Algorithmes Arithmétiques II – Devoir à la maison

Julien Lavauzelle

Transmis le 25/10/2024
à rendre pour le **05/01/2025**

Consignes.

1. Vous devez rendre, par un email adressé à `julien.lavauzelle@univ-paris8.fr` et jusqu'au **dimanche 05 janvier 2025** dernier délai, une archive au format `.zip` contenant les documents ci-dessous :
   (a) un fichier édité sur ordinateur (pas de photos, l'utilisation de LaTeX est fortement conseillée), au format `.pdf`, contenant vos réponses aux questions de nature théorique ;
   (b) vos fichiers de programmation pour les questions d'implantation.
2. Pour les questions de **programmation**, vous pouvez utiliser le langage/logiciel de votre choix. Les logiciels de calcul formel comme `sage` ou `magma` sont conseillés, mais toutes les questions peuvent être réalisées en `python` grâce aux classes de polynômes et de matrices utilisées en cours. Voir la page web du cours si besoin.
3. Ce que vous me rendez doit refléter un **travail personnel**. L'utilisation de ressources externes (humaines, ou informatiques telles que ChatGPT) doit être aussi limitée que possible. En particulier, vous devez comprendre et savoir expliquer toutes les réponses et toutes les lignes de code que vous aurez écrites. En cas de doute, je m'autorise à vous convoquer pour vérifier l'authenticité de votre travail.
4. **Évaluation.** Le sujet est assez long et contient quelques questions avancées. Il n'est pas attendu de traiter la totalité les questions pour obtenir la note maximale. En revanche, une attention particulière sera portée à la **qualité et au soin des raisonnements et des implémantations**. Il est préférable de traiter excellement la moitié des questions que moyennement toutes les questions.

Pour se documenter, on pourra notamment se référer aux ressources suivantes :

[Sud97] Madhu Sudan. Decoding of Reed Solomon Codes beyond the Error-Correction Bound. *J. Complex.*, 13(1) :180–193, 1997

[GRS23] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential Coding Theory. `https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/`, 2023

[Aug10] Daniel Augot. Les codes algébriques principaux et leur décodage. In Jean-Guillaume Dumas, Grégoire Lecerf, Delphine Boucher, and Thomas Cluzeau, editors, *Journées Nationales de Calcul Formel*, volume 1 of *Les cours du CIRM*, pages 31–74, Luminy, France, May 2010. CIRM. `https://inria.hal.science/inria-00543322v1/file/Cours.pdf`

Une introduction au décodage en liste

1 Présentation du sujet

Le sujet a pour objectif de découvrir une **notion avancée de décodage d'erreurs**, dans le contexte où le nombre d'erreurs dépasse de rayon de décodage unique. En particulier, en fin de sujet nous étudierons l'**algorithme de décodage en liste de Sudan** pour les codes de Reed–Solomon.

Dans tout le sujet, on s'intéresse à des codes correcteurs linéaires, de longueur n, définis un le corps fini $\mathbb{F}_q$. Pour faciliter l'implémentation, on pourra se restreindre à un corps premier.

On rappelle que la distance minimale d'un code $\mathcal{C} \subseteq \mathbb{F}_q^n$ est donnée par

$$d_{\min}(\mathcal{C}) \coloneqq \min\{\mathrm{wt}(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathcal{C} \setminus \mathbf{0}\}$$

où $\mathrm{wt}(\boldsymbol{c}) \coloneqq |\{i \in \{1, \dots, n\} \mid c_i \neq 0\}$ est le poids de Hamming d'un mot $\boldsymbol{c}$.

On sait alors que, si un mot $\boldsymbol{y} \in \mathbb{F}_q^n$ se trouve à distance $\leq t \coloneqq \lfloor \frac{d_{\min}(\mathcal{C})-1}{2} \rfloor$ d'un mot de code $\boldsymbol{c} \in \mathcal{C}$, alors $\boldsymbol{c}$ est l'unique élément de $\mathcal{C}$ vérifiant cette propriété. Autrement dit, on peut (théoriquement) décoder de manière univoque n'importe quelle erreur de poids $\leq t$. Le paramètre t est appelé le **rayon de décodage unique** du code $\mathcal{C}$.

Par ailleurs, pour certaines familles de codes, des **algorithmes de décodage** permettent de retrouver efficacement [1] le mot de code $\boldsymbol{c}$ à partir de $\boldsymbol{y}$. C'est par exemple le cas de l'algorithme de Berlekamp–Welch pour les codes de Reed–Solomon.

Mais que se passe-t-il si le poids de l'erreur dépasse t ? Peut-on espérer retrouver $\boldsymbol{c}$? Dans quel cas ? Avec quelle complexité algorithmique ? Nous essaierons de répondre à ces questions tout au long de ce sujet, en introduisant puis en étudiant la notion de **décodage en liste**.

Remarque. Par *algorithme de décodage*, on entend : un algorithme qui prend en entrée (1) un mot $\boldsymbol{y} \in \mathbb{F}_q^n$ et (2) les paramètres d'un code correcteur $\mathcal{C}$, et qui retourne un des (ou plusieurs) mots du code $\mathcal{C}$ les plus proches (ou assez proches) de $\boldsymbol{y}$.

2 Aspects théoriques du décodage en liste

2.1 Premiers exemples

Pour bien saisir les enjeux du décodage en liste, commençons par étudier quelques exemples élémentaires.

Exemple 2.1

On considère le code de répétition de longueur $n = 4$ sur l'alphabet $\mathbb{F}_2$. Ce code admet exactement deux mots : le mot nul $\mathbf{0} = (0,0,0,0)$ et le mot $\boldsymbol{c} = (1,1,1,1)$.

La distance minimale du code est donc 4, et son rayon de décodage unique $\lfloor \frac{4-1}{2} \rfloor = 1$. Ainsi, tous les mots à distance ≤ 1 d'un mot de code peuvent aisément être décodés, de manière unique, au maximum de vraisemblance. Plus précisément :

- *tous les mots de $\mathbb{F}_2^4$ de poids 0 ou 1 peuvent être décodés en le mot nul ;*
- *tous les mots de $\mathbb{F}_2^4$ de poids 3 ou 4 peuvent être décodés en le mot $\boldsymbol{c}$.*

Mais qu'en est-il des mots de poids 2 de $\mathbb{F}_2^4$? Ils se trouvent à égale distance de $\mathbf{0}$ et de $\boldsymbol{c}$, ; on ne peut donc pas "choisir" vers quel mot de code les décoder : le décodage **unique** *est impossible.*

1. c'est-à-dire, avec une complexité polynomiale — en pratique, au pire cubique — en la longueur n du code

Exemple 2.2

Prenons un deuxième exemple : le code $\mathcal{C}$ engendré sur $\mathbb{F}_2$ par la matrice génératrice

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

On vérifie aisément que

$$\mathcal{C} = \{(0,0,0,0,0,0), (0,0,1,1,1,1), (1,1,0,0,0,0), (1,1,1,1,1,1)\},$$

que sa distance minimale est 2, et que son rayon de décodage unique est donc $\lfloor (2-1)/2 \rfloor = 0$. Ainsi, a priori *il existe des erreurs de poids 1 qui ne sont pas décodables de manière univoque.*

C'et par exemple le cas pour le mot $\boldsymbol{x} = (1,0,0,0,0,0)$, qui se trouve à égale distance du mot nul $(0,0,0,0,0,0)$ et du mot $(1,1,0,0,0,0)$. À chaque fois, la distance est égale à 1. En revanche, $\boldsymbol{x}$ est à distance 5 des mots de code $(0,0,1,1,1,1)$ et $(1,1,1,1,1,1)$. Il n'y a donc pas de sens à décoder $\boldsymbol{x}$ en ces deux mots.

D'un autre côté, le mot $\boldsymbol{y} = (0,0,1,0,0,0)$ se trouve à distance 1 du mot nul, à distance 3 des mots $(0,0,1,1,1,1)$ et $(1,1,0,0,0,0)$, et à distance 5 du mot $(1,1,1,1,1,1)$. Il est donc raisonnable de décoder $\boldsymbol{y}$ en le mot nul $(0,0,0,0,0,0)$.

Ce dernier exemple nous permet de faire deux remarques.

1. Pour certains mots $\boldsymbol{x}$ de l'espace ambiant $\mathbb{F}_q^n$, le nombre de mots de code les plus proches de $\boldsymbol{x}$ est strictement supérieur à 1 et strictement inférieur au nombre de mots de code. On pourrait donc dire que le mot $\boldsymbol{x}$ devrait être décodé en **l'un des mots les plus proches** (mais il n'y a pas unicité).
2. Pour d'autres mots $\boldsymbol{y} \in \mathbb{F}_q^n$, le nombre de mots de code les plus proches de $\boldsymbol{y}$ est égal à 1, **même si** la distance entre ces mots est **supérieure au rayon de décodage unique**. Il est alors raisonnable de décoder $\boldsymbol{y}$ en l'**unique mot le plus proche**.

Un troisième exemple est proposé en guise d'exercice.

Question 1.– Soit $\mathcal{A}$ le code défini sur $\mathbb{F}_2$ par la matrice génératrice

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

1. Donner le rayon de décodage unique t du code $\mathcal{A}$.
2. Trouver un mot $\boldsymbol{x}$ de l'espace ambiant, qui admet au moins 3 mots les plus proches.

Dans la suite de cette Section 2, étant donné un code $\mathcal{C} \subseteq \mathbb{F}_q^n$ et un mot $\boldsymbol{y}$ de l'espace ambiant $\mathbb{F}_q^n$, nous nous intéresserons au nombre de mots de code dont la distance à $\boldsymbol{y}$ est bornée, c'est-à-dire à

$$|\mathcal{B}(\boldsymbol{y}, r) \cap \mathcal{C}|,$$

où $\mathcal{B}(\boldsymbol{y}, r) \coloneqq \{\boldsymbol{x} \in \mathbb{F}_q^n \mid \mathrm{wt}_H(\boldsymbol{y} - \boldsymbol{x}) \leq r\}$ et r est un entier fixé.

Plus précisément, dans des contextes parfois favorables, on essaiera de :

1. donner des bornes sur $|\mathcal{B}(\boldsymbol{y}, r) \cap \mathcal{C}|$,
2. estimer la probabilité que $|\mathcal{B}(\boldsymbol{y}, r) \cap \mathcal{C}| \leq 1$,
3. dans des situations favorables, trouver tous les élements de $\mathcal{B}(\boldsymbol{y}, r) \cap \mathcal{C}$.

2.2 Code décodable en liste

Dans cette partie, commençons par étendre la notion théorique de décodabilité (unique) en une notion plus générale.

Définition 2.3

Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code linéaire, $L \geq 1$ et $0 \leq r \leq n$ deux entiers. On dit que $\mathcal{C}$ est (r, L)-décodable en liste si, pour tout $\boldsymbol{y} \in \mathbb{F}_q^n$, on a

$$|\mathcal{B}(\boldsymbol{y}, r) \cap \mathcal{C}| \leq L\,.$$

L'entier r est appelé **rayon** *de décodage en liste, et l'entier L* **taille** *de liste.*

Ainsi, un code (r, L)–décodable en liste permet en théorie, à partir d'un mot de code ayant subi r erreurs, d'obtenir **une liste d'au plus L éléments** contenant le mot de code recherché.

Exemple 2.4

1. *Tout code de rayon de décodage unique t est $(t, 1)$–décodable en liste. Par ailleurs, si un code $\mathcal{C}$ est $(t', 1)$–décodable en liste, alors $d_{\min}(\mathcal{C}) \geq 2t' + 1$.*
2. *Le code de répétition de longueur 4 (Exemple 2.1) est $(2, 2)$–décodable en liste (mais ce n'est pas très intéressant, car il n'admet que 2 mots de code).*
3. *Le code de l'Exemple 2.2 est $(1, 2)$–décodable en liste.*

Question 2.– Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code (r, L)–décodable en liste.

1. Est-il également $(r, L-1)$–décodable en liste ?
2. Est-il également $(r, L+1)$–décodable en liste ?
3. Est-il également $(r-1, L)$–décodable en liste ?

Justifiez vos réponses.

Bien entendu, si L est trop grand (par exemple, s'il est exponentiel en n), cela n'a aucun intérêt pratique, car on ne pourra pas retrouver le mot de code initial dans une liste qui serait trop grande. On cherchera donc des contextes dans lesquels la grandeur L est polynomiale en n, et si possible $L \in \mathcal{O}(n)$ ou $L \in \mathcal{O}(1)$.

Question 3.– 💻 Soit $\mathcal{B}$ le code défini sur $\mathbb{F}_7$ par la matrice génératrice

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 1 & 4 & 2 & 2 & 4 & 1 \\ 0 & 1 & 1 & 6 & 1 & 6 & 6 \\ 0 & 1 & 2 & 4 & 4 & 2 & 1 \end{pmatrix}.$$

Pour information, ce code est le code de Reed–Solomon de points d'évaluation $(0, 1, 2, 3, 4, 5, 6)$ et de dimension 5 sur $\mathbb{F}_7$. Soit également $\boldsymbol{y} := (5, 1, 1, 3, 1, 0, 2) \in \mathbb{F}_7^7$.

Donner le nombre de mots de code à distance w de $\boldsymbol{y}$, pour toutes les valeurs de w comprises entre 0 et 7 inclus. Pour cela, il est vivement conseillé de ne pas faire les calculs à la main, mais plutôt d'implémenter une fonction.

Question 4.– 💻 Soit $\mathcal{A}$ le code binaire de longueur 15 engendré par la matrice

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

Ce code est en réalité un code BCH de distance minimale 7. Son rayon de décodage unique est donc égal à 3.

1. Trouver le plus petit L pour lequel le code $\mathcal{A}$ est $(4, L)$–décodable en liste.
2. Trouver le plus grand r pour lequel le code $\mathcal{A}$ est (r, L)–décodable en liste avec des tailles de liste L inférieures ou égales à la moitié du cardinal du code.

2.3 Rayon de Johnson

Le cas binaire. Le premier objectif de cette partie est de démontrer le résultat suivant.

Proposition 2.5

(Borne de Johnson binaire.) Soit $\mathcal{C} \subseteq \mathbb{F}_2^n$ un code binaire de distance minimale d. Le code $\mathcal{C}$ est $(\frac{1}{2}J(d), 2n)$–décodable en liste, pour

$$J(d) \coloneqq n - \sqrt{n(n-2d)}\,.$$

L'application $d \mapsto J(d)$ est appelée fonction de Johnson.

Pour la démonstration de la Proposition 2.5, nous aurons besoin d'un lemme intermédiaire de nature géométrique. On représente par $\langle \cdot, \cdot \rangle$ le produit scalaire usuel de $\mathbb{R}^n$.

Lemme 2.6

Soient $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ des vecteurs deux à deux distincts de $\mathbb{R}^n$. Supposons que pour tous $1 \leq i, j \leq n$, on a $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \leq 0$. Alors $m \leq 2n$.

Question 5.– Démontrer le Lemme 2.6.
Indication : la preuve peut se faire par récurrence sur n.

Démontrons maintenant la borne de Johnson binaire (Proposition 2.5). Pour cela, nous aurons besoin de considérer l'application

$$\begin{array}{rccc} V: & \mathbb{F}_2^n & \to & \mathbb{R}^n \\ & \boldsymbol{x} & \mapsto & ((-1)^{x_1}, \ldots, (-1)^{x_n}) \end{array}$$

qui envoie un vecteur de $\mathbb{F}_2^n$ dans $\mathbb{R}^n$. On note par ailleurs $\mathbf{1} \coloneqq (1, 1, \ldots, 1) \in \mathbb{R}^n$.

Question 6.– Soit $\mathcal{C} \subseteq \mathbb{F}_2^n$ un code binaire de distance minimale d. Fixons $\boldsymbol{y} \in \mathbb{F}_2^n$, un entier $w \in [0, n]$, et notons $\mathcal{A}_w \coloneqq \mathcal{B}(\boldsymbol{y}, w) \cap \mathcal{C}$.

1. Soit $\alpha > 0$. Démontrer que pour tous $\boldsymbol{c}, \boldsymbol{c}' \in \mathcal{A}_w$, on a
$$\langle V(\boldsymbol{c}) - \alpha\mathbf{1}, V(\boldsymbol{c}') - \alpha\mathbf{1} \rangle \leq (n - 2d) + 2\alpha(2w - n) + \alpha^2\,.$$
2. Démontrer que si $\alpha = \sqrt{n(n-2d)}$ et $w \leq \frac{1}{2}(n - \alpha)$, alors $\langle V(\boldsymbol{c}) - \alpha\mathbf{1}, V(\boldsymbol{c}') - \alpha\mathbf{1} \rangle \leq 0$ pour tous $\boldsymbol{c}, \boldsymbol{c}' \in \mathcal{A}_w$.
3. Conclure concernant la Proposition 2.5.

Le cas général. Pour un alphabet fini de taille quelconque, on peut démontrer le résultat suivant.

Theorème 2.7

(Borne de Johnson indépendante de l'alphabet) Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code de distance minimale d. Alors, le code $\mathcal{C}$ est $(J(d), qn)$–décodable en liste.

Démonstration : Voir Chapitre 7.2 de [GRS23]. ■

La taille de liste étant linéaire en n et en q, si q n'est pas trop grand, il est alors théoriquement possible de décoder (en liste) des codes binaires jusqu'à un taux d'erreur $j(\delta) := 1 - \sqrt{1-\delta}$, où $\delta = d/n$. Comparons cela au décodage unique, pour lequel on sait que le décodage réussit lorsque le poids relatif de l'erreur est inférieur à $\tau(\delta) = \frac{\delta}{2}$. La Figure 1 nous montre que le décodage en liste est d'autant plus avantageux que la distance minimale du code est importante.

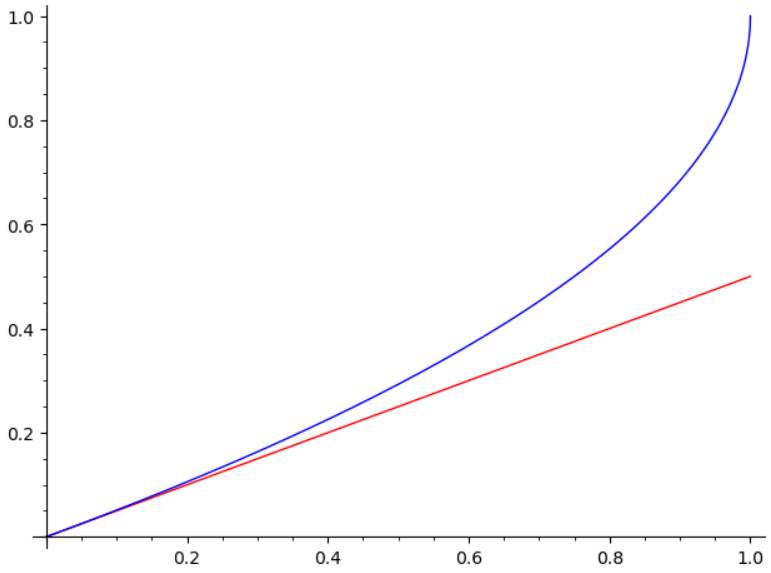

FIGURE 1 – Rayon de décodage unique (en rouge) et en liste (en bleu) en fonction de δ, la distance minimale du code.

Néanmoins, les notions développées dans cette partie ne restent pour l'instant que "théoriques" : même si les listes de mots de code sont de taille raisonnable, est-il possible de les retrouver efficacement ?

Dans la partie suivante, nous allons comprendre comment mettre en œuvre un **algorithme de décodage** en liste pour les codes de Reed–Solomon qui, pour certains paramètres, permet de retrouver le(s) mot(s) de code au-delà du rayon de décodage unique.

3 Décodage en liste des codes de Reed–Solomon

Dans cette partie, on présente un **algorithme** de décodage en liste pour les codes de Reed–Solomon, dû à Sudan en 1997 [Sud97].

Nous verrons que cet algorithme permet de retrouver une liste de taille $\mathcal{O}(n^2)$ (mais souvent de taille 1) comprenant tous les mots de code à distance $\leq w$ du mot reçu, et cela pour n'importe quelle valeur de w inférieure à $n - \sqrt{2n(k-1)}$.

Remarque : un lecteur attentif remarquera que w est très semblable à la borne de Johnson pour un code MDS, qui serait $n - \sqrt{n(k-1)}$. L'apparition du facteur "2" dans la racine carrée n'est pas une faute de frappe : l'algorithme de Sudan n'atteint pas le rayon de Johnson. Pour l'atteindre, il faudra généraliser cet algorithme en prenant en compte des multiplicités, mais c'est une autre (longue) histoire... voir les travaux de Guruswami et Sudan [GS99] pour les détails.

3.1 Taille de liste

Rappelons qu'un code de Reed–Solomon de longueur n, de dimension k et de points d'évaluation distincts $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ est défini comme

$$\mathrm{RS}_k(\boldsymbol{x}) \coloneqq \{(f(x_1), \ldots, f(x_n)) \mid f \in \mathbb{F}_q[x], \deg(f) < k\}\,.$$

Ce code est MDS : sa distance minimale est $d_{\min}(\mathrm{RS}_k(\boldsymbol{x})) = n - k + 1$ et son rayon de décodage unique est donc $t = \lfloor \frac{n-k}{2} \rfloor$.

Question 7.– Soit $\mathrm{RS}_k(\boldsymbol{x}) \in \mathbb{F}_q^n$ un code de Reed–Solomon. On rappelle que le rayon de décodage unique de ce code est $t = \lfloor \frac{n-k}{2} \rfloor$. Donner un mot $\boldsymbol{y} \in \mathbb{F}_q^n$ à distance $\leq t+1$ de $\mathrm{RS}_k(\boldsymbol{x})$, tel que

$$|\mathcal{B}(\boldsymbol{y}, t+1) \cap \mathrm{RS}_k(\boldsymbol{x})| \geq 2\,.$$

Justifier la réponse.

La question précédente illustre une situation légèrement défavorable, où la taille de la liste n'est pas égale à 1. Nous allons voir que cette situation est exceptionnelle. Pour cela, on souhaite donner une estimation de la probabilité, pour des mots $\boldsymbol{y} \in \mathbb{F}_q^n$ à distance $\leq t+1$ d'un mot $\boldsymbol{c} \in \mathrm{RS}_k(\boldsymbol{x})$, que ce mot soit **unique**.

Question 8.– 💻 Dans cette question, on fixe $n = 8$ et $k = 3$, et on considère des codes MDS. Par conséquent, $t = 2$. Pour les codes de Reed–Solomon de points d'évaluation $\boldsymbol{x} = (0, 1, 2, 3, 4, 5, 6, 7) \in \mathbb{F}_p^8$, avec des valeurs de p premières et grandissantes (partir de $p = 11$) :

1. Écrire une fonction qui prend en entrée $\boldsymbol{y} \in \mathbb{F}_p^8$ de poids $t+1 = 3$, et qui calcule le nombre de mots de code de $\mathrm{RS}_3(\boldsymbol{x})$ à distance ≤ 3 de $\boldsymbol{y}$.
2. En tirant des $\boldsymbol{y} \in \mathbb{F}_p^8$ de poids 3 aléatoirement (ou, pour les petits p, en parcourant tous les $\boldsymbol{y}$ possibles), estimer la probabilité que

$$|\mathcal{B}(\boldsymbol{y}, 3) \cap \mathrm{RS}_3(\boldsymbol{x})| \geq 2\,.$$

3.2 Décodage unique par l'algorithme de Berlekamp–Welch

Avant d'introduire le décodage en liste dans la section suivante, rappelons[2] l'algorithme de Berlekamp–Welch, qui permet de décoder n'importe quel code de Reed–Solomon jusqu'à son rayon de décodage unique t.

L'idée de l'algorithme de Berlekamp–Welch est la suivante. Notons $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$ le mot reçu, qui est la somme d'un mot de code $\boldsymbol{c} = (F(x_1), \ldots, F(x_n)) \in \mathrm{RS}_k(\boldsymbol{x})$ avec $\deg(F) \leq k-1$, et d'une erreur $\boldsymbol{e} = (e_1, \ldots, e_n) \in \mathbb{F}_q^n$ de poids $\mathrm{wt}(\boldsymbol{e}) \leq t$.

Notons $S = \{i \in \{1, \ldots, n\}, e_i \neq 0\}$ le support de l'erreur et $E(x) \coloneqq \prod_{i \in S}(X - x_i)$ le **polynôme localisateur** de l'erreur. Par définition, puisque $E(x_i) = 0$ pour tout $i \in S$, les n relations suivantes sont satisfaites :

$$y_i E(x_i) = F(x_i)E(x_i) \quad \forall i = 1, \ldots, n\,.$$

Pour retrouver le mot de code $\boldsymbol{c}$ à partir de $\boldsymbol{y}$, on peut alors chercher à résoudre cet ensemble d'équations, dont les inconnues sont les coefficients de $E(X)$ et de $F(X)$. Néanmoins, ces équations ne sont pas linéaires (il y a des produits d'inconnues dans le terme $F(x_i)E(x_i)$). L'idée est donc de **linéariser** les équations en posant $N(X) = -F(X)E(X)$. On obtient alors le système d'équations linéaires

$$y_i E(x_i) + N(x_i) = 0 \quad \forall i = 1, \ldots, n \tag{1}$$

2. cet algorithme a déjà été vu dans le cours de M1 de Codes Algébriques

où les polynômes $E(X), N(X) \in \mathbb{F}_q[X]$, de degrés respectifs $\deg(E) \leq t$ et $\deg(N) \leq t+k-1$, sont donc constitués d'au plus $(t+1)+(t+k) = 2t+k+1$ coefficients inconnus.

Pour résoudre explicitement le système, notons

$$E(X) = u_0 + u_1 X + \cdots + u_t X^t \quad \text{et} \quad N(X) = v_0 + v_1 X + \ldots v_{k+t-1} X^{k+t-1}$$

et définissons les matrices

$$\boldsymbol{A}(\ell; s) := \begin{pmatrix} y_1^\ell & y_1^\ell x_1 & \ldots & y_1^\ell x_1^{s-1} \\ y_2^\ell & y_2^\ell x_2 & \ldots & y_2^\ell x_1^{s-1} \\ \vdots & & & \vdots \\ y_n^\ell & y_n^\ell x_n & \ldots & y_n^\ell x_n^{s-1} \end{pmatrix} \in \mathbb{F}_q^{n \times \ell} \quad \text{pour } 0 \leq \ell, s \leq n\,.$$

Alors, le système à résoudre (1) s'écrit

$$\underbrace{\begin{pmatrix} \boldsymbol{A}(1; t+1) & \boldsymbol{A}(0; k+t) \end{pmatrix}}_{\text{matrice en blocs } \boldsymbol{M} \in \mathbb{F}_q^{n \times (2t+k+1)}} \cdot \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = \boldsymbol{0}$$

et peut être résolu, notamment, par un algorithme d'élimination gaussienne.

Une fois que les coefficients des polynômes $E(X)$ et $N(X)$ sont obtenus, on peut retrouver $F(X)$ — puis éventuellement le mot de code $\boldsymbol{c}$ correspondant — en observant que $F(X) = -N(X)/E(X)$. Il est important de noter que **toute** solution $(E(X), N(X))$ du système (1) est telle que $F(X) = -N(X)/E(X)$.

Résumons les résultats obtenus.

Lemme 3.1

Le couple de polynômes $(E(X), -F(X)E(X))$ est une solution du système (1)*, où $E(X)$ est le polynôme localisateur de l'erreur. Par ailleurs, si $(E_1(X), N_1(X))$ et $(E_2(X), N_2(X))$ sont deux solutions du système* (1)*, alors $N_1/E_1 = N_2/E_2 = -F$.*

Démonstration : Le fait que $(E(X), -F(X)E(X))$ est une solution du système (1) se vérifie immédiatement. Posons maintenant $R(X) = N_2(X)E_1(X) - N_1(X)E_2(X)$. Alors,

$$\deg(R) \leq \max\{\deg(N_2) + \deg(E_1), \deg(N_1) + \deg(E_2)\} \leq k + 2t - 1$$

Par ailleurs, pour tout $1 \leq i \leq n$, on a $R(x_i) = -y_i E_2(x_i) E_1(x_i) + y_i E_1(x_i) E_2(x_i) = 0$, donc R admet au moins n racines distinctes. Comme $n > k + 2t - 1$, nécessairement R est nul et on obtient le résultat escompté. ∎

Finalement, l'Algorithme 1 résume des différentes étapes de l'algorithme de Berlekamp–Welch.

Algorithme 1 : Algorithme de Berlekamp–Welch pour le décodage d'un code de Reed–Solomon $\mathrm{RS}_k(\boldsymbol{x})$.

Entrée : $\boldsymbol{y} \in \mathbb{F}_q^n$ un vecteur à distance $\leq t$ de $\boldsymbol{c} = (F(x_1), \ldots, F(x_n)) \in \mathrm{RS}_k(\boldsymbol{x})$.
Sortie : Le polynôme $F(X)$.

1. Construire la matrice $\boldsymbol{M} = \big(\boldsymbol{A}(1; t+1)\ ;\ \boldsymbol{A}(0; k+t)\big) \in \mathbb{F}_q^{n \times (k+2t+1)}$.
2. Trouver une solution $\boldsymbol{s} \in \mathbb{F}_q^{k+2t+1}$ de $\boldsymbol{M}\boldsymbol{s} = \boldsymbol{0}$.
3. Définir les vecteurs $\boldsymbol{u} \in \mathbb{F}_q^{t+1}$ et $\boldsymbol{v} \in \mathbb{F}_q^{t+k}$ de sorte que $\boldsymbol{s} = (\boldsymbol{u}, \boldsymbol{v})$, puis les polynômes $E(X) = \sum_{i=0}^{t} u_i X^i$ et $N(X) = \sum_{i=0}^{k+t-1} v_i X^i$.
4. Vérifier que $E(X)$ divise $N(X)$ et calculer $F(X) = -\frac{N(X)}{E(X)}$.
5. **Retourner** $F(X)$.

3.3 Algorithme de Sudan

L'algorithme de Sudan généralise l'algorithme de Berlekamp–Welch, et permet de « capturer » des erreurs de poids plus important. Pour y parvenir, on commence par reformuler les équations (1), en écrivant

$$Q(x_i, y_i) = 0 \quad \forall i = 1, \dots, n\,. \tag{2}$$

avec $Q(X, Y) \coloneqq N(X) + YE(X)$. On a alors vu dans la section précédente que, dans le contexte du **décodage unique**, le mot de code recherché $\boldsymbol{c} = (F(x_1), \dots, F(x_n))$ vérifie $Q(x_i, F(x_i)) = 0$ pour tout $i = 1, \dots, n$.

Pour le décodage **en liste**, l'idée est d' « autoriser » davantage de solutions en cherchant un polynôme

$$Q(X, Y) \coloneqq Q_0(X) + Q_1(X)Y + \cdots + Q_L(X)Y^L \in \mathbb{F}_q[X, Y]\,,$$

de degré $L \geq 1$, tel que $Q(x_i, y_i) = 0$ pour tout $i = 1, \dots, n$. La valeur de L donne une borne au nombre de solutions que l'on recherche, et sera fixée en fonction de w, le poids maximal de l'erreur que l'on souhaite corriger.

Une fois que $Q(X, Y)$ est retrouvé, on cherche à obtenir la liste des polynômes $F(X)$ tels que $Q(x_i, F(x_i)) = 0$ pour tout $i = 1, \dots, n$. On montrera plus tard que, sous des contraintes de degré sur les $Q_j(X)$, les mots de code de $\mathrm{RS}_k(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{y}, w)$ sont tous dans cette liste.

Définition des paramètres. Les contraintes de degré sur les $Q_j(X)$ sont les suivantes[3] :

$$\deg(Q_j(X)) \leq n - w - 1 - (k-1)j, \quad \text{pour tout } j = 0, \dots, L\,.$$

Elles permettent donc de définir la taille de liste L en fonction de w par

$$L = \left\lfloor \frac{n - w - 1}{k - 1} \right\rfloor .$$

Remarque. Il est clair que la nouvelle définition de $Q(X, Y)$ généralise la précédente, en posant $Q_0(X) = N(X)$ et $Q_1(X) = E(X)$.

L'Algorithme 2 résume les différentes étapes présentées ci-dessus. Nous allons voir par la suite comment effectuer efficacement les deux étapes principales : l'**interpolation** et la **recherche de racine**.

Algorithme 2 : Algorithme de Sudan pour le décodage en liste d'un code de Reed–Solomon $\mathrm{RS}_k(\boldsymbol{x})$.

Entrée : $\boldsymbol{y} \in \mathbb{F}_q^n$ et $w \leq n$ un poids d'erreur.
Sortie : L'ensemble des polynômes $F(X)$ tel que $\boldsymbol{c} = (F(x_1), \dots, F(x_n))$ est à distance $\leq w$ de $\boldsymbol{y}$.

1 Définir $L = \lfloor (n - w - 1)/(k - 1) \rfloor$.

2 **Étape d'interpolation.** Trouver les coefficients du polynôme $Q(X, Y) = \sum_{i=0}^{L} Q_j(X)Y^j$ tel que :

$$\begin{cases} Q(x_i, y_i) = 0, \ \forall i = 1, \dots, n \\ \deg(Q_j(X)) \leq n - w - 1 - (k-1)j, \quad \forall j = 0, \dots, L \end{cases}$$

3 **Étape de recherche de racines.** Trouver tous les polynômes $F(X) \in \mathbb{F}_q[X]$ de degré $\leq k - 1$, tels que $Q(X, F(X)) = 0$ et $\mathrm{d}_H(\boldsymbol{y}, (F(x_1), \dots, F(x_n))) \leq w$.

4 **Retourner** la liste de ces polynômes.

Étape d'interpolation. Définissons formellement ce que nous souhaitons réaliser dans cette étape. Étant donné $\boldsymbol{y} \in \mathbb{F}_q^n$ à distance $\leq w$ d'un mot du code $\mathrm{RS}_k(\boldsymbol{x})$, le but est de retrouver les coefficients d'un **polynôme interpolateur**

$$Q(X, Y) \coloneqq Q_0(X) + Q_1(X)Y + \cdots + Q_L(X)Y^L \in \mathbb{F}_q[X, Y]$$

3. En réalité, on souhaite que le degré de $Q(X, Y)$, **pondéré** par un facteur k en Y, soit $\leq n - w - 1$.

tel que

$$Q(x_i, y_i) = 0, \quad \forall i = 1, \ldots, n$$

et

$$\deg(Q_j(X)) \leq r_j \coloneqq n - w - 1 - (k-1)j, \quad \forall j = 0, \ldots, L\,.$$

Nous avons donc au plus $r = \sum_{j=0}^{L}(r_j + 1) = \sum_{j=0}^{L}(n - w - (k-1)j)$ coefficients à retrouver.

Question 9.– Rappelons que $L = \lfloor (n-w-1)/(k-1) \rfloor$. Démontrer que $r > n$.

Comme pour le décodage unique (et l'algorithme de Berlekamp–Welch), ces r coefficients peuvent être trouvés en résolvant un système d'équations linéaires :

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A}(L; r_L) & \boldsymbol{A}(L-1; r_{L-1}) & \ldots & \boldsymbol{A}(0; r_0) \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{u}_L \\ \boldsymbol{u}_{L-1} \\ \vdots \\ \boldsymbol{u}_0 \end{pmatrix} = \boldsymbol{0}$$

où les matrices $\boldsymbol{A}(j; r_j)$ ont été définies précédemment, et où chaque $\boldsymbol{u}_j$ contient les coefficients du polynôme $Q_j(X)$ recherché. Du fait que $r > n$, ce système admet au moins une solution non-nulle.

Algorithme 3 : Étape d'interpolation (par élimination gaussienne).

Entrée : $\boldsymbol{y} \in \mathbb{F}_q^n$, $w \leq n$ le poids d'erreur et $L = \lfloor (n-w-1)/(k-1) \rfloor$.

Sortie : Un polynôme $Q(X, Y) = \sum_{i=0}^{L} Q_j(X) Y^j$ tel que $Q(x_i, y_i) = 0$ pour tout $i = 1, \ldots, n$, et $\deg(Q_j(X)) \leq n - w - 1 - (k-1)j$ pour tout $j = 0, \ldots, L$.

1 Définir $r_j = n - w - 1 - (k-1)j$ pour tout $j = 0, \ldots, L$ et $r = \sum_{j=0}^{L}(r_j + 1)$

2 Construire la matrice

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A}(L; r_L + 1) & \boldsymbol{A}(L-1; r_{L-1} + 1) & \ldots & \boldsymbol{A}(0; r_0 + 1) \end{pmatrix} \in \mathbb{F}_q^{n \times r}$$

3 Résoudre le système linéaire $\boldsymbol{M}\boldsymbol{u} = \boldsymbol{0}$

4 Découper $\boldsymbol{u} = (\boldsymbol{u}_L, \ldots, \boldsymbol{u}_0)$ en $L+1$ morceaux de taille $r_L + 1, \ldots, r_0 + 1$, puis construire les polynôme $Q_j(X) = \sum_{i=0}^{r_j} (\boldsymbol{u}_j)_i X^i$ pour $j = 0, \ldots, L$.

5 **Retourner** le polynôme $Q(X, Y) = \sum_{j=0}^{L} Q_j(X) Y^j$.

Question 10.– 💻 Implanter l'étape d'interpolation.

Vérifier l'implantation avec les valeurs de test suivantes. Pour le code de Reed–Solomon sur $\mathbb{F}_{11}$, de longueur $n = 8$ et de dimension $k = 2$, et pour le mot $\boldsymbol{y} = (3,3,10,6,5,9,10,2) \in \mathbb{F}_{11}^8$ à distance $w = 4$ du code, on obtient :

— la matrice

$$\boldsymbol{M} = \begin{pmatrix} 1 & 0 & 0 & 0 & 3 & 0 & 0 & 9 & 0 & 5 \\ 1 & 1 & 1 & 1 & 3 & 3 & 3 & 9 & 9 & 5 \\ 1 & 2 & 4 & 8 & 10 & 9 & 7 & 1 & 2 & 10 \\ 1 & 3 & 9 & 5 & 6 & 7 & 10 & 3 & 9 & 7 \\ 1 & 4 & 5 & 9 & 5 & 9 & 3 & 3 & 1 & 4 \\ 1 & 5 & 3 & 4 & 9 & 1 & 5 & 4 & 9 & 3 \\ 1 & 6 & 3 & 7 & 10 & 5 & 8 & 1 & 6 & 10 \\ 1 & 7 & 5 & 2 & 2 & 3 & 10 & 4 & 6 & 8 \end{pmatrix} \in \mathbb{F}_{11}^{8\times 10},$$

— dont une solution de l'équation $\boldsymbol{M}\boldsymbol{u} = \boldsymbol{0}$ est $\boldsymbol{u} = (4,7,2,10,7,8,1,7,1,0)$,

— et qui produit par conséquent le polynôme

$$Q(x,y) = (4 + 7x + 2x^2 + 10x^3) + (7 + 8x + x^2)y + (7 + x)y^2 .$$

Remarque : la matrice **doit** être celle donnée ci-dessus, mais il se peut que vous obteniez une autre solution et un autre polynôme $Q(x,y)$.

Étape de recherche de racines. Supposons maintenant qu'un polynôme interpolateur $Q(X,Y)$ a été retrouvé. On cherche maintenant à identifier tous les polynômes $F(X)$ de degré $\leq k-1$ tels que $Q(x_i, F(x_i)) = 0$ pour tout $i = 1,\dots,n$.

Lemme 3.2

Soit $F(X) \in \mathbb{F}_q[X]$ de degré $\leq k-1$. Supposons que $Q(x_i, F(x_i)) = 0$ pour tout $i = 1,\dots,n$. Alors, $Q(X, F(X))$ est le polynôme nul de $\mathbb{F}_q[X]$; autrement dit, $Y - F(X)$ divise $Q(X,Y)$.

Question 11.– Démontrer le Lemme 3.2.
Indication : compter les racines et comparer au degré.

Le lemme précédent signifie donc que l'on peut retrouver les polynômes cherchés parmi les "racines en Y" de $Q(X,Y)$, c'est-à-dire parmi ses facteurs de la forme $Y - F(X)$.

Roth et Ruckenstein ont proposé dans [RR00] un algorithme "simple" permettant d'effectuer cette recherche de racine (qui est plus simple que de cherche une factorisation complète de $Q(X,Y)$). La méthode est détaillée dans l'Algorithme 4.

Algorithme 4 : Étape de recherche de racines, par l'algorithme de Roth–Ruckenstein.

Entrée : Un polynôme $Q(X,Y) = \sum_{i=0}^{L} Q_j(X)Y^j$ tel que $Q(x_i, y_i) = 0$ pour tout $i = 1,\dots,n$, et $\deg(Q_j(X)) \leq n - w - 1 - (k-1)j$ pour tout $j = 0,\dots,L$.

Sortie : La liste de tous les polynômes $F(X) \in \mathbb{F}_q[X]$ de degré $\leq k-1$ tels que $Y - F(X)$ divise $Q(X,Y)$.

1 $\phi(X) \leftarrow 0$
2 `solutions` $\leftarrow$ `[]`
3 Appeler la fonction auxiliaire `Reconstruct`$(Q(X,Y), 0)$ (voir Algorithme 5)
4 **Retourner** la liste `solutions`

Algorithme 5 : Fonction récursive auxiliaire `Reconstruct`, pour la recherche de racines.

Entrée : Un polynôme $A(X,Y) = \sum_{i=0}^{L} A_j(X)Y^j$, un entier $1 \leq j \neq n$, un accès à une liste de solutions `solutions`, un accès à un polynôme $\phi(X) = \sum_{i=0}^{k-1} \phi_i X^i$.

Sortie : $\varnothing$

Tant que $A(X,Y)$ *est divisible par* X **faire**
  $A(X,Y) \leftarrow A(X,Y)/X$
Calculer les racines dans $\mathbb{F}_q$ de $A_0(X)$
Pour tout *racine* γ *de* $A_0(X)$ **faire**
  Dans $\phi(X)$, modifier le coefficient ϕ_j en γ
  Si $j = k-1$
    Ajouter $\phi(X)$ à la liste `solutions`.
    Terminer.
  Modifier $A(X,Y)$ en $A(X, X(Y+\gamma))$
  Appeler récursivement `Reconstruct`$(A(X,Y), j+1)$.
Terminer.

Question 12.– 💻 Implanter l'étape de recherche de racines, puis l'algorithme de décodage en liste de Sudan.

Vérifier l'implantation avec les valeurs de test reprises de la question 10. Pour le polynôme

$$Q(x,y) = (4 + 7x + 2x^2 + 10x^3) + (7 + 8x + x^2)y + (7+x)y^2 \in \mathbb{F}_{11}[x,y]\,,$$

on obtient la liste de polynômes

$$\{3 + 3x, 7 + 7x\}$$

qui correpondent aux mots de code

$$\boldsymbol{c}_1 = (3,6,9,1,4,7,10,2) \quad \text{et} \quad \boldsymbol{c}_2 = (7,3,10,6,2,9,5,1)\,.$$

On vérifie enfin que seul $\boldsymbol{c}_2$ se trouve à distance ≤ 4 de $\boldsymbol{y}$.

Question 13.– Estimer la complexité en temps de votre implantation de l'algorithme de Sudan.

Références

[Aug10] Daniel Augot. Les codes algébriques principaux et leur décodage. In Jean-Guillaume Dumas, Grégoire Lecerf, Delphine Boucher, and Thomas Cluzeau, editors, *Journées Nationales de Calcul Formel*, volume 1 of *Les cours du CIRM*, pages 31–74, Luminy, France, May 2010. CIRM. `https://inria.hal.science/inria-00543322v1/file/Cours.pdf`.

[GRS23] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential Coding Theory. `https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/`, 2023.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inf. Theory*, 45(6) :1757–1767, 1999.

[RR00] Ron M. Roth and Gitit Ruckenstein. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. *IEEE Trans. Inf. Theory*, 46(1) :246–257, 2000.

[Sud97] Madhu Sudan. Decoding of Reed Solomon Codes beyond the Error-Correction Bound. *J. Complex.*, 13(1) :180–193, 1997.