Codes correcteurs d'erreurs

pour l'UE HAI906I "Calcul formel avancé et applications"

polycopié - version du 01/10/2026

Rocco Mora

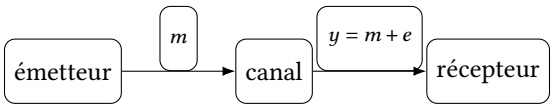
Table des matières

Chapitre 1

Introduction

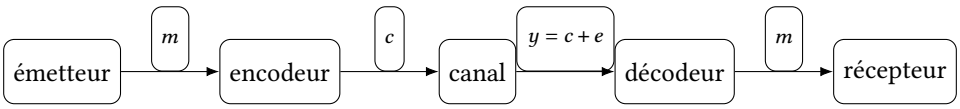
Le 20 août 1977, la sonde Voyager 2 quittait la Terre en emportant, parmi d'autres instruments, une caméra dont les images allaient devoir parcourir des milliards de kilomètres à travers le bruit interplanétaire avant d'atteindre une antenne radio au sol. Lorsque le signal arrivait, il était noyé dans un bruit thermique bien en deçà du niveau qu'une oreille ou un œil humain pouvait décoder directement, et pourtant les images de Jupiter, Saturne, Uranus et Neptune qui nous sont parvenues sont nettes. L'écart entre un canal bruité et un message fiable est comblé par les *codes correcteurs d'erreurs* : un filet de sécurité algébrique qui permet à un récepteur de reconstituer exactement ce qui a été envoyé, même si ce qui arrive réellement est corrompu.

Un exemple jouet. Avant tout formalisme, il est utile de voir l'idée centrale sur le plus petit exemple possible. Chaque fois que nous communiquons numériquement, nous le faisons à travers un *canal bruité* : ce que reçoit le récepteur n'est pas tout à fait ce qu'a envoyé l'émetteur.



Disons que l'émetteur souhaite transmettre le message binaire $m = (1, 0, 1)$, et que le canal inverse un bit, si bien que le récepteur reçoit $y = (0, 0, 1)$. En écrivant $y = m + e$, le *vecteur d'erreur* est ici $e = (1, 0, 0)$ — mais le récepteur ne voit jamais que y, jamais m ni e séparément, et n'a aucun moyen de savoir que quelque chose s'est mal passé : $(0, 0, 1)$ semble tout aussi plausible comme message que ne l'était $(1, 0, 1)$.

La solution de la théorie des codes n'est pas de modifier le canal — nous ne le pouvons pas — mais de modifier ce que nous envoyons. Nous *encodons* m au préalable, en ajoutant délibérément de la redondance, de sorte que les séquences transmises possibles (les *mots de code*) se trouvent loin les unes des autres.



Par exemple, au lieu d'envoyer directement $m = (1, 0, 1)$, répétons-le trois fois : $c = (1, 0, 1,\ 1, 0, 1,\ 1, 0, 1)$. Si la même erreur d'un seul bit se produit, le récepteur reçoit alors $y = (1, 0, 0,\ 1, 0, 1,\ 1, 0, 1)$ — et comme les huit mots de code possibles (un pour chaque choix de $m \in \{0, 1\}^3$) sont largement espacés les uns des autres, il existe exactement *un seul* mot de code à un bit inversé de y, à savoir c lui-même : l'erreur n'est pas seulement détectée mais *corrigée*, et m est retrouvé exactement, sans aucune ambiguïté.

Cet exemple jouet contient déjà tous les ingrédients auxquels le reste de ce cours consacrera de nombreux chapitres pour les préciser et les rendre efficaces : une notion de *distance* entre séquences, pour que « éloignés » et « mot de code le plus proche » aient un sens précis (Section 2.1) ; un *encodeur* transformant des messages courts en mots de code plus longs et bien séparés, et ce de façon systématique plutôt que par répétition ad hoc (Chapitre 3) ; et un *décodeur* qui, étant donné y, doit trouver efficacement le mot de

code le plus proche — trivial par force brute pour des messages de 3 bits, totalement infaisable par force brute dès que m compte un millier de bits (Chapitre 5) — le tout soumis à des compromis difficiles entre la quantité de redondance que nous pouvons nous permettre d'ajouter et le nombre d'erreurs que nous pouvons nous permettre de corriger (Chapitre 4).

La même idée, sous des habits différents, se retrouve dans chaque disque compact, chaque paquet Wi-Fi, chaque QR code sur une tasse de café, chaque baie RAID et chaque contrôleur SSD, et — plus surprenant encore — dans les schémas cryptographiques conçus pour résister aux ordinateurs quantiques, ainsi que dans les protocoles de partage de secret qui permettent d'ouvrir le coffre d'une banque seulement lorsque trois directeurs sur cinq sont d'accord.

Deux racines historiques. Le sujet trouve véritablement son origine dans le même bâtiment, aux Bell Telephone Laboratories, vers 1948. **Claude Shannon**, dans son article *A Mathematical Theory of Communication* (1948), montre, de façon purement existentielle et probabiliste, qu'un canal possède une *capacité* en deçà de laquelle une communication arbitrairement fiable est possible, et au-delà de laquelle elle ne l'est pas — mais l'argument de Shannon n'exhibe aucun code utilisable. **Richard Hamming**, qui travaillait dans le couloir voisin sur des calculateurs à relais s'arrêtant régulièrement à cause d'erreurs de parité détectées par des lecteurs de cartes perforées, posa la question complémentaire et constructive : étant donné qu'une erreur a été *détectée*, peut-elle aussi être *localisée et corrigée* ? Son article de 1950, *Error Detecting and Error Correcting Codes*, répond par l'affirmative et donne la première famille non triviale de codes, encore utilisée aujourd'hui. Un troisième nom couramment associé à cette période initiale est celui de **Marcel Golay**, qui publia en 1949 (dans un article d'à peine une page) deux codes exceptionnels qui demeurent, soixante-quinze ans plus tard, parmi les objets combinatoires les plus remarquables connus.

Point de vue de ce cours. La théorie de Shannon porte sur la capacité *probabiliste* ; la théorie que nous développons ici est *combinatoire* : au lieu des capacités de canal et des probabilités d'erreur, notre objet central est la **distance de Hamming** entre deux chaînes, et notre question centrale est de savoir combien de mots de code, deux à deux éloignés selon cette distance, peuvent être placés dans un espace fini. C'est ce point de vue combinatoire qui rend possibles des constructions explicites et des algorithmes de décodage explicites.

Jouons au jeu du théoricien des codes. Avant d'étudier rigoureusement la théorie des codes correcteurs d'erreurs, ses problèmes fondamentaux et les solutions trouvées au fil des décennies, il peut être intéressant d'essayer de construire nous-mêmes quelques codes. Par exemple :

1. Construire un code capable de toujours détecter une erreur.
2. Construire un code capable de toujours corriger une erreur.
3. Construire un code capable de toujours détecter une erreur en n'ajoutant qu'un seul bit de redondance.
4. Construire un code capable de toujours corriger une erreur en n'ajoutant qu'un seul bit de redondance.

Certaines de ces questions sont-elles plus faciles que d'autres ? Pensez-vous qu'elles sont toutes réalisables, ou que certaines sont théoriquement impossibles ?

Chapitre 2

Codes (en blocs) : notions de base

§ 2.1 Alphabets, codes, distance, taux

Definition 2.1. Soit A un ensemble fini (un *alphabet*) avec $|A| = q$, et soit $n \in \mathbb{N}^*$. Un *code de longueur n sur A* est simplement un sous-ensemble $C \subseteq A^n$. Ses éléments sont des *mots de code*. On note $M = |C|$ sa taille.

Aucune structure algébrique sur A ou C n'est supposée à ce stade — c'est délibéré : certains des meilleurs codes connus pour de petits paramètres ne sont pas linéaires, et il est important de voir le cadre général avant de se spécialiser.

Definition 2.2. Le *poids de Hamming* de $x \in A^n$ est

$$\mathrm{wt}(x) := |\{i \in \{1, \dots, n\} \mid x_i \neq 0\}|.$$

Definition 2.3. La *distance de Hamming* entre $x, y \in A^n$ est

$$\mathrm{d}(x, y) := |\{i \in \{1, \dots, n\} \mid x_i \neq y_i\}|.$$

On peut voir de façon équivalente la distance de Hamming entre $x, y \in A^n$ comme $\mathrm{wt}(x - y)$.

Proposition 2.4. d *est une métrique sur A^n, c'est-à-dire que*

1. *elle est positive,* $\mathrm{d}(x, y) \geq 0$ *et* $\mathrm{d}(x, y) = 0$ *si et seulement si* $x = y$,
2. *elle est symétrique,* $\mathrm{d}(x, y) = \mathrm{d}(y, x)$,
3. *et elle satisfait l'inégalité triangulaire* $\mathrm{d}(x, z) \leq \mathrm{d}(x, y) + \mathrm{d}(y, z)$.

Exercise 2.5. Démontrer la Proposition 2.4.

Remark 2.6 (D'autres métriques existent). La distance de Hamming est la métrique que ce cours utilise presque exclusivement, mais ce n'est pas la seule qui présente un intérêt, et il vaut mieux le savoir dès le départ. Lorsque l'alphabet lui-même porte une structure cyclique, en « cadran d'horloge » — par exemple $A = \mathbb{Z}_q = \{0, 1, \dots, q-1\}$, par opposition à un simple ensemble fini non structuré — la *distance de Lee*

$$\mathrm{d}_L(x, y) := \sum_{i=1}^{n} \min\left(|x_i - y_i|,\ q - |x_i - y_i|\right)$$

est souvent celle qui a un sens physique : elle mesure l'écart de chaque coordonnée comme le plus court

chemin autour d'un cycle de longueur q, plutôt que comme un simple indicateur de correspondance/non-correspondance, ce qui modélise les erreurs de type « phase » (par exemple dans certains schémas de modulation numérique) bien plus fidèlement que ne le fait la distance de Hamming. Voir l'Exercice 2.7 ci-dessous pour une preuve que d_L est bien une métrique, et pour sa relation avec d.

Un autre exemple est donné par la métrique de rang : dans ce cas, les mots de code ne sont pas des vecteurs mais des matrices, le poids est calculé comme le rang de la matrice et la distance comme le rang de la différence.

Exercise 2.7 (La métrique de Lee)**.** Soit $A = \mathbb{Z}_q$ et rappelons la distance de Lee $\mathrm{d}_L(x,y) := \sum_{i=1}^{n} \min(|x_i - y_i|, q - |x_i - y_i|)$ de la Remarque 2.6.

(a) Montrer que d_L est une métrique sur $\mathbb{Z}_q^n$: positivité, symétrie, $\mathrm{d}_L(x,y) = 0 \iff x = y$, et l'inégalité triangulaire. (Pour l'inégalité triangulaire, commencer par prouver le cas coordonnée par coordonnée, $n = 1 : u \mapsto \min(u, q-u)$ pour $u \in \{0, \dots, q-1\}$ vérifie $\min(a+b, q-a-b \bmod q) \le \min(a, q-a) + \min(b, q-b)$ pour tous a, b ; puis sommer sur les coordonnées.)

(b) Montrer que $\mathrm{d}_L(x,y) \ge \mathrm{d}(x,y)$ toujours, avec égalité lorsque $q \in \{2,3\}$, et donner un exemple explicite avec $q \ge 4$ où l'inégalité est stricte.

Definition 2.8. La *distance minimale* d'un code C ayant au moins deux mots de code est

$$d(C) := \min_{\substack{x,y \in C \\ x \neq y}} \mathrm{d}(x,y).$$

On note les paramètres de C par $(n, M, d)_q$, ou simplement (n, M, d) lorsque q est clair. Le *taux* (d'information) de C est $R := \log_q(M)/n \in [0,1]$, et sa *distance relative* est $\delta := d/n \in [0,1]$.

Il existe une tension entre R et δ : intuitivement, écarter M mots de code de sorte qu'ils soient deux à deux très éloignés (grand δ) laisse moins de « place » pour en placer beaucoup (grand R). Cette tension est le mieux capturée par une unique quantité combinatoire, qu'il vaut la peine d'isoler dès maintenant puisqu'elle organisera pour ainsi dire tout ce qui suit.

Exercise 2.9 (L'inclusion contrôle la distance)**.** Soient $C_1 \subseteq C_2 \subseteq A^n$ deux codes, chacun ayant au moins deux mots de code.

(a) Montrer que $d(C_1) \ge d(C_2)$. (Indication : les paires de mots de code distincts parcourues dans le minimum définissant $d(C_1)$ forment un sous-ensemble de celles parcourues dans le minimum définissant $d(C_2)$; que produit le fait de prendre un minimum sur un ensemble de candidats plus petit ?)

(b) Donner un exemple explicite, avec $C_1 \subsetneq C_2 \subseteq \mathbb{F}_2^4$ de tailles toutes deux ≥ 2, où l'inégalité est *stricte* ($d(C_1) > d(C_2)$), ainsi qu'un autre exemple où il y a égalité.

Definition 2.10 (Le problème combinatoire central)**.** Pour des entiers $n, d \ge 1$ et $q \ge 2$, on définit

$$A_q(n,d) := \max\{|C| \mid C \subseteq A^n,\ |A| = q,\ d(C) \ge d\},$$

la plus grande taille possible pour *n'importe quel* code (linéaire ou non) de longueur n sur un alphabet de taille q garantissant une distance minimale au moins d. Un code $C \subseteq A^n$ avec $d(C) \ge d$ atteignant $|C| = A_q(n,d)$ est dit $A_q(n,d)$*-optimal* (ou simplement *optimal* pour ces paramètres, lorsque q, n, d sont clairs d'après le contexte).

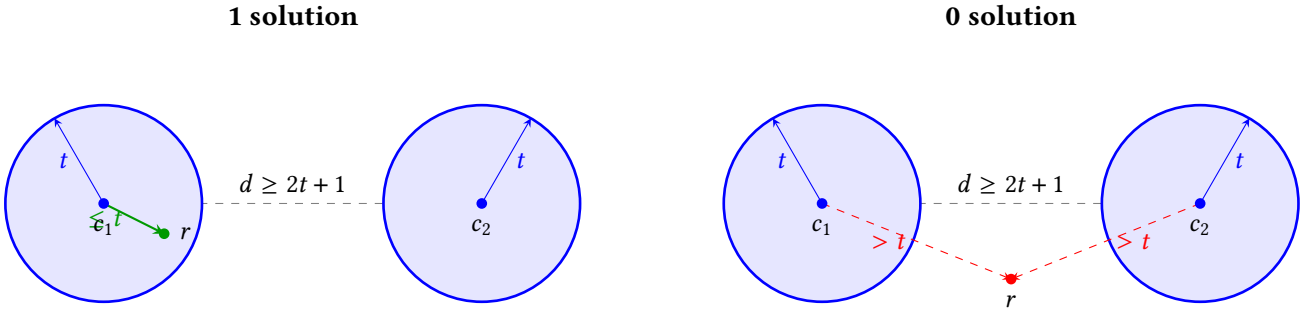
Déterminer $A_q(n,d)$ exactement, pour tous n, d, q, est sans doute *le* problème combinatoire central de

tout ce cours : c'est précisément la question « à quel point un code peut-il être bon ? », dépouillée de toute considération hormis la longueur, la taille de l'alphabet et la distance minimale garantie. Le Chapitre 4 développe plusieurs bornes générales sur $A_q(n,d)$ (Singleton, Hamming, Plotkin, Gilbert–Varshamov, Griesmer), chacune capturant une obstruction combinatoire ou un argument d'existence différent ; pour ainsi dire chaque famille spécifique de codes construite plus loin dans ce cours est, au fond, une tentative de cerner exactement $A_q(n,d)$ pour une certaine famille infinie de paramètres — ou de s'en approcher utilement avec une construction qui se trouve aussi être efficacement *encodable et décodable*, ce dont $A_q(n,d)$ seul ne dit rien. Même pour $q = 2$ et des n assez modestes, la valeur exacte de $A_2(n,d)$ reste inconnue pour une infinité de paires (n,d) : après soixante-quinze ans, il s'agit toujours d'un domaine ouvert de la combinatoire énumérative, dont les records actuels sont répertoriés en ligne (`https://www.codetables.de/`).

Exercise 2.11. Soit C un code $A_2(n,d)$-optimal (Définition 2.10) avec d impair. Montrer qu'ajouter un bit de parité global à chaque mot de code (*extension*) transforme C en un code de longueur $n+1$ et de distance minimale $d+1$, de même taille. En déduire que $A_2(n+1,d+1) \geq A_2(n,d)$ pour tout d impair.

Proposition 2.12 (Distance et correction d'erreurs). *Si $d(C) \geq 2t+1$, alors pour tout $c \in C$ et tout $y \in A^n$ avec $\mathrm{d}(y,c) \leq t$, le mot de code c est l'*unique *élément de C à distance au plus t de y. Par conséquent, un code de distance minimale d peut corriger tout motif de $t = \lfloor (d-1)/2 \rfloor$ erreurs, et peut* détecter *(sans nécessairement les corriger) tout motif d'au plus $d-1$ erreurs.*

Démonstration. Si $c' \in C$, $c' \neq c$, vérifiait aussi $\mathrm{d}(y,c') \leq t$, l'inégalité triangulaire donnerait $\mathrm{d}(c,c') \leq 2t < d(C)$, ce qui contredit la définition de $d(C)$. ■



Definition 2.13. La *boule* de rayon t autour de $x \in A^n$ est $B(x,t) := \{z \in A^n \mid \mathrm{d}(x,z) \leq t\}$, la *sphère* de rayon t autour de $x \in A^n$ est $S(x,t) := \{z \in A^n \mid \mathrm{d}(x,z) = t\}$.

La Proposition 2.12 dit exactement que les boules $B(c, \lfloor (d-1)/2 \rfloor)$, $c \in C$, sont deux à deux disjointes — une image d'*empilement de sphères* qui réapparaîtra plus loin sous le nom de borne de Hamming.

Proposition 2.14 (Cardinaux des boules et des sphères). *Pour tout $x \in A^n$ et $0 \leq t \leq n$, $|S(x,t)| = \binom{n}{t}(q-1)^t$ et $|B(x,t)| = \sum_{i=0}^{t} \binom{n}{i}(q-1)^i$.*

Exercise 2.15. Démontrer la Proposition 2.14.

Exercise 2.16. Soit $A = \{0,1\}$ ($q = 2$) et considérons le code $C = \{(1,1,0,1,0), (0,0,0,1,1), (0,1,1,0,1),\} \subseteq A^5$.

(a) Calculer les trois poids de Hamming et les trois distances de Hamming deux à deux entre les trois mots de code.

(b) En déduire $d(C)$, et indiquer le plus grand t pour lequel C est garanti de détecter/corriger t erreurs quelconques.

(c) $y = (0, 1, 1, 1, 0)$ est-il à distance t d'un unique mot de code ? Si oui, lequel ?

Exercise 2.17. Soit $A = \{0, 1, 2\}$ ($q = 3$) et considérons le code $C = \{(0, 0, 1, 2), (1, 2, 0, 0), (2, 1, 2, 1), (0, 1, 1, 0)\} \subseteq A^4$.

(a) Calculer les quatre poids de Hamming et les six distances de Hamming deux à deux entre les quatre mots de code.

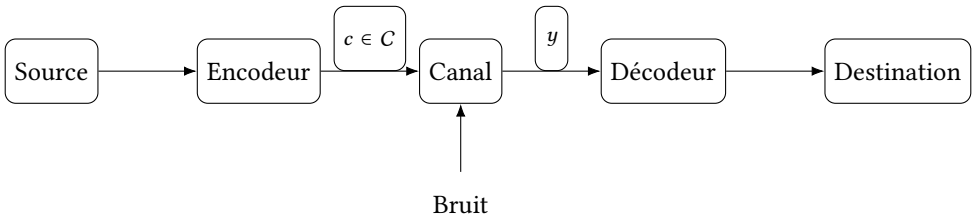
(b) En déduire $d(C)$, et indiquer le plus grand t pour lequel C est garanti de détecter/corriger t erreurs quelconques.

(c) $y = (0, 1, 2, 0)$ est-il à distance t d'un unique mot de code ? Si oui, lequel ?

Exercise 2.18. Sur $A = \{0, 1\}$, $n = 6$, calculer $|B(x, 2)|$ (le nombre de chaînes binaires de longueur 6 à distance de Hamming au plus 2 d'une chaîne fixée x). Recommencer pour $n = 10$, $q = 4$, rayon $t = 3$, en utilisant la formule générale pour la taille des boules. Comparer chaque réponse à q^n et commenter la rapidité avec laquelle les boules remplissent l'espace ambiant lorsque t croît.

§ 2.2 Canaux de communication

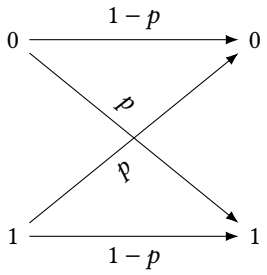
D'où viennent physiquement l'espace ambiant A^n et toute l'idée de corriger jusqu'à t erreurs ? Cette section précise, et généralise, l'image jouet du Chapitre 1 : là, un message m devenait un mot de code c, le canal ajoutait une erreur inconnue e, et le récepteur ne voyait que $y = c + e$. Nous fixons maintenant les modèles de bruit spécifiques auxquels ce cours se réfère chaque fois que nous demandons combien d'erreurs — ou combien de symboles *effacés* — un décodeur doit être construit pour tolérer.



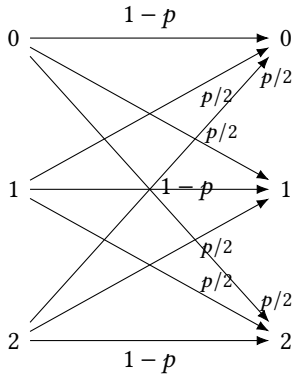
Definition 2.19. Un *canal discret* d'alphabet d'entrée A et d'alphabet de sortie B est spécifié par des probabilités de transition $P(y \mid x)$, $x \in A$, $y \in B$, avec $\sum_{y \in B} P(y \mid x) = 1$ pour tout x. Le canal est dit *sans mémoire* si, sur une séquence d'entrée $x = (x_1, \ldots, x_n) \in A^n$, chaque symbole de sortie est produit indépendamment : $\Pr[y \mid x] = \prod_{i=1}^{n} P(y_i \mid x_i)$.

Tout canal utilisé dans ce cours est un canal discret sans mémoire (CDSM, ou DMC en anglais). Les canaux réels sont rarement *exactement* sans mémoire, mais le modèle sans mémoire est la bonne première approximation, et c'est celui par rapport auquel le théorème de capacité de Shannon et la plupart des constructions de ce cours sont énoncés.

Definition 2.20 (Canal binaire symétrique)**.** Le *canal binaire symétrique* $\mathrm{BSC}(p)$, $0 \le p \le \frac{1}{2}$, a $A = B = \{0, 1\}$ et inverse le bit transmis indépendamment avec probabilité p : $P(1 \mid 0) = P(0 \mid 1) = p$, $P(0 \mid 0) = P(1 \mid 1) = 1 - p$.



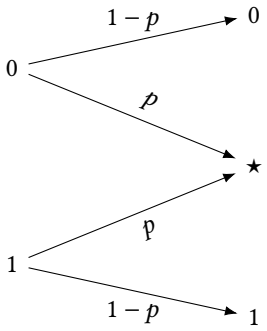
Definition 2.21 (Canal symétrique q-aire)**.** Le *canal symétrique q-aire* $\mathrm{QSC}_q(p)$ a $A = B = \{0, \ldots, q-1\}$, transmet chaque symbole correctement avec probabilité $1-p$, et sinon le remplace par *chacun* des $q-1$ autres symboles avec probabilité égale $p/(q-1)$: $P(x \mid x) = 1-p$, $P(y \mid x) = p/(q-1)$ pour $y \neq x$.



(Représenté pour $q = 3$, où chaque flèche hors diagonale porte la probabilité $p/(q-1) = p/2$; en général, chaque entrée possède $q-1$ flèches hors diagonale, chacune de probabilité $p/(q-1)$.)

Les erreurs ne sont pas le seul type de corruption qu'il vaille la peine de modéliser séparément : parfois le récepteur sait exactement *quels* symboles ont été perdus (un secteur de disque qui n'a pas pu être lu, un paquet réseau qui n'est jamais arrivé) sans savoir ce qu'ils disaient — une situation qualitativement plus facile à gérer qu'une erreur en position inconnue, et suffisamment fréquente pour mériter son propre modèle de canal.

Definition 2.22 (Canal binaire à effacement)**.** Le *canal binaire à effacement* $\mathrm{BEC}(p)$ a pour alphabet d'entrée $\{0, 1\}$ et pour alphabet de sortie $\{0, 1, \star\}$, où $\star$ (l'*effacement*) signifie « perdu, position connue » : chaque bit est transmis correctement avec probabilité $1-p$ et effacé avec probabilité p — il n'est *jamais* inversé vers la mauvaise valeur.



Definition 2.23 (Canal q-aire à effacement)**.** Le *canal q-aire à effacement* généralise ceci à l'alphabet d'entrée $\{0, \ldots, q-1\}$ et à l'alphabet de sortie $\{0, \ldots, q-1, \star\}$: chaque symbole est transmis

correctement avec probabilité $1-p$ et effacé avec probabilité p.

Exercise 2.24. Représenter graphiquement le canal q-aire à effacement.

Remark 2.25. Un code MDS est exactement le bon outil contre un canal à effacement : un code $[n, k, n-k+1]_q$ permet de récupérer le message entier à partir de *n'importe quels* k des n symboles transmis ayant survécu sans effacement (c'est la Proposition **??** du Chapitre 9, bien que le fait sous-jacent — que toute famille de k coordonnées d'un code MDS forme un ensemble d'information — soit disponible dès que le Chapitre 4 introduit les codes MDS). C'est précisément pourquoi le stockage distribué à codage par effacement, sujet central du Chapitre 9, modélise une panne de disque ou de nœud comme un effacement, et non comme une erreur de symbole : le système sait toujours *quel* nœud est tombé en panne.

Enfin, nous précisons la règle de décodage que ce cours a déjà utilisée de façon informelle, et que nous étudierons algorithmiquement dans toute sa généralité au Chapitre 5.

Definition 2.26 (Décodage au plus proche voisin / au maximum de vraisemblance)**.** Étant donné un mot reçu $y \in B^n$, le *décodage au plus proche voisin* produit

$$\hat{c} := \arg\min_{c \in C} \mathrm{d}(y, c)$$

(en départageant arbitrairement les ex æquo), c'est-à-dire un mot de code le plus proche de y pour la distance de Hamming. Pour un $\mathrm{BSC}(p)$ ou un $\mathrm{QSC}_q(p)$ avec $p < \frac{q-1}{q}$ (un symbole a, individuellement, plus de chances de survivre que d'être corrompu), le décodage au plus proche voisin coïncide exactement avec le décodage au *maximum de vraisemblance* — le décodeur maximisant $\Pr[y \mid c]$ sur $c \in C$ — car sous ce canal, un nombre plus faible de changements de symboles est toujours l'explication la plus probable de ce qui a été reçu.

§ 2.3 Un code à chiffre de contrôle du monde réel : le numéro de sécurité sociale français

Tous les schémas de contrôle d'erreur utilisés au quotidien ne sont pas des constructions algébriques sophistiquées ; certains des plus courants ne sont qu'une simple vérification modulaire, et il vaut la peine de les voir explicitement avant que la machinerie des chapitres suivants ne fasse paraître tout plus compliqué qu'il ne l'est réellement.

Le *numéro d'inscription au répertoire* (NIR, couramment appelé « numéro de sécurité sociale ») est un nombre N à 13 chiffres codant le sexe, l'année et le mois de naissance, un code de département (et, pour les personnes nées à l'étranger, un code pays), la commune de naissance, et un numéro d'ordre. Pour se prémunir contre les erreurs de transcription, une *clé de contrôle* $K \in \{01, \ldots, 97\}$ à 2 chiffres est ajoutée, calculée comme

$$K := 97 - (N \bmod 97)$$

(avec la convention qu'un code de département « 2A » ou « 2B », pour les personnes nées en Corse, est remplacé respectivement par « 19 » ou « 18 » avant de calculer $N \bmod 97$, et que $K = 97$ plutôt que $K = 0$ lorsque $97 \mid N$). La séquence complète à 15 chiffres (N, K) est ce qui apparaît effectivement sur une carte de sécurité sociale et une fiche de paie françaises.

Example 2.27.

$$\underbrace{2\ 69\ 05\ 49\ 588\ 157}_{\text{NIR}}\quad\underbrace{80}_{\text{clé de contrôle}}$$

— 2 ⟶ sexe féminin (1 pour les hommes)

- 69 → année de naissance (1969)
- 05 → mois de naissance
- 49 → département de naissance
- 588 → code de la commune
- 157 → 157-ème personne née cette année-là et ce mois-là dans cette commune (si > 999, un peu plus compliqué)
- 80 → clé de contrôle

Algébriquement, il s'agit d'une unique *équation de contrôle de parité sur* $\mathbb{Z}_{97}$: en notant $\overline{N} := N \bmod 97$, les paires valides sont exactement celles vérifiant $\overline{N} + K \equiv 0 \pmod{97}$ — l'analogue direct, sur le module 97 au lieu de $\mathbb{F}_2$, du code de contrôle de parité binaire de la section suivante. Le module 97 n'est pas arbitraire : c'est le plus grand nombre premier inférieur à 100 (de sorte que K tienne encore sur deux chiffres), et c'est précisément la primalité qui rend les garanties de détection d'erreur du schéma (voir les exercices) démontrables plutôt qu'accidentelles.

Exercise 2.28. Vérifier la clé de contrôle avec votre propre numéro de sécurité sociale français.

Exercise 2.29 (Le NIR détecte toute erreur portant sur un seul chiffre)**.** Supposons que le nombre véritable soit N avec la clé de contrôle $K = 97 - (N \bmod 97)$, et qu'exactement un chiffre de N soit mal transcrit (à une position $i \in \{0, \dots, 12\}$, en comptant à partir du chiffre des unités, le chiffre passant de d à $d' \neq d$), donnant un nombre corrompu $N' \neq N$, tandis que K lui-même est transmis correctement.

(a) Montrer que $N' - N = (d' - d) \cdot 10^i$ pour un certain entier $d' - d$ avec $1 \leq |d' - d| \leq 9$.

(b) En utilisant le fait que 97 est premier et ne divise pas 10 (donc $10^i \not\equiv 0 \pmod{97}$), montrer que $N' - N \not\equiv 0 \pmod{97}$.

(c) En déduire que recalculer la clé de contrôle à partir de N' donne $97 - (N' \bmod 97) \neq K$: une erreur portant sur un seul chiffre est *toujours* détectée.

Exercise 2.30 (Deux erreurs peuvent s'annuler)**.** Supposons maintenant que *deux* chiffres de N soient simultanément mal transcrits, aux positions $i \neq j$, changeant respectivement de $\Delta_i, \Delta_j \in \{-9, \dots, -1, 1, \dots, 9\}$ (donc $N' - N = \Delta_i 10^i + \Delta_j 10^j$).

(a) Exhiber un choix explicite de i, j, Δ_i, Δ_j pour lequel $N' - N \equiv 0 \pmod{97}$ — c'est-à-dire que l'erreur passe complètement inaperçue. (Indication : fixer Δ_j et chercher parmi les 9 valeurs possibles de Δ_i celle, s'il en existe une, qui pour les i, j choisis rend la somme $\equiv 0 \pmod{97}$; ajuster i, j si la première tentative n'a pas de solution dans la plage autorisée.)

(b) Supposons les positions $i \neq j$ fixées et Δ_i, Δ_j indépendants et uniformes sur les 9 différences de chiffres non nulles $\{-9, \dots, -1, 1, \dots, 9\}$. Montrer que pour chacune des 9 valeurs de Δ_j, *au plus une* valeur de Δ_i parmi les 9 possibles rend l'erreur indétectable (utiliser que deux éléments distincts de $\{-9, \dots, -1, 1, \dots, 9\}$ diffèrent de moins de 97, donc ne peuvent être congrus modulo 97 que s'ils sont égaux). En déduire que la probabilité qu'une double erreur soit indétectable est au plus $9/81 = 1/9$.

§ 2.4 Premier aperçu de bons et de mauvais codes

Example 2.31 (Répétition et parité)**.** Sur $A = \{0, 1\}$, le *code de répétition* de longueur n, $\{0^n, 1^n\}$, a pour paramètres $(n, 2, n)_2$: excellente distance, taux exécrable. Le *code de contrôle de parité* $\{u \in \mathbb{F}_2^n \mid u_1 + \cdots + u_n = 0\}$ a pour paramètres $(n, 2^{n-1}, 2)_2$: excellent taux, distance minimale — il ne fait que *détecter* une seule erreur, il ne peut pas la corriger. La majeure partie de ce cours consiste à trouver des compromis raisonnables entre ces deux extrêmes.

Example 2.32 (Un compromis parfait : le code de Hamming)**.** Pour $r \geq 2$, posons $n = 2^r - 1$. Il existe un code binaire de longueur n, à 2^{n-r} mots de code, et de distance minimale 3 (construit au Chapitre 3) : il corrige une seule erreur tout en ne sacrifiant que $r = \log_2(n+1)$ symboles de redondance — un taux tendant vers 1 lorsque n croît.

§ 2.5 Équivalence de codes

Renommer l'alphabet coordonnée par coordonnée, ou permuter les n positions, produit un code de paramètres identiques et de comportement de correction d'erreurs identique ; il est donc naturel d'identifier de tels codes.

Definition 2.33. Deux codes $C, C' \subseteq A^n$ sont *équivalents* si $C' = \{(\sigma_1(c_{\pi(1)}), \ldots, \sigma_n(c_{\pi(n)})) \mid c \in C\}$ pour une certaine permutation $\pi \in S_n$ des coordonnées et certaines permutations $\sigma_1, \ldots, \sigma_n$ de l'alphabet appliquées coordonnée par coordonnée.

Des codes équivalents ont la même taille et la même distribution des distances. La réciproque échoue de façon spectaculaire — des codes inéquivalents peuvent partager tous les mêmes paramètres (n, M, d) — un fait à garder à l'esprit chaque fois qu'un tableau (comme celui de la Section 3.5) ne liste que des paramètres et non les codes eux-mêmes.

Exercise 2.34. Démontrer que l'équivalence de codes (Définition 2.33) est une relation d'équivalence, et que des codes équivalents ont la même taille et la même distribution des distances $(A_0, A_1, \ldots, A_n)$, où $A_i := |\{(x, y) \in C^2 : \mathrm{d}(x, y) = i\}|/|C|$. Trouver (ou retrouver dans la littérature) un exemple de deux codes *inéquivalents* partageant les mêmes paramètres (n, M, d).

Chapitre 3

Codes linéaires

Désormais, nous nous spécialisons aux alphabets qui sont des corps finis, $A = \mathbb{F}_q$, et à C un $\mathbb{F}_q$-sous-espace de $\mathbb{F}_q^n$: c'est le cadre dans lequel deux astuces calculatoires portent leurs fruits : une base (plutôt qu'une liste complète) décrit le code, et une application linéaire (plutôt qu'une table de correspondance) décrit l'encodeur.

§ 3.1 Matrices génératrice et de contrôle de parité, dualité

Definition 3.1. Un *code linéaire* C de longueur n et de dimension k sur $\mathbb{F}_q$ est un $\mathbb{F}_q$-sous-espace de $\mathbb{F}_q^n$ de dimension k ; on dit que C est un code $[n,k]_q$, ou un code $[n,k,d]_q$ si de plus sa distance minimale est d.

Proposition 3.2. *Si C est un code linéaire de dimension k sur $\mathbb{F}_q$, alors il possède q^k éléments.*

Démonstration. Une base du code est donnée par k vecteurs. Toute combinaison linéaire de ces vecteurs peut être obtenue en sommant ces vecteurs multipliés chacun par un élément de $\mathbb{F}_q$. Il y a donc au total $\left|\mathbb{F}_q\right|^k = q^k$ choix possibles. ■

Proposition 3.3. *Si C est un code linéaire, alors*

$$d(C) = \min_{0 \neq c \in C} \mathrm{wt}(c).$$

Démonstration. Puisque $\mathrm{d}(x,y) = \mathrm{wt}(x-y)$, le résultat découle immédiatement du fait que C est un sous-groupe de $(\mathbb{F}_q^n, +)$. ■

La proposition précédente implique que le calcul de la distance minimale pour un code linéaire ramène les $\binom{|C|}{2}$ calculs de distances deux à deux nécessaires pour un code générique à « seulement » $|C|-1$ calculs de poids.

Remark 3.4 (Combien de bits un code coûte-t-il réellement à stocker ?). Il vaut la peine de rendre complètement quantitative l'économie « base plutôt que liste complète » du paragraphe d'ouverture du chapitre. Un code générique $C \subseteq \mathbb{F}_q^n$ de taille M n'a, dans le pire des cas, aucun raccourci du tout : écrire tous les mots de code coûte $Mn \log_2 q$ bits. Un code linéaire $[n,k]_q$, en revanche, possède $M = q^k$ mots de code mais est entièrement déterminé par une matrice génératrice $k \times n$, ne coûtant que

$$kn \log_2 q \text{ bits} \qquad \text{au lieu de} \qquad q^k\, n \log_2 q \text{ bits}.$$

Pour $k = 100$, $n = 200$, $q = 2$: la matrice génératrice coûte $100 \times 200 = 20\,000$ bits (environ 2,5 Ko) ; la liste naïve nécessiterait $2^{100} \times 200$ bits — bien plus que le nombre d'atomes dans l'univers observable, sans même

parler de ce qui serait stockable. Cet écart exponentiel contre polynomial, et non une simple préférence esthétique pour l'algèbre, est la véritable raison pour laquelle pratiquement tous les codes utilisés en pratique sont linéaires ou très proches de l'être : un code non linéaire $A_q(n,d)$-optimal (Définition 2.10) peut occasionnellement battre le meilleur code linéaire d'un petit facteur constant en *taille* (Section 3.5), mais il n'offre en général aucune description aussi compacte, ni aucun algorithme de décodage efficace.

Exercise 3.5. Pour chaque code ci-dessous, déterminer s'il est linéaire (c'est-à-dire un $\mathbb{F}_q$-sous-espace de $\mathbb{F}_q^n$) ou non, en justifiant votre réponse soit en exhibant une base (si linéaire), soit en exhibant un échec concret de la clôture, tel que deux mots de code dont la somme n'appartient pas au code (sinon). Rappelons que le fait que $|C|$ soit une puissance de q est *nécessaire* pour la linéarité mais, comme le montrent certains des exemples ci-dessous, loin d'être *suffisant*.

(a) Les deux codes *triviaux* $C = \{0\} \subseteq \mathbb{F}_q^n$ et $C = \mathbb{F}_q^n$, pour q, n arbitraires.

(b) $C = \{000, 100, 010, 111\} \subseteq \mathbb{F}_2^3$.

(c) $C = \{000, 110, 101, 011\} \subseteq \mathbb{F}_2^3$.

(d) $C = \{(0,0), (1,1), (2,0), (0,2), (1,0)\} \subseteq \mathbb{F}_3^2$.

(e) $C = \{(a, b, a+2b, 3a-b) \mid a, b \in \mathbb{F}_5\} \subseteq \mathbb{F}_5^4$.

(f) $C = \{(a, b, ab) \mid a, b \in \mathbb{F}_3\} \subseteq \mathbb{F}_3^3$.

(g) $C = \{(a, b, a^2+b) \mid a, b \in \mathbb{F}_3\} \subseteq \mathbb{F}_3^3$.

Definition 3.6. Une *matrice génératrice* $G \in \mathbb{F}_q^{k\times n}$ de C est toute matrice dont les lignes forment une base de C, de sorte que $C = \{uG : u \in \mathbb{F}_q^k\}$. Une *matrice de contrôle de parité* $H \in \mathbb{F}_q^{(n-k)\times n}$ de C est une matrice de rang plein telle que $C = \{c \in \mathbb{F}_q^n : Hc^T = 0\}$.

Exercise 3.7. La matrice

$$G = \begin{pmatrix} 1 & 2 & 2 & 4 \\ 3 & 2 & 1 & 0 \\ 2 & 1 & 4 & 4 \end{pmatrix}$$

est-elle une matrice génératrice d'un code linéaire sur $\mathbb{F}_5$? Et sur $\mathbb{F}_7$?

Definition 3.8 (Forme systématique)**.** Une matrice génératrice G est sous *forme systématique* si $G = [\, I_k \mid A \,]$ pour un certain $A \in \mathbb{F}_q^{k\times(n-k)}$, c'est-à-dire que les k premières colonnes forment la matrice identité. Dans ce cas, encoder un message $u \in \mathbb{F}_q^k$ en $c = uG = (u \mid uA)$ reproduit littéralement u inchangé dans les k premières coordonnées (les *symboles d'information*) et ajoute $n-k$ *symboles de parité* redondants uA.

Example 3.9. Le code
$C = \{(0,0,0), (1,2,2), (2,1,1), (0,1,2), (0,2,1), (1,0,1), (2,0,2), (1,2,0), (2,1,0)\} \subseteq \mathbb{F}_3^3$ est linéaire de dimension 2. La matrice

$$G = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 0 & 2 \end{pmatrix}$$

est une matrice génératrice de C et

$$G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

en est la matrice génératrice sous forme systématique.

Proposition 3.10 (Tout code possède une matrice génératrice systématique, après réordonnancement des coordonnées)**.** *Tout code $[n,k]_q$ C possède* une *matrice génératrice sous forme systématique, éventuellement après permutation des n coordonnées (c'est-à-dire que C est toujours équivalent, via une permutation de coordonnées, à un code possédant une matrice génératrice systématique).*

Démonstration. Partons d'une matrice génératrice quelconque G_0. Puisque G_0 est de rang k, certaines k de ses colonnes sont linéairement indépendantes ; permutons les coordonnées pour que ce soient les k premières colonnes, formant une sous-matrice inversible M de taille $k \times k$. En multipliant à gauche par M^{-1}, on obtient $G := M^{-1}G_0 = [\, I_k \mid A \,]$, qui est toujours une matrice génératrice du code (à coordonnées permutées), puisque la multiplication à gauche par une matrice inversible ne fait que changer le choix de la base. ■

Proposition 3.11 (De la matrice génératrice à la matrice de contrôle de parité, et retour)**.** *Si $G = [\, I_k \mid A \,]$ est une matrice génératrice systématique de C, alors*

$$H := [\, -A^T \mid I_{n-k} \,]$$

est une matrice de contrôle de parité de C. Réciproquement, si $H = [\, B \mid I_{n-k} \,]$ est une matrice de contrôle de parité systématique quelconque, alors $G := [\, I_k \mid -B^T \,]$ est une matrice génératrice systématique de C.

Démonstration. H est de rang $n-k$ (elle contient I_{n-k}), donc elle est bien une matrice de contrôle de parité dès lors que l'on vérifie $GH^T = 0$:

$$GH^T = [\, I_k \mid A \,] \begin{pmatrix} -A \\ I_{n-k} \end{pmatrix} = -A + A = 0.$$

Par la définition du code dual, H est alors automatiquement une matrice génératrice de $C^\perp$, et elle est visiblement sous forme systématique (bloc identité à droite). La réciproque est le même calcul lu à l'envers, en utilisant $(C^\perp)^\perp = C$. ■

Example 3.12. Pour $C \subseteq \mathbb{F}_5^5$ de matrice génératrice systématique $G = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} = [\, I_3 \mid A \,]$, la Proposition 3.11 donne immédiatement $H = [\, -A^T \mid I_2 \,] = \begin{pmatrix} -1 & -2 & -1 & 1 & 0 \\ -2 & -1 & -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 3 & 4 & 1 & 0 \\ 3 & 4 & 4 & 0 & 1 \end{pmatrix}$ (en réduisant modulo 5) — sans aucune réduction d'une nouvelle matrice à effectuer, H se lit directement, et est simultanément une matrice génératrice pour $C^\perp$, un code $[5,2]_5$.

Exercise 3.13. Donner la matrice génératrice sous forme systématique du code $C = \langle (3,4,0,2), (6,6,1,5) \rangle \subseteq \mathbb{F}_7^4$ ainsi qu'une matrice de contrôle de parité.

Exercise 3.14. Soit $C \subseteq \mathbb{F}_3^5$ le code linéaire de matrice génératrice

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

(a) Écrire une matrice de contrôle de parité H pour C.

(b) À l'aide de H, déterminer $d(C)$ (Proposition 3.18).

(c) Donner les paramètres $[n, k, d]_3$ de C et de son dual $C^\perp$.

Definition 3.15. Le *code dual* de C est $C^\perp := \{x \in \mathbb{F}_q^n : \langle x, c\rangle = 0 \ \forall c \in C\}$, où $\langle x, y\rangle = \sum x_i y_i$. Si G, H sont des matrices génératrice et de contrôle de parité pour C, alors H, G sont des matrices de contrôle de parité et génératrice pour $C^\perp$, et $(C^\perp)^\perp = C$. En particulier $\dim C^\perp = n - \dim C$.

Puisque $C^\perp$ est le sous-espace linéaire orthogonal à C, on a

$$\dim C^\perp = n - \dim C.$$

Remark 3.16. Lorsqu'on travaille avec la métrique euclidienne usuelle sur les réels, on sait que l'intersection de sous-espaces orthogonaux est triviale, c'est-à-dire qu'elle ne contient que le vecteur nul. Ici au contraire, il est possible d'avoir des codes qui s'intersectent de façon non triviale avec leur dual. Une telle intersection est appelée le *hull* d'un code linéaire. L'exercice suivant demande de les calculer.

Exercise 3.17. Hull Soit $n = 4$. Pour chaque $0 \le k \le n$, construire un code C de longueur n tel que $\dim C \cup C^\perp = k$, si possible. Faire de même pour $n = 5$.

Proposition 3.18 (Poids et matrice de contrôle de parité)**.** *C a une distance minimale $\ge d$ si et seulement si toute famille de $d-1$ colonnes d'une matrice de contrôle de parité H est linéairement indépendante.*

Démonstration. Un mot de code non nul c de poids w vérifie $Hc^T = 0$, c'est-à-dire qu'une combinaison $\mathbb{Z}_q$-linéaire, avec pour coefficients les entrées non nulles de c, des w colonnes de H indexées par $\mathrm{Supp}(c)$ s'annule : les mots de code de poids w correspondent donc exactement aux familles *dépendantes* de w colonnes de H. ■

§ 3.2 Somme et intersection de codes

Deux autres opérations élémentaires combinent une *paire* existante de codes de même longueur pour en former un nouveau. Elles interviennent constamment en coulisses : la concaténation parallèle au cœur des turbocodes (Chapitre 11) est construite à partir de deux codes partageant les mêmes bits d'information, et les identités de dualité ci-dessous sont précisément l'outil qui permet de passer entre les descriptions par matrice génératrice et par matrice de contrôle de parité de tels codes combinés.

Definition 3.19. Soient $C_1, C_2 \subseteq \mathbb{F}_q^n$ deux codes de même longueur n. Leur *somme* est

$$C_1 + C_2 := \{c_1 + c_2 : c_1 \in C_1,\ c_2 \in C_2\} \subseteq \mathbb{F}_q^n,$$

et leur *intersection* $C_1 \cap C_2$ est l'intersection ensembliste ordinaire. Les deux sont encore des codes linéaires de longueur n (des sous-espaces de $\mathbb{F}_q^n$).

Proposition 3.20 (Dimension)**.** $\dim(C_1 + C_2) + \dim(C_1 \cap C_2) = \dim(C_1) + \dim(C_2)$.

Démonstration. C'est la formule des dimensions de Grassmann pour les sous-espaces d'un espace vectoriel, appliquée à $C_1, C_2 \le \mathbb{F}_q^n$: on étend une base de $C_1 \cap C_2$ en une base de C_1, et séparément en une base de C_2 ; les deux extensions, ensemble avec la base de l'intersection, forment une base de $C_1 + C_2$ (une brève vérification de génération et d'indépendance linéaire, exactement comme en algèbre linéaire ordinaire), ce qui donne $\dim(C_1 + C_2) = \dim(C_1) + \dim(C_2) - \dim(C_1 \cap C_2)$. ■

Une matrice génératrice de $C_1 + C_2$ s'obtient en empilant les matrices génératrices de C_1 et C_2 et en éliminant les lignes dépendantes ; une matrice génératrice de $C_1 \cap C_2$ s'obtient, de façon duale, en empilant les matrices de contrôle de parité de C_1 et C_2 et en lisant une base du noyau conjoint. Cette dualité entre les deux constructions est exacte :

Proposition 3.21 (Dualité). *$(C_1 + C_2)^\perp = C_1^\perp \cap C_2^\perp$ et $(C_1 \cap C_2)^\perp = C_1^\perp + C_2^\perp$.*

Démonstration. Pour la première identité : $x \perp (C_1 + C_2)$ si et seulement si $\langle x, c_1 + c_2 \rangle = 0$ pour tous $c_1 \in C_1$, $c_2 \in C_2$; en prenant $c_2 = 0$ (resp. $c_1 = 0$), on voit que cela force $x \perp C_1$ et $x \perp C_2$ séparément, et réciproquement $x \perp C_1, C_2$ donne clairement $x \perp (C_1 + C_2)$ par linéarité ; donc $(C_1 + C_2)^\perp = C_1^\perp \cap C_2^\perp$. La seconde identité s'obtient en appliquant la première à $C_1^\perp, C_2^\perp$ à la place de C_1, C_2 et en utilisant $(C^\perp)^\perp = C$. ■

Proposition 3.22 (Distance). *$d(C_1 \cap C_2) \geq \max\big(d(C_1), d(C_2)\big)$ dès que $C_1 \cap C_2 \neq \{0\}$, et $d(C_1 + C_2) \leq \min\big(d(C_1), d(C_2)\big)$ dès que $C_1 + C_2$ possède au moins deux mots de code.*

Démonstration. Tout mot de code non nul de $C_1 \cap C_2$ est, en particulier, un mot de code non nul de C_1 et de C_2 séparément, donc de poids $\geq d(C_1)$ et $\geq d(C_2)$, ce qui donne la première inégalité. Pour la seconde, $C_1 \subseteq C_1 + C_2$ et $C_2 \subseteq C_1 + C_2$, donc l'Exercice 2.9(a) (l'inclusion contrôle la distance) s'applique deux fois : $d(C_1 + C_2) \leq d(C_1)$ et $d(C_1 + C_2) \leq d(C_2)$. ■

Example 3.23. Soient $q = 2$, $n = 4$, $C_1 := \langle (1,1,0,0), (0,0,1,1) \rangle$ et $C_2 := \langle (1,0,1,0), (0,1,0,1) \rangle$, aucun n'étant contenu dans l'autre (par exemple $(1,1,0,0) \in C_1 \setminus C_2$). En listant les mots de code, $C_1 = \{0000, 1100, 0011, 1111\}$ et $C_2 = \{0000, 1010, 0101, 1111\}$, donc $C_1 \cap C_2 = \{0000, 1111\}$, le code de répétition $[4,1,4]_2$, et l'on calcule que $C_1 + C_2$ est le code de contrôle de parité $[4,3,2]_2$. La formule de Grassmann se vérifie : $\dim(C_1 + C_2) + \dim(C_1 \cap C_2) = 3 + 1 = 4 = 2 + 2 = \dim(C_1) + \dim(C_2)$. Conformément à la Proposition 3.22, $d(C_1 \cap C_2) = 4 \geq \max(2,2)$ (strictement plus grand ici) et $d(C_1 + C_2) = 2 \leq \min(2,2)$ (égalité ici).

Exercise 3.24. Soient $q = 2$, $n = 5$, $C_1 := \langle (1,1,0,0,0), (0,0,1,1,0) \rangle$ et $C_2 := \langle (1,1,0,0,0), (0,0,0,1,1) \rangle$.

(a) Lister les mots de code de C_1, C_2, et calculer $C_1 \cap C_2$ directement à partir des deux listes.

(b) Donner une matrice génératrice pour $C_1 + C_2$ (empiler et réduire), et vérifier numériquement la Proposition 3.20 sur cet exemple.

(c) Calculer $d(C_1)$, $d(C_2)$, $d(C_1 \cap C_2)$ et $d(C_1 + C_2)$, et vérifier les deux inégalités de la Proposition 3.22.

§ 3.3 Poinçonnage et raccourcissement : fabriquer de nouveaux codes

Deux opérations élémentaires, duales l'une de l'autre, permettent d'échanger de la longueur contre de la dimension ou de la distance sans repartir d'une construction entièrement nouvelle ; elles sont utilisées constamment, à la fois pour construire des codes pratiques de longueur « irrégulière » à partir d'un beau code algébrique, et à l'intérieur de plusieurs des preuves de bornes du chapitre suivant.

Definition 3.25 (Poinçonnage). Fixons une position de coordonnée $i \in \{1, \dots, n\}$. Le *code poinçonné* C^{*i} est obtenu en supprimant la i-ème coordonnée de chaque mot de code de C :

$$C^{*i} := \{(c_1, \dots, c_{i-1}, c_{i+1}, \dots, c_n) : c \in C\} \subseteq \mathbb{F}_q^{n-1}.$$

Proposition 3.26. *Soit C un code $[n,k,d]_q$ avec $d \geq 2$. Alors C^{*i} est un code $[n-1,k,d']_q$ avec $d' \in \{d-1,d\}$; plus précisément, $d' = d-1$ si C contient un mot de code de poids minimal non nul en position i, et $d' = d$ sinon. (Poinçonner en plusieurs positions à la fois se définit de la même manière, en les supprimant toutes simultanément.)*

Démonstration. La suppression est une surjection linéaire $C \to C^{*i}$; elle est injective sauf si deux mots de code de C coïncident partout sauf éventuellement en position i, ce qui, puisque $d \geq 2$, ne peut se produire pour deux mots de code *distincts* ne différant qu'à cette position sans différer aussi ailleurs — une brève vérification montre que le noyau est trivial lorsque $d \geq 2$, donc $\dim C^{*i} = k$. Chaque mot de code perd au plus une coordonnée non nulle lors de la suppression, d'où $d' \geq d-1$; et $d' \geq d$ n'est impossible que si un mot de code de poids minimal est non nul exactement en i, auquel cas $d' = d-1$ est atteint par ce mot et ne peut pas descendre davantage d'après la phrase précédente. ■

Definition 3.27 (Raccourcissement)**.** Fixons une position de coordonnée i. Le *code raccourci* C_i est

$$C_i := \{(c_1,\dots,c_{i-1},c_{i+1},\dots,c_n) : c \in C,\ c_i = 0\} \subseteq \mathbb{F}_q^{n-1}.$$

Proposition 3.28. *Soit C un code $[n,k,d]_q$. Alors C_i est un code $[n-1,k',d']_q$ avec $k' \in \{k-1,k\}$ (typiquement $k-1$, et exactement k seulement dans le cas dégénéré où tout mot de code s'annule déjà en position i) et $d' \geq d$.*

Proposition 3.29 (Poinçonnage et raccourcissement sont duaux l'un de l'autre)**.** *Pour tout i, $(C^{*i})^\perp = (C^\perp)_i$ et $(C_i)^\perp = (C^\perp)^{*i}$.*

Démonstration. Un vecteur $x \in \mathbb{F}_q^{n-1}$ (vu avec un 0 réinséré en position i) est orthogonal à tout mot de C^{*i} exactement lorsque son extension par zéro est orthogonale à tout mot de C ayant pu le produire par suppression, c'est-à-dire à tout mot de C — mais tout $y \in C^\perp$ qui est nul en i réalise déjà cela, ce qui donne $(C^\perp)_i \subseteq (C^{*i})^\perp$, et un décompte de dimensions (utilisant les Propositions 3.26–3.28) montre l'égalité. ■

Example 3.30. Poinçonner le code de répétition $[n,1,n]_2$ en une position donne le code de répétition $[n-1,1,n-1]_2$ (le taux augmente très légèrement, la distance chute exactement de 1). Raccourcir le code de contrôle de parité $[n,n-1,2]_2$ en une position donne le code de contrôle de parité $[n-1,n-2,2]_2$: cohérent avec la Proposition 3.29, puisque parité et répétition sont duaux l'un de l'autre.

Exercise 3.31. Soit C le code $[6,3]_2$ de matrice génératrice $G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$.

(a) Poinçonner C en position 5 : donner une matrice génératrice et les paramètres de C^{*5}.

(b) Raccourcir C en position 5 : donner une matrice génératrice et les paramètres de C_5.

(c) Vérifier directement, sur cet exemple, la relation de dualité $(C^{*5})^\perp = (C^\perp)_5$ de la Proposition 3.29.

La construction suivante peut être vue comme l'inverse du raccourcissement/poinçonnage, puisqu'elle insère une coordonnée supplémentaire et redondante.

Definition 3.32 (Extension)**.** Étant donné C un code $[n,k,d]_q$, son *code étendu* est $\widehat{C} := \{(c, c_{n+1}) : c \in C,\ \sum_i c_i + c_{n+1} = 0\}$, un code $[n+1,k,d^+]_q$ avec $d^+ \in \{d, d+1\}$.

Il existe bien d'autres constructions qui, à partir d'un ou plusieurs codes linéaires, en fournissent un nouveau ; nous mentionnons simplement le produit de codes et la somme de Plotkin.

§ 3.4 Une galerie de familles classiques

Example 3.33 (Codes de Hamming et simplexe)**.** Pour $r \geq 2$, $n = 2^r - 1$, soit $H_r \in \mathbb{F}_2^{r\times n}$ ayant pour colonnes *tous* les vecteurs non nuls de $\mathbb{F}_2^r$, chacun exactement une fois. Le code de matrice de contrôle de parité H_r est le *code de Hamming* $\mathcal{H}_r$, un code $[n,\ n-r,\ 3]_2$: aucune colonne de H_r n'est nulle et deux colonnes ne sont jamais égales, donc par la Proposition 3.18, $d \geq 3$; et trois colonnes quelconques dont la somme s'annule (qui doivent exister puisque $r < n$) montrent que $d = 3$. Son dual, le *code simplexe* $\mathcal{S}_r = \mathcal{H}_r^\perp$, est un code à poids constant : tout mot de code non nul a un poids exactement égal à 2^{r-1}, ce que l'on déduit directement du fait qu'une forme linéaire non nulle sur $\mathbb{F}_2^r$ s'annule sur exactement la moitié des vecteurs non nuls.

Exercise 3.34. Montrer que raccourcir le code de Hamming binaire $\mathcal{H}_r$ en une seule coordonnée donne toujours un code linéaire qui corrige encore une erreur, mais qui n'est en général plus lui-même un code de Hamming (sa longueur n'est plus de la forme $2^{r'}-1$). Calculer les paramètres exacts $[n',k',d']_2$ de ce code raccourci en fonction de r.

Exercise 3.35. Démontrer que tout mot de code non nul du code simplexe $\mathcal{S}_r$ a un poids exactement égal à 2^{r-1} (compléter l'argument en une ligne esquissé dans le texte : une forme linéaire non nulle sur $\mathbb{F}_2^r$ s'annule sur exactement la moitié des 2^r-1 vecteurs non nuls). En déduire l'énumérateur des poids complet de $\mathcal{S}_r$.

Example 3.36 (Codes de Reed–Muller)**.** En identifiant les points de $\mathbb{F}_2^m$ aux 2^m affectations de vérité possibles, le code $\mathrm{RM}(r,m)$ d'ordre r est constitué des vecteurs d'évaluation de tous les polynômes booléens de degré $\leq r$ sur $\mathbb{F}_2^m$. C'est un code $\left[2^m,\ \sum_{i=0}^{r}\binom{m}{i},\ 2^{m-r}\right]_2$, et $\mathrm{RM}(r,m)^\perp = \mathrm{RM}(m-r-1,m)$. Le code d'ordre 1, $\mathrm{RM}(1,m)$, a été utilisé, dans une instance de longueur 32, sur la mission Mariner 9 de la NASA — un exemple précoce et concret de la théorie des codes quittant le tableau noir.

Exercise 3.37. Montrer que $\mathrm{RM}(1,m)^\perp = \mathrm{RM}(m-2,m)$ est cohérent avec la formule de dualité générale $\mathrm{RM}(r,m)^\perp = \mathrm{RM}(m-r-1,m)$, et vérifier le décompte des dimensions $\dim \mathrm{RM}(r,m) + \dim \mathrm{RM}(m-r-1,m) = 2^m$ directement à partir de la formule $\dim \mathrm{RM}(r,m) = \sum_{i=0}^{r}\binom{m}{i}$, en utilisant l'identité binomiale $\sum_{i=0}^{m}\binom{m}{i} = 2^m$.

Example 3.38 (Les codes de Golay)**.** Il existe un code binaire $[24,12,8]_2$ essentiellement unique, auto-dual, appelé le *code de Golay binaire étendu* $\mathcal{G}_{24}$; le poinçonner une fois donne le *code de Golay* $\mathcal{G}_{23}$, un code $[23,12,7]_2$ qui est *parfait* (égalité dans la borne de Hamming du chapitre suivant). Un analogue ternaire, $\mathcal{G}_{12}$, est un code parfait $[11,6,5]_3$. Les deux ont été découverts par Golay en 1949 et possèdent tous deux des groupes d'automorphismes exceptionnellement grands (les groupes simples sporadiques de Mathieu M_{24} et M_{12}), un fait exploité bien au-delà de la théorie des

codes, en théorie des groupes et dans les problèmes de réseaux/d'empilement de sphères (le réseau de Leech est construit à partir de $\mathcal{G}_{24}$).

Les codes de Reed–Solomon, la famille la plus utilisée en pratique, sont suffisamment importants pour mériter leur propre chapitre (Chapitre 6) ; les codes cycliques, le cadre algébrique unificateur sous lequel les codes de Hamming, de Reed–Muller (sous une certaine forme) et de Reed–Solomon peuvent tous être placés, font l'objet du chapitre situé juste après celui sur les bornes et le décodage.

§ 3.5 Les codes non linéaires peuvent tout de même l'emporter

Le Chapitre 2 a introduit $A_q(n, d)$ — la plus grande taille possible pour *n'importe quel* code, linéaire ou non, de longueur donnée et de distance minimale garantie (Définition 2.10) — comme la quantité combinatoire centrale de tout le sujet. Maintenant que les codes linéaires et leurs paramètres $[n, k, d]_q$ sont disponibles pour comparaison, il vaut la peine de le dire clairement : pour certains paramètres, aucun code linéaire n'est optimal, alors même que ce chapitre vient de consacrer plusieurs sections à bâtir une riche boîte à outils de constructions linéaires.

Example 3.39 (Le code de Best)**.** $A_2(8, 3) = 20$, réalisé par un code binaire non linéaire découvert par M. Best (aussi appelé le code de Julin). En revanche, tout code *linéaire* $[8, k, 3]_2$ vérifie $k \leq 4$ (une conséquence rapide des bornes de Hamming et de Griesmer du Chapitre 4), c'est-à-dire au plus 16 mots de code — strictement moins que 20. Donc pour ces paramètres, aucun code linéaire n'est optimal, même si 16 est déjà honorablement proche de 20.

Example 3.40 (Le code de Nordstrom–Robinson)**.** Il existe un code binaire non linéaire de longueur 16, avec 256 mots de code et de distance minimale 6, découvert indépendamment par Nordstrom et Robinson (1967). Aucun code linéaire $[16, k, 6]_2$ n'a $2^k \geq 256 = 2^8$ mots de code : le meilleur code *linéaire* avec ces paramètres a pour dimension $k = 7$, soit seulement 128 mots de code — le code non linéaire est exactement *deux fois* plus grand.

Remark 3.41 (Où trouver le record actuel). Les deux exemples ci-dessus sont des phénomènes exceptionnels à petits paramètres plutôt qu'une règle générale : asymptotiquement (taux fixé, $n \to \infty$), les meilleures constructions connues munies d'encodeurs et de décodeurs explicites et efficaces sont linéaires ou proches de l'être. Les codes non linéaires importent surtout pour les petits paramètres tabulés, où chaque mot de code supplémentaire compte. En conséquence, maintenant que les codes linéaires sont sur la table, il vaut la peine de savoir où consulter les meilleurs paramètres *actuellement connus* plutôt que de se fier à une liste figée dans un manuel : les tables de Markus Grassl sur `http://www.codetables.de` constituent la référence standard, activement maintenue, pour les meilleurs codes linéaires connus $[n, k, d]_q$ pour tout $q \leq 9$, incluant des constructions explicites, et sont continuellement mises à jour à mesure que de nouveaux codes sont découverts. (Elles ne suivent que les codes *linéaires* — il n'existe pas de table publique aussi complète pour la quantité pleinement générale, pas nécessairement linéaire, $A_q(n, d)$ de la Définition 2.10, précisément parce qu'abandonner l'exigence de linéarité rend l'espace de recherche, et donc le suivi de ce qui est actuellement connu, beaucoup plus vaste.)