Calcul efficace de la forme normale de Hermite de matrices entières

Clément PERNET
LIG, Grenoble Université
travail conjoint avec William Stein (U. of Washington)

Séminaire du projet ALGORITHMS,
11 janvier 2010

Introduction : formes normales matricielles

Etant donné une transformation $B = f(A)$,

- Identifier les invariants
- Représentant unique de la classe d'équivalence
- Faciliter les calculs (forme structurée)

Différentes types:

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

- forme normal de Frobenius (ou forme normale rationnelle):

$$F = \begin{bmatrix} C_{P_0} & & & \\ & C_{P_1} & & \\ & & \ddots & \\ & & & C_{P_k} \end{bmatrix}, \text{ avec } p_{i+1} | p_i \text{ et } P_0 = \text{MinPoly}(A).$$

- Méthode de Krylov ou élimination ZigZag

Introduction : formes normales matricielles

Etant donné une transformation $B = f(A)$,

- Identifier les invariants
- Représentant unique de la classe d'équivalence
- Faciliter les calculs (forme structurée)

Différentes types:

Equivalence dans un corps: $B = UA$, où U est inversible

- Forme échelonnée réduite:

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

- Elimination de Gauss-Jordan

Introduction : formes normales matricielles

Etant donné une transformation $B = f(A)$,

- Identifier les invariants
- Représentant unique de la classe d'équivalence
- Faciliter les calculs (forme structurée)

Différentes types:

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

- Forme normale de Hermite: $H = \begin{bmatrix} p_1 & * & x_{1,2} & * & * & x_{1,3} & * \\ & & p_2 & * & * & x_{2,3} & * \\ & & & & & p_3 & * \\ & & & & & & \end{bmatrix}$, avec

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

Motivation

Théorie des Nombres:

- Forme échelonnée réduite dans K $\Rightarrow K$-espaces-vect.
- Forme normale de Hermite dans $\mathbb{Z}$ $\Rightarrow \mathbb{Z}$-modules

Motivation

Théorie des Nombres:

- Forme échelonnée réduite dans K $\Rightarrow K$-espaces-vect.
- Forme normale de Hermite dans $\mathbb{Z}$ $\Rightarrow \mathbb{Z}$-modules

Calcul de la saturation d'un sous-module

- M un sous-module de $\mathbb{Z}^n$. Saturation: $\mathbb{Z}^n \cap (\mathbb{Q}M)$.
- Utilisé pour le calcul avec les formes modulaires.

Lien:

- A, une base de M, H la forme de Hermite A^T.
- $(H^{-1}A)^T$ est une base de la saturation de M

Introduction: Modèles de complexité

Calcul dans $\mathbb{Z}$: complexité binaire $f(n, \log \|A\|_\infty)$

- Multiplication d'entiers: $M(k) = \mathcal{O}(n \log n \log^* n) = \mathcal{O}^\sim(n)$
- Multiplication de matrices: $\mathcal{O}(n^\omega M(\log \|A\|)) = \mathcal{O}^\sim(n^\omega \log \|A\|)$

Introduction: Modèles de complexité

Calcul dans $\mathbb{Z}$: complexité binaire $f(n, \log \|A\|_\infty)$

- Multiplication d'entiers: $M(k) = \mathcal{O}(n \log n \log^* n) = \mathcal{O}^\sim(n)$
- Multiplication de matrices: $\mathcal{O}(n^\omega M(\log \|A\|)) = \mathcal{O}^\sim(n^\omega \log \|A\|)$

Similitude dans un corps: Forme Normale de Frobenius

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

Introduction: Modèles de complexité

Calcul dans $\mathbb{Z}$: complexité binaire $f(n, \log \|A\|_\infty)$

- Multiplication d'entiers: $M(k) = \mathcal{O}(n \log n \log^* n) = \mathcal{O}^\sim(n)$
- Multiplication de matrices: $\mathcal{O}(n^\omega M(\log \|A\|)) = \mathcal{O}^\sim(n^\omega \log \|A\|)$

Similitude dans un corps: Forme Normale de Frobenius

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

Equivalence dans un corps: Forme échelonnée réduite

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

Equivalence dans $\mathbb{Z}$: Forme Normale de Hermite

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

Plan

Plan

Quelques ingrédients pour le calcul exact

De la complexité...

Briques de base

Dense, dans $\mathbb{Z}_p$:	Produit de matrice	$\mathcal{O}(n^\omega)$
Boite noire, dans $\mathbb{Z}_p$:	Polynôme minimal	$\mathcal{O}(ndE(n))$
Dense, dans $\mathbb{Z}, \mathbb{Q}$:	système linéaire	$\tilde{\mathcal{O}}(n^\omega \log \|\|A\|\|)$

Quelques ingrédients pour le calcul exact

De la complexité...

Briques de base

Dense, dans $\mathbb{Z}_p$: Produit de matrice $\mathcal{O}(n^\omega)$

Boite noire, dans $\mathbb{Z}_p$: Polynôme minimal $\mathcal{O}(ndE(n))$

Dense, dans $\mathbb{Z}, \mathbb{Q}$: système linéaire $\mathcal{O}^\sim(n^\omega \log \|A\|)$

Réductions: exemple `Det`

Dense, dans $\mathbb{Z}_p$: [Bunch & Hopcroft 74] $\mathcal{O}(n^\omega)$

- Det $\Rightarrow$LU récursif par bloc $\Rightarrow$MM

Boîte noire, dans $\mathbb{Z}_p$: [Chen & Al. 01] $\mathcal{O}\left(n^2 E(n)\right)$

- Det $\Rightarrow$Charpoly $\Rightarrow$Minpoly + Précond.

Dense, dans $\mathbb{Z}, \mathbb{Q}$: [Abbott & Al. 99] $\mathcal{O}^\sim(n^\omega \log \|A\|)$

- Det $\Rightarrow$Dernier facteur invariant $\Rightarrow$LinSys

Quelques ingrédients pour le calcul exact

De la complexité...

Briques de base

Dense, dans $\mathbb{Z}_p$: Produit de matrice $\mathcal{O}(n^\omega)$

Boite noire, dans $\mathbb{Z}_p$: Polynôme minimal $\mathcal{O}(ndE(n))$

Dense, dans $\mathbb{Z}, \mathbb{Q}$: système linéaire $\mathcal{O}^\sim(n^\omega \log \|A\|)$

Réductions: exemple `Det`

Dense, dans $\mathbb{Z}_p$: [Bunch & Hopcroft 74] $\mathcal{O}(n^\omega)$

- Det $\Rightarrow$LU récursif par bloc $\Rightarrow$MM

Boîte noire, dans $\mathbb{Z}_p$: [Chen & Al. 01] $\mathcal{O}(n^2 E(n))$

- Det $\Rightarrow$Charpoly $\Rightarrow$Minpoly + Précond.

Dense, dans $\mathbb{Z}, \mathbb{Q}$: [Abbott & Al. 99] $\mathcal{O}^\sim(n^\omega \log \|A\|)$

- Det $\Rightarrow$Dernier facteur invariant $\Rightarrow$LinSys

.... à la mise en pratique

Le produit de matrices dans $\mathbb{Z}_p$

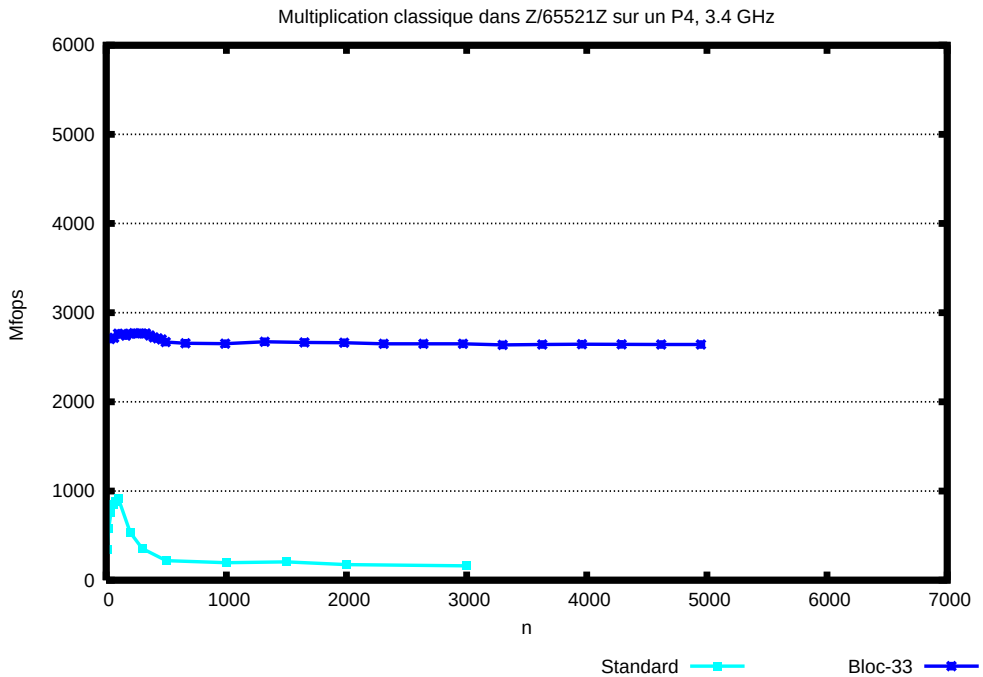
Principe:

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

Le produit de matrices dans $\mathbb{Z}_p$

Principe:

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

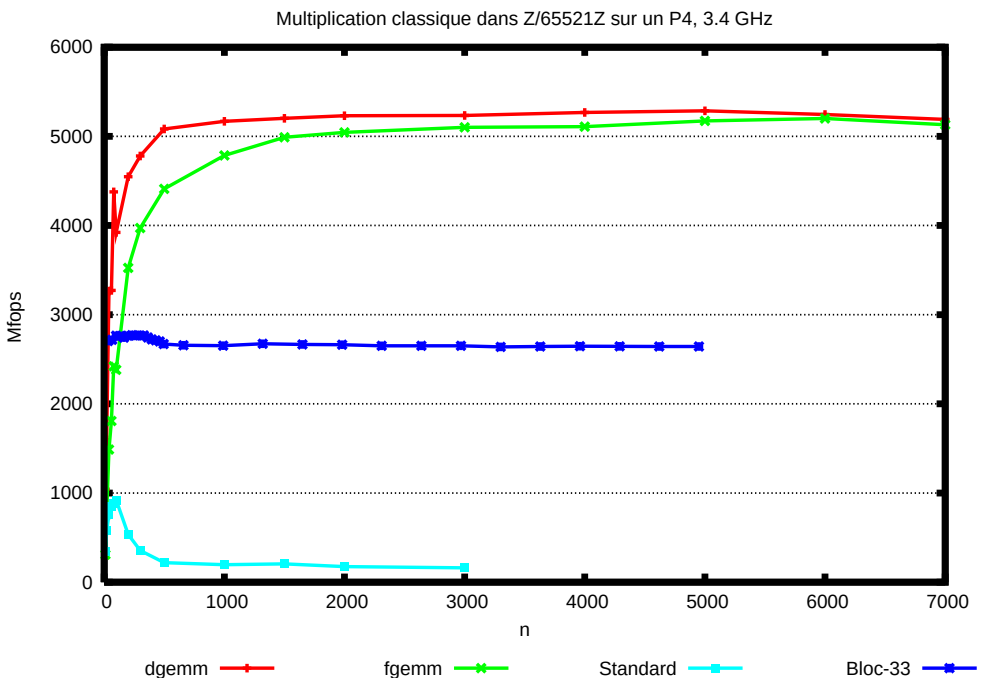


Le produit de matrices dans $\mathbb{Z}_p$

Principe:

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

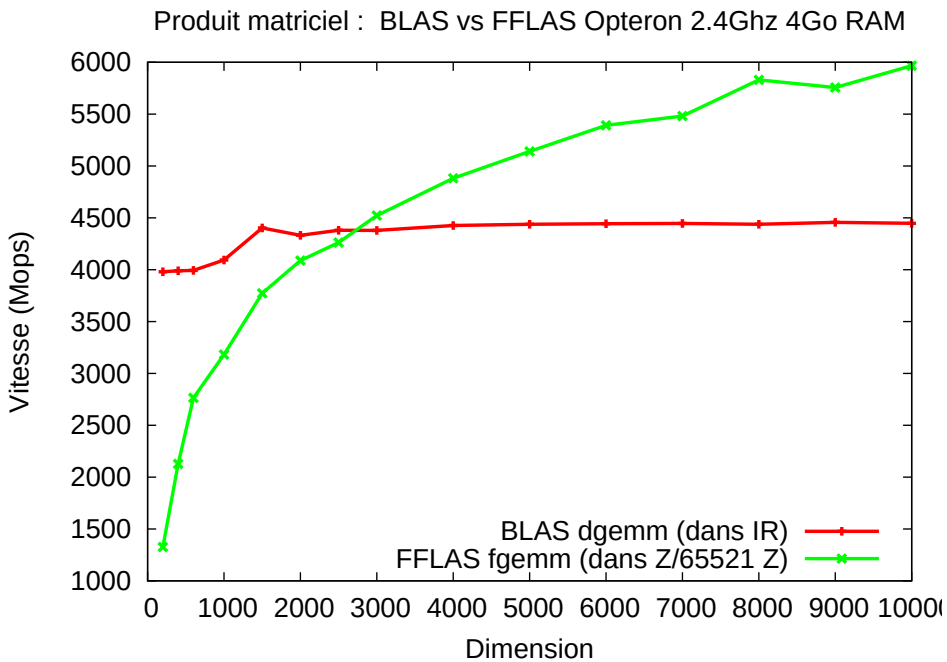
  $\Rightarrow$repose sur les BLAS existants



Le produit de matrices dans $\mathbb{Z}_p$

Principe:

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



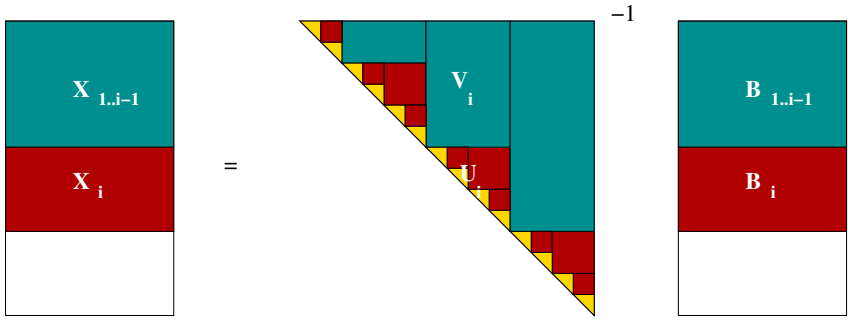
Autres routines d'algèbre linéaire dense

- Réduction au produit de matrice
- Bornes pour la réduction modulaire différée

Autres routines d'algèbre linéaire dense

- Réduction au produit de matrice
- Bornes pour la réduction modulaire différée

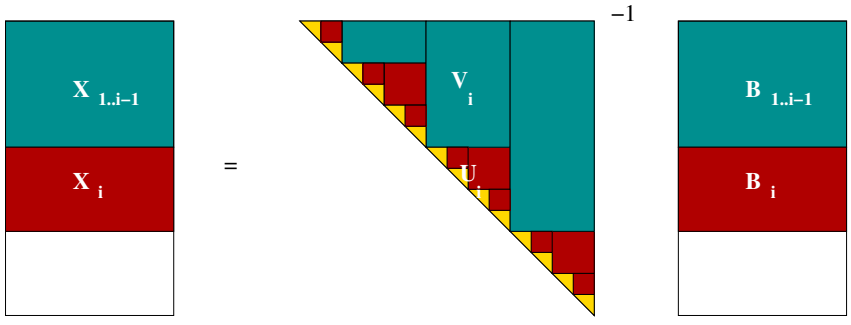
$\Rightarrow$Algorithmes par blocs en cascade



Autres routines d'algèbre linéaire dense

- Réduction au produit de matrice
- Bornes pour la réduction modulaire différée

$\Rightarrow$Algorithmes par blocs en cascade



	n	1000	2000	3000	5000	10 000
TRSM	$\frac{ftrsm}{dtrsm}$	1,66	1,33	1,24	1,12	1,01
LQUP	$\frac{lqup}{dgetrf}$	2,00	1,56	1,43	1,18	1,07
INVERSE	$\frac{inverse}{dgetrf+dgetri}$	1.62	1.32	1.15	0.86	0.76

Polynôme caractéristique:

n	500	5000	15 000
`LinBox`	0.91s	4m44s	2h20m
`magma-2.13`	1.27s	15m32s	7h28m

Système linéaire dans $\mathbb{Z}$

Restes Chinois: $\mathcal{O}\left(n^{\omega+1}(\log n + \log \|A\|)\right)$

Remontée p-adique: $\mathcal{O}^{\sim}\left(n^3 \log \|A\|\right)$

Remontée de haut rang: $\mathcal{O}^{\sim}\left(n^{\omega} \log \|A\|\right)$

Système linéaire dans $\mathbb{Z}$

Restes Chinois: $\mathcal{O}\left(n^{\omega+1}(\log n + \log \|A\|)\right)$

Remontée p-adique: $\mathcal{O}^{\sim}\left(n^3 \log \|A\|\right)$

Remontée de haut rang: $\mathcal{O}^{\sim}\left(n^{\omega} \log \|A\|\right)$

Repose sur

- LU dans $\mathbb{Z}_p$
- arithmétique entière rapide: Karatsuba, Toom-Cook, FFT (`GMP`)
- compromis dimension/iterations: remontée multi-adique

Plan

Contexte

Complexité:

- pire cas: $\mathcal{O}^{\sim}\left(n^5 \log \|A\|\right)$
- 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 (dist. uniforme), fréquentes en théorie des nombre

Contexte

Complexité:

- pire cas: $\mathcal{O}^{\sim}\left(n^5 \log \|A\|\right)$
- 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 (dist. uniforme), fréquentes en théorie des nombre

Mise en oeuvre:

- Reduction au briques de base optimisées
- amélioration des constantes

Algorithme naïf

begin
  foreach i **do**
    $(g, t_i, \ldots, t_n) = \mathrm{xgcd}(A_{i,i}, A_{i+1,i}, \ldots, A_{n,i})$;
    $L_i \leftarrow \sum_{j=i+1}^{n} t_j L_j$;
    for $j = i+1 \ldots n$ **do**
      $L_j \leftarrow L_j - \frac{A_{j,i}}{g} L_i$; /* élimine */
    for $j = 1 \ldots i-1$ **do**
      $L_j \leftarrow L_j - \lfloor \frac{A_{j,i}}{g} \rfloor L_i$; /* réduit */
end

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

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

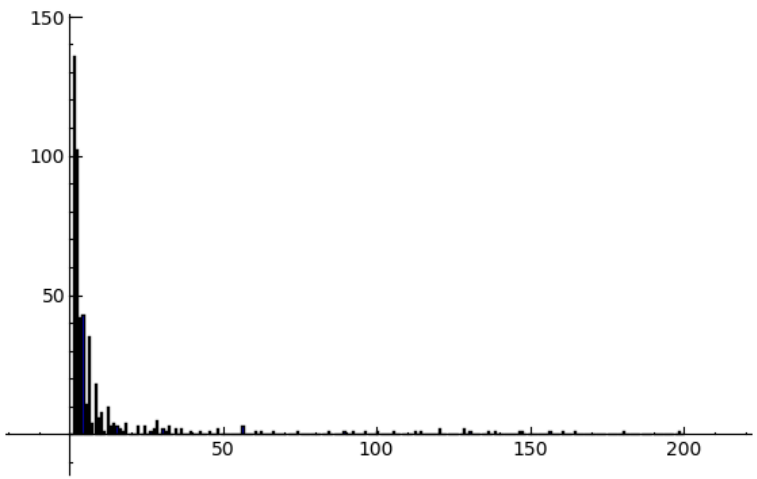
[Micciancio & Warinschi 01]

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

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

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

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



Algorithme de Micciancio & Warinschi

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 */`

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

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 */`

end

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 */

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

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 */

end

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

- 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

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

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

$\Rightarrow$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

AddRow

Problème

Calculer la forme de Hermite H' de $\begin{bmatrix} B \\ a^T \end{bmatrix}$ connaissant H, celle de B.

- Réduit a^T par rapport aux pivots de H
- Met à jour H si besoin (xgcd)
- Si b reste non nulle, l'insérer à la ligne appropriée (forme échelonnée)

Cas général, non inversible

- Calcul du profil de rang par ligne (par PLUQ mod p aléatoire)
- Extraire la sous-matrice $r \times r$ inversible
- Calculer sa forme de Hermite
- Ajouter les colonnes restantes: AddCol, avec un bloc colonne
- Ajouter les colonnes restantes: AddRow, voire LLL

Plan

Benchmarketing

- Implantation dans `Sage`
- `IML`, `LinBox`, `GMP`
- Xeon-2.66Ghz, Linux

n	ntl (8)	ntl (32)	pari (8)	pari (32)	gap (8)	gap (32)	sage (32)
50	0.09	0.31	0.09	0.26	0.12	0.19	0.24
250	74.58	494.46	163.69	776.91	36.08	165.49	7.3
500	1975	12199	2925	15263	712	3982	47

Matrices $n \times n$, coefficients dans $[-2^b, 2^b]$. où $b = 8, 32$.

Benchmarketing

Sage

n	8 bits	32 bits	128 bits	256 bits	512 bits
50	0.1	0.2	0.7	2.8	12.45
250	4.7	6.8	34.8	105.4	470.22
500	25.7	41.1	183.9	550.7	2169.41
1000	177.0	274.3	1113.4	2976.9	10506
2000	1500.4	2837.5			
4000	11537.3	17105.9			

Magma

n	8 bits	32 bits	128 bits	256 bits	512 bits
50	0.0	0.1	0.3	0.9	2.73
250	1.0	13.0	68.1	223.4	764.6
500	7.7	203.5	990.5	2630.9	7100.91
1000	72.8	1795.4	8076.0	21370.2	58339
2000	886.1	14917.8			
4000	6096.5				

Benchmarketing