Gauss-Bareiss et méthodes modulaires : calculer efficacement avec des coefficients entiers
Notes de Cours- Prépa Agreg option C*

Thomas Hausberger

27 mars 2025

Table des matières

Il est question de matrices à coefficients dans $\mathbb{Z}$ dont on souhaite calculer l'inverse (dans $\mathrm{GL}_n(\mathbb{Q})$ à priori) ou juste le déterminant (dans $\mathbb{Z}$) et de systèmes linéaires à coefficients dans $\mathbb{Z}$ que l'on souhaite résoudre (dans $\mathbb{Q}^n$). Les algorithmes basés sur le pivot de Gauss où l'on travaille dans $\mathbb{Q}$ sont coûteux car la manipulation de coefficients fractionnaires nécessite beaucoup plus d'opérations que les coefficients entiers. On se propose de comparer deux stratégies utilisées en calcul formel et qui tirent parti du fait que les coefficients sont dans $\mathbb{Z}$: d'une part la méthode de Gauss-Bareiss et d'autre part des méthodes modulaires (majoration a priori, calcul modulo différents nombres premiers, reconstruction par le théorème chinois).

1. Algorithme de Gauss-Bareiss

Bareiss a inventé une variante astucieuse du pivot de Gauss dans laquelle toutes les divisions restent dans $\mathbb{Z}$. Elle est valable en fait dans tout anneau intègre et est très souvent utilisée en calcul formel.

1.1. pivot de Gauss sans division

Lors du pivot de Gauss classique appliqué à une matrice $A = ((a_{i,j}))$ (supposée inversible pour simplifier l'exposition), on obtient des 0 sous le $k+1$-ième pivot $a^{(k)}_{k+1,k+1}$ de $A^{(k)}$ par des opérations sur les lignes :

$$L_i^{(k+1)} = L_i^{(k)} - \frac{a^{(k)}_{i,k+1}}{a^{(k)}_{k+1,k+1}} L_{k+1}^{(k)} \text{ pour } i > k+1.$$

En fait, on a d'abord permuté, si nécessaire, $L^{(k)}_{k+1}$ avec une ligne $L^{(k)}_i$, $i > k+1$, pour avoir un $k+1$-ième pivot en position $k+1, k+1$. On obtient $A^{(k+1)}$.

Pour éviter de faire des divisions, il suffit de faire plutôt :

$$L_i^{(k+1)} = a^{(k)}_{k+1,k+1} L_i^{(k)} - a^{(k)}_{i,k+1} L_{k+1}^{(k)} \text{ pour } i > k+1. \tag{1}$$

Cependant, les coefficients doublent de taille à chaque étape ; cette croissance exponentielle des coefficients n'est pas acceptable.

1.2. méthode de Gauss-Bareiss

La méthode Bareiss repose sur le fait (déjà connu depuis Jordan) que le k-ième pivot $a^{(k-1)}_{k,k}$ divise tous les coefficients des lignes $L_i^{(k+1)}$ pour tout $i > k+1$ (avec les formules (1)). Il propose donc :

$$L_i^{(k+1)} = \frac{a^{(k)}_{k+1,k+1} L_i^{(k)} - a^{(k)}_{i,k+1} L_{k+1}^{(k)}}{a^{(k-1)}_{k,k}} \text{ pour } i > k+1. \tag{2}$$

Démonstration. On considère les mineurs, notés aussi $a^{(k)}_{i,j}$ (vous comprendrez tout de suite pourquoi) :

$$a^{(k)}_{i,j} = \begin{vmatrix} a_{1,1} & \cdots & a_{1,k} & a_{1,j} \\ \vdots & & \vdots & \vdots \\ a_{k,1} & \cdots & a_{k,k} & a_{k,j} \\ a_{i,1} & \cdots & a_{i,k} & a_{i,j} \end{vmatrix} \quad (i \geq k+1, j \geq k+1),$$

L'identité fondamentale est

$$a^{(k+1)}_{i,j} = \frac{1}{a^{(k-1)}_{k,k}} \begin{vmatrix} a^{(k)}_{k+1,k+1} & a^{(k)}_{k+1,j} \\ a^{(k)}_{i,k+1} & a^{(k)}_{i,j} \end{vmatrix} \quad \forall i > k+1, \forall j > k+1$$

(où $a^{(0)}_{i,j} = a_{i,j}$). Cela montre que les coefficients des $A^{(k)}$ (qui sont définis par cette relation de récurrence et $a^{(k+1)}_{i,j} = a^{(k)}_{i,j}$ si $i \leq k+1$ ou $j < k+1$ et $a^{(k+1)}_{i,j} = 0$ si $i > k+1$ et $j = k+1$) dans la méthode de Gauss-Bareiss sont en fait des mineurs de A, donc en particulier des entiers relatifs.

Démontrons l'identité fondamentale. Tout d'abord, nous utilisons le résultat suivant : si M est une matrice qui se décompose par blocs en $M = \begin{pmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{pmatrix}$, le bloc

carré $M_{1,1}$ de taille k étant une matrice inversible, alors $\det M = \det M_{1,1} \det(M_{2,2} - M_{2,1}M_{1,1}^{-1}M_{1,2})$. En effet, il suffit d'écrire

$$M = \begin{pmatrix} M_{1,1} & 0 \\ M_{2,1} & I_{n-k} \end{pmatrix} \cdot \begin{pmatrix} I_k & M_{1,1}^{-1}M_{1,2} \\ 0 & M_{2,2}-M_{2,1}M_{1,1}^{-1}M_{1,2} \end{pmatrix}$$

Appliquant ce résultat à la matrice

$$M = \begin{pmatrix} a_{1,1} & \cdots & a_{1,k} & a_{1,j} \\ \vdots & & \vdots & \vdots \\ a_{k,1} & \cdots & a_{k,k} & a_{k,j} \\ a_{i,1} & \cdots & a_{i,k} & a_{i,j} \end{pmatrix},$$

on obtient l'égalité

$$a_{i,j}^{(k)} = a_{k,k}^{(k-1)}(a_{i,j} - {}^t a M_{1,1}^{-1} b),$$

où ${}^t a = (a_{i,1}, \ldots, a_{i,k})$ et ${}^t b = (a_{1,j}, \ldots a_{k,j})$ (la seconde matrice est égale à son déterminant).

Prenant

$$M = \begin{pmatrix} a_{1,1} & \cdots & a_{1,k} & a_{1,k+1} & a_{1,j} \\ \vdots & & \vdots & \vdots & \vdots \\ a_{k,1} & \cdots & a_{k,k} & a_{k,k+1} & a_{k,j} \\ a_{k+1,1} & \cdots & a_{k+1,k} & a_{k+1,k+1} & a_{k+1,j} \\ a_{i,1} & \cdots & a_{i,k} & a_{i,k+1} & a_{i,j} \end{pmatrix},$$

avec le même bloc carré $M_{1,1}$ que précédemment, on obtient

$$a_{i,j}^{(k+1)} = a_{k,k}^{(k-1)} \det(M_{2,2} - M_{2,1}M_{1,1}^{-1}M_{1,2}).$$

Or tout coefficient de $M_{2,2} - M_{2,1}M_{1,1}^{-1}M_{1,2}$ est de la forme $a_{i',j'} - {}^t a M_{1,1}^{-1} b$, où ${}^t a = (a_{i',1}, \ldots, a_{i',k})$ et ${}^t b = (a_{1,j'}, \ldots a_{k,j'})$. On a donc

$$a_{k,k}^{(k-1)}(M_{2,2} - M_{2,1}M_{1,1}^{-1}M_{1,2}) = \begin{pmatrix} a_{k+1,k+1}^{(k)} & a_{k+1,j}^{(k)} \\ a_{i,k+1}^{(k)} & a_{i,j}^{(k)} \end{pmatrix},$$

d'où, en prenant le déterminant :

$$(a_{k,k}^{(k-1)})^2 \det(M_{2,2} - M_{2,1}M_{1,1}^{-1}M_{1,2}) = \begin{vmatrix} a_{k+1,k+1}^{(k)} & a_{k+1,j}^{(k)} \\ a_{i,k+1}^{(k)} & a_{i,j}^{(k)} \end{vmatrix}.$$

Finalement :

$$a_{i,j}^{(k+1)} a_{k,k}^{(k-1)} = \begin{vmatrix} a_{k+1,k+1}^{(k)} & a_{k+1,j}^{(k)} \\ a_{i,k+1}^{(k)} & a_{i,j}^{(k)} \end{vmatrix} \quad \forall i > k+1, \forall j > k+1.$$

□

1.3. analyse de la complexité

On s'intéresse à la résolution d'un système linéaire régulier de n équations à n inconnues. Le pivot de Gauss classique a une meilleure complexité (en temps) arithmétique (dans $\mathbb{Q}$) que la méthode de Barreiss, mais une complexité binaire moindre. Nous allons préciser cela.

Proposition 1.3.1. *Le nombre d'opérations arithmétiques nécessaires pour résoudre le système est équivalent à $\frac{2}{3}n^3$ pour le pivot de Gauss et à $\frac{4}{3}n^3$ pour la méthode de Gauss-Bareiss.*

Démonstration. Dans les deux cas, le passage de $A^{(k-1)}$ à $A^{(k)}$ (pour $1 \leq k \leq n$) nécessite de calculer $n-k$ lignes (les k premières restent inchangées), et $n-k+1$ coefficients pour chaque ligne (car il y a $k-1$ zéros inchangés par ligne). Pour chaque coefficient, on compte une multiplication par une fraction et une soustraction pour Gauss classique contre deux multiplications, une soustraction et une division entière pour Gauss-Bareiss. De plus, $\sum_{k=1}^{n}(n-k+1)^2 = \sum_{k=1}^{n} k^2 = n(n-1)(2n-1)/6 \sim n^3/3$, d'où le résultat. □

Proposition 1.3.2. *Si tous les coefficients du système sont au plus de taille t (en base b), alors la taille des coefficients des matrices $A^{(k-1)}$ dans l'algorithme de Gauss-Bareiss est majorée par $k(\frac{1}{2}\log_b k + t)$.*

Autrement dit, la croissance de la taille des coefficients est à peine plus que linéaire. Cela permet de majorer la complexité binaire, sachant qu'une addition de deux entiers n et n' a une complexité binaire au pire en $O(max(\log n, \log n'))$ et une multiplication ou une division entière $O(\log n \log n')$ (arithmétique élémentaire).

Démonstration. On utilise la formule de Hadamard : $|\det M|^2 \leq \prod_{i=1}^{k}\left(\sum_{j=1}^{k}|m_{i,j}|^2\right)$ pour tout matrice $M = ((m_{i,j}))$ à coefficients complexes.

Comme les coefficients des $A^{(k-1)}$ sont des mineurs extraits de A, dont les coefficients $a_{i,j}$ sont majorés par b^t, on a $|a_{i,j}^{(k-1)}|^2 \leq \prod_{i=1}^{k}\left(\sum_{j=1}^{k} b^{2t}\right) = (kb^{2t})^k$, d'où le résultat. □

Exercice. En déduire la complexité binaire de l'algorithme de Gauss-Bareiss.

Remarque. On peut montrer que la complexité binaire du pivot de Gauss classique est polynômiale en la taille de la donnée en entrée (mais c'est difficile !), ceci contrairement au fait que la formule $a_{i,j}^{(k+1)} = a_{i,j}^{(k)} - \frac{a_{i,k+1}^{(k)}}{a_{k+1,k+1}^{(k)}} a_{k+1,j}^{(k)}$ donne une majoration exponentielle pour $b_k = \max(|u_{i,j}^{(k)}|, |v_{i,j}^{(k)}|)$ (où $a_{i,j}^{(k)} = \frac{u_{i,j}^{(k)}}{v_{i,j}^{(k)}} \in \mathbb{Q}$) : on a $b_k \leq 2b_{k-1}^4 \leq 2^{1+4+\ldots+4^{k-1}} b_0^{4^k} = 2^{(4^k-1)/3} b_0^{4^k}$. Or la donnée en entrée est de taille $n^2 \log b_0$.

2. Méthodes modulaires

Nous allons expliquer comment calculer le déterminant d'une matrice à coefficients dans $\mathbb{Z}$ par une méthode modulaire. On peut étendre la méthode au cas d'un anneau euclidien A quelconque (par exemple $K[x]$) ; citons également d'autres applications des méthodes modulaires : résoudre des systèmes linéaires, calculer le pgcd de polynômes, etc...

2.1. Algorithme des restes chinois

Soit $m_1, \ldots, m_n$ des nombres entiers premiers entre eux deux à deux. Le théorème des restes chinois affirme que $\mathbb{Z}/(m_1 \cdots m_n)\mathbb{Z} \simeq \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_n\mathbb{Z}$. L'ensemble des solutions du système de congruences

$$\begin{cases} x \equiv c_1 \mod m_1 \\ \ldots \\ x \equiv c_n \mod m_n \end{cases}$$

est $x_0 + (m_1 \cdots m_n)\mathbb{Z}$, où x_0 désigne une solution particulière. Celle-ci se trouve par l'algorithme des restes chinois :

1. $m \leftarrow \prod m_i$
2. pour $1 \leq i \leq n$ calculer les coefficients de Bezout par l'algorithme d'Euclide (semi-)étendu : $s_i \frac{m}{m_i} + t_i m_i = 1$ (en fait, s_i suffit) puis $\nu_i \leftarrow c_i s_i$
3. $x_0 \leftarrow \sum_{i=1}^{n} \nu_i \frac{m}{m_i}$

   Etudions la complexité binaire :
   - dans l'étape 1, on calcule successivement $m_1 m_2$, $(m_1 m_2)m_3$, ...,$(m_1 \ldots m_{n-1})m_n$, d'où un coût au plus

$$\sum_{i=2}^{n} \log(\prod_{j=1}^{i-1} m_j) \log m_i = \sum_{2 \leq j < i \leq n} \log m_j \log m_i \leq \sum_{i,j} \log m_i \log m_j = \log^2 \prod m_i$$

   - dans l'étape 2, on calcule d'abord $\frac{m}{m_i}$, d'où un coût majoré par $\sum_{i=1}^{n} \log m \log m_i = \log^2 m$. Ensuite, on sait que le calcul du pgcd de 2 entiers a et b ($a > b > 0$) a une complexité binaire au pire de $O(\log a \log b)$ et l'algorithme étendu est de même complexité que l'algorithme de base. D'où encore un $O(\log^2 \prod m_i)$ pour le calcul des coefficients s_i car $\sum_{i=1}^{n} \log \frac{m}{m_i} \log m_i \leq \log^2 \prod m_i$. Enfin, on sait que $|s_i| \leq m_i/2$ et $c_i \leq m_i$, d'où un coût majoré par $\sum \log^2 m_i$ pour le calcul des ν_i, en $O(\log^2 \prod m_i)$
   - Dans l'étape 3, comme $|\nu_i| \leq m_i^2$, on a un coût au plus $2 \sum \log m_i \log \frac{m}{m_i}$, i.e. en $O(\log^2 \prod m_i)$ pour le calcul des $\nu_i \frac{m}{m_i}$, puis en $O(n \log \prod m_i) \subset O(\log^2 \prod m_i)$ pour la somme finale.

   Ainsi la complexité binaire de l'algorithme des restes chinois est en $O(\log^2 \prod m_i)$.

2.2. Algorithme de Garner

Il est basé sur la représentation des entiers en base mixte :

Proposition 2.2.1. *Soit n entiers $m_1, \ldots, m_n$ distincts ou non. L'application ψ : $[0, m_1[\times \ldots \times [0, m_n[\to [0, m_1 \ldots m_n[$, $(a_1, \ldots, a_n) \mapsto a_1 + a_2 m_1 + a_3 m_1 m_2 + \ldots + a_n m_1 \ldots m_{n-1}$ définit une bijection.*

Remarque. En prenant $m_i = b$ pour tout i, on retrouve la décomposition en base b, d'où la terminologie.

Démonstration. L'application ψ est bien définie : par récurrence sur r, on montre que $a_1 + a_2 m_1 + a_3 m_1 m_2 + \ldots + a_r m_1 \ldots m_{r-1} < \prod_{i=1}^{r} m_i$. Les deux ensembles ont même cardinal $\prod m_i$. On montre la surjectivité : soit $x \in [0, \prod m_i[$; on effectue des divisions euclidiennes successives (à reste positif) par les m_i : $x = q_1 m_1 + a_1$, $q_1 = q_2 m_2 + a_2$, ..., $q_{n-1} = q_n m_n + a_n$. Alors $a_i \in [0, m_i[$ pour tout i et par récurrence on montre que $\psi(a_1, \ldots, a_n) = x$. □

Ainsi, lorsque les m_i sont premiers deux à deux, nous disposons de 2 bijections $[0, m_1[\times \ldots \times [0, m_n[\to [0, \prod m_i[$: l'une donnée par le théorème chinois, et l'autre par la décomposition en base mixte.

L'algorithme de Garner est un algorithme permettant de passer d'une écriture d'un entier $x \in [0, \prod m_i[$ donné par ses restes $\mod m_i$ (c'est la donnée du système de congruences) à l'écriture en base mixte, sans passer par les entiers (i.e. sans composer les deux bijections).

On rappelle que le système est :

$$\begin{cases} x \equiv c_1 \mod m_1 \\ \ldots \\ x \equiv c_n \mod m_n \end{cases}$$

et l'on cherche à écrire x sous la forme $x = a_1 + a_2 m_1 + a_3 m_1 m_2 + \ldots + a_n m_1 \ldots m_{n-1}$, avec $a_i \in [0, m_i[$. Nécessairement, $x \equiv a_1 \equiv c_1 \mod m_1$, d'où $a_1 = c_1$. Supposons avoir déterminé les a_i jusqu'au rang r, alors $x \equiv c_{r+1} \equiv a_1 + a_2 m_1 + a_3 m_1 m_2 + \ldots + a_{r+1} m_1 \ldots m_r \mod m_{r+1}$, d'où $a_{r+1} = (m_1 \ldots m_r)^{-1}(c_{r+1} - a_1 - a_2 m_1 - \ldots - a_r m_1 \ldots m_{r-1}) \mod m_{r+1}$.

D'où l'algorithme :

1. $a_1 \leftarrow c_1$, $x \leftarrow 0$, $m \leftarrow 1$
2. Pour i de 2 à n faire (dans l'ordre) $x \leftarrow x + a_{i-1} m$, $m \leftarrow m m_{i-1}$, $a_i = m^{-1}(c_i - x) \mod m_i$
3. Renvoyer la liste des a_i ou $x + a_n m$, selon le résultat à fournir.

L'algorithme de Garner coûte au plus $O(\log^2 \prod m_i)$ opérations binaires. En effet, on utilise $n-1$ fois l'algorithme d'Euclide semi-étendu pour calculer les inverses $m^{-1} \mod m_i$, avec un coût $\sum_{i=2}^{n} \log(\prod_{j=1}^{i-1} m_j) \log m_i \in O(\log^2 \prod m_i)$. L'addition dans $\mathbb{Z}/m_i\mathbb{Z}$ nécessite $O(\log m_i)$ et la multiplication $O(\log^2 m_i)$ opérations binaires. Au total, on trouve encore un $O(\log^2 \prod m_i)$.

L'avantage en pratique est que les calculs intermédiaires sont de taille moindre ; en plus, le résultat final est dans $[0, m_1 \ldots m_n[$, ce qui n'est pas le cas du résultat donné par l'algorithme des restes chinois. Cette différence n'apparaît pas dans notre étude de la complexité théorique (la différence passe dans le grand O). Garner est plus efficace en pratique (à vérifier).

Remarque. Gardner est équivalent au fait de faire

$$chinois(m_n, chinois(m_{n-1}, \ldots chinois(m_2, m_1) \cdots),$$

où *chinois* correspond à la résolution d'un système de deux congruences. La contribution à la complexité du calcul des inverses modulaires est dans ce cas $\sum \log m_i \log(\prod_{j<i} m_j)$

soit $\sum_{i=1}^{n} kt^2 = \frac{n(n-1)}{2}t^2$ où les m_i sont supposés de taille t. L'algorithme chinois présenté plus haut, consistant à tout faire d'un coup, donne quant à lui $\sum \log m_i \log(m/m_i)$ soit $n(n-1)t^2$. On gagne donc en gros un facteur 2. On peut même faire mieux (en regroupant les moduli deux par deux, puis les résultats deux par deux, donc les moduli quatre par quatre, *etc.*).

2.3. Calcul du déterminant

Plus généralement, soient $a_1, \ldots, a_k$ des entiers et $F = f(a_1, \ldots, a_k)$ une expression (définie en fonction des opérations arithmétique de $\mathbb{Z}$) à calculer. On suppose que l'on sait que $|F| \leq M$. La méthode est la suivante :

1. Choisir n nombres premiers distincts p_i tels que $m = \prod p_i > 2M$ (il se peut que certains premiers soient à rejeter : ceux pour lesquels $f(a_1 \mod p_i, \ldots, a_k \mod p_i) \neq F \mod p_i$; on dit alors que p_i est de « mauvaise réduction »)
2. Pour chaque i, calculer $c_i = F \mod p_i = f(a_1 \mod p_i, \ldots, a_k \mod p_i)$ (ce qui est peu couteux dans le corps $\mathbb{Z}/p_i\mathbb{Z}$)
3. Résoudre le système de congruences par l'algorithme de Garner ; soit x la solution dans $[0, m[$
4. Comme F est l'unique solution dans $[-M, M]$ de ce système (puisque $m > 2M$), alors $F = x$ ou $x - m$ si $x > M$.

Pour le calcul du déterminant d'une matrice $A = ((a_{i,j}))$ de taille r :

1. On utilise l'inégalité de Hadamard : notant $B = max|a_{i,j}|$, on a $|\det A| \leq r^{r/2}B^r = M$. Il suffit de prendre n premiers distincts où $n = E(\log_2(2M+1))$. En effet, leur produit m vérifiera $m \geq 2^n \geq 2M + 1 > 2M$. On peut prendre les n premiers nombres premiers.
2. On calcule $A_i = A \mod p_i$ puis $\det A_i \mod p_i$ en utilisant le pivot de Gauss dans $\mathbb{Z}/p_i\mathbb{Z}$

Remarque. Soit p_k le k-ième nombre premier. On peut montrer que $p_k < 2k \ln k$ pour $k \geq 20$. Par un crible d'Eratostène, on peut alors trouver les n premiers nombres premiers, ceci en $O(n \log^2 n \log\log n)$ opérations binaires.

La question naturelle est la suivante : faut-il mieux utiliser un seul nombre premier ($n = 1$), ou beaucoup de petits nombres premiers (comme suggéré plus haut) ?

On montre que la complexité binaire est en $O(r^5 \log^2(rB))$ si l'on prend un seul p, où r désigne la taille de la matrice, donc c'est légèrement quadratique en la taille $r^2 \log B$ de la donnée en entrée. Ce n'est pas de progrès par rapport à la méthode de Gauss-Bareiss. Par contre, avec beaucoup de nombres premiers, on fait mieux : en prenant $n = E(\log_2(2M+1))$ (noter que $n \in O(r \log(rB))$ et $\log \prod p_i \in O(n \log n)$), on montre que la complexité binaire est en $O(r^4 \log^2(rB)(\log^2 r + (\log\log B)^2))$ (cf. [VZGG]). On a gagné en gros un facteur r.

Références

[DST] J. Davenport, Y. Siret, E. Tournier, *«Calcul formel : systèmes et algorithmes de manipulations algébriques»*, Masson, 1986.

[SP] Ph. Saux Picart, *«Cours de calcul formel : algorithmes fondamentaux»*, ellipses, 1999.

[VZGG] J. Von Zur Gathen, J. Gerhard, *«Modern Computer Algebra»*, Cambridge University Press, second ed., 2003.