Calcul de formes normales matricielles: de l'algorithmique à la mise en pratique

Clément PERNET,

LIG/INRIA-MOAIS, Université de Grenoble, France

Séminaire SIESTE, ENS-Lyon 12 février 2013

Introduction : formes normales matricielles

Étant donné une famille de transformations $B = f_U(A)$,

- Identifier des invariants
- Représentant unique des classes d'équivalences
- Simplifier les calculs (forme structurée ou creuse)

Différent types:

Équivalence à gauche dans un corps: $B = UA$, où U est inversible

- Forme échelonnée réduite:

$$E = \begin{bmatrix} 1 & * & 0 & * & * & 0 & * \\ & & 1 & * & * & 0 & * \\ & & & & & 1 & * \\ & & & & & & \end{bmatrix}$$

- Élimination de Gauss-Jordan

Introduction : formes normales matricielles

Étant donné une famille de transformations $B = f_U(A)$,

- Identifier des invariants
- Représentant unique des classes d'équivalences
- Simplifier les calculs (forme structurée ou creuse)

Différent types:

Équivalence à gauche dans un anneau: $B = UA$, où $\det(U) = \pm 1$

- Forme de Hermite:

$$H = \begin{bmatrix} p_1 & * & x_{1,2} & * & * & x_{1,3} & * \\ & & p_2 & * & * & x_{2,3} & * \\ & & & & & p_3 & * \\ & & & & & & \end{bmatrix}, \text{ avec}$$

$0 \leq x_{*,j} < p_j$

Introduction : formes normales matricielles

Étant donné une famille de transformations $B = f_U(A)$,

- Identifier des invariants
- Représentant unique des classes d'équivalences
- Simplifier les calculs (forme structurée ou creuse)

Différent types:

Similitude dans un corps: $B = U^{-1}AU$

- Forme normale de Frobenius (ou forme canonique rationnelle): $F = \begin{bmatrix} C_{P_0} & & & \\ & C_{P_1} & & \\ & & \ddots & \\ & & & C_{P_k} \end{bmatrix}$, avec $p_{i+1} | p_i$ et $P_0 = \text{MinPoly}(A)$.
- Méthode de Krylov ou élimination ZigZag

Motivation

Équivalence dans un corps: élimination de Gauss

- Forme échelonnée réduite, profile de rang: résolution de systèmes polynomiaux par bases de Gröbner.
- Résolution de systèmes linéaires: crypto (cribles, calcul d'index, ...)

Équivalence dans un anneau: réduction de réseaux

- Forme normale de Hermite: $\mathbb{Z}$-modules et leur saturation
- calcul de vecteurs courts:
  - difficile (donc applications en crypto)
  - améliore les complexités
  - théorie des codes: décodage en liste

Complexités

Produit de matrices: accès aux algorithmes rapides

- dans un corps: $\mathcal{O}(n^\omega)$. $\omega \in]2.3727, 3]$ (exposant de l'algèbre lin.)
- dans $\mathbb{Z}$: $\mathcal{O}(n^\omega M(\log \|A\|)) = \mathcal{O}^\sim(n^\omega \log \|A\|)$

Équivalence dans un corps: forme echelonnée réduite

- Gauss-Jordan: $\mathcal{O}(n^\omega)$

Complexités

Produit de matrices: accès aux algorithmes rapides

- dans un corps: $\mathcal{O}(n^\omega)$. $\omega \in]2.3727, 3]$ (exposant de l'algèbre lin.)
- dans $\mathbb{Z}$: $\mathcal{O}(n^\omega M(\log \|A\|)) = \mathcal{O}^\sim(n^\omega \log \|A\|)$

Équivalence dans un corps: forme echelonnée réduite

- Gauss-Jordan: $\mathcal{O}(n^\omega)$

Équivalence dans $\mathbb{Z}$: forme normale de Hermite

- [Kannan & Bachem 79]: $\in P$
- [Chou & Collins 82]: $\mathcal{O}^\sim\left(n^6 \log \|A\|\right)$
- [Domich & Al. 87], [Illiopoulos 89]: $\mathcal{O}^\sim\left(n^4 \log \|A\|\right)$
- [Micciancio & Warinschi01]: $\mathcal{O}^\sim\left(n^5 \log \|A\|^2\right)$, $\mathcal{O}^\sim\left(n^3 \log \|A\|\right)$ heur.
- [Storjohann & Labahn 96]: $\mathcal{O}^\sim\left(n^{\omega+1} \log \|A\|\right)$

Complexités

Produit de matrices: accès aux algorithmes rapides

- dans un corps: $\mathcal{O}(n^\omega)$. $\omega \in]2.3727, 3]$ (exposant de l'algèbre lin.)
- dans $\mathbb{Z}$: $\mathcal{O}(n^\omega M(\log \|A\|)) = \mathcal{O}^\sim(n^\omega \log \|A\|)$

Équivalence dans un corps: forme echelonnée réduite

- Gauss-Jordan: $\mathcal{O}(n^\omega)$

Équivalence dans $\mathbb{Z}$: forme normale de Hermite

- [Kannan & Bachem 79]: $\in P$
- [Chou & Collins 82]: $\mathcal{O}^\sim\left(n^6 \log \|A\|\right)$
- [Domich & Al. 87], [Illiopoulos 89]: $\mathcal{O}^\sim\left(n^4 \log \|A\|\right)$
- [Micciancio & Warinschi01]: $\mathcal{O}^\sim\left(n^5 \log \|A\|^2\right)$, $\mathcal{O}^\sim\left(n^3 \log \|A\|\right)$ heur.
- [Storjohann & Labahn 96]: $\mathcal{O}^\sim\left(n^{\omega+1} \log \|A\|\right)$

Similitude dans un corps: forme normale de Frobenius

- [Storjohann00]: $\mathcal{O}^\sim(n^\omega)$
- [P. & Storjohann07]: Las Vegas sans U $\mathcal{O}(n^\omega)$

Complexités

Produit de matrices: accès aux algorithmes rapides

- dans un corps: $\mathcal{O}(n^\omega)$. $\omega \in]2.3727, 3]$ (exposant de l'algèbre lin.)
- dans $\mathbb{Z}$: $\mathcal{O}(n^\omega M(\log \|A\|)) = \mathcal{O}^\sim(n^\omega \log \|A\|)$

Équivalence dans un corps: forme echelonnée réduite

- Gauss-Jordan: $\mathcal{O}(n^\omega)$

Équivalence dans $\mathbb{Z}$: forme normale de Hermite

- [Kannan & Bachem 79]: $\in P$
- [Chou & Collins 82]: $\mathcal{O}^\sim\left(n^6 \log \|A\|\right)$
- [Domich & Al. 87], [Illiopoulos 89]: $\mathcal{O}^\sim\left(n^4 \log \|A\|\right)$
- [Micciancio & Warinschi01]: $\mathcal{O}^\sim\left(n^5 \log \|A\|^2\right)$, $\mathcal{O}^\sim\left(n^3 \log \|A\|\right)$ heur.
- [Storjohann & Labahn 96]: $\mathcal{O}^\sim\left(n^{\omega+1} \log \|A\|\right)$

Similitude dans un corps: forme normale de Frobenius

- [Storjohann00]: $\mathcal{O}^\sim(n^\omega)$
- [P. & Storjohann07]: Las Vegas sans U $\mathcal{O}(n^\omega)$

Motivation

Briques de base algorithmique et logicielles pour le calcul de formes normales en algèbre linéaire.

En bref

Réductions à une brique de base

MatMul : bloc rec. $\sum_{i=1}^{\log n} n \left(\frac{n}{2^i}\right)^{\omega-1} = \mathcal{O}\left(n^\omega\right)$ (Gauss)

MatMul : iteratif $\sum_{k=1}^{n} n \left(\frac{n}{k}\right)^{\omega-1} = \mathcal{O}\left(n^\omega\right)$ (Frobenius)

Linear Sys: dans $\mathbb{Z}$ (Hermite)

Compromis taille/dimension

Hermite : perturb. de rang 1 pour réduire le déterminant

Frobenius : degré k, dimension n/k pour $k = 1 \ldots n$

Motivation

Briques de base algorithmique et logicielles pour le calcul de formes normales en algèbre linéaire.

En bref

Réductions à une brique de base

MatMul : bloc rec. $\sum_{i=1}^{\log n} n \left(\frac{n}{2^i}\right)^{\omega-1} = \mathcal{O}\left(n^\omega\right)$ (Gauss)

MatMul : iteratif $\sum_{k=1}^{n} n \left(\frac{n}{k}\right)^{\omega-1} = \mathcal{O}\left(n^\omega\right)$ (Frobenius)

Linear Sys: dans $\mathbb{Z}$ (Hermite)

Compromis taille/dimension

Hermite : perturb. de rang 1 pour réduire le déterminant

Frobenius : degré k, dimension n/k pour $k = 1 \ldots n$

Mise en oeuvre dans les bibliothèques

- `FFLAS-FFPACK`: dense, $\mathbb{Z}_p$ où $p \approx$ mot machine.
- `M4RI`: dense, GF(2)
- `LinBox`: dense/creux/boîte-noire dans $\mathbb{Z}$, $\mathbb{Z}_p$

Le produit de matrices dans $\mathbb{Z}/p\mathbb{Z}$

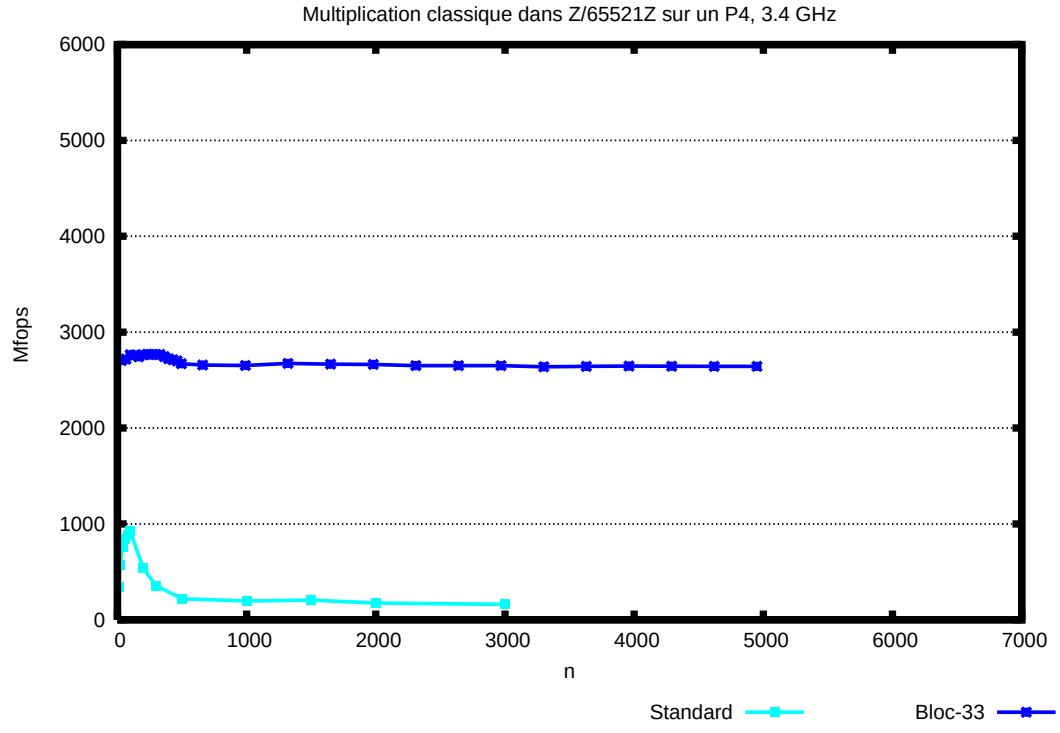
Principe:

- Réductions modulaires différées $\Rightarrow$plongement dans $\mathbb{Z}$
- Arithmétique flottante (`fma`, SSE2, avx ...)

Le produit de matrices dans $\mathbb{Z}/p\mathbb{Z}$

Principe:

- Réductions modulaires différées $\Rightarrow$plongement dans $\mathbb{Z}$
- Arithmétique flottante (`fma`, SSE2, avx ...)
- Optimisation de cache

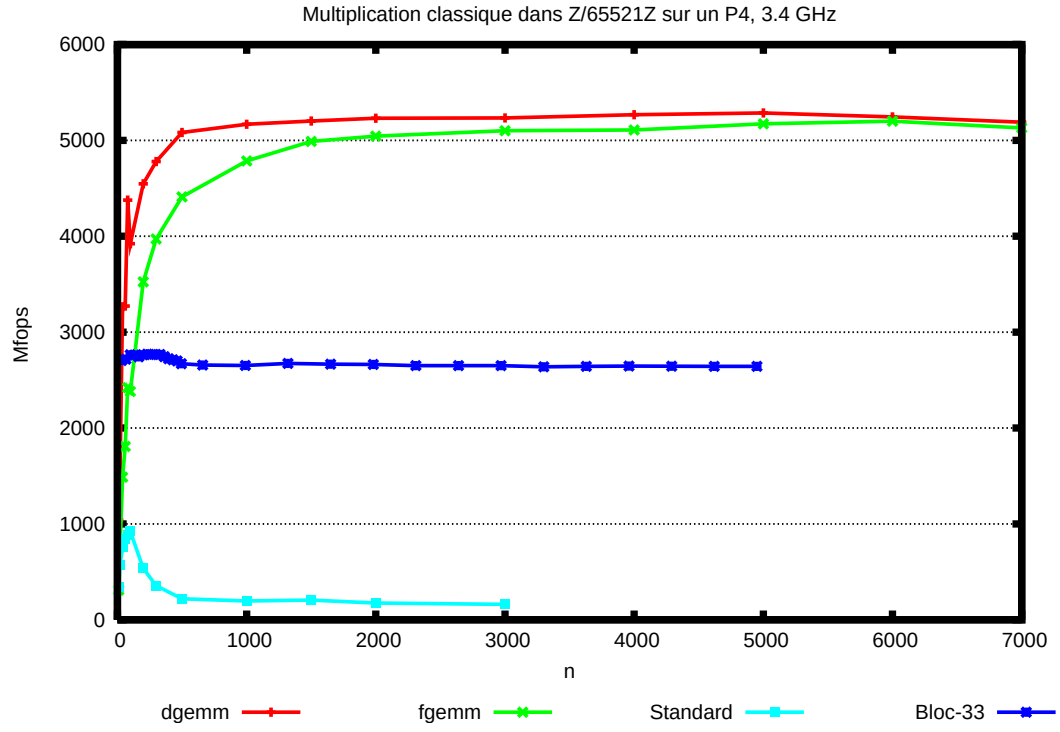


Le produit de matrices dans $\mathbb{Z}/p\mathbb{Z}$

Principe:

- Réductions modulaires différées $\Rightarrow$plongement dans $\mathbb{Z}$
- Arithmétique flottante (`fma`, SSE2, avx ...)
- Optimisation de cache

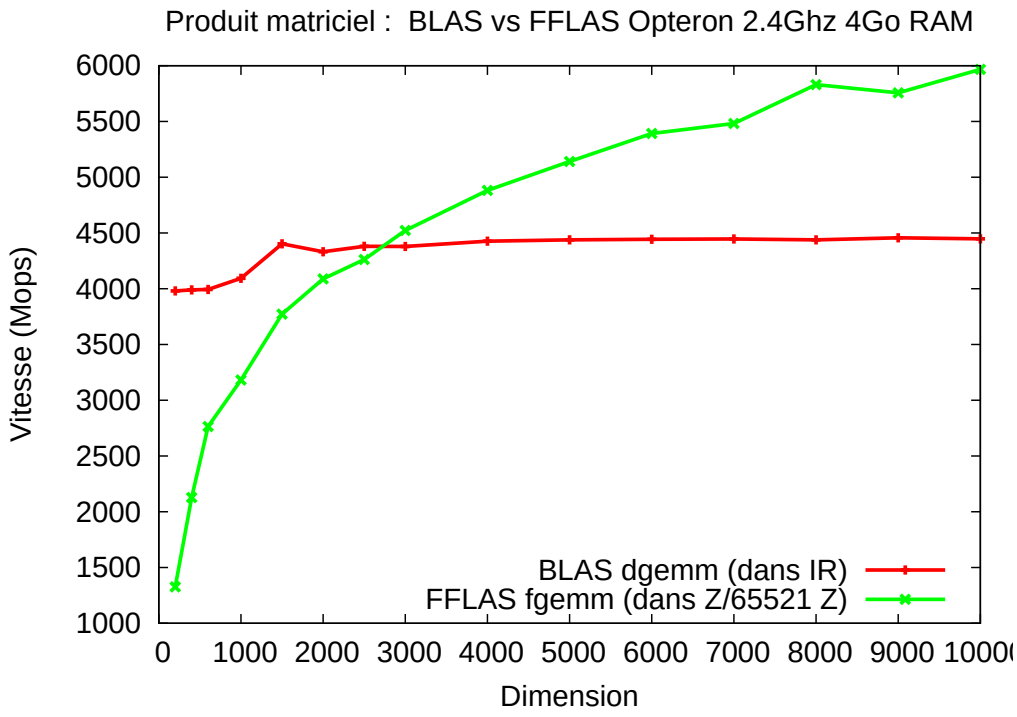
  $\Rightarrow$repose sur les BLAS existants



Le produit de matrices dans $\mathbb{Z}/p\mathbb{Z}$

Principe:

- Réductions modulaires différées $\Rightarrow$plongement dans $\mathbb{Z}$
- Arithmétique flottante (`fma`, SSE2, avx ...)
- Optimisation de cache
  $\Rightarrow$repose sur les BLAS existants
- Algorithme sous-cubique (Strassen-Winograd)



Plan

Plan

Élimination de Gauss pour la forme échelonnée réduite

Élimination de Gauss $<$ Forme échelonnée réduite

- Complétement étudiée pour le calcul numérique
- Spécificités du calcul exact:
  - pas de contrainte de stabilité (pivotage partiel ou total
  - Défaut de rang, profil de rang
- Éventuellement: taille des coefficients ($\mathbb{Z}$ ou GF(2))

Élimination de Gauss pour la forme échelonnée réduite

Élimination de Gauss $<$ Forme échelonnée réduite

- Complétement étudiée pour le calcul numérique
- Spécificités du calcul exact:
  - pas de contrainte de stabilité (pivotage partiel ou total
  - Défaut de rang, profil de rang
- Éventuellement: taille des coefficients ($\mathbb{Z}$ ou GF(2))

Definition (Profil de rang en ligne)

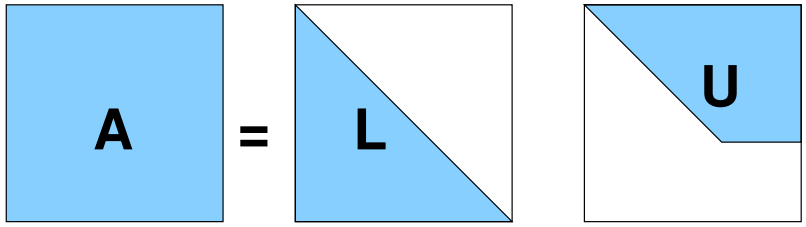
Sous séquence lexico. minimale de $r = \text{rang}(A)$ indices de lignes $\in \{1, 2, \ldots, n\}$ linéaires indépendantes.

Definition (Profil de rang générique)

Les $r = \text{rang}(A)$ mineurs dominants principaux sont non nuls.

Plan

Décomposition LU

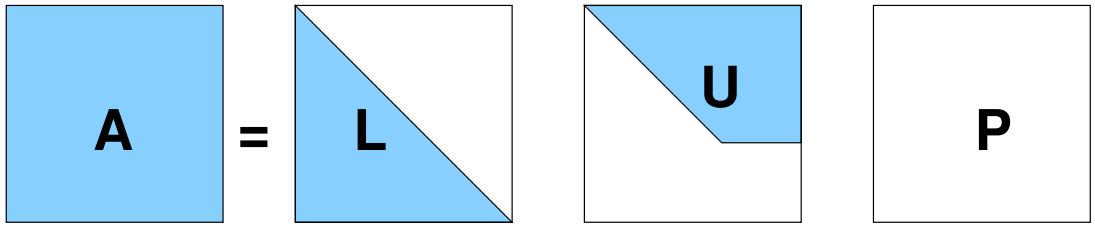


- L: triangulaire inf. unitaire
- U: triangulaire sup. inversible

Existe pour

- les matrices ayant un profil de rang générique (tout mineur principal dominant est non nul)

Décomposition LUP

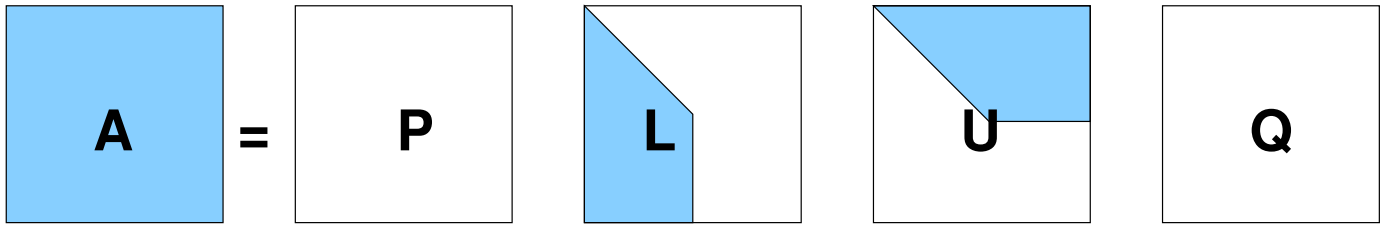


- L: triangulaire inf. unitaire
- U: triangulaire sup. inversible
- P: matrice de permutation

Existe pour

- toute matrice régulière
- ou toute matrice avec profil de rang en ligne générique

Décomposition PLUQ

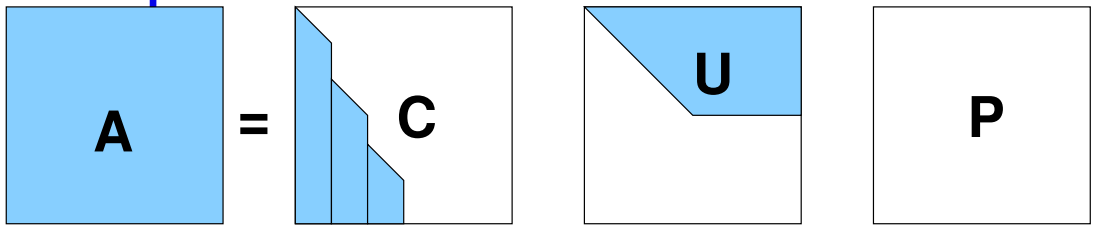


- L: triangulaire inf. unitaire
- U: triangulaire sup. inversible
- P: matrice de permutation
- Q: matrice de permutation

Existe pour

- toute matrice $m \times n$

Décompositions CUP et PLE

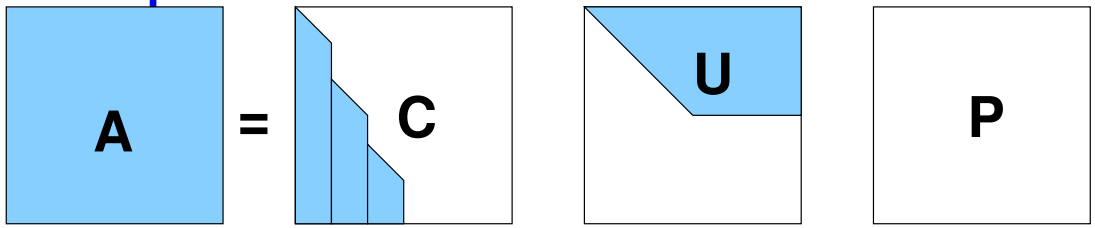


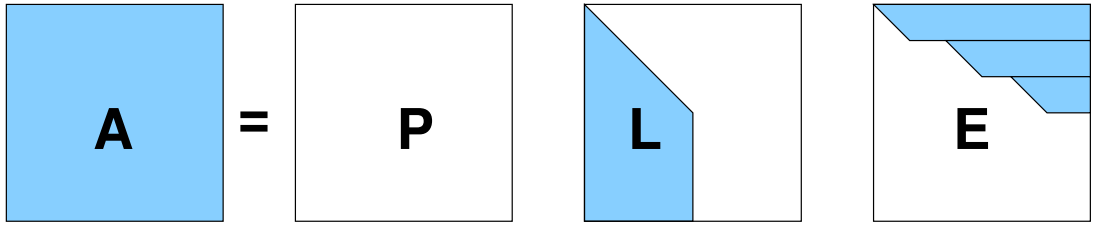
- C: forme échelonnée colonne
- E: forme échelonnée ligne

Existe pour

- toute matrice $m \times n$

Décompositions CUP et PLE



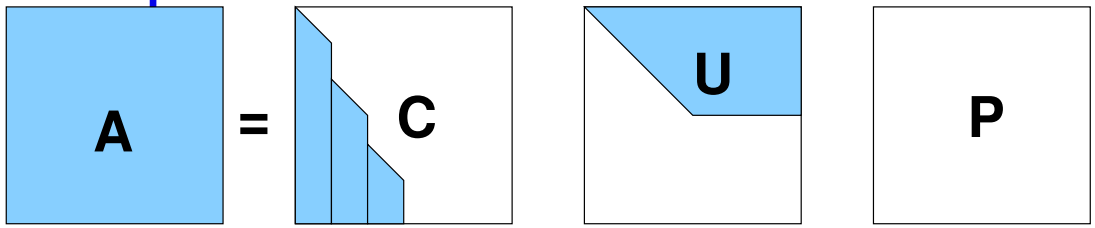


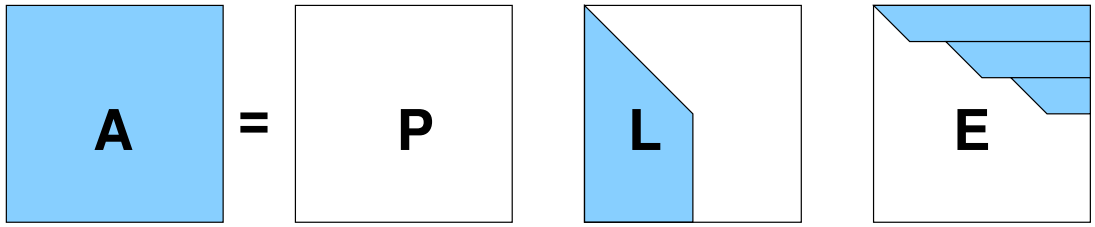
- C: forme échelonnée colonne
- E: forme échelonnée ligne

Existe pour

- toute matrice $m \times n$

Décompositions CUP et PLE





- C: forme échelonnée colonne
- E: forme échelonnée ligne

Existe pour

- toute matrice $m \times n$

Definition

- Profil de rang en ligne: indice des pivots dans C
- Profil de rang en colonne: indice des pivots dans E

Plan

Type d'algorithmes

Localité des données: algorithmes par blocs

- cache aware: itératif par blocs
- cache oblivious: récursif par blocs

Efficacité du cas de base: itératif simple

Complexité asymptotique: récursif par blocs

Parallélisation:

- processor aware: itératif par blocs
- processor oblivious: récursif par blocs

Type d'algorithmes

Localité des données: algorithmes par blocs

- cache aware: itératif par blocs
- cache oblivious: récursif par blocs

Efficacité du cas de base: itératif simple

Complexité asymptotique: récursif par blocs

Parallélisation:

- processor aware: itératif par blocs
- processor oblivious: récursif par blocs

Préliminaires

TRSM: TRiangular Solve with Matrix

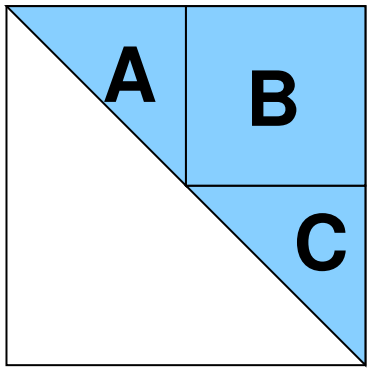
$$\begin{bmatrix} A & B \\ & C \end{bmatrix}^{-1} \begin{bmatrix} D \\ E \end{bmatrix} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & C^{-}1 \end{bmatrix} \begin{bmatrix} D \\ E \end{bmatrix}$$

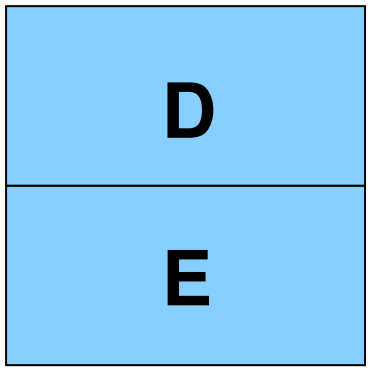
Préliminaires

TRSM: TRiangular Solve with Matrix

$$\begin{bmatrix} A & B \\ & C \end{bmatrix}^{-1} \begin{bmatrix} D \\ E \end{bmatrix} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & C^{-}1 \end{bmatrix} \begin{bmatrix} D \\ E \end{bmatrix}$$

$F = C^{-1}E$ (appel récursif)
$G = D - BF$ (MM)
$H = A^{-1}G$ (appel récursif)
Return $\begin{bmatrix} H \\ F \end{bmatrix}$



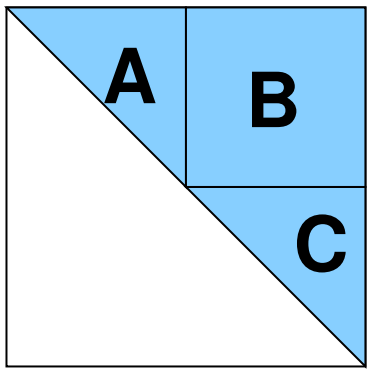


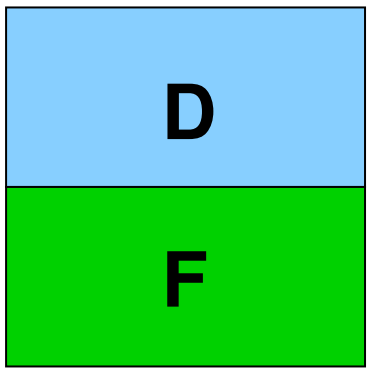
Préliminaires

TRSM: TRiangular Solve with Matrix

$$\begin{bmatrix} A & B \\ & C \end{bmatrix}^{-1} \begin{bmatrix} D \\ E \end{bmatrix} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & C^{-}1 \end{bmatrix} \begin{bmatrix} D \\ E \end{bmatrix}$$

$F = C^{-1}E$ (appel récursif)
$G = D - BF$ (MM)
$H = A^{-1}G$ (appel récursif)
Return $\begin{bmatrix} H \\ F \end{bmatrix}$



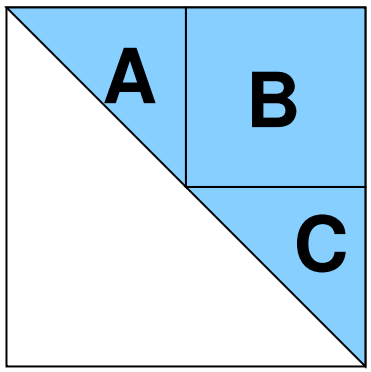


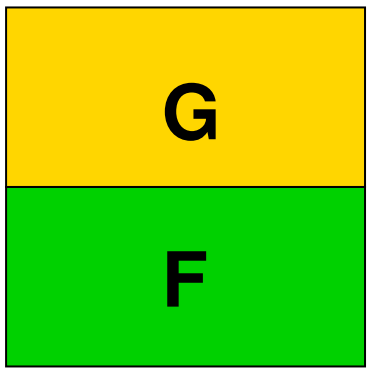
Préliminaires

TRSM: TRiangular Solve with Matrix

$$\begin{bmatrix} A & B \\ & C \end{bmatrix}^{-1} \begin{bmatrix} D \\ E \end{bmatrix} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & C^{-}1 \end{bmatrix} \begin{bmatrix} D \\ E \end{bmatrix}$$

$F = C^{-1}E$ (appel récursif)
$G = D - BF$ (MM)
$H = A^{-1}G$ (appel récursif)
Return $\begin{bmatrix} H \\ F \end{bmatrix}$



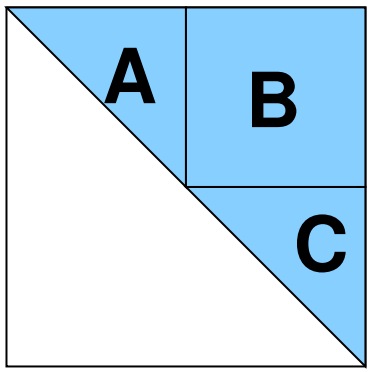


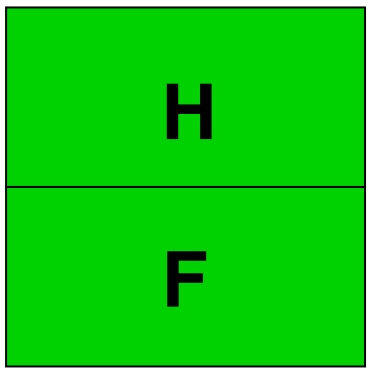
Préliminaires

TRSM: TRiangular Solve with Matrix

$$\begin{bmatrix} A & B \\ & C \end{bmatrix}^{-1} \begin{bmatrix} D \\ E \end{bmatrix} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & C^{-}1 \end{bmatrix} \begin{bmatrix} D \\ E \end{bmatrix}$$

$F = C^{-1}E$	(appel récursif)
$G = D - BF$	(MM)
$H = A^{-1}G$	(appel récursif)
Return $\begin{bmatrix} H \\ F \end{bmatrix}$	



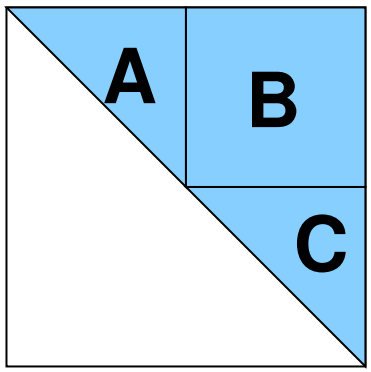


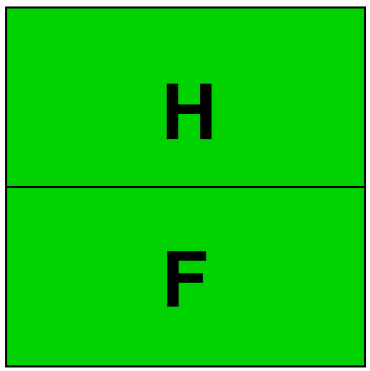
Préliminaires

TRSM: TRiangular Solve with Matrix

$$\begin{bmatrix} A & B \\ & C \end{bmatrix}^{-1} \begin{bmatrix} D \\ E \end{bmatrix} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & C^{-}1 \end{bmatrix} \begin{bmatrix} D \\ E \end{bmatrix}$$

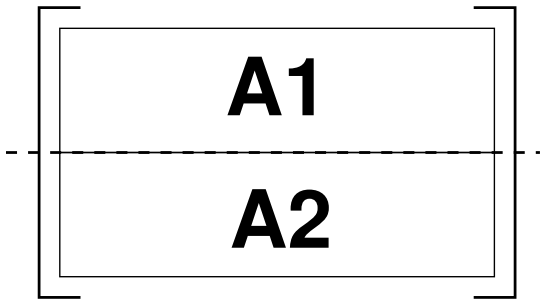
$F = C^{-1}E$ (appel récursif)
$G = D - BF$ (MM)
$H = A^{-1}G$ (appel récursif)
Return $\begin{bmatrix} H \\ F \end{bmatrix}$





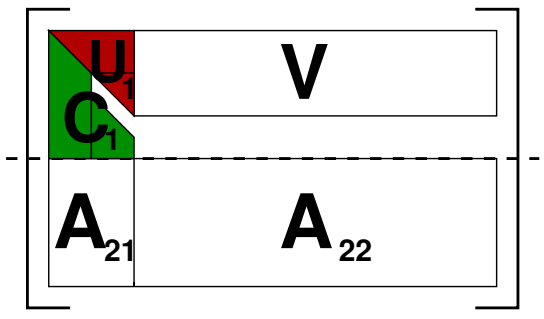
- $\mathcal{O}(n^{\omega})$
- En place

La décomposition CUP



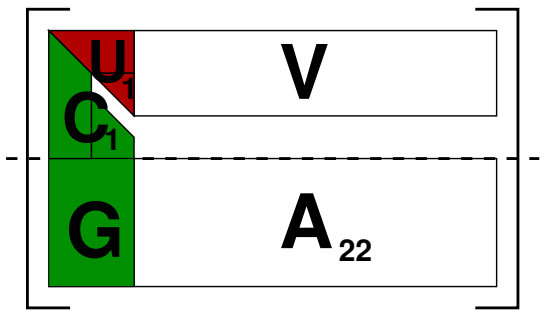
1. Partage A en deux sur les lignes

La décomposition CUP



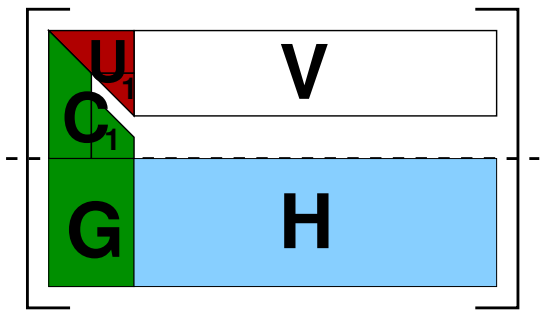
1. Partage A en deux sur les lignes
2. Appel récursif sur A_1

La décomposition CUP



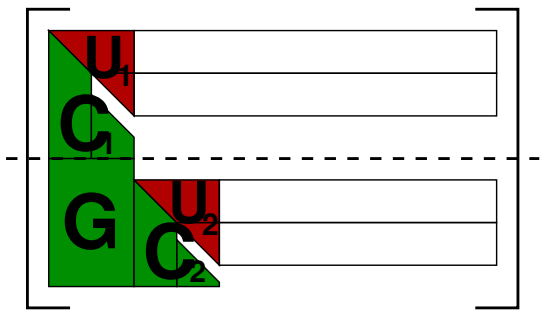
1. Partage A en deux sur les lignes
2. Appel récursif sur A_1
3. $G \leftarrow A_{21} U_1^{-1}$ (`trsm`)

La décomposition CUP



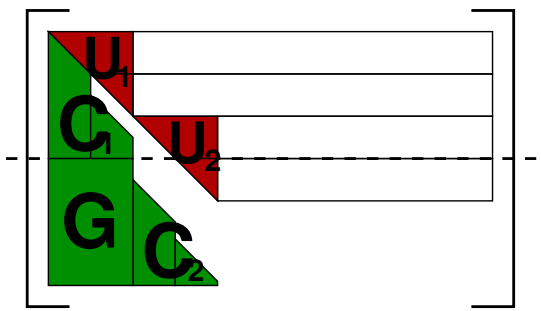
1. Partage A en deux sur les lignes
2. Appel récursif sur A_1
3. $G \leftarrow A_{21} U_1^{-1}$ (`trsm`)
4. $H \leftarrow A_{22} - G \times V$ (`MM`)

La décomposition CUP



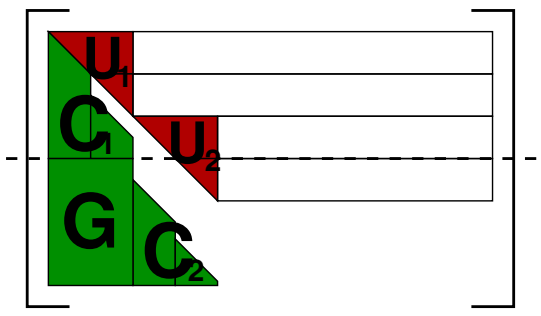
1. Partage A en deux sur les lignes
2. Appel récursif sur A_1
3. $G \leftarrow A_{21} U_1^{-1}$ (`trsm`)
4. $H \leftarrow A_{22} - G \times V$ (`MM`)
5. Appel récursif sur H

La décomposition CUP



1. Partage A en deux sur les lignes
2. Appel récursif sur A_1
3. $G \leftarrow A_{21} U_1^{-1}$ (`trsm`)
4. $H \leftarrow A_{22} - G \times V$ (`MM`)
5. Appel récursif sur H
6. Permutations de lignes

La décomposition CUP



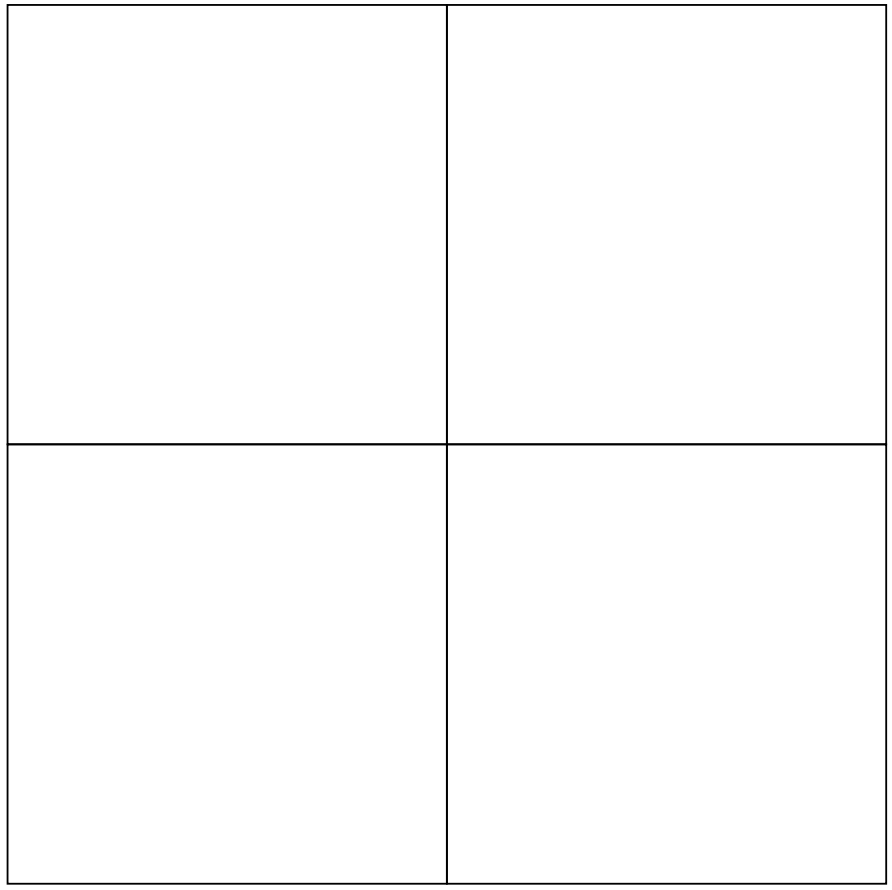
1. Partage A en deux sur les lignes
2. Appel récursif sur A_1
3. $G \leftarrow A_{21} U_1^{-1}$ (`trsm`)
4. $H \leftarrow A_{22} - G \times V$ (`MM`)
5. Appel récursif sur H
6. Permutations de lignes

- Les créneaux de C indiquent le profil de rang en ligne de A.

Algorithme PLUQ Quad-recursif

Principe:

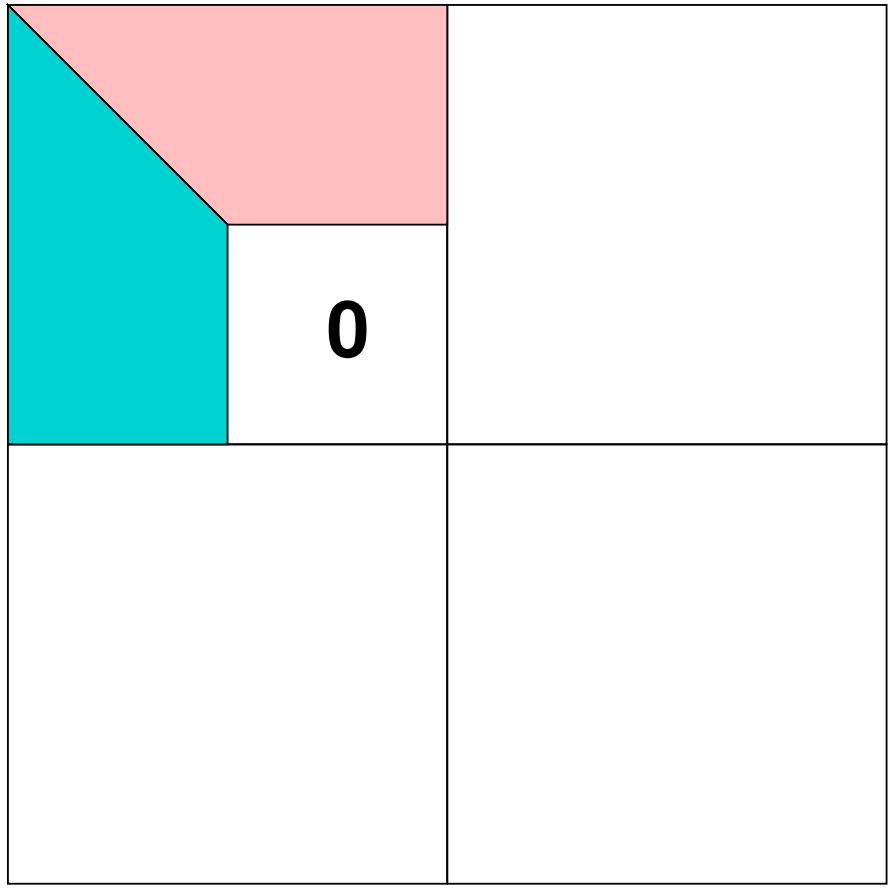
- découpe récursive selon lignes et colonnes
- il existe une façon de permuter les blocs qui révèle les profils de rang en ligne et en colonne à la fois.



Algorithme PLUQ Quad-recursif

Principe:

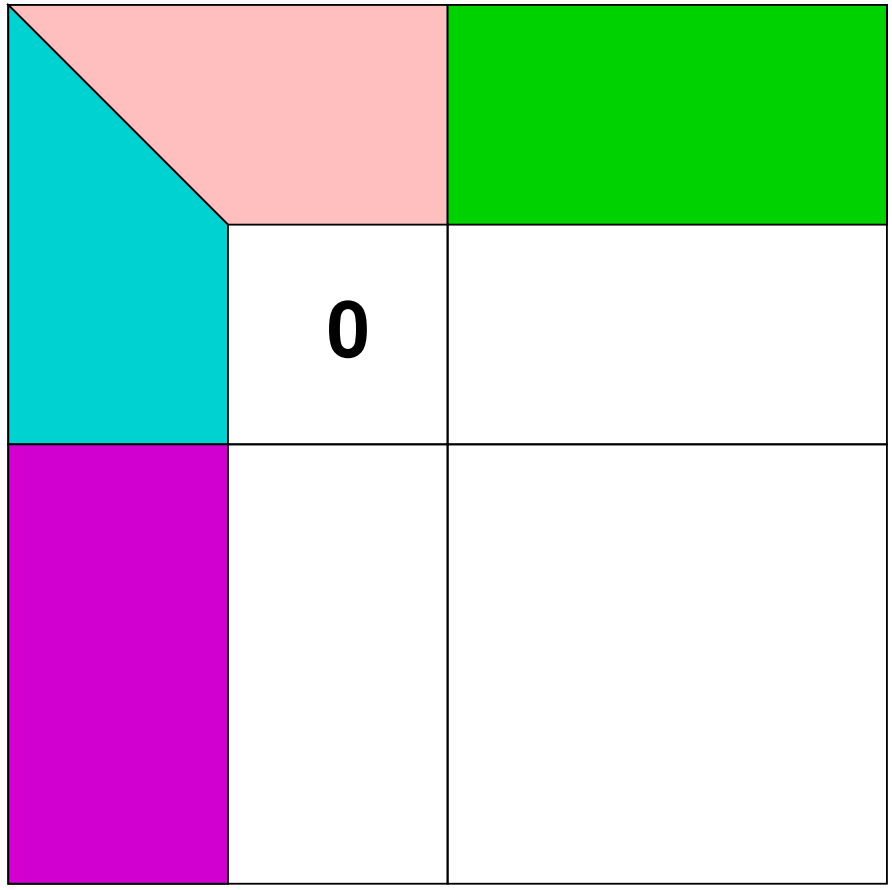
- découpe récursive selon lignes et colonnes
- il existe une façon de permuter les blocs qui révèle les profils de rang en ligne et en colonne à la fois.



Algorithme PLUQ Quad-recursif

Principe:

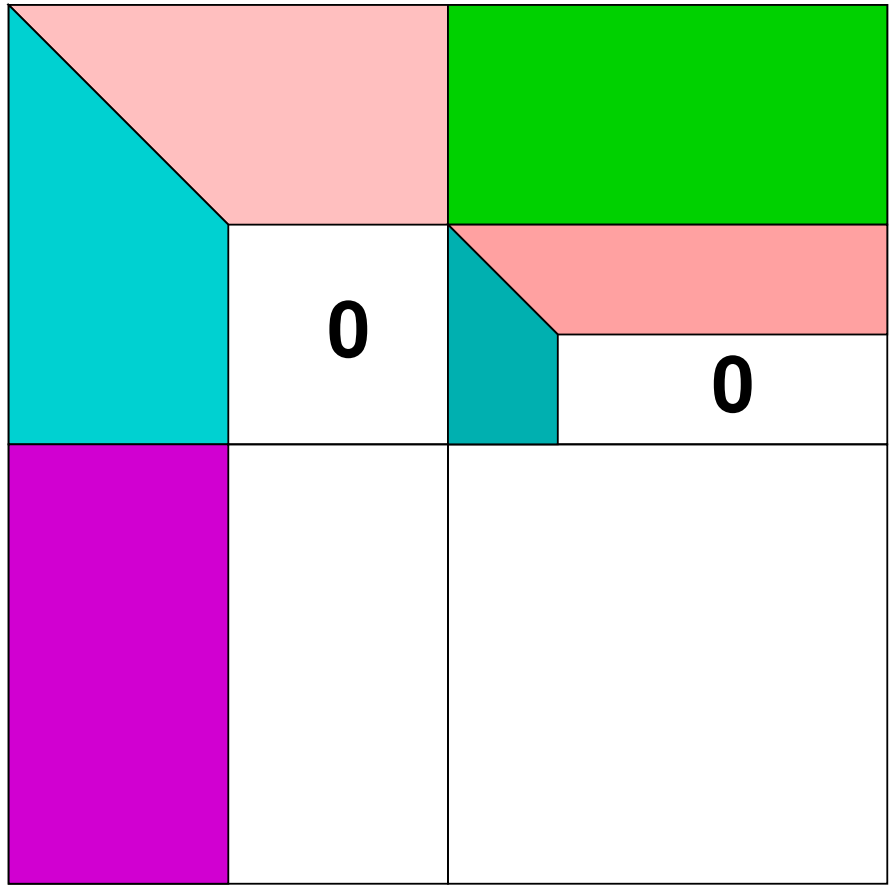
- découpe récursive selon lignes et colonnes
- il existe une façon de permuter les blocs qui révèle les profils de rang en ligne et en colonne à la fois.



Algorithme PLUQ Quad-recursif

Principe:

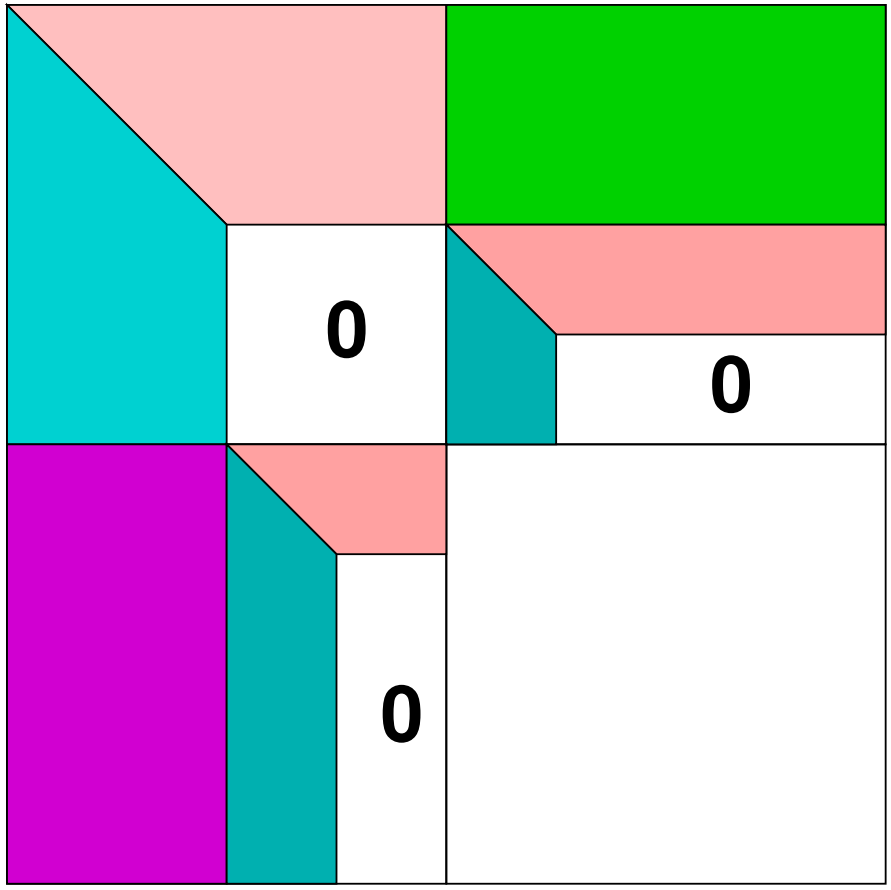
- découpe récursive selon lignes et colonnes
- il existe une façon de permuter les blocs qui révèle les profils de rang en ligne et en colonne à la fois.



Algorithme PLUQ Quad-recursif

Principe:

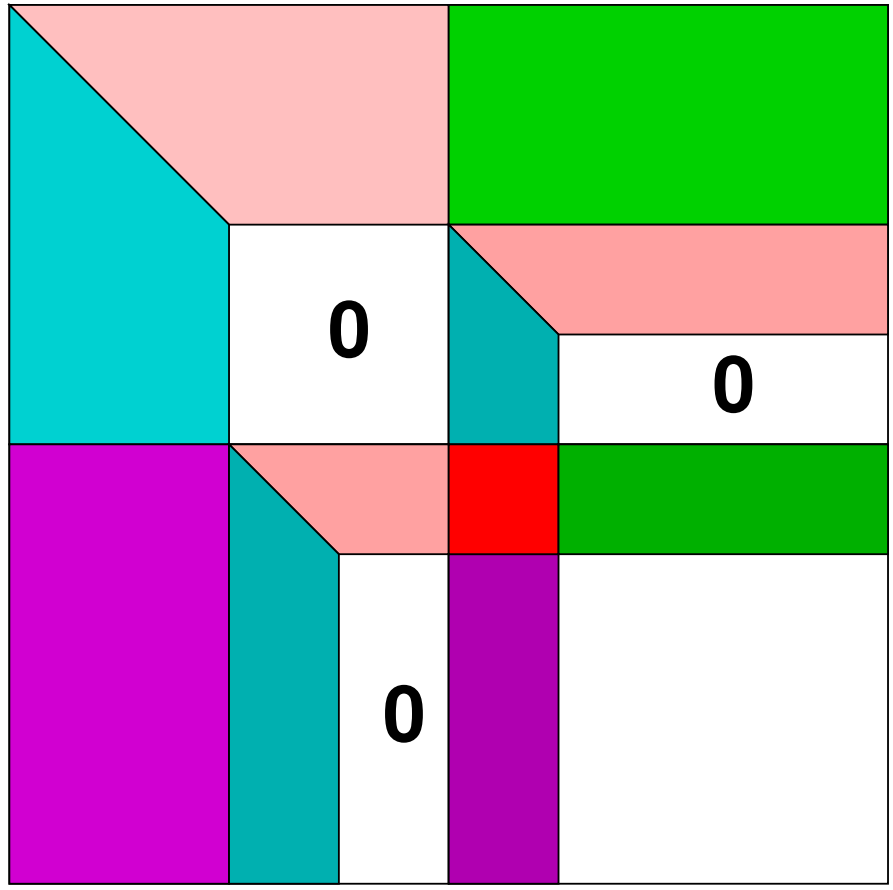
- découpe récursive selon lignes et colonnes
- il existe une façon de permuter les blocs qui révèle les profils de rang en ligne et en colonne à la fois.



Algorithme PLUQ Quad-recursif

Principe:

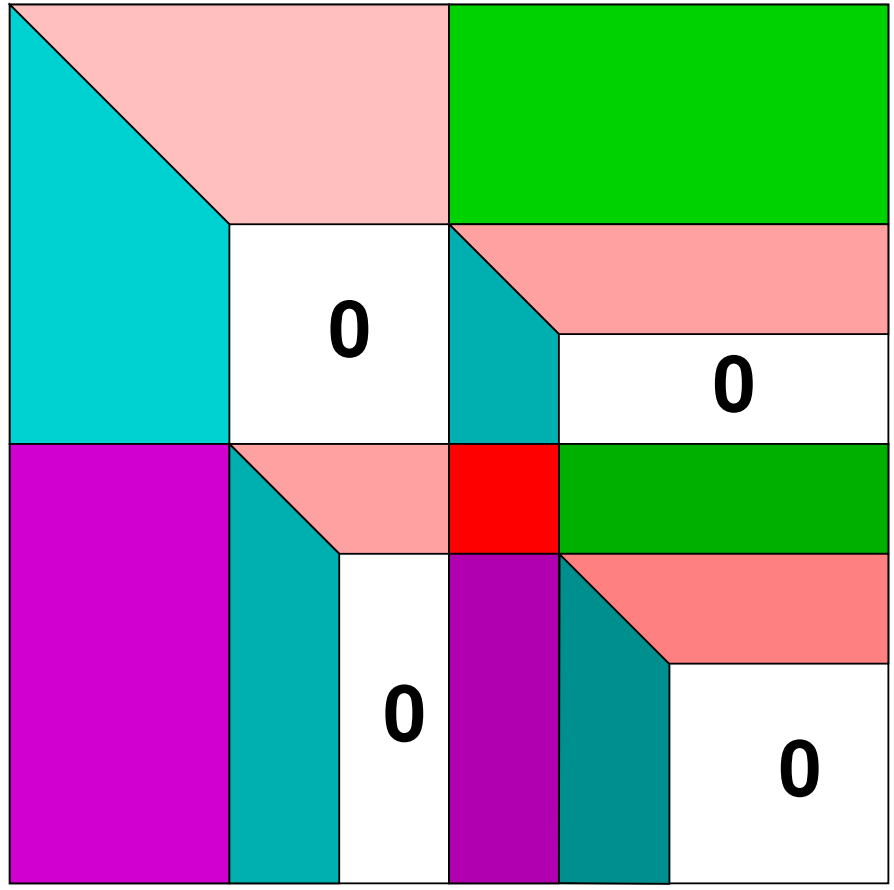
- découpe récursive selon lignes et colonnes
- il existe une façon de permuter les blocs qui révèle les profils de rang en ligne et en colonne à la fois.



Algorithme PLUQ Quad-recursif

Principe:

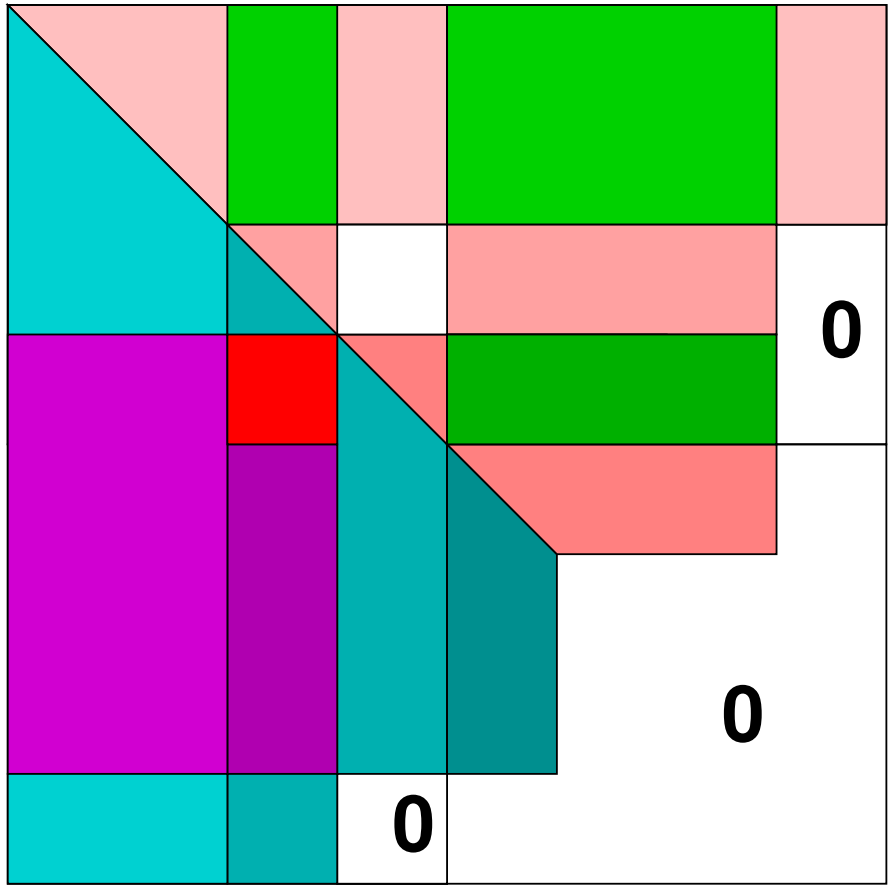
- découpe récursive selon lignes et colonnes
- il existe une façon de permuter les blocs qui révèle les profils de rang en ligne et en colonne à la fois.



Algorithme PLUQ Quad-recursif

Principe:

- découpe récursive selon lignes et colonnes
- il existe une façon de permuter les blocs qui révèle les profils de rang en ligne et en colonne à la fois.



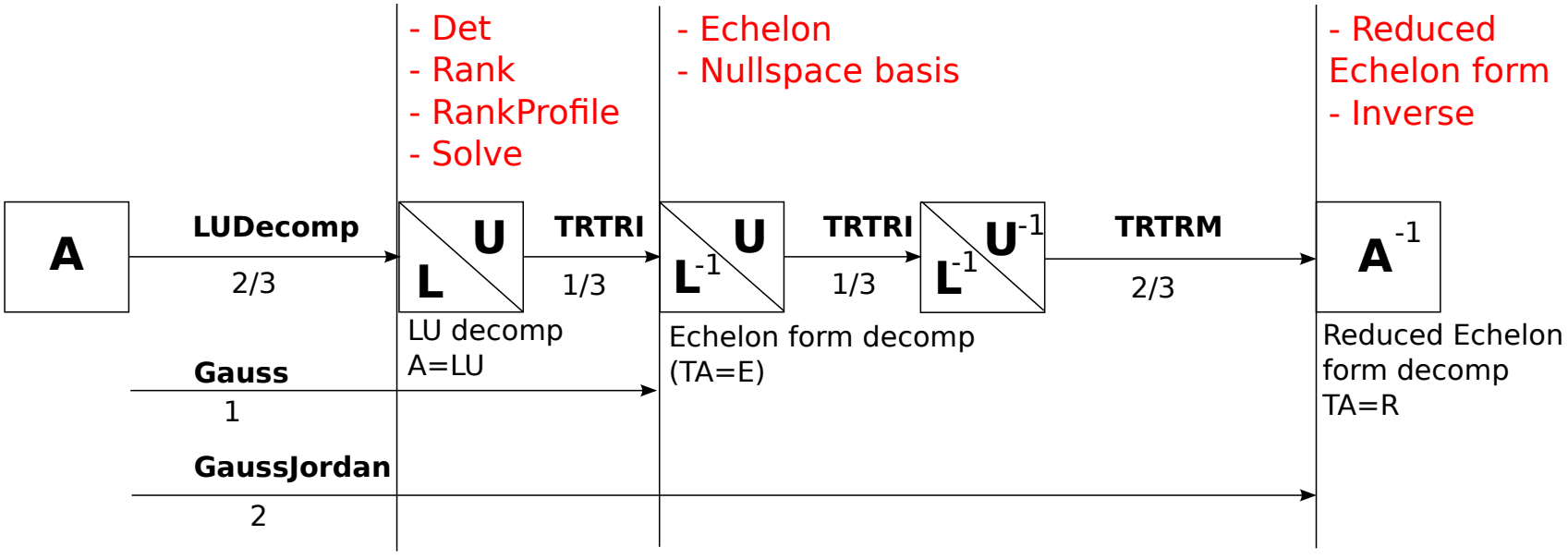
Comparaisons CUP vs PLUQ

- les algorithmes CUP et PLUQ ont une complexité sous-cubique sensible au rang: $\mathcal{O}\left(mnr^{\omega-2}\right)$
- la constant du $\mathcal{O}\left(\right)$ est identique et vaut 2/3 quand $\omega = 3$.
- ils sont *en-place*

Profil de rang

- CUP révèle le profil de rang en ligne
- PLUQ révèle les profils de rang en ligne et colonne (ainsi que ceux de toute sous-matrice dominante)

Applications



Plan

Calcul de la forme normale de Hermite

Équivalence dans un anneau: $H = UA$, où $\det(U) = \pm 1$

Forme de Hermite: $H = \begin{bmatrix} p_1 & * & x_{1,2} & * & * & x_{1,3} & * \\ & & p_2 & * & * & x_{2,3} & * \\ & & & & & p_3 & * \\ & & & & & & \end{bmatrix}$, avec $0 \leq x_{*,j} < p_j$

Forme échelonnée réduite dans un anneau

Amélioration de l'algorithme de Micciancio-Warinshi

- $\mathcal{O}^{\sim}\left(n^5 \log \|A\|\right)$ (heuristiquement: $\mathcal{O}^{\sim}\left(n^3 \log \|A\|\right)$)
- espace: $\mathcal{O}\left(n^2 \log \|A\|\right)$

$\Rightarrow$Bon comportement avec les matrices aléatoires (uniforme), fréquentes en théorie des nombres

Calcul de la forme normale de Hermite

Équivalence dans un anneau: $H = UA$, où $\det(U) = \pm 1$

Forme de Hermite: $H = \begin{bmatrix} p_1 & * & x_{1,2} & * & * & x_{1,3} & * \\ & & p_2 & * & * & x_{2,3} & * \\ & & & & & p_3 & * \\ & & & & & & \end{bmatrix}$, avec $0 \leq x_{*,j} < p_j$

Forme échelonnée réduite dans un anneau

Amélioration de l'algorithme de Micciancio-Warinshi

- $\mathcal{O}^{\sim}\left(n^5 \log \|A\|\right)$ (heuristiquement: $\mathcal{O}^{\sim}\left(n^3 \log \|A\|\right)$)
- espace: $\mathcal{O}\left(n^2 \log \|A\|\right)$

$\Rightarrow$Bon comportement avec les matrices aléatoires (uniforme), fréquentes en théorie des nombres
Implantation, réduction à des briques de base:

- SysLin dans $\mathbb{Z}$,
- PLUQ et MatMul dans $\mathbb{Z}_p$

Algorithme naïf

1 **begin**
2 **foreach** i **do**
3 $(g, t_i, \ldots, t_n) = \mathsf{xgcd}(A_{i,i}, A_{i+1,i}, \ldots, A_{n,i})$;
4 $L_i \leftarrow \sum_{j=i+1}^{n} t_j L_j$;
5 **for** $j = i+1 \ldots n$ **do**
6 $L_j \leftarrow L_j - \frac{A_{j,i}}{g} L_i$; /* élimine */
7 **for** $j = 1 \ldots i-1$ **do**
8 $L_j \leftarrow L_j - \lfloor \frac{A_{j,i}}{g} \rfloor L_i$; /* réduit */

$$\begin{bmatrix} p_1 & * & x_{1,2} & * & * & x_{1,3} & * \\ & & p_2 & * & * & x_{2,3} & * \\ & & & & & p_3 & * \\ & & & & & & \end{bmatrix}$$

Calcul modulo le déterminant [Domich & Al. 87]

Propriété

Pour A inversible: $\max_i \sum_j H_{ij} \leq \det H$

Exemple

$$A = \begin{bmatrix} -5 & 8 & -3 & -9 & 5 & 5 \\ -2 & 8 & -2 & -2 & 8 & 5 \\ 7 & -5 & -8 & 4 & 3 & -4 \\ 1 & -1 & 6 & 0 & 8 & -3 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 & 3 & 237 & -299 & 90 \\ 0 & 1 & 1 & 103 & -130 & 40 \\ 0 & 0 & 4 & 352 & -450 & 135 \\ 0 & 0 & 0 & 486 & -627 & 188 \end{bmatrix}$$

$$\det A = 1944$$

Calcul modulo le déterminant [Domich & Al. 87]

Propriété

Pour A inversible: $\max_i \sum_j H_{ij} \leq \det H$

Exemple

$$A = \begin{bmatrix} -5 & 8 & -3 & -9 & 5 & 5 \\ -2 & 8 & -2 & -2 & 8 & 5 \\ 7 & -5 & -8 & 4 & 3 & -4 \\ 1 & -1 & 6 & 0 & 8 & -3 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 & 3 & 237 & -299 & 90 \\ 0 & 1 & 1 & 103 & -130 & 40 \\ 0 & 0 & 4 & 352 & -450 & 135 \\ 0 & 0 & 0 & 486 & -627 & 188 \end{bmatrix}$$

$$\det A = 1944$$

De plus, tous les calculs peuvent être faits modulo $d = \det A$:

$$U' \begin{bmatrix} A & \\ dI_n & I_n \end{bmatrix} = \begin{bmatrix} H & \\ & I_n \end{bmatrix}$$

$$\Rightarrow \mathcal{O}\left(n^3\right) \times M(n(\log n + \log \|A\|)) = \tilde{\mathcal{O}}\left(n^4 \log \|A\|\right)$$

Calcul modulo le déterminant

- Estimation pessimiste sur l'arithmétique
- d grand mais la plupart des coefficients de H sont petits
- *En moyenne* : seules les dernières colonnes sont *grandes*

$\Rightarrow$Calculer H' *proche* de H mais de petit déterminant

Calcul modulo le déterminant

- Estimation pessimiste sur l'arithmétique
- d grand mais la plupart des coefficients de H sont petits
- *En moyenne* : seules les dernières colonnes sont *grandes*

$\Rightarrow$Calculer H' *proche* de H mais de petit déterminant

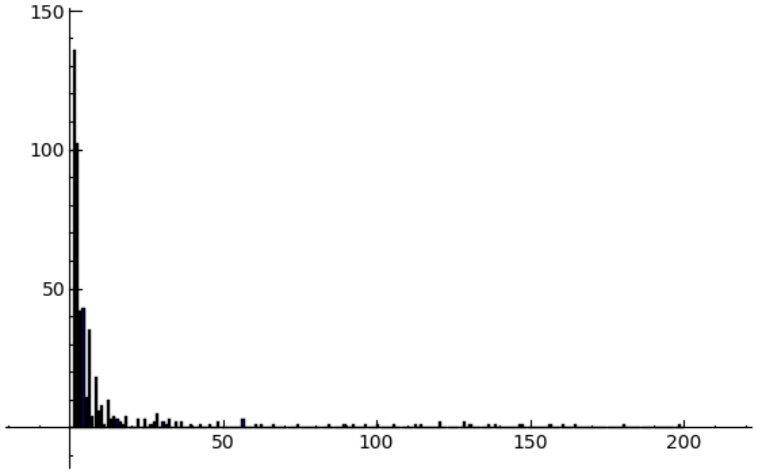
[Micciancio & Warinschi 01]

$$A = \begin{bmatrix} B & b \\ c^T & a_{n-1,n} \\ d^T & a_{n,n} \end{bmatrix}$$

$$d_1 = \det\left(\begin{bmatrix} B \\ c^T \end{bmatrix}\right), d_2 = \det\left(\begin{bmatrix} B \\ d^T \end{bmatrix}\right)$$

$g = \text{pgcd}(d_1, d_2) = sd_1 + td_2$ Alors

$$\det\left(\begin{bmatrix} B \\ sc^T + td^T \end{bmatrix}\right) = g$$



Algorithme de Micciancio & Warinschi

1 **begin**

2 Compute $d_1 = \det\left(\begin{bmatrix} B \\ c^T \end{bmatrix}\right), d_2 = \det\left(\begin{bmatrix} B \\ d^T \end{bmatrix}\right)$; /* Double Det */

3 $(g, s, t) = \mathsf{xgcd}(d_1, d_2)$;

4 Compute H_1 the HNF of $\begin{bmatrix} B \\ sc^T + td^T \end{bmatrix} \mod g$; /* Modular HNF */

5 Recover H_2 the HNF of $\begin{bmatrix} B & b \\ sc^T + td^T & sa_{n-1,n} + ta_{n,n} \end{bmatrix}$; /* AddCol */

6 Recover H_3 the HNF of $\begin{bmatrix} B & b \\ c^T & a_{n-1,n} \\ d^T & a_{n,n} \end{bmatrix}$; /* AddRow */

Algorithme de Micciancio & Warinschi

begin

Compute $d_1 = \det\left(\begin{bmatrix} B \\ c^T \end{bmatrix}\right), d_2 = \det\left(\begin{bmatrix} B \\ d^T \end{bmatrix}\right)$; /* Double Det */

$(g, s, t) = \mathsf{xgcd}(d_1, d_2)$;

Compute H_1 the HNF of $\begin{bmatrix} B \\ sc^T + td^T \end{bmatrix} \mod g$; /* Modular HNF */

Recover H_2 the HNF of $\begin{bmatrix} B & b \\ sc^T + td^T & sa_{n-1,n} + ta_{n,n} \end{bmatrix}$; /* AddCol */

Recover H_3 the HNF of $\begin{bmatrix} B & b \\ c^T & a_{n-1,n} \\ d^T & a_{n,n} \end{bmatrix}$; /* AddRow */

Double Déterminant

Première approche: LU mod $p_1, \ldots, p_k$ + TRC

- Une seule élimination pour les $n-2$ premières lignes
- 2 mises à jour pour les dernières ligne (remontée triang.)
- k *grand* tel que $\prod_{i=1}^{k} p_i > n^n \log \|A\|^{n/2}$

Double Déterminant

Première approche: LU mod $p_1, \ldots, p_k$ + TRC

- Une seule élimination pour les $n-2$ premières lignes
- 2 mises à jour pour les dernières ligne (remontée triang.)
- k *grand* tel que $\prod_{i=1}^{k} p_i > n^n \log \|A\|^{n/2}$

Deuxième approche: [Abbott Bronstein Mulders 99]

- Résoudre $Ax = b$.
- $\delta = \text{ppcm}(q_1, \ldots, q_n)$ tq $x_i = p_i/q_i$

Alors δ est un *grand* diviseur de $D = \det A$.

- Calculer D/δ par LU mod $p1, \ldots, p_k$ + TRC
- k *petit*, tel que $\prod_{i=1}^{k} p_i > n^n \log \|A\|^{n/2}/\delta$

Double Déterminant : amélioré

Propriété

Soit $x = [x_1, \ldots, x_n]$ *la solution de* $\left[\begin{array}{c|c} A & c \end{array}\right] x = d$. *Alors* $y = [-\frac{x_1}{x_n}, \ldots, -\frac{x_{n-1}}{x_n}, \frac{1}{x_n}]$ *est la solution de* $\left[\begin{array}{c|c} A & d \end{array}\right] x = c$.

- 1 résolution de système
- 1 seule LU par p_i

$\Rightarrow$Calcul de d_1, d_2 pour le coût de 1 déterminant

AddCol

Problème

Trouver un vecteur e tq

$$\left[\begin{array}{c|c} H_1 & e \end{array}\right] = U \begin{bmatrix} B & b \\ sc^T + td^T & sa_{n-1,n} + ta_{n,n} \end{bmatrix}$$

$$\begin{aligned} e &= U \begin{bmatrix} b \\ sa_{n-1,n} + ta_{n,n} \end{bmatrix} \\ &= H_1 \begin{bmatrix} B \\ sc^T + td^T \end{bmatrix}^{-1} \begin{bmatrix} b \\ sa_{n-1,n} + ta_{n,n} \end{bmatrix} \end{aligned}$$

$\Rightarrow$Résoudre un système.

- $n-1$ premières lignes *petites*
- dernière ligne *grande*

AddCol

Idée:

Remplacer la dernière ligne par une ligne aléatoire *petite* w^T.

$$\begin{bmatrix} B \\ w^T \end{bmatrix} y = \begin{bmatrix} b \\ a_{n-1,n-1} \end{bmatrix}$$

Soit k une base du noyau de B.Alors

$$x = y + \alpha k.$$

où

$$\alpha = \frac{a_{n-1,n-1} - (sc^T + td^T) \cdot y}{(sc^T + td^T) \cdot k}$$

$\Rightarrow$limite l'arithmétique *chère* à des produits scalaires

Plan

Plan

Méthode de Krylov

Definition (degree d Krylov matrix of one vector v)

$$K = \begin{bmatrix} v & Av & \dots & A^{d-1}v \end{bmatrix}$$

Propriété

$$A \times K = K \times \begin{bmatrix} 0 & & & * \\ 1 & & & * \\ & \ddots & & * \\ & & 1 & * \end{bmatrix}$$

Méthode de Krylov

Definition (degree d Krylov matrix of one vector v)

$$K = \begin{bmatrix} v & Av & \dots & A^{d-1}v \end{bmatrix}$$

Propriété

$$A \times K = K \times \begin{bmatrix} 0 & & & * \\ 1 & & & * \\ & \ddots & & * \\ & & 1 & * \end{bmatrix}$$

$\Rightarrow$if $d = n$,

$$K^{-1}AK = C_{P^A_{car}}$$

Méthode de Krylov

Definition (degree d Krylov matrix of one vector v)

$$K = \begin{bmatrix} v & Av & \dots & A^{d-1}v \end{bmatrix}$$

Propriété

$$A \times K = K \times \begin{bmatrix} 0 & & & * \\ 1 & & & * \\ & \ddots & & * \\ & & 1 & * \end{bmatrix}$$

$\Rightarrow$if $d = n$,

$$K^{-1}AK = C_{P^A_{car}}$$

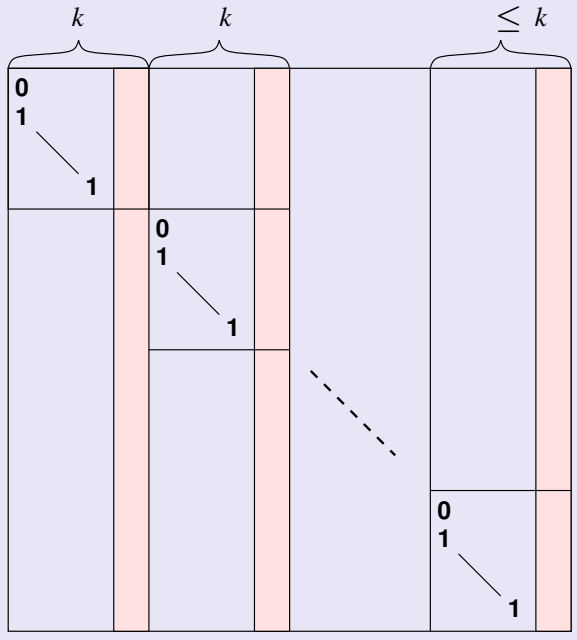
$\Rightarrow$[Keller-Gehrig, alg. 2] computes K in $\mathcal{O}\left(n^\omega \log n\right)$

Definition (degree k Krylov matrix of several vectors v_i)

$$K = \left[\begin{array}{ccc|ccc|c|ccc} v_1 & \dots & A^{k-1}v_1 & v_2 & \dots & A^{k-1}v_2 & \dots & v_l & \dots & A^{k-1}v_l \end{array} \right]$$

Propriété

$A \times K = K \times$



Hessenberg poly-cyclic form

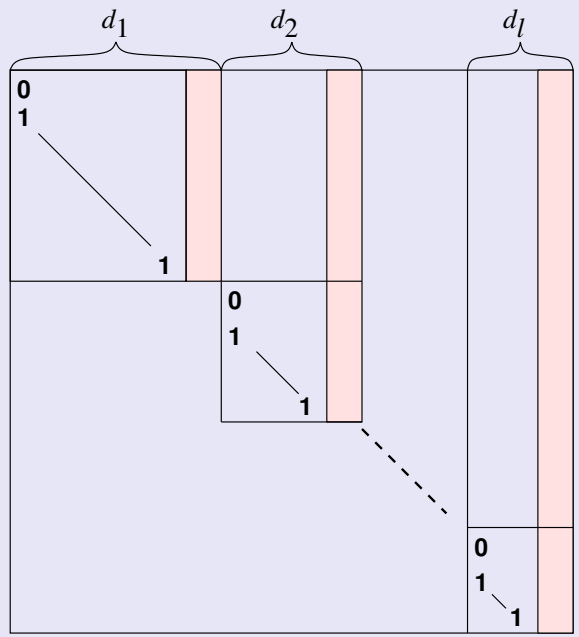
Fact

If $(d_1, \ldots d_l)$ *is lexicographically maximal such that*

$$K = \left[\begin{array}{ccc|c|ccc} v_1 & \ldots & A^{d_1-1}v_1 & \ldots & v_l & \ldots & A^{d_l-1}v_l \end{array} \right]$$

is non-singular, then

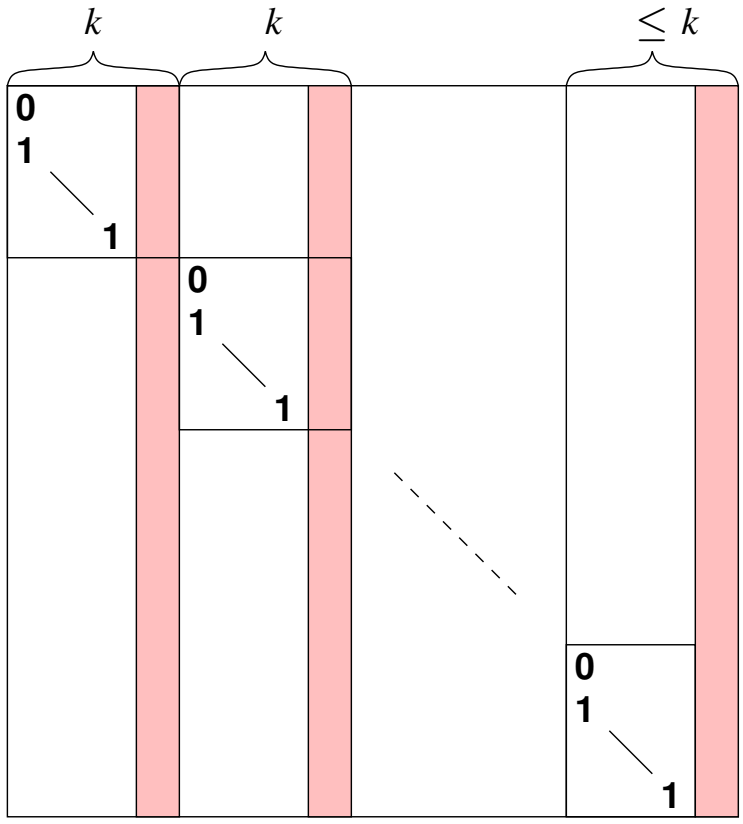
$$A \times K = K \times$$



Plan

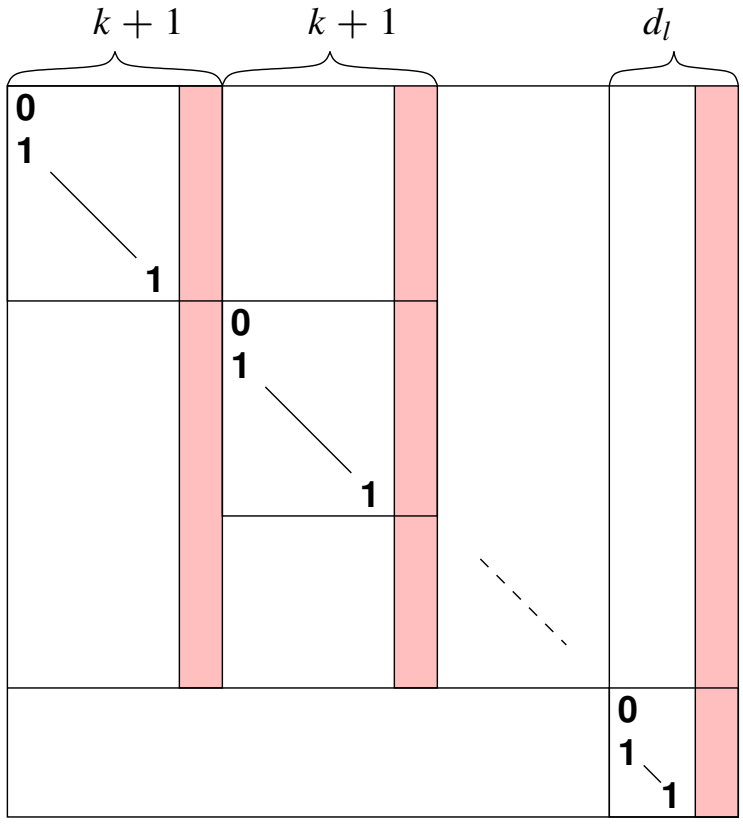
Principle

k-shifted form:



Principle

$k+1$-shifted form:



Principle

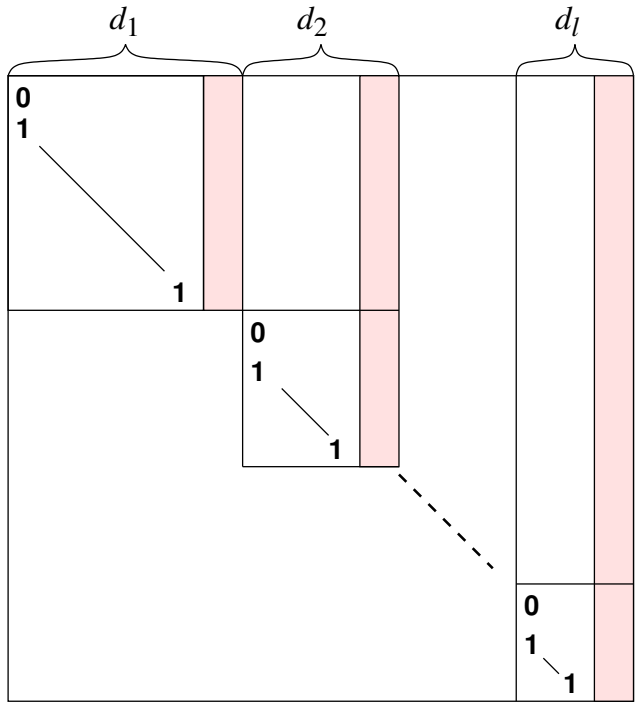
- Compute iteratively from 1-shifted form to d_1-shifted form

Principle

- Compute iteratively from 1-shifted form to d_1-shifted form
- each diagonal block appears in the increasing degree

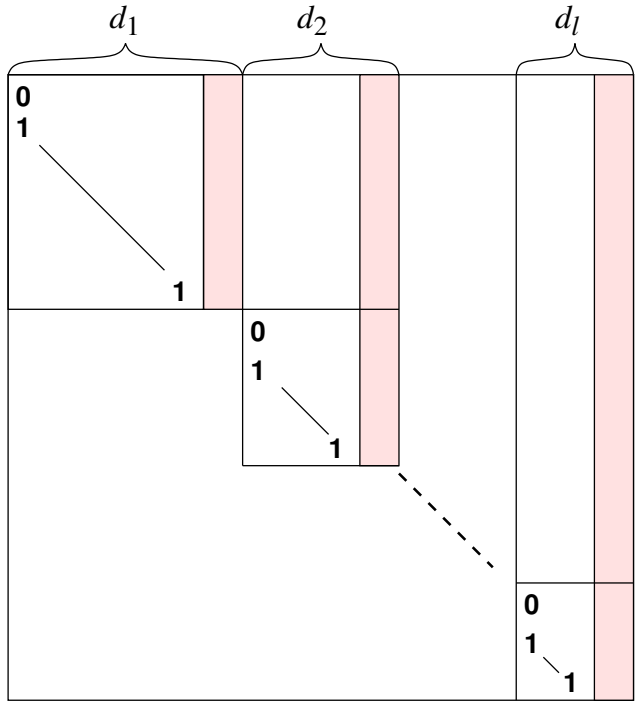
Principle

- Compute iteratively from 1-shifted form to d_1-shifted form
- each diagonal block appears in the increasing degree
- until the shifted Hessenberg form is obtained:



Principle

- Compute iteratively from 1-shifted form to d_1-shifted form
- each diagonal block appears in the increasing degree
- until the shifted Hessenberg form is obtained:

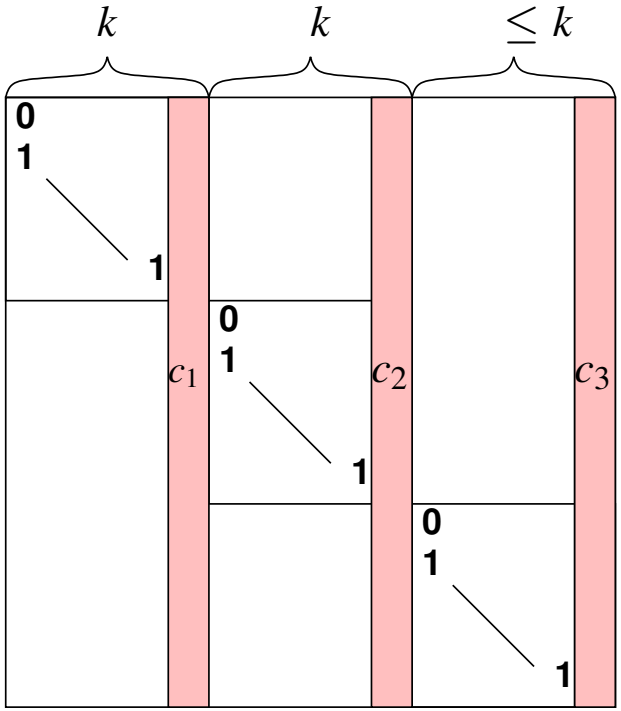


How to transform from k to $k+1$-shifted form ?

Krylov normal extension

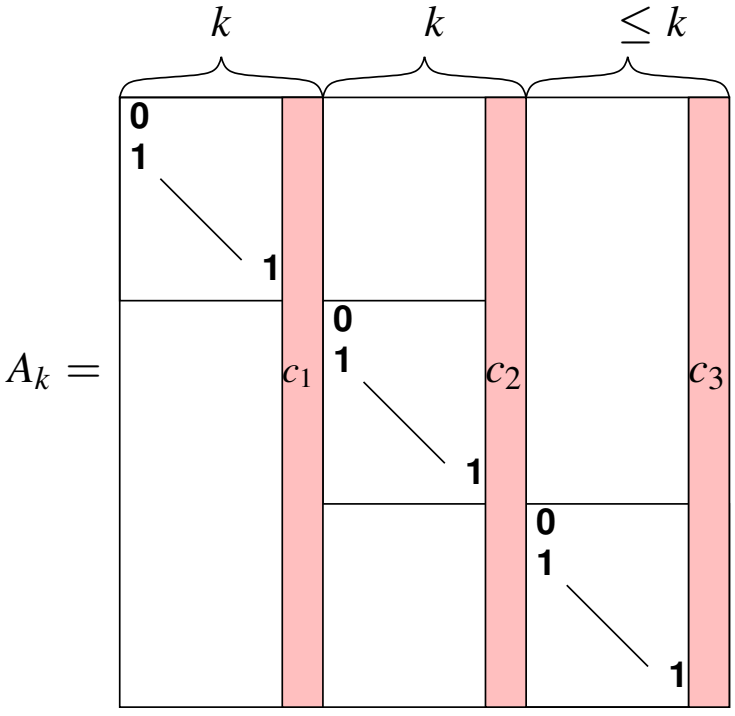
for any k-shifted form

$A_k =$

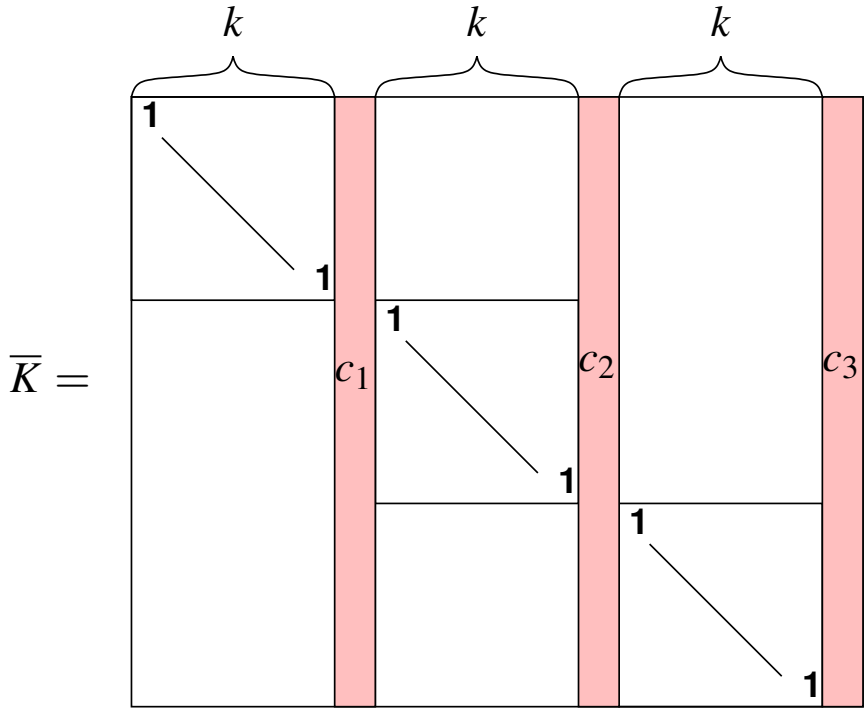


Krylov normal extension

for any k-shifted form

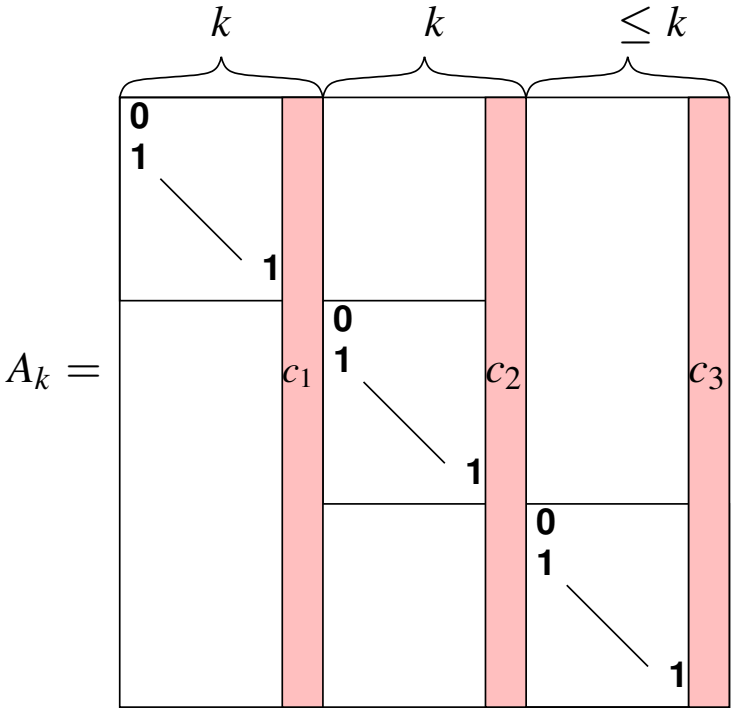


compute the $n \times (n+k)$ matrix

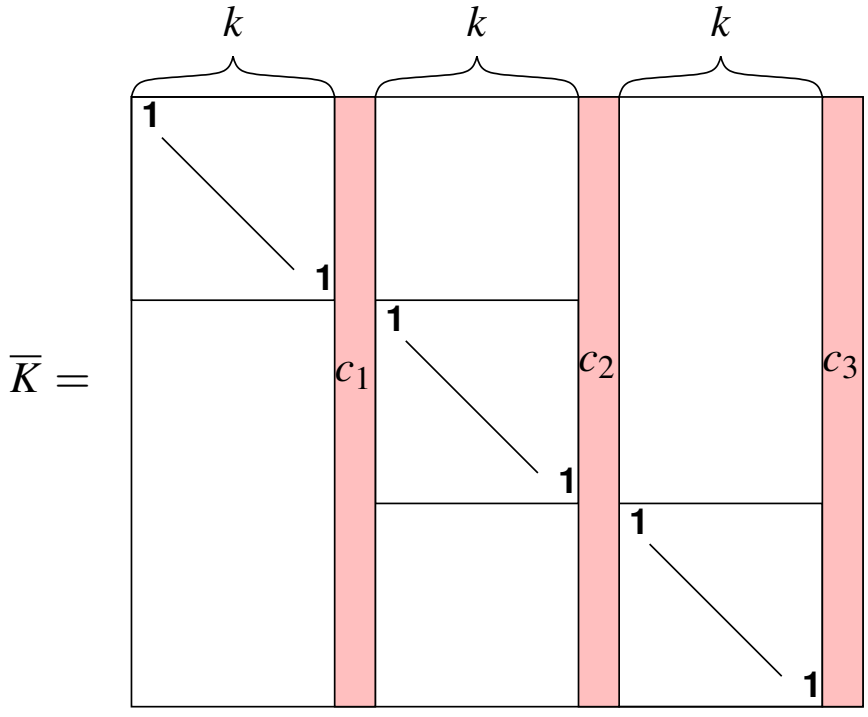


Krylov normal extension

for any k-shifted form



compute the $n \times (n+k)$ matrix



and form K by picking its first linearly independent columns.

The algorithm

- Form $\overline{K}$: just copy the columns of A_k

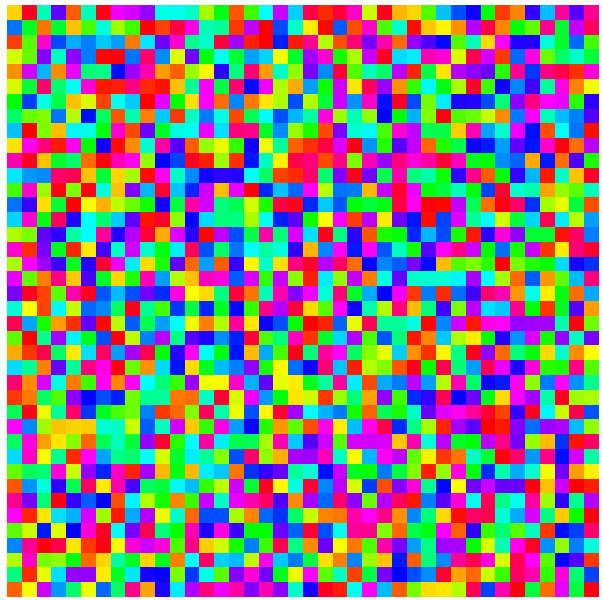
The algorithm

- Form $\overline{K}$: just copy the columns of A_k
- Compute K: rank profile of $\overline{K}$

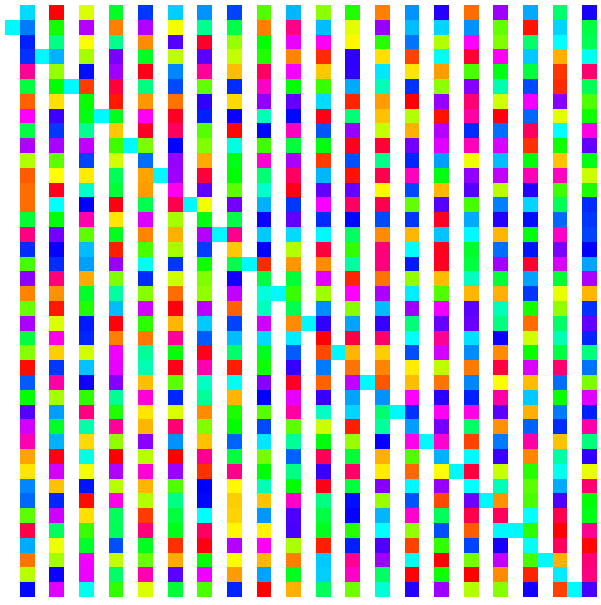
The algorithm

- Form $\overline{K}$: just copy the columns of A_k
- Compute K: rank profile of $\overline{K}$
- Apply the similarity transformation $K^{-1}A_kK$

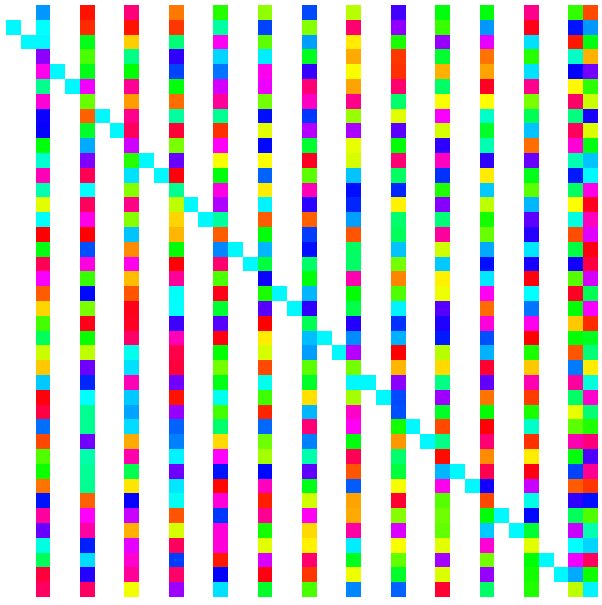
Example



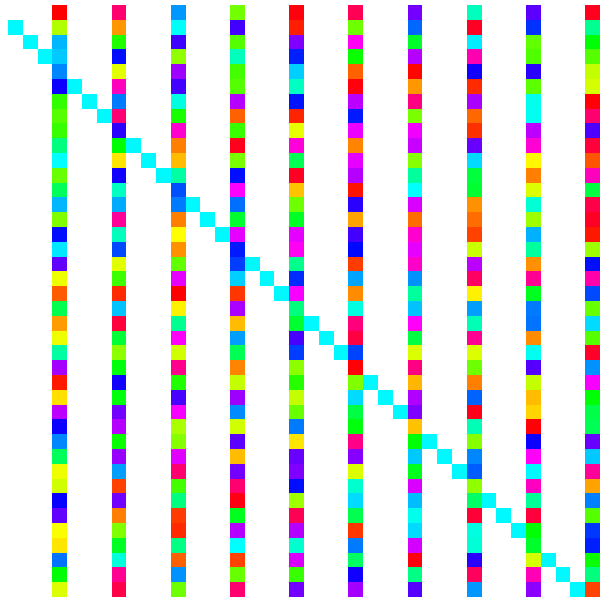
Example



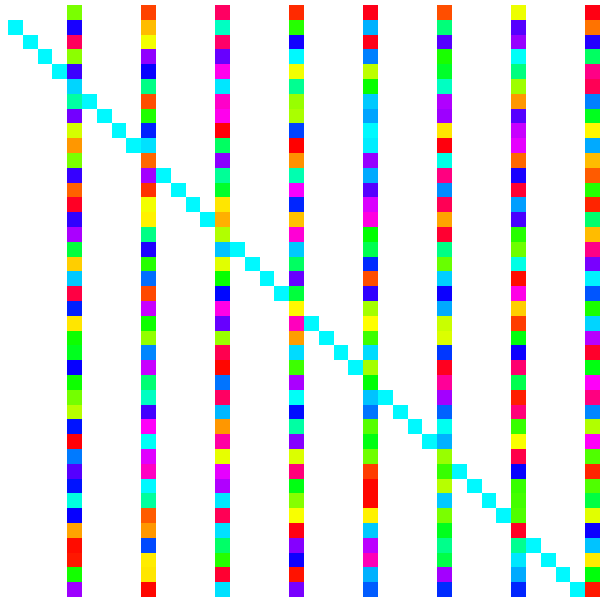
Example



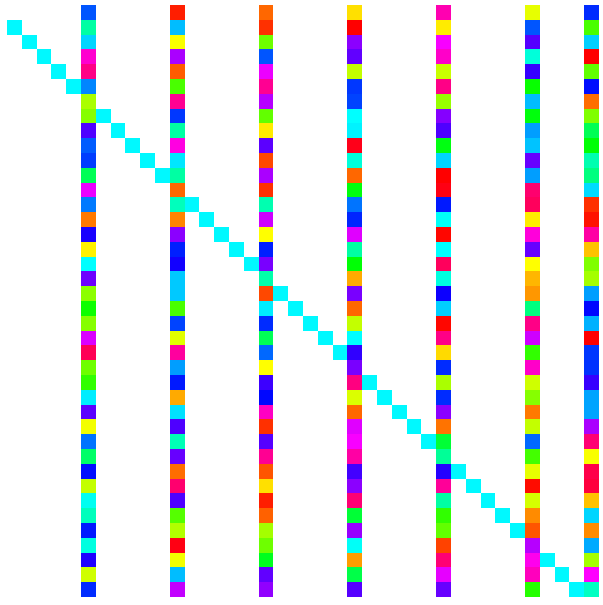
Example



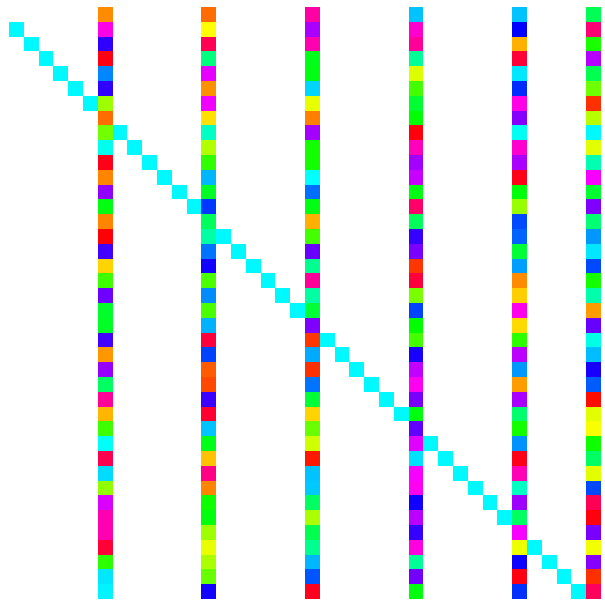
Example



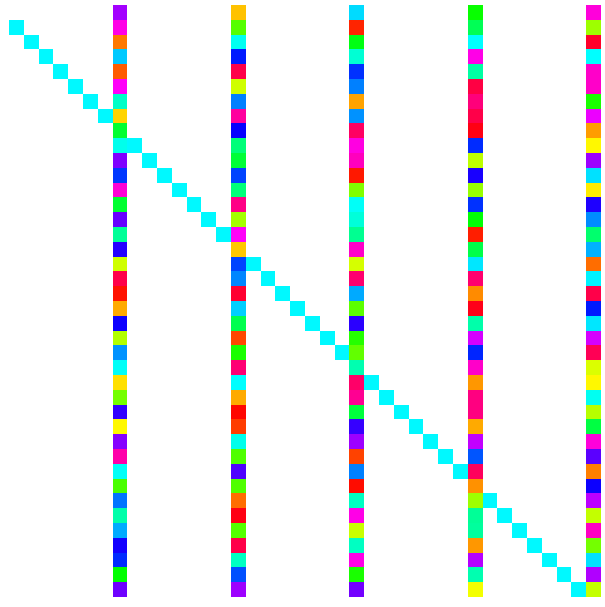
Example



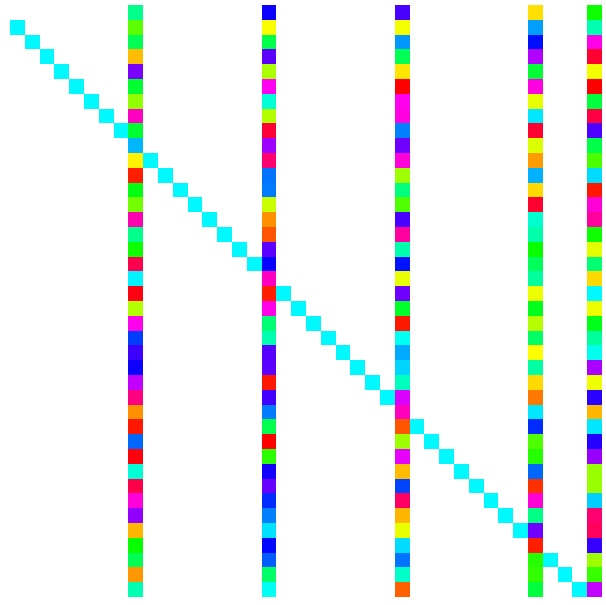
Example



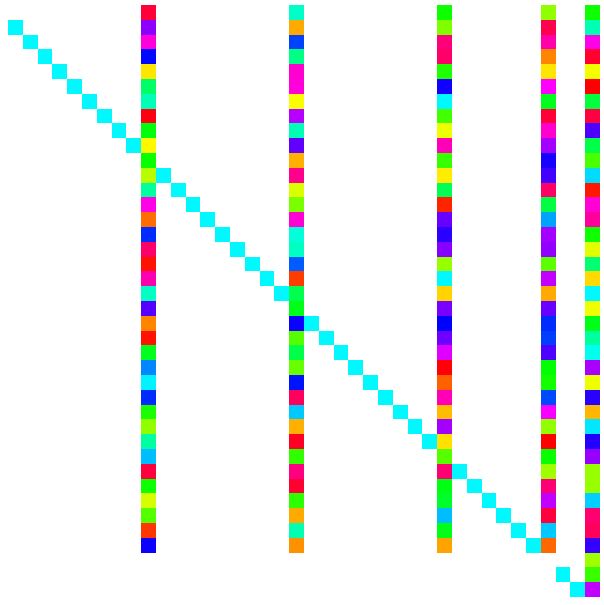
Example



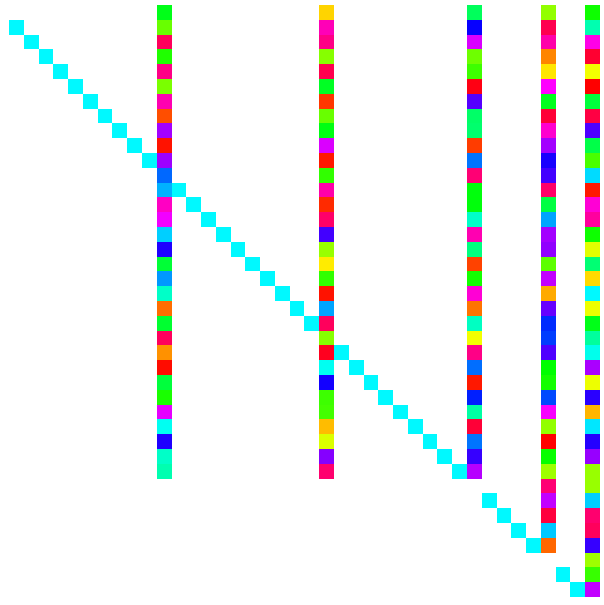
Example



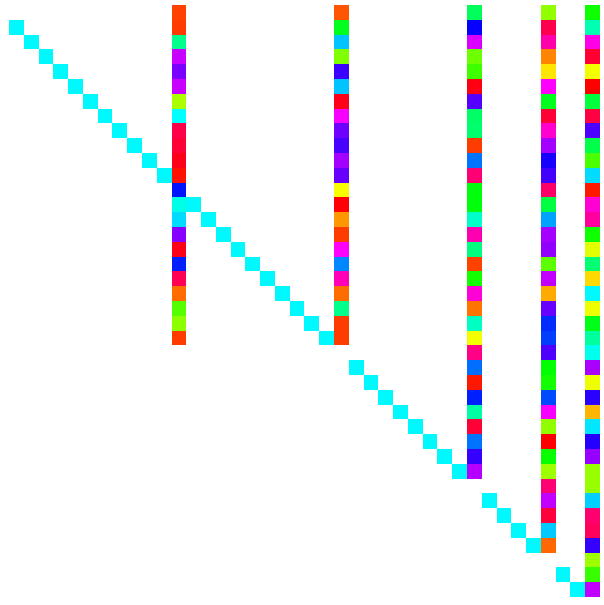
Example



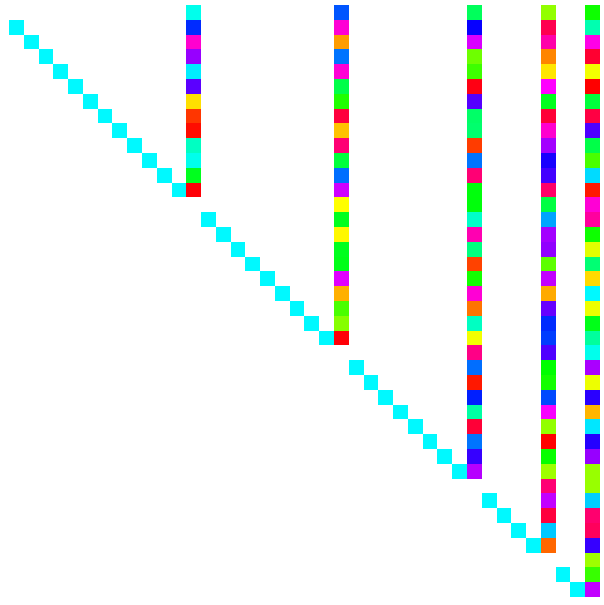
Example



Example



Example



Lemme

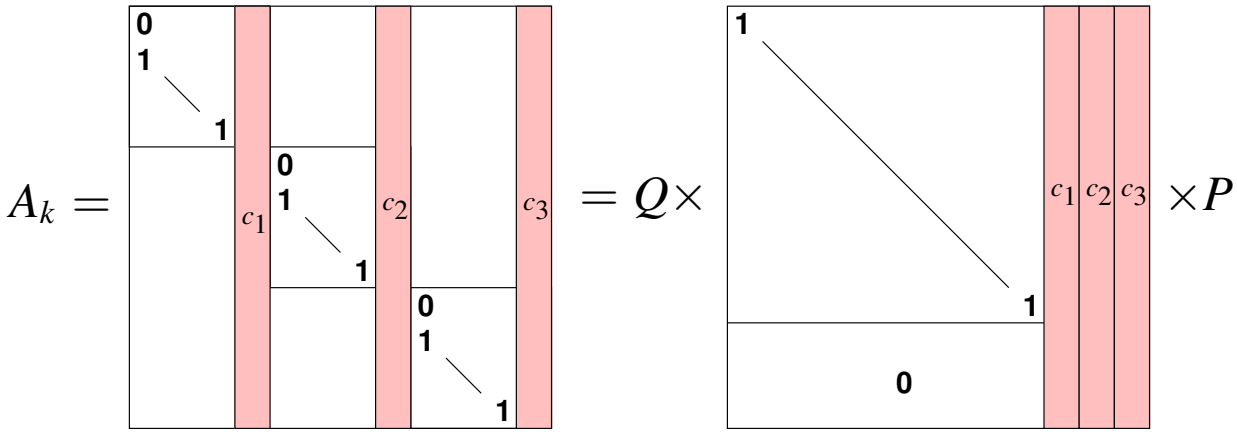
If $\#F > 2n^2$, the transformation will succeed with high probability. Failure is detected.

Lemme

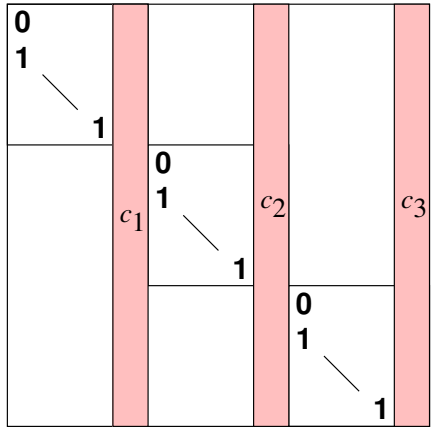
If $\#F > 2n^2$, the transformation will succeed with high probability. Failure is detected.

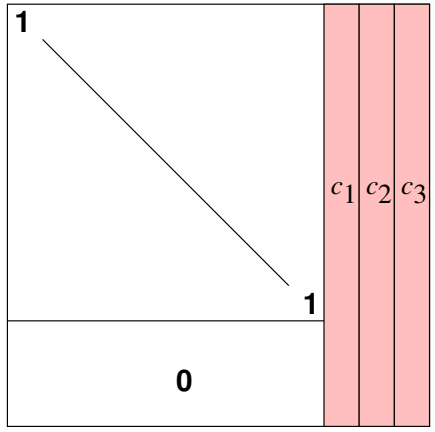
How to use fast matrix arithmetic ?

Permutations: compressing the dense columns

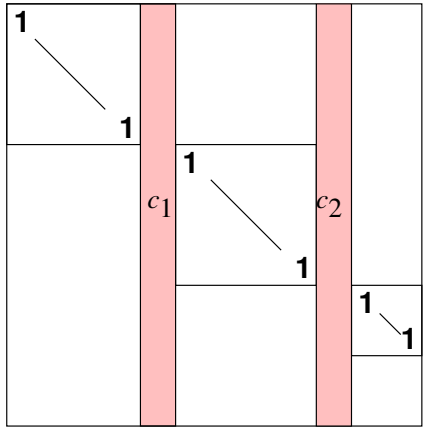


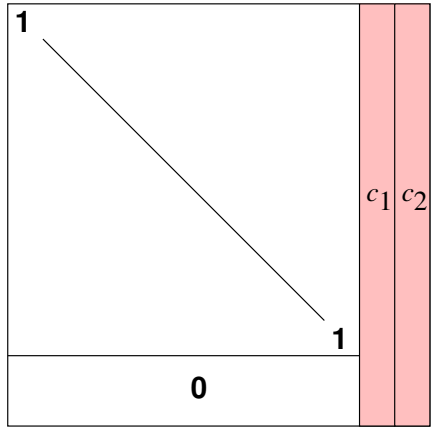
Permutations: compressing the dense columns

$A_k =$ 

$= Q \times$ 

$\times P$

$K =$ 

$= Q' \times$ 

$\times P'$

Plan

Reduction to Matrix multiplication

Similarity transformation: parenthesing

$$K^{-1}AK = Q'^{-1}\begin{bmatrix} I & * \\ 0 & * \end{bmatrix} P'^{-1}Q \begin{bmatrix} I & * \\ 0 & * \end{bmatrix} PQ' \begin{bmatrix} I & * \\ 0 & * \end{bmatrix} P'$$

Reduction to Matrix multiplication

Similarity transformation: parenthesing

$$K^{-1}AK = Q'^{-1}\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(P'^{-1}Q\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(PQ'\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\right)\right)\right)\right)P'$$

Reduction to Matrix multiplication

Similarity transformation: parenthesing

$$K^{-1}AK = Q'^{-1}\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(P'^{-1}Q\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(PQ'\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\right)\right)\right)\right)P'$$

$$\Rightarrow \mathcal{O}\left(k\left(\frac{n}{k}\right)^{\omega}\right)$$

Reduction to Matrix multiplication

Similarity transformation: parenthesing

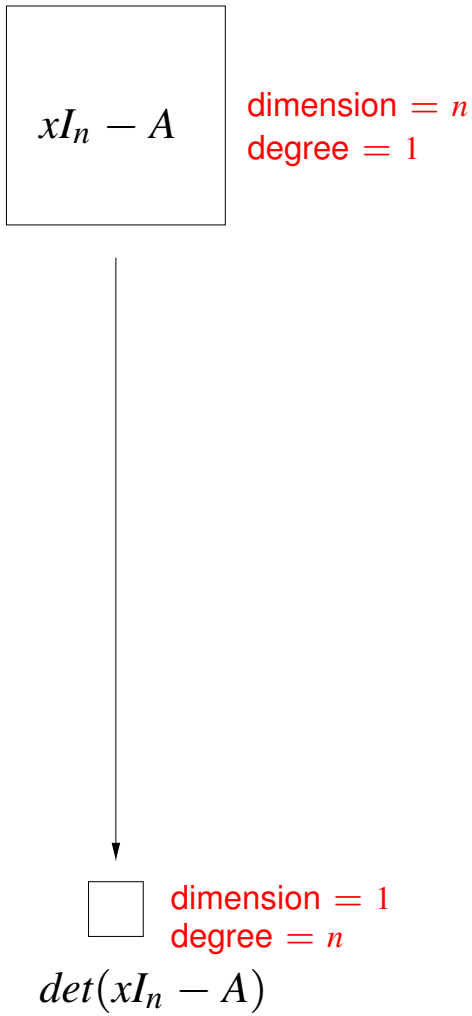
$$K^{-1}AK = Q'^{-1}\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(P'^{-1}Q\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(PQ'\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\right)\right)\right)\right)P'$$

$$\Rightarrow \mathcal{O}\left(k\left(\frac{n}{k}\right)^{\omega}\right)$$

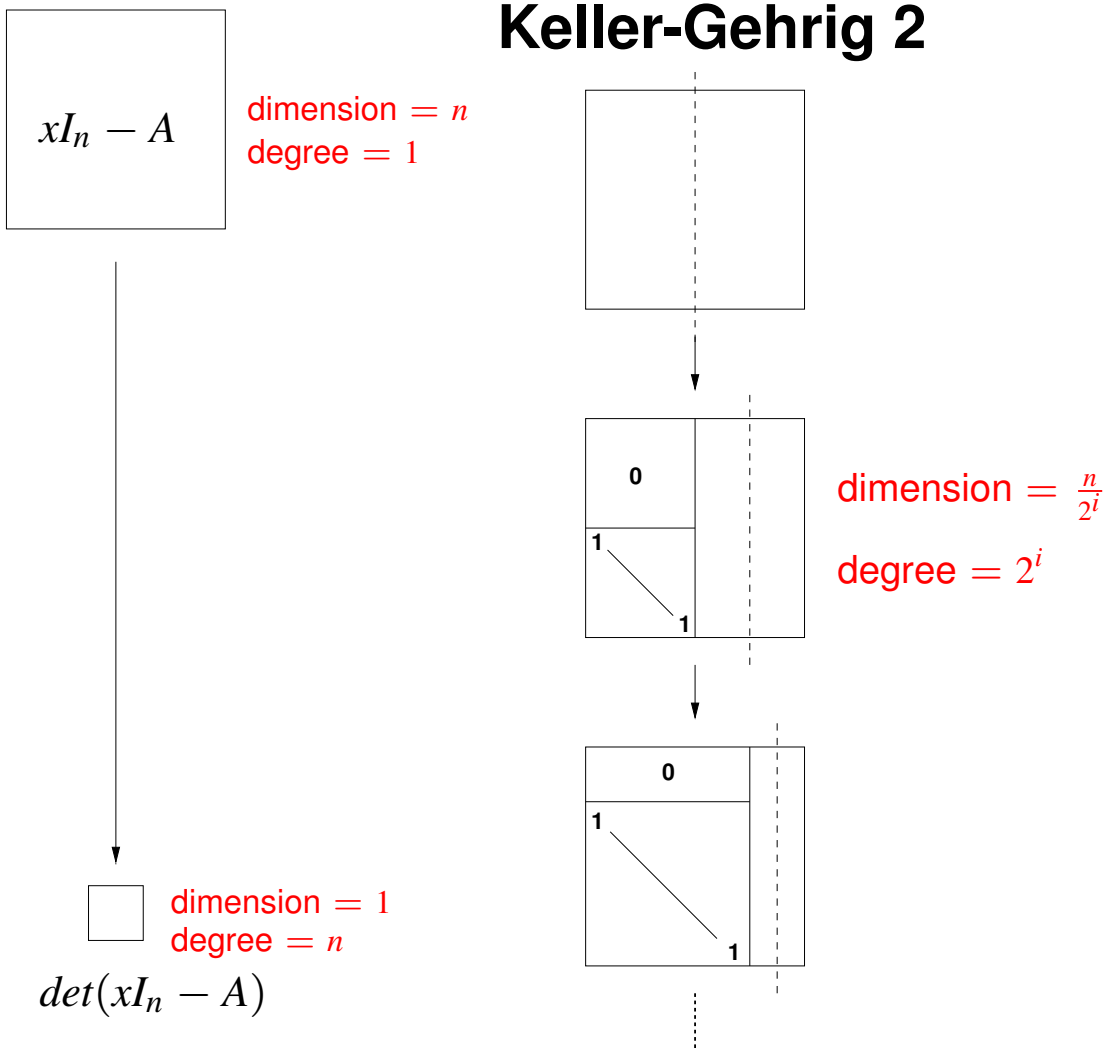
Overall complexity: summing for each iteration:

$$\sum_{k=1}^{n} k\left(\frac{n}{k}\right)^{\omega} = n^{\omega}\sum_{k=1}^{n}\left(\frac{1}{k}\right)^{\omega-1} = \zeta(\omega-1)n^{\omega} = \mathcal{O}\left(n^{\omega}\right)$$

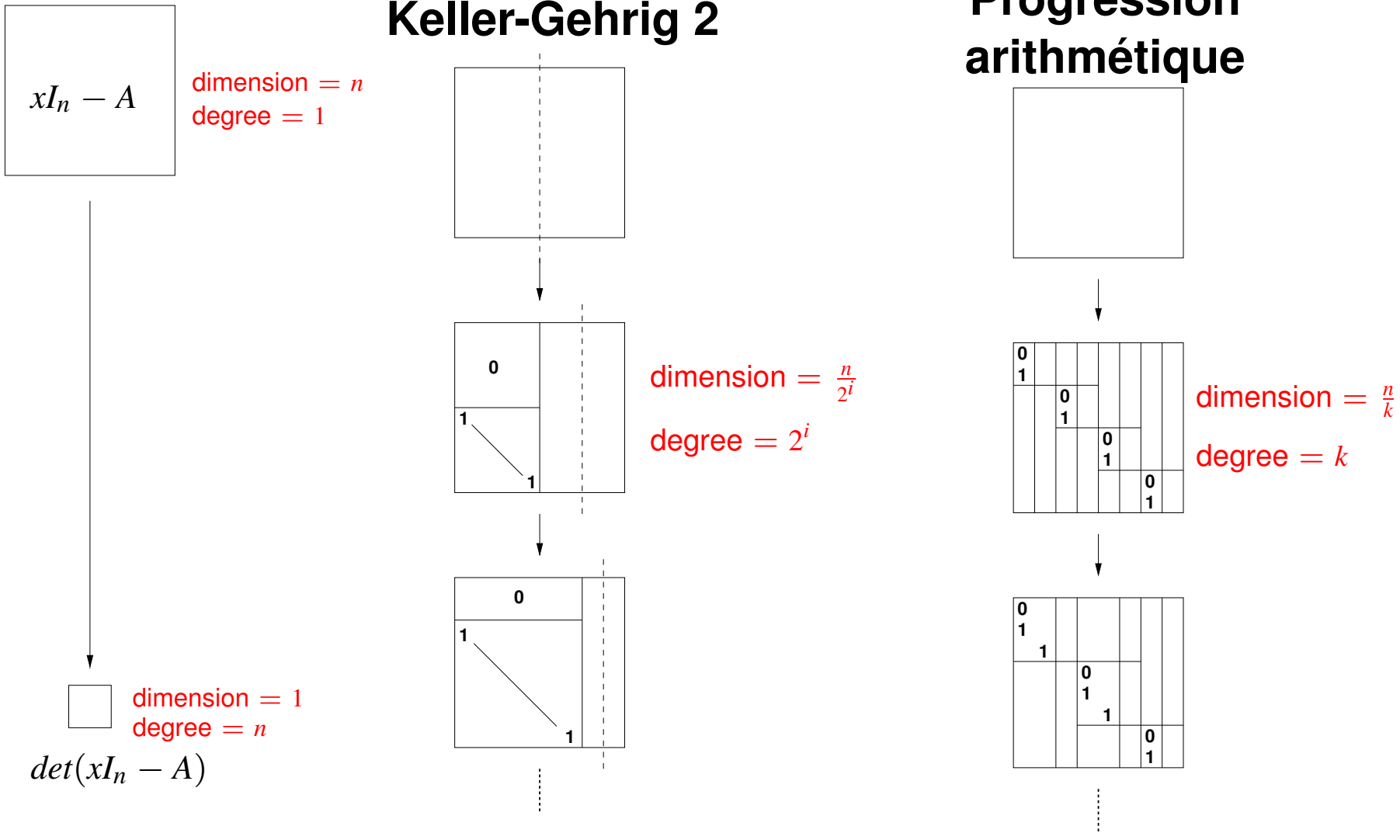
A new type of reduction



A new type of reduction



A new type of reduction



Conclusion

Réductions vers des briques de base

MatMul: récursif par bloc $\sum_{i=1}^{\log n} n \left(\frac{n}{2^i}\right)^{\omega-1} = \mathcal{O}\left(n^\omega\right)$ (Gauss)

MatMul: Itératif $\sum_{k=1}^{n} n \left(\frac{n}{k}\right)^{\omega-1} = \mathcal{O}\left(n^\omega\right)$ (Frobenius)

SysLin: over $\mathbb{Z}$ (forme de Hermite)

Compromis taille/dimension

- Forme de Hermite : perturbations de rang 1 pour réduire le déterminant
- Forme de Frobenius : degré k, dimension n/k pour $k = 1 \ldots n$