Anneau des Séries Formelles, et Applications en Calcul Formel et Cryptographie

Version corrigée, du 31 Octobre 2025

Changelog

Dans la version initiale, il y avait une coquille dans les questions 19 et 20.

1 Introduction

L'objectif de ce DM est d'étudier du point de vue du calcul formel des objets appelés « séries formelles » et généralisant la notion de polynômes en autorisant un nombre infini de coefficients [1]. Aucune connaissance préalable sur les séries formelles n'est requise pour mener à bien ce travail.

1.1 Mise en bouche

Dans la suite, $\mathbb{K}$ sera un corps. Sauf mention préalable, vous pouvez prendre $\mathbb{K} = \mathbb{Q}, \mathbb{R}, \mathbb{C}$ ou encore un corps fini $\mathbb{K} = \mathbb{F}_q$ si vous voulez vous fixer les idées.

1. Si cette notion vous parle, vous pouvez rapprocher la notion de série formelle à celle de série entière en analyse réelle et/ou complexe.

Définition. Soit $\mathbb{K}^{\mathbb{N}}$ l'ensemble des suites à valeurs dans $\mathbb{K}$. On peut le munir d'une structure d'anneau en définissant la somme de deux suites, terme à terme :

$$(a) + (b) \stackrel{\text{def}}{=} (c) \quad \text{où } c_n = a_n + b_n,$$

et le produit est aussi appelé **produit de convolution**

$$(a) \times (b) \stackrel{\text{def}}{=} (c) \quad \text{où } c_n = \sum_{k=0}^{n} a_k b_{n-k}.$$

En général, lorsque l'on considère ces opérations, on note *formellement* les suites comme les coefficients d'une somme infinie

$$(a_n) \longleftrightarrow A(X) \stackrel{\text{def}}{=} \sum_{i \geq 0} a_i X^i,$$

et l'ensemble des suites est alors noté $\mathbb{K}[[X]]$ et est appelé « Anneau des séries formelles à coefficients dans $\mathbb{K}$ ».

Remarque 1. *Il s'agit bien d'une notation. Bien qu'on puisse lui donner un sens, on ne définira pas ici de notion de convergence. C'est pour ça que l'on parle de série « formelle ».*

(Q1) Vérifiez que les opérations $+$ et $\times$ sont bien définies et que $\mathbb{K}[[X]]$ est alors un anneau commutatif. Quels sont les neutres pour $+$ et $\times$?

(Q2) Vérifiez que l'anneau $\mathbb{K}[X]$ s'injecte naturellement dans $\mathbb{K}[[X]]$, c'est-à-dire qu'on peut voir un polynôme comme une série formelle, et que le produit de deux polynômes, vus comme séries formelles, est encore un polynôme et est bien le produit auquel on s'attend (l'inclusion $\mathbb{K}[X] \hookrightarrow \mathbb{K}[[X]]$ est un morphisme d'anneaux injectif).

(Q3) Montrez que le polynôme $1 - X$ est inversible dans $\mathbb{K}[[X]]$, d'inverse $\sum_{i \geq 0} X^i$.

Ainsi, $\mathbb{K}[[X]]$ contient la fraction rationelle $\dfrac{1}{1-X}$, alors que $1 - X$ n'est pas inversible dans $\mathbb{K}[X]$ (on ne demande pas de le montrer). Autrement dit, $\mathbb{K}[[X]]$ est strictement plus gros que $\mathbb{K}[X]$. Par contre, la question suivante nous montre que $\mathbb{K}[[X]]$ n'est pas lui-même un corps ; en particulier il ne contient pas toutes les fractions rationelles.

(Q4) Soit $A(X) \stackrel{\text{def}}{=} \sum_{n \geq 0} a_n X^n$ une série formelle inversible. Montrez que nécessairement $a_0 \neq 0$.

(Q5) En déduire que X n'a pas d'inverse dans $\mathbb{K}[[X]]$.

(Q6) Réciproquement, montrez que l'ensemble des séries formelles inversibles est exactement l'ensemble des séries de coefficient constant non nul.

1.2 Séries Formelles Tronquées

Algébriquement, les séries formelles sont des objets infinis, qu'on ne peut donc pas manipuler de manière exacte. Algorithmiquement, on ne conserve qu'un nombre fini n de termes. Ainsi, une série formelle F sera représentée par un polynôme P de degré au plus $n-1$, et on notera $F = P + O(X^n)$ (ou encore $F = P \mod X^n$). Cette notation aura encore un sens si P est aussi une série formelle, et signifie que F et P coïncident sur les n premiers coefficients.

Formellement, la troncature d'une série formelle F à l'ordre n est son image par le morphisme d'anneaux $\mathbb{K}[[X]] \to \mathbb{K}[[X]]/(X^n)$, ce dernier anneau étant isomorphe à $\mathbb{K}[X]/(X^n)$.

Dans toute la suite, on supposera que $\mathbb{K}$ est un corps effectif et qu'on en a une représentation en machine. On représentera alors les séries formelles tronquées à l'ordre n comme des listes de taille n à coefficients dans $\mathbb{K}$ (comme pour les polynômes dans les TD). Par exemple, la série $1 + X + O(X^3)$ sera représentée comme la liste `[1, 1, 0]`.

(Q7) Expliquez pourquoi le produit de deux séries formelles tronquées à l'ordre $n = 2^\ell$ peut se faire en $O(n \log(n))$ opérations dans $\mathbb{K}$. Donnez explicitement l'algorithme pour le faire (il n'est pas demandé de le programmer).

2 Division selon les Puissances Croissantes

2.1 Un premier algorithme

On considère l'algorithme mystère suivant

Algorithme 1 : Algorithme Mystère

Entrées : Deux polynômes $a(x), b(x) \in K[x]$ tel que $b(0) \neq 0$, et un entier $n \geq 1$.
Sorties : Un polynôme $c(x)$

```
1 Initialiser une liste c de taille n par c ← [0,...,0].
2 Poser inv_b0 ← b[0]^{-1}.
3 pour k = 0 à n − 1 faire
4     s ← a[k];
5     pour i = 1 à k faire
6         s ← s − b[i] · c[k − i];
7     c[k] ← inv_b0 · s;
8 retourner c
```

(Q8) Executez (à la main) l'algorithme pour les polynômes $A(X) = 1 + X$ et $B(X) = 1 + X^2$, avec $n = 3$.

(Q9) Implémentez l'algorithme et le tester avec plusieurs exemples pertinents.

(Q10) Quelle est la complexité de cet algorithme ? On comptera le nombre d'opérations dans $\mathbb{K}$.

(Q11) Soient $A, B \in \mathbb{K}[X]$, et soit $n \geq 0$ un entier. On suppose que $b(0) \neq 0$. Démontrez qu'il existe alors un unique couple de polynômes (C, R) où C est de degré au plus $n-1$ tel que

$$A(X) = B(X) \cdot C(X) + X^n \cdot R(X) = B(X) \cdot C(X) + O(X^n).$$

Indication : On pourra par exemple (méthodes non exhaustives) raisonner sur le polynôme B vu comme série formelle, ou bien donner une preuve algorithmique comme pour la division euclidienne classique.

(Q12) Montrez que l'Algorithme 1 calcule ce polynôme C, appelé *quotient par les puissances croissantes* de A par B, à l'ordre n.

(Q13) Soit $F = P + O(X^n)$ une série formelle inversible. Donnez un algorithme calculant un polynôme G tel que $G + O(X^n)$ soit la troncature à l'ordre n de l'inverse de F.

(Q14) Quel est l'inverse à l'ordre 4 de la série

$$F(X) = 1 + 2X - 3X^2 + X^3 + O(X^4) \in \mathbb{Q}[[X]]?$$

2.2 Méthode de Newton

La méthode de Newton est très utilisée en analyse numérique pour calculer les zéros d'une fonction $f : \mathbb{R} \to \mathbb{R}$ suffisamment régulière. L'idée est de partir d'une solution approchée x_0 et de remplacer l'équation $f(t) = 0$ par l'équation approchée $f(x_0) + (t - x_0)f'(x_0) = 0$ (issue du développement limité de f en x_0), d'où on déduit

$$t = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Bien entendu, cette formule n'a pas toujours de sens, mais on espère que quand elle en a un, la suite

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{1}$$

obtenue en itérant ce processus converge vers un zéro de f. C'est le cas sous de nombreuses hypothèses que nous n'allons pas préciser, et la convergence est même quadratique : le nombre de décimales exactes va doubler à chaque itération.

L'algèbre permet de donner des sens rigoureux à la notion de limite, mais nous n'allons pas préciser plus que ça. On peut par exemple dire qu'un polynôme de $K[X]$ est très petit lorsqu'il est divisible par une grosse puissance de X. C'est d'ailleurs ce qu'on fait en tronquant des séries formelles à l'ordre n : on ignore les « petits » termes. On va maintenant voir que l'intuition analytique nous permet de définir un algorithme très efficace pour inverser des séries formelles.

(Q15) Justifiez qu'il suffit d'écrire un algorithme pour inverser des séries formelles de terme constant égal à 1.

Soit $F = P + O(X^n) \in \mathbb{K}[[X]]$ de terme constant 1. Afin d'inverser F, on se propose de résoudre l'équation

$$\varphi(G) = 0$$

où $\varphi : \mathbb{K}[[X]]^\times \to \mathbb{K}[[X]]$ est donnée par

$$\varphi(G) = \frac{1}{G} - F.$$

L'intuition analytique nous pousse alors à définir la suite

$$G_{k+1} = G_k - \frac{\varphi(G_k)}{\varphi'(G_k)}.$$

Encore faudrait-il définir φ'... Mais poussons le vice plus loin : en considérant G comme une variable formelle, et P comme une constante, vous conviendrez qu'une notion raisonnable pour $\varphi'(G)$ est

$$\frac{-1}{G^2}.$$

On définit alors la récurrence

$$G_{k+1} = G_k - \frac{\frac{1}{G_k} - P}{-\frac{1}{G_k^2}} = 2G_k - G_k^2 P$$

ou encore

$$G_{k+1} = G_k(2 - P \cdot G_k). \tag{2}$$

Si $G_0 = 1$, alors la suite G_k est bien définie, et comme $P \in \mathbb{K}[X]$, on a même $G_k \in \mathbb{K}[X]$ quel que soit l'entier k.

(Q16) Soit $G_k \in \mathbb{K}[X]$ la suite de polynômes définie par la recurrence ci-dessus. Démontrez que quel que soit l'entier k,

$$P \cdot G_k = 1 + O(X^{2^k}).$$

On en déduit l'algorithme suivant pour le calcul de l'inverse d'une série formelle

Algorithme 2 : Inversion par la méthode de Newton

Entrées : $n = 2^k \in \mathbb{N}^\star$ et $F = P + O(X^n) \in \mathbb{K}[[X]]$ de terme constant 1.
Sorties : $G = Q + O(X^n) \in \mathbb{K}[[X]]$ tel que $FG = 1 + O(X^n)$

```
si n = 1 alors
    retourner 1
sinon
    G_0 ← 1;
    Calculer récursivement G_1, ..., G_k comme dans l'équation (2);
    retourner G_k
```

(Q17) Soit M le coût de la multiplication de deux polynômes de degré n. Celui-ci dépend de l'algorithme choisi pour la multiplication (naïf, Karatsuba, FFT ou autre). On suppose que pour tout n, $M(n) \geq 2M(n/2)$ (c'est le cas pour tous les algorithmes que nous avons vus en cours). Démontrez alors que l'algorithme 2 calcule l'inverse d'une série formelle en $O(M(n))$ opérations dans $\mathbb{K}$.

2.3 Application à la Division Rapide de Polynômes

On va maintenant voir que le problème d'inverser des séries formelles tronquées a en réalité une grosse application au calcul de la division euclidienne classique de polynômes.

(Q18) Soient $A, B \in \mathbb{K}[X]$ deux polynômes. On note $n \stackrel{\text{def}}{=} \deg(A)$ et $m \stackrel{\text{def}}{=} \deg(B)$. On suppose que $n > m$. Rappelez un algorithme pour calculer $(Q, R) \in \mathbb{K}[X]^2$ tel que

$$A = BQ + R, \quad \text{et } \deg(R) < m$$

en $O(n\,(n-m))$ opérations dans $\mathbb{K}$.

On se propose de calculer Q à l'aide d'une inversion de série formelle. Pour ça, on écrit (formellement)

$$\frac{A}{B} = Q + \frac{R}{B}.$$

À priori, cette égalité a un sens dans le monde des fractions rationelles, mais pas forcément dans celui des séries formelles (si $B(0) = 0$). Pour pallier ce problème, on va considérer un changement de variable. Si $P \in \mathbb{K}[X]$ est de degré π, on définit

$$P^\star(X) \stackrel{\text{def}}{=} X^\pi P\left(\frac{1}{X}\right)$$

appelé polynôme réciproque de P.

(Q19) Montrez que pour tout polynôme P, son réciproque $P^\star$ est encore un polynôme, de même degré si $P(0) \neq 0$. Montrez également que $X^{\deg P} P^\star(1/X) = P$. En déduire en particulier que si $P(0) \neq 0$, alors $P \mapsto P^\star$ est une involution.

(Q20) Montrez que $B^\star$ est inversible dans $\mathbb{K}[[X]]$ et que

$$\frac{A^\star}{B^\star} = Q^\star + O\left(X^{\deg A - \deg B + 1}\right),$$

où le $O(\cdot)$ dépend explicitement de R.

(Q21) Soit $A, B \in \mathbb{K}[X]$ avec $\deg(A) = n$ et $\deg(B) = m \leq n$. Soit M une fonction telle que le produit de deux polynômes de degrés au plus n puisse se faire en $O(M(n))$ opérations. Donner un algorithme qui effectue la division euclidienne de A par B en $O(M(n))$ opérations également.

(Q22) Donnez un exemple de cas où cet algorithme est asymptotiquement meilleur que l'algorithme classique de division euclidienne.

3 Linear Feedback Shift Registers (LFSR)

Les séries formelles ont de nombreuses applications au-delà du calcul de division euclidienne que nous venons de présenter. Elles sont par exemple extrêmement utiles pour faire du dénombrement. Dans cette dernière partie, nous vous proposons une autre application, cette fois-ci à la génération de bits aléatoires en cryptographie.

3.1 Premières définitions

LFSR. Un générateur à décalage à rétroaction linéaire binaire[2] de longueur ℓ (plus couramment appelé par son nom en anglais *Linear Feedback Shift Registers*) est une machine à états finis formée d'une mémoire de ℓ cellules appelée registre, et produisant une suite récurrente linéaire à coefficients dans $\mathbb{F}_2$ de la forme

$$s_{t+\ell} = c_1 s_{t+\ell-1} + \cdots + c_\ell s_t, \quad \text{pour } t \geq 0,$$

où les coefficients $c_1, \ldots, c_\ell \in \mathbb{F}_2$ sont fixés.

Plus précisément, le registre est initialisé avec un état $S = (s_0, \ldots, s_{\ell-1}) \in \mathbb{F}_2^\ell$, et à chaque unité de temps t (réalisée par une horloge interne), chaque cellule du registre est décalée d'une cellule vers la droite. Le contenu de la cellule la plus à droite sort du registre (et est le bit s_t produit par le LFSR au temps t) et la cellule la plus à gauche reçoit en entrée le bit produit par la suite récurrente linéaire.

Remarque 2. *Il est facile de voir (on ne demande pas de l'écrire) que toute suite récurrente linéaire peut être obtenue comme les bits de sortie d'un LFSR.*

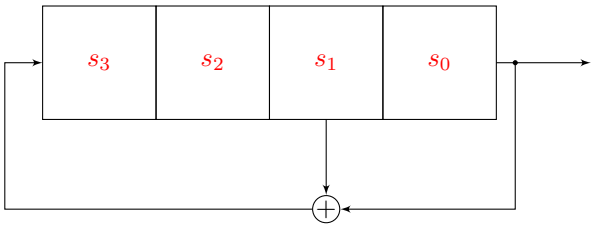

FIGURE 1 – LFSR implémentant la récurrence $s_{t+4} = s_{t+1} + s_t$

Par l'exemple, le LFSR de la figure 1 initialisé avec l'état $(s_0, s_1, s_2, s_3) = (1, 0, 1, 1)$ produit une suite dont les premiers termes sont $(1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0)$.

Remarque 3. *Attention au sens des LFSR avec cette convention. Dans l'état initial,* s_0

2. On peut bien entendu généraliser ceci à des corps plus gros.

est bien à droite (puisque c'est le premier bit à sortir) !

L'intérêt principal des LFSR est qu'ils ont un design si simple qu'ils ont des implémentations matérielles (*i.e.*, hardware) extrêmement efficaces, et qu'ils produisent des suites de bits ayant de bonnes propriétés statistiques (que nous n'allons pas préciser plus que ça). Ça en fait de bons candidats pour la génération de bits aléatoires utilisés pour la cryptographie.

Polynôme de rétroaction. Les coefficients d'un LFSR sont en général représentés sous la forme d'un coefficient à valeurs dans $\mathbb{F}_2$, défini par

$$P \stackrel{\text{def}}{=} 1 + \sum_{i=1}^{\ell} c_i X^i.$$

(Q23) Implémentez en Sage une fonction prenant en entrées un état initial sous forme d'une liste de taille ℓ, un polynôme de rétroaction et un entier N, et produisant les N premiers termes de la suite (faites attention au sens et à vos indices). Vérifiez qu'avec le LFSR de la figure 1 vous obtenez bien la suite définie plus haut.

(Q24) Démontrez que toute suite produite par un LFSR de longueur ℓ d'état initial non nul est forcément périodique de période $\leq 2^\ell - 1$.

On considère le LFSR représenté en figure 2.

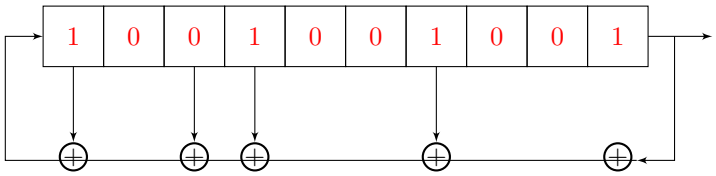

FIGURE 2 – Un LFSR de longueur 10

(Q25) Quel est le polynôme de rétroaction de ce LFSR ?

(Q26) Vérifiez cette même suite est définie par un LFSR de longueur 3, dont le polynôme de rétroaction est $1 + X^3$.

Ainsi, une suite peut-être produite par des LFSR de longueurs différentes. Or, pour les applications cryptographiques, on cherche à générer des suites de période maximale, celle-ci étant liée au degré du polynôme de rétroaction du LFSR. Cette non-unicité semble être un frein à la caractérisation de suite maximale. Nous allons utiliser les séries formelles pour lever cette limitation.

3.2 LFSR et séries formelles

On considère une suite $(s_n) \in \mathbb{F}_2^{\mathbb{N}}$ produite par un LFSR d'ordre ℓ, et on la représente sous la forme de sa série formelle associée

$$S(X) \stackrel{\text{def}}{=} \sum_{i \geq 0} s_i X^i \in \mathbb{F}_2[[X]].$$

On note P un polynôme de rétroaction de ce LFSR (donc de degré ℓ).

(Q27) Démontrez qu'il existe un polynôme $Q \in \mathbb{F}_2[X]$ de degré $\deg Q < \ell$ et tel que

$$S(X) = \frac{Q(X)}{P(X)}.$$

On cherchera une expression explicite des coefficients de $Q(X)$.

(Q28) Montrez qu'il existe un unique polynôme P_0 de terme constant 1 tel que la série formelle $S(X)$ soit de la forme

$$S(X) = \frac{Q_0(X)}{P_0(X)}, \quad \text{et } \operatorname{pgcd}(P_0, Q_0) = 1.$$

(Q29) Montrez que P_0 est de degré minimal parmi tous les polynômes de rétroaction produisant la suite.

(Q30) En déduire que si P est un polynôme de rétroaction pour une suite, et qu'il est irréductible, alors c'est le polynôme minimal.

Ce polynôme P_0 est appelé polynôme de rétroaction minimal de la suite (s_n), et ne dépend que de celle-ci.

(Q31) Donnez un algorithme qui, étant donnée une suite récurrente linéaire définie par ses N premiers termes représentés sous la forme d'une liste de taille N, produit le polynôme de rétroaction minimal.

(Q32) Donnez un exemple de suite récurrente linéaire dont le polynôme de rétroaction minimal n'est pas irréductible.

On peut aussi manipuler les séries formelles directement comme un objet Sage via le constructeur `PowerSeriesRing`. Par exemple

```
sage: F2 = GF(2)
sage: PS.<T> = PowerSeriesRing(F2)
sage: A.<X> = F2[]
```

définit les anneaux $\mathbb{F}_2[[T]]$ stocké dans PS, et $\mathbb{F}_2[X]$ stocké dans A. Notez que j'ai utilisé deux variables différentes pour bien les séparer dans Sage. Comme pour les polynômes, on peut définir une série formelle associée à une suite par la commande suivante

```
sage: s = [1, 0, 0, 1, 0, 0, 1, 0, 0, 1]
sage: S = PS(s)
```

Attention, par défaut, Sage va considérer que les termes suivants sont tous nuls. Pour lui préciser que vous avez tronqué la série, vous pouvez utiliser la notation O comme par exemple :

```
sage: S = PS(s) + O(T^10)
```

(Q33) Retrouvez le polynôme de rétroaction minimal en manipulant directement les séries avec ce constructeur.

(Q34) (**Bonus**) Démontrez qu'une suite récurrente linéaire de polynôme de rétroaction minimal P_0 de degré ℓ est maximalement périodique (*i.e.*, de période $2^\ell - 1$), si et seulement si P_0 est primitif (c'est-à-dire irréductible, et dont les racines engendrent le groupe $(\mathbb{F}_{2^\ell})^\times$).

Remarque 4. *Un algorithme dû à Berlekamp et Massey permet d'obtenir ce polynôme de rétroaction minimal efficacement. Il repose exactement sur ce lien avec les séries formelles.*