UFR Sciences
M2 Calcul Formel M. WEIMANN

Matrices

Beaucoup de problèmes de mathématiques peuvent se ramener à de l'algèbre linéaire. Pour en citer quelques uns :

- Systèmes d'équations linéaires
- Equations différentielles linéaires
- Résultants
- Bases de Gröbner
- Factorisation des polynômes (e.g. Berlekamp)

Développer des algorithmes efficaces pour effectuer les opérations fondamentales sur les matrices est donc un problème crucial du Calcul Formel. Les enjeux majeurs dans ce domaine sont la multiplication, l'inversion et la réduction des matrices que l'on va étudier un peu ainsi que quelques unes de leurs applications (inversion, déterminant, calcul du rang, polynôme caractéristique).

Référence (dont ce chapitre est essentiellement extrait, merci les auteurs)

- Bostan, Chyzak, Giusti, Lebreton, Lecerf, Salvy, Schost, *Algorithmes efficaces en Calcul Formel* (2017).

1 Complexité des opérations élémentaires

Soit K un corps. On note $\mathcal{M}_n(K)$ la K-algèbre des matrices carrées de taille $n \times n$ à coefficients dans K. La plupart des résultats et algorithmes s'étendent aux matrices rectangulaires lorsque cela fait sens (multiplication, pivot de Gauss, calcul du rang, mais évidemment pas l'inversion ou le déterminant par exemple).

On calcule le coût d'un algorithme comme le nombre d'opérations arithmétiques dans K. On suppose que les matrices sont représentées par l'ensemble de tous leurs coefficients (en général donné sous forme de listes de listes). Ainsi, la taille de l'entrée est n^2. On parle de *représentation dense* et d'*algorithmique dense*.

Remarque 1 (Algorithmique creuse et algorithmique structurée) Beaucoup de problèmes linéaires se formulent en termes de matrices possédant un nombre important d'éléments nuls, dites « creuses » (par exemple une matrice compagnon). Dans ce cas, l'algorithmique dense est inappropriée ; il est possible de tirer parti de la forme creuse en utilisant des outils mieux adaptés, à base de récurrences linéaires. On cherche alors à exprimer la complexité en fonction du nombre de coefficients non nuls. On parle d'*algorithmique creuse*. Dans la même idée, il existe des familles de matrices particulières, souvent associées à des applications linéaires entre espaces de polynômes, telles que les matrices de type Sylvester pour le résultant, de type Vandermonde pour l'évaluation et l'interpolation, etc. Pour ces types de matrices clairement identifiés, on peut développer une algorithmique qui tire profit de la structure, on parle d'*algorithmique structurée*. Schématiquement,

l'algorithmique des matrices denses est la plus lente, de complexité située entre $O(n^2)$ et $O(n^3)$. La complexité du calcul avec les matrices creuses est de l'ordre de $O(n^2)$, alors que celle des matrices structurées est en $O(n)$, à des facteurs logarithmiques près.

Théorème 1 Les résultats qui suivent restent valables si K est un anneau.

1. L'addition et la soustraction dans $\mathcal{M}_n(K)$ coûtent n^2.
2. La multiplication d'une matrice de $\mathcal{M}_n(K)$ par un scalaire coûte n^2.
3. La multiplication d'une matrice de $\mathcal{M}_n(K)$ par un vecteur de K^n coûte n^2.
4. La multiplication naïve dans $\mathcal{M}_n(K)$ coûte $O(n^3)$.

Exercice 1 Prouver ce théorème et calculer le nombre exact d'additions et multiplications requises pour la multiplication des matrices.

Pour les opérations 1,2,3, on peut montrer que l'on ne peut pas faire mieux. Mais il existe des algorithmes de multiplications plus efficaces.

2 Multiplication de Strassen

L'algorithme de Strassen pour la multiplication des matrices ressemble dans l'esprit à l'algorithme de Karatsuba pour la multiplication des polynôme : c'est une méthode "diviser pour régner" qui repose sur le gain d'une multiplication pour les matrices 2×2, qui devient un gain dans l'exposant de complexité lors d'une application récursive.

Soient $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ et $B = \begin{pmatrix} x & y \\ z & t \end{pmatrix}$ dans $\mathcal{M}_2(K)$. On a donc

$$AB = \begin{pmatrix} ax + bz & ay + bt \\ cx + dy & cz + dt \end{pmatrix}$$

et la multiplication naïve requiert 4 additions et 8 multiplications. On cherche à calculer les entrées de AB en minimisant le nombre de multiplications. Notons

$$q_1 = a(x + z)\,,\ q_2 = d(y + t)\,,\ q_3 = (d - a)(z - y)\,,\ q_4 = (b - d)(z + t),$$

$$q_5 = (b - a)z\,,\ q_6 = (c - a)(x + y)\,,\ q_7 = (c - d)y.$$

On remarque alors que l'on a les égalités suivantes (exo)

— $ax + by = q_1 + q_5$
— $ay + bt = q_2 + q_3 + q_4 - q_5$
— $cx + dy = q_1 + q_3 + q_6 - q_7$
— $cz + dt = q_2 + q_7$

Ainsi, on peut calculer le produit AB avec 18 additions et seulement 7 multiplications. Comme dans l'algorithme de Karatsuba, bien que le nombre d'addition augmente, le fait de faire une multiplication de moins permet de réduire la complexité asymptotique.

Il reste bien sûr à utiliser cette méthode récursivement. On utilise pour cela la multiplication par bloc. Supposons pour simplifier que $n = 2^k$, $k \in \mathbb{N}$. Dans ce cas, on peut toujours écrire A, B et AB comme ci-dessus, mais les entrées a, b, c, d, x, y, z, t sont cette fois des matrices dans $\mathcal{M}_{n/2}(K)$. Le point crucial est que les formules exprimant AB à l'aide des q_i restent valables si l'on remplace K par l'anneau non commutatif $\mathcal{M}_{n/2}(K)$ (auquel cas l'ordre des facteurs dans les calculs est bien sûr fondamental). On obtient ainsi un algorithme de type "diviser pour régner". Sa complexité se calcule selon la même idée que pour l'algo de Karatsuba.

Proposition 1 L'algorithme de Strassen coûte $O(n^{\log_2(7)}) \approx O(n^{2,81})$ opérations dans K.

Preuve. Soit $C(n)$ la complexité de l'algorithme de Strassen. D'après ce que l'on vient de voir, $C(n)$ égal le coût de 18 additions ou soustractions dans $\mathcal{M}_{n/2}(K)$ et 7 multiplications dans $\mathcal{M}_{n/2}(K)$. On a donc la relation de récurrence :

$$C(n) = 7C(n/2) + 18(n/2)^2$$

avec la condition initiale $C(1) = 1$ (une multiplication dans K). En posant $X(k) = C(2^k)/7^k$, on obtient la relation de récurrence

$$X(k) = X(k-1) + \frac{18}{4}\left(\frac{4}{7}\right)^k = X(0) + \frac{18}{4}\sum_{i=1}^{k}\left(\frac{4}{7}\right)^i \leq 1 + \frac{18}{4}\frac{\frac{4}{7}}{1-\frac{4}{7}} = 7.$$

On obtient alors $C(2^k) \leq 7.7^k$, d'où il suit $C(n) \leq 7n^{\log_2(7)}$. □

Remarque 2 Si n n'est pas une puissance de 2, on peut (par exemple) compléter les matrices avec des zéros à droite et en-dessous pour avoir des matrices de taille $N \times N$ où N est la plus petite puissance de 2 supérieure à n. La complexité est au pire multipliée par 2.

L'exposant de multiplication des matrices : dans ce qui suit, on note ω l'infemum des réels tels que l'on puisse multiplier deux matrices de taille n en $O(n^\omega)$ opérations dans K. On a donc $\omega \leq 2,81$ grâce à l'algorithme de Strassen. D'un autre côté, on sait que $\omega \geq 2$. Il est conjecturé que $\omega = 2$, mais prouver (ou réfuter) cette conjecture semble aujourd'hui hors de portée. Ce problème (de nature théorique) est intimement lié à des questions de géométrie algébrique particulièrement ardues (décomposition de tenseurs). On a cependant le résultat suivant :

Théorème 2 Il existe des algorithmes de multiplication rapides des matrices de complexité $O(n^{2.37})$. Autrement dit, $\omega \leq 2.37$ (François Le Gall, 2014).

Ce résultat est théorique. En pratique, l'algorithme de Le Gall est très sophistiqué et très difficile à mettre en oeuvre. L'algorithme de Strassen est celui le plus couramment implémenté.

3 Elimination Gaussienne

L'algorithme bien connu du pivot de Gauss permet de résoudre la plupart des problèmes classiques en algèbre linéaire de manière bien plus efficace que l'algorithmique naïve.

Définition 1 (Forme échelonnée) Une matrice est échelonnée en ligne lorsque :

1. Les lignes nulles sont toutes en dessous des lignes non nulles,
2. Le premier coefficient non nul (appelé pivot) de chaque ligne non nulle est strictement à droite du premier coefficient non nul de la ligne précédente.

Une forme échelonnée en ligne d'une matrice A est une matrice échelonnée en ligne B équivalente à gauche à A (c'est-à-dire telle qu'il existe une matrice carrée inversible P avec $A = PB$).

Matrice échelonnée en ligne (les $\oplus$ représentent les pivots, non nuls) :

$$B = \begin{pmatrix} \oplus & * & * & * & * & * & * & * & * \\ 0 & 0 & \oplus & * & * & * & * & * & * \\ 0 & 0 & 0 & \oplus & * & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & \oplus & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \oplus \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

L'algorithme de Gauss (sans pivot sur les colonnes) calcule une forme échelonnée en ligne de A, en

n'utilisant que des opérations élémentaires sur les lignes (qui correspondent à des multiplications à gauche par des matrices inversibles) :

— Échange de deux lignes (permutation)
— Multiplication d'une ligne par un scalaire non nul (dilatation)
— Ajout du multiple d'une ligne à une autre ligne (transvection)

On peut lire sur la forme échelonnée le rang et le déterminant de A. En effet, les lignes non nulles d'une forme échelonnée B forment une base de l'espace vectoriel engendré par les lignes de la matrice de départ A. Le nombre de lignes non nulles de B est égal au rang de A. Si A est carrée, son déterminant est égal au produit des éléments diagonaux de B.

Elimination de Gauss-Jordan

On va plutôt décrire en détail une variante de l'algorithme de Gauss, dite de Gauss-Jordan, qui produit même une forme échelonnée *réduite* de A : il s'agit d'une forme échelonnée de A, dont les pivots valent 1, tous les autres coefficients dans les colonnes des pivots étant nuls (même ceux du dessus).

Matrice échelonnée en ligne réduite de la matrice B ci-dessus :

$$\tilde{B} = \begin{pmatrix} 1 & * & 0 & 0 & * & * & 0 & * & 0 \\ 0 & 0 & 1 & 0 & * & * & 0 & * & 0 \\ 0 & 0 & 0 & 1 & * & * & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

On note L_i la i-ème ligne de A dans ce qui suit.

```
Gauss-Jordan(A ∈ M_{m,n}(K))
   r = 0                                   (indice de ligne du dernier pivot trouvé)
   Pour j = 1 à n faire :                                  (j indice des colonnes)
        |a_kj| = max(|a_ij|, r+1 ≤ i ≤ m)                      (a_kj nouveau pivot)
        Si a_kj ≠ 0 alors:
           r = r + 1                          (indice de placement du nouveau pivot)
           L_k = L_k / a_kj                                (pivot prend la valeur 1)
           Si k ≠ r alors:
             Permuter L_k et L_r                    (ligne du pivot en position r)
           Pour i = 1 à m faire:                     (simplifie les autres lignes)
             Si i ≠ r alors:
               L_i = L_i − a_ij L_r                                (on annule a_ij)
```

Noter que l'on choisit en général le pivot en considérant le max des valeur $|a_{ij}|$ pour $r+1 \leq i \leq m$ pour des raisons de stabilité numérique, en supposant que l'on travaille en flottant. Sur $\mathbb{Q}$ ou sur un corps fini, on prendra simplement le premier élément non nul.

Supposons que A est carrée et inversible. Appliquée à la matrice augmentée $\tilde{A} = (A|I_n)$ obtenue par concaténation de la matrice A et de la matrice identité I_n, cet algorithme permet le calcul de l'inverse A^{-1}.

Proposition 2 Suppose A carrée et inversible.

1. L'algorithme de Gauss-Jordan transforme la matrice $\tilde{A}$ en une matrice équivalente dont le bloc gauche est l'identité, c'est-à-dire qu'il remplace $\tilde{A}$ par la matrice $\tilde{B} = (I_n|A^{-1})$.
2. Le même algorithme permet de résoudre le système $Ax = b$, où $b \in K^n$, en bordant cette fois la matrice A par le vecteur b.

Preuve. La forme échelonnée réduite d'une matrice carrée inversible est la matrice identité I_n. Toutes les opérations sur A réalisées par l'algorithme de Gauss-Jordan reviennent à multiplier A à gauche par une matrice inversible P (produit de matrices de permutations, de dilatation et de transvection). On a donc $PA = I_n$ et P est l'inverse de A. En appliquant l'algorithme à $\tilde{A}$, on obtient la matrice $\tilde{B} = P\tilde{A} = (PA|P) = (I_n|P)$ Donc le terme de droite de $\tilde{B}$ est l'inverse de A. En appliquant l'algorithme à la matrice augmentée $(A|b)$, on obtient cette fois la matrice (I, Pb), dont le terme de droite $x = Pb = A^{-1}b$ est bien la solution de $Ax = b$. □

Remarque 3 Attention, on ne peut plus calculer le déterminant sur la forme échelonnée réduite car on a au final multiplié A par des matrices de dilatations de déterminants $\neq 1$. Ceci dit, le calcul du déterminant peut se faire au cours de l'algorithme de Gauss-Jordan.

Exercice 2 Utiliser l'algorithme de Gauss-Jordan pour calculer l'inverse de

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

Décomposition PLU

Une autre variante de l'algorithme d'élimination de Gauss permet de calculer une décomposition PLU qui, à son tour, permet de résoudre des systèmes linéaires, d'inverser des matrices, de calculer des déterminants, etc.

Définition 2 (Matrices L, U, P et décomposition PLU) Une matrice est dite triangulaire supérieure, resp. triangulaire inférieure, si les éléments situés en dessous (resp. au dessus) de la diagonale principale sont nuls. Une matrice de permutation est une matrice carrée P inversible dont les coefficients valent 0 ou 1, et dont chaque ligne et colonne ne contient qu'un seul élément non nul. Une décomposition LU, resp. PLU, d'une matrice A est une factorisation de la forme $A = LU$, resp. $A = PLU$, avec L triangulaire inférieure, U triangulaire supérieure et P une matrice de permutation. Noter que certains auteurs considèrent plutôt des décompositions LUP, ce qui revient essentiellement au même.

Exemple de décomposition PLU d'une matrice carrée :

$$\begin{pmatrix} 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 2 & 1 & -1 & -1 \\ 4 & -11 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{4} & \frac{7}{26} & 1 & 0 \\ \frac{1}{4} & \frac{7}{26} & -\frac{11}{15} & 1 \end{pmatrix} \begin{pmatrix} 4 & -11 & 1 & 1 \\ 0 & \frac{13}{2} & -\frac{3}{2} & -\frac{3}{2} \\ 0 & 0 & \frac{15}{13} & -\frac{11}{13} \\ 0 & 0 & 0 & -\frac{22}{15} \end{pmatrix}$$

Pour toute matrice carrée, on a existence d'une décomposition PLU. Pour une matrice inversible, la décomposition LU existe si et seulement si tous les mineurs principaux sont non nuls.

La décomposition LU permet de ramener la résolution $Ax = b$ à la résolution de deux systèmes triangulaires $Ly = b$ et $Ux = y$, qui se résolvent par des techniques classiques de substitution (en remontant pour U et en redescendant pour L). Si l'on veut résoudre le système $Ax = b$ pour divers b, il est plus intéressant de réaliser la décomposition LU une fois pour toutes et d'effectuer les substitutions de descente-remontée pour les différents b plutôt que d'utiliser l'élimination de Gauss-Jordan à de multiples reprises.

Complexité

La complexité de l'élimination Gaussienne et de ses conséquences peut se résumer ainsi :

Théorème 3 Pour toute matrice carrée $A \in \mathcal{M}_n(K)$, il est possible de calculer en $O(n^3)$ opérations dans K :

1. le rang $rg(A)$, le déterminant $\det(A)$, et l'inverse A^{-1} si A est inversible ;
2. une base (affine) de solutions de $Ax = b$, pour tout b dans K^n ;
3. une décomposition PLU, et une forme échelonnée réduite de A.

Exercice 3 Montrer que l'algorithme de Gauss-Jordan appliqué à une matrice carrée A coûte $O(n^3)$ opérations dans K.

4 L'élimination Gaussienne n'est pas optimale

Il se trouve que l'on peut faire mieux que l'élimination Gaussienne. En effet, on a le théorème de complexité suivant :

Théorème 4 Soit ω l'exposant de multiplication des matrices. Pour toute matrice carrée $A \in \mathcal{M}_n(K)$, il est possible de calculer en $\tilde{O}(n^\omega)$ opérations dans K :

1. le rang $rg(A)$, le déterminant $\det(A)$, et l'inverse A^{-1} si A est inversible ;
2. une base (affine) de solutions de $Ax = b$, pour tout b dans K^n ;
3. une décomposition PLU, et une forme échelonnée réduite de A ;
4. une base du noyau de A ;
5. le polynôme caractéristique de A.

La preuve de ce théorème en toute généralité dépasse le cadre de ce cours. On va cependant se pencher sur le problème de l'inversion et montrer qu'elle est équivalente à la multiplication d'un point de vue complexité. Le point 5 sera abordé en fin de chapitre.

La multiplication n'est pas plus difficile que l'inversion.

Soient $A, B \in \mathcal{M}_n(K)$. On souhaite calculer $C = AB$. Pour cela, on pose

$$D = \begin{pmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{pmatrix}$$

On a alors l'identité remarquable

$$D^{-1} = \begin{pmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{pmatrix}$$

qui permet donc de ramener le produit AB en taille n à l'inversion de D en taille $3n$.

Inversion rapide : diviser pour régner (encore).

Nous allons montrer qu'il est possible d'inverser des matrices $n \times n$ par un algorithme de type « diviser pour régner » en $O(n^\omega)$ opérations arithmétiques.

L'algorithme présenté ici, dû à Strassen, est un algorithme récursif, qui peut s'interpréter comme un « pivot de Gauss par blocs ». Cet algorithme nécessite d'inverser certaines sous-matrices de A ; afin de garantir sa correction, nous allons faire l'hypothèse simplificatrice que toutes ces matrices sont inversibles. Le traitement du cas général (A matrice arbitraire) est plus délicat, et passe par le calcul efficace, toujours de type « diviser pour régner », d'une décomposition LUP de A.

Le point de départ est l'identité suivante (non commutative !), dans laquelle on suppose $n = 2$ et $a \neq 0$:

$$A := \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ ca^{-1} & 1 \end{pmatrix} \times \begin{pmatrix} a & 0 \\ 0 & Z \end{pmatrix} \times \begin{pmatrix} 1 & a^{-1}b \\ 0 & 1 \end{pmatrix}, \quad (1)$$

où $Z = d - cba^{-1} \in K$ est le *complément de Schur* de a dans A (note que $Z = a^{-1}\det(A)$ est non nul par hypothèse sur A). On en déduit

$$A^{-1} = \begin{pmatrix} 1 & -a^{-1}b \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} a^{-1} & 0 \\ 0 & Z^{-1} \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ ca^{-1} & 1 \end{pmatrix} = \begin{pmatrix} a^{-1} + a^{-1}bZ^{-1}ca^{-1} & -a^{-1}bZ^{-1} \\ -Z^{-1}ca^{-1} & Z^{-1} \end{pmatrix}.$$

Le point clé est que ces égalités restent valables dans un anneau non commutatif (on fait donc bien attention de ne pas simplifier les produits) dès lors que a et Z sont inversibles. Ceci permet d'appeler récursivement cette stratégie pour $n > 2$ en faisant une "division par blocs".

On obtient l'algorithme suivant, dans lequel on suppose $A \in \mathcal{M}_n(K)$ inversible et telle que toutes les matrices à inverser sont inversibles, avec $n = 2^k$. :

`InverseRapide(`A`)`

1. `Si` $n = 1$, `retourner` A^{-1}.
2. `Décomposer` $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ `avec` $a, b, c, d \in \mathcal{M}_{n/2}(K)$.
3. `Calculer` $e = a^{-1}$ `récursivement.`
4. `Calculer` $Z = d - ceb$.
5. `Calculer` $t = Z^{-1}$ `récursivement.`
6. `Calculer` $y = -ebt$, $z = -tce$, $x = e - ebz$.
7. `Retourner` $\begin{pmatrix} x & y \\ z & t \end{pmatrix}$.

Théorème 5 Soit A inversibles. Si toutes les sous-matrices a et Z rencontrées sont inversibles, l'algorithme `InverseRapide` calcule l'inverse de A avec $O(n^\omega)$ opérations dans K.

Preuve. D'après ce qui précède, l'algorithme est correct dès lors que les matrices a et Z sont inversibles à chaque étape. Soit $C(n)$ sa complexité. On note $T(n) = O(n^\omega)$, le coût pour multiplier deux matrices de taille n. On a $C(1) = 1$ (une inversion dans K). Suppose $n > 1$. L'étape 3 coûte $C(n/2)$, l'étape 4 coûte $2T(n/2)$ pour les multiplications et $(n/2)^2$ pour l'addition, l'étape 5 coûte $C(n/2)$ et puisque eb a déjà été calculé à l'étape 4, l'étape 6 coûte $4T(n/2) + (n/2)^2$. Ainsi,

$$C(n) = 2C(n/2) + 6T(n/2) + n^2/2.$$

En notant $X(k) = C(2^k)/2^k$, on en déduit (la seconde égalité par récurrence)

$$X(k) = X(k-1) + 3T(2^{k-1})/2^{k-1} + 2^{k-1} = 1 + 3\sum_{i=0}^{k-1} T(2^i)/2^i + \sum_{i=0}^{k-1} 2^i.$$

Or il existe par hypothèse une constante $c > 0$ telle que $T(n) \leq cn^\omega$ pour tout $n \in \mathbb{N}$. Donc $T(2^i)/2^i \leq c(2^{\omega-1})^i$ pour tout $i \in \mathbb{N}$. En majorant les sommes partielles des deux séries géométriques impliquées, on en déduit

$$X(k) \leq 1 + 3c(2^{\omega-1})^k + 2^k = 1 + cn^{\omega-1} + n$$

d'où il suit $C(n) \leq n + 3cn^\omega + n^2 = O(n^\omega)$. □

Résolution rapide de systèmes linéaires.

Une première conséquence immédiate est que sous les hypothèses du théorème 5, la résolution du système linéaire $Ax = b$, pour $b \in K^n$ peut s'effectuer également en $O(n^\omega)$ opérations dans K. En effet, une fois A^{-1} calculée, on calcule $x = A^{-1}b$ avec $n^2 \in O(n^\omega)$ opérations dans K.

Calcul rapide du déterminant.

Une seconde conséquence est que sous les hypothèses du théorème 5, on peut également calculer $\det(A)$ en $O(n^\omega)$ opérations dans K. En effet, la décomposition (1) de A donne immédiatement l'égalité

$$\det(A) = \det(a)\det(Z).$$

Il suffit donc de calculer récursivement le déterminant de a et Z aux étapes 3 et 5. Au final, il suffit de remplacer les étapes $1, 3, 5, 7$ par

`1bis.` `Si` $n = 1$, `retourner` A^{-1} `et` A.
`3bis.` `Calculer` $e = a^{-1}$ `et` $\det(a)$ `récursivement.`
`5bis.` `Calculer` $t = Z^{-1}$ `et` $\det(Z)$ `récursivement.`
`7bis.` `Retourner` $\begin{pmatrix} x & y \\ z & t \end{pmatrix}$ `et` $\det(a)\det(Z)$.

Affaiblissement des hypothèses.

Dans le cas où K est un sous-corps de $\mathbb{R}$, on peut s'affranchir des hypothèses du théorème 5 en observant que, quelle que soit la matrice A inversible dans $\mathcal{M}_n(\mathbb{R})$, la matrice $B = A^t \cdot A$ est définie positive et vérifie les hypothèses du théorème 5. Donc l'inverse de A peut se calculer grâce à l'égalité $A^{-1} = B^{-1} \cdot A^t$ en $O(n^\omega)$ opérations dans K.

Dans le cas d'un corps quelconque, il existe un algorithme général d'inversion dû à Bunch et Hopcroft sans hypothèses sur A. Cet algorithme calcule d'abord une décomposition LUP d'une matrice inversible arbitraire sur un corps quelconque K en $O(n^\omega)$ opérations, et arrive à en déduire le calcul d'inverse (et aussi la résolution de système) pour le même prix.

Résolution des sytèmes surdéterminés.

Lorsqu'une matrice n'est pas carrée, le calcul de son noyau se ramène aisément au cas carré, comme illustré dans le résultat suivant :

Théorème 6 Le calcul d'une base du noyau d'une matrice $A \in \mathcal{M}_{m,n}(K)$ avec $m \geq n$ coûte $\tilde{O}(mn^{\omega-1})$ opérations dans K.

Preuve. Le théorème 4 implique ce lemme si $m = n$. Donc, quitte à ajouter des lignes nulles à la matrice A, on peut supposer que m est un multiple de n. Notons A_1 la matrice composée des n premières lignes de A, A_2 celle des n suivantes, etc. On peut calculer une base du noyau de A_1 (matrice carrée de taille n) avec un coût $\tilde{O}(n^\omega)$. Notons N_1 la matrice dont les colonnes sont les vecteurs de cette base, de sorte que $A_1 N_1 = 0$. On calcule ensuite la matrice N_2 dont les colonnes forment une base du noyau de $A_2 N_1$ (à nouveau en $\tilde{O}(n^\omega)$), de sorte que $A_2 N_1 N_2 = 0$. On continue de la même manière avec A_3, en calculant N_3 telle que $A_3 N_1 N_2 N_3 = 0$, etc. Finalement, les colonnes de $N_1 \cdots N_{m/n}$ forment une base du noyau de A. On obtient ainsi le coût total annoncé $\tilde{O}(mn^{\omega-1})$. □

Exercice 4 Montrer que les colonnes de $N_1 \cdots N_{m/n}$ forment bien une base du noyau de A.

Correction. *Notons E l'espace vectoriel K^n munit d'une base $\mathcal{B}$ et $f_i : E \to E$ l'endomorphisme de matrice A_i dans la base $\mathcal{B}$. Soit $E_i = Ker(f_1) \cap \cdots \cap Ker(f_i) \subset E$. On a donc $E_{m/n} = Ker(A)$.*

Il suffit donc de montrer que les colonnes de $N_1 \cdots N_i$ forment une base $\mathcal{B}_i$ de E_i pour tout i. Par récurrence sur i. Si $i = 1$, c'est vrai par définition de N_1. Soit $i > 1$ et supposons que les colonnes de $N_1 \cdots N_{i-1}$ forment une base $\mathcal{B}_{i-1}$ de E_{i-1}. De manière équivalente, $N_1 \cdots N_{i-1}$ est la matrice de l'injection canonique $\phi : E_{i-1} \hookrightarrow E$ dans les bases $\mathcal{B}_{i-1}$ et $\mathcal{B}$. La matrice $A_i N_1 \cdots N_{i-1}$ est donc la matrice de $f_i \circ \phi : E_{i-1} \to E$ dans les bases $\mathcal{B}_{i-1}$ et $\mathcal{B}$. On a donc $Ker(A_i N_1 \cdots N_{i-1}) = Ker(f_i \circ \phi) = Ker(f_{i|E_{i-1}}) = E_i$. Ainsi, les colonnes de N_i forment par définition une base du s.e.v $E_i \subset E_{i-1}$ (dont les coordonnées sont exprimées dans la base $\mathcal{B}_{i-1}$). Il s'ensuit par composition avec ϕ que les colonnes de $N_1 \cdots N_i$ forment une base $\mathcal{B}_i$ du s.e.v $E_i \subset E$ (dont les coordonnées sont cette fois exprimées dans la base $\mathcal{B}$ de E). $\square$.

Calcul du polynôme caractéristique.

Soit $A \in \mathcal{M}_n(K)$. Dans cette section, nous présentons un algorithme efficace, dû à Keller-Gehrig, pour le calcul du polynôme caractéristique

$$\chi_A(X) = \det(XI_n - A)$$

de A. Nous ne décrirons en détail qu'un cas particulier plus simple (mais fréquent). Plus exactement, nous supposerons dans la suite que χ_A est irréductible dans $K[X]$ ce qui le cas le plus fréquent (si $K = \mathbb{R}$, c'est le cas pour un sous-ensemble dense de $\mathcal{M}_n(\mathbb{R})$ car c'est une propriété "ouverte") ; en particulier, χ_A est supposé ici coïncider avec le polynôme minimal de A. L'idée principale de l'algorithme est de lire le polynôme caractéristique de A sur une matrice compagnon semblable à A.

Matrice de type compagnon. Rappelons qu'une matrice $C = (c_{ij})_{1 \leq i,j \leq n}$ est appelée *de type compagnon* si ses $n-1$ premières colonnes ne contiennent que des éléments nuls, à l'exception des éléments $c_{2,1}, c_{3,2}, \ldots, c_{n,n-1}$ qui valent tous 1. Le polynôme caractéristique d'une matrice de type compagnon C se "lit" gratuitement sur la dernière colonne :

$$\chi_C(X) = -c_{1n} - c_{2n}X - \cdots - c_{n,n}X^{n-1} + X^n.$$

L'algorithme repose de façon cruciale sur le lemme suivant.

Proposition 3 Soit $v \in K^n \setminus \{0\}$ un vecteur colonne et soit $P \in \mathcal{M}_n(K)$ la matrice dont les colonnes sont les vecteurs $v, Av, A^2v, \ldots, A^{n-1}v$. Alors P est inversible et la matrice $C = P^{-1}AP$ est une matrice compagnon.

Preuve. La matrice P est inversible car la famille $\mathcal{B} = (v, Av, A^2v, \ldots, A^{n-1}v)$ est une base de K^n : en effet, sinon il existe un polynôme $Q \in K[X]$ non nul de degré $< n$ tel que $Q(A)v = 0$. L'ensemble des tels polynômes est un idéal de $K[X]$, donc principal. On peut supposer sans perte de généralité que Q est le générateur de cet idéal. Or χ_A appartient à cet idéal par le théorème de Cayley-Hamilton. Donc Q divise χ_A. Or χ_A est irréductible de degré n par hypothèse, donc Q est constant. Ceci implique $v = 0$, contradiction. En notant $v_i = A^{i-1}v$, on remarque que $Av_i = v_{i+1}$. Ainsi l'endomorphisme $w \mapsto Aw$ a une matrice de type compagnon dans la base $\mathcal{B}$. Cette matrice est $C = P^{-1}AP$ par le théorème de changement de base. $\square$

Puisque A et $C = P^{-1}AP$ sont des matrices semblables, elles ont même polynôme caractéristique. Il suffit donc de calculer C pour calculer χ_A, ce qui se ramène à :

1. Calculer P, c'est à dire les vecteurs $v, Av, \ldots, A^{n-1}v$ pour un $v \in K^n$ non nul quelconque.
2. Inverser P (coût $O(n^\omega)$)
3. Calculer le produit $C = P^{-1}AP$ (coût $O(n^\omega)$).

La seule étape coûteuse est la construction de la matrice P. En effet, la méthode directe qui consiste à calculer la suite (dite de Krylov) $v, Av, \cdots A^{n-1}v$ par multiplications successives d'un vecteur par la matrice A a un coût $O(n^3)$ [1]. La remarque cruciale est que l'on peut regrouper les produits matrice-vecteur en plusieurs produits matrice-matrice, de la façon suivante : on calcule en $O(n^2)$ les vecteurs v et Av, on détermine ensuite par exponentiation binaire les $O(\log n)$ matrices $A, A^2, A^4, A^8, \ldots$ pour un coût de $O(n^\omega \log n)$. On termine le calcul des colonnes de P par le calcul des produits

$$(A^2v \,|\, A^3v) = A^2 \times (v \,|\, Av), \qquad (A^4v \,|\, A^5v \,|\, A^6v \,|\, A^7v) = A_4 \times (v \,|\, Av \,|\, A^2v \,|\, A^3v), \qquad \cdots$$

etc. Chaque produit de ce type est effectué à l'aide d'un produit matriciel en taille $n \times n$ de coût $O(n^\omega)$, en rajoutant artificiellement des colonnes nulles aux facteurs droits. Cette méthode conduit à un algorithme de complexité totale $O(n^\omega \log n)$. Ceci prouve le point 5 du théorème 4 dès lors que χ_A est irréductible.

Théorème 7 Soit $A \in \mathcal{M}_n(K)$. Si le polynôme caractéristique de A est irréductible, on peut le calculer avec $O(n^\omega \log n)$ opérations dans K.

1. On peut penser que $v = (1, 0, \ldots, 0)$ accélère les calculs, mais les vecteurs suivants $A^i v$ seront denses ensuite