Université Paris 8 Année 2021–2022

Master 2 Mathématiques et applications

Parcours ACC (Arithmétique, Codage et Cryptologie)

Algorithmes Arithmétiques

Notes de cours

Julien Lavauzelle
julien.lavauzelle@univ-paris8.fr

19 octobre 2021

Table des matières

Quelques références utiles

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (3. ed.)*. Cambridge University Press, 2013

Chapitre 1

Introduction

1.1 Des algorithmes en arithmétique

Todo: à remplir

1.2 Applications

Todo: à remplir

Chapitre 2

Algèbre linéaire

La résolution de nombreux problèmes d'arithmétiques comporte une phase dite d'*algèbre linéaire*, c'est-à-dire de résolution d'un système d'équations linéaires. En cryptanalyse, on la retrouve notamment dans les meilleurs algorithmes de factorisation d'entiers ou de polynômes, et de calcul de logarithme discret (voir chapitres suivants).

Dans ce chapitre, nous rappellerons la méthode générique de résolution par *élimination gaussienne*, puis nous étudierons un algorithme particulière efficace lorsque les équations du système comportent une proportion importante de coefficients nuls.

2.1 Rappels

Dans tout le chapitre, $\mathbb{F}$ désigne un corps dont les éléments sont représentables sur une machine, et pour lequel on peut effectuer efficacement les opérations élémentaires (addition, multiplication, opposé, inverse, test à zéro, etc.). On dit que $\mathbb{F}$ est un *corps effectif*.

Les corps effectifs utilisés en pratique sont par exemple les corps finis $\mathbb{F}_q$, ou le corps $\mathbb{Q}$ des rationnels et ses extensions algébriques.

Notations. Les n-uplets et les vecteurs sont généralement représentés par un caractère gras et en minuscule : $\boldsymbol{a} \in \mathbb{Z}^n$ par exemple. On assimile les vecteurs à des matrices colonnes.

On note $\mathbb{F}^{n\times m}$ l'ensemble des matrices de taille $n \times m$. Généralement, on représente une matrice en gras et en majuscules. La i-ème ligne d'une matrice $\boldsymbol{A}$ est génériquement notée $\boldsymbol{A}_{i,*}$. Sa j-ème colonne est notée $\boldsymbol{A}_{*,j}$.

2.1.1 Le problème de la résolution de systèmes linéaires

Définition 2.1

Un système d'équations linéaires sur $\mathbb{F}$ est la donnée d'une matrice $\boldsymbol{A} \in \mathbb{F}^{m\times n}$ et d'un vecteur $\boldsymbol{b} \in \mathbb{F}^m$. Résoudre le système consiste à trouver l'ensemble des vecteurs $\boldsymbol{x} \in \mathbb{F}^n$ tels que $\boldsymbol{Ax} = \boldsymbol{b}$.

On rappelle le résultat fondamental décrivant la structure algébrique des solutions d'un système d'équations linéaires.

Proposition 2.2

L'ensemble des solutions d'un système d'équations linéaires $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ est
- *ou bien l'ensemble vide,*
- *ou bien un sous-espace affine $\mathcal{A}$ de $\mathbb{F}^n$, auquel cas $\mathcal{A}$ se décompose comme $\boldsymbol{x}^{(0)} + \mathcal{V}$ où $\boldsymbol{x}^{(0)}$ est une* solution particulière *du système et $\mathcal{V} = \ker \boldsymbol{A}$ est le noyau à droite de $\boldsymbol{A}$.*

Exemple 2.3

Sur $\mathbb{Q}$, on considère la matrice de taille 3×4 :

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

Si $\boldsymbol{b} = \begin{pmatrix} 4 \\ 1 \\ 3 \end{pmatrix}$, alors le système $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ admet une infinité de solutions. Ce sont les éléments de la forme

$$\begin{pmatrix} -18 \\ 11 \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} -2 \\ 1 \\ 1 \end{pmatrix}, \quad \lambda \in \mathbb{Q}.$$

Si $\boldsymbol{b} = \begin{pmatrix} 4 \\ 1 \\ 0 \end{pmatrix}$, alors il n'y a aucune solution au système $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

En général le nombre d'équations m et le nombre d'inconnues n ne sont pas égaux. Néanmoins, on peut se ramener au cas $n = m$ sans changer l'espace des solutions :

(i) en ajoutant $n - m$ lignes nulles à $\boldsymbol{A}$ et $n - m$ coefficients nuls à $\boldsymbol{b}$, dans le cas où $n > m$;
(ii) en ajoutant $m - n$ colonnes nulles et $\boldsymbol{A}$ et $m - n$ « inconnues virtuelles nulles », dans le cas où $n < m$.

Dans toute la suite, on supposera donc que $n = m$ et que la matrice $\boldsymbol{A}$ n'est pas nécessairement de rang plein. Notons qu'en pratique, il existe des algorithmes spécifiquement efficaces pour les cas où la matrice $\boldsymbol{A}$ est fortement déséquilibrée ($n \gg m$ ou $n \ll m$).

2.1.2 Le cas simple de la forme échelonnée

Étudions d'abord un cas simple, pour lequel la matrice $\boldsymbol{A} \in \mathbb{F}^{n \times n}$ a une structure facilitant la résolution du système.

Définition 2.4 (*Matrice sous forme échelonnée*)

On dit qu'une matrice $\boldsymbol{A} \in \mathbb{F}^{n \times n}$ est sous forme (triangulaire supérieure) échelonnée si
1. *$A_{2,1} = 0$,*
2. *$A_{i,j} = 0$ implique $A_{i+1,j+1} = A_{i+1,j} = 0$.*

Autrement dit, le nombre de zéros précédant la première valeur non-nulle d'une ligne augmente de ligne en ligne, jusqu'à ce qu'il ne reste éventuellement que des zéros. Une matrice triangulaire supérieure échelonnée a par exemple la forme suivante :

$$\boldsymbol{A} = \begin{pmatrix} \times & & & & \\ & \times & & * & \\ & & \times & & \\ & & & \times & \\ & & & & \times \\ & & \mathbf{0} & & \end{pmatrix}.$$

Apportons quelques précisions : la zone contenant $*$ est constituée d'éléments quelconques de $\mathbb{F}$, excepté pour les positions indiquées par le symbole $\times$ qui représente nécessairement un élément

non nul de $\mathbb{F}$. Ces derniers éléments sont appelés des *pivots*. On dit que la matrice est *échelonnée réduite* si ses pivots sont tous égaux à 1.

Calcul du noyau. Lorsque la matrice $\boldsymbol{A}$ est échelonnée, on peut déterminer le noyau à droite de $\boldsymbol{A}$ grâce à l'Algorithme 1. L'idée est de transformer $\boldsymbol{A}$ en une matrice diagonale $\boldsymbol{D}$ par des opérations sur ses colonnes, autrement dit de calculer une matrice triangulaire supérieure $\boldsymbol{T}$ telle que $\boldsymbol{AT} = \boldsymbol{D}$. Le noyau de $\boldsymbol{D}$ est alors aisé à calculer : ce sont les vecteurs $\boldsymbol{e}_i$ de la base canonique tels que $D_{i,i} = 0$. En déduit qu'une base du noyau de $\boldsymbol{A}$ est formée des vecteurs $\boldsymbol{Te}_i$ pour les même indices i.

Algorithme 1 : Calcul du noyau à droite d'une matrice échelonnée

Entrée : Une matrice $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ échelonnée.
Sortie : Une base du noyau à droite de $\boldsymbol{A}$.

$\boldsymbol{T} \leftarrow \boldsymbol{I}_n$
$\ell \leftarrow n+1$
Calculer $\ell = \max\{i \in [1,n], \boldsymbol{A}_{i,*} \neq \boldsymbol{0}\}$.
Pour tout *i allant de* 1 *à* ℓ **faire**
  Calculer s tel que le pivot est en position (i,s). **Pour tout** *j allant de* $s+1$ *à* n **faire**
    Calculer $\alpha \leftarrow A_{i,j}/A_{i,s}$
    Remplacer la colonne $\boldsymbol{T}_{*,j}$ par $\boldsymbol{T}_{*,j} - \alpha \boldsymbol{T}_{*,s}$.
Retourner les colonnes de $\boldsymbol{T}$ ne comportant pas de pivot.

Calcul d'une solution particulière. Ensuite, l'Algorithme 2 permet de calculer une solution particulière du système $\boldsymbol{Ax} = \boldsymbol{b}$. Dans les premières étapes (1–6), on traite les cas particuliers (aucune solution, solution triviale). Puis, l'étape itérative (7–10) permet de construire efficacement une solution particulière du système.

Algorithme 2 : Algorithme de résolution d'un système linéaire échelonné

Entrée : Une matrice $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ sous forme échelonnée, et un vecteur $\boldsymbol{b} \in \mathbb{F}^n$.
Sortie : Une solution particulière de $\boldsymbol{Ax} = \boldsymbol{b}$.

1 **Si** $\boldsymbol{A} = \boldsymbol{0}$ *et* $\boldsymbol{b} \neq \boldsymbol{0}$
2 **retourner** « aucune solution »
3 **Si** $\boldsymbol{A} = \boldsymbol{0}$ *et* $\boldsymbol{b} = \boldsymbol{0}$
4 **retourner** $\boldsymbol{0}$
5 Calculer $\ell = \max\{i \in [1,n], \boldsymbol{A}_{i,*} \neq \boldsymbol{0}\}$.
6 S'il existe $i > \ell$ tel que $b_i \neq 0$, alors **retourner** « aucune solution ».
7 Assigner $\boldsymbol{x} = (x_1, \ldots, x_n) \leftarrow \boldsymbol{0}$.
8 **Pour tout** *i allant de* ℓ *à* 1 **faire**
9 Calculer la position (i,s) du pivot.
10 Calculer $x_s \leftarrow \frac{1}{A_{i,s}}(b_i - \sum_{j=s+1}^{\ell} A_{i,j}x_j)$.
11 Retourner $\boldsymbol{x}$.

Remarque 2.5

On peut vérifier que la complexité de l'Algorithme 1 est en $O(n^3)$ opérations sur $\mathbb{F}$ tandis que celle de l'Algorithme 2 est en $O(n^2)$.

2.1.3 Algorithme d'élimination Gaussienne

Pour résoudre le problème dans le cas général, on cherche à transformer le système en un système sous forme échelonné. La méthode est connue sous le nom d'éliminiation Gaussienne

(voir Algorithme 3) ou d'algorithme de Gauss–Jordan si l'on obtient une matrice sous forme échelonnée réduite.

Algorithme 3 : Algorithme d'élimination Gaussienne

Entrée : Une matrice $\boldsymbol{A} \in \mathbb{F}^{n \times n}$
Sortie : Un système échelonné équivalent à $\boldsymbol{Ax} = \boldsymbol{b}$, c'est-à-dire trois matrices $\boldsymbol{U}$, $\boldsymbol{T}$ et $\boldsymbol{P}$ telles que $\boldsymbol{UAP} = \boldsymbol{T}$ et $\boldsymbol{T}$ est échelonnée.

$\boldsymbol{T} \leftarrow \boldsymbol{A}$, $\boldsymbol{U} \leftarrow \boldsymbol{I}_n$, $\boldsymbol{P} \leftarrow \boldsymbol{I}_n$,
$\ell \leftarrow n+1$, $i \leftarrow 1$
Tant que $i < \ell$ **faire**
  Rechercher un élément inversible (pivot) dans la i-ème ligne de $\boldsymbol{T}$.
  Si *aucun pivot n'a été trouvé :*
    $\ell \leftarrow \ell - 1$
    Échanger les lignes ℓ et i dans les matrices $\boldsymbol{T}$ et $\boldsymbol{U}$.
  Sinon
    Noter j la position du pivot.
    Échanger les colonnes i et j dans les matrices $\boldsymbol{T}$ et $\boldsymbol{P}$.
    Pour tout k *allant de* $i+1$ *à* n **faire**
      Calculer $\alpha = \frac{T_{k,i}}{T_{i,i}}$.
      Remplacer la ligne $\boldsymbol{T}_{k,*}$ de $\boldsymbol{T}$ par $\boldsymbol{T}_{k,*} - \alpha \boldsymbol{T}_{i,*}$.
      Remplacer la ligne $\boldsymbol{U}_{k,*}$ de $\boldsymbol{U}$ par $\boldsymbol{U}_{k,*} - \alpha \boldsymbol{U}_{i,*}$.
    $i \leftarrow i+1$.
Retourner $\boldsymbol{T}, \boldsymbol{U}, \boldsymbol{P}$.

On doit maintenant comprendre comment l'espace des solutions du système $\boldsymbol{Ax} = \boldsymbol{b}$ évolue par l'application de l'Algorithme 3. Pour cela, on introduit des matrices élémentaires qui représentent les transformations sur les lignes et les colonnes effectuées par l'élimination Gaussienne.

Définition 2.6

On définit les matrices élémentaires suivantes :
- *la matrice élémentaire générique*

$$\boldsymbol{M}_{i,j}\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) := \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & a & & b \\ & & & 1 & \\ & & & & 1 \\ & & c & & d \\ & & & & & 1 \end{bmatrix}$$

  où a est en position (i,i), b est en position (i,j), c est en position (j,i) et d est en position (j,j) ;
- *la matrice de permutation $\boldsymbol{P}_{i,j} = \boldsymbol{M}_{i,j}\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$;*
- *la matrice l'élimination $\boldsymbol{E}_{i,j}(\alpha) := M_{i,j}\left(\begin{smallmatrix} 1 & \alpha \\ 0 & 1 \end{smallmatrix}\right)$ pour $\alpha \in \mathbb{F}$.*

L'action à droite $\boldsymbol{A} \mapsto \boldsymbol{AP}_{i,j}$ d'une matrice de permutation élémentaire a pour effet d'échanger les colonnes i et j de $\boldsymbol{A}$. Similairement, son action à gauche de $\boldsymbol{A}$ permute ses lignes.

L'action à droite $\boldsymbol{A} \mapsto \boldsymbol{AE}_{i,j}(\alpha)$ remplace la j-ème colonne $\boldsymbol{A}_{*,j}$ de $\boldsymbol{A}$ par $\boldsymbol{A}_{*,j} + \alpha \boldsymbol{A}_{*,i}$. De même, la transformation $\boldsymbol{A} \mapsto \boldsymbol{E}_{i,j}(\alpha)\boldsymbol{A}$ remplace la i-ème ligne $\boldsymbol{A}_{i,*}$ de $\boldsymbol{A}$ par $\boldsymbol{A}_{i,*} + \alpha \boldsymbol{A}_{j,*}$.

On peut donc modéliser toutes les opérations effectuées dans l'Algorithme 3 par des produits impliquant des matrices élémentaires. Notons que sur $\boldsymbol{P}$, seuls des échanges de colonnes sont effectués ; ainsi $\boldsymbol{P}$ est une matrice de permutation (non-élémentaire) : chaque ligne et chaque colonne de $\boldsymbol{P}$ contient un unique coefficient non-nul égal à 1. En pratique, cela permet d'avoir un stockage très compact de la matrice $\boldsymbol{P}$.

Les résultats suivants sont élémentaires et vont permettre de décrire le lien entre les solutions des systèmes avant et après élimination gaussienne.

Lemme 2.7

Soit $\boldsymbol{P} \in \mathbb{F}^{n\times n}$ une matrice de permutation et $\boldsymbol{Ax} = \boldsymbol{b}$ un système linéaire. Alors, $\boldsymbol{x}^{(0)}$ est solution de $\boldsymbol{Ax} = \boldsymbol{b}$ si et seulement si $\boldsymbol{y}^{(0)} := \boldsymbol{P}^{-1}\boldsymbol{x}^{(0)}$ est solution du système $(\boldsymbol{AP})\boldsymbol{y} = \boldsymbol{b}$.

Lemme 2.8

Soit $\boldsymbol{U} \in \mathbb{F}^{n\times n}$ une matrice inversible et $\boldsymbol{Ax} = \boldsymbol{b}$ un système linéaire. Alors, $\boldsymbol{x}^{(0)}$ est solution de $\boldsymbol{Ax} = \boldsymbol{b}$ si et seulement si $\boldsymbol{x}^{(0)}$ est solution du système $(\boldsymbol{UA})\boldsymbol{x} = \boldsymbol{Ub}$.

Le Lemme 2.7 indique que la permutation des colonnes effectuée lors de l'étape 10 de l'Algorithme 3 se transcrit en une permutation inversée des coordonnées des solutions du système. Le Lemme 2.8 assure que les étapes 7 et 13 de l'Algorithme 3 préservent l'ensemble des solutions du système.

Donnons maintenant la méthode générique pour résoudre le système linéaire $\boldsymbol{Ax} = \boldsymbol{b}$:

1. Procéder à une élimination gaussienne, c'est-à-dire calculer $\boldsymbol{U}$, $\boldsymbol{T}$ et $\boldsymbol{P}$ tels que $\boldsymbol{UAP} = \boldsymbol{T}$.
2. Calculer une solution particulière $\boldsymbol{y}^{(0)}$ de $\boldsymbol{Ty} = \boldsymbol{Ub}$.
   Si elle n'existe pas, **terminer l'algorithme** avec « pas de solution ».
3. Calculer $\mathcal{K} = \ker \boldsymbol{T}$ le noyau à droite de $\boldsymbol{T}$.
4. **Retourner** une description de l'espace affine $\boldsymbol{P}(\boldsymbol{y}^{(0)} + \mathcal{K}) = \{\boldsymbol{P}(\boldsymbol{y}^{(0)} + \boldsymbol{u}), \boldsymbol{u} \in \mathcal{K}\}$.

En collectant tous les résultats précédents, on obtient un algorithme permettant de calculer les solutions de tout système d'équations linéaires.

Proposition 2.9

L'algorithme d'élimination Gaussienne, couplé à la résolution d'un système échelonné, permet de résoudre tout système linéaire carré $(n = m)$ sur $\mathbb{F}$ en $O(n^3)$ opérations élémentaires sur $\mathbb{F}$.

Démonstration : Pour la validité de la méthode, voir la discussion précédente.
Concernant sa complexité, on peut vérifier que les Algorithmes 1 et 3 sont les plus coûteux et nécessitent $O(n^3)$ opérations. En effet, par exemple l'étape 13 de l'Algorithme 3 est effectuée $O(n^2)$ fois (pour $i = 1, \ldots, n$ si $\boldsymbol{A}$ est de rang n, et pour $k = i+1, \ldots, n$), et elle coûte individuellement $O(n)$ opérations sur $\mathbb{F}$. Une analyse de complexité plus précise est laissée en exercice. ■

2.2 Suites récurrentes linéaires

Les suites récurrentes linéaires sont courantes en informatique. En cryptographie, on utilise notamment des LFSR (*linear feedback shift register*, registre à décalage rebouclé linéairement) pour engendrer des séquences de nombres pseudo-aléatoires. Dans la suite, nous allons également voir que ces suites permettent d'accélérer la résolution de systèmes linéaires dont les équations comportent peu de coefficients non-nuls.

2.2.1 Premières définitions

Définition 2.10 (*suite récurrente linéaire*)

Une suite $\boldsymbol{u} = (u_n)_{n\in\mathbb{N}} \in \mathbb{F}^{\mathbb{N}}$ est dite récurrente linéaire *s'il existe deux entiers $0 \leq D \leq L$ et D coefficients $(c_1, \ldots, c_D) \in \mathbb{F}^D$ tels que :*

$$\forall n \geq L, \quad u_n + \sum_{i=1}^{D} c_i u_{n-i} = 0\,. \tag{2.1}$$

Le plus petit D qui satisfait (2.1) *est appelé* ordre *de la suite. Les L premiers éléments $u_0, \dots, u_{L-1}$ de la suite sont appélés* termes initiaux.

Exemple 2.11

Sur $\mathbb{R}$, la suite $\boldsymbol{u} = (-1, 1, -1, 1, -1, 1, -1, 1, \dots)$ est récurrente linéaire. En effet, on peut l'écrire

$$u_0 = -1 \quad et \quad u_n + u_{n-1} = 0,\ \forall n \geq 1\,.$$

Son ordre est donc 1 (car seule la suite nulle à partir d'un certain rang a un ordre égal à 0).

Exemple 2.12

Sur le corps fini $\mathbb{F}_2$, la suite

$$\boldsymbol{v} = (1, \underbrace{0, 1, 1, 1, 0, 1, 0}_{\text{séquence pérodique}}, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0 \dots)$$

est récurrente linéaire. En effet, on peut l'écrire

$$u_0 = 1,\ u_1 = 0, u_2 = 1, u_3 = 1 \quad et \quad u_n + u_{n-1} + u_{n-3} = 0,\ \forall n \geq 4\,.$$

Son ordre est donc ≤ 3 (bien qu'il y ait 4 coefficients initiaux). On pourra vérifier par la suite que l'ordre de la suite est exactement 3.

Description par une série formelle. Toute suite $\boldsymbol{u} \in \mathbb{F}^{\mathbb{N}}$ peut être écrite comme une *série formelle*, en associant à $\boldsymbol{u}$ la série

$$U(X) \coloneqq \sum_{n\in\mathbb{N}} u_n X^n\,.$$

On note $\mathbb{F}[\![X]\!]$ l'algèbre des séries formelles en X. On rappelle que pour $U, V \in \mathbb{F}[\![X]\!]$, on définit la somme et le produit de séries comme :

$$(U+V)(X) \coloneqq \sum_{n\in\mathbb{N}} (u_n + v_n) X^n,$$

$$(U \cdot V)(X) \coloneqq \sum_{n\in\mathbb{N}} \left(\sum_{i=0}^{n} u_i v_{n-i} \right) X^n\,.$$

Observons qu'un polynôme $P \in \mathbb{F}[X]$ peut être vu comme une série formelle, dont les coefficients sont nuls à partir d'un certain rang.

Lemme 2.13

Soit $\boldsymbol{u} \in \mathbb{F}^{\mathbb{N}}$ une suite récurrente linéaire d'ordre D avec L termes initiaux. La série formelle $U \in \mathbb{F}[\![X]\!]$ de $\boldsymbol{u}$ satisfait $UP = Q$, où P et Q sont des polynômes tels que $\deg(P) \leq D$ et $\deg(Q) < L$.

Démonstration : Par définition de la suite $\boldsymbol{u}$, il existe des coefficients $c_1, \dots, c_d$ tels que

$$u_n + \sum_{i=1}^{d} c_i u_{n-1} = 0, \quad \forall n \geq L\,.$$

Posons $P(X) \coloneqq \sum_{i=0}^{d} c_i X^i$ avec $c_0 = 1$. Alors :

$$PU = \sum_{n\in\mathbb{N}} \left(\sum_{i=0}^{n} c_i u_{n-i} \right) X^n$$

et le coefficient de degré n de PU est donc nul pour tout $n \geq L$. Autrement dit, PU est un polynôme de degré $\leq L-1$. ∎

Le résultat précédent permet d'exprimer la série formelle d'une suite récurrente linéaire comme une fraction rationnelle $\frac{Q}{P} \in \mathbb{F}(X)$.

Exemple 2.14

Sur le corps $\mathbb{F}_3$, on considère la suite définie par

$$u_0 = 1,\ u_1 = 0,\ u_2 = 0,\ u_3 = 1, \qquad u_n = u_{n-1} + 2u_{n-2},\ \forall n \geq 4\,.$$

Les 20 premiers termes de cette suite sont :

$$(1,0,0,1,1,0,2,2,0,1,1,0,2,2,0,1,1,0,2,2,\dots)\,.$$

Le polynôme P associé à la suite est $P(X) = 1 - X - 2X^2 = 1 + 2X + X^2$, et on peut calculer

$$\begin{aligned} Q := PU &= c_0u_0 + (c_1u_0 + c_0u_1)X + (c_2u_0 + c_1u_1 + c_0u_2)X^2 + (c_2u_1 + c_1u_2 + c_0u_3)X^3 \\ &= 1 + 2X + 2X^2 + X^3\,. \end{aligned}$$

Puis,

$$U(X) = \frac{Q}{P}(X) = \frac{1 + 2X + X^2 + X^3}{1 + 2X + X^2}$$

On peut ensuite vérifier que le développement de la fraction rationnelle $\frac{Q}{P}$ en 0 donne les coefficients de la suite $(u_n)_{n\in\mathbb{N}}$.

Définition 2.15

Dans l'écriture $UP = Q$ du Lemme 2.13, le polynôme P est appelé polynôme de connexion *de* $\boldsymbol{u}$.

Lemme 2.16

L'ensemble des polynômes de connexion d'une suite $\boldsymbol{u}$ forme un idéal de $\mathbb{F}[X]$. En particulier, il existe un polynôme de connexion P de degré minimal, tel que $P(0) = 1$ et tel que tout polynôme de connexion est un multiple de P. Ce polynôme est appelé polynôme de connexion minimal *de la suite* $\boldsymbol{u}$.

Démonstration : En exercice. ■

Remarque 2.17

Réciproquement, toute fraction rationnelle $F(X) = \frac{Q(X)}{P(X)} \in \mathbb{F}(X)$ avec $P(0) = 1$ définit une suite récurrente linéaire $\boldsymbol{u}$. Le polynôme P définit la relation de récurrence de la suite (c'est un polynôme de connexion), et il ensuite reste à calculer les termes initiaux. Pour cela, on définit

$$F_0(X) := F(X) \quad \text{et} \quad F_{i+1}(X) = \frac{F_i(X) - F_i(0)}{X} \ \text{ pour tout } i \geq 0\,.$$

Alors, les $L := \deg(Q) - 1$ premiers termes de la suite sont $F_0(0), \dots, F_{L-1}(0)$.

Dans la suite de cette section, notre objectif va être de répondre aux questions suivantes :

1. Étant donnée une suite récurrente $\boldsymbol{u}$, comment retrouver efficacement son polynôme de connexion minimal ?
2. Combien de termes de la suite $\boldsymbol{u}$ sont-ils nécessaires pour obtenir ce polynôme ?

Nous verrons en Section 2.2.3 que l'algorithme de Berlekamp–Massey est une méthode itérative permettant de répondre à ces questions. L'idée est (essentiellement) de construire itérativement les polynômes de connexion minimaux de sous-suites tronquées de $\boldsymbol{u}$.

Avant de décrire cet algorithme, nous avons besoin d'introduire le concept de registre à décalage, aussi couramment utilisé en informatique.

2.2.2 Registres à décalage

Les registres à décalage permettent de représenter un flux de données engendré par une suite récurrente linéaire. En voici une définition formelle.

Définition 2.18 (*LFSR*)

Un LFSR (linear feedback shift register) est la donnée d'un polynôme $P \in \mathbb{F}[X]$ tel que $P(0) = 1$, et d'un entier $L \geq \deg(P)$. L'entier L est appelé la dimension *du LFSR.*

Usuellement, les LFSR sont présentés dans le cas binaire ($\mathbb{F} = \mathbb{F}_2$) ou plus rarement dans le cas où $\mathbb{F}$ est un corps fini quelconque.

Définition 2.19 (*suite engendrée par un LFSR*)

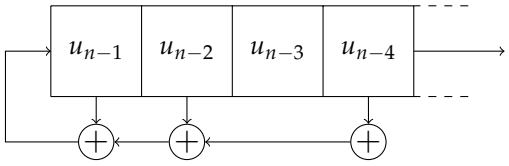
Soit $\mathcal{L} = (P, L)$ un LFSR, avec $P(X) = 1 + \sum_{i=1}^{D} c_i X^i$. La suite engendrée par $\mathcal{L}$ sur le registre initial *$(u_0, \ldots, u_{L-1}) \in \mathbb{F}^L$ est la suite :*

- *dont les L premiers termes sont $u_0, \ldots, u_{L-1}$,*
- *dont les termes suivants sont définis par la relation de récurrence*

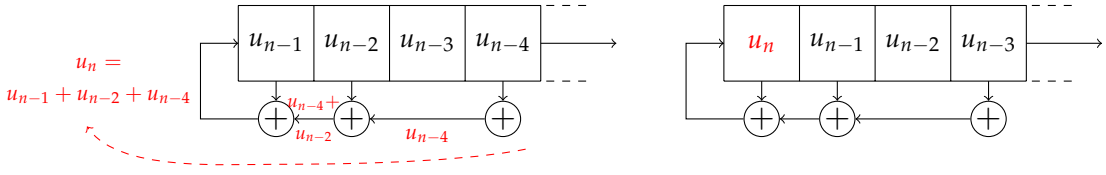
$$u_n + \sum_{i=1}^{D} c_i u_{n-i} = 0, \quad \forall n \geq L.$$

Remarquons que si (P,L) est un polynôme de connexion qui engendre une suite $\boldsymbol{u}$, alors P est un polynôme de connexion de $\boldsymbol{u}$.

Représentation « circuit » d'un LFSR sur $\mathbb{F}_2$. Sur le corps fini $\mathbb{F}_2$, on utilise souvent une représentation des LFSR sous forme de circuit. Par exemple, le LFSR $(1 + X + X^2 + X^4, 4)$ est représenté comme suit :



L'idée de la représentation est la suivante : au polynôme $P(X) = 1 + X + X^2 + X^4$ est associé l'équation de récurrence $u_n = u_{n-1} + u_{n-2} + u_{n-4}$ pour tout $n \geq 4$. À chaque pas de temps n, quatre bits $u_{n-1}, u_{n-2}, u_{n-3}, u_{n-4}$ sont stockés dans le *registre*. Pour créer, le nouveau bit u_n, on effectue la somme des bits précédents $u_{n-1}, u_{n-2}, u_{n-4}$ (cette somme est représentée dans le circuit par les portes $\oplus$), puis on décale le registre d'un cran vers la droite.



état à la fin de l'étape n — état au début de l'étape $n+1$

Dans l'exemple suivant, on peut observer qu'une suite peut être engendrée par des LFSR de polynômes différents et de dimensions différentes.

Exemple 2.20

La suite binaire $\boldsymbol{v} = (1010101010....) \in \mathbb{F}_2^{\mathbb{N}}$ *peut être vue comme*

$$v_0 = 1,\ v_1 = 0, \qquad v_n = v_{n-2} \quad \forall n \geq 2$$

ou comme

$$v_0 = 1,\ v_1 = 0,\ v_2 = 1, \qquad v_n = v_{n-1} + v_{n-2} + v_{n-3} \quad \forall n \geq 3$$

Dans le premier cas, la suite $\boldsymbol{v}$ *est engendrée par le LFSR* $(1 + X^2, 2)$ *avec le registre initial* $(1, 0)$*. Dans le second, elle est engendrée par le LFSR* $(1 + X + X^2 + X^3, 3)$ *avec le registre* $(1, 0, 1)$*.*

Suites tronquées. Pour les applications pratiques, nous n'avons pas accès à tous les termes de la suite, mais seulement aux premiers termes. Les suites rencontrées sont donc « tronquées » à un certain rang. Pour une suite $\boldsymbol{u} \in \mathbb{F}^{\mathbb{N}}$ et un entier $k \geq 1$, on note

$$T_k(\boldsymbol{u}) \coloneqq (u_0, u_1, \ldots, u_{k-1}, 0, 0, \ldots, 0, \ldots) \in \mathbb{F}^{\mathbb{N}}$$

la suite où l'on remplace les termes d'indice $\geq k$ par 0. Cette suite sera souvent assimiliée au k-uplet de ses k permiers termes.

Remarque 2.21

Si $\boldsymbol{u}$ *est une suite récurrente linéaire de série formelle* $U \in \mathbb{F}[\![X]\!]$*, alors la série formelle de* $T_k(\boldsymbol{u})$ *est le polynôme obtenu par la division euclienne de* U *par* X^k*.*

Par la suite, on notera $\mathrm{rem}(A,B)$ le reste de la division euclidienne du polynôme A par le polynôme B.

Définition 2.22 (*suite engendrée sur un nombre fini de termes*)

Soient $\mathcal{L} = (P, L)$ *un LFSR, avec* $P(X) = 1 + \sum_{i=1}^{D} c_i X^i$*, et* $N > L$*. On dit que* $\mathcal{L}$ *engendre une suite* $\boldsymbol{u} \in \mathbb{F}^N$ *sur* N *termes si*

$$u_n + \sum_{i=1}^{D} c_i u_{n-i} = 0, \quad \forall L \leq n < N.$$

Lemme 2.23

Soit $\boldsymbol{u}$ *une suite linéaire récurrente de série formelle* U *et* $N \geq 1$*. Un LFSR* (P, L) *engendre* $\boldsymbol{u}$ *sur* N *termes si et seulement si le reste de la division euclidienne du polynôme* PU *par* X^N *est de degré* $< L$*.*

Démonstration : Notons $P(X) = \sum_{i=0}^{D} c_i X^i$ et effectuons la division euclidienne du polynôme PU par X^N : on a $PU = R + QX^N$ avec $\deg(R) < N$. Plus précisément,

$$R = \sum_{n=0}^{N-1} \left(\sum_{i=0}^{n} c_i u_{n-i} \right) X^n .$$

Si (P,L) engendre $\boldsymbol{u}$ sur N termes, alors pour tout $L \leq n < N$ on a $u_n + \sum_{i=1}^{D} c_i u_{n-i} = 0$. Autrement dit, les coefficients des monômes de degré $\geq L$ de R sont nuls, donc R est de degré $< L$.
Réciproquement, si $\sum_{i+j=n} c_i u_{n-i} = 0$ pour tout $L \leq n < N$, alors on observe que (P, L) est bien un LFSR qui engendre $\boldsymbol{u}$ sur N termes : les L premiers termes peuvent être fixés à u_0, u_1, …, u_{L-1}, et les suivant satisfont la relation de récurrence donnée par P. ∎

Définition 2.24

On note $\mathcal{R}(\boldsymbol{u})$ *l'ensemble des LFSR qui engendrent une suite récurrente linéaire* $\boldsymbol{u}$*, et* $\mathcal{R}_k(\boldsymbol{u})$ *l'ensemble des LFSR qui engendrent* $\boldsymbol{u}$ *sur* k *termes.*

Lemme 2.25

Soit $\boldsymbol{u}$ une suite récurrente linéaire. Alors, la suite $(\mathcal{R}_k(\boldsymbol{u}))_{k\in\mathbb{N}}$ est décroissante pour la relation d'inclusion, et constante à partir d'un certain rang.

Démonstration : Laissée en exercice. ■

Définition 2.26 (*complexité linéaire, profil de complexité*)

Soit $\boldsymbol{v} \in \mathbb{F}^{\mathbb{N}}$ une suite quelconque et $n \geq 1$ un entier. La complexité linéaire *de $\boldsymbol{v}$ à l'ordre n est la plus petite dimension L d'un LFSR (P, L) qui engendre $\boldsymbol{v}$ sur n termes. On la note*

$$\ell_n(\boldsymbol{v}) := \min\{L, \exists P \in \mathbb{F}[X], (P, L) \in \mathcal{R}_n(\boldsymbol{u})\}.$$

Par convention on pose $\ell_0(\boldsymbol{v}) := 0$. Le profil de complexité linéaire *de $\boldsymbol{u}$ est la suite $\boldsymbol{\ell} := (\ell_k)_{k\in\mathbb{N}}$.*

On dit également que (P, L) est un LFSR minimal sur k termes si $(P, L) \in \mathcal{R}_k(\boldsymbol{u})$ et $L = \ell_k(\boldsymbol{u})$.

Exemple 2.27

Soit $\boldsymbol{u} \in \mathbb{F}^{\mathbb{N}}$ une suite commençant par k zéros et telle que $u_k = 1$. Alors, on a $\ell_0(\boldsymbol{u}) = 0$ par convention, puis $\ell_i(\boldsymbol{u}) = 0$ pour tout $i \in \{1, \dots, k\}$. En effet, en posant $P_i = 1$, on obtient la relation de récurrence linéaire d'ordre 0 donnée par $u_i = 0$.
Puis, on a ensuite $\ell_{k+1}(\boldsymbol{u}) = k + 1$, une nouvelle fois avec $P_k = 1$ (par exemple), car $u_k \neq 0$ ne peut pas être calculé comme une combinaison linéaire de termes nuls.

Exemple 2.28

On considère la suite binaire $\boldsymbol{u}$ définie par ses premiers termes

$$\boldsymbol{u} = (1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, \dots).$$

Cherchons à déterminer les premiers termes du profil de complexité linéaire.
- *Ordre 0 : Par convention $\ell_0 = 0$.*
- *Ordre 1 : On a $\ell_1 = 1$, car il faut définir le premier terme de la suite.*
- *Ordre 2 : Pour le calcul de ℓ_2, on s'intéresse aux deux premiers termes $(u_0, u_1) = (1, 1)$. On observe qu'ils sont égaux, donc il suffit d'un seul terme initial $u_0 = 1$. On a donc $\ell_2 = 1$ et le terme suivant u_1 est engendré par le polynôme $P = 1 + X$.*
- *Ordre 3 : Pour le calcul de ℓ_3, on a $(u_0, u_1, u_2) = (1, 1, 1)$, donc le cas précédent se répète : $\ell_3 = 1$ avec le registre $u_0 = 1$ et le polynôme $P = 1 + X$.*
- *Ordre 4 : Le calcul de ℓ_4 diffère car on a $u_3 = 0 \neq u_2$. D'abord, notons qu'aucune relation de récurrence d'ordre 1 ne peut exprimer u_3 en fonction de u_2, il faut donc chercher une relation d'ordre au moins 2. On remarque que $P = 1 + X + X^2$ convient avec un registre de taille 3. Peut-on trouver un registre initial plus petit ? La réponse est non. En effet, il est impossible d'avoir un registre de taille 2 car les seules relations liant u_2 à u_1 et u_0 proviennent des polynômes $1 + X$ et $1 + X^2$, et aucun de ces polynômes ne permet d'engendrer u_3 à partir de u_2 et u_1. On a donc $\ell_4 = 3$.*

En poursuivant le calcul, on obtient les valeurs suivantes (les polynômes P n'étant pas nécessairement uniques) :

$\ell_0 = 0 \quad (P, L) = (1, 0)$ *par convention*	$\ell_4 = 3 \quad (P, L) = (1 + X + X^2, 3)$
$\ell_1 = 1 \quad (P, L) = (1, 1)$	$\ell_5 = 3 \quad (P, L) = (1 + X + X^2, 3)$
$\ell_2 = 1 \quad (P, L) = (1 + X, 1)$	$\ell_6 = 3 \quad (P, L) = (1 + X + X^2, 3)$
$\ell_3 = 1 \quad (P, L) = (1 + X, 1)$	$\ell_7 = 3 \quad (P, L) = (1 + X + X^2, 3)$

2.2.3 Algorithme de Berlekamp–Massey

Rappelons que nous cherchons à calculer le polynôme de connexion minimal d'une suite récurrente linéaire $\boldsymbol{u} \in \mathbb{F}^{\mathbb{N}}$. L'algorithme de Berlekamp–Massey permet d'obtenir ce polynôme par une méthode itérative : plus précisément, l'algorithme calcule une suite de LFSR $((P_k, L_k))_{k\in\mathbb{N}}$ telle que (P_k, L_k) est de dimension minimale parmi $\mathcal{R}_k(\boldsymbol{u})$.

Avant de décrire l'algorithme, présentons quelques résultats théoriques.

Proposition 2.29

Soit $\mathcal{L} = (P, L)$ et $\mathcal{L}' = (P', L')$ deux LFSR et $k \geq L + L'$. Supposons que $\mathcal{L}$ et $\mathcal{L}'$ engendrent une même suite récurrente linéaire $\boldsymbol{u}$ sur k termes. Alors, il existe une suite récurrente $\boldsymbol{v}$ telle que $\mathcal{L}$ et $\mathcal{L}'$ engendrent $\boldsymbol{v}$.

Démonstration : Soit U le polynôme associé à la suite tronquée $T_k(\boldsymbol{u})$. D'après le Lemme 2.23 on a

$$PU = R + X^k Q \quad \text{et} \quad P'U = R' + X^k Q',$$

avec $\deg(R) < L$ et $\deg(R') < L'$. L'égalité

$$PP'U = RP' + X^k QP' = R'P + X^k Q'P$$

permet de déduire que $RP' - R'P = X^k(Q'P - QP')$. Remarquons maintenant que

$$\deg(RP' - R'P) \leq \max\{\deg(R) + \deg(P'), \deg(R') + \deg(P)\} < L + L' \leq \deg(X^k)\,.$$

Nécessairement, on a donc $RP' - R'P = 0$. Il suffit maintenant de définir $\boldsymbol{v}$ comme la suite linéaire récurrente de série formelle $V = \frac{R}{P} = \frac{R'}{P'}$ (voir Remarque 2.17). Comme $PV = R$ et $P'V = R'$, les deux LFSR (P, L) et (P', L') engendrent bien $\boldsymbol{v}$. ∎

Corollaire 2.30

Si $(P, L) \in \mathcal{R}_k(\boldsymbol{u}) \setminus \mathcal{R}_{k+1}(\boldsymbol{u})$, alors pour tout $(P', L') \in \mathcal{R}_{k+1}(\boldsymbol{u})$, on a $L' \geq k + 1 - L$. En particulier, si $\ell_k(\boldsymbol{u}) \neq \ell_{k+1}(\boldsymbol{u})$, alors $\ell_{k+1}(\boldsymbol{u}) \geq k + 1 - \ell_k(\boldsymbol{u})$.

Démonstration : C'est la contraposée de la Proposition 2.29. ∎

La proposition suivante montre qu'on peut maintenant construire explicitement un LFSR engendrant $k + 1$ termes de $\boldsymbol{u}$ à partir de certains LFSR engendrant strictement moins de termes.

Proposition 2.31

Soient $\boldsymbol{u} \in \mathbb{F}^{\mathbb{N}}$ une suite et $U(X) \in \mathbb{F}[\![X]\!]$ sa série formelle associée. Soient également $(P, L) \in \mathcal{R}_k(\boldsymbol{u}) \setminus \mathcal{R}_{k+1}(\boldsymbol{u})$ et $(P', L') \in \mathcal{R}_{k'}(\boldsymbol{u}) \setminus \mathcal{R}_{k'+1}(\boldsymbol{u})$ où $k' < k$. On note α le coefficient de degré k de UP et α' le coefficient de degré k' de UP'. Alors, le LFSR donné par

$$\left(P - \frac{\alpha}{\alpha'} X^{k-k'} P' \;,\; \max\{L, L' + k - k'\}\right)$$

est dans $\mathcal{R}_{k+1}(\boldsymbol{u})$.

Démonstration : Effectuons les divisions euclidiennes de UP par X^{k+1} et UP' par $X^{k'+1}$. Par définition de P et P' et d'après le Lemme 2.23, on obtient :

$$UP = Q + \alpha X^k + X^{k+1} R \quad \text{et} \quad UP' = Q' + \alpha' X^{k'} R'$$

avec $\deg Q < L$ et $\deg Q' < L'$. Par conséquent :

$$U(P - \frac{\alpha}{\alpha'} X^{k-k'} P') = Q - \frac{\alpha}{\alpha'} X^{k-k'} Q' + X^{k+1}(R - \frac{\alpha}{\alpha'} R').$$

Notons que le degré de $Q - \frac{\alpha}{\alpha'} X^{k-k'} Q'$ est plus petit que $\max\{\deg Q, \deg Q' + k - k'\} < \max\{L, L' + k - k'\}$. En utilisant une nouvelle fois le Lemme 2.23, on obtient le résultat escompté. ■

Comme corollaire, on obtient une forme générale pour le profil de complexité linéaire.

Corollaire 2.32

Si $\ell_k(\boldsymbol{u}) < \ell_{k+1}(\boldsymbol{u})$, alors $\ell_{k+1}(\boldsymbol{u}) = k + 1 - \ell_k(\boldsymbol{u})$.

Démonstration : On a montré que $\ell_{k+1}(\boldsymbol{u}) \geq k + 1 - \ell_k(\boldsymbol{u})$ dans le Corollaire 2.30. Montrons l'inégalité inverse par récurrence.

• **Initialisation.** Soit i le premier indice pour lequel $u_i \neq 0$. Par convention on a posé $\ell_0(\boldsymbol{u}) = 0$, et on observe que $\ell_1(\boldsymbol{u}) = \cdots = \ell_i(\boldsymbol{u}) = 1$, puis nécessairement $\ell_{i+1} = i$ (voir Exemple 2.27). Par conséquent on a bien $\ell_0(\boldsymbol{u}) + \ell_1(\boldsymbol{u}) = 1$ d'une part, et $\ell_{i+1}(\boldsymbol{u}) + \ell_i(\boldsymbol{u}) = i + 1$ d'autre part.

• **Induction.** Soient k et k' tels que

$$\ell_{k'}(\boldsymbol{u}) < \ell_{k'+1}(\boldsymbol{u}) = \cdots = \ell_k(\boldsymbol{u}) < \ell_{k+1}(\boldsymbol{u}) \,.$$

Par hypothèse de récurrence, on a $\ell_{k'}(\boldsymbol{u}) = k' + 1 - \ell_{k'+1}(\boldsymbol{u}) = k' + 1 - \ell_k(\boldsymbol{u})$. En utilisant la proposition précédente, on peut construire un LFSR qui engendre $\boldsymbol{u}$ sur $k + 1$ termes, et dont la dimension est $\leq \max\{\ell_k(\boldsymbol{u}), \ell_{k'}(\boldsymbol{u}) + k - k'\} = \ell_{k'}(\boldsymbol{u}) + k - k'$. Par conséquent

$$\ell_{k+1}(\boldsymbol{u}) \leq \ell_{k'}(\boldsymbol{u}) + k - k' = k + 1 - \ell_k(\boldsymbol{u}) \,.$$

■

Par conséquent, le nouveau LFSR construit dans la Proposition 2.31 est de dimension minimale si les LFSR (P,L) et (P',L') qui permettent de le construire sont eux-même de dimension minimale. On en déduit ainsi l'Algorithme 4, nommé algorithme de Berlekamp–Massey, qui permet de construire une séquence de LFSR de dimension minimale engendrant la suite $\boldsymbol{u}$ sur un nombre de termes incrémental.

Algorithme 4 : Algorithme de Berlekamp–Massey

Entrée : Le polynôme $U(X) = \sum_{i=0}^{N-1} u_i X^i \in \mathbb{F}[X]$ correspondant à la suite tronquée $T_N(\boldsymbol{u})$.
Sortie : Une séquence de LFSR $((P_k, L_k))_{0 \leq k < N}$ telle que $(P_k, L_k) \in \mathcal{R}_k(\boldsymbol{u})$ et $L_k = \ell_k(\boldsymbol{u})$

$i \leftarrow 0$, $P_0 \leftarrow 1$, $L_0 \leftarrow 0$
Tant que $u_i = 0$ **faire**
  $i \leftarrow i + 1$
  $P_i = 1, \quad L_i = 0$
$i \leftarrow i + 1$
$P_i \leftarrow 1, \quad L_i \leftarrow i, \quad k' \leftarrow i - 1, \quad \alpha' \leftarrow u_{i-1}$
Pour tout $k \in \{i, \ldots, N-1\}$ **faire**
  $\alpha \leftarrow$ coefficient de degré k du polynôme produit UP_k
  Si $\alpha = 0$ ▷ *Le polynôme P_k engendre toujours $\boldsymbol{u}$ sur $k+1$ termes*
    $P_{k+1} \leftarrow P_k$
    $L_{k+1} \leftarrow L_k$
  Sinon
    $P_{k+1} \leftarrow P_k - \frac{\alpha}{\alpha'} X^{k-k'} P_{k'}$ ▷ *Le polynôme P_k n'engendre plus $\boldsymbol{u}$ au $(k+1)$-ème terme*
    Si $L_k > k/2$
      $L_{k+1} \leftarrow L_k$
    Sinon
      $L_{k+1} \leftarrow k + 1 - L_k$
      $k' \leftarrow k$
      $\alpha' \leftarrow \alpha$
Retourner la séquence $((P_k, L_k))$.

L'idée de l'algorithme de Berlekamp–Massey est la suivante. On garde tout au long de l'algorithme la donnée des LFSR minimaux précédemment calculés (les premiers étant par convention $(P = 1, L = 0)$). Puis, pour des valeurs de k croissantes, on calcule de nouveaux LFSR qui engendrent la suite sur k termes.

Deux cas se présentent. Ou bien le dernier LFSR calculé engendre toujours la suite jusqu'au rang k (lignes 9 à 11), auquel cas le LFSR minimal d'indice k est égal au dernier LFSR calculé. Ou bien il n'engendre plus la suite pour le k-ème terme (lignes 12 à 19) ; il faut alors construire un nouveau LFSR minimal au moyen de la Proposition 2.31.

Notons que l'assignation $L_{k+1} \leftarrow L_k$ de l'étape 15 se justifie par le résultat suivant.

Lemme 2.33

Si $\ell_k(\boldsymbol{u}) > k/2$, alors $\ell_{k+1}(\boldsymbol{u}) = \ell_k(\boldsymbol{u})$.

Démonstration : On montre la contraposée. Supposons $\ell_{k+1}(\boldsymbol{u}) \neq \ell_k(\boldsymbol{u})$. Alors $\ell_k(\boldsymbol{u}) = k + 1 - \ell_{k+1}(\boldsymbol{u}) \leq k - \ell_k(\boldsymbol{u})$ par croissance de la suite $(\ell_k(\boldsymbol{u}))_{k\in\mathbb{N}}$. Par conséquent, $\ell_k(\boldsymbol{u}) \leq k/2$. ■

Exemple 2.34

On considère une suite binaire dont les 11 premiers termes sont :

$$(0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0)$$

L'algorithme de Berlekamp–Massey produit alors la séquence de LFSR suivante :

k	P_k	$\ell_k(\boldsymbol{u})$
0	1	0
1	1	0
2	1	0
3	1	3
4	1	3
5	1	3

k	P_k	$\ell_k(\boldsymbol{u})$
6	$X^3 + 1$	3
7	$X^3 + 1$	3
8	$X^5 + X^3 + 1$	5
9	$X^5 + X^3 + 1$	5
10	$X^3 + X^2 + 1$	5
11	$X^3 + X^2 + 1$	5

2.2.4 Suites récurrentes vectorielles

Dans cette sous-section, on étend les résultats précédents au cas où les termes des suites sont des éléments d'un espace vectoriel plutôt que d'un corps. On appelle les suites obtenues des *suites vectorielles*.

Définition 2.35 (*Suite de vecteurs récurrente linéaire*)

Soit $n \geq 1$. Une suite de vecteurs $\boldsymbol{v} = (\boldsymbol{v}_k)_{k\in\mathbb{N}} \in (\mathbb{F}^n)^{\mathbb{N}}$ est dite récurrente linéaire *s'il existe des entiers $0 \leq D \leq L$ et des éléments $c_0, \ldots, c_D \in \mathbb{F}$ avec $c_0 \neq 0$, $c_d \neq 0$, tels que*

$$c_0\boldsymbol{v}_k + c_1\boldsymbol{v}_{k-1} + \cdots + c_D\boldsymbol{v}_{k-D} = \mathbf{0}, \quad \forall k \geq L\,.$$

Le polynôme $P(X) = c_0 + c_1X + \cdots + c_DX^D \in \mathbb{F}[X]$ est appelé polynôme de connexion *de la suite.*

Si $n = 1$, on dit que la suite est *scalaire* et on retrouve la définition des suites récurrentes linéaires vue dans la section précédente.

Remarque 2.36

Soit $\boldsymbol{v} = (\boldsymbol{v}_k)_{k\in\mathbb{N}} \in (\mathbb{F}^n)^{\mathbb{N}}$ une suite vectorielle récurrente linéaire. Pour tout $j \in \{1, \ldots, n\}$, on note $\boldsymbol{v}^{(j)} \in \mathbb{F}^{\mathbb{N}}$ la suite scalaire constituée des j-ème coordonnées des vecteurs $\boldsymbol{v}_k \in \mathbb{F}^n$. Alors la suite $\boldsymbol{v}^{(j)}$ est une suite récurrente linéaire.

Exemple 2.37

Pour $n = 2$, on considère la suite définie par $\boldsymbol{v}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ et par

$$\begin{pmatrix} v_k^{(1)} \\ v_k^{(2)} \end{pmatrix} = \begin{pmatrix} v_{k-1}^{(2)} \\ v_{k-1}^{(1)} \end{pmatrix}, \quad \forall k \geq 1 .$$

On peut l'écrire

$$\boldsymbol{v}_k = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \boldsymbol{v}_{k-1}, \quad \forall k \geq 1 .$$

Elle a pour polynôme de connexion $X^2 + 1$, car $v_k^{(1)} = -v_{k-2}^{(1)}$ et $v_k^{(2)} = -v_{k-2}^{(2)}$ pour tout $k \geq 1$.

Lemme 2.38

L'ensemble $\mathcal{I}_{\boldsymbol{v}}$ des polynômes annulateurs d'une suite vectorielle récurrente linéaire $\boldsymbol{v} \in (\mathbb{F}^n)^{\mathbb{N}}$ forme un idéal de $\mathbb{F}[X]$.

Démonstration : Si $P, Q \in \mathcal{I}_{\boldsymbol{v}}$, alors on peut vérifier que pour tout $\lambda \in \mathbb{F}$, les polynômes $\lambda P(X)$, $P(X) + Q(X)$, et $XP(X)$ sont également dans $\mathcal{I}_{\boldsymbol{v}}$:
- $\sum_{i=0}^{D} \lambda p_i \boldsymbol{v}_{k-i} = \lambda \sum_{i=0}^{D} p_i \boldsymbol{v}_{k-i} = \boldsymbol{0}$,
- $\sum_{i=0}^{D} (p_i + q_i) \boldsymbol{v}_{k-i} = \sum_{i=0}^{D} p_i \boldsymbol{v}_{k-i} + \sum_{i=0}^{D} q_i \boldsymbol{v}_{k-i} = \boldsymbol{0}$,
- comme $XP(X) = \sum_{i=1}^{D+1} p_{i-1} X^i$, on a $\sum_{i=1}^{D+1} p_{i-1} \boldsymbol{v}_{k+1-i} = \sum_{i=0}^{D} p_i \boldsymbol{v}_{k-i} = \boldsymbol{0}$ pour $k \geq L$. ■

Comme $\mathbb{F}[X]$ est euclidien (donc principal), on peut donner un sens à un polynôme de connexion minimal.

Définition 2.39

Le polynôme (de connexion) minimal d'une suite vectorielle récurrente linéaire $\boldsymbol{v} \in (\mathbb{F}^n)^{\mathbb{N}}$ est l'unique polynôme de connexion de $\boldsymbol{v}$ unitaire et de plus petit degré. C'est un générateur de $\mathcal{I}_{\boldsymbol{v}}$, que l'on note $P_{\boldsymbol{v}}(X)$.

Comme vu précédemment pour le cas scalaire, on montre aisément que $P_{\boldsymbol{v}}(0) \neq 0$ car sinon $P_{\boldsymbol{v}}(X)/X \in \mathcal{I}_{\boldsymbol{v}}$ en étant de degré strictement inférieur à $P_{\boldsymbol{v}}(X)$.

On peut relier le polynôme de connexion minimal d'une suite vectorielle récurrente linéaire, à ceux de ces suites « coordonnées ».

Lemme 2.40

Soit $\boldsymbol{v} \in (\mathbb{F}^n)^{\mathbb{N}}$ une suite vectorielle linéaire récurrente et $P_{\boldsymbol{v}^{(i)}}(X)$ le polynôme de connexion minimal de $\boldsymbol{v}^{(i)}$, pour tout $i \in \{1, \ldots, n\}$. Alors, le polynôme de connexion minimal de $\boldsymbol{v}$ est

$$P_{\boldsymbol{v}}(X) = \mathrm{ppcm}(P_{\boldsymbol{v}^{(1)}}(X), \ldots, P_{\boldsymbol{v}^{(n)}}(X)) .$$

Démonstration : Pour obtenir ce résultat, il suffit de démontrer que

$$\mathcal{I}_{\boldsymbol{v}} = \mathcal{I}_{\boldsymbol{v}^{(1)}} \cap \cdots \cap \mathcal{I}_{\boldsymbol{v}^{(n)}} .$$

En effet, l'intersection $\mathcal{I}_{\boldsymbol{v}^{(1)}} \cap \cdots \cap \mathcal{I}_{\boldsymbol{v}^{(n)}}$ est un idéal de générateur $A(X) = \mathrm{ppcm}(P_{\boldsymbol{v}^{(1)}}(X), \ldots, P_{\boldsymbol{v}^{(n)}}(X))$.
En écrivant l'équation $\sum_{i=0}^{D} p_i \boldsymbol{v}_{n-i}$ sur chacune des coordonnées des vecteurs, il est clair que tout polynôme de connexion $P(X)$ de $\boldsymbol{v}$ est également une polynôme de connexion de $\boldsymbol{v}^{(j)}$, pour tout $j \in \{1, \ldots, n\}$.
Réciproquement, tout polynôme $Q(X)$ qui est un polynôme de connexion de chacune des suites $\boldsymbol{v}^{(j)}$ est également un polynôme de connexion de $\boldsymbol{v}$: l'équation $\sum_{i=0}^{D} p_i \boldsymbol{v}_{n-i}$ sera vérifiée dès lors que n est suffisamment grand. ■

Remarque 2.41

On peut étendre les résultats précédents en remplaçant les suites $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)} \in \mathbb{F}^{\mathbb{N}}$ par les suites issues de produits scalaires $\langle \boldsymbol{v}, \boldsymbol{w}_1 \rangle, \ldots, \langle \boldsymbol{v}, \boldsymbol{w}_n \rangle \in \mathbb{F}^{\mathbb{N}}$, où $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$ sont des vecteurs fixes de $\mathbb{F}^n$ qui en forment une base. La preuve est laissée en exercice.

Suites itérées. Une manière de construire des suites vectorielles récurrentes linéaires et d'appliquer successivement une matrice $\boldsymbol{A} \in \mathbb{F}^{n \times n}$ à un vecteur initial $\boldsymbol{b} \in \mathbb{F}^n$. On obtient une suite $\boldsymbol{v} = (\boldsymbol{A}^k \boldsymbol{b})_{k \in \mathbb{N}} \in (\mathbb{F}^n)^{\mathbb{N}}$ qu'on appelle *suite itérée* de matrice $\boldsymbol{A}$ et de vecteur initial $\boldsymbol{b}$.

On va voir que le polynôme de connexion minimal de la suite $\boldsymbol{v}$ est naturellement lié au polynôme annulateur de la matrice $\boldsymbol{A}$. On rappelle que le *polynôme annulateur* d'une matrice $\boldsymbol{A} \in \mathbb{F}^{n \times n}$ est le polynôme unitaire $P(X) = p_0 + p_1 X + \cdots + p_d X^d$ de plus petit degré tel que $P(\boldsymbol{A}) := p_0 \boldsymbol{I} + p_1 \boldsymbol{A} + \cdots + p_d \boldsymbol{A}^d = \boldsymbol{0}$.

Maintenant, si l'on note

$$P^*(X) := X^d P(1/X) = p_d + p_{d-1} X + \cdots + p_0 X^d$$

le *polynôme réciproque* d'un polynôme $P(X) = p_0 + p_1 X + \cdots + p_d X^d$, alors on peut démontrer le résultat suivant.

Lemme 2.42

Soit $\boldsymbol{A} \in \mathbb{F}^{n \times n}$ non-nulle. Pour tout $\boldsymbol{x} \in \mathbb{F}^n$ non-nul, la suite $\boldsymbol{v} = (\boldsymbol{A}^k \boldsymbol{x})_{k \in \mathbb{N}}$ est une suite récurrente linéaire. Par ailleurs, son polynôme de connexion minimal $P_{\boldsymbol{v}}(X)$ divise $\mu^*_{\boldsymbol{A}}(X)$.

Démonstration : Les vecteurs $\{\boldsymbol{x}, \boldsymbol{A}\boldsymbol{x}, \ldots, \boldsymbol{A}^n \boldsymbol{x}\}$ sont au nombre de $n+1$, donc forment une famille liée de $\mathbb{F}^n$: il existe des coefficients $\lambda_0, \ldots, \lambda_n \in \mathbb{F}$, l'un d'entre eux au moins n'étant pas nul, tels que

$$\sum_{i=0}^{n} \lambda_i \boldsymbol{A}^i \boldsymbol{x} = \boldsymbol{0}\,.$$

Autrement dit, $\sum_{i=0}^{n} \lambda_i \boldsymbol{v}_i = \boldsymbol{0}$, qu'on peut réécrire $\sum_{i=0}^{n} p_i \boldsymbol{v}_{n-i} = \boldsymbol{0}$ en posant $p_i = \lambda_{n-i}$.
Cette équation reste vérifiée pour les termes suivants de la suite $\boldsymbol{v}$. En effet en appliquant $\boldsymbol{A}^k$ à gauche, on obtient :

$$\forall k \geq n, \quad \sum_{i=0}^{n} p_i \, \boldsymbol{v}_{k+n-i} = \boldsymbol{0}\,.$$

Le polynôme minimal $\mu_{\boldsymbol{A}} = \sum_{i=0}^{d} c_i X^i \in \mathbb{F}[X]$ de la matrice $\boldsymbol{A}$ vérifie $\mu_{\boldsymbol{A}}(\boldsymbol{A}) \cdot \boldsymbol{x} = \boldsymbol{0} \cdot \boldsymbol{x} = \boldsymbol{0}$. Par conséquent,

$$\sum_{i=0}^{d} c_i \boldsymbol{A}^i \boldsymbol{x} = \sum_{i=0}^{d} c_{d-i} \boldsymbol{v}_{d-i} = \boldsymbol{0}\,,$$

donc $\mu^*_{\boldsymbol{A}} \in \mathcal{I}_{\boldsymbol{v}}$. Il en résulte que $P_{\boldsymbol{v}}$ divise $\mu^*_{\boldsymbol{A}}$. ∎

Remarque 2.43

Observons que si P et Q sont deux polynômes, alors $P \mid Q \iff P^* \mid Q^*$. Par la suite, nous nous intéresserons principalement aux polynômes réciproques des polynômes de connexion. On notera donc $\mu_{\boldsymbol{v}}(X) := P^*_{\boldsymbol{v}}(X)$. On remarque alors que si $\boldsymbol{v}$ est une suite itérée de matrice $\boldsymbol{A}$ et de vecteur initial $\boldsymbol{v}$, on a alors $\mu_{\boldsymbol{v}}(\boldsymbol{A})\boldsymbol{b} = \boldsymbol{0}$.

Exemple 2.44

Sur $\mathbb{F}_2$, on définit

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \quad \text{et} \quad \boldsymbol{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Le polynôme minimal de $\boldsymbol{A}$ est $X^3 + X^2$. Son polynôme réciproque est $X + 1$. Par ailleurs, on a

$$\boldsymbol{A}\boldsymbol{x} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \text{et} \quad \boldsymbol{A}^2\boldsymbol{x} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Puis, on note que

$$\boldsymbol{A}^k\boldsymbol{x} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \forall k \geq 2\,.$$

Donc le polynôme minimal de la suite $\boldsymbol{v} = (\boldsymbol{A}^k\boldsymbol{x})_{k\in\mathbb{N}}$ est $X + 1$, car $\boldsymbol{v}$ est constante à partir d'un certain rang.

2.3 Algèbre linéaire creuse

2.3.1 Motivation et définitions

Dans la Section 2.1, on a vu qu'il existait un algorithme générique qui permet de calculer l'ensemble des solutions d'un système linéaire $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ en temps $O(n^3)$.

En pratique, des algorithmes évolués on obtient souvent des systèmes linéaires de taille important à résoudre, pour lesquels la matrice $\boldsymbol{A}$ est, ou bien structurée, ou bien contient beaucoup de zéros.

C'est le cas de la phase d'algèbre linéaire des algorithmes modernes de factorisation et de calcul de logarithmes discret que nous verrons dans les chapitres ultérieurs.

Par exemple, dans la factorisation de RSA-768 en 2009 par Kleinjung et ses coauteurs [KAF+10], un système linéaire d'environ 192 millions d'inconnues et équations a dû être résolu. Avec un algorithme générique de résolution du système, et un stockage dense de la matrice associée, plus de 4000*To* de mémoire vive et l'équivalent de 2^{80} opérations auraient été nécessaires, ce qui rend la taĉhe impossible.

Heureusement, la matrice du système contenait seulement $\simeq$ 27 milliards de coefficients non-nuls (soit un peu moins d'un coefficient par million) et a pu être stockée avec une centaine de *Go*. Grâce à un algorithme dérivé de l'algorithme de Wiedemann (que nous verrons dans ce chapitre), la résolution a pu être conduite en un temps estimé à 2000 années de calculs sur un unique processeur classique (AMD Opteron 2,2GHz). Bien entendu, en pratique ce calcul a été effectué sur des clusters de plusieurs dizaines de nœuds.

Commençons par définir la notion de matrice creuse, c'est-à-dire ayant très peu de coefficients non-nuls.

Définition 2.45

Soit $1 \leq t \leq n$. On dit qu'une matrice $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ est t-creuse si chaque ligne de $\boldsymbol{A}$ contient moins de t coefficients non-nuls.

Nos objectifs sont les suivants.

- **Objectif 1 :** obtenir une *solution particulière* du système $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ en temps $O(tn^2)$ et en espace $O(tn)$, où $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ est t-creuse et $\boldsymbol{b} \neq \boldsymbol{0}$.
- **Objectif 2 :** obtenir un *élément du noyau à droite* de $\boldsymbol{A}$ en temps $O(tn^2)$ et en espace $O(tn)$, où $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ est t-creuse.

Pour ce faire, nous allons considérer la suite vectorielle itérée de matrice $\boldsymbol{A}$ et de vecteur initial $\boldsymbol{b}$. Ainsi jusqu'à la fin de ce chapitre on va noter $\boldsymbol{v} \in (\mathbb{F}^n)^{\mathbb{N}}$ la suite définie par

$$\boldsymbol{v} \coloneqq (\boldsymbol{A}^k \boldsymbol{b})_{k\in\mathbb{N}}\,.$$

2.3.2 Calcul d'une solution particulière.

Pour décrire la méthode de calcul efficace d'une solution particulière de $\boldsymbol{Ax} = \boldsymbol{b}$, commençons par une observation. On suppose que l'on connaît $\mu_{\boldsymbol{v}}(X) = \sum_{j=0}^{d} \lambda_j X^j \in \mathbb{F}[X]$, le polynôme réciproque du polynôme de connexion minimal $P_{\boldsymbol{v}}(X)$ de la suite $\boldsymbol{v}$. On a alors :

$$\boldsymbol{0} = \lambda_0 \boldsymbol{b} + \lambda_1 \boldsymbol{A}\boldsymbol{b} + \cdots + \lambda_d \boldsymbol{A}^d \boldsymbol{b} = \mu_{\boldsymbol{v}}(\boldsymbol{A})\boldsymbol{b}\,.$$

Par conséquent, comme $\lambda_0 \neq 0$ on peut écrire

$$\boldsymbol{b} = \boldsymbol{A}Q(\boldsymbol{A})\boldsymbol{b}, \quad \text{avec} \quad Q(X) = \frac{1}{X}\left(1 - \frac{\mu_{\boldsymbol{v}}(X)}{\lambda_0}\right).$$

Il en résulte la méthode de résolution suivante.

1. Calculer le polynôme minimal $P_{\boldsymbol{v}}(X) = \sum_{j=0}^{d} \lambda_j X^j$ de $\boldsymbol{v} = (\boldsymbol{A}^k \boldsymbol{b})_{k\in\mathbb{N}}$, et en déduire $\mu_{\boldsymbol{v}}(X)$.
2. Calculer $Q(X) = \frac{1}{X}(1 - \frac{\mu_{\boldsymbol{v}}(X)}{\mu_{\boldsymbol{v}}(0)})$.
3. Retourner $\boldsymbol{x} = Q(\boldsymbol{A})\boldsymbol{b}$.

Remarque 2.46

Il est important de noter que le calcul de $\boldsymbol{Q}(\boldsymbol{A})\boldsymbol{b}$ s'effectue en temps $O(dnt)$, par une méthode de type « évaluation de Horner » (par exemple).
Plus précisément, si $Q(X) = \sum_{i=0}^{d-1} q_i X^i$, alors

$$Q(\boldsymbol{A})\boldsymbol{b} = \boldsymbol{A}(\boldsymbol{A}\cdots(\boldsymbol{A}(q_{d-1}\boldsymbol{A}\boldsymbol{b} + q_{d-2}\boldsymbol{b}) + q_{d-3}\boldsymbol{b}) + \cdots + q_1\boldsymbol{b}) + q_0\boldsymbol{b}\,,$$

et chaque opération $\boldsymbol{u} \mapsto \boldsymbol{A}\boldsymbol{u}$ coûte $O(nt)$ opérations (les sommes de vecteurs étant en $O(n)$). Par ailleurs, on observe que les calculs se font « en place », donc la complexité en espace reste en $O(nt)$.

Il nous reste à trouver un algorithme efficace pour calculer $P_{\boldsymbol{v}}(X)$. Pour cela démontrons quelques résultats qui caractérisent le lien entre $P_{\boldsymbol{v}}(X)$ et le polynôme annulateur de la matrice $\boldsymbol{A}$.

Lemme 2.47

Soit $Q(X) \in \mathbb{F}[X]$. On a l'équivalence suivante :

$$Q(\boldsymbol{A})\boldsymbol{b} = \boldsymbol{0} \quad \Longleftrightarrow \quad \mu_{\boldsymbol{v}}(X) \text{ divise } Q(X)\,.$$

En particulier, $\mu_{\boldsymbol{v}}(X)$ divise le polynôme annulateur de $\boldsymbol{A}$.

Démonstration : Notons $Q(X) = \sum_{i=0}^{d} q_i X^i$. Si $Q(\boldsymbol{A})\boldsymbol{b} = \boldsymbol{0}$, alors

$$\boldsymbol{0} = \sum_{i=0}^{d} q_i \boldsymbol{A}^i \boldsymbol{b} = \sum_{i=0}^{d} q_i \boldsymbol{v}_i\,.$$

Puis, par application de $\boldsymbol{A}$ à gauche, on a $\sum_{i=0}^{d} q_i \boldsymbol{v}_{n+i} = \boldsymbol{0}$ pour tout $n \geq 0$. Par conséquent $Q(X)$ est un polynôme annulateur de la suite $\boldsymbol{v}$. La réciproque est immédiate. ∎

Lemme 2.48

Soit $\boldsymbol{v} = (\boldsymbol{A}^k\boldsymbol{b})_{k\in\mathbb{N}}$ et $P(X)$ un diviseur de $\mu_{\boldsymbol{v}}(X)$. Alors,

$$\frac{\mu_{\boldsymbol{v}}(X)}{P(X)} = \mu_{\boldsymbol{v}'}(X)$$

où $\boldsymbol{v}' = (\boldsymbol{A}^k P(\boldsymbol{b}))_{k\in\mathbb{N}}$.

Démonstration : La preuve est élémentaire, mais détaillons-la. D'une part on a

$$(\mu_{\boldsymbol{v}'} \cdot P)(\boldsymbol{A})\boldsymbol{b} = \mu_{\boldsymbol{v}'}(\boldsymbol{A})P(\boldsymbol{A})\boldsymbol{b} = \mu_{\boldsymbol{v}'}(\boldsymbol{A})\boldsymbol{b}' = \boldsymbol{0}$$

donc le polynôme $\mu_{\boldsymbol{v}}$ divise le produit $\mu_{\boldsymbol{v}'}P$. D'autre part, soit $Q(X) = \mu_{\boldsymbol{v}}(X)/P(X)$. On observe que

$$Q(\boldsymbol{A})\boldsymbol{b}' = Q(\boldsymbol{A})P(\boldsymbol{A})\boldsymbol{b} = \mu_{\boldsymbol{v}}(\boldsymbol{A})\boldsymbol{b} = \boldsymbol{0}$$

donc $\mu_{\boldsymbol{v}'}$ divise Q. Autrement dit, $\mu_{\boldsymbol{v}'}P$ divise $QP = \mu_{\boldsymbol{v}}$. ■

Pour calculer le polynôme $\mu_{\boldsymbol{v}}(X)$, une idée serait de s'appuyer sur le Lemme 2.40 : on calcule tous les $P_{\boldsymbol{v}^{(j)}}(X)$ par l'algorithme de Berlekamp–Massey, puis on en déduit $P_{\boldsymbol{v}}$ et $\mu_{\boldsymbol{v}}$ avec un calcul de ppcm. Malheureusement, cette méthode est trop coûteuse, car chaque calcul de $P_{\boldsymbol{v}^{(j)}}(X)$ a un coût en $O(n^2)$.

Pour accélérer les calculs (en moyenne), on va donc s'appuyer sur la Remarque 2.41 : essentiellement on va chercher des facteurs de $\mu_{\boldsymbol{v}}(X)$ aléatoires (qu'on espère de grand degré) en calculant les polynômes de conenxion de suites scalaires de la forme $\langle \boldsymbol{x}, \boldsymbol{v}\rangle$ où $\boldsymbol{x}$ est tiré aléatoirement.

Algorithme 5 : Algorithme `MinPoly` de calcul du polynôme minimal $\mu_{\boldsymbol{v}}(X)$ d'une suite itérée $\boldsymbol{v} = (\boldsymbol{A}^k\boldsymbol{b})_{k\in\mathbb{N}}$.

Entrée : Une matrice creuse $\boldsymbol{A} \in \mathbb{F}^{n\times n}$, un vecteur $\boldsymbol{b} \in \mathbb{F}^n$ et un entier $d \leq n$
Sortie : Le polynôme $\mu_{\boldsymbol{v}}(X)$ où $\boldsymbol{v} = (\boldsymbol{A}^k\boldsymbol{b})_{k\in\mathbb{N}}$

1 **Si** $\boldsymbol{b} = \boldsymbol{0}$
2 ⌊ **Retourner** le polynôme 1.
3 Choisir aléatoirement $\boldsymbol{x} \in \mathbb{F}^n$ non-nul.
4 Calculer les $2d$ premiers termes de la suite scalaire $\boldsymbol{a}$ définie par $a_k = \langle \boldsymbol{x}, \boldsymbol{A}^k\boldsymbol{b}\rangle$.
5 **Si** $\boldsymbol{a}$ *est nulle sur ses* $2d$ *termes*
6 ⌊ Revenir à l'étape 3.
7 Calculer le polynôme minimal de $\mu_{\boldsymbol{a}}(X)$ de la suite scalaire $\boldsymbol{a}$, grâce à l'algorithme de Berlekamp–Massey.
8 **Si** $\deg \mu_{\boldsymbol{a}}(X) = d$
9 ⌊ **Retourner** $\mu_{\boldsymbol{a}}(X)$.
10 Calculer $\boldsymbol{b}' = \mu_{\boldsymbol{a}}(\boldsymbol{A})\boldsymbol{b}$.
11 Faire un appel récursif de l'algorithme `MinPoly`, avec en entrée la matrice $\boldsymbol{A}$, le vecteur $\boldsymbol{b}'$ et l'entier $d' := d - \deg \mu_{\boldsymbol{a}}(X)$.

Theorème 2.49

Soit $\boldsymbol{v} \in (\mathbb{F}^n)^{\mathbb{F}}$ la suite itérée de matrice $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ et de vecteur initial $\boldsymbol{b} \in \mathbb{F}^n$. Lancé avec $\boldsymbol{A}$, $\boldsymbol{b}$ et $d = n$, l'Algorithme 5 termine et calcule $\mu_{\boldsymbol{v}}(X)$ en effectuant, en moyenne, $O(tn^2)$ opérations dans le corps $\mathbb{F}$.

Démonstration : À venir. ■

2.3.3 Calcul d'un élément du noyau.

Supposons que $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ est creuse est non-inversible. On cherche un élément non-nul $\boldsymbol{x} \in \mathbb{F}^n$ tel que $\boldsymbol{Ax} = \boldsymbol{0}$. Pour cela, on note d'abord que le polynôme annulateur $\mu_{\boldsymbol{A}}(X)$ s'écrit comme $XQ(X)$, car 0 est valeur propre de $\boldsymbol{A}$.

L'idée de l'algorithme de Wiedemann est alors la suivante : avec bonne probabilité sur le tirage de $\boldsymbol{u} \in \mathbb{F}^n$, le vecteur $\boldsymbol{x} = Q(\boldsymbol{A})\boldsymbol{u}$ est non-nul, et il est dans le noyau de $\boldsymbol{A}$ car $\boldsymbol{A}Q(\boldsymbol{A}) = \boldsymbol{0}$.

Algorithme 6 : Calcul du polynôme annulateur d'une matrice creuse

Entrée : une matrice creuse $\boldsymbol{A} \in \mathbb{F}^{n\times n}$
Sortie : le polynôme minimal $\mu_{\boldsymbol{A}}(X)$ de $\boldsymbol{A}$
1 Initialiser $P(X) \leftarrow 1$.
2 **Tant que** $P(\boldsymbol{A}) \neq \boldsymbol{0}$ **faire**
3 Choisir aléatoirement $\boldsymbol{v}$ et $\boldsymbol{w}$ non-nuls dans $\mathbb{F}^n$.
4 Calculer les $2n$ premiers termes de $\boldsymbol{a} := (\langle \boldsymbol{v}, A^k\boldsymbol{w}\rangle)_{k\in\mathbb{N}}$.
5 Calculer le polynôme réciproque $\mu_{\boldsymbol{a}}(X)$ du polynôme de connexion minimal de $\boldsymbol{a} \in \mathbb{F}^{\mathbb{N}}$.
6 Calculer $P(X) \leftarrow \text{ppcm}(P(X), \mu_{\boldsymbol{a}}(X))$.
7 **Retourner** $P(X)$.

Algorithme 7 : Algorithme de Wiedemann

Entrée : une matrice creuse $\boldsymbol{A} \in \mathbb{F}^{n\times n}$
Sortie : un élément aléatoire du noyau de $\boldsymbol{A}$
1 Calculer le polynôme minimal $\mu_{\boldsymbol{A}}(X)$ de la matrice $\boldsymbol{A}$, et définir $Q(X) = \mu_{\boldsymbol{A}}(X)/X$.
2 Assigner $\boldsymbol{x} \leftarrow \boldsymbol{0}$.
3 **Tant que** $\boldsymbol{x} = \boldsymbol{0}$ **faire**
4 Tirer aléatoirement $\boldsymbol{u} \in \mathbb{F}^n \setminus \{\boldsymbol{0}\}$.
5 Calculer $\boldsymbol{x} = Q(\boldsymbol{A})\boldsymbol{u}$.
6 **Retourner** $\boldsymbol{x}$.

Si l'on suppose $\mathbb{F}$ fini de cardinal q, on peut effectuer un tirage uniforme lors de l'étape 4 de l'Algorithme 7. Alors, la probabilité que $\boldsymbol{x} = \boldsymbol{0}$ après un passage dans la boucle est :

$$\mathbb{P}(\boldsymbol{x} = \boldsymbol{0}) = \mathbb{P}(\boldsymbol{u} \in \ker Q(\boldsymbol{A})) = \frac{q^{\dim\ker Q(\boldsymbol{A})} - 1}{q^n - 1} \simeq \frac{1}{q^{\text{rang}(\boldsymbol{A})}}.$$

Théorème 2.50

Soit $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ une matrice t-creuse. Alors, l'algorithme de Wiedemann calcule un élément du noyau de $\boldsymbol{A}$ en temps moyen $O(tn^2)$.

Démonstration : À venir. ■

Encore plus de détails à venir.

Chapitre 3

Calcul de racines carrées

Ce chapitre a pour objet l'extraction effective de racines carrées. Dans une première section, nous étudierons le problème dans l'ensemble des entiers naturels. Puis, nous nous tournerons vers les corps finis et les anneaux d'entiers modulaires, pour lesquels le calcul de racines carrées a des applications pratiques que nous découvrirons dans les chapitres suivants.

3.1 Racines carrées entières approchées

On s'intéresse d'abord au problème du calcul de racine carrée dans les entiers naturels, qui s'avère sensiblement plus simple que dans d'autres structures algébriques. Explicitons d'abord le problème.

Problème 1

Étant donné $n \in \mathbb{N}$, calculer le plus grand entier $x \in \mathbb{N}$ tel que $x^2 \leq n$.

On appelle *racine carrée* de n l'entier x trouvé, et *reste* l'entier $n - x^2 \geq 0$.

Remarque 3.1

La racine carrée x et le reste r (de n) forment l'unique couple d'entiers tels que $n = x^2 + r$ et $r \leq 2x$.

3.1.1 Méthode élémentaire

La méthode exhaustive (par incrémentation et test) pour calculer la racine carrée a un coût $O(\sqrt{n})$. Une première manière de réaliser ce calcul plus efficacement consiste à calculer de proche en proche la décomposition en base 2 de la racine carrée. Pour cela, on utilise la caractérisation suivante.

Lemme 3.2

Soient $n \geq 0$ et $n' = 4n + s$ où $s \in \{0, 1, 2, 3\}$. Si x est la racine carrée entière de n et $r = n - x^2$, alors :

- *lorsque $4r + m \leq 4x$, l'entier n' admet comme racine carrée $2x$ et comme reste $4r + m$;*
- *lorsque $4r + m \geq 4x + 1$, l'entier n' admet comme racine carrée $2x + 1$ et comme reste $4r + m - 4x - 1$.*

Démonstration : On écrit

$$n' = 4n + m = 4(x^2 + r) + m = (2x)^2 + (4r + m)$$

puis on teste si $4r + m$ est un reste potentiel avec la Remarque 3.1. Si $4r + m \leq 2x$, alors on a le bon reste. Sinon il faut ajouter 1 à $2x$ et on obtient l'écriture $n' = (2x + 1)^2 + (4(r - x) + m - 1)$. ■

Algorithme 8 : Méthode élémentaire pour le calcul de racine carrée entière

Entrée : Un entier $n = \sum_{j=0}^{d} n_j 4^j \in \mathbb{N}$

Sortie : Le plus grand entier $x = \sum_{j=0}^{d} x_j 2^j \in \mathbb{N}$ tel que $x^2 \leq n$.

```
Initialiser x_d ← 0 et r_d ← 0.
Pour tout i allant de d à 0 faire
    Calculer t ← 4r_i + n_i
    Si t > 4x_i
        x_{i-1} ← 2x_i + 1
        r_{i-1} ← t − 4x_i − 1
    Sinon
        x_{i-1} ← 2x_i
        r_{i-1} ← t
Retourner x.
```

Chaque étape de l'algorithme produit un chiffre en base 2 de la racine carrée. La complexité approximative de l'Algorithme 8 est donc en $O(\log_4(n)) = O(\log(n))$ opérations sur les entiers. Notons plus précisément que les opérations sont ou bien l'addition de deux entiers (de coût $O(\log(n))$ opérations binaires), ou bien la multiplication par 2 ou 4 (également de coût $O(\log_2(n))$ opérations binaires). On évite donc les opérations plus coûteuses comme la multiplication « complète » ou la division, et on a donc une complexité binaire en $O(\log_2^2(n))$.

Exemple 3.3

On prend $n = 4724$, c'est-à-dire $n = \sum_{j=0}^{d} n_j 4^j$ avec $d = 6$ et $(n_d, \ldots, n_0) = (1, 0, 2, 1, 3, 1, 0)$. On note $\overline{x}_i$ et $\overline{r}_i$ les valeurs de x et r calculées à l'étape i. On indique également par $\overline{n}_i$ le quotient de la division de x par 4^i, afin de remarque que $\overline{x}_i^2 + \overline{r}_i = \overline{n}_i$ à chaque étape. Voici le déroulement du calcul :

i	$\overline{n}_i$	$\overline{x}_i$	$\overline{r}_i$	t
7		0	0	1
6	$1 = (1)_4$	$1 = (1)_2$	0	0
5	$4 = (10)_4$	$2 = (10)_2$	0	2
4	$18 = (102)_4$	$4 = (100)_2$	2	9
3	$73 = (1021)_4$	$8 = (1000)_2$	9	39
2	$295 = (10213)_4$	$17 = (10001)_2$	6	25
1	$1181 = (102131)_4$	$34 = (100010)_2$	25	100
0	$4724 = (1021310)_4$	$68 = (1000100)_2$	100	

3.1.2 Méthode de Héron

Une seconde manière de résoudre le Problème 1 consiste à effectuer des itérations de la méthode de Newton. Rappelons rapidement cette méthode (vue dans un cours de Master 1).

Étant donnée une fonction $f : \mathbb{R} \to \mathbb{R}$ de classe $\mathcal{C}^2$, on souhaite obtenir une valeur approchée d'un élément x tel que $f(x) = 0$. Pour des raisons techniques, on suppose également que $f''(x) \neq 0$. On construit alors une suite récurrente

$$x_0 \text{ arbitraire} \quad \text{et} \quad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

et on observe que la suite converge vers un élément $a \in \mathbb{R}$ tel que $f(a) = 0$.

Pour le cas du calcul d'une racine carrée entière, la méthode est appelée *méthode Héron* ou *méthode babylonnienne* on choisit la fonction $f(x) = x^2 - n$ et on obtient la suite récurrente

$$x_0 \text{ arbitraire} \quad \text{et} \quad x_{k+1} = \frac{1}{2}\left(x_k + \frac{n}{x_k}\right).$$

Par ailleurs, on peut également estimer la vitesse de convergence de la suite vers la racine carrée de n.

Proposition 3.4

La vitesse de convergence de la suite $(x_k)_{k\in\mathbb{N}}$ est quadratique. On a précisément

$$0 \leq \frac{x_{k+1} - \sqrt{n}}{\sqrt{n}} < \frac{1}{2}\left(\frac{x_k - \sqrt{a}}{\sqrt{a}}\right)^2.$$

Autrement dit, à chaque itération, le nombre de chiffres significatifs obtenus pour x double.

Démonstration : Todo: à venir ■

Remarque 3.5

Le choix du terme initial x_0 de la suite récurrente est arbitraire. Cependant, on peut essayer de faire d'éviter des calculs supplémentaires en prenant x_0 de l'ordre de grandeur de $\sqrt{a}$ (sans faire de calcul). Par exemple, si a est codé sur m bits ($2^{m-1} \leq a < 2^m$), alors on peut choisir $x_0 = 2^{\lceil m/2 \rceil}$.

Complexité en $O(\sqrt{\log(n)})$, mais ce sont des calculs sur des nombres flottants.....

Exemple 3.6

Pour calculer la racine carrée de $n = 3141592653589793$, on obtient le résultat $x = 56049912$ en 5 itérations :

78812208,6341
59336979,9265
56140958,3757
56049985,9908
56049912,164

La méthode élémentaire demande 26 étapes.

3.2 Racines carrées dans les corps finis

3.2.1 Définitions

Commençons par spécifier le problème à étudier.

Définition 3.7

Soit $a \in \mathbb{F}_q$. Une racine carrée de a est un élément $x \in \mathbb{F}_q$ tel que $x^2 = a$.

Problème 2

Étant donné $a \in \mathbb{F}_q$, calculer l'ensemble des racines carrées $x \in \mathbb{F}_q$ de a.

Remarquons que tout élément $a \in \mathbb{F}_q$ admet au plus deux racines carrées dans un corps fini (car un corps est intègre). Ces racines sont opposées l'une de l'autre.

Traitons dès à présent le cas où $\mathbb{F}_q$ est de caractéristique $p = 2$.

Proposition 3.8

Dans $\mathbb{F}_q$ avec $q = 2^k$, tout élément est un carré et toute racine carrée peut être extraite en $O(\log q)$ élévations au carré.

Démonstration : Si $q = 2^k$, alors pour tout $a \in \mathbb{F}_q$:

$$\left(a^{2^{k-1}}\right)^2 = a^{2^k} = a\,.$$

Ainsi, $x = a^{2^{k-1}}$ est l'unique racine carrée de a (unique car $x = -x$ sur $\mathbb{F}_q$) et se calcule en $O(\log(q))$ élévations au carré dans le corps $\mathbb{F}_q$:

$$a \mapsto a^2 \mapsto (a^2)^2 = a^{2^2} \mapsto ((a^2)^2)^2 = a^{2^3} \mapsto \cdots \mapsto a^{2^{k-1}}\,.$$

■

Comme une méthode a été trouvée pour $p := \mathrm{car}(\mathbb{F}_q) = 2$, dans toute la suite on suppose maintenant que $p \neq 2$.

3.2.2 Le cas $q \equiv 3 \bmod 4$.

Lemme 3.9

Soit $a \in \mathbb{F}_q \setminus \{0\}$. Alors, a est un carré dans $\mathbb{F}_q$ si et seulement si $a^{(q-1)/2} = 1$.

Démonstration : Soit $\mathcal{C} = \{u \in \mathbb{F}_q \setminus \{0\}, u \text{ est un carré }\}$ et $\mathcal{R} = \{v \in \mathbb{F}_q, v^{(q-1)/2} = 1\}$. Montrons que $\mathcal{C} = \mathcal{R}$.
Si $u \in \mathcal{C}$, alors $u = x^2$ donc $u^{(q-1)/2} = x^{q-1} = 1$. Donc $\mathcal{C} \subseteq \mathcal{R}$.
Par ailleurs, $|\mathcal{R}| \leq \frac{q-1}{2}$ car ce sont les racines d'un polynôme de degré $\frac{q-1}{2}$. D'autre part, $|\mathcal{C}| \geq \frac{q-1}{2}$, car l'application $t \mapsto t^2$ a au plus 2 antécédents ($\mathbb{F}_q$ est un corps). ■

Pour tout carré $a \in \mathbb{F}_q$, on a donc

$$a^{(q+1)/2} = a\,.$$

Proposition 3.10

Si $q \equiv 3 \bmod 4$, alors on peut trouver les racines carrées d'un carré $a \in \mathbb{F}_q$ en $O(\log q)$ opérations dans $\mathbb{F}_q$.

Démonstration : D'après ce qui précède, il suffit de calculer l'exponentiation $a^{(q+1)/2}$, ce qui s'effectue en $\leq \lceil \log_2((q+1)/2) \rceil$ multiplications et carrés. ■

3.2.3 Autres cas spécifiques

Dans la même veine que le cas $q \equiv 3 \bmod 4$, il existe des algorithmes spéciaux pour extraire une racine carrée lorsque :

— $q \equiv 5 \bmod 8$: c'est l'algorithme d'Atkin, voir exercices ;
— $q \equiv 9 \bmod 16$: par Müller, puis Kong-Cai-Yu-Li.

Ces algorithmes s'exécutent également en $O(\log(q))$ opérations dans le corps $\mathbb{F}_q$.

Néanmoins, il n'y a à l'heure actuelle aucun algorithme déterministe efficace permettant de réoudre le problème de l'extraction de racine carrée dans le cas où $q \equiv 1 \mod 16$. Nous allons donc découvrir, dans la prochaine section, deux algorithmes probabilistes calculant des racines carrées dans un cadre général.

3.2.4 Algorithmes génériques

Avant d'introduire ces algorithmes, effectuons quelques rappels sur la résiduosité quadratique.

3.2.4.1 Rappels

Définition 3.11 (*Symbole de Legendre*)

Soit p un nombre premier et $a \in \mathbb{Z}/p\mathbb{Z}$. Le symbole de Legendre de a modulo p est

$$\left(\frac{a}{p}\right) := \begin{cases} 1 & \text{si } a \text{ est un carré dans } \mathbb{F}_p, \\ -1 & \text{sinon.} \end{cases}$$

Le calcul d'un symbole de Legendre s'effectue en $O(\log a \log p)$ opérations binaires, notamment grâce aux propriétés suivantes. Voir Algorithme 9 pour les détails.

Proposition 3.12

Soit p premier, et a, b deux entiers. Alors

1. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$,
2. $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$,
3. $\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$,
4. *si a est impair, alors* $\left(\frac{a}{p}\right) = (-1)^t$ *où* $t = \sum_{j=0}^{(p-1)/2} \lfloor \frac{aj}{p} \rfloor$.

Proposition 3.13 (*Loi de réciprocité quadratique*)

Si p et q sont premiers, alors

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{(p-1)(q-1)/4}$$

Par multiplicativité, on peut étendre la définition du symbole de Legendre au cas où le module n'est plus premier :

$$\left(\frac{a}{n_1 n_2}\right) = \left(\frac{a}{n_1}\right)\left(\frac{a}{n_2}\right)$$

Le symbole obtenu est appelé *symbole de Jacobi*. Dans ce cas, si a est un carré modulo n, alors $\left(\frac{a}{n}\right) = 1$, mais la réciproque n'est plus vraie.

Enfin, pour tester si un élément est un carré dans $\mathbb{F}_q$, avec q non premier, calculer un sysmbole de Legendre éventuel n'a pas de sens.

Proposition 3.14

Soit $q = p^k$, avec $k \geq 1$. Il existe un algorithme permettant de tester si $a \in \mathbb{F}_q$ est un carré en $O(\log(q))$ opérations dans $\mathbb{F}_q$ et $O(\log(p)^2)$ opérations binaires.

Algorithme 9 : Une méthode de calcul du symbole de Legendre

Entrée : $a \in \mathbb{Z}$ et $p \geq 2$ premier

Sortie : le symbole de Legendre $\left(\frac{a}{p}\right)$

Si $a = 0$
  Retourner 0
Si $a = 1$ *ou* $p = 2$
  Retourner 1
Si $a = p - 1$
  Si $a \equiv 0 \bmod 4$
    Retourner 1
  Sinon
    Retourner -1
Si $a \equiv 0 \bmod 2$
  Si $p \equiv 1 \bmod 8$ *ou* $p \equiv 3 \bmod 8$
    Retourner Legendre$(a/2, p)$
  Sinon
    Retourner $(-1)\times$ Legendre$(a/2, p)$
Si $a \geq p$
  Retourner Legendre$(a \bmod p, p)$
Si $a \bmod 4 == 1$ *ou* $p \bmod 4 == 1$
  Retourner Legendre(p, a)
Sinon
  Retourner $(-1)\times$ Legendre(p, a)

Démonstration : Le test se résume à calculer une norme $\mathbb{F}_q/\mathbb{F}_p$ et un symbole de Legendre modulo p. Voir détails en exercice. ■

3.2.4.2 Algorithme de Cipolla

Présentons un premier algorithme pour l'extraction de racines carrées : l'algorithme de Cipolla.

Définition 3.15

Soit $\mathbb{F}_{q^2} = \mathbb{F}_q(\beta)$ une extension quadratique de $\mathbb{F}_q$. Pour $z = u + \beta v \in \mathbb{F}_{q^2}$, on définit :
- *sa trace comme $\mathrm{Tr}(z) := z + z^q \in \mathbb{F}_q$, qui vérifie alors $\mathrm{Tr}(z) = 2u + \mathrm{Tr}(\beta)v$.*
- *sa norme comme $\mathrm{N}(z) = z^q z = z^{q+1} \in \mathbb{F}_q$, qui vérifie alors $\mathrm{N}(z) = u^2 + \mathrm{Tr}(\beta)uv + \mathrm{N}(\beta)v^2$.*

Rappelons que l'on se place dans le cas où q est impair. L'algorithme de Cipolla repose sur la remarque suivante.

Remarque 3.16

Pour $z \in \mathbb{F}_{q^2}$, si $\mathrm{N}(z)$ est un carré, alors une racine carrée de $\mathrm{N}(z)$ est $z^{(q+1)/2}$, qui se calcule en $O(\log(q))$ opérations dans $\mathbb{F}_{q^2}$

Pour calculer une racine carrée dans $\mathbb{F}_q$, une idée est alors d'essayer d'écrire a comme la norme d'un élément de $\mathbb{F}_{q^2}$. Notons que l'objectif est raisonnable car l'application norme $\mathrm{N} : \mathbb{F}_{q^2} \to \mathbb{F}_q$ est surjective.

Supposons que l'élément β définissant une extension quadratique $\mathbb{F}_{q^2} = \mathbb{F}_q(\beta)$ vérifie $\beta^2 = c$. Autrement dit, $P(X) = X^2 - c$ est irréductible sur $\mathbb{F}_q$. Alors les racines de P sont β et β^q, et on a donc $P(X) = (X-\beta)(X-\beta^q) = X^2 - \mathrm{Tr}(\beta)X + \mathrm{N}(\beta)$.

Autrement dit,

$$\mathrm{Tr}(\beta) = 0 \quad \text{et} \quad \mathrm{N}(\beta) = -c\,.$$

Donc, si $z = u + \beta \in \mathbb{F}_{q^2}$, alors on a

$$\mathrm{N}(z) = u^2 + \mathrm{Tr}(\beta)u + \mathrm{N}(\beta) = u^2 - c\,.$$

Maintenant, si u et c ont été choisis tels que $a = u^2 - c$, alors on a exprimé a comme une norme $N(z)$.

Algorithme 10 : Algorithme de Cipolla

Entrée : $a \in \mathbb{F}_q$ un carré
Sortie : $x \in \mathbb{F}_q$ tel que $x^2 = a$

1 Tirer $u \in \mathbb{F}_q$ et calculer $c = u^2 - a$.
2 **Si** *c est un carré*
3 Revenir à l'étape 1
4 **Sinon**
5 Construire $\mathbb{F}_{q^2} = \mathbb{F}_q[X]/(X^2 - c)$ et noter β la classe de X.
6 Retourner $x = (u+\beta)^{(q+1)/2}$.

Rappelons que tester si c est un carré dans $\mathbb{F}_q$ peut être réduit à un calcul de norme dans $\mathbb{F}_q/\mathbb{F}_p$ puis un calcul de symbole de Legendre. On peut également montrer que le nombre de tirages de u à effectuer est constant. [plus de détails à venir sur ce point] On a donc le résultat de complexité suivant.

Proposition 3.17

L'algorithme de Cipolla calcule une racine carrée dans $\mathbb{F}_q$, avec $q = p^k$ impair, avec en moyenne $O(\log q)$ opérations dans $\mathbb{F}_{q^2}$ et $O(\log q \log p)$ opérations dans $\mathbb{F}_q$.

Exemple 3.18

Voici un exemple d'éxecution de l'algorithme de Cipolla, avec $q = 100000007$ (premier) et $a = 40598710$.

1. Premier tirage : $u = 9871141$ *donne* $c = 77222420$, *mais c est un carré.*
2. Second tirage : $u = 19298050$ *donne* $c = 67134768$, *et c n'est pas un carré.*

On construit donc $P(X) = X^2 - c = X^2 + 32865239$, puis on calcule dans $\mathbb{F}_q[X]/(P)$ la valeur de $(u+X)^{50000004}$.
On obtient le résultat $x = 31460202$, et on peut vérifier que

$$31460202^2 \equiv 40598710 \mod 100000007\,.$$

3.2.4.3 Algorithme de Tonelli-Shanks

Pas vu en cours ; détails à venir.

3.3 Racines carrées dans $\mathbb{Z}/N\mathbb{Z}$

On place maintenant dans l'anneau $\mathbb{Z}/N\mathbb{Z}$, avec $N = \prod_{i=1}^{k} p_i^{e_i}$ non premier. La définition d'une racine racine carrée dans cet anneau reste naturelle :

Définition 3.19

Soit $a \in \mathbb{Z}/N\mathbb{Z}$. Une racine carrée de a modulo N est un entier $x \in \mathbb{Z}/N\mathbb{Z}$ tel que $x^2 \equiv a \mod N$.

Néanmoins, il faut prendre garde que le nombre de racines carrées n'est plus limitée à 2 : il peut atteindre 2^k où k le nombre de premiers distincts divisant N.

Pour calculer une racine carrée modulo N, notre stratégie sera la suivante.

1. Pour tout i, calculer les racines carrées de a modulo p_i.
2. En déduire, pour tout i, les racines carrées de a modulo $p_i^{e_i}$.
3. En déduire les racines carrées de N.

L'étape 1 a été traitée dans la section précédente. L'étape 2 sera traitée dans les Sections 3.3.2 (p_i impair) et 3.3.3 ($p_i = 2$). L'étape 3 est traitée dans la section qui suit.

3.3.1 Le cas $N = N_1 N_2$, avec N_1 et N_2 premiers entre eux

Dans le cas où N est le produit de deux nombres premiers entre eux, l'idée est d'utiliser le théorème des restes chinois pour reconstruire les racines carrées modulo N à partir des racines carrées modulo N_1 et N_2.

Todo: Rappeler CRT dans le chapitre 1

Proposition 3.20

Soit $N = N_1 N_2$ où N_1 et $N_2 \geq 2$ sont deux nombres premiers entre eux. Soient $u, v \in \mathbb{Z}$ tels que $uN_1 + vN_2 = 1$. Soit enfin $a \in \{0, \ldots, N-1\}$ un carré modulo N. Alors,

1. *L'entier a est également un carré dans $\mathbb{Z}/N_1\mathbb{Z}$ et dans $\mathbb{Z}/N_2\mathbb{Z}$.*
2. *Si $\mathcal{R}_{N_1}$ et $\mathcal{R}_{N_2}$ forment les ensembles des racines carrées de a dans $\mathbb{Z}/N_1\mathbb{Z}$ et dans $\mathbb{Z}/N_2\mathbb{Z}$, alors a admet comme racines carrées modulo N l'ensemble :*

$$\mathcal{R} = \{N_1 u x_2 + N_2 v x_1 \mid x_1 \in \mathcal{R}_{N_1}, x_2 \in \mathcal{R}_{N_2}\} \subseteq \mathbb{Z}/N\mathbb{Z}\,.$$

Démonstration : Soit r une racine carrée de a dans $\mathbb{Z}/N\mathbb{Z}$. On a $a = r^2 + kN_1N_2$ pour $k \in \mathbb{Z}$, donc $a = r^2$ dans $\mathbb{Z}/N_1\mathbb{Z}$ et dans $\mathbb{Z}/N_2\mathbb{Z}$.
Si r s'écrit sous la forme $N_1 u x_2 + N_2 v x_1$, alors on trouve

$$r^2 \equiv x_1^2 \equiv a \mod N_1 \quad \text{et} \quad r^2 \equiv r_2^2 \equiv a \mod N_2\,.$$

D'après le théorème des restes chinois, ces valeurs sont les uniques racines de a dans $\mathbb{Z}/N\mathbb{Z}$. ■

3.3.2 Le cas $N = p^k$, avec $p \neq 2$

Dans le cas où N est la puissance d'un nombre premier p, on ne peut plus appliquer le théorème des restes chinois.

L'idée est alors de « remonter » les racines carrées de a modulo p en des racines carrées modulo p^2, p^3, jusqu'à $N = p^k$. Cette technique s'apparente à une méthode plus générale issue du lemme de Hensel (*Hensel lifting lemma*) que nous reverrons dans les chapitre suivants.

Soit $a \in \mathbb{Z}/N\mathbb{Z}$ dont on cherche une racine carrée. Supposons pour l'instant que a et N (donc a et p) sont premiers entre eux.

Lemme 3.21

Soit $p \geq 3$ premier et $a \in \{0, \ldots, p^i - 1\}$ premier avec p, pour un certain $i \geq 2$. Si $x \in \{0, \ldots, p^i - 1\}$ est une racine carrée de a modulo p^i, alors x s'écrit $y + tp^{i-1}$ avec

- *un entier $y \in \{0, \ldots, p^{i-1} - 1\}$ qui forme une racine carrée de a modulo p^{i-1}, et*
- *un entier $t \in \{0, \ldots, p - 1\}$ tel que*

$$t = \frac{a - y^2}{p^{i-1}} \times ((2y)^{-1} \mod p).$$

Démonstration : Écrivons de manière unique $x = y + tp^{i-1}$ avec $y < p^{i-1}$ et $t < p$. En élevant au carré on obtient

$$x^2 = y^2 + p^{i-1}(2ty + t^2 p^{i-1}).$$

On note alors que

$$x^2 \equiv a \mod p^i \iff y^2 \equiv a \mod p^{i-1}.$$

Trouvons maintenant les valeurs de t possibles. Si $x^2 \equiv a \mod p^i$, alors il résulte que

$$\frac{a - y^2}{p^{i-1}} \equiv 2ty \mod p \tag{3.1}$$

Comme $\text{pgcd}(a, p) = 1$, les entiers y^2 et p (donc y et p) sont également premiers entre eux.
Donc l'unique solution de (3.1) est

$$t = \frac{a - y^2}{p^{i-1}} \times ((2y)^{-1} \mod p).$$

■

Dans le cas où a et p ne sont pas premiers entre eux, la remontée de solution est plus aisée.

Lemme 3.22

Soit a un entier tel que a et p ne sont pas premiers entre eux, et tel que a est un carré modulo p^i avec $i \geq 3$. Alors p^2 divise a, et toute racine carrée x de a modulo p^i s'écrit $x = up$ avec $u^2 \equiv a/p^2 \mod p^{i-2}$.

Démonstration : Comme p est premier et $\text{pgcd}(a, p) = 1$, on a $a = vp$ avec $v \in \mathbb{Z}$. Par ailleurs $a = x^2 + kp^i$ avec $k \in \mathbb{Z}$. Par conséquent $x^2 = vp - kp^i$, donc p divise x puis en écrivant $x = up$ on a $u^2p^2 = vp - kp^i$. Comme $i \geq 2$, ceci implique que p^2 divise $vp = a$, et on a bien $u^2 \equiv a/p^2 \mod p^{i-2}$. ■

Exemple 3.23

Todo: faire un joli example

3.3.3 Le cas $N = 2^k$

Pour $N = 2^k$, on raisonne de manière similaire.

On traite d'abord les petits cas, qui se trouvent être particuliers.

1. Dans $\mathbb{Z}/2\mathbb{Z}$:

x	0	1
x^2	0	1

Algorithme 11 : Calcul d'une racine carrée modulo $N = p^k$

Entrée : p premier, $k \geq 1$ et $a < p^k$
Sortie : x tel que $x^2 \equiv a \mod p^k$

$r \leftarrow 1$
$a' \leftarrow a$
$k' \leftarrow k$
Tant que p *divise* a' **faire**
  $a' \leftarrow a/p^2$
  $r \leftarrow rp$
  $k \leftarrow k-2$
Calculer x une racine carrée de a modulo p (par exemple avec l'algorithme de Cipolla)
Pour tout i *allant de* 2 *à* k' **faire**
  Calculer $z \leftarrow (a' - x^2)/p^{i-1}$ dans $\mathbb{Z}$
  Calculer $t \leftarrow z \times (2x)^{-1} \mod p$
  Calculer $x \leftarrow x + tp^{i-1}$
Retourner $x \times r$.

2. Dans $\mathbb{Z}/4\mathbb{Z}$:

x	0	1	2	3
x^2	0	1	0	1

3. Dans $\mathbb{Z}/8\mathbb{Z}$:

x	0	1	2	3	4	5	6	7
x^2	0	1	4	1	0	1	4	1

Pour le cas a pair, on utilise le Lemme 3.22 pour éliminer la partie paire de a. On suppose maintenant a impair.

Lemme 3.24

Soient $k \geq 3$, a un nombre impair et r tel que $r^2 \equiv a \mod 2^k$. Alors, l'équation $x^2 \equiv a \mod 2^k$ admet exactement 4 solutions :

$$x \in \{r,\ -r,\ r - 2^{k-1},\ -r + 2^{k-1}\}.$$

Démonstration : On raisonne par induction. C'est vrai pour $k = 3$ d'après le tableau précédent.
Soit maintenant $x = u + \epsilon 2^k$ où $u < 2^k$ et $\epsilon \in \{0,1\}$.
Si $x^2 \equiv a \mod 2^{k+1}$, alors on a $u^2 \equiv a \mod 2^k$. Autrement dit, les solutions à l'ordre $k+1$ ne peuvent être que :

$$r,\ -r, 2^{k-1} - r, r - 2^{k-1} \qquad \text{(solutions à l'ordre } k\text{)}$$

et

$$r + 2^k,\ -r + 2^k, 2^{k-1} - r + 2^k, r - 2^{k-1} + 2^k\,.$$

En raisonnant modulo 2^{k+1}, on peut réécrire cet ensemble comme

$$\{r, -r, -r + 2^k, r - 2^k, 2^{k-1} - r, -r + 2^{k-1}, r + 2^{k-1}, -r - 2^{k-1}\}\,.$$

On vérifie par un calcul simple que les 4 premières valeurs sont solutions dans $\mathbb{Z}/p^{k+1}\mathbb{Z}$. On va voir dans le lemme suivant que les 4 dernières solutions ne le sont pas. ■

Lemme 3.25

Si a est impair et x tel que $x^2 \equiv a \mod 2^k$, alors :

$$x^2 \equiv a \mod 2^{k+1} \iff (x + 2^{k-1})^2 \not\equiv a \mod 2^{k+1}\,.$$

Démonstration : On a

$$(x + 2^{k-1})^2 \equiv x^2 + 2^k x + 2^{2k-2} \equiv x^2 + 2^k x \equiv x^2 + 2^k \mod 2^{k+1}$$

(pour la dernière équivalence, voir que a impair $\implies$ x impair).
Si $x^2 \equiv a \mod 2^{k+1}$, alors

$$(x + 2^{k-1})^2 \equiv a + 2^k \not\equiv a \mod 2^{k+1}.$$

Si $x^2 \not\equiv a \mod 2^{k+1}$, alors, comme $x^2 \equiv a \mod 2^k$, on a

$$x^2 \equiv a + 2^k \mod 2^{k+1}.$$

Puis,

$$(x + 2^{k-1})^2 \equiv a + 2^k + 2^k \equiv a \mod 2^{k+1}.$$

■

Chapitre 4

Factorisation de polynômes

Ce chapitre est dédié à la factorisation de polynômes à une variable dans différentes structures algébriques : les corps finis dans la Section 4.1, les rationnels et les entiers dans la Section 4.2.

4.1 Factorisation de polynômes dans les corps finis

Soit $\mathbb{F}_q[X]$ l'algèbre des polynômes à une variable sur le corps fini $\mathbb{F}_q$. Posons d'abord le problème de la factorisation de tels polynômes.

Définition 4.1 (*Factorisation en polynômes irréductibles*)

Soit $P \in \mathbb{F}_q[X]$ de degré $n \geq 1$. Une factorisation de P en polynômes irréductibles est une écriture

$$P = \alpha \prod_{i=1}^{k} P_i^{m_i}$$

où les $P_i \in \mathbb{F}_q[X]$ sont unitaires, irréductibles et deux-à-deux distincts, et où les m_i sont des entiers strictement positifs. Pour tout $i \in \{1, \ldots, k\}$, l'entier $m_i \geq 1$ est appelé la multiplicité du facteur P_i. L'élément $\alpha \in \mathbb{F}_q$ est appelé le coefficient dominant de P.

Remarque 4.2

L'anneau $\mathbb{F}_q[X]$ est factoriel, donc toute factorisation en polynôme irréductible est unique à l'ordre près des facteurs.

L'objectif de cette section est de calculer efficacement une factorisation en polynômes irréductibles, autrement dit de résoudre le problème suivant.

Problème 3

Étant donné un polynôme $P \in \mathbb{F}_q[X]$ de degré $n \geq 1$, trouver une séquence $[(P_1, m_1), \ldots, (P_k, m_k)]$ où les $P_i \in \mathbb{F}_q[X]$ sont unitaires, irréductibles et deux-à-deux distincts, les m_i sont des entiers strictement positifs, et un élément $\alpha \in \mathbb{F}_q$ tels que

$$P(X) = \alpha \prod_{i=1}^{k} P_i(X)^{m_i}$$

Si l'on dispose d'un algorithme résolvant le Problème 3, on mesurera sa complexité en fonction de la taille de l'entrée, qui ici est $O(n \log q)$. En effet, le polynôme $P(X)$ est constitué de $n + 1$

polynômes dont les coefficients sont de taille $\log q$ bits. Notons que la taille de la sortie est également en $O(n \log q)$.

Pour évaluer cette complexité, nous noterons $\mathsf{M}(n)$ une borne supérieure sur la complexité de la multiplication de deux polynômes de degré $\leq n$, en nombre d'opérations élémentaires sur $\mathbb{F}_q$ (qui sont de complexité polynomiale en $\log q$). Alors, on rappelle que :

— $\mathsf{M}(n) = \omega(n)$ en général, $\mathsf{M}(n) = O(n^2)$ pour la multiplication naïve, $\mathsf{M}(n) = O(n^{1,59})$ grâce à l'algorithme de Karatsuba et $\mathsf{M}(n) = O(n \log n)$ avec des algorithmes avancés dans certains corps finis,
— la division euclidienne d'un polynôme de degré n par un polynôme de degré $\leq n$ a un coût $O(\mathsf{M}(n))$,
— les additions et produits modulo un polynôme de degré n ont un coût $O(\mathsf{M}(n))$,
— l'algorithme d'Euclide évalué sur deux polynôme de degré $\leq n$ requiert $O(n^2)$ opérations, mais il existe des améliorations pour calculer un pgcd étendu (donc une inversion odulaire) en $O(\mathsf{M}(n) \log n)$ opérations.

Dans la suite, nous aurons besoin des notions de *facteur propre* et de *polynôme sans carré*.

Définition 4.3

Soient P, Q deux polynômes. On dit que Q est un facteur propre *(ou* diviseur propre *de P si Q divise P et* $\deg(Q) \notin \{0, \deg P\}$.

Définition 4.4

Soient $P \in \mathbb{F}_q[X]$. On dit que le polynôme P est sans carré *si la multiplicité m_i de tous ses facteurs irréductibles P_i est égale à 1. Autrement dit, P est un* sans-carré *si pour tout facteur propre Q de P, le polynôme Q^2 ne divise pas P.*

Stratégie de résolution. Pour résoudre le Problème 3, nous allons adopter la stratégie suivante :

1. extraire le coefficient dominant α de P,
2. extraire de P sa partie sans-carré $\widetilde{P}$,
3. extraire de $\widetilde{P}$ un facteur propre de P (étape à itérer),
4. calculer la multiplicité des facteurs irréductibles obtenus.

La première étape de la stratégie de résolution est immédiate : il suffit de lire le coefficient de plus haut degré de P.

De même, la quatrième étape s'effectue efficacement : si P_i est un facteur irréductible de P, il suffit (par exemple) de chercher sa multiplicité m_i en divisant successivement P par P_i. Pour ce faire, notons que tester si un polynôme Q divise un polynôme P correspond à effectuer une division euclidienne entre ces deux polynômes, ce qui se réalise donc en $O(\mathsf{M}(n) \log n)$ opérations sur $\mathbb{F}_q$. On obtient donc une complexité en pire cas en $O(n\mathsf{M}(n) \log n)$ opérations sur $\mathbb{F}_q$ pour cette quatrième étape.

4.1.1 Extraction de la partie sans carré

Problème 4 (*Extraction de la partie sans carré d'un polynôme sur les corps finis*)

Étant donné un polynôme $P = \prod_{i=1}^{k} P_i^{m_i} \in \mathbb{F}_q[X]$ où les P_i sont unitaires, irréductibles et deux-à-deux distincts, calculer la partie sans carré $\widetilde{P} = \prod_{i=1}^{k} P_i$ de P.

Dans le but d'extraire la partie sans carré d'un polynôme P, une idée importante est de comparer P avec son polynôme dérivé P' par un calcul de pgcd.

En effet, écrivons

$$P'(X) = \sum_{i=1}^{k} m_i P_i'(X) \frac{P(X)}{P_i(X)} \tag{4.1}$$

et plaçons-nous d'abord dans le cas favorable où, pour tout $i \in \{1, \ldots, k\}$, l'entier $m_i \neq 0$ dans $\mathbb{F}_q$ (autrement dit, la caractéristique p de $\mathbb{F}_q$ ne divise aucun des m_i). On note alors que pour tout $j \in \{1, \ldots, k\}$, le polynôme $P_j(X)^{m_j - 1}$ divise chacun des quotients $\frac{P(X)}{P_i(X)}$, mais que $P_j(X)^{m_j}$ ne divise pas $P_i'(X)\frac{P(X)}{P_i(X)}$ lorsque $i = j$. Il résulte que

$$\operatorname{pgcd}(P(X), P'X)) = \prod_{i=1}^{k} P_i(X)^{m_i - 1}\,.$$

On a alors (dans ce cas favorable) :

$$\widetilde{P}(X) = \frac{P(X)}{\operatorname{pgcd}(P(X), P'X))}\,.$$

Dans le cas où $p := \operatorname{car}(\mathbb{F}_q)$ divise l'un des m_i, la situation est un peu plus complexe. Notons

$$I := \{i \in \{1, \ldots, k\},\ p \text{ divise } m_i\}\,.$$

Pour $i \in I$, on a $m_i P_i'(X)\frac{P(X)}{P_i(X)} = 0$ dans $\mathbb{F}_q[X]$, donc le polynôme $P_i(X)^{m_i}$ divise tous les termes de la somme définissant $P'(X)$ dans l'équation (4.1). Par conséquent, $P_i(X)^{m_i}$ divise également $\operatorname{pgcd}(P(X), P'(X))$.

D'un autre côté, si $i \notin I$, alors (comme précédemment) le polynôme $P_i(X)^{m_i - 1}$ divise $\operatorname{pgcd}(P(X), P'(X))$. Le résultat suivant résume la situation.

Lemme 4.5

Soit $P(X) = \prod_{i=1}^{k} P_i(X)^{m_i} \in \mathbb{F}_q[X]$ et $I = \{i \in \{1, \ldots, k\},\ p$ divise $m_i\}$ où $p = \operatorname{car}(\mathbb{F}_q)$. Alors :

$$\operatorname{pgcd}(P(X), P'(X)) = \prod_{i=1}^{k} P_i^{m_i - \delta_i}$$

où $\delta_i = 0$ si $i \in I$ et $\delta_i = 1$ sinon.

Par conséquent, dans le cas général le polynôme

$$U(X) := \frac{P(X)}{\operatorname{pgcd}(P(X), P'(X))} = \prod_{i \notin I} P_i(X)\,.$$

n'est pas nécessairement la partie sans carré de $P(X)$. C'en est néanmoins un diviseur, il reste donc à déterminer $\prod_{i \in I} P_i(X)$.

Pour l'obtenir, calculons d'abord

$$V(X) := \frac{P(X)}{\operatorname{pgcd}(U(X)^{\deg P}, P(X))} = \frac{P(X)}{\prod_{i \notin I} P_i(X)^{m_i}} = \prod_{i \in I} P_i(X)^{m_i}\,.$$

Comme les éléments $i \in I$ sont tels que $p \mid i$, on a :

$$V(X) = (\prod_{i=1}^{k} P_i(X)^{m_i/p})^p$$

Supposons un instant que l'on arrive à calculer $W(X) := \prod_{i=1}^{k} P_i(X)^{m_i/p}$ à partir de $V(X)$. On remarque alors que la partie sans carré de $W(X)$ est exactement $\prod_{i \in I} P_i(X)$, et que, comparé à

Algorithme 12 : Extraction de la partie sans-carré

Entrée : Un polynôme $P(X) \in \mathbb{F}_q[X]$ unitaire
Sortie : Une liste de polynômes $Q_1(X), \ldots, Q_r(X) \in FF_q[X]$ tels que le produit $Q_1(X) \ldots Q_r(X)$ est la partie sans-carré de $P(X)$

1 Calculer le polynôme dérivé $P'(X)$.
2 Calculer $D(X) = \text{pgcd}(P(X), P'(X))$.
3 Calculer $U(X) = P(X)/D(X)$.
4 Calculer $V(X) = P(X)/\text{pgcd}(U(X)^{\deg P}, P(X))$.
5 **Si** $V(X) = 1$
6 | **Retourner** la liste $[U(X)]$.
7 **Sinon**
8 | Calculer une racine p-ème $W(X)$ de $V(X)$.
9 | Effectuer un appel récursif avec $W(X)$, pour obtenir une liste L.
10 | **Retourner** $L \cup [U(X)]$.

$V(X)$, les multiplicités des polynômes P_i dans W ont été divisées par p. On peut donc réitérer le procédé (calcul de U, calcul de V, etc.) sur le polynôme W jusqu'à ce que $V = 1$. L'algorithme 12 résume formellement la méthode.

Pour terminer, il nous reste à expliquer comment on peut calculer efficacement $W(X)$ à partir de $V(X) = W(X)^p$, sur un corps fini de caractéristique p. Écrivons $W(X) = \sum_{i=0}^{d} w_i X^i$. On a alors :

$$V(X) = W(X)^p = \left(\sum_{i=0}^{d} w_i X^i \right)^p = \sum_{i=0}^{d} w_i^p X^{pi} .$$

Si $V(X) = \sum_{j=0} v_j X^j$, on note alors que seuls les v_j avec $p \mid j$ sont non-nuls, et que

$$v_{pi} = w_i^p , \quad \forall i \in \{0, \ldots, \deg(V)/p\} .$$

Pour obtenir $W(X)$, il suffit donc d'extraire des racines p-èmes dans $\mathbb{F}_q$ de caractéristique p. De manière assez similaire au calcul de racines carrées dans $\mathbb{F}_{2^k}$ (voir Proposition 3.8), cela s'effectue élégamment en calculant

$$v_{ip}^{q/p} = (w_i^p)^{q/p} = w_i^q = w_i .$$

Proposition 4.6

Il existe un algorithme déterministe permettant d'extraire la partie sans carré d'un polynôme unitaire $P(X) \in \mathbb{F}_q[X]$ de degré n en $O(n\mathsf{M}(n) \log n)$ opérations dans $\mathbb{F}_q$.

Démonstration : La méthode est décrite dans l'Algorithme 12, auquel il faut ajouter l'extraction de racine p-ème présentée ci-dessus. Concernant la complexité de l'algorithme, on a au plus n appels récursifs, chacun avec une complexité $O(\mathsf{M}(n) \log n)$ opéations dans $\mathbb{F}_q$. ■

Donnons un exemple d'exécution de l'Algorithme 12.

Exemple 4.7

Dans $\mathbb{F}_3[X]$, on cherche à obtenir la partie sans-carré du polynôme

$$P(X) = X^{12} + 2 \cdot X^{11} + X^{10} + 2 \cdot X^8 + X^7 + 2 \cdot X^5 + X^4 + 2 \cdot X^2 + X + 2 .$$

Notons que la factorisation de $P(X)$ est $(X+1)^3(X+2)(X^2+1)^4$, mais qu'elle est inconnue en entrée de l'algorithme. Déroulons maintenant l'Algorithme 12.

- *Premier appel :*

$$\begin{aligned}
P(X) &= X^{12} + 2\cdot X^{11} + X^{10} + 2\cdot X^8 + X^7 + 2\cdot X^5 + X^4 + 2\cdot X^2 + X + 2\\
P'(X) &= X^{10} + X^9 + X^7 + X^6 + X^4 + X^3 + X + 1\\
\operatorname{pgcd}(P(X), P'(X)) &= X^9 + X^6 + X^3 + 1\\
U &= \tfrac{P(X)}{\operatorname{pgcd}(P(X),P'(X))} = X^3 + 2\cdot X^2 + X + 2\\
V &= \frac{P(X)}{\operatorname{pgcd}(P(X), U(X)^{\deg P})} = X^3 + 1
\end{aligned}$$

  On a donc obtenu une partie de la décomposition en facteurs sans carré :

$$U(X) = X^3 + 2\cdot X^2 + X + 2\,.$$

  Notons que, bien qu'on n'en ait pas connaissance dans le déroulé de l'algorithme, $U(X)$ est le produit $(X+2)(X^2+1)$.
  Comme $V(X) \neq 1$, l'algorithme ne se termine pas. Il faut calculer une racine cubique de $V(X)$. Après un calcul rapide, on obtient $V(X) = (X+1)^3$.
- *On relance l'algorithme avec le nouveau polynôme $P(X) = X + 1$, ce qui donne :*

$$\begin{aligned}
P(X) &= X + 1\\
P'(X) &= 1\\
\operatorname{pgcd}(P(X), P'(X)) &= 1\\
U(X) &= X + 1\\
V(X) &= 1
\end{aligned}$$

  Comme $V(X) = 1$, l'algorithme s'arrête. Notons que $X + 1$ est irréductible et on obtient le dernier facteur $X + 1$.

La liste renvoyée est $[X^3 + 2\cdot X^2 + X + 2, X + 1]$, et le produit des éléments de cette liste forme bien la partie sans-carrée de $X^{12} + 2\cdot X^{11} + X^{10} + 2\cdot X^8 + X^7 + 2\cdot X^5 + X^4 + 2\cdot X^2 + X + 2$.

4.1.2 Factorisation de la partie sans carré : algorithme de Berlekamp

Après avoir obtenu le coefficient dominant et la partie sans carré du polynôme $P(X) \in \mathbb{F}_q[X]$ à factoriser, intéressons-nous à l'extraction de facteurs propres.

Problème 5

Étant donné un polynôme $P(X)\prod_{i=1}^{k} P_i(X) \in \mathbb{F}_q[X]$ où les $P_i(X)$ sont unitaires, irréductibles dans $\mathbb{F}_q[X]$ et distincts, trouver les polynômes $P_i(X)$.

On suppose donc maintenant que $P(X) = \prod_{i=1}^{k} P_i$ où les P_i sont unitaires, irréductibles dans $\mathbb{F}_q[X]$ et distincts. On note également $n = \deg P$.

Pour obtenir un facteur propre de P, une idée est de reformuler le problème de la façon suivante : dans l'anneau-quotient $\mathbb{F}_q[X]/(P)$, un facteur propre de P correspond à un élément Q tel qu'il existe R vérifiant $QR = 0 \mod P$. Autrement dit, on cherche des diviseurs de 0 dans $\mathbb{F}_q[X]/(P)$.

Cela nous amène donc à étudier la structure de l'anneau $\mathbb{F}_q[X]/(P)$.

Proposition 4.8

L'anneau quotient $\mathbb{F}_q[X]/(P)$:

1. *est aussi un espace vectoriel (donc une $\mathbb{F}_q$-algèbre) de dimension n sur $\mathbb{F}_q$;*
2. *se décompose comme*

$$\mathbb{F}_q[X]/(P) \simeq \Big(\mathbb{F}_q[X]/(P_1)\Big) \times \cdots \times \Big(\mathbb{F}_q[X]/(P_k)\Big)$$

où les $\mathbb{F}_q[X]/(P_i)$ sont des extensions du corps $\mathbb{F}_q$, de degré $\deg P_i$.

Démonstration : Laissée en exercice. ■

Pour factoriser P, Berlekamp propose d'étudier l'endomorphisme

$$\begin{array}{rccc} \phi : & \mathbb{F}_q[X]/(P) & \to & \mathbb{F}_q[X]/(P) \\ & A & \mapsto & A^q - A \end{array}$$

L'isomorphisme $\mathbb{F}_q[X]/(P) \simeq \mathbb{F}_q[X]/(P_1) \times \cdots \times \mathbb{F}_q[X]/(P_k)$ induit un endormorphisme $\widetilde{\phi}$ de $\mathbb{F}_q[X]/(P_1) \times \cdots \times \mathbb{F}_q[X]/(P_k)$. Étudions le noyau de $\widetilde{\phi}$.

Lemme 4.9

On a

$$\ker \widetilde{\phi} = \{(\lambda_1, \ldots, \lambda_k) \mid \lambda_i \in \mathbb{F}_q\}.$$

Par conséquent, le noyau de ϕ a dimension k.

Démonstration : Si $A \in \mathbb{F}_q[X]/(P)$, alors on a

$$\phi(A) = 0 \iff \forall i,\ A^q \equiv A \mod P_i$$

Or, dans le corps $\mathbb{F}_q[X]/(P_i)$ de cardinal $q^{\deg P_i}$, les solutions de $x^q = x$ sont exactement les éléments du sous-corps $\mathbb{F}_q$. Donc, $\widetilde{\phi}$ s'annule exactement sur les éléments de la forme $(\lambda_1, \ldots, \lambda_k)$ où $\lambda_j \in \mathbb{F}_q$. ■

Remarque 4.10

Notons que $\ker \phi$ *est une sous-algèbre de $\mathbb{F}_q[X]/(P)$. On l'appelle* algèbre de Berlekamp *associée à P.*

Fixons maintenant une base monomiale de $\mathbb{F}_q[X]/(P)$: si α est la classe de X modulo P, une telle base s'écrit $\{1, \alpha, \ldots, \alpha^{n-1}\}$. Tout élement $B(\alpha) \in \ker \phi$ peut donc être identifié à un polynôme $B \in \mathbb{F}_q[X]$ de degré $< n$.

Theorème 4.11 (*Berlekamp*)

Si $P(X) = \prod_{i=1}^{k} P_i(X) \in \mathbb{F}_q[X]$ est un produit de polynômes irréductibles distincts, et $B(X) \in \mathbb{F}_q[X]$ est un polynôme de degré $< n$ tel que $B(X)^q \equiv B(X) \mod P(X)$, alors :

$$P(X) = \prod_{\lambda \in \mathbb{F}_q} \operatorname{pgcd}(P(X), B(X) - \lambda).$$

Démonstration : Soit $\phi : A \mapsto A^q - A \mod P$. D'après le Lemme 4.9, l'élément $B(\alpha) \in \ker \phi$ s'écrit $(\beta_1, \ldots, \beta_k)$ dans $\mathbb{F}_q[X]/(P_1) \times \cdots \times \mathbb{F}_q[X]/(P_k)$ avec $\beta_i \in \mathbb{F}_q$ pour tout $i \in \{1, \ldots, k\}$.
Fixons $i \in \{1, \ldots, k\}$. D'après ce qui précède, le polynôme $P_i(X)$ divise $B(X) - \beta_i$ dans $\mathbb{F}_q[X]$. Fixons maintenant $\lambda \in \mathbb{F}_q$ et notons $D_\lambda(X) := \operatorname{pgcd}(P(X), B(X) - \lambda)$. On remarque alors que

$$P_i \mid D_\lambda \iff \beta_i = \lambda.$$

En effet :

- Si $\lambda = \beta_i$, alors $P_i(X)$ divise $B(X) - \beta_i = B(X) - \lambda$, donc $P_i(X)$ divise aussi $D_\lambda(X)$.
- Si $P_i(X)$ divise $D_\lambda(X)$, alors $P_i(X)$ divise $B(X) - \lambda$, ce qui signifie que β_i, défini comme $B(X) \bmod P_i(X)$, vaut λ.

On en déduit :

$$D_\lambda(X) = \prod_{i|\beta_i=\lambda} P_i(X) \qquad \text{puis} \qquad P(X) = \prod_{\lambda\in\mathbb{F}_q} D_\lambda(X)\,.$$

■

Analysons les conséquences pratiques du Théorème de Berlekamp.

D'abord, notons que l'on peut efficacement calculer un polynôme $B(X) \in \mathbb{F}_q[X]$ de degré $1 \leq \deg B < n$ tel que $B(\alpha) \in \ker \phi$. Pour cela, il suffit de calculer un élément du noyau de la matrice de ϕ dans la base $\{1, \alpha, \ldots, \alpha^{n-1}\}$.

Puis, comme $\deg B < n$, deux au moins des $D_\lambda = \text{pgcd}(P, B - \lambda)$ sont des facteurs propres de P. Une idée est donc énumérer $\mathbb{F}_q$ (ou y tirer aléatoirement des éléments) pour obtenir des diviseurs propres de P. On obtient alors l'Algorithme 13.

Algorithme 13 : Algorithme de Berlekamp en petite cardinalité

Entrée : un polynôme $P = \prod_{i=1}^{k} P_i$ sans facteur carré
Sortie : au moins un diviseur propre Q de P

1 Calculer la matrice $\boldsymbol{M}$ de ϕ dans la base polynomiale $(1, \alpha, \ldots, \alpha^{n-1})$.
2 Calculer un élément non-nul du noyau de $\boldsymbol{M}$ de la forme $(b_0 = 0, \ldots, b_{n-1})$.
3 Construire $B(X) = \sum_{j=0}^{n-1} b_j X^j$.
4 **Faire**
5 Choisir un nouveau $\lambda \in \mathbb{F}_q$ — de façon déterministe (énumération) ou probabiliste.
6 Calculer $Q(X) = \text{pgcd}(P(X), B(X) - \lambda)$.
7 **tant que** $Q(X) = 1$;
8 **Retourner** Q.

Lemme 4.12

L'Algorithme 13 produit un facteur propre du polynôme sans carré P en $O(n^3 + q\mathsf{M}(n)\log n)$ opérations sur $\mathbb{F}_q$ en pire cas.

Démonstration : Une grande partie de la preuve de validité a été faite avant la présentation de l'algorithme. Justifions simplement le choix de la forme spécifique pour le vecteur $\boldsymbol{b}$. L'idée est qu'il faut prendre $B(X)$ de degré ≥ 1, car sinon les $D_\lambda = \text{pgcd}(P, B - \lambda)$ seront constants, sauf pour $B = \lambda$ auquel cas $D_\lambda = P$.
Pour la complexité, le calcul de la matrice $\boldsymbol{M}$ et d'un élément de son noyau requiert $O(n^3)$ opérations sur $\mathbb{F}_q$. Une fois le vecteur $\boldsymbol{b}$ obtenu, le calcul d'un pgcd nécessite $O(\mathsf{M}(n)\log n)$ opérations sur $\mathbb{F}_q$, et doit être effectué au plus $q - 1$ fois. ■

Exemple 4.13

On reprend l'Exemple 4.7, c'est-à-dire $q = 3$, et $P = X^{12} + 2\cdot X^{11} + X^{10} + 2\cdot X^8 + X^7 + 2\cdot X^5 + X^4 + 2\cdot X^2 + X + 2$, où l'on avait trouvé la partie sans carré $Q = X^4 + 2$.
Comme prescit par l'Algorithme 13, on calcule d'abord la matrice des $X^{3i} - X^i \bmod Q$ dans la base $(1, X, X^2, X^3)$:

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

On calcule ensuite un $B \in \ker \boldsymbol{M}$ non-constant aléatoire, disons $B(X) = X^2 + 2$. On obtient alors :

λ	$D_\lambda = \text{pgcd}(Q, B - \lambda)$
0	$X^2 + 2$
1	$X^2 + 1$
2	1

> *On a donc trouvé deux diviseurs propres de Q : X^2+2 et X^2+1. On pourrait ensuite appeler récursivement l'algorithme sur ces deux polynômes pour obtenir la factorisation définitive...*

On remarque que la complexité de l'Algorithme 13 reste exponentielle en $\log q$. Il doit donc être utilisé seulement lorsque q est petit.

Lorsque q est trop grand pour être énuméré, l'idée est de regrouper certains diviseurs de la forme $\text{pgcd}(P, B-\lambda)$, en un seul calcul de pgcd. Pour cela, remarquons que si q est impair, alors pour tout $\beta \in \mathbb{F}_q$:

1. ou bien $\beta = 0$,
2. ou bien β est un carré non-nul dans $\mathbb{F}_q$, auquel cas $\beta^{(q-1)/2} - 1 = 0$,
3. ou bien β est un non-carré dans $\mathbb{F}_q$, auquel cas $\beta^{(q-1)/2} + 1 = 0$.

En considérant $\beta = B \mod P_i$, on peut donc considérer trois sous-ensembles d'indices :

1. $I_1 := \{i \in \{1,\dots,k\} \mid B \bmod P_i = 0\}$,
2. $I_2 := \{i \in \{1,\dots,k\} \mid B^{(q-1)/2} - 1 \bmod P_i = 0\}$,
3. $I_3 := \{i \in \{1,\dots,k\} \mid B^{(q-1)/2} + 1 \bmod P_i = 0\}$.

Ensuite, on peut démontrer[1] qu'avec une probabilité $\geq 1/2$ sur le choix de $B(X)$, deux des ensembles I_1, I_2 et I_3 sont non-vides. Dans ce cas (favorable), on obtient donc une factorisation non-triviale

$$P = \text{pgcd}(P, B) \times \text{pgcd}(P, B^{(q-1)/2} - 1) \times \text{pgcd}(P, B^{(q-1)/2} + 1)\,.$$

car 2 de ces facteurs sont des facteurs propres de P.

Algorithme 14 : Algorithme de Berlekamp en grande cardinalité, avec $\text{car}(\mathbb{F}_q) \neq 2$.

Entrée : un polynôme $P = \prod_{i=1}^{k} P_i$ sans facteur carré
Sortie : au moins deux diviseurs propres de P

1 Calculer la matrice $\boldsymbol{M}$ de ϕ dans la base polynomiale $(1, \alpha, \dots, \alpha^{n-1})$.
2 **Faire**
3 Calculer un aléatoirement un élément non-nul du noyau de $\boldsymbol{M}$ de la forme $(b_0 = 0, \dots, b_{n-1})$.
4 Construire $B(X) = \sum_{j=0}^{n-1} b_j X^j$.
5 Calculer $B^{(q-1)/2} - 1 \bmod P$ et $B^{(q-1)/2} + 1 \bmod P$.
6 Calculer $A_0 = \text{pgcd}(P, B)$, $A_1 = \text{pgcd}(P, B^{(q-1)/2} - 1)$ et $A_2 = \text{pgcd}(P, B^{(q-1)/2} + 1)$.
7 **tant que** *l'un des A_i vaut P*;
8 Retourner les A_i différents de 1.

Lemme 4.14

L'Algorithme 14 produit, de manière probabiliste, un facteur propre du polynôme sans carré P en $O(n^3 + \mathsf{M}(n)\log n \log q)$ opérations sur $\mathbb{F}_q$ en moyenne.

4.1.3 Factorisation en degré distincts

Pas vu en cours, à venir

4.1.4 Factorisation à degré égal

Pas vu en cours, à venir

1. démonstration à venir

4.2 Factorisation de polynômes dans les rationnels et les entiers

Bibliographie

[KAF+10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman J. J. te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit RSA modulus. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 333–350. Springer, 2010.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (3. ed.)*. Cambridge University Press, 2013.