UFR Sciences
M2 Calcul Formel

Martin WEIMANN

TP 6 : Codes correcteurs linéaires

Avertissement : Pour implémenter un code linéaire C de manière complète, il faut a priori déterminer des fonctions de codage, de correction et de décodage

$$\texttt{Code} : \mathbb{F}^k \to \mathbb{F}^n \qquad \texttt{Corrige} : \mathbb{F}^n \to \mathbb{F}^n \qquad \texttt{Decode} : \mathbb{F}^n \to \mathbb{F}^k$$

Pour tester ce code, on considérera un mot source $m \in \mathbb{F}^k$ à envoyer. On calculera $c = \texttt{Code}(m)$. On cherchera alors à corriger puis décoder le mot reçu $r \in \mathbb{F}^n$ que l'on construira à partir de c en introduisant artificiellement un nombre donné t d'erreurs. On vérifiera alors que le message reçu m' vérifie $m' = m$ dés lors que le nombre d'erreurs ne dépasse pas la borne théorique.

On pourra utiliser `V=VectorSpace(GF(p),n)` pour déclarer l'espace vectoriel $\mathbb{F}_p^n$. La liste des vecteurs de V est `V.list()`. Explorer les divers attributs associés à un (sous) espace vectoriel V (touche tab).

Il existe une bibliothèque Sage pour les codes. Taper `from sage.coding.codes_catalog import *`, puis regarder `codes?` pour voir la liste des codes accessibles et leurs fonctionnalités.

Exercice 1 Implémenter une fonction `Transmission` qui prend en entrée un mot $m \in \mathbb{F}_2^n$ (une liste de "bits") et un entier t et retourne un mot modifié $r \in \mathbb{F}_2^n$ avec au plus t erreurs (disposées aléatoirement).

Exercice 2 (Code de parité) Implémenter le codage $\mathbb{F}^k \to \mathbb{F}^{k+1}$ du code de parité. Lister les mots du code pour $k = 3$. Combien d'erreurs ce code peut-il détecter ? Corriger ?

Exercice 3 (Code de Hamming) Implémenter (codage, correction, décodage) le code de Hamming $C(s)$ de longueur $2^s - 1$. On utilisera la matrice de contrôle pour la correction d'erreurs et la matrice génératrice pour le codage et le décodage. Combien d'erreurs le code peut-il corriger ? Vérifier expérimentalement.

Exercice 4 (Correction par syndrome) Dans ce qui suit, on se donne en entrée une matrice de contrôle $H \in \mathcal{M}_{m,n}(\mathbb{F}_2)$ d'un code linéaire C.

1. Implémenter la correction par syndrome du code C (calculer les listes des syndromes et des leaders du code une fois pour toutes en variable globale).
2. Ecrire une fonction `TestSyndrome(H,t,N)` qui calcule le nombre de corrections correctes sur N mots de codes transmis avec au plus t erreurs.
3. Tester votre programme sur quelques matrices aléatoires H (penser à vérifier que H est une matrice de contrôle).
4. Ecrire un programme `Distance(H)` qui calcule la distance minimale du code C (importer le module `itertools` et utiliser la commande `itertools.combinations`). Inclure cette fonctionnalité dans `TestSyndrome` afin de vérifier que vos codes corrigent le bon nombre d'erreurs.

Exercice 5 (Code de répétition) 1. Ecrire un programme qui calcule la matrice de contrôle du code de répétition de dimension k (longueur des mots sources) et s répétitions.

2. Tester votre code à l'aide du programme `TestSyndrome(H,t,N)`, en considérant $k = 3$ et $s = 3, 5$. Vérifier que votre code corrige le bon nombre d'erreurs.

Exercice 6 (Codes BCH) 1. Ecrire un programme qui calcule une matrice de contrôle du code BCH de longueur $n = 2^m - 1$ et de distance prescrite δ.

2. Vérifier pour différentes valeurs de m que les codes BCH de distance prescrite $\delta = 3$ sont équivalents aux codes de Hamming $C(m)$.

3. Tester votre code à l'aide du programme `TestSyndrome` pour $m = 4$ et $\delta = 3, 4, 5, 6$. Vérifier que la distance minimale est supérieure ou égale à la distance prescrite. Y a t-il toujours égalité $d = \delta$ (cf exo 12 CM) ? Vérifier.

Exercice 7 (Codes cycliques) 1. Implémenter une fonction qui prend en entrées des entiers n et q premiers entre eux et une liste $L \subset [0, \ldots, n-1]$ et qui renvoie en sortie la plus petite liste ordonnée $\Sigma \subset [0, \ldots, n-1]$ telle que $\Sigma \mod n$ est stable par multiplication par q.

2. À l'aide de la fonction précédente, calculer le polynôme générateur d'un code cyclique binaires t-correcteur de longueur n (utiliser `primitive_element()` pour déterminer un générateur de $\mathbb{F}_{2^m}^\times$). On se référera à la Proposition 1 du CM.

3. Ecrire un programme qui calcule la matrice de contrôle d'un code cyclique de polynôme générateur g. Utiliser votre fonction `TestSyndrome` de l'exo 4 pour tester votre code cyclique t-correcteur. Vérifier qu'il corrige bien le bon nombre d'erreurs prédites.

4. Quel choix de n et de L permet de retrouver le code de Hamming $C(m)$? Les codes BCH de distance prescrite δ ? Vérifier sur la matrice de contrôle.

5. Vérifier que le polynôme générateur du code de Hamming $C(m)$ est un facteur irréductible du polynôme cyclotomique Φ_n où $n = 2^m - 1$. Plus généralement, vérifier que le polynôme générateur g_i associé à une classe cyclotomique $\Sigma_i \subset \mathbb{Z}/n\mathbb{Z}$ est un facteur irréductible d'un polynôme cyclotomique $\Phi_d \in \mathbb{F}_2[X]$ où d est un diviseur de n que l'on déterminera en fonction de i.

Exercice 8 (Correction des codes BCH binaires par polynôme localisateur d'erreurs) 1. Ecrire une procédure de correction des codes BCH binaires de longueur $2^m - 1$ et distance prescrite δ via la méthode du polynôme localisateur d'erreurs. En entrée : les entiers m, δ et le mot reçu $r \in \mathbb{F}_2[X]$. En sortie, le mot de code corrigé $c \in \mathbb{F}_2[X]$

2. Tester votre procédure sur des transmissions avec t erreurs de mots de code aléatoires $c \in C \subset \mathbb{F}_2[X]$. Vérifier que le code est bien t-correcteurs dès lors que $t \leq (\delta - 1)/2$, et que l'on peut même autoriser $t = \delta/2$ lorsque δ est paire (cf exo 12 CM).