TD/TP – Codes correcteurs

Il existe un certain nombre de commandes sur Sage consacrées aux codes correcteurs ; le tableau ci-dessous donne les plus importantes (nous en verrons d'autres ci-bas).

argument(s)	commande	sortie
$G \in \mathrm{M}_{k,n}(\mathbb{F}_q)$	`LinearCode(G)`	code de matrice génératrice G
code C	`C.generator_matrix()`	matrice génératrice de C
code C	`C.parity_check_matrix()`	matrice vérificatrice de C
$c \in C$	`c.hamming_weight()`	poids de c
code C	`C.minimum_distance()`	distance minimale de C
$r, q \geqslant 1$	`codes.HammingCode(GF(q),r)`	code de Hamming de longueur $\frac{q^r-1}{q-1}$
$n \geqslant 1$, $P \in \mathbb{F}_q[X]$	`codes.CyclicCode(n,P)`	code cyclique $(P) \subseteq (\mathbb{F}_q)^n$

Exercice 1. (commandes de base) Considérons la matrice

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \in \mathrm{M}_{4,6}(\mathbb{F}_2)$$

et soit C le code de matrice génératrice G *. Avant de tester les commandes de base :
- déterminer la dimension et le nombre de mots du code C ;
- déterminer une matrice de contrôle de C ;
- calculer la distance minimale de C, et en déduire le nombre d'erreurs que C peut corriger.

`parent` Vérifier tout cela sous `Sage` en construisant le code C. Quel type ont les éléments du code ?

Exercice 2. (matrices génératrices, vérificatrices) Rappelons que des matrices génératrices $A \in \mathrm{M}_{k,n}(\mathbb{F}_q)$ et vérificatrices $B \in \mathrm{M}_{n-k,n}(\mathbb{F}_q)$ d'un code linéaire $\mathrm{C} \subseteq (\mathbb{F}_q)^n$ de dimension k sont reliées par l'égalité $B \cdot {}^tA = 0$, avec $\mathrm{rang}(A) = k$ et $\mathrm{rang}(B) = n - k$. Écrire un programme qui :

`right_kernel`
- prend en entrée une matrice vérificatrice et renvoie une matrice génératrice correspondante ;
- prend en entrée une matrice génératrice et renvoie une matrice vérificatrice correspondante.

Composer ces deux programmes renvoie-t-il la matrice d'origine ? Écrire un programme `ver_to_code` qui prend en entrée $B \in \mathrm{M}_{n-k,n}(\mathbb{F}_q)$ de rang $n - k$ et renvoie un code de matrice vérificatrice B.

Montrer que tout code correcteur de longueur n et de dimension k admet une matrice génératrice de la forme $(\mathrm{I}_k \quad G)$ où $G \in \mathrm{M}_{k,n-k}(\mathbb{F}_q)$. Quel est l'intérêt pratique d'une telle matrice génératrice ? En déduire aisément une matrice vérificatrice.

Exercice 3. (distance minimale) Écrire un programme qui :

`ambient_space`
- qui prend en entrée un code linéaire et renvoie son type (n, k, d) ;

`binomial`
- prend en entrée un code linéaire C, et nous renvoie le plus grand entier t tel que $2t + 1 \leqslant d(\mathrm{C})$,

`base_ring` accompagné du mot « parfait » si C est t-correcteur parfait. Tester la perfection des codes de Hamming binaires et des codes de répétition de longueur inférieure à 32.

On rappelle qu'un code linéaire cyclique $\mathrm{C} \subseteq (\mathbb{F}_q)^n$ a pour image dans $\mathbb{F}_q[X]/(X^n - 1)$ un idéal (P) (par l'isomorphisme naturel d'espaces vectoriels qui envoie base canonique sur base canonique). Ceci définit une bijection $\mathrm{C} \mapsto P$ de l'ensemble des codes linéaires cycliques de longueur n dans l'ensemble des diviseurs unitaires de $X^n - 1$ (dans $\mathbb{F}_q[X]$).

∗. Attention : sous `Sage`, et par une convention assez suivie, les vecteurs qui engendrent le code sont les *lignes* de la matrice génératrice. Contrairement à l'usage du cours, nous suivrons cette convention ici, d'où les transposées plus bas.

Quand n est premier à q, cet ensemble est également en bijection avec l'ensemble des parties de $\mu_n(\bar{\mathbb{F}}_q) = \{\text{racines } n\text{-ièmes de 1 dans } \bar{\mathbb{F}}_q\}$ stables par le morphisme de Frobenius $x \mapsto x^q$, grâce à l'application $P \mapsto P^{-1}(\{0\})$. Si l'on fixe α un générateur de $\mu_n(\bar{\mathbb{F}}_q)$, alors la bijection $\mathbb{Z}/n\mathbb{Z} \simeq \mu_n(\bar{\mathbb{F}}_q)$ donnée par $\bar{m} \mapsto \alpha^m$ en induit une autre, entre l'ensemble des parties I de $\mathbb{Z}/n\mathbb{Z}$ stables par la multiplication par q et les codes linéaires cycliques $\mathrm{C} \subseteq (\mathbb{F}_q)^n$ (on associe à I l'idéal de $\mathbb{F}_q[X]/(X^n-1)$ engendré par $\prod_{m\in \mathrm{I}} (X - \alpha^m)$).

Exercice 4. (codes cycliques)

1. Écrire un programme qui décale d'un cran les coordonnées d'un vecteur.
2. Écrire un programme qui détermine si un code est cyclique ; le tester sur le code de l'exercice 1, sur le code de parité de longueur n, et sur `codes.HammingCode(3,GF(2))`. Justifier le résultat.

`PolynomialRing`

3. Tester `codes.CyclicCode` avec 1, $X-1$, $\frac{X^n-1}{X-1}$ et X^n-1. Que reconnaît-on ?
4. Tester `codes.CyclicCode` avec un polynôme qui ne divise pas X^n-1. Qu'obtient-on ?

`of_degree` `is_irreducible()`

5. Trouver tous les polynômes irréductibles de degré 3 dans $\mathbb{F}_2$, et vérifier que les codes cycliques de longueur 7 qu'ils induisent sont isomorphes † (utiliser `is_permutation_equivalent`). Lequel correspond au code de Hamming standard de type (7,4,3) ? Comparer avec les codes cycliques induits par des polynômes réductibles de degré 3.

Exercice 5. (encodage) On rappelle l'encodage naturel : $(m_0, \dots, m_{k-1}) \in (\mathbb{F}_q)^k \mapsto \sum_{i=0}^{k-1} m_i \vec{v}_i \in (\mathbb{F}_q)^n$, où $(\vec{v}_i)_{0\leqslant i\leqslant k-1}$ est une base du code linéaire C (qu'on peut écrire $(m_0, \dots, m_{k-1}) \mapsto \sum_{i=0}^{k-1} m_i X^i P$ si C est vu comme un idéal engendré par P, en prenant sa base canonique).

1. Implémenter l'encodage naturel rappelé ci-dessus.
2. Supposons C cyclique, de type (n,k,d), engendré par P (donc $\deg(P) = n-k$). Soit R le reste de la division euclidienne de $X^{n-k}M$ par P, où M est l'image de $(m_0, \dots, m_{k-1})$ dans $\mathbb{F}_q[X]_{k-1}$. Justifier que $X^{n-k}M - R$ appartient au code. En déduire un nouvel encodage de $(m_0, \dots, m_{k-1})$. Comparer les encodages obtenus pour un même message, en prenant en exemple le code cyclique de longueur 6 engendré par X^3+1. Quel est son avantage par rapport au précédent ?
3. Tester le dernier encodage avec les codes cycliques engendrés par les diviseurs $X-1$ et $\frac{X^n-1}{X-1}$. Qu'obtient-on ? Justifier que les deux encodages définis ci-dessus coïncident pour un bon choix de matrice génératrice.

Notons l'existence de `encoders_available` et `add_encoder` pour voir les encodages existants pour un code donné, voire en créer. On en utilise un avec `encode(message,encodeur)` (tester la commande).

Exercice 6. (correction des erreurs sous Sage) Nous avons vu en cours comment, étant donné un code 1-correcteur parfait, nous pouvons détecter et corriger une erreur.

1. Écrire un programme `bruitage` qui prend en entrée un entier k et un code, et renvoie un mot du code choisi aléatoirement, dont on a modifié k coordonnées.
2. Écrire un programme qui prend un entrée un mot d'un code, et renvoie le mot corrigé s'il a au plus une erreur. Vérifier son fonctionnement avec le code de Hamming binaire de longueur 7 (`codes.HammingCode(3,GF(2))`) et les mots suivants (vérifier s'ils sont dans le code avant) :

   (1,0,1,0,1,0,1), (1,1,1,1,0,0,0), (0,0,0,0,1,1,1), (1,0,1,1,0,0,0), (0,0,1,1,0,1,0), …

   On peut multiplier les exemples à l'aide de `bruitage` ; introduire deux erreurs dans (1,0,1,0,1,0,1), et décoder (prendre par exemple (1,0,1,0,0,0,0)). Obtient-on à nouveau (1,0,1,0,1,0,1) ?

†. Des codes C et D de même longueur n sont isomorphes s'il existe $\sigma \in \mathfrak{S}_n$ tel que $c \in \mathrm{C} \Leftrightarrow \sigma \cdot c \in \mathrm{D}$.

Quelques commandes par défaut permettent de corriger les erreurs (`decode`, `decode_to_code` ; les méthodes de décodage peuvent être dévoilées avec `decoders_available`) et de décoder un mot (`decode_to_message`, `unencode`). Les tester avec les vecteurs ci-dessus. La dernière commande corrige-t-elle les erreurs avant de décoder ; on peut désactiver cette vérification en ajoutant l'argument `nocheck=False`. Que se passe-t-il alors, si on veut décoder un mot qui n'est pas dans le code ?

Exercice 7. (correction des erreurs) On pose $n = 2^r - 1$, et α une racine primitive n-ième de l'unité. Soit C le code linéaire cyclique sur $\mathbb{F}_2$, de longueur n, engendré par la plus petite partie I de $\mathbb{Z}/n\mathbb{Z}$ contenant $\{1,2,3,\dots,2t\}$ et stable par $\langle 2\rangle$. Ce code est (au moins) t-correcteur, engendré par $\prod_{i\in \mathrm{I}}(X-\alpha^i) \in \mathbb{F}_2[X]$, et $m \in \mathrm{C}$ si, et seulement si $m(\alpha^j) = 0$ pour tout $j \in [\![1,2t]\!]$.

Le problème est le suivant : soit $m' = m + e$ est un message bruité reçu ; sachant que le poids de e est inférieur ou égal à t, comment trouver m ?

Pour cela, on définit le *syndrome* de m' comme la suite

$$(s_1,\dots,s_t) = (m'(\alpha^j))_{j\in[\![1,2t]\!]} = (e(\alpha^j))_{j\in[\![1,2t]\!]};$$

remarquons que si $e = X^{r_1} + \cdots + X^{r_f}$ avec $f \leqslant t$, alors $s_j = \sum_{i=1}^{f}(\alpha^{r_i})^j = \sum_{i=1}^{f}(x_i)^j$. Si on arrive à déterminer e à partir du syndrome de m', alors on sait retrouver m. Pour ce faire, on introduit les polynômes syndrome S (connu) et de localisation σ (à déterminer) :

$$S = \sum_{j=1}^{2t} s_j Z^{j-1}, \quad \sigma = \prod_{i=1}^{f}(1-x_iZ) = \prod_{i=1}^{f}(1-\alpha^{r_i}Z).$$

Voyons comment la connaissance de S, facile à calculer, implique celle de σ, et donc celle des α^{r_i} (dont on tire aisément la connaissance des r_i, puis de e).

1. Montrer qu'il existe $w \in \mathbb{F}_2[Z]$ de degré strictement inférieur à t, premier avec σ, et tel que $S\sigma \equiv w \bmod Z^{2t}$.
2. Montrer que si $\sigma', w' \in \mathbb{F}_2[Z]$ sont deux polynômes tels que $\deg(\sigma') \leqslant t$, $\deg(w') < t$ et $S\sigma' \equiv w' \bmod Z^{2t}$, alors il existe $C \in \mathbb{F}_2[Z]$ tel que $\sigma' = C\sigma$ et $w' = Cw$.
3. En déduire qu'en pratiquant l'algorithme d'Euclide étendu sur Z^{2t} et S, alors w égale le premier reste P de degré strictement inférieur à t (à une constante multiplicative près), et σ est un des coefficients de Bézout de la relation $P = AZ^{2t} + BS$ obtenue *via* l'algorithme (à une constante multiplicative près).
4. Implémenter l'algorithme sous Sage, et le tester sur un code cyclique 2-correcteur.

Exercice 8. (diviseurs de $X^n - 1$, parties de $\mathbb{Z}/n\mathbb{Z}$ stables par $\langle q\rangle$)

1. Comment obtenir une liste de tous les facteurs irréductibles de $X^n - 1$ sur $\mathbb{F}_q$? Et une liste de tous les diviseurs de $X^n - 1$ sur $\mathbb{F}_q$? Comment produire un générateur de $\mu_n(\bar{\mathbb{F}}_q)$?
2. Écrire un programme qui :
   `R.quo(I)`
   - convertit un vecteur de $(\mathbb{F}_q)^n$ en un polynôme de $\mathbb{F}_q[X]/(X^n-1)$ et inversement ;
   - détermine P en partant de C ;
   - prend en entrée $\mathrm{J} \subseteq \mathbb{Z}/n\mathbb{Z}$ et renvoie la plus petite partie stable par $\langle q\rangle$ contenant J ;
   - prend en entrée un générateur α de $\mu_n(\bar{\mathbb{F}}_q)$ et une partie de $\mathbb{Z}/n\mathbb{Z}$, puis renvoie le code cyclique associé à la plus petite partie de $\mathbb{Z}/n\mathbb{Z}$ stable par $\langle q\rangle$ la contenant.

Exercice 9. (codes de Hamming binaire) On prend $q = 2$, $r \geqslant 2$ et $n = 2^r - 1$.

1. Ce tour de prestidigitation est un classique : on demande à un interlocuteur de penser à un nombre entre 0 et 15, et on lui pose les questions suivantes en autorisant au plus un mensonge :
   (a) Est-ce que le nombre est supérieur ou égal à 8 ?
   (b) Est-ce que le nombre est dans $\{4, 5, 6, 7, 12, 13, 14, 15\}$?
   (c) Est-ce que le nombre est dans $\{2, 3, 6, 7, 10, 11, 14, 15\}$?
   (d) Est-ce que le nombre est impair ?
   (e) Est-ce que le nombre est dans $\{1, 2, 4, 7, 9, 10, 12, 15\}$?
   (f) Est-ce que le nombre est dans $\{1, 2, 5, 6, 8, 11, 12, 15\}$?
   (g) Est-ce que le nombre est dans $\{1, 3, 4, 6, 8, 10, 13, 15\}$?

   Le prestidigitateur note alors $m_i = 1$ si la réponse à la i-ième question est « oui », $m_i = 0$ sinon, et pose $m = (m_1, \dots, m_7) \in \mathbb{F}_2^7$. Il calcule ensuite les quantités suivantes :

   $$a = m_1 + m_3 + m_5 + m_7, \quad b = m_2 + m_3 + m_6 + m_7, \quad c = m_4 + m_5 + m_6 + m_7.$$

   Soit k l'entier qui s'écrit $\overline{cba}$ en base 2. Si $k = 0$, alors il n'y a pas de mensonge. Sinon, k indique la question où l'interrogé a menti ; on corrige donc m en changeant m_k, et $\overline{m_1 m_2 m_3 m_4}$ est le nombre à deviner écrit en base 2.

   Expliquer ce tour. On méditera utilement sur le code correcteur de matrice vérificatrice :

   $$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \in \mathrm{M}_{3,7}(\mathbb{F}_2)$$

2. Posons $k = 2^r - 1 - r$. Créer un tour de magie devinant un nombre entre 0 et $2^k - 1$ et permettant de corriger au plus un mensonge, à l'aide d'un code binaire de Hamming. Le programmer pour $r = 2$ (que reconnaît-on ?), $r = 3$ et $r = 4$.