# Algorithmes arithmétiques II
# Cours 1

Julien Lavauzelle

Université Paris 8

Master 2 ACC – Algorithmes arithmétiques II

22/09/2021

# Plan

Pour résoudre un système linéaire $\boldsymbol{Ax} = \boldsymbol{b}$, où $\boldsymbol{A} \in \mathbb{F}^{n \times n}$, dans un cas général :

1. on effectue une **élimination gaussienne** pour rendre le système échelonné :

$$\boldsymbol{UAP} = \boldsymbol{T}, \quad \text{avec } \boldsymbol{T} \text{ échelonnée}$$

2. on calcule une base du noyau à droite $\mathcal{K} \subseteq \mathbb{F}^n$ de la matrice $\boldsymbol{T}$
3. on calcule une solution particulière $\boldsymbol{y}^{(0)}$ de $\boldsymbol{Ty} = \boldsymbol{Ub}$
4. on retourne une description de l'ensemble des solutions :

$$\boldsymbol{P}(\boldsymbol{y}^{(0)} + \mathcal{K})$$

**Complexité** totale de la méthode : $O(n^3)$.

# Rappels : suites récurrentes

Soit $\boldsymbol{u} \in \mathbb{F}^{\mathbb{N}}$ une suite.

Si $P(X) = \sum_{i=0}^{d} c_i X^i$, alors $(P, L)$ est un LFSR qui engendre la suite sur $N$ termes s'il existe $(u_0, \dots, u_{L-1})$ tel que :

$$u_n + c_1 u_{n-1} + \cdots + c_d u_{n-d}, \quad \forall L \leq n < N$$

On dit que $P$ est un polynôme de connexion de la suite.

**Objectif :** étant donnée une suite $\boldsymbol{u}$, trouver un LFSR de dimension $L$ minimale qui engendre $\boldsymbol{u}$ sur $N$ termes.

# Algorithme de Berlekamp–Massey

**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** $k \in \{i, \dots, N-1\}$ **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 $((P_k, L_k))$.

# Plan

# Motivation

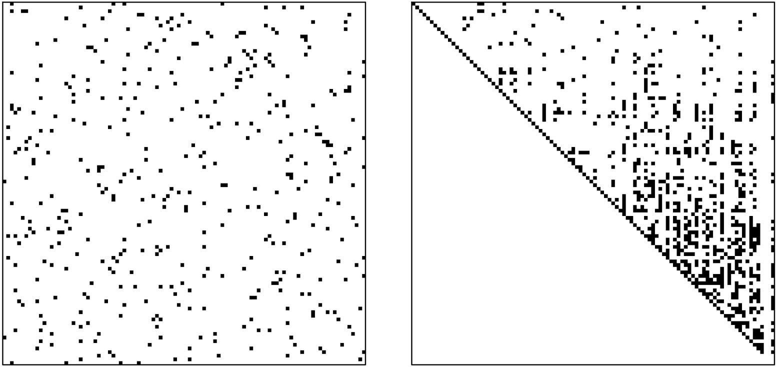
**Cas concrets de matrices creuses :**

- matrices d'adjacence de graphes (réseaux, big data)
- résolution numérique d'équations aux dérivées partielles
- étape d'algèbre linéaire dans le calcul de logarithmes discrets ou dans la factorisation d'entiers

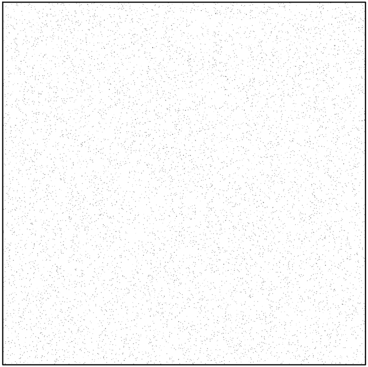
**Exemple « extrême »** (mais réel) : dans la factorisation de RSA-768 en 2009

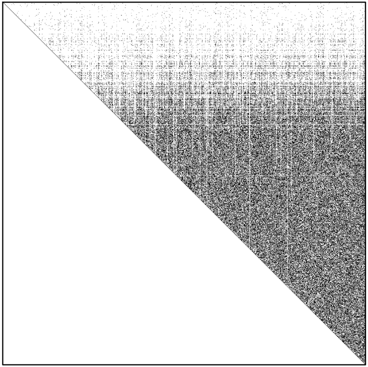
- $n \simeq m \geq 192\,000\,000 \simeq 2^{27}$ lignes/colonnes,
- $27\,000\,000\,000 \simeq 2^{34}$ coefficients non-nuls,
- si stockage creux : 105 Go,
- si stockage dense : $>$ 4000 To (impossible).

Sur le corps $\mathbb{F}_2$, pour $n = 100$ et $t = 4$ :



Sur le corps $\mathbb{F}_2$, pour $n = 1000$ et $t = 10$ :





# Calcul du polynôme annulateur d'une suite vectorielle

**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 $d' := d - \deg \mu_{\boldsymbol{a}}(X)$.

# Algorithme de Wiedemann

**Entrée :** une matrice creuse $A \in \mathbb{F}^{n \times n}$
**Sortie :** un élément aléatoire du noyau de $A$

Calculer le polynôme minimal $\mu_A(X)$ de la matrice $A$, et définir $Q(X) = \mu_A(X)/X$.
Assigner $x \leftarrow \mathbf{0}$.
**Tant que** $x = \mathbf{0}$ **faire**
  Tirer aléatoirement $u \in \mathbb{F}^n \setminus \{\mathbf{0}\}$.
  Calculer $x = Q(A)u$.
**Retourner** $x$.

# Calcul du polynôme annulateur d'une matrice

**Entrée :** une matrice creuse $\boldsymbol{A} \in \mathbb{F}^{n \times n}$
**Sortie :** le polynôme minimal $\mu_{\boldsymbol{A}}(X)$ de $\boldsymbol{A}$

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

# Pour aller plus loin : algorithme de Wiedemann par blocs

**En pratique,** une idée largement développée est de regrouper les opérations par blocs.

**Deux objectifs :**

- Faire décroître la probabilité d'échec de l'algorithme
- Accélérer les calculs (instructions machine spécifiques)

**Idée :** au lieu de raisonner sur des scalaires $a_k = \boldsymbol{v}^\top \boldsymbol{A}^k \boldsymbol{w}$, on utilise de (petites) matrices

$$\boldsymbol{M}_k = \boldsymbol{V}^\top \boldsymbol{A}^k \boldsymbol{W} \in \mathbb{F}^{\alpha \times \beta}$$

Typiquement, $\boldsymbol{M}_k$ est de taille $64 \times 64$.

Alors :

- Chaque matrice $\boldsymbol{M}_k$ contient plus d'information que les scalaires $a_k$.
- On aura alors besoin de moins de termes $\boldsymbol{M}_0, \dots, \boldsymbol{M}_d$ pour obtenir un facteur de $\mu_{\boldsymbol{A}}$ (même si le calcul du polynôme annulateur devient plus compliqué).
- Par ailleurs, les calculs peuvent être parallélisés, car les opérations sur les colonnes des $\boldsymbol{M}_k$ sont indépendantes.

> **Remarque.** C'est ce type d'algorithme qui est utilisé pour l'étape d'algèbre linéaire lors de factorisations et log-discrets records.

# Références

Autour de l'algorithme de Wiedemann

*Solving sparse linear equations over finite fields*. Wiedemann. IEEE TIT. **1986**.

*Solving linear equations over GF(2) via block Wiedemann algorithm*. Coppersmith. Math. Comp.. **1994**.

*Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm*. Thomé. J. Symbolic Comp.. **2002**.

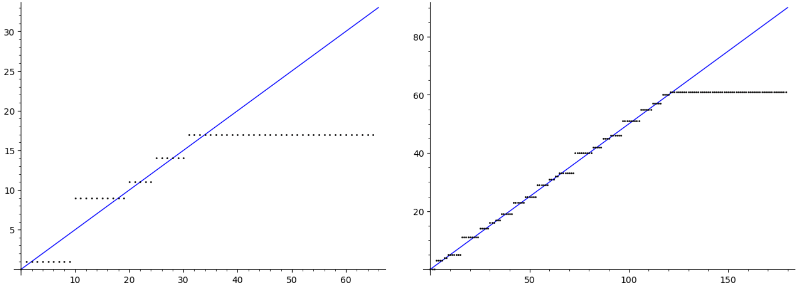
Si vous voulez aller plus loin : adaptation de la méthode de Lanczos

*Solving large sparse linear systems over finite fields*. LaMacchia, Odlyzko. CRYPTO. **1990**.

*A Block Lanczos Algorithm for Finding Dependencies over GF(2)*. Montgomery. EUROCRYPT. **1995**.

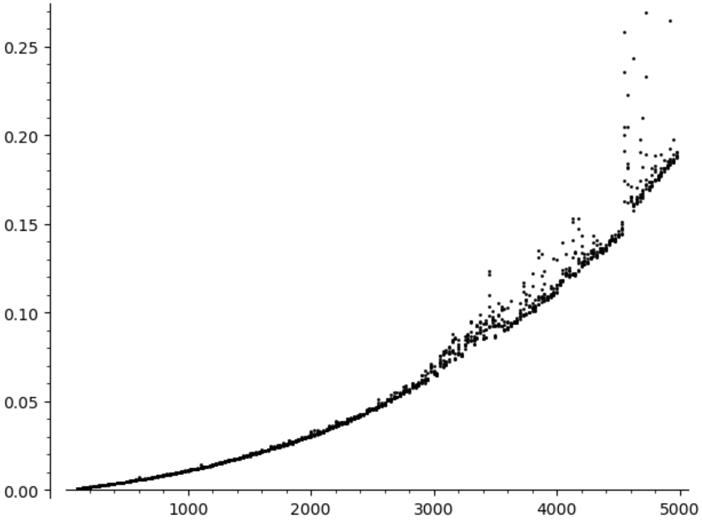
# Plan

Croissance de la dimension minimale du LFSR $\ell_k(\boldsymbol{u})$ :

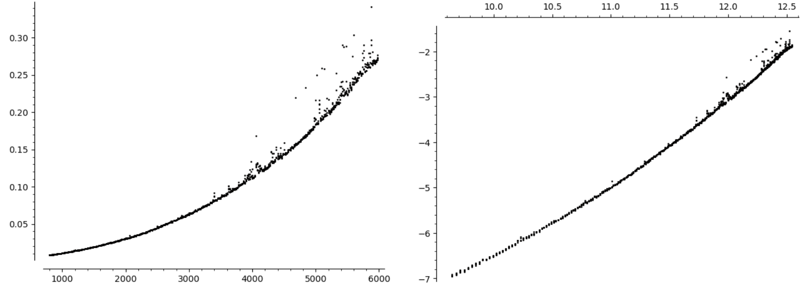



À gauche : $d = 20$,     à droite, $d = 50$.

Complexité en fonction du degré $d$ :



Comment vérifier que la complexité est quadratique (et non $d^3$, $d^4$ ou $2^d$) ?




À gauche : échelle standard, à droite, échelle log-log.