INTRODUCTION

Dénotons par V l'ensemble de tous les mots possibles. Alors tout élément de V est un mot de n symboles tirés de $\{0, 1, \cdots, q-1\}$. Pour $n = 3$ et $p = 3$ on aura

$$\begin{aligned} V &= \{abc \mid a, b, c \in \{0, 1, 2\}\} \\ &= \{000, 001, \cdots, 102, 110, \cdots, 222\} \end{aligned}$$

L'ensemble V est un espace vectoriel de dimension n sur le corps $\mathbb{Z}_p$. On peut voir ceci en écrivant les mots comme vecteurs. On écrira les vecteurs en rangée au lieu de en colonne. C'est pour des raisons pratiques: on aura souvent à écrire des "grands" vecteurs, qui sont plus compactes en rangée qu'en colonne.

$$\begin{aligned} V &= \left\{ \begin{bmatrix} a & b & c \end{bmatrix} \mid a, b, c \in \mathbb{Z}_3 \right\} \\ &= \left\{ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, \cdots, \begin{bmatrix} 1 & 0 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, \cdots, \begin{bmatrix} 2 & 2 & 2 \end{bmatrix} \right\} \end{aligned}$$

On peut construire des espaces vectoriels sur le corps $\mathbb{Z}_p$ comme sur le corps $\mathbb{R}$. Pour $n = 3$ et $p = 3$ on a $V = {\mathbb{Z}_p}^n = {\mathbb{Z}_3}^3$.

Le code $C = \{000, 111, 222\}$ de l'exemple 13.1 est en sous-espace de ${\mathbb{Z}_3}^3$. On peut dire même plus: $C = \operatorname{span}\left\{ \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \right\}$ sur le corps $\mathbb{Z}_2$, car:

$$\begin{aligned} \operatorname{span}\left\{ \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \right\} &= \left\{ t \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \mid t \in \mathbb{Z}_3 \right\} \\ &= \left\{ t \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \mid t \in \{0, 1, 2\} \right\} \\ &= \left\{ 0 \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, 1 \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, 2 \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \right\} \\ &= \left\{ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 2 & 2 \end{bmatrix} \right\} \end{aligned}$$

On voit que un "mot" n'est nul autre qu'un vecteur écrit sans les parenthèse et les espaces: "$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$" est équivalent à "111". Pour cet raison on écrira parfois des vecteurs comme des mots, et parlera de faire des opérations algébriques sur des "mots".

Le fait que C est un sous-espace d'un espace vectoriel permet une approche plus efficace pour Alice et Bob.

CODES LINÉAIRES: MATRICE GÉNÉRATRICE

Un code linéaire est un code qui est un sous-espace vectoriel de l'espace vectoriel ${\mathbb{Z}_p}^n$. Comme tout autre sous-espace, un code possède des bases. Si on écrit les vecteurs de la base d'un code comme rangées d'une matrice, on a une MATRICE GÉNÉRATRICE pour ce code. Note que typiquement il y aura plusieurs bases pour un sous-espace; on préfère une base qui donne une matrice génératrice en forme échelonnée réduite. Une telle base, écrite comme rangées d'une matrice, donne une MATRICE GÉNÉRATRICE STANDARD

Un code linéaire est donc l'espace rangée d'une matrice. On peut faire des opérations de rangée sur une matrice sans changer son espace-rangée, donc on peut toujours trouver une matrice génératrice standard en faisant une réduction de Gauss-Jordan.

Une matrice génératrice "génère" le code: le code est exactement l'ensemble des vecteurs obtenus comme combinaison linéaire des rangées de G. Donc le code C généré par la matrice G est exactement $C = \{\mathbf{x}G \mid \mathbf{x} \in \mathbb{R}^m\}$.

Exemple 15.1. Soit le code $C = \{0000, 1100, 0011, 1111\}$.

On peut voir que c'est un code linéaire, car on peut trouver un ensemble engendrant. Il y en a plusieurs, par exemple, $\{1100, 1111\}$ comme montre les calculs suivants.

$$\begin{aligned} C &= \operatorname{span}\{1100, 1111\} \\ &= \{a(1100) + b(1111) \mid a, b \in \mathbb{Z}_2\} \\ &= \{0(1100) + 0(1111),\ 0(1100) + 1(1111),\ 1(1100) + 0(1111),\ 1(1100) + 1(1111)\} \\ &= \{0000, 1111, 0011, 0011\} \end{aligned}$$

De plus, $\{1100, 1111\}$ est linéairement indépendant, donc c'est une base pour ce code. On écrit ici "1100" pour le vecteur "$\begin{bmatrix}1 & 1 & 0 & 0\end{bmatrix}$".

On a donc une matrice génératrice pour ce code:

$$\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & 1 & 1 & 1\end{bmatrix}$$

On peut trouver une matrice génératrice standard en faisant une opération de rangée:

$$\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & 1 & 1 & 1\end{bmatrix} \xrightarrow[R_2 \to R_2 + 1R_1]{} \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\end{bmatrix}$$

On a la matrice génératrice standard $G = \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\end{bmatrix}$.

Le code est généré par G, car $C = \left\{\mathbf{x}G \mid \mathbf{x} \in \mathbb{R}^2\right\}$. Les quatre possibilités pour $\mathbf{x}$ sont $\{00, 01, 10, 11\}$; en multipliant par G on obtient les quatre mots du code.

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

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

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

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

□

Exercice 15.2. Montrer que le code généré par $\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & 1 & 1 & 1\end{bmatrix}$ est le même code que celui généré par la matrice génératrice standard ci-haut. □

Exemple 15.3. Soit le code généré par la matrice $\begin{bmatrix}1 & 1 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 1 & 1\end{bmatrix}$.

La liste de tous les mots du code est obtenue en pré-multipliant G par tous les vecteurs dans ${\mathbb{Z}_2}^2$. On obtient alors le code $\{000000, 000111, 111000, 111111\}$. La distance minimale est $\delta = 3$. Donc les boules de rayon 1 sont toutes disjointes, et ce code peut détecter et corriger une erreur par mot. □

Une matrice génératrice indique comment transmettre des messages. Alice écrit son message en termes des mots $\{00, 01, 10, 11\}$. Elle veut transmettre son message à Bob, donc pour chaque mot $\mathbf{x}$ elle envoie le mot correspondant du code $\mathbf{y} = \mathbf{x}G$. On imagine que les mots $\mathbf{x}$ représentent le message original, sans aucune redondance: tout pixel compte. Les mots $\mathbf{y}$ représentent la même information, mais plus dispersé: il y a une redondance.

Exemple 15.4. Alice veut envoyer le message "00, 11, 01, 01" à Bob. Alice et Bob ont déjà décidé que leurs communications se feront avec le code C généré par $\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$. Alice fait les calculs suivants.

$$\mathbf{x} = \begin{bmatrix} 0 & 0 \end{bmatrix} \longrightarrow \mathbf{y} = \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 1 & 1 \end{bmatrix} \longrightarrow \mathbf{y} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 0 & 1 \end{bmatrix} \longrightarrow \mathbf{y} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 0 & 1 \end{bmatrix} \longrightarrow \mathbf{y} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Donc Alice envoie "000000, 111111, 000111, 000111". □

Il y a une observation utile pour les codes linéaires. Si $\mathbf{x}_1, \mathbf{x}_2$ sont deux mots (vecteurs) dans un code linéaire (sous-espace) alors $d(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_1 - \mathbf{x}_2, \mathbf{0})$. La distance entre deux mots et égale à la distance entre leur différence et le vecteur zéro. Si $d(\mathbf{x}_1, \mathbf{x}_2) = \delta$, la distance minimale du code, alors $d(\mathbf{x}_1 - \mathbf{x}_2, \mathbf{0}) = \delta$ aussi. Puisqu'il s'agit d'un code linéaire, $\mathbf{x}_1 - \mathbf{x}_2$ est également un mot du code. Donc afin de trouver la distance minimale, il suffit de considérer la distance entre un mot et $\mathbf{0}$. La distance $d(\mathbf{x}, \mathbf{0})$ est le nombre de symboles non-nuls dans $\mathbf{x}$, dit le POIDS de $\mathbf{x}$. On écrit $w(\mathbf{x})$ pour le poids de $\mathbf{x}$. Le POIDS MINIMAL du code est le minimum du poids de tous les mots non-nuls du code.

Théorème 15.5. *Si C est un code linéaire, alors la distance minimale du code est égale au poids minimale du code.* □

L'avantage de ce résultat est qu'on peut plus rapidement calculer la distance minimale: ce n'est que nécessaire de considérer tous les mots, et non tous les paires de mots. (Souvent on peut calculer la distance minimale d'un code linéaire plus directement, en comprenant sa structure particulière.)

Exemple 15.6. Pour le code de l'exemple 15.3, on peut calculer toutes les distances:

$$d(000000, 000111) = 3 \qquad d(000000, 111000) = 3 \qquad d(000000, 111111) = 6$$
$$d(111000, 111111) = 3 \qquad d(000111, 111111) = 3 \qquad d(000111, 111000) = 6$$

On voit que la distance minimale est 3. On aurait pu aussi calculer la liste de tous les poids.

$$w(000111) = 3 \qquad w(111000) = 3 \qquad w(111111) = 6$$

Le poids minimale est 3, donc la distance minimale est aussi 3. □

MATRICE DE CONTRÔLE

Bob ne reçoit pas exactement les mots du code que Alice envoie. Il a besoin d'une méthode d'interpréter les mots reçus.

Si G est une matrice génératrice standard, alors on peut la comprendre comme étant une matrice de la forme $[I_m|A]$: les colonnes de I_m correspondent au colonnes de G avec pivot (ce ne sont pas nécessairement au début), la matrice A représente le "reste" de G. Si G est une matrice de taille $m \times n$, alors A est de taille $m \times (n-m)$. Donc la matrice identité est de taille $m \times m$, ce qui explique la notation I_m.

La MATRICE DE CONTRÔLE, H, est obtenue en écrivant la matrice $-A^T$ dans les colonnes correspondant au pivots de G, et une matrice identité dans les colonnes correspondante au non-pivots de G. Puisque A^T est de taille $(n-m) \times m$, alors H a $n-m$ rangées est la matrice identité ici est de taille $(n-m) \times (n-m)$. Donc on a $H = [-A^T|I_{n-m}]$, de taille $(n-m) \times n$.

La matrice R est obtenue en remplaçant la matrice A dans G avec des zéros. C'est la même taille que G, donc $m \times n$.

Exemple 15.7. On considère le code de l'exemple 15.3. On construit la matrice H en mettant $-A^T$ dans le colonnes pivot de G, et en remplissant le reste avec une identité. On indique les colonnes correspondant au pivots de G en **gras**.

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

$$A = \begin{bmatrix} & 1 & 1 & & 0 & 0 \\ & 0 & 0 & & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$-A^T = \begin{bmatrix} \mathbf{1} & & & \mathbf{0} & & \\ \mathbf{1} & & & \mathbf{0} & & \\ \mathbf{0} & & & \mathbf{1} & & \\ \mathbf{0} & & & \mathbf{1} & & \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

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

$$R = \begin{bmatrix} \mathbf{1} & 0 & 0 & \mathbf{0} & 0 & 0 \\ \mathbf{0} & 0 & 0 & \mathbf{1} & 0 & 0 \end{bmatrix}$$

Note que les matrices G, H, R ont tous $n = 6$ colonnes. Les matrices G et R sont de la même taille, 2×6. Mais la matrice H est de taille 4×6.

Ce code est un code sur le corps $\mathbb{Z}_2$. Donc la négation de A^T se fait par rapport au corps $\mathbb{Z}_2$. Mais dans ce corps $-1 \equiv 1$, donc $-A^T = A^T$ (voir l'exercice 14.8). □

Les matrices H et R sont utiles pour interpréter les mots reçus.

Théorème 15.8. *Si H est la matrice de contrôle qui correspond à la matrice génératrice G, alors $HG^T = \mathbf{0}$. Autrement dit, les rangées de H sont tous orthogonales aux rangées de G. Comme conséquence, si $\mathbf{y} = \mathbf{x}G$ est n'importe quel mot du code, alors $H\mathbf{y}^T = \mathbf{0}$.*

Preuve. On observe que $HG^T = [-A^T|I]\left([I|A]\right)^T = -A^T I + IA^T = \mathbf{0}$. De plus, $H\mathbf{y}^T = H\left(\mathbf{x}G\right)^T = HG^T\mathbf{y} = \mathbf{0}\mathbf{y} = \mathbf{0}$. □

Une façon de comprendre ceci est que le noyau de H est l'espace rangée de G.

L'application est directe. Bob reçoit un mot $\mathbf{z}$, qui n'est peut-être pas exactement le mot $\mathbf{y}$ envoyé par Alice. Il calcul $H\mathbf{z}$: si c'est $\mathbf{0}$, alors $\mathbf{z}$ *est* un mot du code: aucune erreur. Si $H\mathbf{z} \neq \mathbf{0}$ alors il y a eu une erreur.

Exemple 15.9. Rappelant l'exemple 15.4, Bob reçoit le message suivant: "000001, 111111, 000111, 010111". Il dénote ces quatre mots par $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{z}_3$, $\mathbf{z}_4$.

Il connaît la matrice génératrice du code, et aussi la matrice de contrôle (car il a bien étudié l'exemple 15.7). Donc il calcul

$$H\mathbf{z}_1 = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \rightarrow \mathbf{z}_1 \text{ est incorrecte} \qquad H\mathbf{z}_2 = \begin{bmatrix}0\\0\\0\\0\end{bmatrix} \rightarrow \mathbf{z}_2 \text{ est correcte}$$

$$H\mathbf{z}_3 = \begin{bmatrix}0\\0\\0\\0\end{bmatrix} \rightarrow \mathbf{z}_3 \text{ est correcte} \qquad H\mathbf{z}_4 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \rightarrow \mathbf{z}_4 \text{ est incorrecte}$$

Bob sait alors que $\mathbf{z}_1$ et $\mathbf{z}_4$ sont en erreur, mais $\mathbf{z}_2$ et $\mathbf{z}_3$ sont correctes. □

SYNDROME

La matrice de contrôle sert à reconnaître les mots incorrectes, mais elle sert aussi à les corriger. On imagine que Alice envoie $\mathbf{y}$ à Bob, mais Bob reçoit $\mathbf{z}$. On peut écrire $\mathbf{z}$ comme $\mathbf{z} = \mathbf{y} + \mathbf{e}$, où $\mathbf{e}$ est le vecteur de l'erreur. En fait $\mathbf{e} = \mathbf{z} - \mathbf{y}$, mais Bob ne connaît (pour l'instant) ni $\mathbf{e}$ ni $\mathbf{y}$. On voit que

$$H\mathbf{z}^T = H\left(\mathbf{y} + \mathbf{e}\right)^T = H\mathbf{y}^T + H\mathbf{e}^T = \mathbf{0} + H\mathbf{e}^T = H\mathbf{e}^T$$

On dit que $H\mathbf{z}^T$ est le SYNDROME de $\mathbf{z}$. Bob ne peut pas voir l'erreur $\mathbf{e}$, mais il peut connaître le syndrome *de l'erreur*, car $H\mathbf{e}^T = H\mathbf{z}^T$.

Le nombre d'erreurs possibles est relativement petit: Il s'agit de tous les vecteurs $\mathbf{e}$ avec au plus t éléments non-nuls, pour un code qui peut détecter et corriger t erreurs par mot.

Exemple 15.10. Si un code de longueur n sur le corps $\mathbb{Z}_p$ peut détecter et corriger t erreurs, alors le nombre de vecteurs d'erreur est

$$n(p-1) + \binom{n}{2}(p-1)^2 + \cdots + \binom{n}{t}(p-1)^t$$

On à déjà vu cette formule: c'est le nombre de mots dans la boule de rayon t (sans inclure le mot du code lui-même). Voir le théorème 13.11 et la discussion qui la précède. □

Exemple 15.11. Pour le code de l'exemple 15.3, la distance minimale est $\delta = 3$, donc ce code peut corriger une erreur par mot. C'est un code sur le corps $\mathbb{Z}_2$, donc les erreurs possibles sont les vecteurs dans ${\mathbb{Z}_2}^6$ ayant un élément non-nul. Il y en a six. On donne tous les erreurs possibles, ainsi que les syndromes correspondants (les vecteurs sont écrits comme mots pour

conserver l'espace).

$$\begin{aligned}
\mathbf{e}_1 = 000001 \quad &\rightarrow \quad \text{syndrome: } H\mathbf{e}_1 = 0001 \\
\mathbf{e}_2 = 000010 \quad &\rightarrow \quad \text{syndrome: } H\mathbf{e}_2 = 0010 \\
\mathbf{e}_3 = 000100 \quad &\rightarrow \quad \text{syndrome: } H\mathbf{e}_3 = 0011 \\
\mathbf{e}_4 = 001000 \quad &\rightarrow \quad \text{syndrome: } H\mathbf{e}_4 = 0100 \\
\mathbf{e}_5 = 010000 \quad &\rightarrow \quad \text{syndrome: } H\mathbf{e}_5 = 1000 \\
\mathbf{e}_6 = 100000 \quad &\rightarrow \quad \text{syndrome: } H\mathbf{e}_6 = 1100
\end{aligned}$$ □

L'exemple précédant montre une chose très pratique: si un code sur $\mathbb{Z}_2$ corrige une erreur, alors les syndromes sont exactement les colonnes de H. La tâche de Bob est simplifié: il calcul le syndrome de chaque mot reçu. Si le syndrome est $\mathbf{0}$, alors le mot est correct. Si le syndrome n'est pas zéro, alors le syndrome est une des colonnes de H. De plus, la colonne de H correspond exactement à l'erreur! Donc il sait $\mathbf{e}$ et il peut donc calculer $\mathbf{y} = \mathbf{z} - \mathbf{e}$ et récupérer le message original.

Le principe est pareil en général, par contre les erreurs possibles sont tous les vecteurs de ${\mathbb{Z}_p}^n$ ayant poids au plus $\lfloor(\delta - 1)/2\rfloor$. Donc il y a plus de syndromes! Pour les codes à grande distance minimales, on cherche des autres méthodes de "décoder", mais pour nous, l'approche simple suffira.

Exemple 15.12. Rappelant l'exemple 15.9, Bob a reçu le message "000001, 111111, 000111, 010111" et il a calculé les syndromes "0001, 0000, 0000, 1000".

Le premier syndrome 0001 est exactement la dernière colonne de H. Donc c'est le dernier symbole de ce mot qui est en erreur. Donc Bob sait que 000001 était vraiment 000000.

Le deuxième syndrome est 0000, donc le deuxième mot est correcte.

Le troisième syndrome est 0000, donc le troisième mot est correcte.

Le quatrième syndrome est 1000, qui est deuxième colonne de H. Donc c'est le deuxième symbole de ce mot qui est en erreur. Donc Bob sait que 010111 était vraiment 000111.

Bob sait alors que les mots transmis par Alice étaient "000000, 111111, 000111, 000111". □

Bob a corrigé le message. Il veut maintenant récupérer le message original d'Alice. C'est la matrice R qui fait ceci.

Théorème 15.13. *Soit un code généré par G, avec matrice de contrôle H et matrice de "récupérage" R. Si $\mathbf{y} = \mathbf{x}G$ est un mot du code, alors $\mathbf{x}^T = R\mathbf{y}^T$ (ou $\mathbf{x} = \mathbf{y}R^T$).*

Preuve. On a directement que $RG^T = I$. Donc $R\mathbf{y}^T = R\left(\mathbf{x}G\right)^T = RG^T\mathbf{x}^T = I\mathbf{x}^T = \mathbf{x}^T$. □

Exemple 15.14. Rappelant l'exemple 15.12, Bob a reçu le message "000001, 111111, 000111, 010111", il a calculer les syndromes "0001, 0000, 0000, 1000", et il a corrigé pour trouver les mots transmis par Alice "000000, 111111, 000111, 000111". Il peut maintenant calculer le message original.

$$\begin{aligned}
\mathbf{y}_1 = 000000 \quad &\rightarrow \quad \mathbf{x}_1 = \mathbf{y}_1R^T = 00 \\
\mathbf{y}_2 = 111111 \quad &\rightarrow \quad \mathbf{x}_2 = \mathbf{y}_1R^T = 11 \\
\mathbf{y}_3 = 000111 \quad &\rightarrow \quad \mathbf{x}_3 = \mathbf{y}_1R^T = 01 \\
\mathbf{y}_4 = 000111 \quad &\rightarrow \quad \mathbf{x}_4 = \mathbf{y}_1R^T = 01
\end{aligned}$$

Bob sait alors que le message original de Alice était "00, 11, 01, 01". □

Algorithme 15.15. *Soit un code linéaire avec matrice génératrice G de taille $m \times n$ sur un corps $\mathbb{Z}_p$. On calcul la matrice de contrôle H et la matrice de "récupérage" R.*

Si Alice veut envoyer un message à Bob, elle suit les étapes suivants.

1. *Alice écrit son message original en termes de mots de longueur m sur $\mathbb{Z}_p$: chaque mot est un vecteur $\mathbf{x}$ dans l'espace vectoriel ${\mathbb{Z}_p}^m$.*
2. *Pour chaque mot $\mathbf{x}$, Alice calcul $\mathbf{y} = \mathbf{x}G$; c'est un mot du code, sous-espace de ${\mathbb{Z}_p}^n$.*
3. *Alice envoie chaque $\mathbf{y}$ à Bob.*

Si Bob reçoit un message de Alice, il suit les étapes suivants.

1. *Bob reçoit des mots de Alice; ce sont des mots $\mathbf{z}$ en ${\mathbb{Z}_p}^n$.*
2. *Pour chaque mot $\mathbf{z}$, Bob calcul le syndrome $H\mathbf{z}^T$.*
3. *Si le syndrome est $\mathbf{0}$, alors le mot est correcte, et $\mathbf{y} = \mathbf{z}$.*
4. *Si le syndrome n'est pas $\mathbf{0}$, alors le mot est incorrect. Il connaît tous les erreurs possibles. Parmi ces erreurs, il choisi celui qui à le même syndrome que $\mathbf{z}$. C'est cette erreur $\mathbf{e}$ qui a perturbé le mot $\mathbf{y}$, donc il peut maintenant calculer $\mathbf{y} = \mathbf{z} - \mathbf{e}$.*
5. *Bob connaît maintenant $\mathbf{y}$, le mot correcte transmis par Alice. Il obtient $\mathbf{x} = \mathbf{y}R^T$.* □

Note que Bob doit passer à travers une liste de tous les erreurs possibles. C'est beaucoup plus rapide que de passer à travers une liste de tous les mots possibles. Mais si le nombre d'erreurs par mot est plus élevé, on a besoin d'autres techniques.

Le message original est écrit en ${\mathbb{Z}_p}^m$; les mots du code sont écrit en ${\mathbb{Z}_p}^n$. On dit que m est la DIMENSION du code et que n et la LONGUEUR du code. On a déjà vu la longueur, mais dans le contexte des codes linéaires on comprend que "dimension" mesure la quantité d'information dans le message original. Le nombre de mots possibles dans le code est exactement p^m. La "longueur" mesure la quantité de redondance ajoutée pour donner une message transmissible (et corrigeable, au besoin).

r

Exercice 15.16. En utilisant encore le même code que l'exemple 15.3, Bob renvoie un message à Alice. Alice reçoit "100111, 000101, 111111, 110111". Quel est le message original de Bob? Donner les syndromes, les mots corrigés et le message original. □

Exercice 15.17. Les photos prises par les sondes spatiales *Voyager* ont été transmises à la Terre en utilisant un code de Golay. Ce code prend des mots en ${\mathbb{Z}_2}^{12}$ (donc 12 bits) et les transforme en mot du code qui est un sous-espace de ${\mathbb{Z}_2}^{24}$ (donc 24 bits). La distance minimale de ce code est 8. Donner les tailles des matrices G, H et R. Combien d'erreurs par mot est-ce que ce code peut détecter? Combien d'erreurs par mot est-ce que ce code peut corriger?

Combien de syndromes distinctes sont possibles avec ce code? Bob devrait passer à travers cette liste pour chaque mot qu'il reçoit. Est-ce que vous pensez que c'est une méthode efficace pour un code de distance minimale $\delta = 8$? Note que c'est encore beaucoup plus rapide que de passer à travers une liste de tous les mots possibles dans ${\mathbb{Z}_2}^{24}$. Néanmoins, il existe des méthodes même plus rapides de "décoder" ce code. □

CODES DE HAMMING

Un code de Hamming est un code linéaire parfaite. On considère les code Hamming BINAIRE, voulant dire sur le corps $\mathbb{Z}_2$. On obtient sa matrice de contrôle en écrivant tous les vecteurs non-nuls de ${\mathbb{Z}_2}^k$ comme colonnes, pour un entire k. On peut alors déterminer G et R.

Exemple 15.18. Pour $k = 3$, la matrice de contrôle du code de Hamming est

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

On calcul alors que

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

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

On voit que la dimension est $m = 3$ et la longueur est $n = 7$ (c'est la taille de G, 3×7). En général, $k = n - m$, $n = 2^k - 1$ et $m = 2^k - k - 1$.

Ici, on a choisi d'écrire la matrice H avec la matrice identité à la fin. Ce n'est pas strictement nécessaire, il suffit de l'écrire dans un ordre où l'on peut "voir" la matrice identité. Mais dans l'ordre choisi, on aura automatiquement une matrice génératrice standard. □

On peut montrer que le poids minimal du code de Hamming est 3. C'est vrai en toute dimension. Donc ce code peut détecter et corriger une erreur par mot. Ce code comporte 2^m mots. On a que $|C|\,|B_1| = (2^m)\,(1 + n) = 2^{n-k}2^k = 2^n$. La borne d'un code parfaite est atteinte.

Exercice 15.19. Alice veut envoyer le message "0001, 0100, 0110" à Bob en utilisant un code de Hamming binaire de dimension $m = 4$. Donner les mots du code qu'elle transmet. □

Exercice 15.20. Bob reçoit le message "0011010, 1100110, 1110100", envoyé selon le code Hamming binaire de dimension $m = 4$. Déterminer si les mots sont correctes. Si nécessaire, les corriger. Donner le message original (en ${\mathbb{Z}_2}^4$). □

Exercice 15.21. Donner la dimension m et la longueur m pour le code Hamming avec $k = 2$. Donner les matrices G, H et R pour ce code. Faire un graphe de ce code, montrant les boules de rayon 1. Vérifier que les boules sont toutes disjointes et que le code est parfait. □

Exercice 15.22. Donner la dimension m et la longueur m pour le code Hamming avec $k = 4$. Donner les tailles de matrices G, H et R. Les décrire (c'est peut-être un peu long de les écrire explicitement). □