Contents

Chapter 4

Codes Correcteurs d'Erreurs

Nous passons au dernier sujet du cours. Nous allons voir comment il est possible de *protéger* l'information sauvegardée sur un disque des erreurs introduites soit pas les changements de l'état physique (tels que les rayures sur le disque, les particules de poussière qui bloquent la lecture du disque par laser,...) soit les changements produits naturellement (par exemple, le bruit thermique introduit par un système électrique). Nous allons accomplir la protection en introduisant les bits de *redondance*. Ces bits de redondance peuvent être utilisés pendant le *processus de décodage* afin de reconstruire l'information initiale même s'il y a des bits perdus ou erronés.

Pour des raisons de complexité, il est utile d'introduire les bits de redondance d'une manière linéaire. Ceci nécessite la connaissance de l'algèbre linéaire et moderne. Avant de commencer de travailler sur le problème, rappelons quelques notions basiques d'algèbre.

4.1 Quelques notions utiles de l'algèbre linéaire/moderne

4.1.1 Corps

Vous êtes tous familiers avec les manipulations algébriques sur les nombres réels ($Reals$) où complexes ($Comps$). Tous les deux sont des *corps*. En gros, ceci veut dire que vous pouvez utiliser toutes les opérations connues (addition, multiplication et leurs inverses, soustraction et division). Plus formellement, un corps F est un ensemble avec deux opérations basiques, $+$ (addition) et $*$ (multiplication), qui suivent les règles suivantes (rappelez la notion d'un *corps* que nous avons discuté dans Section **??**):

	$+$	$*$
fermeture: $\forall a, b \in F$	$a + b \in F$	$a * b \in F$
associativité: $\forall a, b, c \in F$	$a + (b + c) = (a + b) + c$	$a * (b * c) = (a * b) * c$
identité: $\exists 0, 1 \in F, 0 \neq 1,$ tel que	$0 + a = a$	$1 * a = a$
inverse: $\forall a \in F, \exists (-a), a^{-1} \in F$ tel que	$a + (-a) = 0$	$a * (a^{-1}) = 1, a \neq 0$
commutativité: $\forall a, b \in F$	$a + b = b + a \in F$	$a * b = b * a \in F$

En plus, nous avons la loi distributive: pour tous $a, b, c \in F$, $a * (b + c) = a * b + a * c$.

L'information est supposée d'être discrète (un ensemble de bits). Il nous est donc naturel de travailler avec les *corps finis*, c'est-à-dire les corps ayant un nombre fini d'éléments. Les corps finis les plus simples sont les corps F_p, ou p est un nombre premier. Les corps finis sont aussi appelés les corps de *Galois*, à l'honneur de Jean-Baptiste Galois. Comme on sait, Galois a noté la théorie basique des corps finis (et bien plus) durant les deux semaines avant le duel qui a mis la fin à sa vie à l'âge de 21 ans (elle s'appelait Stéphanie).

Exemple 1 (Corps F_p)**.** *Nous avons déjà rencontré le corps F_p dans Section* **??***, ou il a été noté comme $\mathbb{Z}_p$. C'est un ensemble $\{0, \cdots, p-1\}$ sous l'arithmétique standard sur les entiers, mais faite modulo p. Pour montrer que $\mathbb{Z}_p$ forme vraiment un corps, nous avons besoin de vérifier toutes les propriétés présentées ci-dessus. La plupart d'elles se vérifient directement et sont juste les propriétés de l'arithmétique sur les entiers. La seule propriété qui nécessite un*

peu de réflection est celle de l'existence de l'inverse par rapport à la multiplication. Répétons ici l'argument qui montre pourquoi l'inverse existe. Soit $a \in \mathbb{Z}_p$, $a \neq 0$. Alors, par l'algorithme euclidien étendu il existe deux entiers α et π tels que

$$a\alpha + p\pi = \gcd(a, p) = 1,$$

où nous utilisons le fait que p est un nombre premier et que $0 < a < p$, donc $\gcd(a, p) = 1$. Si nous prenons cette relation modulo p, nous obtenons $a\alpha = 1$. Autrement dit, α est l'inverse multiplicatif de a, $a^{-1} = \alpha$.

Exemple 2 (Corps F_2)**.** *Le corps* binaire *joue un rôle très important. F_2 contient seulement deux éléments, notamment 0 et 1.*

Un autre moyen de représenter F_2 est de voir deux éléments 0 et 1 comme des valeurs logiques. Dans cette représentation, l'addition correspond à l'opération XOR, tandis que la multiplication correspond à faire AND. C'est présenté dans les tableaux suivants. Une telle représentation est très utile parce que ces opérations peuvent être implémentées sur ordinateur d'une manière efficace.

$+$	0	1
0	0	1
1	1	0

$*$	0	1
0	0	0
1	0	1

Le corps F_2 a une propriété intéressante. Si $a \in F_2$, alors $a + a = 0$. Et, au contraire, $a = -a$. Ceci dit, l'addition et la soustraction sont les mêmes et donc les signes ne jouent pas d'importance.

Tout ce que vous avez étudiés sur l'algèbre linéaire (la solution des systèmes d'équations en utilisant les matrices, la condition d'avoir une solution unique en termes du rang de matrice,...) reste valide si l'on utilise un corps fini F_2 au lieu de $\mathbb{R}$ ou $\mathbb{C}$.

Exemple 3 (Système d'équations sur F_7)**.** *Considérons un système d'équations*

$$\begin{aligned} 3x_1 + 4x_2 &= 2, \\ x_1 + 2x_2 &= 0, \end{aligned}$$

ou toutes les variables sont des éléments de F_7. Pour le résoudre, nous procédons exactement comme si les équations étaient sur $\mathbb{R}$. Nous récrivons le système sous forme matricielle comme $\boldsymbol{A}x^T = (2, 0)^T$, où

$$\boldsymbol{A} = \begin{pmatrix} 3 & 4 \\ 1 & 2 \end{pmatrix}, \qquad \boldsymbol{x} = (x_1, x_2).$$

Nous savons que ce système a une solution unique si $\boldsymbol{A}$ est de rang plein. Cela peut être vérifié par le calcul du déterminant de $\boldsymbol{A}$. Dans notre cas, nous avons $|\boldsymbol{A}| = 3 \cdot 2 - 1 \cdot 4 = 2 \neq 0$, toutes les opérations s'étant effectuées sur F_7. Nous obtenons

$$\boldsymbol{A}^{-1} \overset{(i)}{=} \frac{\begin{pmatrix} \boldsymbol{A}_{22} & -\boldsymbol{A}_{12} \\ -\boldsymbol{A}_{21} & \boldsymbol{A}_{11} \end{pmatrix}}{|\boldsymbol{A}|} = \frac{\begin{pmatrix} 2 & -4 \\ -1 & 3 \end{pmatrix}}{|\boldsymbol{A}|} = \frac{\begin{pmatrix} 2 & 3 \\ 6 & 3 \end{pmatrix}}{2} \overset{(ii)}{=} \begin{pmatrix} 2 & 3 \\ 6 & 3 \end{pmatrix} 4 = \begin{pmatrix} 1 & 5 \\ 3 & 5 \end{pmatrix}.$$

Sur l'étape (i) nous avons utilisé la règle de Cramer pour calculer l'inverse. Elle est facilement vérifiée: si $|\boldsymbol{A}| \neq 0$*, alors*

$$\frac{\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix}\begin{pmatrix} \boldsymbol{A}_{22} & -\boldsymbol{A}_{12} \\ -\boldsymbol{A}_{21} & \boldsymbol{A}_{11} \end{pmatrix}}{|\boldsymbol{A}|} = \frac{|\boldsymbol{A}|\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}{|\boldsymbol{A}|} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Sur l'étape (ii) nous avons utilisé le fait que l'inverse multiplicatif de 2 *est* 4 *(car* $2 * 4 = 8 = 1 \mod 7$*). Alors, au lieu de diviser par* 2*, nous pouvons multiplier par* 4*. La solution est donc* $\boldsymbol{x}^T = \boldsymbol{A}^{-1}(2,0)^T = (2,6)$.

4.1.2 Espace vectoriel

Rappelons quelques notions basiques d'algèbre.

Soit $\mathcal{V}$ un groupe commutatif pour l'addition (voir Section **??**) dont les éléments $\boldsymbol{u}, \boldsymbol{v}, \ldots$ sont appelés *vecteurs* et soit $\mathcal{F}$ un champ dont les éléments $a, b, \ldots$ sont appelés *scalaires*. L'ensemble $\mathcal{V}$ est un *espace vectoriel* sur $\mathcal{F}$ si le produit d'un vecteur par un scalaire possède les propriétés suivantes: pour tout $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{V}$ et tout $a, b \in \mathcal{F}$,

- fermeture: $a\boldsymbol{v} \in \mathcal{V}$
- associativité pour la multiplication scalaire: $a \cdot (b\boldsymbol{v}) = (ab)\boldsymbol{v}$
- identité: $1 \cdot \boldsymbol{v} = \boldsymbol{v}$
- distributivité: $a(\boldsymbol{u} + \boldsymbol{v}) = a\boldsymbol{u} + a\boldsymbol{v}$ et $(a + b)\boldsymbol{u} = a\boldsymbol{u} + b\boldsymbol{u}$

L'exemple le plus connu de l'espace vectoriel est l'ensemble $\mathbb{R}^n$ des n-uplets de réels. En effet on vérifie aisément que $\mathcal{V} = \mathbb{R}^n$ est un groupe commutatif pour l'addition, et que $\mathcal{F} = \mathbb{R}$ est un champ et que les propriétés citées plus haut (fermeture, associativité, identité, distributivité) sont satisfaites.

Un sous-ensemble $\mathcal{S}$ de $\mathcal{V}$ est un *sous-espace vectoriel*, si il est aussi un espace vectoriel sur $\mathcal{F}$, *i.e.,* un group commutatif qui satisfait les propriétés ci-dessus. $\mathcal{S}$ étant un sous-ensemble de $\mathcal{V}$, qui est par définition un espace vectoriel, les propriétés d'associativité, d'identité et de distributivité sont directement vérifiées; il n'est donc nécessaire que de vérifier la propriété suivante:

$$a\boldsymbol{u} - \boldsymbol{v} \in \mathcal{S} \text{ pour tout } a \in \mathcal{F} \text{ et } \boldsymbol{u}, \boldsymbol{v} \in \mathcal{S}.$$

Si cette propriété est satisfaite, il est facile de montrer que $\mathcal{S}$ est un groupe commutatif: prenons $\boldsymbol{u} = \boldsymbol{v}$ et $a = 1$, ce qui nous donne l'existence de l'élément neutre pour l'addition; le fait qu'il existe un inverse additif découle du cas $\boldsymbol{u} = \boldsymbol{0}$; en prenant $a = 1$, nous pouvons montrer la propriété de fermeture par rapport a l'addition; finalement, nous pouvons montrer la fermeture par rapport à la multiplication en choisissant $\boldsymbol{v} = 0$.

Exemple 4. *Nous savons déjà que* $\mathbb{R}^2$ *est un espace vectoriel. Montrons que l'ensemble de vecteurs* $\mathcal{S} = (u, 3u)$ *forme un sous-espace.*

$$\begin{aligned} a \cdot (u, 3u) - (v, 3v) &\overset{?}{\in} \mathcal{S} \\ (au - v, 3au - 3v) = (au - v, 3(au - v)) &\overset{\checkmark}{\in} \mathcal{S} \end{aligned}$$

Ceci confirme d'abord que $\mathcal{S}$ *muni de l'addition est un groupe commutatif et, en plus, que* $\mathcal{S}$ *est est un espace vectoriel sur* F*.*

4.1.3 Espace métrique

Nous avons besoin plus que d'un espace vectoriel. Nous avons aussi besoin d'une notion de la *distance* (ou *métrique*).

Un *espace métrique* $(\mathcal{V}, d(\cdot, \cdot))$ est un espace vectoriel $\mathcal{V}$ sur lequel on définit une *distance* (ou *métrique*) $d(\boldsymbol{u}, \boldsymbol{v})$ entre deux vecteurs $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{V}$ qui est un réel positif satisfaisant les trois axiomes suivants: pour tout $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \mathcal{V}$,

- $d(\boldsymbol{u}, \boldsymbol{v}) \geq 0$ et $d(\boldsymbol{u}, \boldsymbol{v}) = 0$ si et seulement si $\boldsymbol{u} = \boldsymbol{v}$
- symétrie: $d(\boldsymbol{u}, \boldsymbol{v}) = d(\boldsymbol{v}, \boldsymbol{u})$
- inégalité du triangle: $d(\boldsymbol{u}, \boldsymbol{v}) \leq d(\boldsymbol{u}, \boldsymbol{w}) + d(\boldsymbol{w}, \boldsymbol{v})$.

Exemple 5. $\mathbb{R}^2$ *muni de la distance euclidienne est un espace métrique, dans lequel, si* $\boldsymbol{u} = (u_1, u_2)$ *et* $\boldsymbol{v} = (v_1, v_2)$*,* $u_1, u_2, v_1, v_2 \in \mathbb{R}$*,*

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

4.1.4 Combinaison linéaire et la notion de la base

Soit $\mathcal{V}$ un espace vectoriel et $\{\boldsymbol{v}_i\}_{i=1}^m$ un ensemble des vecteurs de $\mathcal{V}$. Une *combinaison linéaire* de vecteurs $\boldsymbol{v}_i \in \mathcal{V}$ est la somme suivante

$$\boldsymbol{u} = \sum_{i=1}^{m} a_i \boldsymbol{v}_i, \tag{4.1}$$

où les coefficients $a_i \in \mathcal{F}$. On peut montrer que l'ensemble des vecteurs $\boldsymbol{u}$ engendrés par toutes les combinaisons linéaires de m vecteurs $\boldsymbol{v_i}$ forme un sous-espace vectoriel. On appellera ce sous-espace vectoriel $span(\boldsymbol{v_i})$.

Les vecteurs $\boldsymbol{v_i}$ sont *linéairement indépendants* si et seulement si $\boldsymbol{u} = \sum_{i=1}^m a_i \boldsymbol{v_i} = \boldsymbol{0}$ entraîne que tous les coefficients a_i sont nuls. Dans ce cas, la représentation (4.1) est unique: il n'y a qu'un seul ensemble de coefficients $\{a_i\}$ qui permette d'écrire le vecteur $\boldsymbol{u}$ sous cette forme.

Les vecteurs $\boldsymbol{v_i}$ *engendrent* l'espace vectoriel $\mathcal{V}$ si et seulement si tout vecteur de $\mathcal{V}$ peut s'écrire comme une combinaison linéaire de ces vecteurs.

L'ensemble des vecteurs $\boldsymbol{v_i}$ est une *base* de l'espace vectoriel $\mathcal{V}$ si les vecteurs sont linéairement indépen-dants et engendrent l'espace vectoriel $\mathcal{V}$. Le nombre m d'éléments d'une base est la *dimension* de cet espace vectoriel. Si un ensemble comporte plus de m vecteurs, ceux-ci sont nécessairement linéairement dépendants.

Exemple 6. *Nous pouvons voir que l'ensemble de vecteurs suivant, pris dans* $\mathbb{R}^3$*,*

$$\{(1,0,0),\ (0,1,0),\ (0,0,1)\}$$

est linéairement indépendant, car toute combinaison linéaire sera de la forme

$$(a_1 + a_2 \cdot 0 + a_3 \cdot 0, a_1 \cdot 0 + a_2 + a_3 \cdot 0, a_1 \cdot 0 + a_2 \cdot 0 + a_3) = (a_1, a_2, a_3)$$

et que les composantes ne peuvent être nulles que si $a_1 = a_2 = a_3 = 0$*.*

Nous pouvons voir que ces trois vecteurs engendrent $\mathbb{R}^3$ *car n'importe quel vecteur* $\boldsymbol{u} = (u_1, u_2, u_3)$ *peut être écrit comme leur combinaison linéaire en choisissant* $a_1 = u_1$*,* $a_2 = u_2$ *et* $a_3 = u_3$*. Cet ensemble de vecteurs, ́tant linéairement independant et engendrant* $\mathbb{R}^3$*, forme donc une base de* $\mathbb{R}^3$*. Ce qui signifie que* $\mathbb{R}^3$ *est de dimension 3.*

Nous pouvons facilement voir que le rajout d'un quatrième vecteur à cet ensemble le rendrait linéairement dépendant, car ce quatrième vecteur peut toujours être écrit comme une combinaison linéaire des trois premiers. Par exemple,

$$(3,2,0) = 3 \cdot (1,0,0) + 2 \cdot (0,1,0).$$

4.1.5 Espace vectoriel F^n et la métrique de Hamming

Nous nous intéressons aux espaces vectoriels sur un champ fini F. Nous considérons un ensemble de vecteurs $\mathcal{V}$ formé par les vecteurs $\boldsymbol{v} = (v_1, \ldots, v_n)$, ou $v_i \in F$. Tous nos vecteurs sont les *vecteurs-lignes*. Nous allons voir dans Exercice 1 que $\mathcal{V}$ est un espace vectoriel. Il est clair que sa dimension est n car il possède la base orthogonale des n vecteurs $(1,0,\ldots,0)$, $(0,1,\ldots,0)$, $\ldots$, $(0,0,\ldots,1)$.

On définit le *poids de Hamming* ou *poids de* $\boldsymbol{v}$ comme

$$w(\boldsymbol{v}) = |\{i : v_i \neq 0\}|.$$

En d'autres termes, le poids de Hamming d'un vecteur $\boldsymbol{v}$ est égal au nombre des "1" dans le n-uplet $\boldsymbol{v} = (v_1, \ldots, v_n)$.

La distance entre deux vecteurs $\boldsymbol{u}$ et $\boldsymbol{v}$, appelée la *distance de Hamming*, est le nombre de positions dans lesquelles les composantes u_i et v_i diffèrent:

$$d(\boldsymbol{u}, \boldsymbol{v}) = w(\boldsymbol{u} - \boldsymbol{v}) = \sum_{i=1}^{n} (u_i \oplus v_i).$$

Exemple 7 ($F = F_2$)**.** *Comme* $\boldsymbol{u} = (1,0,1,1,1,0)$ *et* $\boldsymbol{v} = (1,0,0,1,1,1)$ *ne diffèrent qu'en leur troisième et dernière positions, leur distance de Hamming est 2. En effet,*

$$d(\boldsymbol{u},\boldsymbol{v}) = w(\boldsymbol{u}-\boldsymbol{v}) = w(\ (0,0,1,0,0,1)\ ) = 0+0+1+0+0+1 = 2.$$

Remarque: notez que les 0 *et les* 1 *ont le double sens dans cette équation. Dans l'expression* $w(\ (0,0,1,0,0,1)\ )$ *les* 0 *et les* 1 *sont les éléments de* F_2*, tandis que dans l'expression* $0+0+1+0+0+1$ *les* 0 *et les* 1 *sont considérés comme les entiers! Notez aussi que, pour* $F = F_2$*,* $d(\boldsymbol{u},\boldsymbol{w}) = \sum_{i=1}^{n} |u_i - v_i|$.

Vous montrerez dans Exercice 1 que $d(\cdot,\cdot)$ est la distance comme elle est définie dans Section 4.1.3. $\mathcal{GF}(2)^n$ est donc l'espace métrique comme c'est défini dans Section 4.1.3.

4.2 Qu'est-ce que c'est un code?

Dans la plupart des exemples qui suivent nous utilisons $F = F_2$, le corps à deux éléments, mais toutes les formulations données se tiennent en cas général.

Commençons par un exemple.

Exemple 8. *Considérons un code en blocs de* longueur $n = 7$ *sur* $F = F_2$ *ayant* 8 mots de code.

$k=3$		$n=7$	$w(x)$
000	$\mapsto$	0000000	0
001	$\mapsto$	0011100	3
010	$\mapsto$	0111011	5
011	$\mapsto$	0100111	4
100	$\mapsto$	1110100	4
101	$\mapsto$	1101000	3
110	$\mapsto$	1001111	5
111	$\mapsto$	1010011	4

Supposons, nous avons besoin de sauvegarder un fichier. Par simplicité, regardons un fichier très court contenant trois bits seulement. Bien sûr, un fichier réel serait beaucoup plus grand.

Nous ne connaissons pas a priori quel contenu ce fichier peut avoir. Nous avons besoin de prévoir toutes les possibilités. Il y a $2^3 = 8$ *fichiers possibles avec* 3 *bits:* 000, 001, $\cdots$, 111. *A chacun des* 8 *vecteurs d'information possibles qui représentent les réalisations du fichier possibles, nous mettons en correspondance un* mot de code *distinct. Dans notre cas, les mots de code ont la longueur* $n = 7$*. Cela signifie que nous étendons la longueur du fichier de* 3 *à* 7*. Nous espérons pouvoir utiliser l'information* redondante *qui se trouve dans le mot de code ensuite, quand nous voudrons reconstruire le fichier original d'une observation (probablement) altérée. La représentation mettant en correspondance l'information (le fichier) et le mot de code est appelé* l'image d'encodage.

Le code $\mathcal{C}$ *est l'ensemble de tous les mots de code. Nous disons que* $\mathcal{C}$ *est un code en blocs, si nous mettons en correspondance les blocs de données et les blocs de mots de code. Nous disons que le*

code est de longueur $n = 7$. *Nous pouvons aussi dire que le code est de* rendement $r = \frac{k}{n} = \frac{3}{7}$, *parce que nous mettons en correspondance les blocs de longueur* $k = 3$ *et les blocs de longueur* $n = 7$. *Le code lui-même, aussi que l'image entre les vecteurs d'information, est fixé une fois pour toutes, ils sont connus et à la personne qui écrit les données sur le disque (émetteur) et à la personne qui lit les données (destinataire).*

Définition 1 (Code en blocs)**.** *Un* code *en blocs* $\mathcal{C}$ *de longueur* n *est un ensemble de* n*-uplets sur* F. *Le* rendement *du code est défini comme* $r = \frac{1}{n} \log_{|F|} |\mathcal{C}|$.

Discussion: pour l'exemple précédent nous avions $r = \frac{1}{7} \log_2(8) = \frac{3}{7}$, ce qui est compatible avec la définition.

Jusqu'ici nous avons vu qu'un code en blocs a les paramètres suivants: la longueur n et le rendement r. Mais comment nous pouvons choisir le code? Qu'est-ce que c'est un bon code? Nous allons voir que le paramètre très important d'un code en blocs est sa *distance minimale*.

Définition 2 (Distance minimale)**.** *La distance* minimale *d'un code en blocs* $\mathcal{C}$, *notée comme* $d_{min}(\mathcal{C})$, *est donnée par*

$$d_{min}(\mathcal{C}) = \min_{\boldsymbol{x},\boldsymbol{x'} \in \mathcal{C}; \boldsymbol{x} \neq \boldsymbol{x'}} d(\boldsymbol{x}, \boldsymbol{x'}).$$

Autrement dit, la distance minimale d'un code est le plus petit nombre des éléments différents entre deux mots de code. Comme nous allons voir par la suite, un bon code a une grande distance minimale.

Exemple 9 (Distance minimale de notre exemple précédent)**.** *Comme nous le voyons de la dernière définition, pour déterminer la distance minimale d'un code donné avec* M *mots de code, nous devons prendre le minimum sur tous les couples distincts des mots de code. C'est-à-dire, nous devons vérifier* $\binom{M}{2} = \frac{M(M-1)}{2}$ *couples. (Supposons vous êtes à une soirée et tout le monde veut serrer la main à chacun; s'il y a* M *personnes à la soirée, alors le nombre des poignées de main nécessaires est* $\binom{M}{2}$*). Pour notre exemple précédent, nous avons* $M = 8$, *et donc nous devons vérifier* $\binom{8}{2} = 28$ *couples. Ceci est encore faisable, et nous obtenons* $d_{min}(\mathcal{C}) = 3$. *Mais pour les codes plus longs la vérification directe peut ne plus être possible. Par exemple, nous allons bientôt étudier les codes de Reed-Solomon. Pour certains codes dont nous allons discuter, le nombre de mots de code est de l'ordre* $M = 256^{128}$. *Evidemment, nous devrons penser à d'autres méthodes pour déterminer la distance minimale.*

4.3 Modèles de canal

La motivation principale d'utiliser le codage est que, si vous avez sauvegardé l'information sur un disque et vous lisez cette information un certain temps après, il peut arriver, que les données lues sont différentes de celles qui ont été sauvegardées par vous. La nature du phénomène qui a provoqué un tel changement peut être arbitrairement compliquée, et nous voulons éviter sa considération. Alors nous construisons un *modèle de canal*. Ce modèle est une abstraction mathématique décrivant seulement une partie du canal physique étant importante pour nous

sans les détails non nécessaires. Nous parlons ici de deux modèles: le canal *à effacements* et le canal *symétrique*.

Définition 3 (Canal à effacements)**.** *Pour le canal à effacements, nous supposons que chaque composante du mot de code est soit connue parfaitement soit* effacée*. L'effacement signifie que le symbole transmis à une position effacée a été remplacé par un symbole spécial, disons le symbole "?". Le destinataire sait donc les positions qui ont étées effacées, et il doit retrouver l'information contenante sur ces positions. Autrement dit, le canal cache certaines parties du mot transmis.*

Exemple 10 (Canal à effacements)**.** *Supposons que le mot transmis est* $\boldsymbol{x} = (0100111)$*. Soit* $\boldsymbol{y}$ *la sortie du canal (voir Figure* **??***). Supposons que* $\boldsymbol{y} = (0?001?1)$*: les bits effacés du canal sont* 2 *et* 6*. On dit que le pattern d'effacements est de* poids 2*.*

Définition 4 (Canal symétrique)**.** *Pour le canal symétrique, chaque composante du mot de code est soit reçue parfaitement soit est échangée pour un des autres* $|F|-1$ *symboles, uniformément. C'est-à-dire, il y a des positions qui ont changé leur valeurs. Si* $F = F_2$*, alors les valeurs sont échangés de* 0 *à* 1 *et vice versa.*

Exemple 11 (Canal symétrique)**.** *Considérons encore une fois notre exemple tout premier. Supposons que le mot de code transmis est* $\boldsymbol{x} = (0100111)$*. Soit* $\boldsymbol{y}$ *la sortie du canal. Supposons que* $\boldsymbol{y} = (0000101)$*. En la comparant avec* $\boldsymbol{x}$*, nous pouvons voir que le canal a introduit deux erreurs, notamment sur les positions* 2 *et* 6*. Notons* $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{x}$*. Alors nous avons* $\boldsymbol{e} = (0100010)$*. Nous l'appelons le* vecteur d'erreurs*. Il est non-nul exactement dans les positions sur lesquelles le canal a introduit un erreur. Nous avons* $w(\boldsymbol{e}) = 2$*, c'est-à-dire le poids de* $\boldsymbol{e}$ *est égal au nombre d'erreurs introduits par le canal. Notons aussi que* $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$*. Cela signifie que l'effet du canal se traduit par l'ajout d'un vecteur d'erreurs au mot de code.*

4.4 Principes de base de la correction d'erreurs

Maintenant nous allons voir pourquoi la distance minimale est tellement importante dans le codage.

4.4.1 Détection d'erreurs

Supposons que l'opérateur d'une station nucléaire veut d'envoyer un signal d'alarme au réacteur nuclèaire pour réduire sa puissance. Ou une situation moins dramatique, que vous voulez sauvegarder les données de votre compte bancaire sur un disque. Dans les deux cas, nous voulons assurer que le signal émis (l'information sauvegardée) est reçue (est lisible) correctement. Autrement dit, nous voulons assurer que les erreurs de transmission sont détectées.

Soit $\mathcal{C}$ le code donné. Supposons que nous choisissons un mot de code $\boldsymbol{x} \in \mathcal{C}$ et ce $\boldsymbol{x}$ est envoyé par un canal symétrique. Le destinataire obsèrve la sortie du canal appelé $\boldsymbol{y}$. Bien sûr, le destinataire *ne connaît pas* le mot de code émis $\boldsymbol{x}$. Il connaît seulement le mot reçu $\boldsymbol{y}$. Si $\boldsymbol{y} \notin \mathcal{C}$, alors le destinataire sait que le canal a introduit des erreurs. Dans ce cas, le destinataire peut

donner l'alerte pour prévenir l'utilisateur que la transmission a été éronnée. Nous disons que le destinataire peut *détecter l'erreur*.

Lemme 1 (Détection d'erreurs)**.** *Un code $\mathcal{C}$ avec la distance minimale $d_{min} = d_{min}(\mathcal{C})$ est capable de détecter toutes les erreurs du poids $t \leq d_{min} - 1$ ou moins.*

Proof. Soit $\boldsymbol{x}$ le mot transmis et $\boldsymbol{y}$ le mot reçu. Rappelons que $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$, où $\boldsymbol{e}$ est le vecteur d'erreurs. Supposons qu'il y a $d_{\min} - 1$ erreurs introduites au maximum, $w(\boldsymbol{e}) \leq d_{\min} - 1$. Alors, $d(\boldsymbol{x}, \boldsymbol{y}) = w(\boldsymbol{e}) \leq d_{\min} - 1$. Donc, $\boldsymbol{y}$ ne peut pas être un mot de code (et l'erreur peut être détectée), parce que $\boldsymbol{x}$ est un mot de code et, par définition du code, n'importe quels deux mots de code ont la distance au moins $d_{\min}$. □

Exemple 12 (Detection d'erreurs)**.** *Considérons notre premier exemple. Supposons que $\boldsymbol{x} =$ (0100111) et $\boldsymbol{e} =$ (0100010), alors $\boldsymbol{y} =$ (0000101). Nous avons $w(\boldsymbol{e}) = 2 \leq d_{min} - 1 = 2$. Par Lemme 1, nous savons que $\boldsymbol{y}$ ne peut pas être un mot de code et donc l'erreur peut être détectée. Effectivement, si nous regardons la liste de tous les mots de code, nous voyons que $\boldsymbol{y}$ n'y est pas.*

4.4.2 Correction d'effacements

Lemme 2 (Correction d'effacements)**.** *Soit $\mathcal{C}$ le code donné. Nous choisissons un mot de code $\boldsymbol{x} \in \mathcal{C}$ et l'envoyons par un canal à effacements. Soit $\boldsymbol{y}$ la sortie du canal. Si $\boldsymbol{y}$ contient pas plus que $t \leq d_{min} - 1$ effacements, alors nous pouvons reconstruire le mot de code transmis d'une manière unique.*

Proof. Le mot $\boldsymbol{y}$ avec t effacements étant donné, nous trouvons tous les mots de code $\boldsymbol{x'} \in \mathcal{C}$ qui ont les mêmes valeurs que $\boldsymbol{y}$ sur toutes les positions *connues*. Il est clair que le mot de code $\boldsymbol{x} \in \mathcal{C}$ est l'un d'eux. S'il y a un seul mot de code qui vérifie la condition, alors le mot de code $\boldsymbol{x}$ se reconstruit d'une manière unique. Si $\boldsymbol{x'}$ est un autre mot de code qui coincide avec $\boldsymbol{y}$ sur toutes les positions connues, alors $\boldsymbol{x'}$ ne doit pas se différer de $\boldsymbol{x}$ que sur t positions au maximum. Alors, il y a une contradiction, parce que $t(\boldsymbol{x}, \boldsymbol{x'}) \leq t \leq d_{\min} - 1$; mais n'importe quels deux mot de code ont la distance minimale au moins $d_{\min}$. □

Exemple 13 (Correction d'effacements)**.** *Considérons notre premier exemple. Supposons que nous avons reçu $\boldsymbol{y} =$ (0?001?1). Comme $t = 2 \leq d_{min} - 1 = 2$, alors on sait que nous pouvons reconstruire le mot de code d'une façon unique. Effectivement, en passant par la liste de mots de code nous voyons que $\boldsymbol{x} =$ (0100111) est le seul mot de code compatible avec $\boldsymbol{y}$.*

4.4.3 Correction d'erreurs

Lemme 3 (Correction d'erreurs)**.** *Soit $\mathcal{C}$ le code donné. Nous choisissons un mot de code $\boldsymbol{x} \in \mathcal{C}$ et l'envoyons par le canal symétrique. Soit $\boldsymbol{y}$ la sortie du canal. Si le canal introduit t erreurs, où $t \leq \frac{d_{min}-1}{2}$, alors il est possible de corriger les erreurs.*

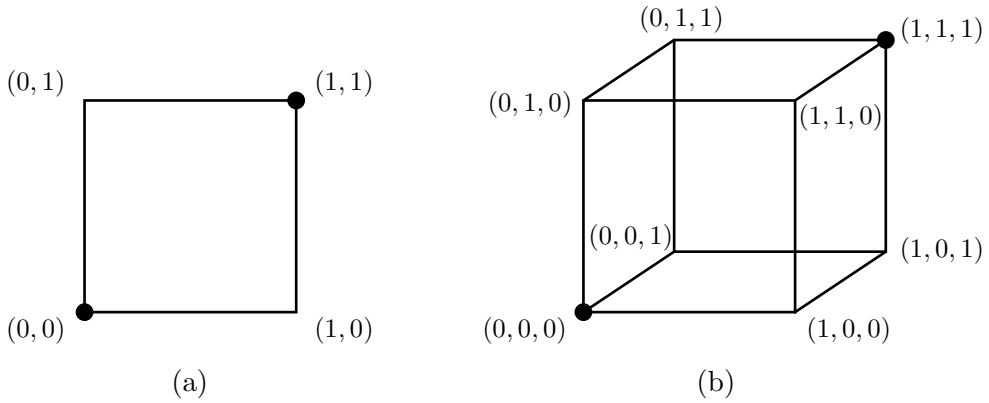


Figure 4.1: $(2, 1)$ (a) et $(3, 1)$ (b) codes de répétition

Proof. Pour corriger les erreurs nous procédons comme suit. Le mot $\boldsymbol{y}$ étant donné, nous trouvons tous les mots de code $\boldsymbol{x'} \in \mathcal{C}$ tels que $d(\boldsymbol{y}, \boldsymbol{x'}) \leq \frac{d_{\min}-1}{2}$. Par hypothèse, le mot de code $\boldsymbol{x}$ est dans cette liste. Nous persistons qu'il en existe un seul. Effectivement, s'il y avait un autre mot $\boldsymbol{x'} \in \mathcal{C}$ tel que $d(\boldsymbol{y}, \boldsymbol{x'}) \leq \frac{d_{\min}-1}{2}$, cela voudrait dire que, par l'inégalité du triangle, $d(\boldsymbol{x}, \boldsymbol{x'}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{x'}) \leq \frac{d_{\min}-1}{2} + \frac{d_{\min}-1}{2} = d_{\min} - 1$, ce qui est une contradiction. □

Exemple 14 (Correction d'erreurs)**.** *Supposons nous avons reçu le mot* $\boldsymbol{y} = (0000111)$*. Si nous passons par la liste des mots de code, nous voyons que le seul élément de* $\mathcal{C}$ *à la distance* $\frac{d_{min}-1}{2} = 1$ *de* y *est le mot* $\boldsymbol{x} = (0100111)$*. Donc, nous déclarons* $\boldsymbol{x}$ *d'être le mot transmis. Si une seule erreur a été introduite par le canal, nous avons décodé correctement.*

De notre discussion précédente, il paraît claire que le problème principal de la théorie de codage est de construire les codes qui auraient à la fois un grand nombre de mots de code (rendement important) et une grande distance minimale (protection contre les erreurs). Dans Exercices 9 et 15 nous allons découvrir qu'il y a un conflit entre ceux deux objectifs. Etant donné le rendement (et la longueur du code), la distance minimale n'est peut être augmentée que jusqu'au certain point. Notre but est donc d'opérer le plus proche possible de l'optimum.

4.5 Codes linéaires et quelques exemples simples

Jusqu'ici, un code $\mathcal{C}$ était une simple collection des n-uplets sur un corps fini F, étant à la distance au moins $d_{\min}$ l'un de l'autre. Pour trouver les procédures efficaces d'effectuer les opérations d'encodage et de décodage, il est utile de demander aux mots de code d'avoir une certaine structure.

Définition 5 (Code Linéaire)**.** *Nous disons qu'un code* $\mathcal{C}$ *sur le corps* F *est* linéaire*, si pour tous* $\boldsymbol{x}, \boldsymbol{x'} \in \mathcal{C}$ *et tous* $a \in F$ *nous avons*

$$a\boldsymbol{x} - \boldsymbol{x'} \in \mathcal{C}.$$

Ceci est important: comme nous avons discuté dans Section 4.1.5, F^n est un espace vectoriel sur le corps F. La condition ci-dessus est dons équivalente à dire qu'un code linéaire est un sous-espace de F^n. Par conséquence, si la dimension du code $\mathcal{C}$ de longueur n est égale à k et la distance minimale est $d_{\min}$, nous disons que les paramètres du code sont $[n, k, d_{\min}]$.

Exemple 15 (Nous premier exemple est un code linéaire)**.** *Nous pouvons vérifier que la condition précédente tient pour notre premier exemple. Alors ce code est linéaire.*

La première simplification importante qui vient si l'on considére un code linéaire, est que sa distance minimale peut être calculée d'une façon plus efficace.

Lemme 4. *Soit $\mathcal{C}$ un code linéaire. Alors*

$$d_{min} = \min_{\boldsymbol{x} \in \mathcal{C}; \boldsymbol{x} \neq 0} w(\boldsymbol{x}).$$

Proof. De Définition **??**, nous avons

$$d_{\min} = \min_{\boldsymbol{x}, \boldsymbol{x'} \in \mathcal{C}, \boldsymbol{x} \neq \boldsymbol{x'}} d(\boldsymbol{x}, \boldsymbol{x'}) = \min_{\boldsymbol{x}, \boldsymbol{x'} \in \mathcal{C}, \boldsymbol{x} \neq \boldsymbol{x'}} w(\boldsymbol{x} - \boldsymbol{x'}) = \min_{\boldsymbol{x} \in \mathcal{C}, \boldsymbol{x} \neq 0} w(\boldsymbol{x}),$$

où à la dernière étape nous avons utilisé le fait que, grâce à la linéarité du code, $\boldsymbol{x} - \boldsymbol{x'}$ est aussi un mot de code (non-nul). □

Donnons quelques simples exemples des codes linéaires.

Exemple 16. ***Code de parité*** $(n, n-1)$**.** *Les mots du code sont obtenus en ajoutant aux messages $\boldsymbol{u_i}$ un seul bit de contrôle r_1, qui est pris de telle manière que le nombre des 1 dans le mot de code $\boldsymbol{y_i}$ soit pair. Par exemple, $\boldsymbol{u} = (1, 0, 1, 1, 0, 0, 1)$ devient $\boldsymbol{y} = (1, 0, 1, 1, 0, 0, 1, 0)$. La distance de ce code $d_{\min}$ vaut donc 2, et d'après notre discussion précédente, ce code peut détecter (mais pas corriger) une seule erreur. (En fait, on remarque que ce code détecte toute configuration d'un nombre impair d'erreurs et décode erronément toute configuration d'un nombre pair d'erreurs.)*

Exemple 17. ***Code à répétition*** $(n, 1)$**.** *Ici, par contre, il n'y a qu'un seul bit d'information et $n-1$ bits de contrôle qui sont obtenus par répétition du bit d'information. La distance de ce code vaut donc $d_{\min} = n$, ce qui permet de détecter toute configuration de $n-1$ erreurs ou de corriger toute configuration de moins de $(n-1)/2$ erreurs. Evidement, si la capacité correctrice/détectrice de ce code est très élevée, son rendement est très faible. A titre d'exemple, on a représenté les codes à répétition $(2, 1)$ et $(3, 1)$ dans la figure 4.1, qui illustre graphiquement les capacités correctrices et détectrices de ces codes.*

4.5.1 Matrice génératrice d'un code linéaire

Comme un code linéaire est un sous-espace, alors il a une base: un mot de code $\boldsymbol{x} \in \mathcal{C}$ peut être écrit d'une manière unique comme une combinaison linéaire des vecteurs de base. Il est convenable d'écrire cette relation en termes matriciels. Un code linéaire $[n, k]$ possède comme

mots du code les 2^k n-uplets engendrés par une *matrice génératrice* $\boldsymbol{G}$ de k lignes et n colonnes. Les k vecteurs lignes de la matrice $\boldsymbol{G}$ sont pris linéairement indépendants. En particulier, les lignes de $\boldsymbol{G}$ sont k vecteurs du code, et le vecteur nul appartient au code.

Soit $\boldsymbol{u}$ un k-uplet à coder. Le mot du code $\boldsymbol{y}$ correspondant est

$$\boldsymbol{y} = \boldsymbol{u}G. \tag{4.2}$$

Exemple 18. *La matrice génératrice de notre premier exemple est:*

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}. \tag{4.3}$$

Le sous-espace engendré par les lignes de $\boldsymbol{G}$ reste inchangé si l'on permute ces lignes ou lorsque l'on rajoute une ligne à une autre. Alors, nous pouvons appliquer les opérations élémentaires sur les lignes de $\boldsymbol{G}$ afin de mettre $\boldsymbol{G}$ sous forme canonique systématique. Mais nous pouvons faire encore plus. Supposons que nous permettons les permutations des colonnes également. Strictement dit, cela change le sous-espace. Mais les caractéristiques principales du code (rendement et distance minimale) restent les mêmes. En utilisant les permutations des colonnes si besoin, nous pouvons écrire $\boldsymbol{G}$ sous forme systématique:

$$G = [I_k \quad P], \tag{4.4}$$

avec I_k étant la matrice identité d'ordre k (une matrice dont les éléments de la diagonale sont égaux à 1 et tous les autres à 0) et P étant une matrice de taille $k \times (n-k)$. Par exemple, la matrice G donnée en (4.3) peut être mise sous la forme suivante

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{4.5}$$

à l'aide des opérations élémentaires sur ses lignes. Les mots du code sont inchangés, mais leur correspondance avec les k-uplets $\boldsymbol{u}_i$ est modifiée. Le code linéaire est alors dit *systématique* et consiste simplement à laisser inchangés les k bits porteurs d'information et à les reporter tels quels au début des mots du code en ajoutant les $n-k$ bits de contrôle r_i:

$$\boldsymbol{x} = (x_1, \ldots, x_n) = (u_1, u_2, \ldots, u_k, r_1, r_2, \ldots, r_{n-k}).$$

Les bits de contrôle sont donc calculés à partir des bits d'information comme suit:

$$\begin{aligned} r_1 &= u_1 \oplus u_3 \\ r_2 &= u_1 \oplus u_2 \oplus u_3 \\ r_3 &= u_1 \oplus u_2 \\ r_4 &= u_1 \oplus u_2. \end{aligned}$$

Un code linéaire systématique présente l'avantage d'être facile à réaliser à l'aide de circuits logiques et de registres à décalages, car il ne faut stocker que la matrice $k \times (n-k)$ P.

4.5.2 Matrice de contrôle

La *matrice de contrôle* H d'un code (n, k) généré par G est une matrice de taille $(n-k) \times n$ dont $(n-k)$ lignes sont indépendantes, et elle a la propriété que pour chaque vecteur $\boldsymbol{x}$ appartenant au code, nous avons

$$\boldsymbol{x}H^T = \mathbf{0}. \tag{4.6}$$

Il est facile à vérifier qu'une telle matrice existe: si $\boldsymbol{G}$ est donnée sous forme systématique, alors H se construit comme

$$H = [-P^T \quad I_{n-k}]. \tag{4.7}$$

Notez que nous avons

$$\boldsymbol{G}H^T = [I_k \quad P] \begin{bmatrix} -P \\ I_{n-k} \end{bmatrix} = -P + P = 0.$$

Quand le code est binaire, les signes ne jouent pas d'importance, et la matrice de contrôle est de forme $\boldsymbol{H} = [P^T \quad I_{n-k}]$.

Exemple 19. *La matrice H correspondante à la matrice $\boldsymbol{G}$ donnée par (4.5) est*

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.8}$$

Les relations données par (4.6) sont *les relations de parité* (de contrôle):

$$\begin{aligned} x_1 + x_3 + x_4 &= 0 \\ x_1 + x_2 + x_3 + x_5 &= 0 \\ x_1 + x_2 + x_6 &= 0 \\ x_1 + x_2 + x_7 &= 0. \end{aligned}$$

4.5.3 Syndrome

Rappelons que $\boldsymbol{x}$ est le mot du code émis et $\boldsymbol{y} = \boldsymbol{x} \oplus \boldsymbol{e}$ est le mot reçu. Pour la détection et/ou la correction d'erreurs, on calcule le vecteur $\boldsymbol{s}$ de $n-k$ bits, appelé *syndrome*, défini comme

$$\boldsymbol{s} = \boldsymbol{y}H^T. \tag{4.9}$$

D'où

$$\boldsymbol{s} = \boldsymbol{x}H^T \oplus \boldsymbol{e}H^T = \boldsymbol{e}H^T \tag{4.10}$$

est nul s'il n'y a pas d'erreurs. Par contre, toutes les erreurs représentées par un vecteur non-orthogonal aux lignes de H sont détectées.

Supposons qu'on utilise le code systématique donné dans Exemple 19, et que $\hat{\boldsymbol{y}} = (1, 0, 1, 0, 1, 1, 1)$ est reçu. Le syndrome $\boldsymbol{s} = (0, 1, 0, 0)$ indique que le mot reçu est erroné.

La *détection* d'erreurs peut donc être réalisée très simplement par le décodage par syndrome.

La *correction* d'erreurs est plus complexe. Si $H = (\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots, \boldsymbol{h_n})$, où $\boldsymbol{h_i}$ est le i-ème vecteur-colonne de H, (4.10) peut être réécrit sous forme

$$\boldsymbol{s} = e_1\boldsymbol{h_1} + e_2\boldsymbol{h_2} + \ldots + e_n\boldsymbol{h_1} = \sum_{i=1}^{n} e_i\boldsymbol{h_i} \tag{4.11}$$

et le syndrome est une combinaison linéaire des vecteurs-colonnes de la matrice de contrôle. Cette relation peut être utilisée pour corriger le mot reçu, les coefficients e_i qui vérifient (4.11) peuvent être retouvés d'une manière unique.

Notons ainsi que, si les n colonnes de H sont toutes distinctes et non nulles, un code linéaire binaire peut *corriger toutes les erreurs simples*. En effet, une erreur à la k-ième position produit un syndrome qui est tout simplement la colonne $\boldsymbol{h_k}$, et d'après (4.11), cet erreur peut être localisée et donc corrigée.

Exemple 20. *Dans l'exemple précédent, on constate que* $\boldsymbol{s} = (0,1,0,0) = \boldsymbol{h_5}$. *Par conséquent,* $e_5 = 1$ *et* $\boldsymbol{e} = (0,0,0,0,1,0,0)$, *d'où on retrouve que le mot émis est* $\boldsymbol{x} = \boldsymbol{y} \oplus \boldsymbol{e} = (1,0,1,0,0,1,1)$.

4.5.4 Code de Hamming

Pour maximiser le rendement $\eta = k/n$ d'un code, on peut se demander quelle est la longueur maximale n des mots de code, capables de corriger une seule erreur, si le nombre $n-k$ de bits de contrôle est fixé. Effectivement, $n - k = n(1 - r)$. Donc, si $n - k$ est fixé et n croit, alors r croit également.

Comme les vecteurs-colonnes de la matrice H sont les $(n-k)$-uplets, on peut en former au maximum $2^{n-k} - 1$, pour quelles soient toutes non nulles et distinctes. On a vu dans la section précédente qu'un code peut corriger une seule erreur si ses n colonnes sont non nulles et distinctes. On demande donc que

$$n \leq 2^{n-k} - 1,$$

la longueur maximale des mots est atteinte lorsque deux parties de l'inégalité sont égales. Un tel code est appelé un *code de Hamming*. Le tableau suivant donne quelques exemples des codes Hamming et leurs rendements:

n	3	7	15	31	63	127
k	1	4	11	26	57	120
$n-k$	2	3	4	5	6	7
η	0.333	0.571	0.733	0.839	0.905	0.945

En codant de longs messages, on remarque qu'il est possible d'obtenir un rendement élevé.

Les paramètres du code de Hamming sont $[n, k, d_{\min}] = [2^m - 1, 2^m - m - 1, 3]$, où m représente le nombre de bits de contrôle, $m = n - k$. La distance minimale $d_{\min} = 3$, c'est-à-dire, on peux corriger les erreurs simples. La preuve de ce fait est présentée dans Section 4.5.3.

Exemple 21. *Prenons la matrice génératrice d'un code de Hamming* $[7, 4]$ *suivante:*

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Si $\boldsymbol{x}$ *est un mot de code formé à partir d'un message* $\boldsymbol{u}$*, alors*

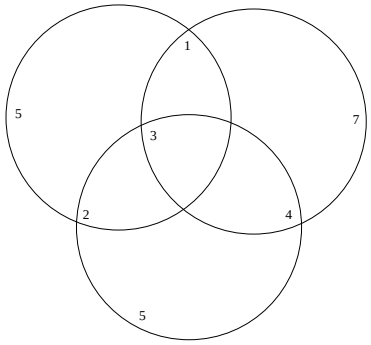
$$\begin{aligned} x_1 &= u_1 \\ x_2 &= u_2 \\ x_3 &= u_3 \\ x_4 &= u_4 \\ x_5 &= u_1 \oplus u_2 \oplus u_3 \\ x_6 &= u_2 \oplus u_3 \oplus u_4 \\ x_7 &= u_1 \oplus u_3 \oplus u_4 \end{aligned}$$

La matrice de contrôle est

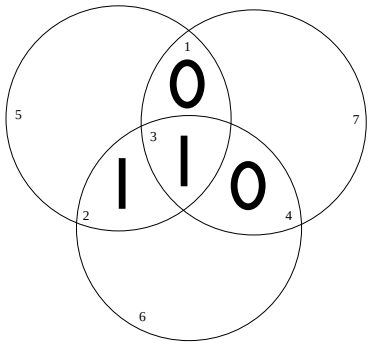
$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Le mot codé de $\boldsymbol{u} = (0, 1, 1, 0)$ *est* $\boldsymbol{x} = (0, 1, 1, 0, 0, 0, 1)$*. Supposons que le mot reçu est* $\boldsymbol{y} = (0, 1, 1, 1, 0, 0, 1)$*. Nous calculons le syndrome* $\boldsymbol{s} = \boldsymbol{y}H^T = (0, 1, 1)$*. En supposant qu'il n'y avait qu'une seule erreur, nous savons de Section 4.5.3 que, si le syndrome est dans la* k*-ième colonne de* H*, alors l'erreur se situe en* k*-ième position. Nous voyons donc que cette erreur se situe en position 4.*

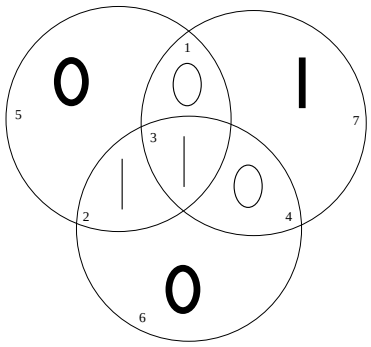
Une autre façon de comprendre consiste à voir le bit de parité x_5 comme le contrôleur de parité u_1, u_2 et u_3, le deuxième bit de parité x_6 comme le contrôleur de parité u_1, u_3 et u_4, et le troisième bit de parité x_7 comme le contrôleur de parité u_3, u_3 et u_4. Nous pouvons le visualiser grâce aux diagrammes de Venn suivants, où chaque cercle contient les éléments d'un groupe de parité. Notez que u_3, étant contrôlé par les trois bits de parité, se retrouve au milieu du diagramme.



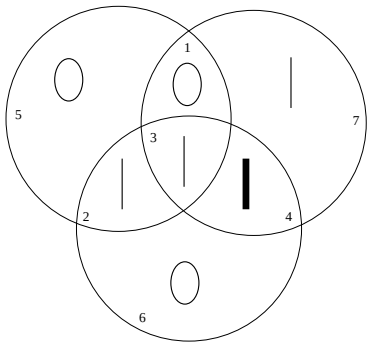
Plaçons maintenant les bits du mot qu'on veut encoder, sur les positions $1, 2, 3$ and 4.



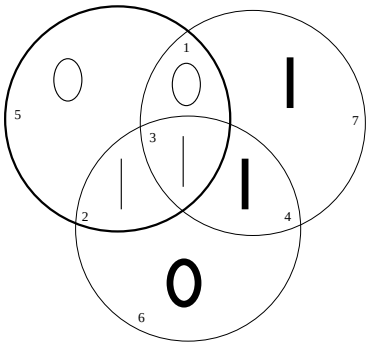
A l'intérieur de chaque cercle, le nombre des "1" doit être paire. Les bits des positions $5, 6$ et 7 sont calculés de façon à satisfaire cette contrainte.



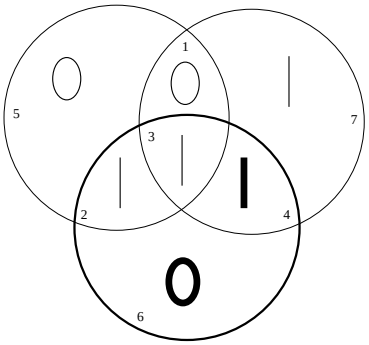
Supposons qu'une erreur se produit lors de la transmission et que le bit de la quatrième position est altéré.



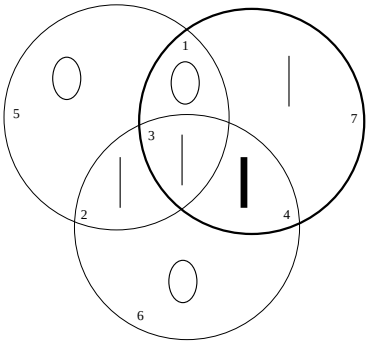
Après la réception, nous plaçons le mot transmis dans le diagramme et analysons chacun des cercles. Nous supposons qu'une seule erreur peut se produire. L'erreur peut être soit à l'intérieur ou à l'extérieur d'un cercle. La relation de parité représentée par le premier cercle est satisfaite (il contient un nombre paire des "1"), donc tous les bits contenus dans ce cercle sont corrects.



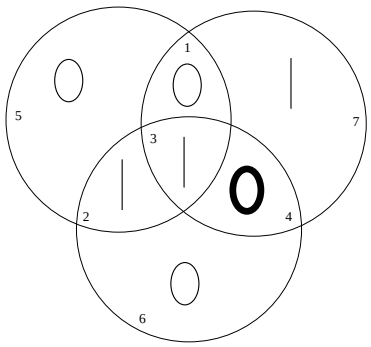
Le cercle suivant contient une erreur car la relation de parité n'est pas satisfaite, ce qui suppose que tous les bits à l'extérieur de ce cercle sont corrects.



La relation de parité représentée par le dernier cercle n'est pas satisfaite non plus. Le cercle contient donc une erreur. Nous pouvons être sûr que les bits à l'extérieur du cercle sont corrects. Ceci nous permet d'identifier le bit erroné comme le bit dans la position 4.



Il ne nous reste donc qu'à corriger la valeur de ce bit pour obtenir le mot de code corrigé.



Ceci nous donne le mot de code correct, $\boldsymbol{x} = (0110001)$, et le mot original $\boldsymbol{u} = 0110$.

Un programme de démonstration de cette méthode se trouve au *http://www.systems.caltech.edu/EE/Faculty/rjm/* sous le lien *Hamming Code Animation*.

4.6 Codes de Reed-Solomon

Le meilleur code qu'on ait vu jusqu'ici a la distance minimale de 3. C'est assez bien pour montrer l'idée principale du codage mais n'a aucune utilité dans les applications réelles. Sur un disque, par exemple, vous pouvez faire le coup radial de $1mm$ et le code utilisé est encore assez puissant pour retrouver l'information. Le fait qu'un bit occupe a peu près 10^{-6} mètres sur le disque veut dire que un tel coup couvre des milliers de bits sur chaque piste. Comment pouvons nous construire des codes avec une grande distance minimale? Ce n'est pas un problème facile, et depuis 60 ans les chercheurs ont fait des efforts considérables pour trouver les codes de mieux en mieux. Le problème est difficile pour les codes binaires, c'est-à-dire, pour les codes sur F_2. Mais si nous considérons les alphabets plus grands, une solution optimale a déjà été proposée aux années 1950 par Irvine Reed et Gus Solomon. Les codes sont appelés les codes de Reed-Solomon (RS) en leur honneur et jusqu'à aujourd'hui sont les codes les plus utilisés. En chaque seconde, il y a des centaines de millions des codes RS en marche, et ils assurent que la plupart de nos communications soit essentiellement sans erreurs.

Même si les codes RS sont puissants, leur définition est remarquablement simple.

Définition 6 (Codes de Reed-Solomon)**.** *Le corps fini F étant donné, choisissons n et k tels que $n \leq |F|$ et $1 \leq k \leq n$. Pour construire un code de Reed-Solomon (RS) avec les paramètres n et k sur le corps F choisissons n éléments* distincts *de F, appelons les $a_0, \cdots, a_{n-1}$. Soit $F[x]$ l'ensemble des polynômes avec les coefficients dans F et la variable x, équipés avec les additions et multiplications standards des polynômes. Alors $\mathcal{C}$ est défini comme suit*

$$\mathcal{C} = \{(A(a_0), \cdots, A(a_{n-1})) : A(x) \in F[x] \text{ s.t. } \deg(A(x)) < k\}.$$

En d'autres termes, considérons l'ensemble des polynômes avec les coefficients dans F et le degré $k-1$ au plus. Nous évaluons chacun des polynômes en n points distincts a_i, $i \in [n]$, et le résultat de ces évaluations donne n composantes d'un mot de code.

Ceci peux être mieux expliqué par un exemple.

Exemple 22 (Codes de Reed-Solomon sur F_5)**.** *Construisons un code RS sur F_5. Nous pouvons choisir n'importe quelle longueur n entre 1 et $5 = |F_5|$. Choisissons le maximum possible, notamment $n = 5$. Maintenant, nous pouvons choisir la dimension du code. Choisissons $k = 2$. Cela nous donne un code de rendement $r = k/n = 2/5$. Commençons par écrire l'ensemble de tous les polynômes $A(x)$ de degré au plus $k-1 = 1$ avec les coefficients dans F_5. Cet ensemble est $\{0, 1, 2, 3, 4, x, 1+x, 2+x, 3+x, 4+x, 2x, 1+2x, 2+2x, 3+2x, 4+2x, 3x, 1+3x, 2+3x, 3+3x, 4+3x, 4x, 1+4x, 2+4x, 3+4x, 4+4x\}$. Comme nous pouvons choisir chacun de deux coefficients dans F_5 et le choix des coefficients est indépendant, nous obtenons $5^2 = 25$ polynômes en total. Par la définition, nous avons besoin de 5 éléments du corps a_i distincts. Comme nous avons en tout exactement 5 éléments distincts, soit $a_i = i$, $i = 0, \cdots, 4$. Maintenant nous attaquons chacun de 25 polynômes. Nous évaluons chacun d'eux pour 5 éléments du corps a_i. Par exemple, prenons le polynôme constant $A(x) = 0$. Si nous l'évaluons pour 5 éléments du corps, nous obtenons le vecteur zéro 00000. C'est le premier mot de code. De la même manière, si nous prenons un autre polynôme constant, nous obtenons les vecteurs constants. Un autre exemple, prenons le polynôme $A(x) = 3 + 2x$. En l'évaluant sur les 5 éléments distincts nous obtenons 30241. Comme ça nous obtenons 25 mots de code en total.*

Lemme 5 (Code de Reed-Solomon)**.** *Un code RS comme défini ci-dessus avec les paramètres n et k sur le corps F a dimension k et la distance minimale $d_{min} = n - k + 1$. C'est la distance minimale la plus grande qu'un code avec les paramètres n et k peut avoir.*

Proof. Pour comprendre la formulation, notez d'abord que le code est linéaire si son image d'évaluation est linéaire. Pour plus de détails, si $A(x)$ et $B(x)$ sont deux éléments de $F[x]$ de degré au plus $k-1$ et si $\alpha, \beta \in F$, alors $C(x) = \alpha A(x) + \beta A(x)$ est aussi un élément de $F[x]$ de degré au plus $k-1$. Dans la suite, pour chaque $a_i \in F$, $\alpha A(a_i) + \beta B(a_i) = C(a_i)$. Cela montre qu'une combinaison linéaire des mots de code est encore un mot de code.

Avant de continuer, nous avons besoin de rappeler que si vous avez un polynôme de degré d avec les coefficients complexes, alors il a exactement d racines complexes, ni plus ni moins. Ceci est le théorème fondamental de l'algèbre. Pous les corps en général, il peut se passer qu'il y en a moins. Par exemple, le polynôme $1 + x^2$, considéré comme un polynôme sur les réels, n'a pas des racines réelles (toutes les racines sont complexes). Mais pour *chaque* corps (les corps finis inclus) le nombre de racines (dans F) d'un polynôme de degré d sur un corps F est au plus d. Ceci n'est pas un fait complètement trivial. Malheureusement, nous n'avons pas le temps de le prouver, alors nous devons juste l'accepter. Maintenant nous pouvons continuer.

Pour voir que le code a la dimension k, notez d'abord qu'il y a exactement $|F|^k$ éléments distincts $A(x)$ de $F[x]$ de degré au plus $k-1$. Il reste à vérifier que l'évaluation de deux polynômes différents donne les mots de code différents. Soit $A(x)$ et $B(x)$ deux polynômes différents de degré au plus $k-1$ et soit $C(x) = A(x) - B(x)$, qui est aussi un polynôme dans $F[x]$ de degré au plus $k-1$. Si

$$(A(a_0), \cdots, A(a_{n-1})) = (B(a_0), \cdots, B(a_{n-1})),$$

alors

$$(C(a_0), \cdots, C(a_{n-1})) = 0.$$

Mais par le théorème fondamental de l'algèbre, le polynôme peut avoir au plus $k-1 < n$ zéros, son degré étant au plus $k-1$. Il suit donc que chaque mot de code non-nul a le poids au moins $n-k+1$. Alors, $d_{\min}(\mathcal{C}) \geq n-k+1$. Mais comme présenté dans Problème 15, la distance minimale de *n'importe quel code* de longueur n et de dimension k est borné par $n-k+1$. Cette borne est appelée la *borne de Singleton*. Nous donc concluons que $d_{\min}(\mathcal{C}) = n-k+1$. □

De point de vue d'implémentation, il vaut mieux de donner une représentation plus traditionnelle du code en termes de matrice génératrice.

Exemple 23. *Considérons un code RS sur F_5 avec $n = 5$ et $k = 2$ d'Exemple 22. Nous pouvons récrire l'image d'évaluation qui prend un polynôme $A(x)$ et l'évalue en cinq points a_i pour obtenir un mot de code $\boldsymbol{x}$: nous avons $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}$, où $\boldsymbol{u}$ est le vecteur de longueur k contenant l'information et $\boldsymbol{G}$ est la matrice génératrice. Soit $\boldsymbol{u} = (A_0, A_1)$, c'est-à-dire les composantes du mot d'information sont les coefficients du polynôme $A(x)$. Dans la suite, définissons la matrice $\boldsymbol{G}$ comme*

$$\boldsymbol{G} = \begin{pmatrix} a_0^0 & a_1^0 & a_2^0 & a_3^0 & a_4^0 \\ a_0^1 & a_1^1 & a_2^1 & a_3^1 & a_4^1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix}.$$

Maintenant l'image d'évaluation est sous forme standard $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}$. Par exemple, si nous commençons par $A(x) = 3 + 2x$, alors $\boldsymbol{u} = (3, 2)$. Le mot de code correspondant est donc $(3, 2)\boldsymbol{G} = 30241$, c'est le même résultat comparant à l'exemple précédent.

Par Lemme 5, un code RS avec les paramètres n et k a la distance minimale $d_{\min} = n-k+1$. Pour notre exemple, nous obtenons $d_{\min} = 5-2+1 = 4$. Concluons qu'un tel code peut corriger un pattern d'effacements avec $d_{\min} - 1 = 3$ effacements au plus.

Exemple 24 (Correction d'effacements pour les codes RS)**.** *Supposons que nous utilisons le code RS construit dans Exemple 23 et que nous reçevons le mot $\boldsymbol{y} = (??2?1)$. Il y a trois effacements. Comme $3 \leq d_{min} - 1 = n - k = 3$, nous savons que nous pouvons reconstruire le mot transmis de façon unique. Soit $\boldsymbol{x}$ le mot transmis. Nous savons que $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}$. Ensuite, nous savons que $\boldsymbol{x}$ et $\boldsymbol{y}$ coincident dans les positions 3 et 5. Nous avons donc un système d'équations*

$$(y_3 y_5) = (21) = (u_1 u_2) \begin{pmatrix} 1 & 1 \\ 2 & 4 \end{pmatrix}.$$

Vous pouvez résoudre ce système d'équations (sur F_5) de la même manière comme c'est montré dans Exemple3. Nous obtenons que

$$A^{-1} = \begin{pmatrix} 2 & 2 \\ 4 & 3 \end{pmatrix},$$

ce qui nous donne $(u_1 u_2) = (32)$. Vous pouvez ensuite retrouver l'information transmise. Si vous voulez, vous pouvez aussi déterminer le mot de code transmis correspondant. Nous obtenons $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G} = 30241$.

Dans les exemples plus réalistes, l'équation matricielle est typiquement assez grande et nous devons résoudre le système d'équations de grande taille, disons 100×100. Dans ce cas, nous ne pouvons plus utiliser la règle de Cramer, mais nous nous tournons vers le pivot de Gauss. Sur les corps finis, le pivot de Gauss est beaucoup plus facile que sur les nombres réels ou complexes. Il n'y a pas des cas des matrices mal définies ou de la précision de la computation. Mettez la matrice dans la forme triangulaire supérieure par les opérations basiques sur les lignes correspondantes (en permutant les colonnes si nécessaire). Ensuite utilisez la substitution arrière pour résoudre le système.

Quand nous voulons construire un code RS, nous avons typiquement un grand degré de liberté. Dès que le corps a été fixé et la longueur n a été choisie, n'importe quel choix de n éléments distincts $a_0, \cdots, a_{n-1}$ va nous convenir. Afin de permettre les appareils divers d'être capable de coopérer entre eux, nous avons besoin de fixer ce choix une fois pour toutes. Le choix suivant est souvent utilisé car il permet d'avoir une implémentation très efficace. Ceci est aussi le choix que vous devez utiliser pour votre projet.

Exemple 25. *Considérons le corps F_7. Comme montré dans Exercice 14, prenons l'élément de corps 3 et considérons les puissances $\{3^0, 3^1, 3^2, 3^3, 3^4, 3^5\} = \{1, 3, 2, 6, 4, 5\}$. Comme vous pouvez voir, l'ensemble de puissances de 3 couvre tous les éléments de corps non-nuls. Autrement dit, le corps F_7 contient un élément tel que $\{a^0, a^1, \cdots, a^{|F|-2}\}$ couvre tous les éléments de corps non-nuls. Ceci n'est pas une propriété spéciale de F_7. En fait, chaque corps fini a au moins un élément pareil. Nous appelons cet élément un* générateur *du corps.*

Etant donnés le corps fini F, la longueur n, où $n \leq |F| - 1$, la dimension k, $1 \leq k < n$, et un générateur du corps a, le code RS canonique *est défini comme suit: prenez n éléments du corps distincts $a_i = a^i$, $i = 0, \cdots, n-1$.*

Considérons encore une fois un code RS sur F_5 (voir Exemple 23). Nous pouvons vérifier que $a = 2$ est un générateur du corps car $\{2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 3\}$ sont tous distincts. Notons que la longueur la plus grande qu'on peut choisir est $n = 4$ car a génére les éléments de corps non-nuls seulement. Si nous prenons $k = 3$, La matrice génératrice est de forme

$$\boldsymbol{G} = \begin{pmatrix} a_0^0 & a_1^0 & a_2^0 & a_3^0 \\ a_0^1 & a_1^1 & a_2^1 & a_3^1 \\ a_0^2 & a_1^2 & a_2^2 & a_3^2 \end{pmatrix} = \begin{pmatrix} a^{0\cdot 0} & a^{1\cdot 0} & a^{2\cdot 0} & a^{3\cdot 0} \\ a^{0\cdot 1} & a^{1\cdot 1} & a^{2\cdot 1} & a^{3\cdot 1} \\ a^{0\cdot 2} & a^{1\cdot 2} & a^{2\cdot 2} & a^{3\cdot 2} \end{pmatrix}$$
$$= \begin{pmatrix} a^0 & a^0 & a^0 & a^0 \\ a^0 & a^1 & a^2 & a^3 \\ a^0 & a^2 & a^4 & a^6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \end{pmatrix}.$$

Exercice 14 donne une idée pourquoi il vaux mieux d'utiliser un code RS canonique. Dans ce cas l'évaluation du polynôme $A(x)$ sur les positions $x = a^i$ correspond à calculer la transformée de Fourier et une telle transformée de Fourier peut être calculée d'une manière très efficace.

4.7 Corps F_{256}

Maintenant, comme nous savons comment construire les codes RS pour un corps fini F donné, il nous reste seulement une chose à faire. Nous voulons construire le corps F_{256}. C'est un corps qui

est utilisé dans la plupart des systèmes en ce moment. Pourquoi F_{256}? Comme nous pouvons le voir du nom, F_{256} a 256 éléments de corps. Ceci est convenable lorsque $256 = 2^8$. Cela signifie qu'on peut étiqueter 256 éléments de corps par les valeurs de 8 bits. Mais 8 bits est exactement un octet, ce qui est la quantité d'information fondamentale des systèmes numériques.

Nous allons faire notre présentation de F_{256} au pure minimum. Il y a beaucoup de belles propriétés des corps finis que vous allez étudier plus tard. Mais pour le moment nous nous contentons de savoir comment le construire et comment faire les computations dans ce corps.

Tout d'abord, notons que nous ne pouvons pas définir F_{256} comme l'ensemble des entiers modulo 256, car 256 n'est pas un nombre premier, et donc, en général, il n'y a pas de l'inverse multiplicatif pour un élément donné. A la place nous allons utiliser la construction suivante.

Définition 7 (Construction de F_{256})**.** *Définissons $f(x) = x^8 + x^4 + x^3 + x^2 + 1$, où $f(x)$ est un polynôme avec variable inconnue x et les coefficients dans F_2.*

Le corps F_{256} contient 256 *polynômes binaires distincts de degré au plus* 7*, c'est-à-dire tous les polynômes $\{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1, \cdots, x^7+x^6+x^5+x^4+x^3+x^2+x+1\}$. Il reste à définir deux opérations de base, notamment addition et multiplication.*

L'addition est l'addition des polynômes usuelle, où les composantes binaires sont additionnées dans le corps F_2.

Pour multiplier deux polynômes $a(x), b(x) \in F_{256}$, faisons d'abord la multiplication des polynômes standards. Ceci donne, disons, le polynôme $\tilde{c}(x)$ qui peut avoir le degré maximal 14*. Réduisons ce polynôme $\tilde{c}(x)$ modulo $f(x)$. C'est-à-dire, écrivons $\tilde{c}(x)$ comme $\tilde{c}(x) = f(x)\alpha(x) + c(x)$, où $c(x)$ est un polynôme de degré* 7 *au maximum. Notons qu'il y a une façon unique d'écrire $\tilde{c}(x)$.* Définissons *le résultat de multiplication de $a(x)$ et de $b(x)$ comme $c(x)$.*

La construction ci-dessus peut être expliquée par l'exemple.

Exemple 26. *Soit $a(x) = x^4 + x^2 + 1$, $b(x) = x^7 + x^4$. Alors,*

$$a(x) + b(x) = x^7 + x^2 + 1.$$

Notons que chaque fois que nous additionnons deux polynômes de degré 7 *au maximum, nous obtenons un polynôme de degré au plus* 7*, qui est aussi un élément de corps.*

La multiplication est un peu plus complexe. Pour calculer $a(x) \cdot b(x)$, ou les deux polynômes sont considérés comme les éléments du corps F_{256}, nous faisons d'abord la multiplication des polynômes ordinaire, notons-la par $a(x) * b(x)$. Nous obtenons

$$a(x) * b(x) = (x^4 + x^2 + 1) * (x^7 + x^4) = x^{11} + x^9 + x^8 + x^7 + x^6 + x^4.$$

La partie droite n'est pas un élément du F_{256} parce que son degré est 11*. Nous avons besoin de le "réduire". Cela veut dire que nous voulons en soustraire un multiple convenant de $f(x)$ pour que le reste soit un polynôme de degré au plus* 7*. Ceci est donné par ce qu'on appelle la* division longue*. Nous avons*

$$x^{11} + x^9 + x^8 + x^7 + x^6 + x^4 = (x^8 + x^4 + x^3 + x^2 + 1)(x^3 + x + 1) + (x^4 + x^3 + x^2 + x + 1).$$

Alors, nous disons que $x^{11} + x^9 + x^8 + x^7 + x^6 + x^4$ modulo $f(x) = x^8 + x^4 + x^3 + x^2 + 1$ est égal à $x^4 + x^3 + x^2 + x + 1$.

Expliquons la division longue plus en détails. Pour les entiers, nous faisons la division avec le reste: pour chaque deux entiers a et b nous pouvons trouver q et r uniques tels que $a = qb+r$ avec $r < b$. D'une manière similaire, pour deux polynômes $a(x)$ et $b(x)$ donnés avec les coefficients dans un corps, nous pouvons trouver les polynômes $q(x)$ et $r(x)$ avec $degr(x) < degb(x)$ tels que $a(x) = q(x)b(x) + r(x)$. Nous ne prouvons pas ce fait, mais remarquons seulement que la condition importante est d'avoir les coefficients des polynômes d'appartenir à un corps. Elle est satisfaite pour la construction de F_{256} où tous les coefficients des polynômes appartiennent au corps F_2. Illustrons le processus de division longue par l'exemple:

$x^{11} + x^9 + x^8 + x^7 + x^6 + x^4$	$x^8 + x^4 + x^3 + x^2 + 1$
$x^{11} + x^7 + x^6 + x^5 + x^3$	x^3
$x^9 + x^8 - x^5 + x^4 - x^3$	
$x^9 + x^8 + x^5 + x^4 + x^3$	$+ x$
$x^9 + x^5 + x^4 + x^3 + x$	
$x^8 - x$	
$x^8 + x$	$+ 1$
$x^8 + x^4 + x^3 + x^2 + 1$	
$-x^4 - x^3 - x^2 + x - 1$	
$x^4 + x^3 + x^2 + x + 1$	

Nous pouvons voir que l'opération de division est similaire à celle des entiers. L'analogie peut aller même plus loin, et nous pouvons définir le plus grand diviseur commun pour deux polynômes $\gcd(a(x), b(x))$ comme le polynôme $c(x)$ (monôme) qui divise tous les deux et tel que n'importe quel autre polynôme $d(x)$ qui les divise, divise aussi $c(x)$. gcd peut être trouvé par l'algorithme euclidien:

$$\begin{aligned} a(x) &= q(x)b(x) + r(x) && \deg(r(x)) < \deg(b(x)) \\ b(x) &= q_1(x)r(x) + r_1(x) && \deg(r_1(x)) < \deg(r(x)) \\ r(x) &= q_2(x)r_1(x) + r_2(x) && \deg(r_2(x)) < \deg(r_1(x)) \\ r_2(x) &= q_3(x)r_2(x) \end{aligned}$$

Dès que le degré du reste diminue, le processus s'arrête (ici nous supposons qu'il est terminé à la quatrième étape). Tout ceci s'explique mieux par l'exemple:

$$\begin{aligned} (x^{11} + x^9 + x^8 + x^7 + x^6 + x^4) &= (x^3 + x + 1)(x^8 + x^4 + x^3 + x^2 + 1) + (x^4 + x^3 + x^2 + x + 1) \\ (x^8 + x^4 + x^3 + x^2 + 1) &= (x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1) + (x^3 + x) \\ (x^4 + x^3 + x^2 + x + 1) &= (x + 1)(x^3 + x) + 1 \\ (x^3 + x) &= (x^3 + x).1 + 0 \end{aligned}$$

Cela montre que

$$\gcd(x^{11} + x^9 + x^8 + x^7 + x^6 + x^4; x^8 + x^4 + x^3 + x^2 + 1) = 1.$$

Comme pour les entiers, en reversant le processus, il peut être vérifié que gcd peut être écrit sous forme suivante

$$\text{gcd} = \alpha(x)a(x) + \beta(x)b(x).$$

Pour l'exemple ci-dessus, nous pouvons vérifier que

$$\text{gcd} = r_2(x) = (1 + q_2(x)q_1(x))a(x) - (q(x)(1 + q_2(x)q_1(x)) - q_2(x))b(x).$$

La plupart des propriétés qu'un corps doit satisfaire (voir le tableau dans Section 4.1.1) sont très faciles à vérifier. La propriété la plus difficile à vérifier est que chaque élément de corps non-nul a son inverse par rapport à la multiplication. Cela signifie que nous avons besoin de vérifier que, pour chaque $a(x) \in F_{256}$, il existe un polynôme, appelons-le $a^{-1}(x) \in F_{256}$, tel que $a(x) \cdot a^{-1}(x) = 1$. Une approche longue, mais directe, est de multiplier l'élément donné $a(x) \in F(x)$ par tous les éléments de corps $b(x) \in F(x)$ et de vérifier qu'il y en un $b(x)$ tel que $a(x) \cdot b(x) = 1$.

La raison pour laquelle cela doit marcher est que l'opération modulo s'effectue par rapport à un polynôme $f(x)$ irréductible. Les polynômes irréductibles jouent le rôle similaire à celui des nombres premiers. Par définition, nous ne pouvons pas les décomposer en termes de degrés plus petits. En plus, comme pour les entiers, nous avons deux lemmes suivants.

Lemme 6 (Propriété du diviseur)**.** *Si $f(x)$ est irréductible et divise le produit $a(x)b(x)$, alors il doit diviser $a(x)$ ou $b(x)$ ou les deux.*

Proof. Supposons que $f(x)$ divise $a(x)$. Alors c'est fait. Si $f(x)$ ne divise pas $a(x)$, alors $\text{gcd}(f(x), a(x)) = 1$, comme $f(x)$ est irréductible. Donc, par l'algorithme euclidien il existe $\phi(x)$ et $\alpha(x)$ tels que $1 = \phi(x)f(x) + \alpha(x)a(x)$. En multipliant par $b(x)$, nous obtenons

$$b(x) = \phi(x)f(x)b(x) + \alpha(x)a(x)b(x)$$

Maintenant, il suffit de noter que $f(x)$ divise la partie droite de l'équation (par l'hypothèse), alors il doit diviser aussi la partie gauche. □

Lemme 7 (Propriètè de la factorisation unique)**.** *Un polynôme avec les coefficients appartenant à un corps peut être décomposé en un produit des termes irréductibles. Cette décomposition est unique jusqu'à l'ordre des termes et le choix de leurs coefficients globaux.*

Nous n'utilisons pas ce lemme explicitement, alors nous passons la preuve, mais remarquons seulement qu'il est un corollaire direct de la propriété du diviseur.

Maintenant nous sommes prêts à expliquer, pourquoi un élément non-nul a sûrement l'inverse unique. Comme on est seulement intéressé à l'*existence* d'un tel élément, nous procédons comme suit. D'abord, nous déclarons qu'il y a *au plus* un seul inverse. Supposons que, au contraire, pour un élément du corps $a(x)$ non-nul donné, $b(x)$ et $c(x)$ sont inverses tous les deux, $b(x) \neq c(x)$. Par $a(x) \cdot b(x) = a(x) \cdot c(x)$ nous concluons que $a(x) \cdot (b(x) - c(x)) = a(x) \cdot d(x) = 0$, où $d(x)$ est un élément de corps non-nul. Cela signifie que $a(x) * d(x)$ doit être un multiple de $f(x)$, par

exemple, $a(x) * d(x) = f(x) * e(x)$ pour un polynôme binaire $e(x)$. Par la propriété du diviseur, ceci voudrait dire que $f(x)$ divise $a(x)$ ou $d(x)$ ou les deux. Mais ce n'est pas possible, car $f(x)$ est irréductible, alors $\gcd(f, a) = \gcd(f, d) = 1$. Ceci impliquerait que soit $a(x)$ où $b(x)$ (où les deux) doivent avoir un facteur en commun avec $f(x)$. Le fait que $f(x)$ est irréductible dans notre exemple précédent, peut être vérifié en le divisant par tous les polynômes de degré au plus 4 (il y en a 32).

Voici l'argument important de la discussion: comme dans notre exemple précédent chaque multiplication d'un élément de corps non-nul $a(x)$ avec l'ensemble des tous les éléments de corps doit donner un élément distinct, l'un d'eux va être l'élément 1. Cela prouve que l'inverse existe.

Si nous voulons *calculer* l'inverse, nous utilisons exactement la même technique comme pour le calcul de l'inverse d'un élément de F_p. C'est-à-dire, pour un élément de corps non-nul donné, nous utilisons l'algorithme euclidien étendu afin de calculer deux polynômes $\alpha(x)$ et $\phi(x)$ tels que

$$a(x) * \alpha(x) + f(x) * \phi(x) = \gcd(a(x), f(x)) = 1,$$

où $\gcd(a(x), f(x)) = 1$ car $f(x)$ est irréductible. Si maintenant nous considérons cette équation modulo $f(x)$, nous voyons que $\alpha(x)$ est l'inverse cherché.

Finalement, nous expliquons comment établir un tableau pour effectuer les opérations sur un corps comme F_{256} d'une manière efficace. Par simplicité, nous faisons cela pour un corps plus petit F_{16} (notez que $16 = 2^4$). Le corps contient tous les polynômes binaires de degré plus petit ou égal à 3 $\{0, 1, x, x^2, x^3, x+1, x^2+1, ...x^3+x^2+x+1\}$, où la multiplication est définie modulo le polynôme irréductible $f(x) = x^4 + x^3 + 1$. Dans cette représentation, l'addition est très facile à faire car nous additionnons les polynômes comme d'habitude, en ajoutant les coefficients modulo 2. Ceci se fait facilement dans la notation binaire en utilisant la correspondance évidente $\{0000, 0001, 0010, 0100, 0011, 0101, ...1111\}$. Quand même, c'est moins convenable pour la multiplication ou pour la recherche de l'inverse. Pour ces opérations, il vaut mieux de présenter les éléments de corps par $\{0, 1, x, x^2, x^3, x^4, x^5, ..., x^{14}\}$. Comme nous travaillons modulo x^4+x^3+1, nous avons que

$$\begin{aligned}
&x^4 = -x^3 - 1 = x^3 + 1 \\
&x^5 = x.(x^3 + 1) = x^4 + x = x^3 + x + 1 \\
&x^6 = x^4 + x^2 + x = x^3 + x^2 + x + 1 \\
&x^7 = x^4 + x^3 + x^2 + x = 2x^3 + x^2 + x + 1 = x^2 + x + 1 \\
&\text{etc.}
\end{aligned}$$

Il vous est aussi recommandé de vérifier que $x^{15} = 1$. Alors, nous pouvons établir le tableau:

Maintenant nous pouvons facilement faire la multiplication et l'inversion. Par exemple,

$$\begin{aligned}
(1110).(0101) &= x^8.x^9 = x^{17} = x^2 = (0100) \\
(0111)^{-1} &= x^{-7} = x^{-7}.x^{15} = x^8 = (1110)
\end{aligned}$$

Conclusion: il y a une théorie de corps entre les lignes ci-dessus. En effet, il peut être prouvé que tous les corps finis ont p^k éléments, où p est un nombre premier et k est un entier. Ensuite,

multiplication	addition	représentation binaire
0	0	0000
1	1	0001
x	x	0010
x^2	x^2	0100
x^3	x^3	1000
x^4	x^3+1	1001
x^5	x^3+x+1	1011
x^6	x^3+x^2+x+1	1111
x^7	x^2+x+1	0111
x^8	x^3+x^2+x	1110
x^9	x^2+1	0101
x^{10}	x^3+x	1010
x^{11}	x^3+x^2+1	1101
x^{12}	$x+1$	0011
x^{13}	x^2+x	0110
x^{14}	x^3+x^2	1100
x^{15}	1	0001

Table 4.1: Opérations dans F_{16}.

un tel corps peut être construit comme l'ensemble des polynômes aves les coefficients dans F_p modulo un polynôme irréductible (qui n'a pas de facteurs sur F_p) de degré k. L'addition de deux polynômes se fait composante par composante et la multiplication est la multiplication régulière de polynômes suivie par la réduction par rapport au polynôme irréductible.

4.8 Références bibliographiques

S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications.* Prentice Hall, 1983.

R. E. Blahut, *Theory and Practice of Error-Control Codes.* Addison-Wesley, 1983.

W. C. Huffman and V. Pless, *Fundamentals of Error Correcting Codes.* Cambridge University Press, 2003

R. J. McEliece, *The Theory of Information and Coding.* Cambridge University Press

4.9 Exercices

Exercice 1. Soit $F = F_5$. Considérons le système d'équations linéaires

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 2, \\ 3x_1 + 2x_2 + 4x_3 &= 1, \\ 3x_1 + x_2 + x_3 &= 0. \end{aligned}$$

a) Ecrire le système sous forme matricielle comme $\boldsymbol{A}\boldsymbol{x}^T = \boldsymbol{u}^T$. Quels sont $\boldsymbol{A}$, $\boldsymbol{x}$, et $\boldsymbol{u}$, et quelles sont leurs dimensions?

b) Supposons que vous savez que le déterminant de la matrice $\boldsymbol{A}$ est égal à 7, en la considérant comme une matrice avec les éléments sur les entiers. Quel est le déterminant de $\boldsymbol{A}$, en la considérant comme une matrice sur F_5?

c) Montrer que le système peut être résolu sur F_5 de manière unique?

d) Résoudre le système sur F_5 en utilisant le pivot de Gauss.

e) Si vous avez besoin de résoudre un système de taille $n \times n$, quelle est la complexité du pivot de Gauss, c-à-d., combien d'opérations élémentaires (addition, multiplication, etc.) est-il nécessaire de faire?

Astuces: a) Ceci est un système de taille 3×3; b) C'est 2; c) $|\boldsymbol{A}| = 2 \neq 0$; d) La solution est $(1, 0, 2)$; e) n^3;

Exercice 2. Soit $\mathcal{V}$ l'ensemble formé par les vecteurs $\boldsymbol{v} = (v_1, \ldots, v_n)$, où v_i est égal à 0 ou 1, et le corps F est le corps de Galois F_2.

a) Montrer que cet ensemble est un espace vectoriel. Quelle est sa dimension? Répéter le même exercise avec v_i dans F_p, où p est un nombre premier.

b) Montrer que la *distance de Hamming* entre deux vecteurs $\boldsymbol{u}$ et $\boldsymbol{v}$,

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{n} |u_i - v_i|.$$

définit une métrique sur cet espace vectoriel.

Astuce: b) Nous voulons montrer que $d(\boldsymbol{u}, \boldsymbol{v}) \leq d(\boldsymbol{u}, \boldsymbol{w}) + d(\boldsymbol{w}, \boldsymbol{v})$; Il suffit de considérer trois composantes, appelez-les $\boldsymbol{u}_i$, $\boldsymbol{w}_i$, et $\boldsymbol{v}_i$; fixez $\boldsymbol{u}_i$ et considérez les quatre cas possibles où $\boldsymbol{w}_i$ et $\boldsymbol{v}_i$ sont soit les mêmes soit différents de $\boldsymbol{u}_i$;

Exercice 3. Soit $\mathcal{S}$ un sous-espace de l'espace vectoriel $\mathcal{W}$ sur le corps F. L'ensemble

$$\mathcal{S}^{\perp} = \{\boldsymbol{w} \in \mathcal{W} : \boldsymbol{w} \bullet \boldsymbol{s} = 0 \text{ pour tout } \boldsymbol{s} \in \mathcal{S}\}$$

est appelé *l'espace dual*. Ici, $\boldsymbol{w} \bullet \boldsymbol{s}$ est défini par $\sum_i \boldsymbol{w}_i \boldsymbol{s}_i$, toutes les computations s'effectuant dans F.

a) Montrer que $\mathcal{S}^{\perp}$ est un sous-espace.

b) Montrer que si $\mathcal{S}$ est un sous-espace de dimension k, alors $\mathcal{S}^\perp$ est un sous-espace de dimension $n-k$.

c) Prendre $F = F_2$ et $\mathcal{S} = \{0000, 0011, 1100, 1111\}$. Déterminer $\mathcal{S}^\perp$ dual.

Astuce: a) Prenez $\boldsymbol{w}, \boldsymbol{w}' \in \mathcal{W}$. Montrez que $\alpha\boldsymbol{w} - \boldsymbol{w}' \in \mathcal{W}$; b) Nous avons besoin de montrer que si $\boldsymbol{A}$ est une matrice de taille $k \times n$, $k \leq n$, avec les lignes indépendantes, alors l'ensemble des solutions $\boldsymbol{A}\boldsymbol{w}^T = 0$ est de dimension $n-k$; ramenez cette équation sous forme triangulaire supérieure par le pivot de Gauss; montrez que vous pouvez choisir $n-k$ composants les plus bas de $\boldsymbol{w}$ d'une manière arbitraire et que les composantes restantes vont être donc fixées;

Exercice 4. Considérons un code binaire généré par la matrice

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

cela signifie que

$$\mathcal{C} = \{\boldsymbol{x} : \boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}, \boldsymbol{u} \in F_2^3\}.$$

a) Le mot $(1, 0, 1, 0, 1, 0)$, est-il un mot de code ?

b) Combien de mots de code sont générés par ce code? Donner la liste.

c) Les premiers deux bits du mot $(x, x, 0, 1, 1, 0)$ ont été supprimés. Quel était le mot émis? Quel est le nombre maximal des symboles binaires qui peuvent être effacés pour que le mot émis puisse encore être retrouvé?

d) Soit $\mathcal{C}$ un espace vectoriel des mots générés par le code, $\mathcal{C}^\perp$ son complément orthogonal; trouver $\mathcal{C}^\perp$ (donner la liste de tous ces éléments).

e) $\mathcal{C}^\perp$ est aussi un espace vectoriel. Donner la base de cet espace. Une telle base est uniquement déterminée par la matrice $\boldsymbol{H}$. La dernière est appelée la matrice de contrôle (voir Section 4.5.2).

f) Soit $\hat{\boldsymbol{y}}$ le mot reçu. Le vecteur donné par $\boldsymbol{s} = \hat{\boldsymbol{y}}\boldsymbol{H}^T$ est appelé le *syndrome* (voir Section 4.5.3). Si le mot reçu est $(0, 0, 1, 1, 0, 1)$, quel est le syndrome? Quel est le mot le plus probablement émis?

g) Quel est le plus petit nombre d'erreurs (la distance de Hamming) qui pourraient changer un mot de code pour un autre?

Astuces: a) combinez la première et la troisième ligne de $\boldsymbol{G}$; b) 8; c) $(1, 1, 0, 1, 1, 0)$; $d_{\min} = 3$, $dmin - 1 = 2$, alors nous pouvons tolérer au plus deux effacements; d+e) $\boldsymbol{G}$ est sous forme systématique; il est donc facile de trouver $\boldsymbol{H}$; f) $(0, 0, 1, 1, 1, 1)$ g) $d_{\min} = 3$

Exercice 5. Soit n la longueur. a) Quelles sont les matrices génératrices et de contrôle pour un code de parité? b) Et pour le code à répétition?

Astuce: a) $\boldsymbol{G}$ est de dimensions $(n-1) \times n$; b) $\boldsymbol{G}$ est de dimensions $1 \times n$; Nous ne l'avons pas encore étudié, mais ces deux matrices sont les matrices duales;

Exercice 6. Les deux mots du code à répétition $(3, 1)$, $(0, 0, 0)$ et $(1, 1, 1)$, ont la même probabilité.

a) Le code marche comme un code de détection d'erreurs. Si p est la probabilité d'erreur sur un seul bit, quelle est la probabilité qu'un mot incorrect ne va pas être détecté? Evaluer la probabilité pour $p = 1/3$.

b) Maintenant le code marche comme un code de correction d'erreurs. En fonction du mot reçu, quand décidons nous que le mot $(0, 0, 0)$ a été émis? Quelle est la probabilité qu'un mot incorrect va être corrigé aux valeurs incorrectes? Evaluer la probabilité pour $p = 1/3$.

Exercice 7. Les 16 mots de code d'un code $(7, 4)$ sont $(0, 0, 0, 0, 0, 0, 0)$, $(1, 1, 1, 1, 1, 1, 1)$, et toutes les permutations cycliques de $(0, 0, 0, 1, 0, 1, 1)$ et $(0, 0, 1, 1, 1, 0, 1)$.

a) Quelle est la matrice génératrice du code?

b) Quelle est la matrice de contrôle correspondante?

c) Quelle est la distance minimale du code? Son rendement? Est-ce que ce code est équivalent au code de Hamming $(7, 4)$, donné dans Section 4.5.4?

Exercice 8. En pratique, une matrice génératrice systématique d'un code linèaire n'a jamais une colonne toute-à-zéros. Pourquoi?

Astuce: Supposons que vous avez éliminé une colonne, c-à-d., supposons que vous ne transmettez pas la composante correspondante. Qu'est-ce qui change à la réception? Est-ce que le récepteur perd de l'information?

Exercice 9. Prenez un code binaire en blocs ayant m mots de longueur n. C'est-à-dire, chaque mot de code est une séquence de n bits. Supposons que ce code est capable de corriger jusqu'à e erreurs.

a) Pour un mot de code x, appelons S l'ensemble des séquences binaires pour lesquelles le décodeur retourne x. Montrer que le nombre total des séquences dans l'ensemble $|S|$ a la borne inférieure suivante:

$$|S| \geq \sum_{i=0}^{e} \binom{n}{i},$$

où $\binom{n}{i}$ est un coefficient binomial qui donne le nombre des combinaisons possibles de placer i éléments en n places.

b) Montrer que le nombre M des mots de code satisfait:

$$M \leq \frac{2^n}{\sum_{i=0}^{e} \binom{n}{i}}$$

Cette borne supérieure est connue comme la "borne d'empilement des sphères", parce qu'elle est équivalente à considérer une sphère (de dimension n) avec le rayon e, centrée sur chaque mot de code et à remplir l'espace des séquences possibles par les sphères sans qu'elles se recouvrent.

c) Nous avons vu pendant le cours (Section **??**) que les codes de Hamming peuvent corriger jusqu'à une erreur. Pour la longueur de code $n = 2^m - 1$ ces codes ont $2^{2^m - m - 1}$ mots de code. Montrer que les codes de Hamming vérifient la borne précédente avec l'équalité. Les codes pareils sont appelés *parfaits*.

Exercice 10. Montrer que, dans un code binaire linéaire, soit tous les mots de code contiennent un nombre pair des 1's soit la moitié des mots de code a un nombre pair des 1's et l'autre moitié a un nombre impair des 1's.

Astuce: Notez que la somme des deux mots de code de poids impairs donne un mot de poids pair, que la somme des deux mots de poids pairs donne un mot de poids pair et que la somme d'un mot de poids pair et d'un mot de poids impair donne un mot de poids impair. Supposons donc que tous ces mots de code n'ont pas tous le nombre pair des 1's. Choisissez un mot de code de poids impair, appelons-le $\boldsymbol{w}$. Mettez les mots de code en couples. A chaque mot de code $\boldsymbol{x} \in \mathcal{C}$ associez le mot de code $\boldsymbol{x} + \boldsymbol{w}$. Montrez que cela prouve l'hypothèse.

Exercice 11. Construire un code RS de longueur $n = 7$ et de dimension $k = 3$ sur F_7.

a) Quelle est sa matrice génératrice $\boldsymbol{G}$?

b) Supposons que le mot d'information est $\boldsymbol{u} = (123)$. Quel est le mot de code correspondant?

c) Supposons qu'un mot de code a été émis sur le canal à effacements et le mot de code reçu est (64?4?0?). Est-il possible de retrouver le mot transmis? Si oui, quel est ce mot.

Astuce: a) Les lignes de $\boldsymbol{G}$ sont $((1,2,3,4,5,6,0),(1,4,2,2,4,1,0),(1,4,2,2,4,1,0))$; b) $(6,1,6,0,4,4,0)$; c) $(6,4,1,4,6,0,0)$

Exercice 12. Considérons le corps F_{256} généré par le polynôme primitif $f(x) = x^8 + x^4 + x^3 + x^2 + 1$. Comme il était démontré pendant le cours, ceci signifie qe les éléments de corps sont 256 polynômes de degré 7 au maximum et ont les coefficients binaires. L'addition est l'addition des polynômes usuelle, où les composantes binaires sont additionées sur le corps F_2. Pour multiplier deux polynômes $a(x)$ et $b(x)$, faites d'abord la multiplication standard des polynômes. Ceci vous donnera un polynôme, disons $\tilde{c}(x)$, qui peut avoir le degré maximal 14. Réduisez le polynôme $\tilde{c}(x)$ modulo $f(x)$. C'est-à-dire, écrivez $\tilde{c}(x)$ comme $\tilde{c}(x) = f(x)\alpha(x) + c(x)$, où $c(x)$ est un polynôme de degré au plus 7. Notez qu'il y a une manière unique d'écrire $\tilde{c}(x)$. Définissez maintenant le résultat de multiplication de $a(x)$ et $b(x)$ comme $c(x)$.

a) Calculer $(a(x) + b(x))c(x)$, où $a(x) = 1 + x^4 + x^7$, $b(x) = x + x^2 + x^6$, et $c(x) = 1 + x^3$.

Sans le prouver, nous notons que le corps a la propriété suivante: l'ensemble $\{x^0, x^1, \cdots, x^{254}\}$, réduit modulo le polynôme primitif $f(x)$, contient tous les éléments de corps non-nuls, et $x^{255} = x^0 = 1$. En principe, vous le vérifierez par le calcul direct. Il y a donc une correspondance entre les éléments de corps non-nuls et les premiers 255 puissances de x.

b) Utilisant cette propriété, trouver x^{-247} écrit comme un élément de F_{256}?

c) Supposons, vous sauvegardez un tableau dans lequel chaque élément de corps (non-nul) est écrit et comme un polynôme $a(x)$ de degré maximal 7 et comme x^i, où $0 \leq i \leq 254$. Ecrivez les 10 premières lignes du tableau, correspondant aux entrées x^i, $0 \leq i \leq 9$.

d) Supposons, vous voulez calculer la multiplication des deux éléments caractéristiques $a(x)$ et $b(x)$. Comment pouvez-vous utiliser le tableau précédent d'une manière efficace? Démontrer cela par l'exemple simple de $a(x) = x^3$ et $b(x) = x^5 + x^4 + x^3 + x$.

Exercice 13. Construire le code RS sur F_{11} de longueur $n = 8$ et de $k = 4$. Utilisez pour cette construction les éléments de corps non-nuls $x_i = i$, $i = 1, \cdots, 8$.

a) Ecrivez la matrice génératrice $\boldsymbol{G}$ correspondante.

b) Supposons que vous voulez transmettre le vecteur d'information $\boldsymbol{u} = (1407)$. Quel est le mot de code correspondant?

c) Supposons vous avez reçu le mot (62?90???). Vous voulez retrouver le mot transmis. Ecrivez le système d'équations correspondant. Maintenant utilisez le pivot de Gauss afin de recouvrir l'information transmise. Quel est le mot de code transmis?

Exercice 14 Dans la première partie du cours, vous avez rencontré la transformée de Fourrier de N positions sur les nombres complexes. Les transformées de Fourier jouent un rôle fondamental dans le traitement de signal et elles seront un des outils de base pendant vos études. Cet exercice va vous montrer par un exemple concret que le concept des transformées de Fourier est très général.

a) Considérons le corps F_7. Calculer explicitement l'ensemble $\{3^i \mod 7\}$, $i = 0, \cdots, 5$. Combien vaut $3^6 \mod 7$? Un élément de corps a, tel que ses puissances a^i, $i = 0, \cdots, |F| - 2$, couvrent tous les éléments de corps non-nuls, est appelé un *générateur*. Nous avons déjà vu dans Exercice 12 que x est un générateur du corps F_{256} produit par le polynôme primitif $f(x) = x^8 + x^4 + x^3 + x^2 + 1$.

b) Considérons un vecteur $\boldsymbol{u}$ de longueur 6 dont les composantes sont les éléments de F_7. Définissons la "Transformée de Fourier par paires" suivante entre les vecteurs $\boldsymbol{u}$ et $\hat{\boldsymbol{u}}$ de longueur 6 dont les composantes appartiennent à F_7. Nous avons

$$\hat{\boldsymbol{u}}_i = \sum_{j=0,\cdots,5} \boldsymbol{u}_j 3^{ij}, \text{transformée de Fourier}$$
$$\boldsymbol{u}_j = 6 \sum_{i=0,\cdots,5} \hat{\boldsymbol{u}}_i 5^{ij}, \text{transformé de Fourier inverse}$$

où toutes les computations s'effectuent sur F_7. Notons que 5 est l'inverse multiplicatif de 3.

c) Commençons par $\boldsymbol{u} = (123456)$. Calculer $\hat{\boldsymbol{u}}$. Ensuite vérifier que $\boldsymbol{u}$ est bien obtenu en faisant la transformé inverse.

d) Calculer la convolution cyclique de deux vecteurs $\boldsymbol{u} = (123456)$ et $\boldsymbol{v} = (121212)$. Ensuite calculer la transformée de Fourier du résultat. Alternativement, calculer la transformée de Fourier des $\boldsymbol{u}$ et $\boldsymbol{v}$ et après faire la multiplication composante par composante. Des commentaires?

e) (Bonus) Pouvez-vous montrer en général que, si vous commencez par $\boldsymbol{u}$ et calculez d'abord $\hat{\boldsymbol{u}}$ en utilisant la transformée de Fourier et ensuite vous calculez la transformée de Fourier inverse de $\hat{\boldsymbol{u}}$, vous obtenez à nouveau $\boldsymbol{u}$?

Exercice 15 Considérons un code binaire linéaire $\mathcal{C}$ de longueur n et de dimension k. Cela signifie que $\mathcal{C}$ est un sous-espace de $\{0, 1\}$ de dimension k. Soit $d_{\min}$ la distance minimale du code $\mathcal{C}$.

a) Prouver l'inégalité fondamentale de Singleton:

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

Astuce: Ecrivez 2^k mots de code dans une matrice binaire de dimensions $(2^k) \times n$. Supprimez

tout sauf les k premières colonnes de cette matrice. Notez que la distance entre deux mots de code est la somme de la différence entre k premiers composants et la différence parmi $(n-k)$ dernières composantes.

b) Donner un exemple d'un code qui atteint la borne supérieure précédente avec les paramètres n et $k = 1$.

Astuce: code à répétition.