mathrm{X} \,|\, y_j)$$

$$\mathrm{H}(\mathrm{Y} \,|\, \mathrm{X}) = \sum_{i=1}^{m} \mathbb{P}[x_i] \cdot \mathrm{H}(\mathrm{Y} \,|\, x_i)$$

Ces valeurs mesurent l'incertitude sur le symbole émis (respectivement reçu), quel que soit le symbole reçu (respectivement émis), mais en tenant compte de la loi de probabilité d'émission (respectivement de réception), puisque c'est une moyenne. Ce n'est pas la même chose que l'entropie marginale

H(X) (respectivement H(Y)) mesurant l'incertitude de l'émission (respectivement réception) indépendamment de la transmission.

Par simple substitution de la valeur de l'entropie résiduelle et du théorème de Bayes

$$\mathbb{P}[x_i, y_j] = \mathbb{P}[x_i] \cdot \mathbb{P}[y_j \,|\, x_i] = \mathbb{P}[y_j] \cdot \mathbb{P}[x_i \,|\, y_j]$$

on obtient

$$\mathrm{H}(\mathrm{X}\,|\,\mathrm{Y}) = -\sum_{i,j} \mathbb{P}[x_i, y_j] \cdot \log_m \mathbb{P}[x_i \,|\, y_j] \tag{V.5}$$

$$\mathrm{H}(\mathrm{Y}\,|\,\mathrm{X}) = -\sum_{i,j} \mathbb{P}[x_i, y_j] \cdot \log_n \mathbb{P}[y_j \,|\, x_i] \tag{V.6}$$

On voit ainsi clairement que, *a priori*, ces deux valeurs sont distinctes, puisqu'il s'agit de moyennes sur la même distribution liée, mais de valeurs différentes. Bien sûr, $0 \leq \mathrm{H}(\mathrm{X}\,|\,\mathrm{Y})$ et $0 \leq \mathrm{H}(\mathrm{Y}\,|\,\mathrm{X})$, puisque ce sont, par définition, des sommes de termes non négatifs.

V.2.3 Entropie croisée

Comme on l'a fait pour les entropies résiduelles ou marginales, on peut définir l'entropie sur les couples de symboles émis et reçus (x_i, y_j) sur base de leur loi bidimensionnelle $\{\mathbb{P}[x_i, y_j]\}$. C'est ce qu'on appelle l'*entropie croisée* ou jointe de X et Y :

$$\mathrm{H}(\mathrm{X},\mathrm{Y}) = \mathrm{H}(\mathrm{Y},\mathrm{X}) = -\sum_{i,j} \mathbb{P}[x_i, y_j] \cdot \log_{mn} \mathbb{P}[x_i, y_j] \tag{V.7}$$

Elle est naturellement symétrique, puisque $\{\mathbb{P}[x_i, y_j]\}$ l'est. C'est la mesure de l'incertitude sur l'ensemble des $m \cdot n$ couples de valeurs possibles.

Ces différentes notions sont liées par une propriété essentielle, semblable au théorème de Bayes pour les probabilités :

Propriété V.2.1 (Entropie croisée)**.**

$$\mathrm{H}_b(\mathrm{X},\mathrm{Y}) = \mathrm{H}_b(\mathrm{Y},\mathrm{X}) = \mathrm{H}_b(\mathrm{X}) + \mathrm{H}_b(\mathrm{Y}\,|\,\mathrm{X}) = \mathrm{H}_b(\mathrm{Y}) + \mathrm{H}_b(\mathrm{X}\,|\,\mathrm{Y})$$

Pour s'écrire simplement, cette propriété suppose que toutes les entropies s'expriment dans la même unité, dans une base commune b ; théoriquement $b = m \cdot n$.

Démonstration. Nous avons

$$\begin{aligned}
\mathrm{H}_b(\mathrm{X},\mathrm{Y}) &= -\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbb{P}[x_i, y_j] \log_b \mathbb{P}[x_i, y_j] \\
&= -\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbb{P}[x_i, y_j] \left(\log_b \mathbb{P}[x_i] + \log_b \mathbb{P}[y_j \,|\, x_i]\right) \\
&= -\sum_{i=1}^{m} \log_b \mathbb{P}[x_i] \left(\sum_{j=1}^{n} \mathbb{P}[x_i, y_j]\right) - \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbb{P}[x_i, y_j] \log_b \mathbb{P}[y_j \,|\, x_i] \\
&= -\sum_{i=1}^{m} \mathbb{P}[x_i] \log_b \mathbb{P}[x_i] - \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbb{P}[x_i, y_j] \log_b \mathbb{P}[y_j \,|\, x_i] \\
&= \mathrm{H}_b(\mathrm{X}) + \mathrm{H}_b(\mathrm{Y}\,|\,\mathrm{X})
\end{aligned}$$

La seconde égalité s'obtient, évidemment, de façon analogue. ∎

Si nous l'écrivions conformément à la base normalisée des différentes entropies, nous aurions

$$\mathrm{H}(\mathrm{X},\mathrm{Y}) = \frac{\mathrm{H}(\mathrm{X})}{1 + \log_m n} + \frac{\mathrm{H}(\mathrm{Y}\,|\,\mathrm{X})}{1 + \log_n m} = \frac{\mathrm{H}(\mathrm{Y})}{1 + \log_n m} + \frac{\mathrm{H}(\mathrm{X}\,|\,\mathrm{Y})}{1 + \log_m n}$$

ce qui est bien moins sympathique ! Notons que si $m = n$, il subsiste un facteur d'échelle $\frac{1}{2}$ (car il y a n^2 couples).

V.2.4 Information mutuelle

Ce concept mesure l'information portée par la transmission, par les couples de valeurs ; il n'a pas une définition immédiate. Elle se déduit plutôt des définitions des différentes entropies présentées ci-dessus.

De manière intuitive, dans le modèle du canal bruité, nous disposons sur les symboles émis de $(n+1)$ lois de probabilité pour la variable X :

- la distribution *a priori* : $\{\mathbb{P}[x_i]\}$ (ou distribution marginale) ;
- les n distributions *a posteriori* ; $\{\mathbb{P}[x_i \,|\, y_j]\}$ (ou distributions conditionnelles) ; chacune tient compte du symbole y_j effectivement reçu.

La différence d'information apportée par la connaissance du symbole effectivement émis (en plus du symbole reçu) est le *gain ou la perte d'information* entre la distribution *a posteriori* et la distribution *a priori*. Dit autrement, c'est *le gain ou la perte d'information* liée au bruit introduit par le canal, à la détérioration du message qu'il produit, au fait de recevoir potentiellement différents y_j pour un même x_i émis à cause du bruit. Ce concept est symétrique : l'information mutuelle est également la différence d'information sur la réception selon qu'on tient compte ou non du symbole émis.

Nous avons de manière directe $\mathcal{I}(x_i) = -\log_m \mathbb{P}[x_i]$ et $\mathcal{I}(x_i \,|\, y_j) = -\log_m \mathbb{P}[x_i \,|\, y_j]$.

Définition (Différence d'information mutuelle d'un couple)**.** Par définition, la différence d'information mutuelle du symbole émis x_i et du symbole reçu y_j *via* un canal bruité est le gain d'information

$$\mathcal{I}_b(x_i, y_j) = \mathcal{I}_b(x_i) - \mathcal{I}_b(x_i \,|\, y_j) = \mathcal{I}_b(y_j) - \mathcal{I}_b(y_j \,|\, x_i) \tag{V.8}$$

Ici aussi, la définition suppose que toutes les quantités d'information s'expriment dans la même unité, dans la base commune b. En toute rigueur, $b = m \cdot n$ pour assurer la normalisation.

Une simple substitution donne l'expression directe

$$\mathcal{I}_b(x_i, y_j) = -\log_b \frac{\mathbb{P}[x_i]}{\mathbb{P}[x_i \,|\, y_j]} = -\log_b \frac{\mathbb{P}[y_j]}{\mathbb{P}[y_j \,|\, x_i]} = -\log_b \frac{\mathbb{P}[x_i]\mathbb{P}[y_j]}{\mathbb{P}[x_i, y_j]}$$

Cette dernière forme montre clairement qu'il s'agit d'une valeur symétrique entre x_i et y_j. On constate également que $\mathcal{I}(x_i, y_j)$ ne vaut pas $-\log_{mn} \mathbb{P}[x_i, y_j]$, contrairement à ce que l'on aurait pu imaginer [2].

Remarque 1. Il faut faire très attention de ne pas se laisser abuser par le vocabulaire : la différence d'information mutuelle *d'un couple* n'est pas une quantité d'information au sens usuel, c'est *une différence* de telles quantités. Par conséquent, **cette valeur peut être négative dans certaines circonstances**, comme nous l'illustrerons ci-dessous. Par contre, l'information mutuelle *transmise par canal* que nous présentons ci-dessous est bien elle une quantité d'information.

Ceci conduit à la définition suivante.

Définition (Information mutuelle d'un canal)**.** Par définition, la quantité d'information mutuelle entre l'émission et la réception d'un canal bruité est la moyenne des quantités d'informations des couples de valeurs émises et reçues :

$$\mathcal{I}_b(\mathrm{X}, \mathrm{Y}) = \sum_{i,j} \mathbb{P}[x_i, y_j]\, \mathcal{I}_b(x_i, y_j) = -\sum_{i,j} \mathbb{P}[x_i, y_j] \log_b \frac{\mathbb{P}[x_i]\mathbb{P}[y_j]}{\mathbb{P}[x_i, y_j]} \tag{V.9}$$

Propriété V.2.2. *L'information mutuelle est une quantité d'information.*

$$0 \leq \mathcal{I}_b(\mathrm{X}, \mathrm{Y})$$

Démonstration. Reprenons sa définition par (V.9) ; nous constatons que les expressions au numérateur satisfont $\sum_{i,j} \mathbb{P}[x_i]\mathbb{P}[y_j] = 1$, donc par l'inégalité de Gibbs (théorème III.6) appliquée à la distribution croisée $\mathbb{P}[x_i, y_j]$, on a $\sum_{i,j} \mathbb{P}[x_i, y_j] \log_b \frac{\mathbb{P}[x_i]\mathbb{P}[y_j]}{\mathbb{P}[x_i, y_j]} \leq 0$. ∎

2. Ce serait la quantité d'information $\mathcal{I}(\mathbb{P}[x_i, y_j])$ portée par le couple *per se*, isolé, sans tenir compte du canal de transmission, ce qui est une autre notion. C'est pourquoi on parle ici d'information « *mutuelle* » du couple, avec une notation différente ($\mathcal{I}(x_i, y_j)$).

Il existe quelques propriétés intéressantes liant entropie et information mutuelle :

Propriété V.2.3.

$$\begin{aligned}\mathcal{I}_b(\mathrm{X},\mathrm{Y}) &= \mathrm{H}_b(\mathrm{X}) + \mathrm{H}_b(\mathrm{Y}) - \mathrm{H}_b(\mathrm{X},\mathrm{Y})\\ &= \mathrm{H}_b(\mathrm{X}) - \mathrm{H}_b(\mathrm{X}\,|\,\mathrm{Y})\\ &= \mathrm{H}_b(\mathrm{Y}) - \mathrm{H}_b(\mathrm{Y}\,|\,\mathrm{X})\\ &= \mathrm{H}_b(\mathrm{X},\mathrm{Y}) - \mathrm{H}_b(\mathrm{X}\,|\,\mathrm{Y}) - \mathrm{H}_b(\mathrm{Y}\,|\,\mathrm{X})\\ &\leq \log_b \min(m,n)\end{aligned}$$

Démonstration. La première égalité découle directement de (V.9) par substitution :

$$\begin{aligned}\mathcal{I}_b(\mathrm{X},\mathrm{Y}) = &-\sum_i \left(\sum_j \mathbb{P}[x_i,y_j]\right)\log_b \mathbb{P}[x_i]\\ &-\sum_j \left(\sum_i \mathbb{P}[x_i,y_j]\right)\log_b \mathbb{P}[y_j]\\ &+\sum_{i,j}\mathbb{P}[x_i,y_j]\log_b \mathbb{P}[x_i,y_j]\\ = &-\sum_i \mathbb{P}[x_i]\log_b \mathbb{P}[x_i] - \sum_j \mathbb{P}[y_j]\log_b \mathbb{P}[y_j] - \mathrm{H}_b(\mathrm{X},\mathrm{Y})\\ = &\ \mathrm{H}_b(\mathrm{X}) + \mathrm{H}_b(\mathrm{Y}) - \mathrm{H}_b(\mathrm{X},\mathrm{Y})\end{aligned}$$

Les trois égalités suivantes s'obtiennent en utilisant la propriété V.2.1 sur les précédentes. La dernière inégalité en découle *via* le corollaire III.1.5. ∎

En corollaire, on trouve immédiatement $\mathrm{H}_b(\mathrm{X},\mathrm{Y}) = \mathrm{H}_b(\mathrm{X}) + \mathrm{H}_b(\mathrm{Y}) - \mathcal{I}_b(\mathrm{X},\mathrm{Y})$, ce qui dit que l'entropie croisée, l'incertitude sur les couples de symboles émis et reçus, est inférieure à la somme des incertitudes à l'émission et à la réception prises isolément ; donc la transmission apporte une petite quantité d'information — précisément l'information mutuelle — qui réduit l'incertitude [3]. Par contre, si le canal n'était pas bruité, on aurait $\mathrm{H}_b(\mathrm{X},\mathrm{Y}) = \mathrm{H}_b(\mathrm{X}) = \mathrm{H}_b(\mathrm{Y}) = \mathcal{I}_b(\mathrm{X},\mathrm{Y})$.

Corollaire V.2.4.

$$\mathrm{H}_b(\mathrm{X},\mathrm{Y}) \leq \mathrm{H}_b(\mathrm{X}) + \mathrm{H}_b(\mathrm{Y}) \quad \textit{et} \quad \mathrm{H}(\mathrm{X}\,|\,\mathrm{Y}) \leq \mathrm{H}(\mathrm{X}) \quad \textit{et} \quad \mathrm{H}(\mathrm{Y}\,|\,\mathrm{X}) \leq \mathrm{H}(\mathrm{Y})$$

Démonstration. Ces inégalités découlent directement des propriétés V.2.2 et V.2.3. ∎

Réécrivons la définition de l'information mutuelle ; nous avons :

$$\begin{aligned}\mathcal{I}_b(\mathrm{X},\mathrm{Y}) &= -\sum_{i,j}\mathbb{P}[x_i,y_j]\log_b\frac{\mathbb{P}[x_i]\mathbb{P}[y_j]}{\mathbb{P}[x_i,y_j]}\\ &= -\sum_{i,j}\mathbb{P}[x_i]\mathbb{P}[y_j\,|\,x_i]\log_b\frac{\mathbb{P}[y_j]}{\mathbb{P}[y_j\,|\,x_i]}\\ &= -\sum_i \mathbb{P}[x_i]\left(\sum_j \mathbb{P}[y_j\,|\,x_i]\log_b\frac{\mathbb{P}[y_j]}{\mathbb{P}[y_j\,|\,x_i]}\right)\end{aligned}$$

Cette dernière expression présente l'information mutuelle comme une moyenne de valeurs appelées *quantités d'information mutuelle conditionnelle*.

Définition (Information mutuelle conditionnelle)**.** La quantité d'information mutuelle conditionnelle portée par un symbole x_i *via* un canal bruité est :

$$\mathcal{I}_b(\mathrm{Y}\,|\,x_i) = -\sum_j \mathbb{P}[y_j\,|\,x_i]\log_b\frac{\mathbb{P}[y_j]}{\mathbb{P}[y_j\,|\,x_i]} = -\sum_j \mathbb{P}[y_j\,|\,x_i]\log_b\mathbb{P}[y_j] - \mathrm{H}_b(\mathrm{Y}\,|\,x_i) \tag{V.10}$$

3. Ce qui est heureux, sinon pourquoi transmettre ?

Propriété V.2.5. *L'information mutuelle conditionnelle est une quantité d'information.*

$$\forall\, i \,:\, 0 \leq \mathcal{I}_b(\mathrm{Y}\,|\,x_i)$$

Démonstration. Nous constatons que les expressions au numérateur satisfont $\sum_j \mathbb{P}[y_j] = 1$, donc par l'inégalité de Gibbs (théorème III.6) appliquée à la distribution conditionnelle $\mathbb{P}[y_j\,|\,x_i]$, on a

$$\sum_j \mathbb{P}[y_j\,|\,x_i] \log_b \frac{\mathbb{P}[y_j]}{\mathbb{P}[y_j\,|\,x_i]} \leq 0$$

∎

$\mathcal{I}_b(\mathrm{Y}\,|\,x_i)$ est l'information transportée par le symbole x_i au travers du canal. L'information mutuelle est ainsi la moyenne, pour tout symbole émis, des informations mutuelles conditionnelles.

$$\mathcal{I}_b(\mathrm{X},\mathrm{Y}) = \sum_i \mathbb{P}[x_i] \cdot \mathcal{I}_b(\mathrm{Y}\,|\,x_i) \tag{V.11}$$

V.2.5 Un exemple

Considérons une source X de 4 symboles ($m = 4$) de loi de probabilité suivante.

$$\mathbb{P}[x_i] = \begin{bmatrix} 1/4 & 1/8 & 3/8 & 1/4 \end{bmatrix}$$

Et un canal transmettant 5 symboles Y ($n = 5$) avec les probabilités de transmission suivantes.

$$\mathbb{P}[y_j\,|\,x_i] = \begin{bmatrix} 3/4 & 0 & 1/4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 1/8 & 1/8 & 0 & 0 & 3/4 \end{bmatrix}$$

Choisissons d'effectuer tous les calculs de quantité d'information et d'entropie en binaire ($b = 2$), afin de pouvoir utiliser une unité commune (le bit).

Nous calculons immédiatement :

$$\mathrm{H}_2(\mathrm{X}) = \sum_i \mathbb{P}[x_i] \log_2 \mathbb{P}[x_i] = \frac{5}{2} - \frac{3}{8}\log 3 \approx 1.90564$$

$$\mathbb{P}[x_i, y_j] = \mathbb{P}[x_i]\mathbb{P}[y_j\,|\,x_i] = \begin{bmatrix} 3/16 & 0 & 1/16 & 0 & 0 \\ 0 & 1/8 & 0 & 0 & 0 \\ 0 & 0 & 3/16 & 3/16 & 0 \\ 1/32 & 1/32 & 0 & 0 & 3/16 \end{bmatrix}$$

$$\mathbb{P}[y_j] = \sum_i \mathbb{P}[x_i]\mathbb{P}[y_j\,|\,x_i] = \begin{bmatrix} 7/32 & 5/32 & 1/4 & 3/16 & 3/16 \end{bmatrix}$$

$$\mathrm{H}_2(\mathrm{Y}) = \sum_j \mathbb{P}[y_j] \log_2 \mathbb{P}[y_j] = \frac{31}{8} - \frac{3}{8}\log_2 3 - \frac{5}{32}\log_2 5 - \frac{7}{32}\log_2 7 \approx 2.30373$$

$$\mathbb{P}[x_i\,|\,y_j] = \mathbb{P}[x_i, y_j] / \mathbb{P}[y_j] = \begin{bmatrix} 6/7 & 0 & 0 & 1/7 \\ 0 & 4/5 & 0 & 1/5 \\ 1/4 & 0 & 3/4 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De là, nous calculons l'entropie croisée

$$\mathrm{H}_2(\mathrm{X},\mathrm{Y}) = -\sum_{ij} \mathbb{P}[x_i, y_j] \log_2 \mathbb{P}[x_i, y_j] = \frac{63}{16} - \frac{3}{4}\log_2 3 \approx 2.748778$$

Et les entropies conditionnelles

$$\mathrm{H}_2(\mathrm{Y}\,|\,\mathrm{X}) = -\sum_{ij} \mathbb{P}[x_i, y_j] \log_2 \mathbb{P}[y_j\,|\,x_i] = \frac{23}{16} - \frac{3}{4}\log_2 3 \approx 0.843139$$

$$\mathrm{H}_2(\mathrm{X}\,|\,\mathrm{Y}) = \mathrm{H}_2(\mathrm{X},\mathrm{Y}) - \mathrm{H}_2(\mathrm{Y}) = \frac{1}{16} - \frac{3}{8}\log_2 3 + \frac{5}{32}\log_2 5 + \frac{7}{32}\log_2 7 \approx 0.44505$$

Ce qui conduit à la quantité d'information mutuelle

$$\mathcal{I}_2(\mathrm{X},\mathrm{Y}) = \mathrm{H}_2(\mathrm{X}) - \mathrm{H}_2(\mathrm{X}\,|\,\mathrm{Y}) = \frac{39}{16} - \frac{5}{32}\log_2 5 - \frac{7}{32}\log_2 7 \approx 1.4606$$

soit une perte de près d'un bit et demi dû au bruit.

Si nous calculons les quantités d'information mutuelle conditionnelle, on trouve

$$\mathcal{I}_2(\mathrm{Y}\,|\,x_i) = \begin{cases} \frac{9}{4} + \frac{3}{4}\log_2 3 - \frac{3}{4}\log_2 7 & \approx 1.3332 \\ 5 - \log_2 5 & \approx 2.67807 \\ 2 - \frac{1}{2}\log_2 3 & \approx 1.20752 \\ 2 - \frac{1}{8}\log_2 5 - \frac{1}{8}\log_2 7 & \approx 1.35884 \end{cases}$$

Mais, comme signalé, les différences d'information des couples peuvent être négatives :

$$\mathcal{I}_2(x_i, y_j) = \begin{bmatrix} 3 + \log_2 3 - \log_2 7 & -\infty & 0 & -\infty & -\infty \\ -\infty & 5 - \log_2 5 & -\infty & -\infty & -\infty \\ -\infty & -\infty & 1 & 3 + \log_2 3 & -\infty \\ 2 - \log_2 7 & 2 - \log_2 5 & -\infty & -\infty & 2 \end{bmatrix}$$

$$\approx \begin{bmatrix} +1.778 & -\infty & 0 & -\infty & -\infty \\ -\infty & +2.678 & -\infty & -\infty & -\infty \\ -\infty & -\infty & +1 & +1.415 & -\infty \\ -0.807 & -0.322 & -\infty & -\infty & +2 \end{bmatrix}$$

Effectivement, recevoir y_1 ou y_2 lorsque x_4 a été émis est une perte d'information : il est plus probable lorsqu'on reçoit l'un de ces symboles que respectivement x_1 ou x_2 aient été émis (selon le principe du maximum de vraisemblance, comme on le verra plus loin).

V.3 Capacité d'un canal

Étant donné un canal de transmission et la perte d'information qu'il engendre, il est légitime de s'interroger sur les choix conduisant à minimiser cette perte. Le seul degré de liberté disponible est celui de la loi de probabilité des symboles émis. Un « bon » code à l'émission (qui influence les probabilités d'occurrence des caractères de l'alphabet dans lequel sont écrits les mots du code) sera de nature à minimiser les pertes, les erreurs de transmission. Le concept de *capacité du canal* décrit la perte minimale d'information correspondant à ce choix idéal.

Définition (Capacité d'un canal)**.** La capacité d'un canal est la borne supérieure de quantité d'information mutuelle entre l'émission et la réception, pour toute loi de probabilité d'émission.

$$\mathcal{C}_b = \max_{\mathbb{P}[\mathrm{X}]} \mathcal{I}_b(\mathrm{X},\mathrm{Y}) \tag{V.12}$$

Pour justifier le bien-fondé de cette définition, il nous faut montrer maintenant que cette borne supérieure existe. Comme l'ensemble des symboles est fini et que les probabilités sont des nombres réels, nous pouvons les faire varier continûment sous la contrainte $\sum_i \mathbb{P}[x_i] = 1$. Par (V.9), l'information mutuelle est une fonction dérivable en ces variables $\mathbb{P}[x_i]$ et elle est bornée (*cf.* la propriété V.2.3, par

exemple). C'est donc une fonction réelle, continue et bornée sur un fermé ($0 \leq \mathbb{P}[x_i] \leq 1$)et continue; sa borne supérieure est donc atteinte (c'est bien un maximum).

De la propriété V.2.3, on a immédiatement

$$0 \leq \mathcal{C}_b \leq \log_b \min(m, n)$$

De plus, de l'existence d'une loi de probabilité de la source maximisant l'information mutuelle, on déduit

Propriété V.3.1. *La capacité d'un canal est une propriété intrinsèque de celui-ci, qui ne dépend donc que des probabilités de transmission $\mathbb{P}[y_j \,|\, x_i]$.*

La capacité d'un canal est donc la *quantité d'information maximale qu'il peut transporter* de l'émetteur au récepteur (et réciproquement).

En toute généralité, les quantités d'information mutuelle conditionnelle $\mathcal{I}_b(\mathrm{Y} \,|\, x_i)$ intervenant dans l'expression de l'information mutuelle dépendent évidemment du symbole émis x_i, mais nous avons la propriété suivante.

Théorème V.3.2. *Lorsque l'information mutuelle $\mathcal{I}_b(\mathrm{X}, \mathrm{Y})$ est maximale (i.e. elle vaut la capacité $\mathcal{C}_b$ du canal), les informations mutuelles conditionnelles $\mathcal{I}_b(\mathrm{Y} \,|\, x_i)$ sont toutes égales à cette capacité, donc indépendantes de x_i.*

Démonstration. Nous devons maximiser $\mathcal{I}_b(\mathrm{X}, \mathrm{Y})$ vue comme une fonction des m variables $p_i = \mathbb{P}[x_i]$ liées par $\sum_i p_i = 1$. Nous notons également $p_{ij} = \mathbb{P}[y_j \,|\, x_i]$ les $m \cdot n$ valeurs *constantes* définissant le canal.

Dans la définition de $\mathcal{I}_b(\mathrm{X}, \mathrm{Y}) = -\sum_i p_i \sum_j p_{ij} \left(\log_b \mathbb{P}[y_j] - \log_b p_{ij}\right)$, ces variables p_i interviennent non seulement explicitement, mais également indirectement *via* les $\mathbb{P}[y_j] = \sum_i p_i p_{ij}$.

Observons que $\frac{\partial \mathbb{P}[y_j]}{\partial p_k} = p_{kj}$, tout simplement. Par conséquent,

$$\begin{aligned}
\frac{\partial \mathcal{I}_b(\mathrm{X}, \mathrm{Y})}{\partial p_k} &= -\sum_j p_{kj} \log_b \frac{\mathbb{P}[y_j]}{p_{kj}} - \frac{1}{\ln b} \sum_i p_i \sum_j p_{ij} \frac{p_{kj}}{\mathbb{P}[y_j]} \\
&= \mathcal{I}_b(\mathrm{Y} \,|\, x_k) - \frac{1}{\ln b} \sum_j \frac{p_{kj}}{\mathbb{P}[y_j]} \sum_i p_i p_{ij} \\
&= \mathcal{I}_b(\mathrm{Y} \,|\, x_k) - \frac{1}{\ln b} \sum_j p_{kj} \\
&= \mathcal{I}_b(\mathrm{Y} \,|\, x_k) - \frac{1}{\ln b}
\end{aligned}$$

Pour minimiser $\mathcal{I}_b(\mathrm{X}, \mathrm{Y})$ sous contrainte, nous appliquons la méthode des multiplicateurs de Lagrange, donc nous maximisons $\mathrm{F}(p_1, p_2 \ldots p_m, \lambda) = \mathcal{I}_b(\mathrm{X}, \mathrm{Y}) + \lambda \left(1 - \sum_i p_i\right)$, donc nécessairement:

$$\begin{cases} \frac{\partial \mathrm{F}}{\partial p_k} = 0 & = \mathcal{I}_b(\mathrm{Y} \,|\, x_k) - \frac{1}{\ln b} - \lambda \\ \frac{\partial \mathrm{F}}{\partial \lambda} = 0 & = 1 - \sum_i p_i \end{cases} \tag{V.13}$$

On en déduit donc que

$$\forall k : \mathcal{I}_b(\mathrm{Y} \,|\, x_k) = \frac{1}{\ln b} + \lambda$$

pour la valeur λ conduisant au maximum de $\mathcal{I}_b(\mathrm{X}, \mathrm{Y})$, ce qui est donc bien une valeur identique pour toutes les informations mutuelles conditionnelles.

Comme l'information mutuelle est une moyenne de ces informations mutuelles conditionnelles (*cf.* (V.11)) et qu'elles sont toutes égales, elles valent nécessairement toutes cette valeur moyenne, c'est-a-dire $\mathcal{C}_b$. ∎

Ainsi, chaque symbole émis « travaille », sert la transmission à la capacité du canal.

Exemple. Reprenons l'exemple du canal présenté ci-dessus.

Rechercher sa capacité est un problème d'optimisation non linéaire que l'on résout plus simplement par le système (V.13) de $m+1$ équations. Dans ce cas-ci, on trouve numériquement

$$\mathbb{P}[x_i] \approx [0.22067, 0.31148, 0.2838, 0.184]$$

ce qui correspond à la capacité $\mathcal{C}_2 \approx 1.58$ légèrement supérieure à l'information à mutuelle de notre exemple (≈ 1.46).

V.4 Canal symétrique

V.4.1 Canal binaire symétrique

Un cas particulier important, déjà étudié par Claude Shannon, est celui du canal binaire symétrique (*binary symmetric channel* ou **BSC**) pour lequel il n'y a que deux symboles tant à l'émission qu'à la réception ($m = n = 2$) et les probabilités conditionnelles de transfert ne dépendent que d'un seul paramètre p d'erreur de transmission, c'est-à-dire que le bit reçu est l'inverse de celui émis (*bit flipping*).

Les probabilités conditionnelles définissant un BSC sont donc une matrice 2×2 symétrique :

$$\mathbb{P}[y_j \,|\, x_i] = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$

Si la source émet un bit « 0 » avec la probabilité $\mathbb{P}[\mathrm{X}=0] = p_0$ (donc $\mathbb{P}[\mathrm{X}=1] = 1-p_0$), on trouve

$$\mathbb{P}[x_i, y_j] = \begin{bmatrix} p_0\,(1-p) & p_0\,p \\ (1-p_0)\,p & (1-p_0)(1-p) \end{bmatrix}$$

L'entropie de la source est évidemment donnée par la fonction d'entropie (voir (III.4)) :

$$\mathrm{H}(\mathrm{X}) = \mathcal{H}(p_0)$$

On trouve immédiatement

$$\begin{aligned} \mathrm{H}(\mathrm{Y}\,|\,\mathrm{X}) &= -p_0(1-p)\log(1-p) - p_0 p \log p - (1-p_0)p\log p - (1-p_0)(1-p)\log(1-p) \\ &= \mathcal{H}(p) \end{aligned}$$

qui ne dépend pas de la loi de probabilité de la source et

$$\mathrm{H}_2(\mathrm{X},\mathrm{Y}) = \mathrm{H}_2(\mathrm{X}) + \mathrm{H}_2(\mathrm{Y}\,|\,\mathrm{X}) = \mathcal{H}(p_0) + \mathcal{H}(p)$$

Ainsi, $\mathcal{I}_2(\mathrm{X},\mathrm{Y}) = \mathrm{H}(\mathrm{Y}) - \mathrm{H}(\mathrm{Y}\,|\,\mathrm{X}) = \mathrm{H}(\mathrm{Y}) - \mathcal{H}(p)$.

Pour calculer H(Y), posons $q_0 = \mathbb{P}[\mathrm{Y}=0] = p_0(1-p) + (1-p_0)p$; donc $\mathbb{P}[\mathrm{Y}=1] = 1-q_0$. Ceci donne $\mathrm{H}(\mathrm{Y}) = -q_0\log q_0 - (1-q_0)\log(1-q_0) = \mathcal{H}(q_0)$, donc $\mathcal{I}_2(\mathrm{X},\mathrm{Y}) = \mathcal{H}(q_0) - \mathcal{H}(p)$.

Dans cette dernière expression, seul $\mathcal{H}(q_0)$ dépend de p_0 (indirectement). Pour trouver la capacité $\mathcal{C}_2 = \max_{p_0} \mathcal{I}_2(\mathrm{X},\mathrm{Y})$, nous devons donc maximiser $\mathcal{H}(q_0)$. Connaissant la fonction d'entropie (III.4), ce maximum vaut 1 (et il est atteint en $q_0 = \frac{1}{2}$). Par conséquent,

$$\mathcal{C}_2 = 1 - \mathcal{H}(p) \tag{V.14}$$

qui ne dépend donc, comme prévu, que de p, pas de p_0.

Pour maximiser la capacité du canal, il faut donc que $\mathcal{H}(p)$ soit nul, donc $p = 0$ (il n'y a pas d'erreurs de transmission) ou $p = 1$ (tous les bits sont systématiquement inversés).

Vu la symétrie de la fonction d'entropie $\mathcal{H}(p)$ autour de $\frac{1}{2}$, dans l'étude du canal binaire symétrique, on peut supposer que $p < \frac{1}{2}$, l'égalité étant le cas particulier décrit ci-dessous.

Canal totalement bruité

Un canal binaire symétrique est totalement bruité, si $p = \frac{1}{2}$, puisque les bits ont autant de chance d'être perdus que d'être préservés. Il est absolument non fiable.

On trouve alors $\mathbb{P}[Y = 0] = p_0 + \frac{1}{2} - p_0 = \frac{1}{2}$, ce qui ne dépend pas de p_0, donc de ce qui a été envoyé ! Y est ici une variable aléatoire indépendante de X.

Bien sûr $H(Y) = 1 = H(Y|X)$, $\mathcal{I}(X,Y) = 0$ et $\mathcal{C} = 0$, ce qui traduit de différentes façons l'absence de toute information transmise.

En cryptographie, ce modèle est celui du chiffrement par *masque jetable* (*One-Time Pad* ou OTP), aussi appelé *schéma de Vernam*[4] qui, comme l'a démontré Claude Shannon en 1948, est le seul et unique système de chiffrement totalement et parfaitement incassable.

V.4.2 Canal symétrique général

Le cas binaire vu ci-dessus peut se généraliser au cas d'un alphabet C à r symboles ($|C| = r$) de taille quelconque. Un canal symétrique est un canal où $X = Y = C$ et dans lequel chaque symbole a la même probabilité p d'être transformé en un autre *de manière uniforme*.

$$\mathbb{P}[y_j|x_i] = \begin{cases} 1-p & \text{si} \quad i = j \\ \frac{p}{r-1} & \text{sinon} \end{cases}$$

Si $\mathbb{P}[x_i] = p_i$, $\mathbb{P}[y_j] = (1-p)p_j + \frac{p}{r-1}(1-p_j)$. On a également

$$H_r(Y|x_i) = -\sum_j \mathbb{P}[y_j|x_i] \log_r \mathbb{P}[y_j|x_i] = -(1-p)\log_r(1-p) - (r-1)\frac{p}{r-1}\log_r \frac{p}{r-1} = \mathcal{H}_r(p)$$

fonction d'entropie généralisée (voir (III.5)). Celle-ci ne dépend pas du x_i émis, donc

$$H_r(Y|X) = \mathcal{H}_r(p)$$

Comme $\mathcal{I}_r(X,Y) = H_r(Y) - H_r(Y|X) = H_r(Y) - \mathcal{H}_r(p)$, la capacité du canal s'obtient en maximisant $H_r(Y)$ (ce qui s'obtient d'ailleurs en maximisant également $H_r(X)$, vu la symétrie de la matrice $\mathbb{P}[y_j|x_i]$), c'est-à-dire avec une source de loi uniforme ($\mathbb{P}[x_i] = \frac{1}{r}$). Donc :

$$\mathcal{C}_r = 1 - \mathcal{H}_r(p) \tag{V.15}$$

avec $0 \leq \mathcal{C}_r \leq 1$.

Le canal symétrique est totalement bruité quand $\mathcal{C}_r = 0$, ce qui correspond à $p = \frac{r-1}{r}$ (tous les $\mathbb{P}[y_j|x_i] = \frac{1}{r}$).

4. Du nom de celui qui le redécouvrit 1917 et le fit breveter, Gilbert Sandford Vernam. En fait, il avait été déjà utilisé dès 1882 par Frank Miller pour des transmissions télégraphiques confidentielles.

Chapitre VI

Codes correcteurs d'erreur

La théorie moderne des codes correcteurs d'erreurs a été véritablement fixée par les travaux de Richard Wesley Hamming en 1950 (voir [18]).

VI.1 Définitions et propriétés fondamentales

Nous nous plaçons dans le contexte où les informations sont transmises *via* un canal bruité. Par conséquent, il est possible que le symbole reçu ne permette pas de retrouver le symbole émis avec certitude. Un canal parfait serait, en effet, un canal dont les probabilités de transfert sont déterministes :

$$\forall j \,:\, \mathbb{P}[x_i \,|\, y_j] \in \{0,1\}$$

ce qui veut dire qu'un seul x_i correspond à un y_j donné, ce qui n'empêche pas que plusieurs y_j puissent correspondre à un même x_i. Cette situation modélise la situation où ces différents symboles reçus pour un même symbole émis sont le résultat d'erreurs de transmission, mais qu'il existe une fonction de décodage $x_i = f(y_j)$, dite fonction de *correction d'erreurs* qui permet de retrouver le message émis [1] avec certitude, malgré les erreurs de transmission.

C'est bien sûr un modèle idéal. De manière plus réaliste, nous pouvons plutôt espérer que la probabilité d'interpréter erronément un symbole reçu sera la plus faible possible.

VI.1.1 Code à répétition et fiabilité

Supposons que nos symboles originels, nos messages sont des chaînes de caractères d'un alphabet C de taille r. Une idée simple pour accroître les chances de pouvoir décoder un message transmis à travers un canal bruité consiste à envoyer plusieurs fois chaque caractère. Ainsi, les x_i transmis sont des blocs de n caractères identiques. C'est ce qui s'appelle un *code à répétition*.

Définition (Code à répétition)**.** Un code à répétition de longueur n sur un alphabet C de r caractères est un code en bloc constitué des r mots du code

$$\mathrm{K} : \mathrm{C} \to \mathrm{C}^n \,:\, c \mapsto \overbrace{cc\ldots c}^{n}$$

Plaçons-nous dans le cas d'un canal symétrique avec une probabilité p d'altérer chaque caractère transmis. Sous l'hypothèse markovienne, la probabilité que le message codé par un code à répétition de n caractères arrive intact est $(1-p)^n$, ce qui peut être très faible même pour de petites valeurs de p.

Mais nous pouvons appliquer une meilleure stratégie : celle qui consiste à décider que le caractère émis était celui ayant la valeur la plus fréquente dans le mot reçu. C'est ce qu'on appelle un *décodage par majorité* : on modélise alors le canal comme opérant sur des blocs de n caractères avec une fonction de correction d'erreur par majorité. Celle-ci permet alors de décoder correctement un message reçu avec une probabilité élevée en vertu du résultat suivant.

1. Dans un modèle de canal parfait, cette fonction est définie pour tout y_j, sinon $\mathbb{P}[y_j]$ serait nulle.

Propriété VI.1.1. *Soit $\mathbb{P}_{err}$ la probabilité d'erreur de décodage par majorité du code à répétition sur n caractères au travers d'un canal symétrique. Si $p < \frac{1}{2}$, elle est bornée par*

$$\mathbb{P}_{err} \leq \frac{p(1-p)}{n(\frac{1}{2}-p)^2} \quad \textit{donc} \quad p < \frac{1}{2} \Rightarrow \lim_{n\to\infty} \mathbb{P}_{err} = 0$$

et si $p = \frac{1}{2}$, $\mathbb{P}_{err} = \frac{1}{2}$. Dans le cas binaire, si $p > \frac{1}{2}$, on a la même propriété.

Démonstration. Dans ce contexte, regroupons tout caractère mal transmis sous une seule catégorie « caractère erroné ». La loi de probabilité de transmission des mots du code pour un caractère donné (*i.e.* une ligne de $\mathbb{P}[y_j \,|\, x_i]$ ainsi « binarisé ») est une distribution binomiale. Si $p < \frac{1}{2}$, pour qu'un mot reçu ne puisse être correctement décodé, il faut qu'au moins $\frac{n}{2}$ caractères aient été altérés, c'est-à-dire :

$$\mathbb{P}_{\text{err}} = \sum_{k=\left\lceil \frac{n}{2} \right\rceil}^{n} \binom{n}{k} p^k (1-p)^{n-k}$$

ce qui est le complément de la fonction de répartition de la distribution binomiale $\mathrm{Binom}(n,p)$.

Dans le cas binaire, si $p > \frac{1}{2}$, on inverse le décodage, donc les rôles de p et $(1-p)$ s'échangent.

Si $p = \frac{1}{2}$, chaque caractère reçu est indépendant du caractère émis et, vu la symétrie de la distribution, sa médiane vaut sa moyenne et $\mathbb{P}_{\text{err}} = \frac{1}{2}\ \forall\, n$.

Pour prouver une borne supérieure si $p < \frac{1}{2}$, nous allons appliquer l'inégalité de Bienaymé-Tchebychev[2] : pour toute variable aléatoire X de variance finie[3] ($\mathrm{Var}[X] < \infty$),

$$\forall\, \alpha > 0 \,:\, \mathbb{P}\left[\left|X - \mathbb{E}[X]\right| \geq \alpha\right] \leq \frac{\mathrm{Var}[X]}{\alpha^2}$$

Dans le cas de la distribution binomiale $X = \mathrm{Binom}(n,p)$, on a $\mathbb{E}[X] = np$ et $\mathrm{Var}[X] = np(1-p)$. Ainsi

$$\mathbb{P}\left[X \geq \left\lceil \frac{n}{2} \right\rceil\right] = \mathbb{P}\left[X - np \geq \left\lceil \frac{n}{2} \right\rceil - np\right] \leq \mathbb{P}\left[\left|X - np\right| \geq \frac{n}{2} - np\right] \leq \frac{np(1-p)}{\left(\frac{n}{2} - np\right)^2} = \frac{p(1-p)}{\left(\frac{1}{2}-p\right)^2} \cdot \frac{1}{n}$$

ce qui démontre la propriété. ∎

On a immédiatement le corollaire suivant.

Corollaire VI.1.2. *Pour un code à répétition sur un canal binaire symétrique*

$$p \neq \frac{1}{2} \Rightarrow \lim_{n\to\infty} \mathbb{P}_{err} = 0$$

Définition (Fiabilité)**.** On appelle fiabilité d'un canal la valeur $(1-\mathbb{P}_{\text{err}})$ représentant la probabilité qu'un message puisse être décodé correctement s'il a été transmis via un canal bruité.

Par conséquent, la fiabilité d'un canal symétrique peut être aussi proche de l'unité, aussi bonne qu'on le souhaite en augmentant la longueur des mots du code à répétition. Évidemment, l'efficacité d'un tel canal de communication en devient exécrable : en un temps fini, on peut transmettre parfaitement... une information nulle !

2. Des noms des mathématiciens Irénée-Jules Bienaymé, qui la formula en 1853, et de Пафнутий Львович Чебышёв (Pafnouti Lvovitch Tchebychev), qui la démontra en 1867.
3. Ce qui implique que l'espérance est également finie ($-\infty < \mathbb{E}[X] < \infty$).

VI.1.2 Codes correcteurs ou détecteurs d'erreurs

Afin d'améliorer cette situation, d'autres choix que le code à répétition s'imposent. Nous allons donc considérer des codes par bloc de longueur n fixée et tenter d'optimiser la sélection des mots du code pour minimiser $\mathbb{P}_{\text{err}}$ *via* un canal bruité.

Le plus souvent, il s'agira d'un canal binaire symétrique, mais on considérera également le cas d'un canal symétrique général, celui opérant sur un alphabet de r caractères ($|\mathrm{C}| = r$).

Définition (Code correcteur)**.** Un code correcteur de t erreurs (*error-correcting code* ou ECC) est un code en bloc de longueur n ($\mathrm{K} \subset \mathrm{C}^n$) pour lequel, si on émet un mot de ce code selon une loi uniforme au travers d'un canal bruité symétrique, le mot reçu est toujours correctement décodé si au plus t caractères ont été altérés. Le paramètre t s'appelle la *capacité de correction* du code.

Ainsi, certains mots qui auraient subi plus de t erreurs peuvent[4] ne plus être identifiés ou bien risquent d'être décodés de manière incorrecte (on associerait un autre mot du code à celui reçu).

La fonction permettant de retrouver le message originel le plus probable est appelée *fonction de correction*, selon le principe de *maximum de vraisemblance* :

$$\mathrm{Cor}: \mathrm{C}^n \to \mathrm{K} \cup \{\perp\} : y \mapsto \begin{cases} \hat{x} : \mathbb{P}[\hat{x}\,|\,y] = \max_{x \in \mathrm{K}} \mathbb{P}[x\,|\,y] & \text{s'il est unique} \\ \perp & \text{sinon} \end{cases}$$

Si plusieurs x correspondent à une même probabilité $\mathbb{P}[x\,|\,y]$ maximale, la correction est ambiguë ; elle ne permet donc pas de *corriger* les erreurs, mais elle permet de *détecter* avec certitude que des erreurs de transmission se sont produites. Tout code pour lequel $\mathrm{K} \subsetneq \mathrm{C}^n$ est donc un *code détecteur d'erreurs* pour tout $y \notin \mathrm{K}$ reçu (voir ci-dessous).

Propriété VI.1.3 (Maximum de vraisemblance)**.** *La fonction de correction par maximum de vraisemblance est celle qui minimise la probabilité $\mathbb{P}_{err}$ d'erreur de décodage.*

Démonstration. Soit un canal donné par ses probabilités conditionnelles $\mathbb{P}[y_j\,|\,x_i]$. Si la source obéit à la loi de probabilité $\mathrm{P}[x_i]$ et qu'on applique la fonction de décodage définie ci-dessus, la probabilité de décodage correct d'un y_j reçu vaut $\mathbb{P}[\hat{x}_j\,|\,y_j]$, par définition. En moyenne pour tout symbole émis, donc pour tout symbole reçu, on a $1-\mathbb{P}_{\text{err}} = \sum_j \mathbb{P}[y_j].\mathbb{P}[\hat{x}_j\,|\,y_j] = \sum_j \mathbb{P}[\hat{x}_j, y_j]$, soit une somme de n termes, un par colonne j de la matrice $\mathbb{P}[x_i, y_j]$. Or dans chaque colonne, le terme $\mathbb{P}[\hat{x}_j, y_j]$ est maximum, car $\mathbb{P}[\hat{x}_j, y_j] = \mathbb{P}[y_j].\mathbb{P}[\hat{x}_j\,|\,y_j]$ avec $\mathrm{P}[y_j]$ identique pour sa colonne. ∎

Corollaire VI.1.4. *Si tous les mots du code émis sont équiprobables (i.e. $\mathbb{P}[x_i] = \frac{1}{m}$ et $\mathrm{H}_m(\mathrm{X}) = 1$), la fonction de correction ne dépend que du canal et peut s'écrire directement*

$$\mathrm{Cor}(y_j) = \hat{x} \Rightarrow \mathbb{P}[y_j\,|\,\hat{x}] = \max_i \mathbb{P}[y_j\,|\,x_i] \tag{VI.1}$$

Démonstration. En effet, $\mathbb{P}[x_i\,|\,y_j] = \frac{\mathbb{P}[x_i].\mathbb{P}[y_j\,|\,x_i]}{\mathbb{P}[y_j]} = \frac{\mathbb{P}[y_j\,|\,x_i]}{m \cdot \mathbb{P}[y_j]}$ où le dénominateur est constant pour un y_j donné. Donc, pour maximiser $\mathbb{P}[x_i\,|\,y_j]$, on peut maximiser $\mathbb{P}[y_j\,|\,x_i]$. ∎

En pratique, nous verrons que l'on peut souvent appliquer la fonction définie par (VI.1), ce qui est plus simple, sans détériorer trop la fiabilité du code.

Définition (Code détecteur)**.** Un code détecteur de τ erreurs (*error-detecting code* ou EDC) est un code en bloc de longueur n ($\mathrm{K} \subset \mathrm{C}^n$) pour lequel, si on émet un mot de ce code selon une loi uniforme au travers d'un canal bruité symétrique, un mot reçu dont au plus τ caractères ont été altérés n'appartient jamais[5] au code ; il peut donc être détecté comme erroné. Le paramètre τ est la *capacité de détection* du code.

4. Il est possible toutefois que certains mots fortement détériorés puissent encore être décodés. Cela dépend du code, du mot particulier et des erreurs subies.

5. De nouveau, certains mots ayant subi plus d'erreurs peuvent ne pas appartenir au code. Cela dépend du code, du mot particulier et des erreurs subies.

La fonction de détection est

$$\mathrm{Det}: \mathsf{C}^n \to \{\bot, \mathsf{T}\} : y \mapsto \begin{cases} \mathsf{T} & \text{si} \quad y \in \mathsf{K} \\ \bot & \text{si} \quad y \notin \mathsf{K} \end{cases}$$

On a trivialement

Propriété VI.1.5. *Pour tout code en bloc, la capacité de détection τ est supérieure ou égale à la capacité de correction t :*

$$t \leq \tau$$

Exemple. Un code à répétition de longueur n permet toujours de corriger $\left\lfloor \frac{n-1}{2} \right\rfloor$ erreurs. Si n est pair, il permet toujours de détecter également $\frac{n}{2}$ erreurs, sans pouvoir les corriger dans ce cas limite.

Code par somme de contrôle

Un autre exemple de code détecteur est celui du code par *somme de contrôle* (ou *checksum*).

Définition (Code à somme de contrôle)**.** Un code à somme de contrôle de longueur n sur un alphabet C de r caractères transmet des messages de $(n-1)$ caractères $c_1 c_2 \ldots c_{n-1}$ complétés d'un n-ième caractère c_n garantissant la relation

$$\sum_{i=1}^{n} c_i \equiv 0 \pmod{r}$$

La fonction de correction d'un tel code permet de corriger 0 erreur, mais détecte avec certitude 1 erreur. Notons que cette erreur peut concerner n'importe quel caractère, un des caractères d'information ou le caractère de contrôle en appendice. Le code par *bit de parité* est le cas binaire du code par somme de contrôle avec $\bigoplus_i c_i = 0$.

Ces deux exemples montrent que le paramètre de correction ou de détection est la borne supérieure en deçà de laquelle *toutes* les erreurs sont détectées, respectivement corrigées, mais il est possible que dans certains cas, un message ayant subi plus d'erreurs de transmission puisse être détecté comme erroné, voire corrigé.

VI.1.3 Distance de Hamming

Cette distance a été définie pour la première fois par Richard Wesley Hamming qui étudiait les codes en bloc et plus particulièrement les codes correcteurs d'erreur (voir [18]).

Définition (Distance et poids de Hamming)**.** La distance de Hamming est définie sur $\mathsf{V}^n \times \mathsf{V}^n$, les paires de n-uplets d'éléments d'un ensemble V ($\mathsf{V}^n = \{\vec{x} = [x_1 x_2 \ldots x_n]\}$), par le nombre de positions où une composante diffère entre les deux n-uplets :

$$\mathrm{d_H}(\vec{x}, \vec{y}) = \left| \{ i \mid x_i \neq y_i \} \right|$$

Si l'ensemble V possède une valeur nulle, le poids de Hamming d'un n-uplet est sa distance au n-uplet nul, c'est-à-dire le nombre de ses composantes non nulles :

$$\mathrm{w_H}(\vec{x}) = \mathrm{d_H}(\vec{x}, \vec{0}) = \left| \{ i \mid x_i \neq 0 \} \right|$$

Ces définitions sont très générales ; elles s'appliquent directement aux mots de n caractères (on suppose bien sûr que $0 \in \mathsf{C}$). De plus, si C est un groupe additif (par exemple si $\mathsf{C} = \mathbb{Z}/r\mathbb{Z}$, l'ensemble des entiers modulo r ou si c'est un corps fini à r éléments), la distance de Hamming peut être définie à partir du poids :

$$\mathrm{d_H}(\vec{x}, \vec{y}) = \mathrm{w_H}(\vec{y} - \vec{x}) \tag{VI.2}$$

où la différence modulo r se calcule composante par composante. Dans le cas binaire :

$$\mathrm{d_H}(\vec{x}, \vec{y}) = \mathrm{w_H}(\vec{x} \oplus \vec{y})$$

Propriété VI.1.6.

$$\left|\mathrm{w_H}(\vec{x}) - \mathrm{w_H}(\vec{y})\right| \leq \mathrm{w_H}(\vec{x} - \vec{y}) \leq \mathrm{w_H}(\vec{x}) + \mathrm{w_H}(\vec{y})$$

Démonstration. La partie droite est immédiate : on ne peut avoir plus de composantes non nulles que ce qu'il y en a dans les deux vecteurs initiaux. Pour la partie droite, soit δ le nombre de composantes non nulles identiques (*i.e.* $x_i = y_i \neq 0$). On a $\mathrm{w_H}(\vec{x}-\vec{y}) = \big(\mathrm{w_H}(\vec{x}) - \delta\big)+\big(\mathrm{w_H}(\vec{y}) - \delta\big)$. Mais comme trivialement $\mathrm{w_H}(\vec{x}) \geq \delta$ et $\mathrm{w_H}(\vec{y}) \geq \delta$, la propriété est établie, car si $a \geq 0$ et $b \geq 0$, $a + b \geq |a - b|$. ∎

Définition (Distance mathématique)**.** Une distance, au sens mathématique, sur un ensemble E est une fonction réelle positive $\mathrm{E} \times \mathrm{E} \to \mathbb{R}^+$ satisfaisant les 3 axiomes suivants.

1. **Séparation** : $\forall a, b : \mathrm{d}(a, b) = 0 \Leftrightarrow a = b$.
2. **Symétrie** : $\forall a, b : \mathrm{d}(a, b) = \mathrm{d}(b, a)$.
3. **Inégalité triangulaire** : $\forall a, b, c : \mathrm{d}(a, c) \leq \mathrm{d}(a, b) + \mathrm{d}(b, c)$.

Un ensemble E muni d'une telle distance est appelé « espace métrique ».

Dans un tel espace, une *boule* (ou *disque* ou *sphère*) de rayon ϱ centré en c est l'ensemble des éléments de E à distance inférieure ou égale à ϱ :

$$\mathcal{B}(c, \varrho) = \{x \in \mathrm{E} \mid d(x, c) \leq \varrho\}$$

On précise parfois boule *fermée*, pour la distinguer de la boule *ouverte* pour laquelle la distance ϱ n'est pas atteinte (inégalité stricte).

Propriété VI.1.7. *La distance de Hamming est effectivement une distance mathématique (i.e. elle satisfait les 3 axiomes ci-dessus) dont les valeurs sont des nombres entiers positifs bornés par la longueur des mots :*

$$\mathrm{d_H}(\vec{x}, \vec{y}) \in \mathbb{N} \quad \textit{et} \quad 0 \leq \mathrm{d_H}(\vec{x}, \vec{y}) \leq n$$

La démonstration est immédiate.

Par contre, le poids de Hamming n'est pas une *norme* au sens mathématique, mais seulement une *pseudo-norme*. En effet, une pseudo-norme doit satisfaire les propriétés de séparation ($\mathrm{w_H}(\vec{x}) = 0 \Leftrightarrow \vec{x} = 0$) et de sous-additivité ($\mathrm{w_H}(\vec{x} + \vec{y}) \leq \mathrm{w_H}(\vec{x}) + \mathrm{w_H}(\vec{y})$), ce qui est bien le cas ici, mais si c'était une norme, on devrait avoir également l'homogénéité ($\mathrm{w_H}(\lambda\vec{x}) = |\lambda|\,\mathrm{w_H}(\vec{x})$), ce qui n'est évidemment pas vrai avec le poids de Hamming, sauf en binaire.

Propriété VI.1.8. *Dans l'ensemble* C^n *des mots de longueur n sur un alphabet* C *de r caractères, les boules de Hamming (i.e. au sens de la distance de Hamming) de rayon* $t \leq n$ *contiennent un nombre fini de mots, indépendant du centre de la boule et donné par la formule suivante.*

$$\mathrm{B}_t = \left|\mathcal{B}(c, t)\right| = \sum_{i=0}^{t} \binom{n}{i} (r-1)^i \tag{VI.3}$$

Démonstration. Les mots à distance i d'un mot donné diffèrent en exactement i composantes de ce mot. Nous avons $\binom{n}{i}$ choix pour les positions de ces composantes et, pour chacune, $(r - 1)$ valeurs possibles différentes de celle du mot donné. ∎

On peut sans problème étendre la définition de boule à tout rayon $\varrho \in \mathbb{R}^+$, même non entier. Dans ce cas, $\varrho \leq n \Rightarrow |\mathcal{B}(c, \varrho)| = \mathrm{B}_{\lfloor\varrho\rfloor}$.

Pour un n grand, on a la borne suivante sur ce nombre de mots. Elle nous servira par la suite.

Lemme VI.1.9. *Pour tout* $0 < \lambda < \frac{1}{2}$*, pour n suffisamment grand,* $\mathrm{B}_{n\lambda} \leq r^{n\mathcal{H}_r(\lambda)}$ *où* $\mathcal{H}_r$ *est la fonction d'entropie généralisée (III.5).*

Démonstration. Comme n est grand, nous pouvons utiliser la formule de James Stirling :

$$\sqrt{2\pi}\, n^n e^{-n}\sqrt{n} \leq n! \leq e\, n^n e^{-n}\sqrt{n}$$

Donc

$$\begin{aligned}
\binom{n}{n\lambda}(r-1)^{n\lambda} &\leq \frac{e\, n^n e^{-n}\sqrt{n}\,(r-1)^{n\lambda}}{\left[\sqrt{2\pi}\,(n\lambda)^{n\lambda}e^{-n\lambda}\sqrt{n\lambda}\right]\left[\sqrt{2\pi}\,(n-n\lambda)^{n-n\lambda}e^{-n+n\lambda}\sqrt{n-n\lambda}\right]} \\
&= \frac{n^n}{n^{n\lambda}n^{n-n\lambda}}\,\frac{(r-1)^{n\lambda}}{\lambda^{n\lambda}(1-\lambda)^{n(1-\lambda)}}\,\frac{e^n}{e^{-n\lambda}e^{-n(1-\lambda)}}\,\frac{e\sqrt{n}}{2\pi\sqrt{n\lambda}\sqrt{n-n\lambda}} \\
&= \left[\frac{(r-1)^{\lambda}}{\lambda^{\lambda}(1-\lambda)^{(1-\lambda)}}\right]^n \frac{e}{2\pi\sqrt{\lambda(1-\lambda)}\sqrt{n}} \\
&= \left[r^{-\lambda\log_r\lambda-(1-\lambda)\log_r(1-\lambda)+\lambda\log_r(r-1)}\right]\frac{e}{2\pi\sqrt{\lambda(1-\lambda)}}\frac{1}{\sqrt{n}} \\
&= r^{n\mathcal{H}_r(\lambda)}\,\frac{e}{2\pi\sqrt{\lambda(1-\lambda)}}\frac{1}{\sqrt{n}}
\end{aligned}$$

Considérons la séquence des $\binom{n}{i}(r-1)^i$. Passer du terme i à $(i-1)$ revient à le multiplier par $\frac{i}{n-i+1} / (r-1)$, croissants, donc majorés par le plus grand $\frac{n\lambda}{n-n\lambda+1} / (r-1) < \frac{\lambda}{1-\lambda} / (r-1) < 1 / (r-1)$, puisque $\lambda < \frac{1}{2}$. Donc

$$\sum_{i=0}^{n\lambda}\binom{n}{i}(r-1)^i < \binom{n}{n\lambda}(r-1)^{n\lambda}\sum_{i=0}^{\infty}\left(\frac{\lambda}{(r-1)(1-\lambda)}\right)^i = \binom{n}{n\lambda}(r-1)^{n\lambda}\frac{(r-1)(1-\lambda)}{(r-1)(1-\lambda)-\lambda}$$

En combinant les deux inéquations, on a $\mathrm{B}_{n\lambda} < \frac{\mathrm{M}_r(\lambda)}{\sqrt{n}}\, r^{n\mathcal{H}_r(\lambda)}$ où $\mathrm{M}_r(\lambda)$ est une constante par rapport à n (elle ne dépend que de r et λ), ce qui donne le résultat pour n suffisamment grand. ∎

VI.1.4 Contexte d'étude des codes correcteurs

Nous allons à présent étudier les codes correcteurs comme tels. Cela signifie que l'on se place dans le contexte particulier où les hypothèses suivantes sont satisfaites.

1. Un code (correcteur) est un **code en bloc** K de longueur n sur un alphabet C de r caractères ($\mathrm{K} \subset \mathrm{C}^n$).
2. Les caractères sont transmis *via* un **canal symétrique** de probabilité d'erreur $p < \frac{1}{2}$.
3. Les messages reçus appartiennent donc à l'espace C^n aussi.
4. Les messages émis sont tous **équiprobables** ($\forall\, x \in \mathrm{K} : \mathbb{P}[x] = \frac{1}{|\mathrm{K}|}$).

Sous ces hypothèses, il est possible d'exprimer la fonction de correction par maximum de vraisemblance à l'aide de la distance de Hamming.

Théorème VI.1.10 (Maximum de vraisemblance)**.** *Pour tout code en bloc* K *transmis via un canal symétrique de probabilité d'erreur $p < \frac{1}{2}$ et dont les mots émis sont équiprobables, la fonction de correction selon la méthode du maximum de vraisemblance est*

$$\mathrm{Cor} : y \mapsto \hat{x} : \mathrm{d_H}(\hat{x}, y) = \min_{x\in\mathrm{K}} \mathrm{d_H}(x, y)$$

Démonstration. Par définition, la correction par maximum de vraisemblance d'un message y reçu est le mot du code x maximisant $\mathbb{P}[x\,|\,y]$. Or, la probabilité pour qu'un mot x donné à distance k de y donné est qu'il ait subi exactement k erreurs correspondant aux caractères différents entre x et y. Pour un canal symétrique, elle vaut $p^k\,(1-p)^{n-k}$. Comme les mots émis sont équiprobables et que $p < \frac{1}{2}$, le maximum de cette expression est atteint pour le k minimum correspondant à un $x \in \mathrm{K}$. ∎

Dans ce contexte, les fonctions d'encodage et de décodage prennent une signification particulière :

Encodage La fonction d'encodage est une application injective de l'espace des symboles ou mots d'information dans celui des mots codés $\mathrm{Enc} : \mathsf{S} \to \mathsf{C}^n$, avec $\mathrm{Enc}(\mathsf{S}) = \mathsf{K}$, qui adjoint la redondance nécessaire afin de permettre de recouvrer l'information à partir d'un mot dont certains caractères ont été modifiés ($|\mathsf{S}| = |\mathsf{K}| < r^n$).

Décodage La fonction de décodage est une application surjective de l'espace de codage sur l'espace des symboles étendu d'une marque d'erreur $\mathrm{Dec} : \mathsf{C}^n \to \mathsf{S} \cup \{\perp\}$ qui applique la correction par méthode de maximum de vraisemblance et retire la redondance.

Fréquemment, on a $\mathsf{S} = \mathsf{C}^k$, avec $k < n$.

VI.2 Théorème fondamental de Shannon

VI.2.1 Débit d'un code

Permettre la détection ou la correction d'erreur impose que $\mathsf{K} \subsetneq \mathsf{C}^n$, puisque certains mots erronés doivent être détectés et, le cas échéant, être corrigés selon le principe de maximum de vraisemblance. Elle se fait donc au prix d'une certaine *redondance*, donc d'un surcoût de transmission, puisqu'on transmet plus de caractères que nécessaire pour représenter strictement l'information originelle.

Le *débit d'un code* (en anglais, *code rate*, parfois *information rate*, notamment selon Huffman) mesure la quantité minimale d'information transmise par symbole. C'est donc une quantification de ce surcoût, plus exactement l'inverse : une mesure de l'efficience, du rendement de la transmission.

Définition (Débit)**.** Le débit d'un code K sur un alphabet C de r caractères et de longueur maximale $\mathsf{L}_{\max}(\mathsf{K})$ est, par définition :

$$\mathsf{R}(\mathsf{K}) = \frac{\log_r |\mathsf{K}|}{\mathsf{L}_{\max}(\mathsf{K})}$$

Propriété VI.2.1. *Pour tout code* K*, on a*

$$0 \leq \mathsf{R}(\mathsf{K}) \leq 1 \tag{VI.4}$$

$\mathsf{R}(\mathsf{K}) = 0$ *est le code trivial (un seul message possible qui ne doit pas être transmis) et* $\mathsf{R}(\mathsf{K}) = 1$ *correspond à un code en bloc où tout message possible est un mot du code.*

Démonstration. De (I.2), on sait que $\mathsf{L}_{\max}(\mathsf{K}) \geq \log_r |\mathsf{K}|$.

On a évidemment $|\mathsf{K}| = 1 \Leftrightarrow \log_r |\mathsf{K}| = 0$.

$\mathsf{R}(\mathsf{K}) = 1 \Leftrightarrow \mathsf{L}_{\max}(\mathsf{K}) = \log_r |\mathsf{K}|$, ce qui correspond à un arbre de code complet. ∎

Cette définition générale peut s'appliquer à tout code. Par exemple, le code de Huffman binaire sur 6 symboles présenté plus haut a un débit de $\mathsf{R} = \frac{\log_2 6}{4} \approx 0.64624$, même si son taux de compression, défini de partir de la longueur moyenne, est de $\frac{\log_2 6}{77/32} \approx 1.07427 > 1$.

Mais dans les codes en bloc, ce qui est la situation où ce concept est le plus utilisé, le débit est *indépendant de la loi de probabilité de la source*. Si de plus, hypothèse fréquente en théorie des codes correcteurs, il s'agit d'un code en bloc représentant tous les messages de longueur k, *i.e.* avec une redondance de $(n - k)$ caractères exactement pour chaque mot, on a $|\mathsf{K}| = r^k$ et tout simplement

$$\mathsf{R}(\mathsf{K}) = \frac{k}{n} \tag{VI.5}$$

En particulier, $\mathsf{R}(\mathsf{K}) = 1$ correspond à la situation d'un code en bloc sans redondance, qui ne peut donc être utilisé que sur un canal sans bruit.

Exemple. Un code à répétition a un débit $\mathrm{R(K)} = \frac{1}{n}$; donc $\lim_{n\to\infty} \mathrm{R} = 0$.

Par contre, le débit d'un code à somme de contrôle est $\mathrm{R(K)} = \frac{n-1}{n} = 1 - \frac{1}{n}$. On trouve donc une propriété intéressante : $\lim_{n\to\infty} \mathrm{R} = 1$, mais sur un canal bruité, on a aussi

$$\lim_{n\to\infty} (1 - \mathbb{P}_{\mathrm{err}}) = \lim_{n\to\infty} (1-p)^n = 0$$

On peut donc avoir un débit quasi idéal (100 %), mais avec une fiabilité nulle, même si le taux d'erreur p est infinitésimal !

Note. Le débit d'un code est un nombre sans dimension, un rapport de longueurs de mots ; c'est une propriété d'un code, pas d'un canal de transmission. À ce titre, il ne faut pas le confondre avec des notions similaires, mais liées à la transmission physique du signal : celle de *bande passante* (*pass band*), ni celle de *largeur de bande* (*bandwidth*), toutes deux mesurées en **hertz**, ni encore celle de *débit binaire* (*bit rate*) qui s'exprime en **bits par seconde**, unité distincte du **baud**, nombre de symboles transmis par seconde en tenant compte du multiplexage éventuel.

VI.2.2 Second théorème de Shannon

L'exemple ci-dessus peut, heureusement, être considéré comme très mauvais. Plus exactement, dès ses travaux publiés en 1948, Shannon a énoncé une propriété fondamentale : pour tout canal **bruité** de capacité donnée, il existe une famille de codes « idéaux » dont l'erreur de transmission peut être aussi petite que l'on veut, tout en garantissant un débit proche de la capacité du canal. C'est ce qu'on appelle le second théorème de Shannon ou encore le « théorème fondamental de Shannon » (*Shannon's Fundamental Noisy Channel Coding Theorem*).

Théorème VI.2.2 (Claude Shannon, 1948)**.** *Pour tout canal bruité symétrique de probabilité d'erreur $p < \frac{1}{2}$ et de capacité $\mathcal{C}_r$, il existe une famille $\{\mathrm{K}_n\}$ de codes en bloc de longueur n, de probabilité d'erreur de décodage $\mathbb{P}_{err}(\mathrm{K}_n)$ et de débit $\mathrm{R}(\mathrm{K}_n) < \mathcal{C}_r$ telle que, simultanément,*

$$\lim_{n\to\infty} \mathbb{P}_{err}(\mathrm{K}_n) = 0 \quad \textit{et} \quad \lim_{n\to\infty} \mathrm{R}(\mathrm{K}_n) = \mathcal{C}_r$$

Il s'agit d'un *théorème d'existence* qui ne suggère aucune méthode constructive d'un code idéal qui satisferait cette propriété. Comme le dit Shannon lui-même : « *The demonstration of theorem 11, while not a pure existence proof, has some of the deficiencies of such proofs.* » ([32, p.414, 14. Discussion]). En fait, même aujourd'hui, on ne connaît pas d'exemple explicite d'une famille de codes satisfaisant ce critère !

En toute rigueur d'analyse, Shannon ne fournit pas une démonstration complète et générale, mais plutôt une esquisse, une solide argumentation dans le cas d'un canal symétrique binaire. La première démonstration générale rigoureuse est due à Amiel Feinstein en 1954 (*cf.* [9]) ; nous allons nous en inspirer (indirectement *via* [19]).

Démonstration. Tout d'abord, nous allons tenter de nous approcher suffisamment du débit visé : $\mathcal{C}_r - \delta < \mathrm{R} \leq \mathcal{C}_r$ pour tout $\delta > 0$ aussi petit que l'on veut. À partir d'une longueur n suffisamment grande ($n > \delta^{-1}$), pour chaque n, on choisit [6] un entier k_n tel que $\mathcal{C}_r - \delta < \frac{k_n}{n} \leq \mathcal{C}_r$, ce qui est possible puisque $\mathbb{Q}$ est dense dans $\mathbb{R}$. On peut alors fixer le nombre de mots du code à $|\mathrm{K}_n| = r^{k_n}$, donc $\mathrm{R}(\mathrm{K}_n) = \frac{k_n}{n}$ qui satisfait bien l'hypothèse.

À présent, il nous faut déterminer les mots de code $\mathrm{K}_n \subset \mathrm{C}^n$ afin de minimiser la probabilité d'erreur $\mathbb{P}_{\mathrm{err}}(\mathrm{K}_n)$. L'idée géniale de Shannon est que l'on peut choisir ce code totalement *au hasard* !

Définissions ainsi K_n comme une application *aléatoire* [7] $\mathrm{C}^{k_n} \to \mathrm{C}^n$ sélectionnée uniformément parmi toutes celles possibles. Il y en a $(r^n)^{r^{k_n}}$.

6. On applique donc implicitement l'axiome du choix.
7. On ne doit même pas supposer que le code soit univoque !

Dans ce contexte, l'espace de tous ces codes K_n suit une loi de probabilité uniforme et la probabilité d'erreur de décodage $\mathbb{P}_{\text{err}}(\mathsf{K}_n)$ sur cet espace est donc une *variable aléatoire*. On va montrer que

$$\forall\,\varepsilon>0,\exists\,\mathsf{N}:n>\mathsf{N}\Rightarrow\mathbb{E}[\mathbb{P}_{\text{err}}(\mathsf{K}_n)]<\varepsilon$$

C'est la partie difficile de la démonstration.

Supposons que nous émettions un mot de code $x\in\mathsf{K}_n$ quelconque au travers du canal symétrique. Chacun des n caractères de x a la même probabilité p indépendante d'être détérioré. En moyenne, il y aura donc np caractères erronés dans le message reçu y. Considérons dans C^n, l'espace des messages à décoder, la boule $\mathcal{B}(x,np)$ centrée sur x (le message reçu sans erreur) et de rayon np.

Choisissons un $\xi>0$, arbitrairement petit et tel que $p+\xi<\frac{1}{2}$ (c'est possible, puisque par hypothèse $p<\frac{1}{2}$) et gonflons la boule pour obtenir un rayon $n(p+\xi)$. Par la loi des grands nombres (puisque le canal symétrique est markovien), pour n suffisamment grand, presque tous les messages reçus y satisfont $n(p-\xi)<\mathsf{d}(x,y)<n(p+\xi)$, puisque np est la moyenne. Explicitement,

$$\forall\,\xi,\forall\,\varepsilon,\exists\,\mathsf{N}:n>\mathsf{N}\to\mathbb{P}\left[\left|\frac{\mathsf{d}(x,y)}{n}-p\right|\geq\xi\right]<\varepsilon$$

Inversons maintenant le point de vue : regardons une boule $\mathcal{B}_y$ de rayon $n(p+\xi)$ centrée sur un message y reçu. Si l'on applique la méthode du maximum de vraisemblance, par le théorème VI.1.10, une erreur de décodage se produira dans deux circonstances exclusives : soit le mot émis est hors de cette boule [8] ($x\notin\mathcal{B}_y$), soit cette boule contient au moins un autre mot de code ($\exists\,x'\neq x:x'\in\mathcal{B}_y$). La probabilité que ces événements se produisent est négligeable, mais nous allons l'évaluer.

La probabilité du premier cas peut être directement bornée par application de la loi des grands nombres : $n>\mathsf{N}\Rightarrow\mathbb{P}[x\notin\mathcal{B}_y]<\varepsilon$. Cette borne ε est purement combinatoire (avec ξ, elle détermine N), donc *indépendante du code* K_n *particulier utilisé*.

La probabilité du second est majorée par $\sum_{x'\neq x}\mathbb{P}[x'\in\mathcal{B}_y]$. Pour l'évaluer, nous allons considérer tous les codes aléatoires K_n possibles et prendre la moyenne, ce qui permet d'oublier le x émis particulier et de considérer tous les mots reçus comme équiprobables :

$$\mathbb{E}[\mathbb{P}_{\text{err}}]<\varepsilon+(r^{k_n}-1)\frac{\mathsf{B}\big(n\,(p+\xi)\big)}{r^n}$$

Utilisant le lemme VI.1.9, pour n grand on trouve

$$\mathbb{E}[\mathbb{P}_{\text{err}}]<\varepsilon+(r^{k_n}-1)\frac{r^{n\mathcal{H}_r(p+\xi)}}{r^n}=\varepsilon+(r^{k_n}-1)r^{-n[1-\mathcal{H}_r(p+\xi)]}$$

Par (V.15), nous substituons

$$1-\mathcal{H}_r(p+\xi)=1-\mathcal{H}_r(p)+\mathcal{H}_r(p)-\mathcal{H}_r(p+\xi)\geq\mathcal{C}_r-\xi\log_r\left((r-1)\frac{1-p}{p}\right)$$

en prenant l'approximation linéaire de $\mathcal{H}_r(p)$, convexe [9] et croissante en $p<\frac{1}{2}$. Ce dernier terme est donc proportionnel à ξ d'une constante multiplicative positive indépendante de n ; notons-la $\mathsf{M}_r(p)>0$.

Nous avons ainsi

$$\mathbb{E}[\mathbb{P}_{\text{err}}]<\varepsilon+(r^{k_n}-1)r^{-n[\mathcal{C}_r-\xi\mathsf{M}_r(p)]}<\varepsilon+r^{-n[\mathcal{C}_r-\mathsf{R}(\mathsf{K}_n)-\xi\mathsf{M}_r(p)]}$$

Si l'on a choisi $\xi<\frac{\mathcal{C}_r-\mathsf{R}(\mathsf{K}_n)}{\mathsf{M}_r(p)}$ (*i.e.* suffisamment petit, puisque $\mathcal{C}_r-\mathsf{R}(\mathsf{K}_n)<\delta$), on a borné $\mathbb{E}[\mathbb{P}_{\text{err}}]<\varepsilon$ pour n grand.

Enfin, la troisième et dernière étape consiste à montrer que l'on peut sélectionner un « bon » code K_n. Pour un n fixé, il existe nécessairement un code $\hat{\mathsf{K}}_n$ pour lequel $\mathbb{P}_{\text{err}}(\hat{\mathsf{K}}_n)\leq\mathbb{E}[\mathbb{P}_{\text{err}}(\mathsf{K}_n)]$.

En résumé, on a donc

$$\forall\,\delta>0,\forall\,\varepsilon>0,\exists\,\mathsf{N}:\forall\,n>\mathsf{N},\exists\,\mathsf{K}:\mathcal{C}-\mathsf{R}(\mathsf{K})<\delta\wedge\mathbb{P}_{\text{err}}(\mathsf{K})<\varepsilon$$

ce qui achève la démonstration. ∎

8. Qu'il y ait un autre message possible dans la boule ou non.
9. Sa dérivée seconde en p est négative.

VI.2.3 L'inverse du théorème de Shannon

En 1959, Jacob Wolfowitz a publié [46] une propriété « inverse » du théorème fondamental de Shannon, ce qui illustre mieux encore le sens de « capacité » d'un canal bruité. La démonstration de ce théorème faisait usage du lemme suivant, utile en soi.

Lemme VI.2.3 (Borne de Fano ~1950, publié en 1961)**.** *Pour toute source de m symboles et tout canal bruité* $\mathbb{P}[y_j\,|\,x_i]$ *de probabilité d'erreur de décodage* $\mathbb{P}_{err}$ *on a*

$$\mathrm{H}(\mathrm{X}\,|\,\mathrm{Y}) \leq \mathcal{H}_m(\mathbb{P}_{err})$$

où $\mathcal{H}_m(p)$ *est la fonction d'entropie à m symboles (III.5).*

Démonstration. Si $\hat{x}_j$ maximise $\mathbb{P}[x_i\,|\,y_j]$ (maximum de vraisemblance), nous savons (voir démonstration de la propriété VI.1.3) que $1-\mathbb{P}_{\mathrm{err}} = \sum_j \mathbb{P}[\hat{x}_j, y_j]$. De même, $\mathbb{P}_{\mathrm{err}} = \sum_j \sum_{x_i \neq \hat{x}_j} \mathbb{P}[x_i, y_j]$.

Considérons le membre de droite

$$\begin{aligned}
\mathcal{H}_m(\mathbb{P}_{\mathrm{err}}) &= -\mathbb{P}_{\mathrm{err}} \log_m \mathbb{P}_{\mathrm{err}} - (1-\mathbb{P}_{\mathrm{err}}) \log_m (1-\mathbb{P}_{\mathrm{err}}) + \mathbb{P}_{\mathrm{err}} \log_m (m-1) \\
&= \mathbb{P}_{\mathrm{err}} \log_m \frac{m-1}{\mathbb{P}_{\mathrm{err}}} + (1-\mathbb{P}_{\mathrm{err}}) \log_m \frac{1}{\mathbb{P}_{\mathrm{err}}} \\
&= \sum_j \sum_{x_i \neq \hat{x}_j} \mathbb{P}[x_i, y_j] \log_m \frac{m-1}{\mathbb{P}_{\mathrm{err}}} + \sum_j \mathbb{P}[\hat{x}_j, y_j] \log_m \frac{1}{1-\mathbb{P}_{\mathrm{err}}}
\end{aligned}$$

De (V.5), nous pouvons écrire celui de gauche de façon similaire

$$\begin{aligned}
\mathrm{H}(\mathrm{X}\,|\,\mathrm{Y}) &= -\sum_j \sum_i \mathbb{P}[x_i, y_j] \log_m \mathbb{P}[x_i\,|\,y_j] \\
&= \sum_j \sum_{x_i \neq \hat{x}_j} \mathbb{P}[x_i, y_j] \log_m \frac{1}{\mathbb{P}[x_i\,|\,y_j]} + \sum_j \mathbb{P}[\hat{x}_j, y_j] \log_m \frac{1}{\mathbb{P}[\hat{x}_j\,|\,y_j]}
\end{aligned}$$

Soustrayons membre à membre, nous avons

$$\mathrm{H}(\mathrm{X}\,|\,\mathrm{Y}) - \mathcal{H}_m(\mathbb{P}_{\mathrm{err}}) = \sum_j \sum_{x_i \neq \hat{x}_j} \mathbb{P}[x_i, y_j] \log_m \frac{\mathbb{P}_{\mathrm{err}}}{(m-1)\mathbb{P}[x_i\,|\,y_j]} + \sum_j \mathbb{P}[\hat{x}_j, y_j] \log_m \frac{1-\mathbb{P}_{\mathrm{err}}}{\mathbb{P}[\hat{x}_j\,|\,y_j]}$$

Du lemme III.1.6, nous trouvons une majoration

$$\begin{aligned}
\mathrm{H}(\mathrm{X}\,|\,\mathrm{Y}) - \mathcal{H}_m(\mathbb{P}_{\mathrm{err}}) &\leq \sum_j \sum_{x_i \neq \hat{x}_j} \frac{\mathbb{P}[x_i, y_j]}{\ln m} \left(\frac{\mathbb{P}_{\mathrm{err}}}{(m-1)\mathbb{P}[x_i\,|\,y_j]} - 1 \right) + \sum_j \frac{\mathbb{P}[\hat{x}_j, y_j]}{\ln m} \left(\frac{1-\mathbb{P}_{\mathrm{err}}}{\mathbb{P}[\hat{x}_j\,|\,y_j]} - 1 \right) \\
&= \frac{\mathbb{P}_{\mathrm{err}}}{\ln m} \frac{1}{(m-1)} \sum_j \sum_{x_i \neq \hat{x}_j} \frac{\mathbb{P}[x_i, y_j]}{\mathbb{P}[x_i\,|\,y_j]} + \frac{1-\mathbb{P}_{\mathrm{err}}}{\ln m} \sum_j \frac{\mathbb{P}[\hat{x}_j, y_j]}{\mathbb{P}[\hat{x}_j\,|\,y_j]} - \sum_j \sum_i \frac{\mathbb{P}[x_i, y_j]}{\ln m} \\
&= \frac{\mathbb{P}_{\mathrm{err}}}{\ln m} \sum_j \frac{1}{(m-1)} \sum_{x_i \neq \hat{x}_j} \mathbb{P}[y_j] + \frac{1-\mathbb{P}_{\mathrm{err}}}{\ln m} \sum_j \mathbb{P}[y_j] - \frac{1}{\ln m} \\
&= \frac{1}{\ln m} \left(\mathbb{P}_{\mathrm{err}} \sum_j \mathbb{P}[y_j] + (1-\mathbb{P}_{\mathrm{err}}) - 1 \right) \\
&= 0
\end{aligned}$$

∎

Théorème VI.2.4 (Jacob Wolfowitz, 1959)**.** *Pour tout canal bruité symétrique de probabilité d'erreur* $p < \frac{1}{2}$ *et de capacité* $\mathcal{C}_r$*, pour toute famille* $\{\mathrm{K}_n\}$ *de codes en bloc de longueur n et de débit* $\mathrm{R}(\mathrm{K}_n) \geq \mathcal{C}_r + \epsilon$*,* ϵ *réel positif fixé arbitrairement petit, on a* $\lim_{n\to\infty} \mathbb{P}_{err}(\mathrm{K}_n) = 1$*.*

La démonstration est très technique et semblable à celle du théorème fondamental de Shannon (voir [19, 2d ed., p.207]).

Par conséquent, pour toute famille de codes fiables, le débit est borné par la capacité du canal :

$$1 - \mathbb{P}_{\mathrm{err}}(\mathrm{K}_n) \approx 1 \Rightarrow \mathrm{R}(\mathrm{K}_n) \leq \mathcal{C}_r$$

VI.3 Théorie algébrique des codes correcteurs

Nous nous plaçons évidemment sous les hypothèses générales décrites à la section VI.1.4.

VI.3.1 Distance minimale d'un code

Les notions de distance et de poids de Hamming s'étendent aux codes et constituent les paramètres d et w intervenant dans plusieurs propriétés.

Distance minimale La distance minimale d'un code est la plus petite distance de Hamming séparant deux mots distincts du code : $d = \mathrm{d}(\mathrm{K}) = \min\{\mathrm{d_H}(x,y) \mid x,y \in \mathrm{K} \wedge x \neq y\}$.

Poids du code Le poids d'un code est le plus petit poids de Hamming des mots non nuls du code ; c'est la distance du mot le plus proche de 0 : $w = \mathrm{w}(\mathrm{K}) = \min\{\mathrm{w_H}(x) \mid x \in \mathrm{K} \wedge x \neq 0\}$. Si $0 \in \mathrm{K}$, on a $\mathrm{d}(\mathrm{K}) \leq \mathrm{w}(\mathrm{K})$.

La distance minimale d'un code est, comme nous le verrons par les nombreuses propriétés suivantes, un paramètre essentiel. Voilà pourquoi les codes algébriques sont généralement notés

$$\boxed{\mathrm{A}_r(n,d)}$$

ce qui indique leurs paramètres essentiels.

Définition (Code maximal)**.** Un code $\mathrm{A}_r(n,d)$ est dit « maximal » si tout ajout de mot dans le code réduit sa distance minimale.

Borne de Singleton

Une borne générale sur la taille d'un code a été énoncée par Richard Collom Singleton [10] ([35]).

Propriété VI.3.1 (Borne de Singleton, 1964)**.** *Pour tout code* $\mathrm{A}_r(n,d)$

$$|\mathrm{A}_r(n,d)| \leq r^{n-d+1}$$

En particulier, cette borne doit être respectée par tout code maximal.

Démonstration. L'ensemble de tous les mots possibles C^n est de taille r^n. Les mots du code $\mathrm{K} = \mathrm{A}_r(n,d)$ sont tous distincts et à distance au moins $d = \mathrm{d}(\mathrm{K})$ les uns des autres, par définition. Considérons l'espace tronqué où l'on oublie les $(d-1)$ caractères de chaque mot. Les mots du code y constituent un sous-ensemble, un code $\tilde{\mathrm{K}}$ avec $\mathrm{d}(\tilde{\mathrm{K}}) \geq 1$. Tous les mots de ce code sont donc distincts et $|\tilde{\mathrm{K}}| = |\mathrm{K}|$. Or $|\tilde{\mathrm{K}}| \leq r^{n-d+1}$. ∎

Corollaire VI.3.2. *Si* $|\mathrm{K}| = r^k$ *(i.e.* K *code tous les mots de k caractères), la borne de Singleton devient* $d \leq n - k + 1$*.*

Nous verrons que certains codes peuvent atteindre cette borne. Ainsi, parmi les codes linéaires, les codes MDS sont précisément ceux qui présentent cette propriété remarquable ; les codes de Reed-Solomon en sont un exemple.

10. Il ne faut pas confondre ce nom propre avec le terme, le nom commun désignant un ensemble d'un seul élément (qui vient de l'anglais *single*). La « borne de Singleton » ne concerne pas les codes triviaux !

VI.3.2 Relations entre distance et correction

On trouve immédiatement des bornes et propriétés liées aux paramètres d'un code correcteur d'erreur.

Propriété VI.3.3 (Détection d'erreur)**.** *Un code* K *de distance minimale d détecte* τ *erreurs si et seulement si*

$$\tau \leq d - 1 < d$$

Démonstration. Par l'absurde : il existe une paire de mots du code $\{x_1, x_2\}$ à distance $d = \mathrm{d}(x_1, x_2)$. Si l'on émet x_1 et que l'on reçoit x_2, donc avec d erreurs, celles-ci ne sont pas détectées, puisque $x_2 \in \mathrm{K}$.

A contrario, si l'on émet x et qu'on reçoit y avec τ erreurs où $\tau < d$, donc $t = \mathrm{d}(x, y)$, on a nécessairement $y \notin \mathrm{K}$ par définition de d. ∎

Propriété VI.3.4 (Correction d'erreur)**.** *Un code* K *de distance minimale d corrige t erreurs si et seulement si*

$$t \leq \left\lfloor \frac{d-1}{2} \right\rfloor < \frac{d}{2}$$

Démonstration. Il existe une paire de mots du code $\{x_1, x_2\}$ à distance $d = \mathrm{d}(x_1, x_2)$. Supposons que l'on émette x_1 et reçoive y où t caractères de x_1 sont remplacés par ceux de x_2 en des positions où ils sont différents ($t < d$, donc). On a $\mathrm{d}(x_1, y) = t$ et $\mathrm{d}(x_2, y) = d - t$. En application du théorème VI.1.10, la correction de y en x_1 n'aura lieu que si $t < d - t$; ce sera en x_2 si $t < d - t$, mais si $t = \frac{d}{2}$, ce sera ambigu.

Pourrait-il y avoir x_3 préféré aux deux autres ? Supposons $t < d - t$.
Par définition de d, $\mathrm{d}(x_1, x_3) \geq d$. Par l'inégalité triangulaire, $\mathrm{d}(x_1, x_3) \leq \mathrm{d}(x_1, y) + \mathrm{d}(y, x_3) \leq 2t$, donc $t \geq \frac{d}{2}$ ce qui est contraire à l'hypothèse. Le cas $t > d - t$ se démontre de manière équivalente *via* x_2. ∎

VI.3.3 Rayons d'empilement et de recouvrement

On sait que C^n muni de la distance de Hamming est un espace métrique. Une autre façon de décrire les propriétés précédentes est donc de regarder les familles Γ de boules de rayon ϱ fixé centrées sur les mots du code[11] : $\Gamma(\mathrm{K}, \varrho) = \{\mathcal{B}(x, \varrho) \mid x \in \mathrm{K}\}$. Celles-ci correspondent en fait au voisinage des mots qui peuvent être reçus si *au plus* ϱ erreurs se sont produites. L'union de ces boules (ou fermeture) $\Gamma^*(\mathrm{K}, \varrho) = \bigcup_{x \in \mathrm{K}} \mathcal{B}(x, \varrho) \subset \mathrm{C}^n$ est un sous-ensemble de l'espace des mots ; c'est celui des mots contenant au plus ϱ erreurs.

De manière évidente, si toutes les boules d'une famille $\Gamma(\mathrm{K}, \varrho)$ sont disjointes, les mots de chaque boule seront corrigés en leur centre, en vertu du théorème VI.1.10 ; le code peut donc corriger au moins ϱ erreurs. Ceci conduit à la définition suivante.

Définition (Rayon d'empilement)**.** Le rayon d'empilement (*packing radius*) d'un code K est le plus grand rayon s tel que les boules de $\Gamma(\mathrm{K}, s)$ sont disjointes.

Ainsi, si le code corrige t erreurs, les boules de rayon t doivent être disjointes, donc

$$t \leq s \tag{VI.6}$$

Si $\Gamma^*(\mathrm{K}, \varrho) \subsetneq \mathrm{C}^n$, il existe au moins un mot y tel que $\forall\, x \in \mathrm{K} : \mathrm{d}(x, y) > \varrho$. Voilà pourquoi on définit :

Définition (Rayon de recouvrement)**.** Le rayon de recouvrement (*covering radius*) d'un code K est le plus petit rayon c tel que les boules de $\Gamma(\mathrm{K}, c)$ recouvrent l'espace, c'est-à-dire $\Gamma^*(\mathrm{K}, c) = \mathrm{C}^n$.

Si l'on compare les deux rayons, on a trivialement $s \leq c$. Mais, de manière plus précise :

11. Dans notre contexte, nous parlerons systématiquement de « code », mais il est évident qu'il s'agit d'un sous-ensemble quelconque de l'espace.

Propriété VI.3.5.

$$s = \left\lfloor \frac{d-1}{2} \right\rfloor \leq \frac{d-1}{2} \leq \left\lfloor \frac{d}{2} \right\rfloor \leq c$$

Démonstration. La première égalité (sur s) se démontre en considérant deux mots du code à distance minimale : $d = \mathrm{d}(x_1, x_2)$. Les boules $\mathcal{B}(x_i, s)$ ne peuvent se toucher, donc $s < \frac{d}{2}$, c'est-à-dire $s \leq \frac{d-1}{2}$, et comme s est un entier, $s \leq \delta = \left\lfloor \frac{d-1}{2} \right\rfloor$. Si s était strictement plus petit, tout mot y à distance δ de x_1, par exemple, serait à distance strictement supérieure à δ de tout autre mot x, sinon $\mathrm{d}(x_1, x) \leq \mathrm{d}(x_1, y) + \mathrm{d}(y, x) \leq 2\delta < d$, ce qui est contradictoire. Donc la borne δ pour s peut être atteinte.

Pour la dernière (celle qui concerne c), on observe un mot y à mi-distance entre ces deux x_i. Comme cette distance est entière, on a $\mathrm{d}(x_1, y) = \left\lfloor \frac{d}{2} \right\rfloor$ ou l'analogue pour x_2 (les deux si d est pair). Or ce mot doit être recouvert par une boule centrée en ce x_i le plus proche des deux, car il n'y a pas de x_3 strictement plus proche sinon $\mathrm{d}(x_i, x_3) \leq \mathrm{d}(x_i, y) + \mathrm{d}(y, x_3) < 2\left\lfloor \frac{d}{2} \right\rfloor < d$, ce qui est en contradiction avec la définition de d. Mais cette borne peut être largement dépassée ; il suffit de considérer un code minimal $|\mathrm{K}| = 2$ avec $d = 1$ et $n \geq 2$, donc $c = n$ si $r > 2$ et $c = n - 1$ en binaire. ∎

VI.3.4 Borne de Hamming et codes parfaits

La situation limite de la propriété VI.3.5 est celle des codes dits « *parfaits* ».

Un simple raisonnement arithmétique sur la propriété VI.3.5 montre que si $s = c$, d doit être un nombre impair. Sous ces hypothèses, tout mot y appartient à une et une seule boule de rayon $\frac{d-1}{2}$ et peut donc être décodé comme le centre de la boule à laquelle il appartient. On peut donc corriger $\frac{d-1}{2} = t$ erreurs. Les boules $\Gamma(\mathrm{K}, t)$ constituent une *partition* de l'espace ; ces boules *pavent* l'espace.

Définition (Code parfait)**.** Un code parfait (*perfect code*) est un code pour lequel les rayons d'empilement et de recouvrement sont égaux. Il permet donc de corriger t erreurs :

$$t = s = c \Rightarrow d = 2t + 1 \tag{VI.7}$$

La question, que nous aborderons plus loin, est évidemment de construire de tels codes parfaits.

Dans [18], Richard Hamming a démontré l'inégalité suivante sur le nombre de mots d'un code correcteur :

Propriété VI.3.6 (Borne de Hamming, 1950)**.** *Tout code* $\mathrm{A}_r(n, d)$ *satisfait*

$$|\mathrm{A}_r(n, d)| \leq \frac{r^n}{\mathrm{B}_s}$$

où $s = \left\lfloor \frac{d-1}{2} \right\rfloor$ *est le rayon d'empilement du code et où* B_t *est le nombre de mots d'une boule de rayon t donné par (VI.3).*

Démonstration. Considérons le rayon d'empilement du code : s. Les boules de $\Gamma(\mathrm{K}, s)$ sont disjointes et contiennent chacune B_s mots. Donc leur union $\Gamma^*(\mathrm{K}, s)$ contient exactement $\mathrm{B}_s . |\mathrm{A}_r(n, d)|$ mots distincts, nombre borné par la taille de l'espace r^n. ∎

Théorème VI.3.7. *Un code est parfait si et seulement si sa taille atteint la borne de Hamming.*

Démonstration. Cette propriété découle immédiatement de la définition des rayons d'empilement et de recouvrement. Si $s = c$, $\Gamma(\mathrm{K}, s) = \Gamma(\mathrm{K}, c)$, donc $\mathrm{B}_s . |\mathrm{A}_r(n, d)| = r^n$. Réciproquement, si $\mathrm{B}_s . |\mathrm{A}_r(n, d)| = r^n$, $\Gamma(\mathrm{K}, s)$ recouvre l'espace et $s = c$. ∎

Corollaire VI.3.8. *Si* $|\mathrm{K}| = r^k$ *(i.e.* K *code tous les mots de k caractères), la borne de Hamming devient* $\mathrm{B}_s \leq r^{n-k}$*. Ces boules sont donc des sous-ensembles correspondant à la redondance de* $(n - k)$ *caractères.*

VI.3.5 Borne de Gilbert-Varshamov

Les bornes de Singleton et de Hamming sont des bornes supérieures du nombre de mots d'un code correcteur. Nous allons décrire maintenant une borne inférieure. Elle provient des travaux indépendants de Edgar Nelson Gilbert ([13], en 1952, de manière probabiliste) et Ром Рубенович Варшамов (Rom Rubenovitch Varshamov, en arménien : Ռոմ Ռուբենի Վարշամով) ([42], en 1957, pour le cas linéaire). Cette borne est ainsi connue sous l'acronyme « borne **GV** ».

Propriété VI.3.9 (Borne GV, 1952–1957)**.** *Un code* $\mathrm{A}_r(n,d)$ *maximal satisfait*

$$|\mathrm{A}_r(n,d)| \geq \frac{r^n}{\mathrm{B}_{d-1}}$$

où B_t *est le nombre de mots d'une boule de rayon* t *donné par (VI.3).*

Démonstration. Comme le code est maximal, tout mot hors code est à distance $\leq d-1$ d'un mot du code. Ainsi, l'union des boules $\Gamma^*(\mathrm{K}, d-1)$ recouvre l'espace, chacune contenant B_{d-1} mots, mais elles ne sont pas nécessairement disjointes. On a donc $r^n = |\Gamma^*(\mathrm{K}, d-1)| \leq \mathrm{B}_{d-1}.|\mathrm{A}_r(n,d)|$. ∎

Une autre façon d'exprimer ce résultat est sous forme d'une borne sur le rayon de recouvrement :

$$\mathrm{K} \text{ maximal } \Rightarrow c \leq d-1$$

Le débit d'un code est lié asymptotiquement à la borne GV. Définissons la *distance relative* d'un code correcteur $\mathrm{A}_r(n,d)$ comme le rapport

$$\mathcal{D} = \frac{d}{n}$$

et le *débit optimum*, pour une distance relative donnée, comme

$$\mathcal{R}(\mathcal{D}) = \limsup_{n\to\infty} \frac{\log_r |\mathrm{A}_r(n, n\mathcal{D})|}{n}$$

Ceci permet d'énoncer le théorème suivant.

Théorème VI.3.10 (Borne GV asymptotique)**.** *Pour toute distance relative* $0 \leq \mathcal{D} \leq 1$*,*

$$\mathcal{R}(\mathcal{D}) \geq 1 - \mathcal{H}_r(\mathcal{D})$$

où $\mathcal{H}_r(p)$ *est la fonction entropie généralisée (III.5).*

Nous ne démontrerons pas ce théorème ici.

Les codes de Goppa que nous évoquerons rapidement plus loin sont un exemple de codes linéaires dont le débit dépasse cette borne GV asymptotique ; ils sont donc efficaces.

Chapitre VII

Codes linéaires

Les codes linéaires constituent une famille très riche de codes correcteurs d'erreur largement utilisés. Nous allons en présenter quelques cas particuliers importants, après avoir introduit leur définition et leurs propriétés générales.

VII.1 Outils mathématiques

Un code *linéaire* se décrit et se traite évidemment à l'aide d'outils d'algèbre *linéaire*. Mais comme il s'agit d'ensembles *finis*, il nous faut au préalable rappeler (ou introduire) quelques notions de mathématiques discrètes indispensables dans ce contexte.

VII.1.1 Anneau et module sur anneau

Aux hypothèses de la section VI.1.4, on ajoute souvent celle qui consiste à représenter les caractères par un nombre. L'alphabet est alors vu comme l'**anneau unitaire commutatif** $\mathsf{C} \simeq \mathbb{Z}/r\mathbb{Z} = \mathbb{Z}_r$ des *entiers modulo r*, ce qui permet de faire des opérations usuelles $(+, -, \times)$ sur les caractères et d'exprimer certaines propriétés ou constructions d'un code sous une forme purement algébrique.

Rappelons qu'un *anneau* est un ensemble qui est muni d'une opération additive qui constitue un groupe commutatif et d'une opération multiplicative, distributive par rapport à l'addition [1]. Il est *unitaire* s'il contient une unité, un élément neutre pour la multiplication [2]. Il est *commutatif* si sa multiplication l'est. On montre que le neutre pour l'addition, l'élément nul, est absorbant pour la multiplication.

Un **idéal** $\mathfrak{I}$ d'un anneau A est un sous-anneau *stable pour la multiplication*, c'est-à-dire que ses éléments constituent un sous-groupe additif $(\forall\, x, y \in \mathfrak{I} : x - y \in \mathfrak{I})$ et que leur produit par un élément *quelconque* de l'anneau est toujours dans l'idéal $(\forall\, x \in \mathfrak{I}, \forall\, a \in \mathsf{A} : a \cdot x \in \mathfrak{I}$ et $x \cdot a \in \mathfrak{I})$. Dans un anneau A commutatif, si $\mathfrak{I}$ est un idéal, on peut parler de *l'anneau-quotient* $\mathsf{A}/\mathfrak{I}$ où les opérations se font *modulo* les éléments de $\mathfrak{I}$.

La notation $\mathbb{Z}/r\mathbb{Z}$ est un exemple, car $r\mathbb{Z}$ est bien un idéal de $\mathbb{Z}$; les opérations s'y font bien « modulo r ».

Module

Ces opérations modulo r s'étendent naturellement aux n-uplets, les mots de C^n, que l'on peut additionner entre eux et multiplier par tout nombre (un *scalaire*). On peut également définir un produit scalaire, mais il faut se méfier, car le carré scalaire ne peut s'interpréter comme le carré d'une norme (par exemple, le carré scalaire du vecteur $[1, 2, 1]$ est nul modulo 3).

1. $\forall a, b, c : a(b + c) = ab + ac$ et $(a + b)c = ac + bc$.
2. Un contre-exemple est l'anneau $2\mathbb{Z}$ des entiers pairs qui n'a pas d'unité, donc aucun inverse.

On dit naturellement que ces mots sont des « vecteurs », même si le terme est ici partiellement impropre. Cette structure est, plus précisément, un « **module sur un anneau »** (ici le module $\mathbb{Z}_r^n$) et la taille des mots (n) est appelée la *dimension* du module [3]. Dans un module sur anneau, les éléments de l'anneau sont appelés les « nombres *scalaires* ».

Diviseurs de zéro

En général, l'anneau $\mathbb{Z}_r$ possède des *diviseurs de zéro*, c'est-à-dire des éléments tels que $a \cdot b = 0$ avec $a \neq 0$ et $b \neq 0$.

Comme tout anneau unitaire, il se partitionne en deux familles d'éléments : les « inverses », qui sont ici les éléments premiers avec r pour lesquels il existe donc [4] a^{-1} tel que $a \cdot a^{-1} = 1$, et les « diviseurs de zéro », ceux qui possèdent un diviseur non trivial (*i.e.* différent de 1) commun avec r.

Dans un anneau unitaire, il existe toujours au moins un inverse : l'unité. Un anneau commutatif sans diviseurs de zéro est appelé un anneau *intègre* ; l'ensemble des nombres entiers $\mathbb{Z}$ est un exemple d'anneau intègre. L'anneau $\mathbb{Z}_r$ est intègre si et seulement si r est un *nombre premier* [5] ; c'est alors un corps commutatif (tout anneau intègre *fini* est un corps).

VII.1.2 Anneau de polynômes

Un mot de n éléments de $\mathbb{Z}_r$ peut également être vu comme les coefficients d'un *polynôme* de degré $< n$ sur cet anneau. Ces polynômes peuvent s'additionner (ils constituent un groupe commutatif), mais également se multiplier. Ces polynômes constituent ainsi un autre anneau noté $\mathbb{Z}_r[X]$; c'est un ensemble infini.

Lorsqu'on multiplie deux polynômes, leur degré risque de croître, puisque, en toute généralité, $\deg(P \cdot Q) \leq \deg P + \deg Q$ [6]. Un anneau de polynômes est intègre si et seulement si celui de ses coefficients l'est ; alors $\deg(P \cdot Q) = \deg P + \deg Q$.

Comme $\mathbb{Z}_r$ est unitaire, il existe des polynômes M de tout degré dont le coefficient dominant (*i.e.* celui de son degré) est un inverse [7]. Dans ce cas, il est possible d'effectuer une *division euclidienne* par $M : \forall P, \exists Q, R : P = Q \cdot M + R$ avec $\deg R < \deg M$ (et $\deg M + \deg Q = \deg P$). Dans ce cas, on dit que « $R = P \bmod M$ ». Si l'on fixe un tel polynôme M, la multiplication *modulo* M maintient le degré inférieur à celui de M. Cet ensemble constitue également un anneau noté $\mathbb{Z}_r[X]/\langle M \rangle$ ($\langle M \rangle$ est l'*idéal* engendré par M, c'est-à-dire le sous-anneau stable constitué des multiples de M).

Propriété VII.1.1. *Si le polynôme* $M \in \mathbb{Z}_r[X]$ *de degré n a un coefficient dominant qui est un inverse, l'anneau* $\mathbb{Z}_r[X]/\langle M \rangle$ *contient exactement* r^n *éléments correspondant à tous les polynômes de degré* $< n$ *sur* $\mathbb{Z}_r$.

Démonstration. Écrivons $M(X) = \sum_{i=0}^{n} m_i X^i$ avec m_n inverse. Il existe donc m_n^{-1} dans $\mathbb{Z}_r$. Si l'on calcule « modulo M », cela veut dire $M \cong 0$, donc $X^n \cong -m_n^{-1} \sum_{i=0}^{n-1} m_i X^i \pmod M$. Par conséquent, dans les calculs, on peut simplement substituer X^n par ce polynôme $X^n - m_n^{-1} M(X)$ de degré $< n$, et ainsi de suite pour toutes les puissances de X supérieures ou égales à n. ∎

Par conséquent, un espace de mots de longueur n sur un alphabet de r caractères peut être vu comme un tel anneau de polynômes où l'on effectue les calculs modulo un polynôme fixé de degré n ayant un coefficient dominant qui est un inverse dans $\mathbb{Z}_r$.

3. En toute rigueur, on ne peut pas toujours définir la dimension d'un module, parce qu'il n'est pas toujours possible d'y définir une base, même s'il est engendré de façon finie (à partir d'un nombre fini d'éléments générateurs). Mais, dans le cas de $\mathbb{Z}_r$, tout se passe « bien ».

4. Dans ce cas, on calcule aisément l'inverse d'un nombre modulo r par l'algorithme d'Euclide étendu.

5. Un nombre premier est un naturel > 1 qui n'est divisible que par 1 et lui-même.

6. On n'a pas toujours l'égalité, comme c'est le cas avec les polynômes sur $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ ou $\mathbb{C}$, à cause de la présence ici de diviseurs de zéro.

7. On appelle cela un polynôme « unitaire » ou polynôme « monique » (*monic polynomial* en anglais).

VII.1.3 Corps fini et espace vectoriel

Si l'ensemble des scalaires est un **corps** [8], le module sur un corps est simplement un **espace vectoriel**. Dans ce cas, plus encore si le corps est commutatif, on peut lui appliquer la plupart des concepts et propriétés classiques de l'algèbre linéaire réelle ou complexe.

Dans le cas qui nous préoccupe, l'ensemble des scalaires est fini (il contient r éléments). Une première restriction consisterait à ne considérer que les alphabets ayant *un nombre premier p de caractères*. En effet, si p est premier, $\mathbb{Z}_p$ est un corps commutatif fini. C'est une situation très fréquente en théorie des codes, en particulier le cas binaire (*i.e.* $\mathbb{Z}_2 = \{0,1\}$ où l'addition $\oplus$ est XOR et la multiplication $\odot$ est AND).

Mais on peut généraliser : la théorie introduite par Évariste Galois nous enseigne qu'il existe un (et un seul) **corps fini** à q éléments si et seulement $q = p^\nu$, une puissance d'un nombre premier [9] ; on dit que q est un « *nombre primaire* ». Un tel corps fini noté $\mathbb{F}_q$ ou $\mathrm{GF}(q)$ (pour *Galois Field* ou « champ de Galois ») a plusieurs propriétés intéressantes (que nous ne démontrons pas ici, évidemment).

Propriété VII.1.2 (Propriétés essentielles des corps finis)**.**

- *Pour tout $q = p^\nu$, il existe* un et un seul *corps fini, à un isomorphisme près ; on peut donc par « du » corps à q éléments.*
- *Tout corps fini est* commutatif*.*
- *Tout élément satisfait la* relation de Pierre de Fermat*: $a^q - a = 0$.*
- *Tout corps fini est* cyclique*: il existe (au moins) un élément* primitif *g, générateur de son groupe multiplicatif tel que $\forall\, a \neq 0, \exists\, i \,:\, a = g^i$ avec $0 \leq i \leq q-2$. L'indice i est aussi appelé le* logarithme discret *de a en base g.*
- *Pour tout élément a, $\underbrace{a + a + \cdots + a}_{p} = p \cdot a = 0$. On dit que p est la* caractéristique *du corps.*
- *Pour tout $\kappa \mid \nu$, il existe un sous-corps de $\mathbb{F}_{p^\nu}$ ayant p^κ éléments qui sont les racines du polynôme $X^{p^\kappa} - X$ dans ce corps.*

Toutes ces propriétés sont évidemment également vraies dans le cas particulier de $\mathbb{Z}_p$, donc dans la simple arithmétique modulo un nombre premier.

Construction d'un corps fini

Il y a plusieurs façons de construire un corps fini $\mathbb{F}_q$ avec $q = p^\nu$, donc de décrire ses opérations. Nous en choisissons ici une, plus facile à mettre en œuvre dans notre contexte [10].

1. Prendre le corps fini $\mathbb{Z}_p$, dit *corps de base*.
2. Considérer l'anneau des polynômes sur ce corps $\mathbb{Z}_p[X]$; c'est un anneau intègre (sans diviseur de zéro), mais infini.
3. Choisir dans cet anneau un polynôme M de degré ν *irréductible* sur le corps de base (*irreducible polynomial*), c'est-à-dire qu'il est impossible de le factoriser en effectuant les calculs modulo p.
4. Se restreindre aux polynômes modulo M : $\mathbb{Z}_r[X]/\langle M\rangle \simeq \mathbb{F}_q$.

Code comme espace vectoriel

Par la propriété VII.1.1, tous les polynômes de degré inférieur à ν et à coefficients dans le corps de base $\mathbb{Z}_p$ sont les éléments de $\mathbb{F}_q$ si l'on effectue les calculs modulo ce polynôme irréductible M.

8. Un *corps* est un anneau dans lequel tout élément est inverse ; la multiplication sur un corps (privé de son nul, de son neutre additif) constitue un groupe. Un corps commutatif s'appelle un *champ*.
9. Il y a donc un corps à 2, 3, 4, 5, 7, 8, 9, 11, 13, 16, 17… éléments, mais pas à 6, 10, 12, 15… éléments, par exemple.
10. Même si dans certains cas, d'autres représentations des éléments du corps sont plus efficaces pour certains algorithmes.

Ceci permet donc de représenter les mots de longueur n sur un alphabet à $r = p^\nu$ caractères C^n comme l'*espace vectoriel* $\mathbb{F}_q^n$ de dimension n sur le corps fini $\mathbb{F}_q$. Les mots sont donc les *vecteurs* de cet espace.

Un cas particulier très important est celui où les « caractères » sont des octets. Ceux-ci peuvent donc être vus comme des éléments de $\mathbb{F}_{256}$, un corps fini de caractéristique 2 (l'addition est un XOR bit à bit) dont les bits sont les coefficients binaires d'un polynôme de degré < 8. Attention, il s'agit bien d'un corps fini, donc de calculs effectués modulo un polynôme binaire irréductible[11] de degré 8, pas de calculs modulo 256 ! Les mots de longueur n sont alors les vecteurs d'un espace de dimension n sur ce corps.

Définition (Espace vectoriel dual)**.** Pour tout espace vectoriel V sur un corps F, on définit son *espace dual* V^* comme l'espace des formes linéaires sur V, c'est-à-dire les applications $\varphi : V \to F$ telles que $\varphi(\lambda a + \mu b) = \lambda\,\varphi(a) + \mu\,\varphi(b)$.

Comme ici la dimension de V est finie[12] (elle vaut n, la longueur des mots), le dual est isomorphe à l'espace d'origine : $V^* \simeq V$, donc $\dim V^* = \dim V$.

Notons enfin que, même si les messages constituent un espace vectoriel de dimension *finie*, nous rencontrerons également d'autres cas. Ainsi, les polynômes sur un corps, en particulier les polynômes de $\mathbb{F}_q[X]$, sont un exemple d'espace vectoriel de dimension infinie sur le corps ; les monômes $\{X^i\}$ en constituent une base, puisque tout vecteur (ici polynôme) est bien une combinaison linéaire d'un nombre *fini* de vecteurs de cette base.

VII.2 Définition et propriétés d'un code linéaire

Définition (Code linéaire)**.** Un code linéaire est un code satisfaisant les hypothèses suivantes.

1. L'espace des messages codés (les mots du code ainsi que les mots erronés) est un *espace vectoriel* $\mathbb{F}_q^n$ de dimension n sur un corps fini à $q = p^\nu$ éléments (où p est un nombre premier, caractéristique du corps).
2. L'espace des symboles originels est également un espace vectoriel $\mathbb{F}_q^k$ de dimension k sur le même corps.
3. La fonction de codage est une application linéaire ; le code est donc un *sous-espace vectoriel* $K \subset \mathbb{F}_q^n$ de dimension k.

Afin de nommer ces deux paramètres, k est appelée la *dimension* du code (il y a q^k mots codés), tandis que n est sa *longueur*, le nombre de caractères utilisés pour encoder un mot.

Par définition, un sous-espace vectoriel est un sous-ensemble K contenant le vecteur nul et toutes les combinaisons linéaires de ses éléments :

$$0 \in K \quad \wedge \quad a, b \in K \Rightarrow \lambda a + \mu b \in K$$

Mais on a d'autres critères :

Propriété VII.2.1. *Un sous-ensemble* K *d'un espace vectoriel* E *sur un corps* F *est un sous-espace s'il satisfait*

- $K \neq \varnothing$.
- $\forall\, a, b \in K : (a - b) \in K$.
- $a \in K \Rightarrow g \cdot a \in K$ *où* g *est générateur du groupe multiplicatif du corps ; c'est une hypothèse inutile si* $F = \mathbb{F}_p$.

Démonstration. Comme K est non vide, il contient un élément a, donc $a - a = 0 \in K$.

Ensuite, $-a = 0 - a \in K$, $2a = a - (-a) \in K$ et par itération, $i \cdot a \in K$ et $-i \cdot a \in K$ pour tout nombre i.

11. Par exemple $M(X) = X^8 + X^4 + X^3 + X + 1$.
12. En dimension infinie, V est un sous-espace strict de V^* : $V \subsetneq V^*$.

Si le corps K est simplement celui des entiers modulo p, on obtient ainsi toutes les combinaisons linéaires. Sinon, on utilise $a \mapsto g \cdot a$ et de manière itérative, on obtient $\forall\, \lambda \in \mathsf{F} : \lambda \cdot a \in \mathsf{K}$.

En combinant le tout on a

$$\forall\, \lambda, \mu \in \mathsf{F}, \forall\, a, b \in \mathsf{K} : \lambda a + \mu b \in \mathsf{K}$$

donc toute combinaison linéaire de deux vecteurs de K appartient bien à K. ∎

Comme le code est univoque, le code K est également de dimension $\dim \mathsf{K} = k \leq n$, donc $\mathsf{K} \simeq \mathbb{F}_q^k$, puisque tous les espaces vectoriels de même dimension finie sur le même corps sont isomorphes. Et comme c'est un sous-espace, il possède une base (de k vecteurs) qui permet de le décrire complètement.

On note généralement un code linéaire par ses paramètres essentiels

$$\boxed{[n, k, d]_q}$$

où

- n est la longueur du code, celle des mots encodés ;
- k est la dimension du code, le nombre de caractères des symboles originels ;
- d la distance minimale du code ;
- q le nombre primaire de caractères ($q = p^\nu$ pour p premier).

Ce code $[n, k, d]_q$ a donc un débit $\mathsf{R} = \frac{k}{n}$ et les inégalités remarquables deviennent : $d + k \leq n + 1$ (Singleton), $\mathsf{B}_s \leq q^{n-k}$ (Hamming).

VII.2.1 Matrice génératrice

Comme nous travaillons dans des espaces vectoriels (sur un corps fini $\mathbb{F}_q$), la fonction d'encodage est une *application linéaire injective* (sinon le code ne serait pas univoque) de l'espace vectoriel $\mathbb{F}_q^k$ dans l'espace vectoriel $\mathbb{F}_q^n$. La fonction d'encodage peut donc être représentée par une *matrice* G d'éléments du corps de rang k maximal (sur ce corps). Cette matrice G s'appelle la *matrice génératrice* du code.

Ainsi, dans la notation d'algèbre linéaire, l'encodage du message x est simplement $\mathsf{G}\,x$ où x est un *vecteur-colonne* et G une matrice $n \times k$ (n lignes et k colonnes). Mais dans de nombreux ouvrages de théorie des codes, les mots sont notés sous forme de *vecteurs-ligne*, comme des éléments de l'espace dual. Pour suivre cette convention, il suffit de transposer G en G^τ.

Définition (Matrice génératrice)**.** La matrice génératrice d'un code linéaire est la matrice G de l'application linéaire injective de $\mathbb{F}_q^k$ dans $\mathbb{F}_q^n$ dont l'image est le code $\mathsf{K} \subset \mathbb{F}_q^n$.

$$\mathsf{G}(\mathsf{K}) : \mathbb{F}_q^k \to \mathbb{F}_q^n : x \mapsto y = \mathsf{G}\,x \quad \text{avec} \quad \operatorname{im} \mathsf{G} = \mathsf{G}(\mathbb{F}_q^k) = \mathsf{K} \quad \text{et} \quad \operatorname{rank} \mathsf{G} = k$$

Cette matrice G possède une *pseudo-inverse* G^+ de dimensions $k \times n$ (les dimensions sont celles de sa transposée G^τ).

Définition (Pseudo-inverse)**.** Si G est une application linéaire de V dans W, une pseudo-inverse G^+ est une application de W dans V qui doit satisfaire simultanément les deux propriétés duales :

$$\mathsf{G}\,\mathsf{G}^+\,\mathsf{G} = \mathsf{G} \quad \text{et} \quad \mathsf{G}^+\,\mathsf{G}\,\mathsf{G}^+ = \mathsf{G}^+$$

Propriété VII.2.2. *Si* G^+ *est une pseudo-inverse de* G*, alors* $\operatorname{rank} \mathsf{G} = \operatorname{rank} \mathsf{G}^+$ *; de plus, les espaces* V *et* W *peuvent se décomposer en sommes directes :* $\mathsf{W} \simeq (\operatorname{im} \mathsf{G} \oplus \ker \mathsf{G}^+)$ *et* $\mathsf{V} \simeq (\ker \mathsf{G} \oplus \operatorname{im} \mathsf{G}^+)$.

Les définitions de ces concepts (noyau et somme directe) sont rappelées ci-dessous.

Définition (Noyau)**.** Le *noyau* d'une application linéaire noté « ker » pour (*kern*, en anglais) est l'ensemble des éléments dont l'image est nulle. C'est un sous-espace linéaire du domaine de l'application.

En effet, $\forall x, y \in \ker\varphi, \forall \lambda, \mu : \varphi(\lambda x + \mu y) = \lambda\varphi(x) + \mu\varphi(y) = \lambda 0 + \mu 0 = 0$.

Définition (Somme directe)**.** La somme directe de deux espaces vectoriels V et W est l'espace $V \oplus W$ dont les vecteurs sont des couples d'un vecteur du premier espace et d'un vecteur du second.

En dimension finie, on a $\dim(V \oplus W) = \dim V + \dim W$ et l'on peut donc trouver une base de $V \oplus W$ dans laquelle les premières composantes sont celles du premier et les dernières, celles du second. On décrit cette propriété en nommant ces deux espaces *supplémentaires*.

De manière générale, il n'y a aucune garantie d'unicité, ni même d'existence d'une pseudo-inverse G^+, mais en dimension finie, il en existe toujours *au moins* une. Elle peut se calculer notamment par *élimination de Gauß-Jordan*[13] ou méthode du pivot.

Comme une matrice génératrice est de rang maximal, sous nos hypothèses (ses dimensions sont $n \times k$ avec $k \leq n$), on peut écrire

$$G^+ G = \mathbb{1}_k \tag{VII.1}$$

où $\mathbb{1}_k$ est l'identité dans l'espace des symboles originels. G^+ est donc une inverse « à gauche ». Elle va nous permettre ainsi de décoder les messages : on a en effet $y \in K \Rightarrow G^+ y = x$ avec $Gx = y$. De cette façon, tout mot reçu, même erroné, pourrait être décodé, c'est-à-dire associé à un et un seul symbole, mais rien ne permet d'affirmer que ce serait dans le respect du principe de maximum de vraisemblance !

Exemple. Pour illustrer cela, considérons le code binaire à répétition de longueur 3. C'est un code linéaire binaire $[3,1,3]_2$ qui permet de coder 2 mots (les bits 0 et 1) et de corriger 1 erreur (par majorité).

Sa matrice génératrice est

$$G = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Pour calculer une pseudo-inverse $G^+ = [g_1 g_2 g_3]$, par (VII.1), on doit satisfaire $g_1 + g_2 + g_3 = 1 \pmod 2$. Il y a donc 4 solutions :

$$G^+ \in \Big\{ [100], [010], [001], [111] \Big\}$$

qui décodent toutes correctement les 2 mots du code : $G^+ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = [0]$ et $G^+ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = [1]$, mais aucune ne correspond au décodage par majorité des mots en dehors du code.

VII.2.2 Syndrome et matrice de contrôle

Pour toute pseudo-inverse G^+ de la matrice génératrice G d'un code linéaire, son noyau de $\ker G^+$ est, par définition, un sous-espace de dimension $(n-k)$ des messages décodés comme nuls. Ce sous-espace est celui des *vecteurs d'erreurs*.

On peut également voir G^+ comme une matrice génératrice d'un code dans l'espace dual, celui des fonctions linéaires sur les messages. Elle applique les formes f^τ du dual des symboles (un vecteur-ligne à k composantes) sur les formes $g^\tau = f^\tau G^+$ de l'espace dual des messages. Son image est un code isomorphe au code originel.

Comme toute G^+ est une fonction de décodage (qui ne fait sens que restreinte au domaine $K \subset \mathbb{F}_q^n$), une façon de réaliser une *fonction de détection* d'erreur (*i.e.* de vérifier aisément si $y \in K$) est de l'utiliser pour projeter l'espace des messages sur K ($y \overset{?}{=} G G^+ y$). La matrice $G G^+$ de dimensions $n \times n$ est symétrique et de rang k ; elle réalise donc cette projection, mais il existe autant de telles projections qu'il y a de pseudo-inverses.

13. Des noms de Johann Carl Friedrich Gauß et de Wilhelm Jordan (à ne pas confondre avec l'autre mathématicien Camille Jordan connu pour ses travaux en théorie de la mesure et de l'intégration), bien que cette méthode existât déjà au I^e siècle en Chine !

Pour construire une *fonction de correction* d'erreurs, il nous faut au contraire visualiser les erreurs (*sic!*) et pas les projeter toutes sur 0. On a besoin à cet effet d'une application « *supplémentaire* » de l'espace des mots $\mathbb{F}_q^n$ vers celui des erreurs $\mathbb{F}_q^{n-k}$ qui serait de rang maximal $(n-k)$ et aurait précisément le code K pour noyau. C'est une application définie par ce qu'on appelle une *matrice de contrôle*.

Définition (Matrice de contrôle et code dual)**.** Une matrice de contrôle d'un code linéaire K engendré par G $(n \times k)$ est une matrice H de dimensions $(n-k) \times n$ de rang maximal. Dans les espaces duaux, c'est la matrice génératrice d'un code de même longueur n, mais de dimension $(n-k)$, appelé *code dual* de K. Son image est le sous-espace de toutes les formes linéaires qui s'annulent du K ; c'est ce qu'on appelle le sous-espace orthogonal[14] de K noté $\mathsf{K}^\perp$.

$$\mathsf{H}: \left(\mathbb{F}_q^{n-k}\right)^* \to \left(\mathbb{F}_q^n\right)^* : f^\tau \mapsto g^\tau = f^\tau \mathsf{H} \quad \text{où} \quad \operatorname{im} \mathsf{H} = \mathsf{K}^\perp$$

avec

$$\mathsf{H}\,\mathsf{G} = \mathbb{0}_{(n-k)\times k} \tag{VII.2}$$

Si l'on regarde plutôt la matrice H comme définissant l'application linéaire duale σ de $\mathbb{F}_q^n$ sur $\mathbb{F}_q^{n-k}$, c'est une application des mots vers l'espace vectoriel des erreurs que l'on appelle « *syndromes* » et (VII.2) dit que son noyau dans l'espace des messages est exactement le sous-espace des mots du code : $\ker \sigma = \mathsf{K}$. Comme on est en dimension finie, il y a correspondance biunivoque entre l'espace des syndromes et l'espace orthogonal du code ; il est isomorphe à cette image : $\operatorname{im} \sigma \simeq \mathsf{K}^\perp$.

Définition (Syndrome)**.** L'image d'un mot par l'application définie par une matrice de contrôle est appelée « syndrome » du mot.

$$\sigma: \mathbb{F}_q^n \to \mathbb{F}_q^{n-k} : y \mapsto \sigma(y) = \mathsf{H}y \quad \text{avec} \quad \ker \sigma = \mathsf{K}$$

Ce nom se justifie par le fait que ce vecteur $\sigma(y)$ est une représentation des erreurs contenues dans y : $\sigma(y) = 0 \Leftrightarrow \exists\, x : y = \mathsf{G}x$; un syndrome non nul signifie ainsi que le mot contient des erreurs.

Plus précisément, par linéarité, nous avons la propriété suivante.

Propriété VII.2.3. *Si y est un mot résultant de l'encodage du message x et ayant subi des erreurs $\varepsilon = y - \mathsf{G}x$, alors son syndrome ne dépend que de ces erreurs ($\sigma(y) = \mathsf{H}\,\varepsilon$) et le poids de Hamming $\mathrm{w}_\mathsf{H}(\varepsilon) = \mathrm{d}_\mathsf{H}(\mathsf{G}x, y)$ est le nombre d'erreurs contenues dans y par rapport à x.*

Démonstration. De $y = \mathsf{G}x + \varepsilon$, on a, par linéarité, $\mathsf{H}y = \mathsf{H}\,\mathsf{G}x + \mathsf{H}\,\varepsilon = 0 + \mathsf{H}\,\varepsilon$. ∎

Les « vecteurs d'erreurs » ε peuvent donc être analysés indépendamment du message particulier. Inversement, pour un y donné, tout x peut *a priori* convenir, mais l'on peut choisir un sous-espace particulier — dont les vecteurs doivent être tous linéairement indépendants de K — auquel on fait appartenir les erreurs. Par définition, ils constituent un espace orthogonal $\mathsf{K}^\perp \subset \mathbb{F}_q^n$ associé à ce choix.

Tout comme G^+ n'est pas unique pour une matrice G donnée, il existe plusieurs matrices de contrôle H associées à une matrice génératrice G fixée. Mais, du point de vue de sa structure et de ses propriétés linéaires, un code peut être défini indifféremment par une matrice génératrice G, une de ses inverses G^+, une matrice de contrôle H ou une de ses inverses H^+, même si ces différentes définitions ne conduisent pas à des codes parfaitement identiques ; ils sont simplement duaux ou isomorphes.

Exemple. Pour illustrer, considérons à nouveau le code à répétition $[3,1,3]_2$.

Nous pouvons construire pour lui une matrice de contrôle

$$\mathsf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

On vérifie aisément que le code est son noyau : $\mathsf{H}\begin{bmatrix}0\\0\\0\end{bmatrix} = \mathsf{H}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$.

Même pour une matrice génératrice G fixée, cette matrice de contrôle H n'est pas unique : les 6 matrices obtenues en permutant les colonnes de H entre elles sont également des matrices de contrôle du même code engendré par G.

14. Terme justifié précisément par la relation (VII.2).

VII.2.3 Forme canonique et matrice de parité

Le choix d'une base dans les espaces vectoriels des symboles $\mathbb{F}_q^k$, des messages $\mathbb{F}_q^n$ — donc des mots du code et des erreurs — et des syndromes $\mathbb{F}_q^{n-k}$ influence la forme, la valeur des éléments des matrices G et H, sans modifier les propriétés du code, notamment sa capacité potentielle de détection ou de correction d'erreurs. C'est ce qu'on appelle un *code équivalent*. Un tel changement de base dans un espace vectoriel y est représenté par une matrice carrée inversible par laquelle il faut multiplier les matrices G ou H, à gauche ou à droite selon l'espace dans lequel la base a été modifiée.

Par exemple, un changement de base $x \mapsto x'$ dans l'espace des messages originels est représenté par une matrice B carrée $k \times k$ et inversible telle que $x' = \mathrm{B}\,x$. Dans ce cas, l'encodage devient :

$$y = \mathrm{G}\,x = \mathrm{G}\,\mathrm{B}^{-1}\,x'$$

Le code défini par la matrice génératrice transformée $\mathrm{G}' = \mathrm{G}\,\mathrm{B}^{-1}$ est dit *code équivalent* de celui défini par G.

Par conséquent, on choisit plutôt des formes « sympathiques » pour ces matrices en sélectionnant intelligemment les bases dans les différents espaces.

Comme la matrice génératrice G est, par définition, de rang k, on peut choisir une forme équivalente dans laquelle apparaît une matrice identité $\mathbb{1}_k$ dans les premières lignes de G. La matrice génératrice sous cette forme est dite *canonique* ou *systématique* :

$$\mathrm{G} = \left[\frac{\mathbb{1}_k}{\mathrm{P}}\right]$$

Ainsi, l'encodage $y = \mathrm{G}\,x$ s'écrit

$$y = \left[\frac{\mathbb{1}}{\mathrm{P}}\right] x = \left[\frac{x}{\mathrm{P}x}\right]$$

et les k premiers caractères de chaque mot du code exactement sont ceux du message originel, tandis que les $(n-k)$ suivants constituent la redondance introduite par le code.

Définition (Matrice de parité)**.** Si la matrice génératrice est représentée sous forme canonique, la sous-matrice P de dimensions $(n-k) \times k$ est la matrice de parité du code. Elle définit les valeurs $\mathrm{P}\,x$ des $(n-k)$ caractères de redondance en suffixe de chaque mot du code.

Pour les mêmes raisons, on peut choisir une forme canonique associée pour la matrice de contrôle en choisissant d'appliquer les caractères de redondance, plus précisément les $(n-k)$ derniers vecteurs de la base canonique de $\mathbb{F}_q^n$ sur la base canonique de l'espace des syndromes. Comme le code est le noyau de H, on doit avoir $\mathrm{H}\,\mathrm{G} = \mathbb{0}$, et si G est sous sa forme canonique, la forme canonique suivante s'impose pour la matrice de contrôle du code :

$$\mathrm{H} = \left[-\mathrm{P} \,\middle|\, \mathbb{1}_{n-k}\right]$$

On voit ainsi immédiatement que le syndrome est la différence de la parité calculée sur les caractères du message originel supposé (*i.e.* les k premiers caractères du mot), c'est-à-dire $\mathrm{P}\,x$, et la valeur de cette redondance contenue dans le mot (*i.e.* ses $(n-k)$ derniers caractères) :

$$\sigma(y) = \left[-\mathrm{P} \,\middle|\, \mathbb{1}\right] \left[\frac{x'}{(\mathrm{P}x)'}\right] = -\mathrm{P}\,x' + (\mathrm{P}x)'$$

qui s'annule si les deux demi-mots sont inaltérés (altération que nous avons notée « ′ »).

Une matrice de parité P (ayant $(n-k) \times k$ coefficients) est donc une façon économique de définir un code linéaire et son couple de matrices génératrice et de contrôle G et H sous forme canonique. Notons qu'il n'y a aucune contrainte sur cette matrice (sur son rang, par exemple, qui pourrait même être nul) et que pour un code donné, P n'est pas unique.

Exemple. Le code à répétition de longueur 3 que nous avons présenté était déjà sous une forme canonique. Sa matrice de parité est simplement $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$.

Pour illustrer plus largement, considérons maintenant le code linéaire binaire suivant (nous verrons qu'il s'agit du code de Hamming étendu $[8,4,4]_2$).

Ce code est défini, par exemple, par la matrice de parité

$$\mathrm{P} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Les matrices génératrice et de contrôle canoniques associées sont donc

$$\mathrm{G} = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right]$$

$$\mathrm{H} = \left[\begin{array}{cccc|cccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

VII.2.4 Décodage par syndrome

La matrice de contrôle définit essentiellement une fonction de détection d'erreur. Mais nous pouvons utiliser les syndromes pour décoder un message en corrigeant certaines erreurs si le code est correctement choisi et défini. En effet, la distance et le poids de Hamming ont des propriétés intéressantes dans les espaces vectoriels, donc, plus spécifiquement pour les codes linéaires.

Propriété VII.2.4. *La distance de Hamming dans un espace vectoriel* V *satisfait*

$$\forall\, a, b, c \in \mathrm{V} : \mathrm{d}(a, b) = \mathrm{d}(a + c, b + c)$$

Démonstration. On examine ces vecteurs composante par composante : $a_i \stackrel{?}{=} b_i$ est équivalent à $a_i + c_i \stackrel{?}{=} b_i + c_i, \forall\, c_i$, puisqu'on est dans un groupe additif. ∎

Propriété VII.2.5. *Le poids de Hamming dans un espace vectoriel* V *sur un corps* F *satisfait*

$$\forall\, \lambda \neq 0 \in \mathrm{F},\ \forall\, a \in \mathrm{V} : \mathrm{w}(\lambda a) = \mathrm{w}(a)$$
$$\forall\, a, b \in \mathrm{V} : \mathrm{w}(a + b) \leq \mathrm{w}(a) + \mathrm{w}(b)$$

Démonstration. Si l'on multiplie un vecteur par un scalaire non nul, ses composantes nulles restent nulles et ses composantes non nulles le restent aussi, puisqu'un corps n'a pas de diviseurs de zéro.

La seconde propriété découle directement de la propriété VI.1.6. ∎

Propriété VII.2.6. *La distance minimale d'un code linéaire est le poids minimal du code.*

Démonstration. Par (VI.2), on sait que $\mathrm{d}(a, b) = \mathrm{w}(a - b)$. Par conséquent, si a et b sont deux mots du code à distance minimale, le mot $a - b$ est également dans le code, puisque c'est un sous-espace linéaire, donc le poids minimal est au plus cette distance. Inversement, considérons c de pois minimal. On a $\mathrm{w}(c) = \mathrm{d}(c, 0)$, donc la distance minimale est au plus ce poids. ∎

On a une méthode directe pour déterminer ce poids minimal à partir de la matrice de contrôle.

Propriété VII.2.7. *Dans un code linéaire de matrice de contrôle* H*, la distance minimale du code est le nombre minimum de colonnes dépendantes de* H*.*

Démonstration. Soit c un mot du code de poids minimal d. Donc $\mathrm{H}\,c = 0$, ce qui est une combinaison linéaire nulle d'exactement d colonnes de H, celles des coefficients non nuls de c. Donc H contient d colonnes dépendantes ; son nombre minimal est au plus le poids du code.

Inversement soit un ensemble d'indices minimal tel que les colonnes correspondantes de H sont dépendantes. On construit le vecteur c avec les facteurs de la combinaison linéaire à ces positions et des zéros ailleurs. On a $\mathrm{H}\,c = 0$, donc c est un mot du code, donc de poids au moins d, et le nombre minimal est ainsi au moins le poids du code. ∎

Reprenant la propriété VII.2.3, pour étudier la capacité de correction d'un code linéaire K, il nous suffit d'examiner les images altérées du mot nul, ce que nous appellerons les « *erreurs à l'origine* » ou erreurs en 0. En effet, on a directement

Lemme VII.2.8. *Dans un code linéaire de matrice génératrice* G*, si y se décode en* Gx*, alors*

$$\mathrm{d}(y, \mathrm{G}x) = \mathrm{w}(y - \mathrm{G}x) = \mathrm{w}(\varepsilon) = \mathrm{d}(\varepsilon, 0)$$

Ainsi, le mot ε se décode en 0 avec le même nombre de corrections que lorsqu'on décode y en Gx.

Propriété VII.2.9. *Dans un code linéaire de matrice génératrice* G*, le décodage $\hat{x}$ du mot y par maximum de vraisemblance satisfait*

$$\mathrm{G}\,\hat{x} = y - \hat{\varepsilon} \quad \textit{avec} \quad \mathrm{w}(\hat{\varepsilon}) = \min_{\varepsilon}\{\mathrm{w}(\varepsilon) \mid \mathrm{H}\,\varepsilon = \mathrm{H}\,y\}$$

Démonstration. Par la propriété VI.1.10, on sait que le décodage par maximum de vraisemblance se fait en minimisant la distance $\mathrm{d}(y, \mathrm{G}x)$. Par le lemme VII.2.8, on a immédiatement le résultat. ∎

Ainsi, si l'on s'intéresse à la correction d'au plus t erreurs, on peut se limiter aux vecteurs d'erreur contenus dans la boule centrée à l'origine et de rayon t : $\mathcal{B}(0, t)$. Si t respecte (VI.6), on a la propriété suivante.

Propriété VII.2.10. *Dans un code linéaire de matrice génératrice* G*, de matrice de contrôle* H *et de distance minimale d, à chaque syndrome correspond au plus un vecteur dans la boule $\mathcal{B}(0, t)$ si $t < \frac{d}{2}$.*

Démonstration. Cela se démontre par l'absurde. S'il existait ε et ε' ayant le même syndrome et de poids $< \frac{d}{2}$, le mot $v = \varepsilon - \varepsilon'$ appartiendrait au code, puisque $\mathrm{H}\,v = \mathrm{H}\,\varepsilon - \mathrm{H}\,\varepsilon' = 0$, par hypothèse. Et $\mathrm{w}(\varepsilon - \varepsilon') \leq \mathrm{w}(\varepsilon) + \mathrm{w}(\varepsilon) < d$, or d est la distance minimale. ∎

Par contre, ce n'est plus le cas pour des mots de poids supérieur ; un même syndrome est potentiellement associé à plusieurs mots (exactement à q^k mots si l'on va jusqu'à la distance n, c'est-à-dire tout l'espace des mots). Inversement, il est possible que certains syndromes ne soient pas dans l'image de la boule $\mathrm{H}\,\mathcal{B}\left(0, \left\lfloor \frac{d-1}{2} \right\rfloor\right)$; c'est le cas si le code détecte plus d'erreurs qu'il ne peut en corriger.

On note que, *pour les codes linéaires*, la borne de Hamming est un corollaire immédiat de cette propriété.

Ces différents résultats conduisent à la méthode de *décodage par syndrome* qui s'appuie sur le théorème suivant.

Théorème VII.2.11 (Décodage par syndrome)**.** *Pour un code linéaire de matrice génératrice* G*, de matrice de contrôle* H *et de distance minimale d, un mot y altéré par au plus $\left\lfloor \frac{d-1}{2} \right\rfloor$ erreurs se corrige en $\mathrm{G}x = y - \varepsilon$ où ε est l'unique erreur à l'origine dans la boule $\mathcal{B}\left(0, \left\lfloor \frac{d-1}{2} \right\rfloor\right)$ correspondant au syndrome $\sigma(y) = \mathrm{H}y = \mathrm{H}\varepsilon$.*

Démonstration. C'est une conséquence immédiate des résultats précédents : s'il existe x tel que $\mathrm{d}(\mathrm{G}x, y) < \frac{d}{2}$, le syndrome $\sigma(y)$ est associé à un seul vecteur d'erreur à l'origine. ∎

Concrètement, si G est sous forme canonique, ce décodage de y en un mot du code Gx permet de retrouver immédiatement x, puisque ce sont ses premières composantes.

Mais le lien entre les syndromes et les erreurs à l'origine n'est, de manière générale, pas simple à formaliser. En pratique, on utilise une table, éventuellement avec une représentation rendant la recherche efficace (une table triée ou une table de hachage, par exemple).

On peut encore limiter la taille de cette table en utilisant la linéarité du code :

Propriété VII.2.12. *Dans un code linéaire de matrice génératrice* G*, de matrice de contrôle* H *et de distance minimale d, un mot y altéré par au plus* $\left\lfloor \frac{d-1}{2} \right\rfloor$ *erreurs, il existe exactement un vecteur d'erreur à l'origine de même poids* ε_0 *correspondant à un syndrome* $\sigma_0 = \mathrm{H}\,\varepsilon_0$ *dont la première composante non nulle vaut 1 et un nombre scalaire* λ *non nul tels que*

$$y = \mathrm{G}\,x + \lambda\,\varepsilon_0$$

Démonstration. Considérons l'erreur à l'origine ε non nulle telle que $\sigma(y) = \mathrm{H}y = \mathrm{H}\,\varepsilon$ et soit λ la première composante non nulle de $\sigma(y)$. Définissons $\varepsilon_0 = \lambda^{-1}\varepsilon$, et on a bien $\mathrm{w}(\lambda^{-1}\varepsilon) = \mathrm{w}(\varepsilon)$. ∎

On peut donc se limiter aux *directions* (*i.e.* les vecteurs à un facteur multiplicatif près ou *l'hyperplan à l'infini*), tant pour les erreurs à l'origine que dans l'espace des syndromes ; cela fait gagner une dimension.

Définition (Table de décodage)**.** Une table de décodage est une liste de couples matérialisant l'application injective $\sigma \mapsto \varepsilon$, inverse de $\sigma = \mathrm{H}\varepsilon$ où $0 < \mathrm{w}(\varepsilon) \leq s = \left\lfloor \frac{d-1}{2} \right\rfloor$ et où les syndromes sont représentés à un facteur près. Elle est de taille $\frac{\mathrm{B}_s - 1}{q-1}$ (*cf.* (VI.3)).

Bien sûr, il n'est plus possible d'utiliser cette technique lorsque les nombres de redondances et de corrections possibles deviennent trop élevés. On cherchera donc plutôt des codes pour lesquels il existe un *algorithme de décodage*, c'est-à-dire une méthode de calcul de cette application $\sigma \mapsto \varepsilon$.

Exemple. Reprenons le code $[8, 4, 4]_2$ vu précédemment.

Il peut corriger au plus $\left\lfloor \frac{3}{2} \right\rfloor = 1$ erreur. Il y a $\frac{\mathrm{B}_1 - 1}{1} = 8$ mots d'erreur à l'origine à tabuler. Ce sont les images de H e_i où les e_i sont les 8 vecteurs de base de l'espace des mots, c'est-à-dire exactement H ! Si on la trie selon les syndromes, on obtient le tableau VII.1.

TABLEAU VII.1 – Décodage par syndrome de $[8, 4, 4]_2$

Syndrome σ_0^τ	Erreur ε_0^τ
0 0 0 1	0 0 0 0 0 0 0 1
0 0 1 0	0 0 0 0 0 0 1 0
0 1 0 0	0 0 0 0 0 1 0 0
0 1 1 1	0 0 1 0 0 0 0 0
1 0 0 0	0 0 0 0 1 0 0 0
1 0 1 1	0 1 0 0 0 0 0 0
1 1 0 1	1 0 0 0 0 0 0 0
1 1 1 1	0 0 0 1 0 0 0 0

VII.2.5 Identité de MacWilliams

Pour étudier et concevoir des codes linéaires facilement décodables, il faut donc connaître la structure des poids des erreurs. Ceux-ci sont énumérables à l'aide de ce qu'on appelle l'*identité de MacWilliams* établie par la mathématicienne anglaise Florence Jessie MacWilliams (voir [23] et [24]).

Définition (Polynôme énumérateur des poids)**.** Pour tout code en bloc K de longueur n, le polynôme énumérateur des poids de K est un polynôme homogène de degré n à deux indéterminées [15] $W_K(X,Y)$ à coefficients entiers positifs (donc $W_K \in \mathbb{Z}[X,Y]$) pour lequel le coefficient de $X^i Y^{n-i}$ est le nombre de mots de poids i dans K.

$$W_K(X,Y) = \sum_{i=0}^{n} w_i X^i Y^{n-i} \quad \text{où} \quad w_i = \Big|\big\{c \in K \,\big|\, w(c) = i\big\}\Big| \tag{VII.3}$$

En particulier, pour tout code $W_K(0,0) = 0$, $W_K(0,1) = 1$ et $W_K(1,1) = |K|$.

Par exemple (même si ce cas est inintéressant en pratique), dans un code sans redondance (*i.e.* $n = k$), les w_i sont exactement le nombre de mots à distance i de l'origine, c'est-à-dire

$$w_i = \binom{n}{i}(q-1)^i$$

On trouve donc

$$W_K(X,Y) = \Big(Y + (q-1)X\Big)^n$$

On définit de même le *polynôme énumérateur des distances*:

$$A_K(X,Y) = \sum_{i=0}^{n} A_i X^i Y^{n-i} \quad \text{où} \quad A_i = \frac{1}{|K|}\Big|\big\{(c_1,c_2) \in K \times K \,\big|\, d(c_1,c_2) = i\big\}\Big|$$

Propriété VII.2.13. *Si* K *est un code linéaire, alors* $A_K(X,Y) = W_K(X,Y)$*.*

Démonstration. Si K est un code linéaire, $d(c_1,c_2) = d(c_1+c, c_2+c) = w(c_1-c_2)$ pour tout mot c du code. Le même mot $c_1 - c_2$ apparaît don $q^k = |K|$ fois. ∎

C'est cette propriété qui justifie la normalisation (la division par $|K|$) dans la définition des coefficients A_i.

L'identité de MacWilliam exprime le lien entre le polynôme énumérateur d'un code linéaire et celui de son code dual.

Théorème VII.2.14 (Identité de MacWilliams, 1963)**.** *Si* K *est un code linéaire de dimension k dans l'espace vectoriel* $\mathbb{F}_q^n$ *de polynôme énumérateur des poids* W*, alors le polynôme énumérateur des poids* $W^\perp$ *du code dual* $K^\perp$ *vaut*

$$W^\perp(X,Y) = \frac{1}{q^k}\, W\Big(Y - X,\ Y + (q-1)X\Big) \tag{VII.4}$$

La démonstration originelle se trouve dans [23]. Une démonstration dans le cas binaire est également donnée par [2, p.130].

On remarque que (VII.4) est bien une relation involutive:

$$\begin{aligned}\Big(W^\perp\Big)^\perp(X,Y) &= \frac{1}{q^{n-k}}\,\frac{1}{q^k}\, W\Big([Y + (q-1)X] - [Y - X],\ [Y + (q-1)X] + (q-1)\,[Y - X]\Big)\\ &= \frac{1}{q^n}\, W(qX, qY)\\ &= W(X,Y)\end{aligned}$$

puisque le polynôme est homogène de degré n.

Comme illustration, reprenons le code sans redondance ($n = k$).

$$W^\perp(X,Y) = \frac{1}{q^n}\Big(\big(Y + (q-1)X\big) + (q-1)\big(Y-X\big)\Big)^n = \frac{q^n}{q^n}\,(Y)^n = Y^n$$

Le code dual contient effectivement un seul mot: le mot nul de poids nul.

15. Notons que certains auteurs le définissent comme le polynôme univarié $W_K(X) = W_K(X,1)$.

Chapitre VIII

Quelques familles de codes linéaires

En toute généralité, le décodage par syndrome n'est pas efficace dès que le code devient grand. La théorie des codes vise donc à déterminer des codes ayant des propriétés intéressantes :

- Efficience : corriger un maximum d'erreurs pour une redondance fixée (*i.e.* être un code MDS, ou presque) ;
- Efficacité : corriger toutes les erreurs détectables (*i.e.* être un code parfait, ou presque) ;
- Performance : corriger et décoder rapidement (*i.e.* avoir un bon algorithme de calcul du vecteur d'erreur).

Le code idéal n'est qu'une utopie, mais plusieurs familles de codes ont été découvertes qui présentent ces propriétés à des degrés divers. Nous allons en examiner quelques-uns ci-dessous.

VIII.1 Codes polynomiaux

Nous avons vu que les mots de longueur n sur un alphabet de q caractères peuvent être vus non seulement comme les éléments d'un espace vectoriel de dimension n sur le corps fini $\mathbb{F}_q$, mais également comme l'ensemble des polynômes sur ce même corps de degré inférieur à n. On peut donc, dans certains cas, traduire l'encodage selon la matrice génératrice G d'un code linéaire et le calcul du syndrome par sa matrice de contrôle H sous forme d'expressions d'algèbre polynomiale.

Ainsi, un message à coder x de k caractères est vu comme le polynôme[1] en la variable Z $X(Z) = \sum_{i=0}^{k-1} x_i Z^i$ de degré inférieur à k, un mot à décoder (à corriger) y de n caractères est un polynôme $Y(Z) = \sum_{j=0}^{n-1} y_j Z^j$ de degré inférieur à n, un syndrome σ, un polynôme $\Sigma(Z) = \sum_{l=0}^{k-n-1} \sigma_l Z^l$ de degré inférieur à $(n-k)$, etc.

Définition (Code polynomial)**.** Un code polynomial de dimension k et de longueur n sur un corps F est l'ensemble des multiples de degré inférieur à n d'un polynôme $G(Z)$ dans l'anneau de polynômes $F[Z]$. Ce polynôme $G(Z)$, de degré $(n-k)$, est appelé polynôme générateur du code.

Comme $F[Z]$ est un anneau, certes infini, les opérations y préservent (*i.e.* se distribuent, sont associatives…) l'addition et la multiplication par un scalaire (un élément du corps) ; ce sont donc des applications linéaires sur les vecteurs de coefficients. Les codes polynomiaux constituent donc un cas particulier de codes linéaires.

Dans un code polynomial, la fonction d'encodage s'exprime sous forme d'arithmétique de polynômes et veille, d'une manière ou d'une autre, à produire toujours un multiple de $G(Z)$. Inversement, le calcul du syndrome est un simple résidu : $\Sigma(Z) = Y(Z) \bmod G(Z)$.

1. Pour certains auteurs ou dans certaines circonstances, on écrit les polynômes dans l'autre sens : $X'(Z) = Z^k X(\frac{1}{Z}) = \sum_{i=0}^{k-1} x_{k-i} Z^i$. Cela ne change rien aux propriétés des codes engendrés.

VIII.1.1 Code polynomial canonique

De manière naturelle, on peut définir la fonction d'encodage par $X(Z) \mapsto X(Z)G(Z)$, ce qui produit bien un ensemble de polynômes de degré inférieur à n (puisque $\deg X < k$ et $\deg G = n - k$) multiples de $G(Z)$. Mais le décodage nécessite alors d'effectuer une division de polynômes, ce qui n'est pas très efficace ($O(nk)$ opérations, sauf optimisations efficaces seulement pour des degrés très élevés).

Par conséquent, on utilise plutôt une *forme canonique* similaire à celle des codes linéaires, qui place directement les coefficients du « message-polynôme » $X(Z)$ dans ceux du code. La fonction d'encodage canonique est ainsi définie par

$$K : X(Z) \mapsto K_x(Z) = Z^{n-k} X(Z) - \left[Z^{n-k} X(Z)\right] \bmod G(Z) \tag{VIII.1}$$

On voit immédiatement que $K_x(Z) \bmod G(Z) = 0$, donc qu'il s'agit bien d'un multiple de $G(Z)$, qu'il est de degré inférieur à n et que ces coefficients de degrés $\geq (n - k)$ sont ceux de $X(Z)$.

Propriété VIII.1.1. *Le polynôme* $P_x(Z) = -\left[Z^{n-k} X(Z)\right] \bmod G(Z)$ *de degré inférieur à* $(n - k)$ *correspond aux bits de redondance du code polynomial.*

Ainsi, chaque multiple de $G(z)$ de degré inférieur à n est bien un mot du code.

Sous cette forme, le calcul du syndrome d'un mot y peut être optimisé en calculant le résidu d'un polynôme de plus petit degré, celui de ses k coefficients dominants $Y(Z) \div Z^{n-k}$ (où « $\div$ » représente la division entière) et en ajoutant ensuite celui de ses coefficients des degrés inférieurs (formellement $Y(Z) \bmod Z^{n-k}$).

Et toutes ses opérations peuvent facilement se transcrire sous forme de matrice génératrice et de matrice de contrôle. Mais, à nouveau, il n'y a pas de méthode simple de correction des erreurs si l'on n'impose pas une forme spéciale au polynôme générateur. Sinon, on se ramène donc à un simple décodage par syndrome.

Notons que cette représentation canonique concerne tous les codes polynomiaux ; elle peut s'appliquer à tous les cas particuliers décrits dans la suite de ce document. Il suffit pour cela d'y effectuer préalablement la transformation

$$X(Z) \mapsto X'(Z) = Z^{n-k} X(Z)$$

Exemple. Considérons le code polynomial (canonique) binaire de dimension 2 (4 mots) et de longueur 5 engendré par $G(Z) = Z^3 + Z + 1$, donc avec 3 bits de redondance (il faut au moins cela pour espérer pouvoir corriger une erreur).

$$K = \left\{0,\ 1+Z+Z^3,\ Z+Z^2+Z^4,\ 1+Z^2+Z^3+Z^4\right\}$$

La matrice génératrice est celle dont les colonnes sont les codes, les images des 2 vecteurs de base $\{1, Z\}$; elle contient donc ici la matrice de parité dans ses premières lignes :

$$G = \left[\begin{array}{cc} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ \hline 1 & 0 \\ 0 & 1 \end{array}\right]$$

ce qui correspond à la matrice de contrôle

$$H = \left[\begin{array}{cc|ccc} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{array}\right]$$

VIII.1.2 Codes cycliques

Les codes cycliques ou codes « **CRC »** (pour *Cyclic Redundancy Check*) sont, comme nous le verrons, un cas particulier de codes polynomiaux qui introduit une structure d'anneau sur l'espace des mots afin de permettre une expression algébrique de la correction d'erreurs. On attribue la première description de leur usage pour la *détection* d'erreur à William Wesley Peterson en 1961 (*cf.* [29]).

Définition (Code cyclique)**.** Un code cyclique de longueur n sur le corps fini $\mathbb{F}_q$ est un idéal dans l'anneau quotient de polynômes $\mathbb{F}_q[Z]/\langle Z^n-1\rangle$.

Les opérations (addition et multiplication des polynômes) se font donc « modulo (Z^n-1) » ; le code constitue un idéal, il est donc ici fermé également pour la multiplication par tout polynôme, ce qui n'est pas le cas pour un code polynomial en général.

Le terme « cyclique » vient du fait que multiplier un élément de cet anneau par le monôme Z revient à effectuer un *décalage circulaire* de ses coefficients :

$$Z \cdot \sum_{j=0}^{n-1} p_j Z^j \bmod (Z^n-1) = \sum_{j=1}^{n-1} p_{j-1} Z^j + p_{n-1} Z^0$$

et de la propriété suivante.

Propriété VIII.1.2. *Un code linéaire est cyclique si et seulement s'il contient tous les décalages circulaires de ses mots.*

Démonstration. Si c'est un idéal, pour tout mot du code (*i.e.* un polynôme P(Z)), Z P(Z) doit également être dans le code et l'on vient de voir que ce produit correspond à un décalage circulaire.

Inversement, si tous les décalages sont dans le code, comme il est supposé linéaire, donc contenir les sommes de ses éléments et leurs produits par un scalaire, on peut multiplier par tout polynôme en sommant les produits par chaque monôme, c'est-à-dire par le produit d'un de ses mots décalés par un scalaire. ∎

On a également le résultat fondamental suivant.

Théorème VIII.1.3 (Propriétés essentielles des codes cycliques)**.** *Tout code cyclique* K *satisfait les trois propriétés :*

- *C'est un idéal principal, c'est-à-dire qu'il existe toujours un polynôme générateur* G(Z) *dont les multiples constituent le code ;*
- *Ce polynôme générateur est un diviseur du polynôme* $Z^n - 1$ *;*
- *Posons* $\deg G = n - k$ *; ce polynôme a* $(n - k + 1)$ *coefficients. k est alors la dimension du code et une base de* K *(vu comme espace vectoriel) est* $\left\{Z^i G(Z) \mid i = 0 \dots (k-1)\right\}$.

Démonstration. Soit G(Z) un polynôme de plus petit degré dans K et soit P(Z) un mot du code. La division euclidienne $P(Z) = Q(Z)G(Z) + R(Z)$ où $\deg R < \deg G$ implique $R \equiv 0$, parce que $P(Z) - Q(Z)G(Z)$ est dans K (c'est un idéal) et nous avons supposé deg G minimal dans K.

En utilisant le même argument, on écrit $Z^n - 1 = Q(Z)G(Z) + R(Z)$ (dans l'anneau $\mathbb{F}_q[Z]$), donc $R(Z) = -Q(Z)G(Z)$ (dans l'anneau $\mathbb{F}_q[Z]/\langle Z^n-1\rangle$, cette fois), c'est donc un multiple de G(Z) de degré inférieur, donc il est nul.

Enfin, on sait que les monômes $Z^i G(Z)$ sont tous dans K et, pour tout mot du code P(Z), on a $P(Z) = Q(Z)G(Z)$, c'est-à-dire une combinaison linéaire de ces monômes ; c'est donc une famille génératrice. Comme $\deg G = n - k$, vu l'addition des degrés (on travaille sur un corps), on peut se limiter aux puissances inférieures à k, parce que l'on travaille modulo $Z^n - 1$. Cette famille est libre, car s'il existait une combinaison linéaire nulle, en la divisant par G(Z), ce serait un polynôme de degré inférieur à k qui devrait être nul dans l'anneau. ∎

Ce résultat permet donc d'affirmer que les codes cycliques sont bien des codes polynomiaux donc aussi des codes linéaires ; la base ci-dessus constitue les colonnes d'une matrice génératrice du code qui a ainsi une forme très particulière.

$$G = \begin{bmatrix} g_0 & 0 & & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ g_{n-k} & \cdots & & \ddots & 0 \\ 0 & \ddots & & & g_0 \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & g_{n-k} & g_{n-k-1} \\ 0 & \cdots & & 0 & g_{n-k} \end{bmatrix}$$

Corollaire VIII.1.4. *Tout diviseur de* $Z^n - 1$ *engendre un code cyclique.*

Démonstration. Soit $G(Z) \,|\, (Z^n - 1)$. Considérons dans l'anneau l'idéal $\langle G(Z) \rangle$. C'est un code cyclique, par définition. ∎

L'étude des codes cycliques de longueur n revient donc à celle des diviseurs de $Z^n - 1$ dans $\mathbb{F}_q[Z]$. Notons que, comme c'est un corps, on peut se limiter aux polynômes moniques, c'est-à-dire dont le coefficient dominant est l'unité ($g_{n-k} = 1$).

Propriété VIII.1.5 (Polynôme de contrôle)**.** *Pour un code cyclique engendré par le polynôme* $G(Z)$*, le polynôme de contrôle est*

$$H(Z) = \frac{Z^n - 1}{G(Z)} \tag{VIII.2}$$

Démonstration.
On a trivialement $\deg H = k$ et $H(Z)\,G(Z) = 0$ dans $\mathbb{F}_q[Z]/\langle Z^n - 1 \rangle$. Donc $P(Z) \in K \Rightarrow H(Z)\,P(Z) = 0$.
Inversement, $H(Z)\,P(Z) = 0$ veut dire $(Z^n - 1) \,|\, H(Z)\,P(Z)$ (dans $\mathbb{F}_q[Z]$). Si l'on écrit $P(Z) = Q(Z)G(Z) + R(Z)$ avec $\deg R < (n-k)$, on a $\deg HR < k + (n-k) = n$, donc $R \equiv 0$. ∎

Ainsi, le polynôme $H(Z)$ engendre également un code cyclique, qui est le code dual de celui engendré par $G(Z)$.

Propriété VIII.1.6. *Une matrice de contrôle associée au polynôme de contrôle* $H(Z)$ *est*

$$H = \begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & h_k & \ddots & & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & & \cdots & h_k & \cdots & h_1 & h_0 \end{bmatrix}$$

Démonstration. En effet, calculons le produit matriciel HG. En position $(i, j), i = 0 \ldots (n-k-1), j = 0 \ldots (k-1)$, c'est l'expression correspondant au coefficient du monôme de degré $(k+i-j)$ du produit polynomial $H(Z)\,G(Z)$. Or ils sont tous nuls, puisque ce produit vaut $Z^n - 1$ et que $0 < k+i-j < n$. Donc $HG = 0$. De plus, comme par (VIII.2) h_k est non nul, on a une sous-matrice triangulaire de diagonale non nulle, donc le rang de H est maximal. ∎

Comme on le sait, une matrice de contrôle n'est pas univoquement déterminée pour une matrice génératrice donnée. Cette matrice H ne calcule pas le même syndrome que le calcul polynomial du résidu par $G(Z)$.

a) Encodage et syndrome : registres à décalage

Vu le théorème VIII.1.3, pour encoder un message-polynôme, on le multiplie simplement par le polynôme générateur : $K_x(Z) = X(Z)\,G(Z) \bmod (Z^n - 1)$. La structure « cyclique » des opérations dans l'anneau $\mathbb{F}_q[Z]/\langle Z^n - 1\rangle$ permet de les effectuer à l'aide de *registres à décalage*. Nous allons décrire ici la multiplication par le polynôme générateur.

Observons que, comme le polynôme G(Z) est de degré $(n-k)$ et divise $Z^n - 1$, ses coefficients g_{n-k} et g_0 sont non nuls. On utilise un registre à décalage (*Shift Register*) à $(n-k)$ cellules $\{c_i \mid i = 0 \dots (n-k-1)\}$ où, à chaque étape (clic d'horloge), $c_{i-1} \mapsto c_i$ et c_0 reçoit une donnée. En entrée, il reçoit ainsi le message (le caractère x_t au temps t), suivi de $(n-k)$ caractères nuls (ce qui revient à écrire le message-polynôme avec n coefficients, ceux des degrés $\geq k$ étant nuls), en sortie, il produit $y_t = g_0\,x_t + \sum_{i=1}^{n-k} g_i\,c_{i-1}$. Si les cellules du registre sont initialisées à zéro, la sortie en n étapes est bien le mot codé.

De même, le calcul du syndrome — un calcul de résidu — peut se faire à l'aide d'un registre à décalage de même taille (à $(n-k)$ cellules) à rétroaction linéaire (LFSR ou *Linear-Feedback Shift Register*) : sa sortie, le contenu de la cellule c_{n-k-1}, est réinjectée dans les autres cellules combinée avec le contenu de la cellule précédente, tandis que l'entée est constituée des caractères du mot *dans l'ordre opposé* (le mot-polynôme $Z^n\,Y(1/Z)$). Ainsi, à l'étape t, si l'entrée est y_{n-t-1}, $c_0 = y_{n-t-1} - g_0\,g_{n-k}\,c_{n-k-1}$ et, pour $i > 0$, $c_i = c_{i-1} - g_i\,g_{n-k}\,c_{n-k-1}$. Si les cellules du registre sont initialisées à zéro, après n étapes, le contenu des cellules est le syndrome de la donnée.

Nous avons présenté ces deux calculs à partir du polynôme générateur, ce qui est efficace si $n-k \leq k$. Par contre, dans le cas inverse, c'est-à-dire s'il y a plus de caractères de redondance que dans le message, on peut travailler de manière duale à partir du polynôme de contrôle et avec des registres à k cellules.

Ce mode de calcul par registres à décalage fait que les codes cycliques sont particulièrement prisés dans les implantations matérielles ou microcodées (en *firmware*), en particulier dans les matériels de transmission (Bluetooth ou Ethernet, par exemple) ou de stockage (CD ou DVD, par exemple). Ainsi, de nombreux protocoles, dont HDLC ou ISO/IEC/IEEE 8802-3 (802.3), utilisent le code CRC-32, un code cyclique binaire engendré par un polynôme de degré 32.

Exemple. Considérons le code cyclique binaire de dimension 4 (16 mots) et de longueur 7 engendré par $G(Z) = 1+Z+Z^3$ (3 bits de redondance). On a bien $G(Z) \mid (Z^7+1)$ et le quotient vaut $H(Z) = 1+Z+Z^2+Z^4$.

Le message $1 + Z^2$ est encodé en $1 + Z + Z^2 + Z^5$. Si on utilise un registre à décalage, cela suit les étapes du tableau VIII.1.

TABLEAU VIII.1 – Encodage par registre à décalage

t	x_t	c_0	c_1	c_2	y_t
	1	1	0	1	
0	1	0	0	0	1
1	0	1	0	0	1
2	1	0	1	0	1
3	0	1	0	1	0
4	(0)	0	1	0	0
5	(0)	0	0	1	1
6	(0)	0	0	0	0

Si l'on reçoit le mot $1 + Z + Z^2 + Z^3 + Z^5$, son syndrome $(1 + Z)$ se calcule de manière similaire (voir tableau VIII.2).

b) Correction : algorithme de Meggitt

Une méthode de décodage par syndrome plus efficace qu'une consultation de table a été proposée par John E. Meggitt en 1961 ([26]). Elle se fonde sur le théorème suivant.

Théorème VIII.1.7. *Les syndromes des décalages circulaires d'un mot de syndrome* $\Sigma(Z)$ *sont les multiples* $Z^i\,\Sigma(Z) \bmod G(Z)$.

TABLEAU VIII.2 – Calcul du syndrome par registre à décalage

t	y_t	c_0	c_1	c_2
	1	1	0	(1)
6	0	0	0	0
5	1	0	0	0
4	0	1	0	0
3	1	0	1	0
2	1	1	0	1
1	1	0	0	0
0	1	1	0	0
	-	1	1	0

Démonstration. Faisons les calculs dans $\mathbb{F}_q[Z]$. On a $Y(Z) = X(Z)\,G(Z) + \Sigma(Z)$ pour un certain $X(Z)$. Le décalage circulaire (de 1 position) de $Y(Z)$ où $\deg Y < n$ est

$$Z\,Y(Z) \bmod (Z^n - 1) = Z\,Y(Z) - y_{n-1}(Z^n - 1) = Z\,Y(Z) - y_{n-1}\,G(Z)\,H(Z)$$

Donc il a le même syndrome que $Z\,Y(Z)$. De $Z\,Y(Z) = Z\,X(Z)\,G(Z) + Z\,\Sigma(Z)$, on déduit que son syndrome est $Z\,\Sigma(Z) \bmod G(Z) = Z\,\Sigma(Z) - \sigma_{n-k-1} g_{n-k}^{-1} G(Z)$. Et ainsi de suite pour les décalages suivants. ∎

Une conséquence est que nous pouvons limiter la table des syndromes à un sous-ensemble permettant d'engendrer tous ceux correspondant à un nombre d'erreurs potentiellement corrigibles. Ainsi, on considère les syndromes correspondant à au moins une erreur en position $(n-1)$, ce qui réduit la table à B_{t-1} éléments où $t = \left\lfloor \frac{n-k-1}{2} \right\rfloor$.

L'algorithme de Meggitt se fonde sur la technique de calcul du syndrome par registre à décalage. Après n étapes, le contenu du registre est le syndrome de la donnée. Ensuite, on va poursuivre les étapes avec une entrée nulle [2] : le contenu du registre parcourt alors les différents syndromes glissés du théorème VIII.1.7. Si à un moment on rencontre un syndrome « connu » (*i.e.* contenu dans notre table réduite), on doit appliquer la correction d'erreur correspondante (éventuellement multipliée par le facteur scalaire adéquat), mais avec le décalage circulaire correspondant au nombre d'étapes effectuées. En pratique, le décodeur de Maggitt décale simultanément le mot à décoder et le syndrome.

Exemple. Reprenons l'exemple vu ci-dessus ($K = \langle 1 + Z + Z^3 \rangle$). On a $n = 7$, $k = 4$, donc $n - k = 3$. On doit ainsi considérer une table de syndrome de $B_0 = 1$ éléments (ce n'est vraiment pas beaucoup). L'erreur dans le bit le plus significatif est Z^6, donc le seul syndrome intéressant est $Z^6 \bmod (1+Z+Z^3) = 1 + Z^2$. Ceci est illustré dans le tableau VIII.3 où nous repartons du tableau VIII.2 dans l'état où nous l'avions laissé. On voit qu'un doit corriger en position $t = 3$, c'est-à-dire Z^3, étape ou le contenu du registre

TABLEAU VIII.3 – Algorithme de correction de Meggitt

t	y_t	c_0	c_1	c_2	ε_t
	1	1	0	(1)	
6	(0)	1	1	0	0
5	(0)	0	1	1	0
4	(0)	1	1	1	0
3	(0)	1	0	1	1
2	(0)	1	0	0	0
1	(0)	0	1	0	0
0	(0)	0	0	1	0
	-	1	1	0	0

correspond à $1 + Z^2$, ce qui permet bien de retrouver le mot-polynôme $1 + Z + Z^2 + Z^5$.

2. On note que si le syndrome est nul, il le restera par la suite, ce qui correspond bien à une donnée sans erreurs.

VIII.1.3 Codes BCH

Ces codes sont un cas particulier de codes cycliques pour lesquels on peut se passer totalement d'une table de syndromes pour corriger un message altéré et surtout qui garantissent une distance minimale au code engendré. C'est pour cette raison sans doute qu'ils constituent la famille de codes correcteurs qui sont les plus utilisés en pratique (transmissions *via* satellites, CD, DVD, SSD...et même certains codes-barres). Ils doivent leur nom « **BCH »** aux initiales de trois auteurs qui les ont décrits pour la première fois : d'une part Alexis Hocquenghem ([20] en 1959) et, indépendamment d'autre part, Raj Chandra Bose et Dwijendra Kumar Ray-Chaudhuri ([5] en 1960).

La construction des codes BCH s'inspire des résultats de la théorie de Galois concernant les corps finis et leurs extensions. En effet, pour construire un code cyclique, nous travaillons dans un anneau de polynômes sur un corps fini $\mathbb{F}_q$ modulo Z^n-1, où n est la longueur du code. Ce polynôme est ce qu'on appelle un *polynôme cyclotomique*, puisqu'il a pour racines — c'est un truisme — toutes les racines n-ième de l'unité sur $\mathbb{F}_q$; certaines appartiennent déjà au corps ($+1$, ainsi que -1 s'il est distinct et que n est pair, par exemple), d'autres sont dans une *extension du corps*, un corps plus grand (comme i est une racine de Z^4-1 sur $\mathbb{R}$, mais réside dans $\mathbb{C} \supset \mathbb{R}$).

Dans une extension du corps à laquelle toutes ces n racines de l'unité α_j appartiennent (la plus petite telle extension s'appelle le corps de décomposition ou de déploiement, *splitting field* en anglais), ces racines sont les puissances successives de l'une d'entre elles, dite *racine primitive de l'unité* [3] (pensez à i comme racine 4e : ses puissances successives sont i, -1, $-i$ et $+1$, mais cela marche aussi avec les puissances de $-i$ qui sont $-i$, -1, $+i$ et $+1$). Dans ce corps de décomposition, si on note α une telle racine *primitive*, on a $Z^n-1 = \prod_{j=1}^{n}(Z-\alpha^j)$. Et, comme dans un corps extension de $\mathbb{F}_q$ de degré μ, tous les éléments sont racines de $Z^{q^\mu}-Z$, les éléments non nuls sont racines de $Z^{q^\mu-1}-1$. Ceci justifie le choix de la longueur du code : $n = q^\mu - 1$. Un élément α d'ordre $(q^\mu-1)$ est appelé un *élément primitif* de $\mathbb{F}_{q^\mu}$ sur $\mathbb{F}_q$.

Si l'on revient dans le corps initial, le polynôme Z^n-1 se factorise en général, non comme un produit de polynômes du 1er degré, mais en un produit de polynômes moniques de degrés divers *irréductibles* sur ce corps et les racines de l'unité se répartissent entre eux. Chacun de ces polynômes est ce qu'on appelle le *polynôme minimal* de ses racines, en ce sens que c'est un polynôme de degré minimal qui engendre le plus petit corps qui les contient.

Par exemple, sur $\mathbb{F}_3$, $Z^8-1 = (Z-1)(Z+1)(Z^2+1)(Z^2-Z-1)(Z^2+Z-1)$; les deux premiers facteurs correspondent aux racines $+1$ et -1 déjà dans le corps, le suivant à une extension par une racine 4e, tandis que les deux derniers correspondent aux racines 8e primitives regroupées deux par deux en racines *conjuguées* [4].

Maintenant, nous pouvons définir un code BCH (au sens strict) [5] :

Définition (Code BCH)**.** Soit $n = q^\mu - 1$ (pour tout μ fixé) et soit α une racine n-ième primitive sur le corps $\mathbb{F}_q$ (c'est un élément primitif de $\mathbb{F}_{q^\mu}$). Soit $M_i(Z)$ un polynôme minimal (monique) de α^i sur $\mathbb{F}_q$. Alors, le code polynomial engendré par $G(Z) = \operatorname{lcm}\{M_i(Z) \mid i = 1 \dots (\delta-1)\}$ pour $0 < \delta < q^\mu$ est un code BCH de longueur $n = q^\mu - 1$; le paramètre δ est appelé la distance « par construction » (*designed distance*).

Note. La notation « lcm » indique le plus petit commun multiple (*least common multiple*). Il se calcule aisément par l'algorithme d'Euclide de calcul du plus grand commun diviseur (« gcd » ou *greatest common divisor*), grâce à la relation $\gcd(a,b)\cdot\operatorname{lcm}(a,b) = a\cdot b$.

Les deux cas extrêmes sont sans intérêt : $\delta = 1$ correspond à $G(Z) = 1$, le polynôme constant engendrant le code trivial qui correspond à tout l'espace (il n'y a aucune redondance ; il ne peut ni détecter

3. Il existe en général plusieurs racines primitives ; leur choix est sans importance, car elles engendrent toutes des corps isomorphes.

4. Notons que ces trois facteurs engendrent le même corps $\mathbb{F}_9$, même s'il ne s'agit pas d'un polynôme primitif (un polynôme irréductible suffit). Pour s'en convaincre, choisissons i racine de Z^2+1. Ce corps contient donc $1+i$ et son conjugué $1-i$ qui sont racines 8e. En effet, $[Z-(1+i)][Z-(1-i)] = [(Z-1)-i][(Z-1)+i] = (Z-1)^2-i^2 = Z^2+Z+1+1 = Z^2+Z-1$. Bien sûr, i n'est pas une racine primitive, puisqu'elle est d'ordre 4 et que $1+i$ n'est pas une puissance de i.

5. On peut étendre la définition en démarrant non en $M_1(Z)$, mais en $M_c(Z)$ et en choisissant $n = \operatorname{ord}(\alpha)$, l'ordre de la racine choisie (qui divise toujours $q^\mu-1$). Cela correspond aux codes BCH généralisés, mais ne change pas les propriétés essentielles de ces codes. On note que les indices des puissances de α doivent rester contigus.

ni corriger aucune erreur), tandis que $\delta = n$ est, à l'opposé, son dual, le code nul ($G(Z) = 0$ et $K = \{0\}$, il corrige n erreurs).

Remarquons bien le sens conventionnel de la distance par construction δ: n'interviennent que $(\delta-1)$ polynômes minimaux, ceux de $\alpha, \alpha^2 \dots \alpha^{\delta-1}$, puissances *successives* de la racine primitive choisie [6]. La raison de ce choix étrange au premier abord est liée à la propriété VIII.1.10 ci-dessous.

Propriété VIII.1.8. *Les codes BCH sont des codes cycliques.*

Démonstration. On remarque que

$$G(Z) = \operatorname{lcm}\left\{M_i(Z) \,|\, i = 1 \dots (\delta-1)\right\} \;\Big|\; \operatorname{lcm}\left\{M_i(Z) \,|\, i = 1 \dots (n-1)\right\} = Z^n - 1$$ ∎

Propriété VIII.1.9. *Le degré du polynôme générateur d'un code BCH est au plus* $\mu(\delta-1)$*. Celui d'un code BCH binaire est au plus* $\frac{\mu\delta}{2}$*.*

Démonstration. Comme α^i est une racine μ-ième, son polynôme minimal $M_i(Z)$ est au plus de degré μ. Donc leur commun multiple est de degré inférieur à celui de leur produit, c'est-à-dire $\mu(\delta-1)$.

Dans le cas binaire, on sait que $\alpha^2 = \alpha$ (c'est vrai pour tout élément), donc $M_i(Z) = M_{2i}(Z)$. Donc le commun multiple ne doit tenir compte que d'un polynôme sur deux (au plus), donc $\frac{\delta}{2}$ polynômes de degré μ au plus, ce qui donne le résultat. ∎

La propriété suivante est essentielle, car elle garantit une distance minimale au code engendré.

Propriété VIII.1.10 (Distance minimale)**.** *La distance minimale d'un code BCH est au moins égale à* δ*, sa distance par construction.*

Démonstration. Par la propriété VII.2.6, on sait que cette distance est le poids minimal du code. Par l'absurde, supposons qu'il existe un mot-polynôme du code $Y(Z) = \sum_{i=0}^{n-1} y_i Z^i$ avec $w(y) = w < \delta$; notons $I = \{i_1, i_2 \dots i_w\}$ les indices des coefficients non nuls ($i_k \in I \Leftrightarrow y_{i_k} \neq 0$). Comme $Y(Z)$ est multiple de $G(Z)$, toutes les racines primitives $\alpha^j, j = 1 \dots (\delta-1)$ sont des racines de $Y(Z)$, c'est-à-dire $Y(\alpha^j) = 0$. Considérons ces w premières relations; ce sont des combinaisons linéaires (*i.e.* constituent un système d'équations linéaires homogène) des y_{i_k} avec les $\alpha^{j \cdot i_k}$ comme coefficients. Son déterminant vaut:

$$\det\begin{bmatrix} \alpha^{i_1} & \alpha^{i_2} & \cdots & \alpha^{i_w} \\ \alpha^{2i_1} & \alpha^{2i_2} & \cdots & \alpha^{2i_w} \\ \vdots & \vdots & & \vdots \\ \alpha^{wi_1} & \alpha^{wi_2} & \cdots & \alpha^{wi_w} \end{bmatrix} = \prod_{j=1}^{w} \alpha^{i_j} \prod_{1 \le j < k \le w} \left(\alpha^{i_j} - \alpha^{i_k}\right)$$

puisque, lorsqu'on a mis le produit des α^{i_j} en évidence (un par colonne), il reste un déterminant de Vandermonde. Comme les racines primitives sont distinctes et non nulles, ce déterminant est non nul et la seule solution (en y_{i_k}) est nulle, ce qui contredit l'hypothèse. ∎

Remarquons qu'il s'agit d'une borne inférieure, souvent largement dépassée. En effet, si plusieurs racines α^j, pour $1 \le j < \delta$ partagent le même polynôme minimal, en prendre une seule suffit pour avoir une distance minimale au moins égale au degré de ce polynôme (> 1).

En résumé, les codes $BCH_q(\mu, \delta)$ sont des codes de paramètres $[q^\mu - 1, q^\mu - 1 - \varrho, d]_q$ où avec $d \ge \delta$ et ϱ est la redondance avec $\varrho \le \mu(\delta-1)$.

Un cas intéressant est celui correspondant à $\mu = 1$. La racine α est alors un élément primitif de $\mathbb{F}_q$; ses $(q-1)$ puissances parcourent les éléments non nuls du corps et leurs polynômes primitifs dans $\mathbb{F}_q$ sont tous du 1er degré: $(Z - \alpha^j)$. Le polynôme générateur est alors exactement de degré $(\delta-1)$, puisqu'il vaut: $G(Z) = \prod_{j=1}^{\delta-1}(Z - \alpha^j)$.

Propriété VIII.1.11. *Un code* $BCH_q(1, \delta)$ *est un code MDS de paramètres* $[q-1, q-\delta, \delta]_q$*.*

6. D'autres puissances de cette racine primitive peuvent éventuellement être également être des racines de $G(Z)$, bien sûr.

Les codes MDS sont étudiés ci-dessous ; ils correspondent aux codes de distance minimale maximale appelée borne de Singleton (*cf.* propriété VI.3.1).

Démonstration. On sait que, pour tout code BCH, $\delta \leq d$. Par ailleurs, la borne de Singleton dit que $d \leq n - k + 1$. Or, $n - k + 1 = (q - 1) - (q - \delta) + 1 = \delta$, puisque la redondance $\deg G = \delta - 1 = n - k$. On a donc $d = \delta$. ∎

Exemple. L'exemple de corps cyclique binaire vu ci-dessus est un code BCH. En effet, $n = 7 = 2^3 - 1$, donc $\mu = 3$ et le polynôme $G(Z) = 1 + Z + Z^3$ est le polynôme minimal d'une racine 7^e primitive.

Si nous regardons plus en détail, on factorise $Z^7 + 1 = (Z + 1)(Z^3 + Z + 1)(Z^3 + Z^2 + 1)$ et si nous prenons α comme racine de $Z^3 + Z + 1$, ce polynôme est le polynôme minimal, non seulement de α, mais également de α^2 et α^4 (*i.e.* $M_1(Z) = M_2(Z) = M_4(Z)$)[7]. Nous pouvons ainsi choisir $\delta = 1$, mais aussi $\delta = 3$ et conserver le même $G(Z) = \operatorname{lcm}\{M_1(Z), M_2(Z)\}$. C'est donc un code $\mathrm{BCH}(3, 3)_2$ et sa distance minimale vaut $d = 3$.

a) Calcul du syndrome

Dans les codes BCH, l'évaluation du syndrome peut se faire sans utiliser de calcul du résidu polynomial ni de matrice de contrôle, mais directement à l'aide des racines primitives de l'unité. En effet, on sait que $\Sigma_y(Z) = Y(Z) - X(Z) \cdot G(Z)$, donc, en particulier, par construction du code :

$$\Sigma_y(\alpha^i) = Y(\alpha^i) \quad \text{pour} \quad 0 < i < \delta$$

Cela suggère que le syndrome pourrait se construire comme le polynôme d'interpolation passant par les $(\delta - 1)$ points $(\alpha^i, Y(\alpha^i))$, mais ces coordonnées ne sont pas des éléments de $\mathbb{F}_q$: ce sont des éléments appartenant à son extension $\mathbb{F}_{q^\mu} \simeq \mathbb{F}_q[Z]/\langle M_1(Z)\rangle$, puisque α est une racine n-ième de l'unité. Ce sera pourtant la voie suivie pour optimiser le décodage, comme nous allons le voir.

Dans $\mathbb{F}_{q^\mu}$, le polynôme générateur du code est exactement $\prod_{i=1}^{\delta-1}(Z - \alpha^i)$, de degré $(\delta - 1)$; le syndrome de $Y(Z)$ est donc un polynôme de degré inférieur.

Travailler dans $\mathbb{F}_{q^\mu}$ permettra de simplifier les calculs et surtout, de définir une méthode directe de correction. Par conséquent, on définit ainsi le *syndrome dans l'extension* suivant (que nous noterons $S_y(Z) \in \mathbb{F}_{q^\mu}[Z]$ pour le distinguer du syndrome classique $\Sigma_y \in \mathbb{F}_q[Z]$) :

$$S_y(Z) = \sum_{j=0}^{\delta-2} s_j Z^j = \sum_{i=1}^{\delta-1} Y(\alpha^i) Z^{i-1} \tag{VIII.3}$$

C'est un polynôme de degré au plus $(\delta - 2)$, mais dont les coefficients $Y(\alpha^i)$ ne sont pas tous dans $\mathbb{F}_q$.

Propriété VIII.1.12.

$$S_y(Z) = 0 \Leftrightarrow G(Z) \mid Y(Z) \Leftrightarrow Y(Z) \in K$$

Démonstration. $S_y(Z)$ s'annule bien lorsque $Y(Z)$ est un mot du code. Inversement, si un polynôme $Y(Z)$ conduit à un $S_y(Z)$ nul, on a $Y(\alpha^i) = 0, \forall\, i$, donc $Y(Z)$ est multiple de $\prod_{i=1}^{\delta-1}(Z - \alpha^i)$. Comme il est dans $\mathbb{F}_q[Z]$, il est multiple des polynômes minimaux des α^i, donc multiple de $G(Z)$. ∎

Cette propriété permet d'appeler $S_y(Z)$ également un « syndrome ».

Propriété VIII.1.13. *Le syndrome dans l'extension* $S_y(Z)$ *permet de calculer complètement le syndrome* $\Sigma_y(Z)$*.*

Démonstration. Soit $\varrho = \deg G = n - k$. On sait que $G(Z)$ possède exactement ϱ racines qui sont des α^j pour certains j, donc le système $\Sigma_y(\alpha^j) = Y(\alpha^j)$ est un système de ϱ équations en les ϱ inconnues $\sigma_i, 0 \leq i \leq \varrho - 1$ (car $\sigma_\varrho = 1$, par construction) dont le déterminant est non nul, puisque c'est un déterminant de Vandermonde (voir la démonstration de la propriété VIII.1.10). ∎

Ceci suffirait donc à déterminer complètement le polynôme $\Sigma_y(Z)$, mais il nous faudrait, pour ce faire, connaître toutes les racines de $G(Z)$, en plus des $(\delta - 1)$ déjà connues par construction du code.

7. Le polynôme $Z^3 + Z^2 + 1$ est lui le polynôme minimal de α^3, α^5, α^6, tandis que $Z + 1$ est celui de $\alpha^7 = 1$.

b) Équation-clé

Heureusement, il n'est pas nécessaire d'expliciter $\Sigma_y(Z)$ pour identifier les erreurs. En effet, nous savons que le code peut corriger (au moins) $t = \left\lfloor \frac{\delta-1}{2} \right\rfloor$ erreurs (où δ est la distance par construction du code BCH). Donc, les seuls cas « intéressants » sont ceux où $\Sigma_y(Z)$ n'a que t coefficients non nuls, au plus. Ainsi, les syndromes intéressants sont de la forme $\Sigma(Z) = \sum_{j=1}^{t} \sigma_{p_j} Z^{p_j}$ où $2t \leq \delta - 1$, avec $0 \leq p_j < n$ et $\sigma_{p_j} \neq 0$; les autres monômes sont absents[8]. Ces coefficients indiquent la position d'une erreur (par leur indice p_j), ainsi que la correction à apporter (par leur valeur σ_{p_j}) lorsqu'on n'est pas dans le cas binaire. De plus, comme les codes BCH sont des codes linéaires, on peut s'intéresser aux seules erreurs à l'origine (les erreurs du mot nul).

Pour s'en convaincre, observons le cas où il n'y aurait qu'une seule erreur en position p ($0 \leq p < n$). Dans ce cas, le vecteur d'erreur $\Sigma_y(Z) = \sigma_p Z^p$, donc $\Sigma_y(\alpha) = \sigma_p \alpha^p$ et $\Sigma_y(\alpha^2) = \sigma_p \alpha^{2p} = \alpha^p \Sigma_y(\alpha)$. Deux[9] équations suffisent pour retrouver directement p et σ_p. S'il y a plusieurs erreurs corrigibles, même s'il n'est pas simple d'imaginer une généralisation immédiate de cette relation, on va se servir du *polynôme de localisation des erreurs*.

Définition (Polynôme de localisation des erreurs)**.** Le polynôme de localisation de t erreurs aux positions $P = \{p_1, p_2 \ldots p_t\}$ d'un code $\mathrm{BCH}_q(\mu, \delta)$ est

$$\Lambda_P(Z) = \prod_{j=1}^{t} (1 - \alpha^{p_j}.Z)$$

avec $0 \leq p_j < n = q^\mu - 1$ et $2t \leq \delta - 1$.

Ses racines sont exactement les α^{-p_j}, ce qui va nous permettre de localiser et d'évaluer les erreurs.

En effet, comme $\Sigma_y(Z)$ ne contient que les t monômes $\sigma_{p_j} Z^{p_j}$, nous réécrivons

$$S_y(Z) = \sum_{i=1}^{\delta-1} Y(\alpha^i) Z^{i-1} = \sum_{i=1}^{\delta-1} \Sigma_y(\alpha^i) Z^{i-1} = \sum_{i=1}^{\delta-1} \left(\sum_{j=1}^{t} \sigma_{p_j} \left(\alpha^i\right)^{p_j} \right) Z^{i-1} = \sum_{j=1}^{t} \sigma_{p_j} \alpha^{p_j} \sum_{i=1}^{\delta-1} \left(\alpha^{p_j} Z\right)^{i-1}$$

C'est une progression géométrique d'exposants de 0 à $(\delta - 2)$, donc

$$S_y(Z) = \sum_{j=1}^{t} \sigma_{p_j} \alpha^{p_j} \frac{1 - \left(\alpha^{p_j} Z\right)^{\delta-1}}{1 - \alpha^{p_j} Z}$$

Multiplions par $\Lambda_P(Z)$, le polynôme localisateur des erreurs aux positions de $Y(Z)$, on a

$$\begin{aligned}
\Lambda_P(Z).S_y(Z) &= \sum_{j=1}^{t} \sigma_{p_j} \alpha^{p_j} \left(1 - \alpha^{p_j(\delta-1)} Z^{\delta-1}\right) \prod_{i \neq j} (1 - \alpha^{p_i} Z) \\
&= \sum_{j=1}^{t} \sigma_{p_j} \alpha^{p_j} \prod_{i \neq j} (1 - \alpha^{p_i} Z) \; - \; Z^{\delta-1} \sum_{j=1}^{t} \sigma_{p_j} \alpha^{p_j(\delta-1)} \prod_{i \neq j} (1 - \alpha^{p_i} Z) \\
&= \Omega_y(Z) \; - \; Z^{\delta-1} \, \Xi_y(Z)
\end{aligned}$$

où $\Omega_y(Z)$ est un polynôme de degré inférieur à $(\delta - 1)$ et $\Xi_y(Z)$, inférieur à t.

Définition (Polynôme d'évaluation des erreurs)**.** Le polynôme

$$\Omega_y(Z) = \sum_{j=1}^{t} \sigma_{p_j} \alpha^{p_j} \prod_{i \neq j} (1 - \alpha^{p_i} Z)$$

est le polynôme d'évaluation des erreurs du syndrome $\Sigma_y(Z)$ aux positions $P = \{p_1, p_2 \ldots p_t\}$ telles que $\sigma_i \neq 0 \Leftrightarrow i \in P$.

8. Voilà pourquoi certains auteurs n'utilisent pas comme paramètre la distance par construction δ, mais le nombre d'erreurs que l'on peut corriger t et ils construisent le polynôme générateur à partir de $2t$ puissances successives de la racine primitive.
9. Si $t = 1$, alors $2t + 1 = \delta = 3$ et le nombre d'équations est $\delta - 1 = 2$.

On constate, qu'il ne dépend que des α^{p_j} et des σ_{p_j} en les seules positions dans P.

Ce polynôme $\Omega_y(Z)$ d'évaluation des erreurs est déterminé à partir du polynôme de localisation des erreurs et du syndrome dans l'extension grâce à l'*équation-clé* :

$$\Omega_y(Z) = \Lambda_P(Z) \cdot S_y(Z) \bmod Z^{\delta-1} < \tag{VIII.4}$$

c) Positions des erreurs : algorithmes d'Euclide et de Chien

Si nous réécrivons l'équation-clé, nous trouvons une relation de Bézout [10]

$$\Lambda_P(Z) \cdot S_y(Z) + \Xi_y(Z) \cdot Z^{\delta-1} = \Omega_y(Z)$$

Celle-ci, plus exactement ses polynômes-coefficients inconnus $\Lambda_P(Z)$ et $\Xi_y(Z)$, peuvent être produits par l'algorithme d'Euclide étendu, comme l'ont indiqué Yasuo Sugiyama *et al.* en 1975 [37].

On divise le polynôme $Z^{\delta-1}$ par $S_y(Z)$ (de degré nécessairement inférieur), ce qui donne un reste de degré strictement inférieur à ce dernier et on poursuit le processus de divisions successives jusqu'à obtenir un reste de degré inférieur t.

Pour rappel, l'algorithme d'Euclide étendu sur les deux valeurs a et b calcule une suite de quotients q_i et de restes r_i successifs, mais également les coefficients α_i et β_i, tels que $\alpha_i \cdot a + \beta_i \cdot b = r_i$ selon la même récurrence

$$X_i = X_{i-2} - q_i X_{i-1} \text{ où } q_i = \left\lfloor \frac{r_{i-2}}{r_{i-1}} \right\rfloor \text{ pour } i > 1$$

mais avec des valeurs initiales différentes :

$$\begin{aligned} r_{-1} &= a & r_0 &= b \\ \alpha_{-1} &= 1 & \alpha_0 &= 0 \\ \beta_{-1} &= 0 & \beta_0 &= 1 \end{aligned}$$

Si a et b sont des nombres entiers, les r_i décroissent, tandis que les α_i et β_i croissent en valeur absolue et leurs signes alternent.

Ici, ce sont des polynômes sur un corps ; la décroissance, respectivement la croissance, vise donc leurs degrés. À la fin, le polynôme $\Lambda_P(Z)$ est le coefficient de $S_y(Z)$ à une constante près, aisément fixée parce que, par définition, son terme indépendant $\lambda_0 = 1$.

Ayant déterminé $\Lambda_P(Z)$, il faut connaître ses racines, par exemple à l'aide de l'algorithme « Chien search » (1964, [6]) de Robert Tienwen Chien (錢天問). Il s'agit essentiellement d'une méthode efficace de recherche *par essai et erreur* parmi les α^i qui en fournit immédiatement l'indice. Comme par construction, les racines de $\Lambda_P(Z)$ sont les t valeurs α^{-p_j}, la relation $\alpha^{-i} = \alpha^{q-1-i}$ donne les positions p_j recherchées.

Notons qu'il existe de nombreuses autres méthodes de détermination du polynôme de localisation des erreurs ou de résolution de l'équation-clé, notamment l'algorithme de Berlekamp-Massey (1969) qui utilise des registres à décalage de manière semblable à l'algorithme de Meggitt vu ci-dessus. À l'origine, il s'agit d'une idée de Elwyn Ralph Berlekamp, présentée en 1967 comme application particulière de son algorithme de factorisation des polynômes sur un corps fini. Cet algorithme a ensuite été amélioré et étendu (au calcul du polynôme minimal d'une récurrence, notamment) par James Lee Massey (voir [25]).

d) Valeurs des erreurs : algorithme de Forney

Nous connaissions maintenant l'ensemble P des positions des erreurs. Si on est dans le cas binaire, on a fini, puisque pour corriger y, il suffit de basculer les bits aux positions de P. Mais dans les autres cas ($q > 2$), il nous faut connaître la correction à appliquer.

10. Du nom du mathématicien français Étienne Bézout.

Pour déterminer les *valeurs* des erreurs connaissant leurs positions, on peut appliquer une variante d'interpolation de Lagrange connue sous le nom d'*algorithme de Forney* (décrit en 1965 par George David « Dave » Forney, Jr. [10]). On observe que

$$\Omega_y\left(\alpha^{-p_j}\right) = \sigma_{p_j}\,\alpha^{p_j}\prod_{i\neq j}\left(1-\alpha^{p_i-p_j}\right)$$

puisque tous les produits contenant $\left(1-\alpha^{p_j}Z\right)$ s'annulent ; cette formule ne contient plus que ce seul terme. De plus, la dérivée formelle

$$\Lambda'_{\mathrm{P}}(Z) = -\sum_{j=1}^{t}\alpha^{p_j}\prod_{i\neq j}\left(1-\alpha^{p_i}.Z\right)$$

permet de l'écrire plus simplement, car

$$\Lambda'_{\mathrm{P}}\left(\alpha^{-p_j}\right) = -\alpha^{p_j}\prod_{i\neq j}\left(1-\alpha^{p_i-p_j}\right)$$

Ainsi, la correction à appliquer à y est

$$-\sigma_{p_j} = \frac{\Omega_y\left(\alpha^{-p_j}\right)}{\Lambda'_{\mathrm{P}}\left(\alpha^{-p_j}\right)}$$

aux t positions $p_1, p_2 \ldots p_t$, ce qui est une forme explicite des valeurs de la correction à appliquer. On constate que tous les calculs se sont faits dans $\mathbb{F}_{q^\mu}$, mais que les valeurs σ_{p_j} obtenues à la fin sont bien dans l'alphabet du code $\mathbb{F}_q$.

VIII.2 Codes linéaires parfaits

Nous avons vu que les codes parfaits sont ceux pour lesquels le rayon d'empilement et le rayon de recouvrement sont égaux, ou pour lesquels la borne de Hamming est atteinte, ou encore qui corrigent t erreurs avec $2t+1 = d$. Cette définition peut s'appliquer à tout code, mais dans le contexte des codes linéaires, la relation fournie par la borne de Hamming — *i.e.* l'espace des mots est l'union des boules de rayon t centrées sur les mots du code — induit des contraintes de divisibilité entre expressions entières :

$$\sum_{i=0}^{t}\binom{n}{i}(q-1)^i = q^{n-k} \tag{VIII.5}$$

Par exemple, on identifie immédiatement quelques cas simples :

- $t = 0$ (donc $d = 1$) : on a $1 = q^{n-k}$ donc $n = k$ et le code est le code trivial, celui où tout mot est mot du code ;
- $t = n$, c'est-à-dire $\mathrm{K} = \{0\}$, c'est le code nul ;
- $t = 1$ (donc $d = 3$) : on a $n = \frac{q^{n-k}-1}{q-1}$; ce sont les codes de Hamming décrits ci-dessous.

Attention, il s'agit bien sûr d'une condition *nécessaire* d'existence d'un code linéaire parfait. Mais en fait, ce sont presque les seuls cas comme l'ont démontré de manière complémentaire Jacobus Hendricus van Lint en 1971 [41] et Aimo Antero Tietäväinen en 1973 [38].

Théorème VIII.2.1 (van Lint-Tietäväinen, 1973)**.** *Les seuls codes linéaires parfaits sont ceux repris dans le tableau VIII.4.*

La démonstration sort du contexte de ces notes (voir [24], par exemple).

Nous allons donc décrire ci-dessous les codes de Hamming et esquisser ceux de Golay.

Ce théorème VIII.2.1 peut être étendu aux codes parfaits *non linéaires* pour $t > 2$, grâce aux travaux de M.R. Best (en 1978) et de Y. Hong (en 1984) . Mais le problème reste ouvert pour $t = 2$: en dehors des cas triviaux, n'y aurait-il que le seul code de Golay ternaire ? Par contre, pour $t = 1$, on connaît des codes non linéaires (qui ne sont donc pas repris dans le tableau VIII.4).

TABLEAU VIII.4 – Codes linéaires parfaits

nom	base q	longueur n	dimension k	distance d	correction t
Nul ($K = \{0\}$)	toutes	toutes	0	n/a	n
Trivial ($K = \mathbb{F}_q^n$)	toutes	toutes	n	1	0
Répétition binaire	2	$2t+1$	1	n	toutes
Golay binaire	2	23	12	7	3
Golay ternaire	3	11	6	5	2
Hamming $\mathrm{Ham}_q(\varrho)$	toutes	$\frac{q^\varrho-1}{q-1}$	$n-\varrho$	3	1

Exemple (Code parfait non linéaire)*.* Le code

$$K = \left\{ \left(x, x\oplus y, (\bigoplus_i x_i) \oplus (\bigwedge_j y_j)\right) \middle| x \in \mathbb{F}_2^7, y \in \mathrm{Ham}_2(3) \right\}$$

est un code binaire parfait de longueur $n = 15 = 7+7+1$, de dimension $k = 11 = 7+4$, de distance $d = 3$ et qui corrige donc $t = 1$ erreur. On peut montrer qu'il n'est équivalent à aucun code linéaire [11].

VIII.2.1 Codes de Hamming

Comme le dit le théorème VIII.2.1, les codes de Hamming sont la seule famille infinie de codes linéaires parfaits non triviaux. Le premier exemple de tels codes a été découvert par Richard Wesley Hamming en 1950 [18] ; c'était $\mathrm{Ham}_2(3)$, un code binaire linéaire parfait de paramètres $[7,4,3]_2$. C'était, à l'époque de la naissance des ordinateurs, une technique révolutionnaire et efficace pour encoder un nibble (un chiffre hexadécimal, un demi-octet) en mémoire et sur les supports externes, tant les taux d'erreur étaient élevés.

Très rapidement, il a généralisé ce cas particulier par la construction argumentée suivante.

De la condition nécessaire de l'équation (VIII.5) où $t = 1$, il nous faut construire une matrice de contrôle H de ϱ lignes (la redondance) et $n = \frac{q^\varrho-1}{q-1}$ colonnes. Celle-ci applique les q^n vecteurs de l'espace des mots vers les q^ϱ syndromes. Considérons donc l'hyperplan à l'infini de cet espace vectoriel de dimension ϱ sur $\mathbb{F}_q$; c'est l'espace projectif des directions (ou des droites) de cet espace. Il y en a exactement n (le nombre $(q^\varrho-1)$ de vecteurs non nuls, divisé par le nombre $(q-1)$ de multiples non nuls). Et dans cet espace des syndromes, les ϱ directions des vecteurs de base sont, par essence, linéairement indépendantes. Par conséquent, la matrice dont les colonnes correspondent à ces directions est une matrice de bonnes dimensions et de rang ϱ maximal ; c'est donc une matrice de contrôle d'un code linéaire.

Propriété VIII.2.2. *Pour tout $\varrho > 1$, le code linéaire de paramètres $\left[\frac{q^\varrho-1}{q-1}, \frac{q^\varrho-1}{q-1} - \varrho, 3\right]_q$ existe, est parfait et est unique (à un isomorphisme près).*

Démonstration. La construction explicite d'une matrice de contrôle ci-dessus prouve l'existence.

L'unicité découle du fait que toutes les colonnes possibles sont reprises dans cette matrice. En effet, si l'on remplaçait l'une d'entre elles, ce serait soit par un multiple soit d'elle-même, ce qui ne change rien par linéarité, soit par un multiple d'une autre. Dans ce cas, le code ne serait plus parfait, parce que les deux vecteurs d'erreurs à l'origine (des vecteurs de base de l'espace des mots) correspondant à ces positions se projetteraient sur le même syndrome.

Par la propriété VII.2.7, on sait que la distance minimale est le nombre minimum de colonnes dépendantes de la matrice de contrôle. Deux colonnes sont toujours libres, par construction (le contraire voudrait dire que l'une est multiple de l'autre ou nulle). Par contre, pour toute paire de colonnes, leur somme pondérée y est présente, donc la distance minimale est 3. Il est donc parfait, puisqu'il satisfait (VIII.5) pour $t = 1$. ∎

11. Le seul ayant les bons paramètres serait $\mathrm{Ham}_2(4)$.

Définition (Code de Hamming)**.** Le code ayant pour paramètres ceux de la propriété VIII.2.2 est appelé « code de Hamming de redondance ϱ en base q » et est noté $\mathrm{Ham}_q(\varrho)$.

Remarque 2*.* La construction de la matrice de contrôle décrite ci-dessus ne détermine évidemment pas l'ordre des colonnes ; on peut donc les permuter. Et elle ne conduit pas directement à une matrice génératrice du code. Mais puisque toutes les directions sont présentes, les μ colonnes constituant une matrice identité sont présentes. On peut donc par permutation mettre cette matrice de contrôle sous forme canonique, ce qui permet d'expliciter une matrice génératrice sous forme canonique également.

Remarque 3*.* Les matrices de contrôle des codes de Hamming binaires sont simplement constituées de colonnes correspondant à la représentation binaire des nombres 1 à $2^\varrho-1$ (à une permutation près). Celles en base q, de la représentation dans cette base du sous-ensemble des nombres 1 à $q^\varrho-1$ *dont le chiffre de poids fort est* 1.

Pour le cas où $q = p^\nu$ avec $\nu > 1$, un élément de $\lambda \in \mathbb{F}_q$ est assimilé au « chiffre » $\mathrm{I}(\lambda)$ où

$$\mathrm{I}(\lambda) = \begin{cases} 0 & \text{si } \lambda = 0 \\ c \,:\, \alpha^{c-1} = \lambda & \text{sinon} \end{cases}$$

pour un α élément primitif de $\mathbb{F}_q$.

Nous appellerons cette construction « par énumération ».

Propriété VIII.2.3. *Les codes de Hamming ont asymptotiquement un débit optimal.*

Démonstration. Le débit du code par (VI.5) est

$$\mathrm{R}(\mathrm{Ham}_q(\varrho)) = 1 - \frac{\varrho\,(q-1)}{q^\varrho - 1}$$

Donc $\lim_{\varrho\to\infty} \mathrm{R}(\mathrm{H}_q(\varrho)) = 1$. ∎

Mais sa capacité de correction s'amenuise, puisque sur $n = \frac{q^\varrho-1}{q-1}$ caractères, une seule erreur peut être corrigée ($t = 1$ pour tous les codes de Hamming). Ce sont donc plutôt les « petits » codes qui sont utilisés. Le tableau VIII.5 reprend les premiers exemples (par longueur croissante $n < 256$).

Notons que certains codes de Hamming appartiennent également à d'autres familles. Ainsi, le code $\mathrm{Ham}_2(2)$ est le code à répétition binaire de longueur 3 ; $\mathrm{Ham}_2(3)$ est le code BCH binaire engendré par $\mathrm{Z}^3 + \mathrm{Z} + 1$, etc. Plus généralement, nous avons la propriété suivante.

Propriété VIII.2.4. *Tous les codes de Hamming* ***binaires*** *sont des codes BCH (donc cycliques, donc polynomiaux).*

Démonstration. Soit α une racine $(2^\varrho-1)$-ième primitive binaire (*i.e.* un élément primitif de $\mathbb{F}_{2^\mu}$). Considérons le code polynomial des mots de longueur $n = 2^\varrho - 1$ qui ont α pour racine : $y \in \mathrm{K} \Leftrightarrow \mathrm{Y}(\alpha) = 0$. Ils sont tous multiples du polynôme minimal de α ; c'est bien un code BCH.

Si nous considérons le vecteur y et celui des puissances de α : $a^\tau = [\alpha^0, \alpha^0 \ldots \alpha^{n-1}]$, on peut écrire $\mathrm{Y}(\alpha) = a^\tau \cdot y$ et les composantes de a correspondent à tous les éléments non nuls du corps $\mathbb{F}_{2^\varrho}$, puisqu'il est cyclique engendré par un de ses éléments primitifs.

Transformons maintenant chaque élément du vecteur-ligne a^τ en une colonne correspondant aux coefficients du polynôme binaire représentant α^j ; on obtient une matrice à ϱ lignes dont les colonnes sont distinctes et parcourent également toutes les configurations de ϱ bits. C'est donc une matrice de contrôle du code de Hamming $\mathrm{Ham}_2(\varrho)$.

Vu l'unicité du code $\mathrm{Ham}_2(\varrho)$ et que ce raisonnement est valable pour tout ϱ, la propriété en découle. ∎

Par contre, cette propriété n'est pas nécessairement vraie pour les codes de Hamming non binaires, mais il y a des situations où c'est bien le cas. Par exemple, on peut prouver que

TABLEAU VIII.5 – Petits codes de Hamming

q	ϱ	n	k	R
2	2	3	1	0,33333
3	2	4	2	0,50000
4	2	5	3	0,60000
5	2	6	4	0,66667
2	3	7	4	0,57143
7	2	8	6	0,75000
8	2	9	7	0,77778
9	2	10	8	0,80000
11	2	12	10	0,83333
3	3	13	10	0,76923
13	2	14	12	0,85714
2	4	15	11	0,73333
16	2	17	15	0,88235
17	2	18	16	0,88889
4	3	21	18	0,85714
2	5	31	26	0,83871
5	3	31	28	0,90323
3	4	40	36	0,90000
7	3	57	54	0,94737
2	6	63	57	0,90476
8	3	73	70	0,95890
4	4	85	81	0,95294
9	3	91	88	0,96703
3	5	121	116	0,95868
2	7	127	120	0,94488
11	3	133	130	0,97744
5	4	156	152	0,97436
13	3	183	180	0,98361
2	8	255	247	0,96863

Propriété VIII.2.5. *Soit α est une racine n-ième de l'unité dans $\mathbb{F}_{q^\varrho}$ où $n = \frac{q^\varrho-1}{q-1}$. Si* $\gcd(\varrho, q-1) = 1$ *(i.e. la redondance ϱ est première avec l'ordre de α), alors le code cyclique engendré par* $\mathrm{M}_\alpha(Z)$*, le polynôme minimal de α, est le code de Hamming* $\mathrm{Ham}_q(\varrho)$*.*

Toutefois, certains codes de Hamming ne sont pas cycliques ni même polynomiaux, comme le montre l'exemple suivant.

Exemple. Considérons le code $\mathrm{Ham}_3(2)$, c'est-à-dire $[4, 2, 3]_3$. Une matrice de contrôle par énumération est

$$\mathrm{H} = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Une forme canonique est, par exemple

$$\mathrm{H} = \left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right]$$

ce qui correspond à la matrice génératrice canonique

$$\mathrm{G} = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \hline 2 & 1 \\ 2 & 2 \end{array}\right]$$

qui engendre le code de 9 mots : $\mathrm{K} = \left\{ \begin{bmatrix}0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\2\\2\end{bmatrix}, \begin{bmatrix}2\\0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\1\\1\\2\end{bmatrix}, \begin{bmatrix}1\\1\\0\\1\end{bmatrix}, \begin{bmatrix}2\\1\\2\\0\end{bmatrix}, \begin{bmatrix}0\\2\\2\\1\end{bmatrix}, \begin{bmatrix}1\\2\\1\\0\end{bmatrix}, \begin{bmatrix}2\\2\\0\\2\end{bmatrix} \right\}$.

Ce code n'est pas cyclique[12] : par exemple $\begin{bmatrix}1\\0\\2\\2\end{bmatrix}$ est dans le code, mais pas $\begin{bmatrix}2\\2\\1\\0\end{bmatrix}$. Il n'est même pas polynomial, puisque $\gcd\left(1+2Z^2+2Z^3,\ Z+Z^2+2Z^3\right)=1$, donc les deux vecteurs-colonnes de G, base de K, ne peuvent être multiples d'aucun polynôme de degré 2.

a) Décodage systématique des codes de Hamming

Le fait que les codes de Hamming soient parfaits avec un pouvoir de correction $t = 1$ permet de concevoir une technique efficace de décodage par syndrome. Il faut en effet déterminer le seul caractère à corriger par examen des $\varrho(q-1)$ syndromes correspondant aux vecteurs de base et leurs multiples non nuls. Mais si l'on applique la construction par énumération de la matrice de contrôle, la réponse est immédiate, par simple observation de la valeur du syndrome :

Propriété VIII.2.6. *Si la matrice de contrôle d'un code de Hamming est construite par énumération, le j-ème caractère de contrôle ($1 \le j \le \varrho$) est en position ω_j ($1 \le \omega_j \le n$) où*

$$\omega_j = \frac{q^{j-1}+q-2}{q-1}$$

solution de la récurrence

$$\omega_{j+1} = \omega_j + q^{j-1} \quad \textit{avec} \quad \omega_1 = 1$$

Démonstration. Dans le cas binaire, c'est immédiat puisque les colonnes de la matrice de contrôle sont exactement la codification binaire des nombres 1 à $n = 2^\varrho$ successifs. Les bits de parité sont donc aux positions des puissances de 2 successives (2^{j-1}).

Dans les autres cas, la récurrence est : chaque fois qu'on ajoute une j-ième composante non nulle, nécessairement 1, il faut la compléter de toutes les combinaisons possibles des composantes précédentes (il y en a q^{j-1}). On vérifie aisément que la forme explicite est la solution de la récurrence linéaire ; c'est en effet une série en progression géométrique. ∎

La valeur du syndrome calculé par cette matrice de contrôle donne ainsi directement le caractère à corriger. En effet, si $\sigma(y) = Hy$ et que y contient au plus une erreur εe_p (où e_j est le j-ième vecteur de base), alors le caractère à corriger est en position

$$p = \sum_{j=0}^{\varrho-1} \mathrm{I}\left(\varepsilon^{-1}\sigma_j\right)\omega_j \tag{VIII.6}$$

une valeur nulle indiquant bien sûr l'absence d'erreur (y est alors un mot du code).

Dans le cas binaire, $\varepsilon = 1$ (*i.e.* la valeur de l'erreur ne doit pas être déterminée, seule sa position importe) et la configuration binaire du syndrome indique directement la position du bit à inverser.

Dans les autres cas, il faut déterminer la valeur de l'erreur ε et se ramener ainsi à la *direction* du syndrome correspondant à une erreur unité. Or ces directions sont précisément les colonnes de H. Pour décoder, avant d'utiliser (VIII.6), on devrait rechercher les $(q-1)$ multiples du syndrome pour trouver celui qui figure dans H. Mais par construction, on connaît la structure de H : il faut que le chiffre le plus significatif soit l'unité. Par conséquent, on a

$$\varepsilon = \sigma_j \text{ avec } j = \max\{i \mid \sigma_i \neq 0\}$$

Exemple. Considérons tout d'abord un cas binaire : le code historique $\mathrm{Ham}_2(3)$. Sa matrice de contrôle par énumération est

$$\mathrm{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

12. L'hypothèse $\gcd(\varrho, q-1) = 1$ n'est pas satisfaite, puisque $\varrho = q-1 = 2$.

Un mot du code est, par exemple $\begin{bmatrix}0\\0\\1\\0\\1\\1\\0\end{bmatrix}$. Si nous devons décoder $y = \begin{bmatrix}0\\0\\0\\0\\1\\1\\0\end{bmatrix}$, nous avons $Hy = \begin{bmatrix}1\\1\\0\end{bmatrix}$, ce qui correspond à la représentation binaire du nombre 3 ; c'est bien le 3e bit qui est erroné.

Reprenons maintenant l'exemple $\mathrm{Ham}_3(2)$ vu ci-dessus [13].

Pour décoder $\begin{bmatrix}0\\1\\0\\1\end{bmatrix}$, image altérée de $\begin{bmatrix}0\\1\\1\\1\end{bmatrix}$, on calcule le syndrome : $\begin{bmatrix}2\\2\end{bmatrix}$. Son chiffre le plus significatif est $2 = \varepsilon$. On le multiplie par l'inverse $2^{-1} = 2$, ce qui donne $\begin{bmatrix}1\\1\end{bmatrix}$. En appliquant ensuite (VIII.6), on trouve $p = 3$. L'erreur est donc « 2 en position 3 ».

b) Codes simplex et distribution des poids des codes de Hamming

Un code de Hamming étant un code linéaire, on peut s'intéresser à son dual $\mathrm{Ham}^{\perp}$, c'est-à-dire celui dont la matrice H^{τ}, transposée de la matrice de contrôle du code de Hamming, est une matrice génératrice. Il est appelé *code simplex* de dimension ϱ.

Propriété VIII.2.7. *(Code simplex) Le dual du code de Hamming* $\mathrm{Ham}_q(\varrho)$ *est un code simplex de paramètres* $\Sigma_q(\varrho) = \left[\frac{q^{\varrho}-1}{q-1}, \varrho, q^{\varrho-1}\right]_q$ *où tous les mots distincts sont à cette distance minimale.*

Démonstration. Il nous faut simplement établir que la distance minimale de $\mathrm{Ham}_q^{\perp}(\varrho)$ est $q^{\varrho-1}$. Soit un mot non nul quelconque de ce code. On choisit une base de l'espace des mots $\mathbb{F}_q^{\varrho}$ contenant ce mot-vecteur ; c'est donc une des colonnes de H^{τ}, appelons-la v. Par la propriété VIII.2.2, on sait que les lignes de H^{τ} sont toutes distinctes. Pour connaître le poids de v, il nous faut compter les lignes de H^{τ} dont la composante en colonne v est une non nulle : il y en a potentiellement $(q-1)q^{\varrho-1}$. Mais v est donné à un multiple non nul près, donc il faut diviser cette expression par $(q-1)$. Tout vecteur-colonne de H^{τ} a donc un poids $q^{\varrho-1}$. ∎

Par conséquent, son polynôme énumérateur des poids (*cf.* (VII.3)) est

$$W^{\perp}(X,Y) = Y^{\frac{q^{\varrho}-1}{q-1}} + (q^{\varrho}-1)\,X^{q^{\varrho-1}}\,Y^{\frac{q^{\varrho-1}-1}{q-1}}$$

Grâce à l'identité de MacWilliams (VII.4), on trouve celui du code de Hamming originel :

$$\begin{aligned} W(X,Y) &= \frac{1}{q^{\varrho}}\left(\left(Y+(q-1)X\right)^{\frac{q^{\varrho}-1}{q-1}} + (q^{\varrho}-1)\left(Y-X\right)^{q^{\varrho-1}}\left(Y+(q-1)X\right)^{\frac{q^{\varrho-1}-1}{q-1}}\right)\\ &= \left(Y+(q-1)X\right)^{\frac{q^{\varrho-1}-1}{q-1}}\left[\frac{1}{q^{\varrho}}\left(Y+(q-1)X\right)^{q^{\varrho-1}} + \left(1-\frac{1}{q^{\varrho}}\right)\left(Y-X\right)^{q^{\varrho-1}}\right] \end{aligned}$$

C'est peu parlant, même si c'est un peu plus sympathique dans le cas binaire, mais on peut l'expliciter dans de cas simples. Par exemple,

$$W_{\mathrm{Ham}_2(3)}(X,Y) = \frac{1}{8}(Y+X)^3\left[(Y+X)^4 + 7(Y-X)^4\right] = Y^7 + 7\,X^3Y^4 + 7\,X^4Y^3 + X^7$$

La structure est symétrique et très simple : 7 mots à distance 3, 7 à distance 4 (les compléments des précédents) et un à distance 7 (le complément du mot de référence).

13. Mais attention, on n'encode pas ici selon la forme canonique, mais selon la forme énumérative.

c) Codes de Hamming étendu et codes d'Hadamard

Les codes de Hamming binaires sont de longueur $2^\varrho - 1$. Dans l'architecture traditionnelle des supports informatiques, le nombre de bits est classiquement une puissance de 2, ce qui laisserait un bit inutilisé. Voilà pourquoi on ajoute fréquemment un bit de contrôle qui est un « vrai » bit de parité. Le code ainsi obtenu corrige toujours $t = 1$ erreur, mais il en détecte $2 = \frac{d}{2}$, car $d = 4$ maintenant.

Ce sont les codes de paramètres $[2^\varrho, 2^\varrho - 1 - \varrho, 4]_2$; ils ne sont plus parfaits.

Cette technique d'extension peut s'appliquer également non seulement au code de Hamming en toute base, mais aussi, de manière générale, à tous les codes linéaires : on construit la matrice de contrôle $\mathring{\mathrm{H}}$ par adjonction d'une dernière ligne (la $(\varrho + 1)$-ième ligne) formée de 1 et d'une dernière colonne (la $(n + 1)$-ième colonne) unité, le dernier caractère du syndrome servant alors de test de parité.

$$\mathring{\mathrm{H}} = \left[\begin{array}{ccc|c} & \mathrm{H} & & \begin{matrix}0\\\vdots\\0\end{matrix} \\ \hline 1 & \cdots & 1 & 1 \end{array}\right]$$

La matrice génératrice correspondante $\mathring{\mathrm{G}}$ a simplement une $(n + 1)$-ième ligne supplémentaire où $\mathring{g}_{nj} = -\sum_{i=0}^{n-1} g_{ij}$.

Les *codes d'Hadamard* — dont le nom fait référence à Jacques Salomon Hadamard, non parce qu'il en est l'inventeur, mais parce qu'on peut les construire à partir de toute *matrice d'Hadamard*[14] — sont des codes linéaires binaires (exclusivement) dont la matrice génératrice est formée de toutes les configurations de k bits ; elle contient donc 2^k lignes. Mais c'est en fait le code simplex binaire étendu, puisqu'on le complète d'une ligne nulle ce qui est bien la valeur d'un bit de parité.

Par la propriété VIII.2.7 de ce code vue ci-dessus,

Définition. (Code d'Hadamard) Un code d'Hadamard est un code linéaire binaire de paramètre $\mathrm{Had}(k) = \left[2^k, k, 2^{k-1}\right]_2$.

VIII.2.2 Codes de Golay

Les codes de Golay sont des cas exceptionnels, en lien avec les structures algébrico-géométriques des *groupes finis simples sporadiques*, en particulier les *groupes de Mathieu*[15]. Leur étude approfondie sort ainsi du contexte de ce document.

Ils ont été décrits par Marcel Jules Édouard Golay en 1949 dans un article très court [14].

Le code de Golay binaire parfait a pour paramètres $[23, 12, 7]_2$; il corrige donc $t = 3$ erreurs (avec pour débit $\mathrm{R} = 0,52174$). Son extension est $[24, 12, 8]_2$. C'est à partir de cette version étendue que le code parfait a été identifié, car il a une structure très régulière, très symétrique. Par exemple, le code étendu est autodual (il a donc autant de bits de parité que de bits d'information) et ses mots ont un poids de Hamming de 0, 8, 12, 16 ou 24. Les groupes d'automorphismes des codes (vu comme n-uplets binaires) sont respectivement les groupes de Mathieu $\mathcal{M}_{23}$ et $\mathcal{M}_{24}$.

On peut décrire la construction du code de Golay binaire étendu de nombreuses façons : comme un ordre lexicographique des n-uplets à distance $d \geq 8$ (code étendu), comme un code linéaire dont la matrice de parité (12×12) est le complément de la matrice d'incidence de l'icosaèdre, polyèdre régulier de Platon (Πλάτων)[16] à 20 faces et 12 sommets (code étendu), etc. Le code parfait est aussi simplement le code cyclique engendré par le polynôme $1 + \mathrm{Z}^2 + \mathrm{Z}^4 + \mathrm{Z}^5 + \mathrm{Z}^6 + \mathrm{Z}^{10} + \mathrm{Z}^{11}$.

Le code de Golay ternaire a pour paramètres $[11, 6, 5]_3$; il corrige donc $t = 2$ erreurs (avec pour débit $\mathrm{R} = 0,54545$). Son extension est donc $[12, 6, 6]_3$, également un code autodual. Leurs groupes d'automorphismes sont respectivement $2 \times \mathcal{M}_{11}$ et $2 \times \mathcal{M}_{12}$.

14. Une *matrice d'Hadamard* H est une matrice carrée composée exclusivement de valeurs ± 1 et dont les lignes sont orthogonales entre elles (*i.e.* leur produit scalaire est nul) : $\mathrm{H} \cdot \mathrm{H}^\intercal = n\mathbb{1}$, si n est la dimension de la matrice.

15. Du nom du mathématicien français Émile Mathieu qui les a découverts en 1861. C'était historiquement les premiers exemples de groupes finis simples sporadiques. Il existe 5 groupes de Mathieu : $\mathcal{M}_n$ pour $n = 11, 12, 22, 23, 24$.

16. Les cinq solides réguliers sont explicitement cités et décrits vers 358 AC dans Τίμαιος (*Le Timée*), un « dialogue » de Πλάτων (Platon), mais ils étaient vraisemblablement connus bien avant.

On peut également décrire sa construction de nombreuses façons. Voici simplement sa matrice de parité.

$$P = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 & 0 \\ -1 & -1 & 1 & -1 & 0 & 1 \\ -1 & 1 & -1 & 0 & -1 & 1 \\ -1 & 1 & 0 & -1 & 1 & -1 \\ -1 & 0 & 1 & 1 & -1 & -1 \end{bmatrix}$$

VIII.3 Codes de Reed-Muller

Les codes de Reed-Muller sont un exemple déjà ancien de tentative d'une autre systématisation du décodage par syndrome. Ils ont été décrits David Eugene Muller [27] en 1954 et, la même année, Irving Stoy Reed [30] a proposé une méthode efficace de décodage. Ils ont été longtemps utilisés pour les communications avec les satellites à l'époque des débuts de la conquête spatiale (années '60).

Ils peuvent être vus comme une généralisation des codes de Hadamard, les codes duaux des codes de Hamming[17]. Ce sont des codes qui garantissent une distance minimale.

La forme originelle est celle d'un code binaire. Un mot y est codé sous la forme d'une fonction booléenne de μ variables représentée par sa table de vérité, c'est-à-dire par sa valeur prise en les 2^μ combinaisons de valeurs possibles pour ses variables. Il existe en effet 2^{2^μ} telles fonctions. Comme toute fonction booléenne peut s'écrire sous forme algébrique normalisée, c'est-à-dire sous forme d'un somme (XOR) de produits (AND) de 0 à μ variables, les composantes du mot à coder correspondent aux coefficients de ces différents termes possibles (apparaissent-ils ou non dans la forme normale de la fonction).

On définit les mots du code comme ceux correspondant à une fonction booléenne de dégré $\leq \delta$. La matrice génératrice est ainsi composée de colonnes reprenant les valeurs des différents termes de degré $\leq \delta$ possibles (il y en a $\sum_{i=0}^{\delta} \binom{\mu}{i}$) pour une configuration de valeurs des variables (il y en a 2^μ). Les mots erronés sont donc ceux qui diffèrent en au moins une entrée, c'est-à-dire ceux dont la table de vérité représentée par les 2^μ composantes du mot correspond à une fonction booléenne de degré $> \delta$.

Définition (Code de Reed-Muller)**.** Un code de Reed-Muller $RM_2(\mu, \delta)$ (avec $0 \leq \delta \leq \mu$) est un code linéaire binaire dont les mots du code sont les fonctions de degré au plus δ parmi l'ensemble de toutes les fonctions booléennes de μ variables. La longueur du code est 2^μ (la table de vérité d'une fonction), tandis que sa dimension est $\sum_{i=0}^{\delta} \binom{\mu}{i}$.

Propriété VIII.3.1 (Construction inductive)**.**

- *Le code* $RM_2(\mu, 0)$ *est le code à répétition de longueur* μ*, pour toute longueur (sa dimension vaut* 1*).*
- *Le code* $RM_2(\mu, \mu) = \mathbb{Z}_2^{2^\mu}$ *est le code trivial formé de tout l'espace de mots (de dimension et longueur* 2^μ*).*
- *Enfin, pour* $\mu > \delta > 0$*, le code est*

$$RM_2(\mu, \delta) = \left\{ (u, u \oplus v)^\tau \mid u \in RM_2(\mu{-}1, \delta), v \in RM_2(\mu{-}1, \delta{-}1) \right\}$$

Démonstration. Le premier est immédiat, puisque la dimension $k = \binom{\mu}{0} = 1$. Le deuxième aussi, puisque $\sum_{i=0}^{\mu} \binom{\mu}{i} = 2^\mu$, donc sa dimension $k = n$.

Pour le cas général, on constate d'abord que pour passer de $(\mu{-}1)$ à μ, la longueur du code double, ce qui est cohérent avec la forme $(u, u \oplus v)$. Ensuite, on peut raisonner sur la matrice génératrice du code. Dans l'énumération de la table de vérité, les lignes de sa première moitié correspondent à une valeur 0 pour la nouvelle μ-ième variable, tandis que celles de la seconde moitié, à une valeur 1, de manière dupliquée. Une fonction f du code peut s'écrire $f = u \oplus Z_\mu.v$ où u et v sont des fonctions booléennes en les $(\mu{-}1)$ autres variables. Si $Z_\mu = 0$, seule la première partie intervient : il s'agit bien d'un code $RM_2(\mu{-}1, \delta)$ ayant la même dimension. Pour la seconde moitié, $Z_\mu = 1$ et la fonction v doit être de degré $(\delta{-}1)$ seulement. ∎

17. Voire, sous leur forme étendue à une base q quelconque, comme une généralisation des codes de Reed-Solomon, même si ces derniers furent découverts plus tard, voir plus loin.

Note. Notons que cette construction par doublement de la longueur peut s'appliquer *mutatis mutandis* à tout code linéaire.

Propriété VIII.3.2. *Le dual du code* $\mathrm{RM}_2(\mu,\delta)$ *est un code* $\mathrm{RM}_2(\mu,\mu-\delta-1)$.

Démonstration.
Les fonctions booléennes en μ variables identiquement nulles sur le code $\mathrm{RM}_2(\mu,\delta)$ sont celles dont tous les termes sont de degré strictement supérieur à δ. Comme $\sum_{i=\delta+1}^{\mu}\binom{\mu}{i}=\sum_{i=0}^{\mu-\delta-1}\binom{\mu}{\mu-i}=\sum_{i=0}^{\mu-\delta-1}\binom{\mu}{i}$, on a la propriété en « supprimant » ces δ variables nulles. ∎

Propriété VIII.3.3. *La distance minimale d'un code* $\mathrm{RM}_2(\mu,\delta)$ *vaut* $2^{\mu-\delta}$.

Démonstration. On le démontre par induction, comme dans la propriété VIII.3.1.

C'est vrai pour $\mathrm{RM}_2(\mu,0)$ (distance du code à répétition $d=n=2^{\mu-0}$) et pour $\mathrm{RM}_2(\mu,\mu)$ (distance du code trivial $d=1=2^{\mu-\mu}$).

Supposons-le vrai pour les codes $\mathrm{RM}_2(\mu-1,\delta)$ ($d=2^{\mu-\delta-1}$) et $\mathrm{RM}_2(\mu-1,\delta-1)$ ($d=2^{\mu-\delta}$). Soient f et f' deux mots de $\mathrm{RM}_2(\mu,\delta)$. si $v=v'$, $\mathrm{d_H}(f,f')=2.\,\mathrm{d_H}(u,u')=2.2^{\mu-\delta-1}=2^{\mu-\delta}$ par définition de la distance et par la propriété VIII.3.1. Sinon, $\mathrm{d_H}(f,f')=\mathrm{w_H}(f\oplus f')$. $=\mathrm{w_H}(u\oplus u')+\mathrm{w_H}(u\oplus u'\oplus v\oplus v')\geq \mathrm{w_H}(v\oplus v')=2^{\mu-\delta}$ par la propriété VI.1.6. ∎

Un code de Reed-Muller est donc un code linéaire de paramètres

$$\mathrm{RM}_2(\mu,\delta)=\left[2^{\mu},\ \sum_{i=0}^{\delta}\binom{\mu}{i},\ 2^{\mu-\delta}\right]_2$$

La méthode de correction efficace de Reed est essentiellement une méthode de *décodage par majorité*. L'idée est que si nous écrivons les relations liées aux variables ; il y a en a trop, puisqu'il y a redondance. S'il y a une erreur, on sélectionne la majorité d'équations satisfaites, ce qui amène à oublier celles faisant intervenir les composantes erronées. Nous ne décrirons toutefois pas les détails de cet algorithme ici.

VIII.4 Codes MDS

Les codes parfaits sont ceux qui atteignent la borne maximale de Hamming. Nous allons étudier ici quelques codes ayant des propriétés limites par rapport à d'autres bornes que nous avons énoncées.

On appelle « codes **MDS** » (pour *Maximum Distance Separable*) les familles de codes linéaires dont la taille, le nombre de mots du code atteint la borne supérieure de Singleton (*cf.* propriété VI.3.1 et son corollaire). Donc, pour une longueur n et une dimension k fixées, la distance minimale (ou le poids minimal) du code est la plus grande possible : $d=n-k+1$; le nombre d'erreurs que l'on peut corriger l'est donc également : $t=\left\lfloor\frac{n-k}{2}\right\rfloor$, c'est-à-dire la moitié de la redondance.

Définition (Code MDS)**.** Un code MDS est un code linéaire de paramètres $[n,k,n-k+1]_q$. Il est donc défini par les deux premiers paramètres : $\mathrm{MDS}_q(n,k)$.

De manière simplifiée, on peut dire que les codes parfaits tendent à empiler le plus possible les boules entourant leurs mots, pour un rayon de ces boules donné (ou possible), tandis que les codes MDS essaient de les espacer le plus possible, pour un nombre de mots donné. Ce sont deux types d'optimisation différents.

VIII.4.1 Propriétés des codes MDS

Ces codes MDS ont plusieurs propriétés intéressantes.

Propriété VIII.4.1. *Un code linéaire est* $\mathrm{MDS}_q(n,k)$ *si et seulement si tout sous-ensemble de k lignes de toute matrice génératrice est linéairement indépendant.*

Un code linéaire est $\mathrm{MDS}_q(n,k)$ *si et seulement si tout sous-ensemble de* $(n-k)$ *colonnes de toute matrice de contrôle est linéairement indépendant.*

Démonstration. Les colonnes de G sont l'image de la base canonique de l'espace des symboles et constitue une base du sous-vectoriel $\mathsf{K} = \mathrm{im}\,\mathsf{G}$. Considérons k composantes quelconques de ces vecteurs, c'est-à-dire k lignes quelconques de G. Toutes les combinaisons linéaires des colonnes de G sont, par définition, les mots du code. Si l'une d'entre elles s'annule sur les k composantes choisies, ce mot ne peut être à distance $(n-k+1)$ du mot nul.

De manière duale, considérons toute combinaison linéaire des lignes de H ; c'est une forme linéaire qui s'annule exactement sur les mots du code. Si l'une d'entre elles ne faisait pas intervenir $(n-k)$ composantes (*i.e.* ces $(n-k)$ colonnes de H produiraient une combinaison nulle), tout mot du code qui serait différent sur ces seules composantes serait encore dans le code et à distance $\leq n-k$.

A contrario, si $(n-k)$ colonnes de H sont toujours indépendantes, comme la dimension du code est k et que pour tout mot du code y on a $\mathsf{H}y = 0$ (c'est-à-dire que tout y est une combinaison linaire de colonnes de H qui s'annule), un tel mot y non nul doit nécessairement impliquer plus de $(n-k)$ colonnes de H, donc être de poids $> n-k$. ∎

Cette propriété a de nombreux corollaires.

Corollaire VIII.4.2. *Un code linéaire est MDS si et seulement si toute sous-matrice carrée de sa matrice de parité est inversible.*

Démonstration. Il suffit de considérer les matrices génératrice et de contrôle sous leur forme canonique. Supposons $k \leq n-k$. Si une combinaison linéaire non nulle des colonnes de G devaient s'annuler sur les $(n-k)$ lignes de P, ce mot serait de poids k. Si $k > n-k$, on raisonne sur la matrice H. ∎

Corollaire VIII.4.3. *Un code linéaire est MDS si et seulement si son code dual l'est.*

Démonstration. Les matrices G et $\mathsf{H}^\intercal$ ont la même propriété d'indépendance de lignes, l'une pour le code, l'autre pour son dual. ∎

Un raisonnement similaire à celui utiliser pour démontrer la propriété VI.3.1 (la borne de Singleton) permet d'établir la distribution des poids d'un code MDS :

Propriété VIII.4.4. *Le polynôme énumérateur des poids d'un code* $\mathrm{MDS}_q(n,k)$ *est*

$$\begin{aligned}\mathsf{W}(\mathsf{X},\mathsf{Y}) &= \mathsf{Y}^n + \sum_{i=n-k+1}^{n} \mathsf{X}^i\,\mathsf{Y}^{n-i}\binom{n}{i}(q-1)\sum_{j=0}^{i-(n-k+1)}(-1)^j\binom{i-1}{j}q^{i-(n-k+1)-j}\\ &= \mathsf{Y}^n + \sum_{i=n-k+1}^{n} \mathsf{X}^i\,\mathsf{Y}^{n-i}\binom{n}{i}\sum_{j=0}^{i-(n-k+1)}(-1)^j\binom{i}{j}\left(q^{i-n+k-j}-1\right)\end{aligned}$$

Démonstration. Si K est un code $\mathrm{MDS}_q(n,k)$, considérons le code $\mathsf{K}(\Delta)$ tronqué de δ caractères de l'ensemble d'indices Δ ($|\Delta| = \delta$), c'est-à-dire la projection formée de la restriction aux mots du code qui s'annulaient en ces composantes. Ainsi, si $\delta < k$, il reste un code qui a les paramètres d'un code $\mathrm{MDS}_q(n-\delta, k-\delta) = [n-\delta, k-\delta, d]$, sinon c'est un code nul. S'il est non nul, K_δ contient $q^{k-\delta}$ mots, sinon le seul mot 0.

Définissons N_δ comme le nombre de mots de K de poids au plus $(n-\delta)$ dans K. On a donc $N_\delta = \sum_\Delta |K(\Delta)|$, pour tous les ensembles d'indices Δ de taille δ possibles, puisque les mots de $K(\Delta)$ sont nuls sur ces indices, par construction. On trouve ainsi

$$N_\delta = \begin{cases} \binom{n}{\delta} q^{k-\delta} & \text{si } \delta < k \\ \binom{n}{\delta} & \text{si } \delta \geq k \end{cases}$$

Par la méthode d'inclusion-exclusion, on a $|K| = \sum_{\delta=0}^{n} (-1)^\delta N_\delta$ et parmi eux, le nombre de mots de K ayant exactement le poids i, pour $d \leq i \leq n$ est $w_i = \sum_{\delta=n-i}^{n} (-1)^{\delta-(n-i)} \binom{\delta}{i} N_\delta = \sum_{j=0}^{i} (-1)^j \binom{n-i+j}{i} N_{n-i+j}$.

La suite de la démonstration repose sur des identités très techniques des coefficients binomiaux que nous ne développerons pas ici. ∎

À partir de ces différentes propriétés, on peut trouver immédiatement quelques exemples triviaux de codes MDS :

- Le code nul $[n, 0, ?]_q$ et son dual, le code trivial $[n, n, 1]_q$.
- Le code à répétition $[n, 1, n]_q$ et son dual, le code par somme de contrôle $[n, n-1, 2]_q$.

Et nous avons vu ci-dessus une famille de codes MDS : les codes $BCH_q(1, \delta)$ qui sont $[q-1, q-\delta, \delta]_q$, ce qui est un cas particulier de codes de Reed-Solomon (qui sont décrits ci-dessous).

En fait, comme pour les codes parfaits qui se réduisent aux codes de Hamming, à de très rares exceptions, les codes MDS semblent essentiellement correspondre aux codes de Reed-Solomon. Cette idée résulte de différentes propriétés des codes MDS.

Propriété VIII.4.5. *Les seuls codes MDS binaires sont triviaux (les 4 cas ci-dessus).*

Démonstration. Par l'absurde, supposons que nous ayons un code MDS binaire qui ne soit ni nul ni tout l'espace et soit G sa matrice génératrice sous forme canonique : elle contient k lignes constituant une matrice identité, suivies de celles d'une matrice de parité. Sélectionnons $(k-1)$ lignes parmi ces k premières et une ligne quelconque de la matrice de parité. Par la propriété VIII.4.1, cette matrice est de rang k, donc toute combinaison linéaire des lignes est non nulle. Regardons celle qui annule les $(k-1)$ composantes correspondant aux premières lignes : pour qu'elle soit non nulle, il faut que la composante non encore prise en compte intervienne et qu'elle soit non nulle. Comme il en est de même pour tout sous-ensemble de $(k-1)$ lignes parmi ces k premières, toutes les composantes de la ligne de la matrice de parité doivent être non nulles. Pour la même raison d'indépendance, il ne peut y avoir deux telles lignes. Donc, soit on n'a pas pu sélectionner $(k-1)$ lignes parce que $k = 1$ et c'est le code à répétition, soit il n'y a qu'une seule ligne dans P formée de 1, donc $k = n-1$ et c'est le code à somme de contrôle. ∎

On se ramène nécessairement au cas de codes sur un corps à q éléments avec $q > 2$, puisque dans ce cas, il y a plus d'un choix pour les coefficients dans la matrice de parité. On conjecture naturellement que la propriété est vraie pour toute base.

On peut démontrer que

Théorème VIII.4.6. *Pour tout code* $MDS_q(n, k)$*, si* $k > q$*, alors* $n \leq k+1$ *et pour* $n = k+1$*, c'est le code à somme de contrôle.*

Démonstration. Il nous faut seulement envisager le cas $n = k+1$ avec $q > 2$. Puisque le code est MDS, on sait que tout sous-ensemble de k lignes de G est linéairement indépendant. On peut choisir un changement de base de manière à ce que G comporte les k lignes $\lambda_i e_i$ (e_i vecteur de base canonique) et la ligne $\sum e_i$. (On note que les λ_i sont non nuls, mais certains différents de 1.)

Considérons une autre ligne de G. Comme ce vecteur comporte k composantes de $\mathbb{F}_q$ et que $k > q$, deux d'entre elles, x_i et x_j, sont nécessairement égales. Mais alors le sous-espace de $\ker(x_i - x_j) \subset \mathbb{F}_q^k$ de dimension $(k-1)$ contiendrait k lignes de G (ce vecteur, le vecteur $\sum e_i$ et les $(k-2)$ vecteurs e_m, $m \neq i, j$), qui seraient donc linéairement dépendantes. ∎

Et pour $n \leq k$, les seuls cas possibles sont triviaux (code nul et tout l'espace). La question ouverte porte donc sur les valeurs de $k \leq q$. Une conjecture à ce propos a été formulée, du moins sous différentes formes préalables, dès les années 1950 par Beniamino Segre initialement.

Conjecture VIII.4.7 (Codes MDS)**.** *Pour tout code* $\mathrm{MDS}_q(n,k)$ *avec* $k \leq q$*, on a* $n \leq q+1$*, en général, et* $n \leq q+2$ *si* $q = 2^\nu$ *et* $k = 3$ *ou* $k = q-1$*.*

Et comme on peut construire des exemples de codes atteignant ces bornes supérieures sur n dans ces conditions, on ne peut espérer améliorer cette conjecture, sauf à expliciter ou classer les cas $n = q+1$ ou $n = q+2$ dans les cas particuliers.

Cette conjecture est fondée sur de nombreux résultats partiels allant dans ce sens ($n \leq q+k-p$ si $q = p^\nu$, les cas $k = 2$ ou $k = q$, etc.). Un des plus récents[18] montre que cette conjecture est vérifiée pour $k < p$ si $q = p^\nu$ et que dans ce cas, les codes MDS de longueur $n = q+1$ sont des codes de Reed-Solomon (Siméon Ball, 2012 [3]).

Note. Un code de Hamming $\mathrm{Ham}_q(\varrho)$ n'est jamais un code MDS si $\varrho \geq 3$.
En effet, on a $d = 3 < \varrho + 1 = n - k + 1$. Et pour $\varrho < 3$, les codes de Hamming non triviaux sont $\mathrm{Ham}_q(2) = [q+1, q-1, 3]_q$ avec $q > 2$. Ce sont bien des codes MDS, de la famille des codes de Reed-Solomon.

VIII.4.2 Codes de Reed-Solomon

Les codes de Reed-Solomon ont été présentés par Irving Stoy Reed et Gustave « Gus » Solomon en 1960 [31], dans une version « binaire », c'est-à-dire en base $q = 2^\nu$ (sur un corps de caractéristique 2) et pour $n = q$. Nous la présentons ici sur tout corps $\mathbb{F}_q$ et pour toute longueur telle que $k \leq n \leq q$.

La méthode d'encodage originelle consiste, comme dans tout code polynomial, à considérer le message x de k symboles (des éléments de $\mathbb{F}_q$) comme un polynôme $X(Z) = \sum_{i=0}^{k-1} x_i Z^i$ de degré $\deg X < k$. L'idée de Reed et Solomon est que ce polynôme est complètement déterminé par interpolation à l'aide de ses valeurs pour k valeurs de son indéterminée Z distinctes. La redondance consiste donc à encoder plus de k valeurs[19].

Définition (Code de Reed-Solomon)**.** Le code de Reed-Solomon de dimension k et de longueur n ($k \leq n \leq q$) sur le corps $\mathbb{F}_q$, noté $\mathrm{RS}_q(n,k)$, est défini de la façon suivante. Pour coder x vu comme un polynôme X(Z), on sélectionne n éléments a_j ($0 \leq j < n$) *distincts* de $\mathbb{F}_q$; le mot codé est alors le n-uplet y avec $y_j = X(a_j)$.

Le décodage d'un mot non altéré consiste à rechercher dans $\mathbb{F}_q[Z]$ le polynôme d'interpolation passant par les points (a_j, y_j). Comme $n \geq k$, il est unique.

Propriété VIII.4.8. *Les codes de Reed-Solomon sont des codes MDS.*

Démonstration. Comme un polynôme de degré inférieur à k est univoquement déterminé par k points d'abscisses distinctes, deux mots distincts de n composantes ne peuvent coïncider qu'en $(k-1)$ au plus ; ils ont donc au moins $(n-k+1)$ composantes distinctes. ∎

La matrice génératrice du code ainsi construit est une matrice de Vandermonde :

$$G = \begin{bmatrix} 1 & a_0 & a_0^2 & \cdots & a_0^{k-1} \\ 1 & a_1 & a_1^2 & \cdots & a_1^{k-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & a_{n-1} & a_{n-1}^2 & \cdots & a_{n-1}^{k-1} \end{bmatrix}$$

Si l'on veut une forme canonique pour cette matrice génératrice, il suffit de choisir simplement X(Z) comme le polynôme d'interpolation sur les k points : (a_i, x_i). On conserve l'encodage $y_j = X(a_j)$ et on a bien $y_j = x_j$ pour $0 \leq j < k$.

18. À ma connaissance, mais c'est un sujet de recherches encore très actif et des progrès partiels sont régulièrement publiés.
19. À l'origine, Reed et Solomon proposaient toutes les valeurs possibles et dans l'ordre $\{0, \alpha, \alpha^2, \ldots \alpha^{q-1} = 1\}$ où α est un élément primitif de $\mathbb{F}_q$, avec $q = 2^\nu$, bien sûr.

Bien sûr, la définition fonctionne pour toute longueur et tout choix de valeurs distinctes, mais les deux cas $n = q$ et $n = q-1$ (*i.e.* avec $a_j \neq 0$) sont les plus fréquemment utilisés. Pour de nombreux auteurs, les codes de Reed-Solomon correspondent exclusivement au premier cas, tandis que le second est appelé code de Reed-Solomon « pointé »[20] (*punctured RS code*).

On peut en effet raisonner sur le cas maximum (et originel) $n = q$ et traiter les autres comme des codes tronqués : le code est la projection supprimant $(q-n)$ composantes. Dans ce cas $n = q$, les matrices génératrice et de contrôle ont une forme simple.

En réorganisant l'ordre des composantes, si α est un élément primitif du corps, on a :

$$G = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(k-1)} \\ \vdots & \vdots & \vdots & \alpha^{ij} & \vdots \\ 1 & \alpha^{q-1} & \alpha^{2(q-1)} & \cdots & \alpha^{(q-1)(k-1)} \end{bmatrix}$$

Lemme VIII.4.9. *Dans le corps fini* $\mathbb{F}_q$, $x \neq 1 \Rightarrow \sum_{j=1}^{q-1} x^j = 0$ *et* $x = 1 \Rightarrow \sum_{j=1}^{q-1} x^j = -1$.

Démonstration. On observe que formellement $\sum_{j=1}^{q-1} x^j = \frac{x^q - x}{x-1}$. Comme tout élément d'un corps fini est racine de l'équation de Fermat $Z^q - Z$, on a la propriété si $x \neq 1$. Par contre, dans ce dernier cas, la somme vaut « $(q-1) \times 1$ », au sens de la multiplication par un scalaire. Mais si $q = p^\nu$ (p est la caractéristique du corps), on sait que $p \times 1 = 0$, ce qui donne la propriété. ∎

Grâce à ce lemme, la matrice de contrôle est construite dans la démonstration de la propriété suivante.

Propriété VIII.4.10. *Le dual d'un code* $RS_q(q,k)$ *et un code* $RS_q(q, q-k)$.

Démonstration. La matrice de contrôle correspondant au G ci-dessus est

$$H = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & \alpha & \alpha^2 & \cdots & \alpha^{q-1} \\ 0 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(q-1)} \\ \vdots & \vdots & \vdots & \alpha^{ij} & \vdots \\ 0 & \alpha^{n-k-1} & \alpha^{2(q-1)} & \cdots & \alpha^{(n-k-1)(q-1)} \end{bmatrix}$$

En effet, si l'on calcule $M = HG$, on a $m_{ij} = \sum_{m=1}^{q-1} \alpha^{im+mj} = \sum_{m=1}^{q-1} \left(\alpha^{i+j}\right)^m = 0$ pour $0 < i < q-k$ et $0 < j < k$ en vertu du lemme précédent et parce que $2 \leq i + j \leq q - 2 \Rightarrow \alpha^{i+j} \neq 1$. Et les cas $i = 0$ ou $j = 0$ sont équivalents si l'un des deux est non nul (on ajoute un terme nul) et sont $q \times 1 = 0$ pour $i = j = 0$. ∎

Pour le cas $n = q-1$, on retire simplement la 1re ligne de G et la 1re colonne de H. On a immédiatement la propriété suivante.

Propriété VIII.4.11. *Le dual d'un code* $RS_q(q-1,k)$ *et un code* $RS_q(q-1, q-k-1)$.

Dans ce cas, on peut aussi choisir une autre méthode d'encodage : on construit un code polynomial en multipliant le polynôme X(Z) issu indifféremment de l'une ou l'autre méthode, par le polynôme générateur (de degré $(n-k)$) $G(Z) = \prod_{j=1}^{n-k} (Z - \alpha^j)$ où α est un élément primitif de $\mathbb{F}_q$. On voit immédiatement qu'il s'agit d'un code $BCH_q(1, n-k+1)$, mais il faut montrer qu'il s'agit du même code, à un isomorphisme près.

Propriété VIII.4.12. *Les codes de Reed-Solomon pour* $n = q-1$ *sont des codes BCH.*

20. Réduire la longueur d'un code peut se faire de deux façons : soit on réduit sa dimension, ce qui donne un code *tronqué* (*truncated*), soit on réduit sa redondance, ce qui donne un code *pointé* (*punctured*).

Démonstration. Un élément $Y(Z)$ du code $BCH_q(1, n - k + 1)$ est, par définition un multiple de $G(Z) = \prod_{j=1}^{n-k}(Z - \alpha^j)$. Une condition nécessaire et suffisante pour que $G(Z) \mid Y(Z)$ est que $\forall j : Y(\alpha^j) = 0$. Par une permutation des éléments y_j du code de Reed-Solomon, on a $y_j = X(\alpha^j)$, donc

$$\begin{aligned} Y(\alpha^j) &= \sum_{j=0}^{n-1} y_j\, \alpha^j \\ &= \sum_{j=0}^{n-1} X(\alpha^j)\, \alpha^j \\ &= \sum_{j=0}^{n-1} \alpha^j \left(\sum_{i=0}^{k-1} x_i \alpha^{ij} \right) \\ &= \sum_{i=0}^{k-1} x_i \left(\sum_{j=0}^{n-1} \alpha^j \alpha^{ij} \right) \\ &= \sum_{i=0}^{k-1} x_i \sum_{j=0}^{n-1} \left(\alpha^{i+1}\right)^j \\ &= 0 \end{aligned}$$

en vertu du lemme VIII.4.9, puisque $1 \leq i + 1 \leq k < q - 1$, donc $\alpha^{i+1} \neq 1$ si $k < n$ (et si $k = n$, le code est le code trivial). ∎

Note. La correspondance biunivoque entre les deux représentations (valeurs du polynôme *vs.* coefficients du polynôme) utilisée dans la démonstration ci-dessus est exactement celle de la *transformée de Fourier discrète* (DFT pour *Discrete Fourier Transform*) ou plus exactement de la *Transformée de Walsh*.

Les concepteurs de ces codes ont proposé une méthode de correction directe, méthode améliorée et connue sous le nom d'algorithme de Peterson-Gorenstein-Zierler (*cf.* [28]), mais en vertu de cette propriété VIII.4.12, on utilise généralement les méthodes de correction des codes BCH, plus récentes et bien plus efficaces. Il existe aussi une amélioration spécifique (brevetée aux USA [21] en 1986) au cas $n = q-1$ qui est l'algorithme de décodage de Berlekamp-Welch [22] qui n'utilise même pas les syndromes (nous ne la décrirons pas ici).

VIII.5 Codes de Goppa

Les codes de Goppa ne sont pas des codes MDS, mais des codes efficaces par rapport à la borne de Gilvert-Varshamov (*cf.* propriété VI.3.9). En effet, ces codes et les codes MDS en général ne le sont pas vraiment : le débit d'un code $MDS_q(n, k)$ est $R = \frac{k}{n}$ et sa *distance relative* $\mathcal{D} = \frac{d}{n} = \frac{n-k+1}{n}$. Donc, ces deux valeurs sont liées par $R + \mathcal{D} = 1 + \frac{1}{n}$; si l'une croît, l'autre décroît de manière égale. Or l'idéal serait de pouvoir les faire croître toutes les deux, comme le suggère le second théorème de Shannon (théorème VI.2.2), en dépassant la borne GV asymptotique (voir théorème VI.3.10).

Est-ce possible ? Cette question est restée longtemps ouverte. En 1977 [24], MacWilliam et Sloane ont montré qu'aucun code BCH (et en particulier les codes de Reed-Solomon) ne peut atteindre de bonnes performances, notamment la borne GV. Par contre, en 1982 Цфасман (Tsfasman), Vlăduţ et Zink [39] ont exhibé une famille de codes, et parmi eux les codes de Goppa, qui satisfont cette propriété.

Déjà en 1970 ([16] et [17]), Валерий Денисович Гоппа (Valery Denisovitch Goppa) avait proposé une nouvelle méthode de codage qui généralise les codes de Reed-Solomon. Ceux-ci, en effet, codent un polynôme $X(Z)$ en l'évaluant en n éléments distincts de $\mathbb{F}_q$, c'est-à-dire en n points d'une *droite* (un

21. US Patent №4,633,470 : "Error Correction of Algebraic Block Codes".
22. Lloyd R. Welch, pas Terry A. Welch connu pour son algorithme de compression.

sous-espace unidimensionnel) sur $\mathbb{F}_q$. Les codes de Goppa évaluent ce polynôme en n points rationnels[23] distincts d'une *courbe algébrique*[24] *projective non singulière*[25] sur $\mathbb{F}_q$; une droite est donc un simple cas particulier.

La description de ces objets mathématiques et de leurs propriétés sort des limites de ce document. Mais l'intérêt de ces codes dépasse largement ce contexte : les codes de Goppa binaires sont par exemple à la base de la méthode de chiffrement de McEliece.

Vu leur grande similarité, les codes de Goppa se décodent de manière semblable aux codes de Reed-Solomon et BCH : algorithmes de Berlekamp-Massey, de Berlekamp-Welch ou de Peterson.

23. Un point rationnel d'une courbe (ou d'une variété) est un point dont les coordonnées sont dans le corps.
24. Une courbe est une variété unidimensionnelle. Une variété est *algébrique* si les coordonnées de ses points satisfont un ensemble fini d'équations polynomiales.
25. C'est-à-dire sans point singulier (point multiple, point de rebroussement, point anguleux, etc.), donc avec une tangente parfaitement définie en tout point.

Bibliographie

[1] Abu-Mostafa, Yaser Said & McEliece, Robert James, “Maximal Codeword Lengths in Huffman Codes”, *TDA Progress Report*, **42**(110), 188–193 (Aug. 1992).

[2] Adámek, Jiří, *Foundations of Coding*, 352 pp. John Wiley (Mar. 1991).

[3] Ball, Siméon, “On sets of vectors of a finite vector space in which every subset of basis size is a basis”, *J. Eur. Math. Soc.*, **14**(3), 733–748 (2012).

[4] Bavaud, François, Chappelier, Jean–Cédric, Kohlas, Jürg, *Introduction à la Théorie de l’Information et ses applications*, École Polytechnique Fédérale de Lausanne (EPFL), (Jan. 2008).
En ligne : <http://icwww.epfl.ch/ chappeli/it/courseFR/index.php?shownav=yes>,

[5] Bose, Raj C. & Ray-Chaudhuri, Dwijendra K., “On a Class of Error Correcting Binary Group Codes”, *Inform. Comput.*, **3**(1), 68–79 (Mar. 1960)

[6] Chien, Robert T., “Cyclic Decoding Procedures for the Bose-Chaudhuri-Hocquenghem Codes”, *IEEE Trans. Inf. Theory*, **IT-10**(4), 357–363 (Oct. 1964)

[7] Faddeev, D. K. (Фаддеев, Д. К.), “On the Concept of Entropy of a Finite Probabilistc Scheme” (“К понятию энтропии конечной вероятностной схемы”), *Uspekhi Mat. Nauk* (*Успехи Математических Наук*), **11**,Vol.1(67) 227–231 (Jan–Feb. 1956).

[8] Faller, Newton, “An Adaptive System for Data Compression”, in *Record of the 7th Asilomar Conference on Circuits, Systems and Computers*, *IEEE Press* 593–597 (1973).

[9] Feinstein, Amiel, “A New Basic Theorem of Information Theory”, *IEEE Trans. Inf. Theory*, **IT-4**(4), 2–22 (Sep. 1954).

[10] Forney, George D., Jr., “On Decoding BCH Codes”, *IEEE Trans. Inf. Theory*, **IT-11**(4), 549–557 (Oct. 1965).

[11] Franklin, Joël N. & Golomb, Solomon Wolf, “A function-theoretic approach to the study of nonlinear recurring sequences”, *Pacific J. Math.*, **56**(2), 455–468 (Dec. 1975).

[12] Gallager, Robert G., “Variations on a Theme by Huffman”, *IEEE Trans. Inf. Theory*, **IT-24**(6), 668–674 (Nov. 1978).

[13] Gilbert, Edgar N., “A Comparision of Signaling Alphabets”, *Bell Syst. Tech. J.*, **31**(3), 504–522 (May 1952).

[14] Golay, Marcel J. E., “Notes on Digital Coding”, *Proc. IRE* (devenu *Proc. IEEE*), **37**, 657 (June 1949).

[15] Golomb, Solomon Wolf, “Sources which Maximize the Choice of a Huffman Coding Tree”, *Information and Control*, **45**, 263–272 (1980).

[16] Goppa, Valery D. (Гоппа, Валерий Д.), “A New Class of Linear Error Correcting Codes” (“Новый класс линейных корректирующих кодов”), *Problems of Information Transmission* (*Проблемы передачи информации*), **6**(3), 24–30 (Sep. 1970).

[17] Goppa, Valery D. (Гоппа, Валерий Д.), “Rational Representation of Codes and (L,g)-Codes” (“Рациональное представление кодов и (L,g)-коды”), *Problems of Information Transmission* (*Проблемы передачи информации*), **7**(3), 41–49 (Sep. 1971).

[18] Hamming, Richard Wesley, “Error Detecting and Error Correcting Codes”, *Bell Syst. Tech. J.*, **29**(2), 147–160 (Apr. 1950).

[19] Hamming, Richard W., *Coding and Information Theory*, 239 pp. Prentice-Hall (Feb. 1980). 2d ed.: 259 pp. (1986).

[20] Hocquenghem, Alexis, “Codes correcteurs d’erreurs”, in *Revue Chiffres*, Paris, **2**, 147–156 (Sep. 1959)

[21] Huffman, David A., “A Method for the Construction of Minimum-Redundancy Codes”, *Proc. IEEE*, **40**(9), 1098–1101 (Sep. 1952).

[22] Knuth, Donald E., “Dynamic Huffman Coding”, *J. Algorithms*, **6**, 163–180 (June 1985).

[23] MacWilliams, F. Jessie, “A Theorem on the Distribution of Weights in a Systematic Code”, *Bell Syst. Tech. J.*, **42**(1), 79–94 (Jan. 1963).

[24] MacWilliams, F. Jessie & Sloane, Neil J. A., *The Theory of Error-Correcting Codes*, Vol.I : 369pp, Vol.II : 393 pp. Elsevier / North-Holland (1977).

[25] Massey, James Lee, “Shift-register synthesis and BCH decoding”, *IEEE Trans. Inf. Theory*, **IT-15**(1), 122–127 (Jan. 1969).

[26] Meggitt, John E., “Error Correcting Codes and their Implementation for Data Transmission Systems”, *IEEE Trans. Inf. Theory*, **IT-7**(5), 234–244 (Oct. 1961).

[27] Muller, David E., “Application of Boolean Algebra to Switching Circuit Design and to Error Detection”, *IEEE Trans. on Computers*, **EC-3**(3), 6–12 (Sep. 1954).

[28] Peterson, W. Wesley, “Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes”, *IEEE Trans. Inf. Theory*, **IT-6**(4), 459–470 (Sep. 1960).

[29] Peterson, W. Wesley & Brown, David. T., “Cyclic Codes for Error Detection”, *Proc. IEEE*, **49**(1), 228–235 (Jan. 1961).

[30] Reed, Irving S., “A Class of Multiple-Error-Correcting Codes and the Decoding Scheme”, *IEEE Trans. Inf. Theory*, **IT-4**(4), 38–49 (Sep. 1954).

[31] Reed, I. S. & Solomon, G., “Polynomial Codes Over Certain Finite Fields”, *SIAM J. Appl. Math.* **8**(2), 300–304 (June 1960).

[32] Shannon, Claude E., “A Mathematical Theory of Communication”, *Bell Syst. Tech. J.*, **27**(3), 379–423 (Jul. 1948).

[33] Shannon, Claude E., “Communication Theory of Secrecy System”, *Bell Syst. Tech. J.*, **27**(4), 656–715 (Oct. 1948).

[34] Shannon, Claude E. & Weaver, Warren, *The Mathematical Theory of Communication*, Vol.1, 132 pp. The University of Illinois Press, Urbana (Sep. 1949).

[35] Singleton, Richard Collom, “Maximum Distance q-nary Codes”, *IEEE Trans. Inf. Theory*, **IT-10**(2), 116–118 (Apr. 1964).

[36] Storer, James Andrew & Szymanski, Thomas Gregory, “Data compression via textual substitution”, *J. ACM*, **29**(4), 928–951 (Oct. 1982).

[37] Sugiyama, Yasuo, Kasahara, Masao, Hirasawa, Shigeichi & Namekawa, Toshihiko, “A Method for Solving Key Equation for Decoding Goppa Codes”, *Information and Control*, **27**(1), 87–99 (Jan. 1975).

[38] Tietäväinen, Aimo, “On the Nonexistence of Perfect Codes over Finite Fields”, *SIAM J. Appl. Math.* **24**(1), 88–96 (Jan. 1973).

[39] Tsfasman, Mikhail Anatolievitch, Vlăduţ, Sergei Georgievitch & Zink, Thomas, “Modular curves, Shimura curves, and Goppa codes, better than the Varshamov-Gilbert bound”, *Math. Nachr.* (*Mathematische Nachrichten*), **109**, 21–28 (1982).

[40] van Lint, Jacobus Hendricus, *Introduction to Coding Theory*, 174 pp. Springer-Verlag (1982). 2d ed. : 234 pp. (1992).

[41] van Lint, Jacobus Hendricus, “Nonexistence Theorems for Perfect Error-Correcting-Codes”, in *Computers in Algebra and Number Theory*, Vol.IV, *SIAM-AMS Proceedings* 89–95 (1971).

[42] Varshamov, Rom R. (Варшамов, Ром Р.), “Estimate of the Number of Signals in Error Correcting Codes” (“Оценка числа сигналов в кодах с коррекцией ошибок”), *Dokl. Akad. Nauk SSSR* (*Доклады Академии Наук СССР*), **117**(5), 739–741 (1957).

[43] Vitter, Jeffrey Scott, “Design and Analysis of Dynamic Huffman Codes”, *J. ACM*, **34**(4), 825–845 (Oct. 1984).

[44] Vitter, Jeffrey Scott, “ALGORITHM 673 Dynamic Huffman Coding”, *ACM Transactions on Mathematical Software (TOMS)*, **15**(2), 158–167 (June 1989).

[45] Welch, Terry A., “A Technique for High-Performance Data Compression”, *Computer*, **17**(6), 8–19 (June 1984).

[46] WOLFOWITZ, Jacob, “Strong Converse of the Coding Theorem of Semi-continuous Channels”, *Illinois J. Math.*, **3**(4), 477–489 (Dec. 1959).

[47] ZIV, Jacob & LEMPEL, Abraham, “A Universal Algorithm for Sequential Data Compression”, *IEEE Trans. Inf. Theory*, **IT-23**(3), 337–343 (May 1977).

[48] ZIV, Jacob & LEMPEL, Abraham, “Compression of Individual Sequences via Variable-Rate Coding”, *IEEE Trans. Inf. Theory*, **IT-24**(5), 530–536 (Sep. 1978).

Table des matières