Travaux dirigés : Codes correcteurs d'erreurs

1 Premières manipulations

1. Espaces vectoriels.
   (a) Utiliser la commande `VectorSpace` pour construire l'espace vectoriel $\mathbb{K}^n$ pour un corps fini $\mathbb{K}$ et un entier n de votre choix.
   (b) Utiliser la commande `MatrixSpace` pour construire l'espace vectoriel $\mathcal{M}_{k,n}(\mathbb{K})$ pour un corps fini $\mathbb{K}$ et des entiers $1 \leq k \leq n$ de votre choix.
   (c) Utiliser la commande `random_element` pour générer aléatoirement des éléments de $\mathcal{M}_{k,n}(\mathbb{K})$ jusqu'à obtenir une matrice `M` de rang k.
   Que renvoie `span(M)` ? Que se passe-t-il si `M` n'est pas de rang k ?
2. Poids et distance de Hamming.
   (a) Écrire une fonction qui calcule le poids de Hamming d'un élément de $\mathbb{K}^n$ / d'un n-uplet d'éléments de $\mathbb{K}$ / d'une liste d'éléments de $\mathbb{K}$.
   (b) Écrire une fonction calculant la distance de Hamming entre deux mots de longueur n sur $\mathbb{K}$.
3. Distance minimale.
   Soit C une partie de $\mathbb{K}^n$.
   (a) Écrire une fonction qui calcule le plus petit poids de Hamming d'un élément non nul de C.
   (b) Écrire une fonction qui calcule la plus petite distance de Hamming entre deux mots distincts de C.
   (c) Vérifier sur des exemples que ces deux minimums coincident si C est un sous-espace vectoriel de $\mathbb{K}^n$. Trouver un exemple où ces deux minimums sont distincts.
4. Matrice de contrôle.
   Soit $G \in \mathcal{M}_{k,n}(\mathbb{K})$. Écrire un programme qui détermine une matrice H telle que $\mathrm{Im}(H) = \ker(G)$, d'abord en utilisant la commande `right_kernel`, puis en passant par la forme échelon de G (on pourra supposer que les k premières colonnes de G sont linéairement indépendantes).

   A-t-on alors $\mathrm{Im}(v \mapsto vG) = \ker(v \mapsto vH)$?

2 Codes de Hamming

On rappelle la construction d'un code de Hamming sur $\mathbb{F}_2$ de paramètres $(n, k) = (2^r - 1, 2^r - 1 - r)$:
- on prend une matrice $H = \begin{pmatrix} P \\ I_r \end{pmatrix}$ obtenue en listant en ligne tous les mots non nuls de $\mathbb{F}_2^r$, en terminant par $10\ldots00$, …, $00\ldots01$
- on pose $G = \left(\begin{array}{c|c} I_{2^r-1-r} & P \end{array} \right)$ la matrice génératrice correspondante ; ses lignes engendrent le code C voulu.

1. Écrire un programme qui prend en entrée un entier r et renvoie un code de Hamming correspondant.
2. Écrire la fonction d'encodage $\mathbb{F}_2^{2^r-1-r} \to \mathbb{F}_2^{2^r-1}$, puis la fonction de décodage $\mathbb{F}_2^{2^r-1} \to \mathbb{F}_2^{2^r-1-r}$ correspondante, qui doit pouvoir corriger jusqu'à une erreur de transmission.

3. Un canal de transmission complet.
   Les fonctions `ascii_to_bin` et `bin_to_ascii` du module `sage.crypto.util` permettent de convertir une chaîne de caractères en suite de bits et inversement. On peut utiliser la commande `int(str(b[j]))` pour récupérer le j-ème bit d'une suite de bits b.
   (a) Écrire une fonction qui prend en entrée une chaîne de caractères, la convertit en suite de bits, regroupe ces bits en blocs de 11 (en complétant éventuellement par des 0), encode chacun de ces blocs avec un code de Hamming $(11, 15)$ et renvoie la concaténation des mots du code obtenu.
   (b) Écrire une fonction qui prend en entrée un mot sur $\mathbb{F}_2$ et un paramètre p, et simule des erreurs de transmission en modifiant chaque caractère du mot avec une probabilité p.
   (c) Écrire une fonction qui prend en entrée un mot sur $\mathbb{F}_2$ (dont la longueur est un multiple de 15), le découpe en blocs de 15 bits, décode et éventuellement corrige chacun de ces blocs avec le code de Hamming $(15, 11)$ ci-dessus, puis convertit en chaîne de caractères la concaténation des blocs décodés.
   (d) Tester sur des exemples la composition des trois fonctions ci-dessus.

3 Codes polynomiaux

Dans la suite, on identifie $\mathbb{F}_q^n$ avec $\mathbb{F}_q[X]_{n-1}$ via la bijection $a_0a_1 \dots a_{n-1} \mapsto a_0X^{n-1} + a_1X^{n-2} + \dots + a_{n-1}$.
On rappelle que pour $g \in \mathbb{F}_q[X]$ de degré $n-k$, le code polynomial de paramètre (n, k) engendré par g est l'ensemble

$$C_g = \{gh : h \in \mathbb{F}_q[X]_{k-1}\} = \{P \in \mathbb{F}_q[X]_{n-1} : g \mid P\}.$$

Un tel code est cyclique si $g \mid X^n - 1$; réciproquement, tout code linéaire cyclique (c'est-à-dire invariant par permutation circulaire des coordonnées) de paramètre (n, k) est de la forme C_g pour un certain polynôme g de degré $n - k$ avec $g \mid X^n - 1$.

1. Écrire des fonctions de conversion entre $\mathbb{F}_q^n$ et $\mathbb{F}_q[X]_{n-1}$, passant de $a_0a_1 \dots a_{n-1}$ à $a_0 + a_1X + \dots a_{n-1}X^{n-1}$.
2. Déterminer tous les codes polynomiaux binaires de paramètres $(7, 4)$ et donner leur matrice génératrice (sous forme standard) et leur distance minimale. Lesquels sont cycliques ? Lesquels sont de Hamming ? Obtient-on ainsi tous les codes de Hamming $(7, 4)$?
3. Le protocole Bluetooth utilise le polynôme $X^8 + X^7 + X^5 + X^2 + X + 1 \in \mathbb{F}_2[X]$. Quelle est la plus petite valeur de n pour laquelle ce polynôme engendre un code cyclique ?

4 Codes de Reed-Solomon

Pour construire un code de Reed-Solomon de paramètre $(n, k, n + 1 - k)$ sur $\mathbb{F}_q$, avec $1 \leq k \leq n$, on commence par choisir n éléments distincts $x_0, \dots, x_{n-1} \in \mathbb{F}_q$ (ce qui impose $q \geq n$). On pose alors $C = \{(P(x_0), \dots, P(x_{n-1})) \in \mathbb{F}_q^n : P \in \mathbb{F}_q[X]_{k-1}\}$.

1. Redémontrer que C est bien de paramètres $(n, k, n + 1 - k)$ (code MDS).
2. On suppose fixé un choix de $x_0, \dots, x_n$. Écrire un programme qui prend en entrée un mot $w = a_0a_1 \dots a_{k-1}$ et renvoie l'unique mot du code C commençant par w.
3. En s'aidant du programme précédent, écrire un test d'appartenance à C.

4. Un exemple concret.
   Comme à la question 2.3, écrire des fonctions pour convertir une chaîne de caractères en blocs de 12 éléments de $\mathbb{F}_{16}$, coder ces blocs avec l'unique code de Reed-Solomon de paramètres $(16, 12, 5)$, simuler des erreurs de transmission, décoder et corriger les blocs et retrouver (ou non) le message de départ.
   Pour corriger au plus 2 erreurs, on pourra procéder par recherche exhaustive.

Un algorithme de décodage.

On note $t = \lfloor (n-k)/2 \rfloor$ la capacité de correction d'erreurs du code C. Soit $w = (P(x_0), \dots, P(x_{n-1}))$ un mot du code (avec $P \in \mathbb{F}_q[X]_{k-1}$). On considère le message reçu $w' = (y_0, \dots, y_{n-1}) = w + e \in \mathbb{F}_q^n$, avec e un vecteur d'erreurs.

— On détermine deux polynômes non nuls $Q_1, Q_2 \in \mathbb{F}_q[X]$ tels que

$$\begin{cases} \deg(Q_1) \leq k - 1 + \lceil \frac{n-k}{2} \rceil \\ \deg(Q_2) \leq \lceil \frac{n-k}{2} \rceil \\ Q_1(x_i) - y_i Q_2(x_i) = 0 \ \text{ pour tout } 0 \leq i \leq n-1 \end{cases}$$

— S'il y a moins de t erreurs (c'est-à-dire que le poids de Hamming de e est inférieur ou égal à t), alors $Q_2 \mid Q_1$ et $P = Q_1/Q_2$, ce qui permet de retrouver w. De plus, pour tout i où $y_i \neq P(x_i)$, le point x_i est racine de Q_2 (on dit que Q_2 est un polynôme localisateur des erreurs).

5. Expliquer comment trouver Q_1 et Q_2 en résolvant un système linéaire, dont on justifiera qu'il admet toujours des solutions non nulles.
6. Démontrer les assertions du deuxième point. On pourra considérer le polynôme $Q_1 - PQ_2$.
7. Quelle est la complexité de cette méthode de décodage ? Comparer avec une recherche exhaustive.
8. Implémenter cet algorithme et le tester sur des exemples.