Complexité

Algèbre linéaire I : on compte les opérations

Exercice 1 : Soit $A \in \mathrm{GL}(n, \mathbf{k})$. On résoud $Ax = b$ en utilisant la méthode de Gauss.
Compter le nombre d'additions/soustractions et le nombre de multiplications/divisions faites sur chaque ligne lors de la k^e étape, pour l'ensemble de la k^e étape, pour l'ensemble de la résolution (on ne compte pas les éventuels échanges de lignes).
Montrer que les nombres d'opérations pour l'ensemble de la résolution sont des polynômes de degré 3 en n qu'on calculera.

Correction 1 : À la k^e étape on a déjà fait apparaître des zéros sous la diagonale dans les $k-1$ premières colonnes. Pour chaque j^e ligne $(k+1 \leq j \leq n)$ on calcule le coefficient multiplicateur $c_j = \frac{a_{jk}}{a_{kk}}$ (1 division) puis on fait les différences $a_{ji} - c_j a_{ki}$ $(k+1 \leq i \leq n)$ et $b_j - c_j b_k$ ($n-k+1$ multiplications et soustractions).
On a donc pour l'ensemble de la k^e étape $(n-k)(n-k+1)$ additions/soustractions et $(n-k)(n-k+2)$ multiplications/divisions et pour l'ensemble de la résolution $\sum_{k=1}^{k=n-1}(n-k)(n-k+1)$ additions/soustractions et $\sum_{k=1}^{k=n-1}(n-k)(n-k+2)$ multiplications/divisions. On pose $l = n-k$.
On doit calculer $\sum_{l=1}^{l=n-1} l(l+1)$ et $\sum_{l=1}^{l=n-1} l(l+2)$.
Trouver des coefficients a, b, c, d tels que $\sum_{l=1}^{l=n-1} l(l+1) = an^3 + bn^2 + cn + d$ revient à résoudre le système :

$$\begin{aligned} an^3 + bn^2 + cn + d - (a(n-1)^3 + b(n-1)^2 + c(n-1) + d) &= (n-1)n \\ a2^3 + b2^2 + c2 + d &= 1 \times 2 \end{aligned}$$

En développant et en identifiant les coefficients des puissances de n on obtient un système triangulaire dont l'unique solution donne $\frac{n^3}{3} - \frac{n}{3}$ additions/soustractions.
Faisant de même pour la deuxième somme on obtient $\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}$ multiplications/divisions.

Exercice 2 : Reprendre l'exercice précédent pour le calcul de la décomposition LU de A (on suppose qu'il n'y a pas d'échanges de lignes, cf. td méthodes directes de résolution des systèmes linéaires).
Application : Quel est le coût du calcul de l'inverse de A par la méthode de Gauss en résolvant un seul système ? par la méthode de Gauss en résolvant n systèmes ? par la décomposition LU de A ?

Correction 2 : Rappel : On peut représenter la k^e étape de la méthode de Gauss comme la multiplication par la matrice : $L_k = Id - L'_k$ où

$$L'_k = \begin{pmatrix} 0 & & & & \\ & \ddots & & 0 & \\ & & 0 & & \\ & & \frac{a_{k+1k}}{a_{kk}} & & \\ & 0 & \vdots & \ddots & \\ & & \frac{a_{nk}}{a_{kk}} & & 0 \end{pmatrix}$$

Donc $A = (L_1^{-1} \dots L_k^{-1})(L_k \dots L_1 A)$, on remarque que $L_k^{-1} = Id + L'_k$ car $(L'_k)^2 = (0)$ et que $L_1^{-1} \dots L_k^{-1} = Id + L_1'^{-1} + \dots + L_k'^{-1}$ car $L_i'^{-1} L_j'^{-1} = (0)$ si $j > i$. Le calcul de la matrice L ne demande donc aucun calcul supplémentaire mais simplement un stockage.
La méthode de Gauss sans second membre demande $\sum_{k=1}^{k=n-1}(n-k)^2 = \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6}$ additions/soustractions et $\sum_{k=1}^{k=n-1}(n-k)(n-k+1) = \frac{n^3}{3} - \frac{n}{3}$ multiplications/divisions.
La résolution de $Ly = b$ demande $\frac{n(n-1)}{2}$ additions/soustractions et $\frac{n(n-1)}{2}$ multiplications/divisions.
La résolution de $Ux = y$ demande $\frac{n(n-1)}{2}$ additions/soustractions et $\frac{n(n+1)}{2}$ multiplications/divisions.
Application : si l'on calcule A^{-1} en posant le système correspondant à $AA^{-1} = Id$ et en le résolvant par la méthode de Gauss on a n^2 équations et n^2 inconnues, donc le nombre d'additions/soustractions et le nombre de multiplications/divisions équivalent à $\frac{n^6}{3}$.
Si l'on constate qu'on a n systèmes de n équations à n inconnues (un système pour chaque colonne de A^{-1}), le nombre d'additions/soustractions et le nombre de multiplications/divisions équivalent à $\frac{n^4}{3}$.
Si l'on s'aperçoit que dans les n systèmes seul le second membre change et qu'on utilise la décomposition LU de A le nombre d'additions/soustractions et le nombre de multiplications/divisions équivalent à $\frac{n^3}{3} + nn^2 = \frac{4n^3}{3}$.

Algèbre linéaire II : la taille des opérandes

Exercice 3 : On suppose maintenant $A \in \mathcal{M}_{n,n}(\mathbf{Z})$. Comment adapter la méthode de Gauss pour ne pas introduire de nombres rationnels ? Pourquoi ne veut-on pas introduire de nombres rationnels ?
Si on suppose que les entiers de A ont au plus l chiffres quelle est la taille (maximale) des entiers calculés lors de la k^e étape ? (on verra en tp qu'on peut diminuer considérablement cette taille)
Les opérations sur des entiers de grande taille demandent plusieurs opérations de la machine. En utilisant les algorithmes "naïfs" on peut considérer qu'une addition/soustraction d'entiers à l chiffres demandent $c_1 l$ additions/soustractions de la machine et qu'une multiplication/division d'entiers à l chiffres demandent $c_2 l^2$ multiplications/divisions de la machine (c_1 et c_2 sont des constantes). Pourquoi (cf. Demazure) ?
Reprendre l'exercice 1 en tenant compte de n et l.

Correction 3 : À la k^e étape pour chaque j^e ligne au lieu de calculer $a_{ji} - \frac{a_{jk}}{a_{kk}} a_{ki}$ qui introduit des fractions, on calcule $a_{kk}a_{ji} - a_{jk}a_{ki}$. Les additions/soustractions dans $\mathbf{Q}$ sont couteuses (plusieurs opérations et un pgcd).
À chaque étape on double le nombre de chiffres donc à la k^e étape les entiers auront $2^k l$ chiffres.
C'est l'addition et la multiplication par tranches dans une base donnée, c_1 et c_2 dépendent de la base utilisée.
On obtient donc $c_1 l \sum_{k=1}^{k=n-1} 2^k (n-k)(n-k+1)$ additions/soustractions et $c_2 l^2 \sum_{k=1}^{k=n-1} 2^{2k}(n-k)(n-k+2)$ multiplications/divisions (on calculera ces sommes en tp).

Le pgcd

Exercice 4 : L'algorithme d'Euclide peut ne prendre qu'un pas même avec des entiers grands (ex : n et $n-1$). À l'inverse quelle doit être la relation entre deux restes consécutifs pour que le nombre de pas soit le plus grand possible ? En déduire que si deux entiers $x > y > 0$ ont pour pgcd d et que l'algorithme d'Euclide prend n pas alors $x \geq dF_{n+2}$ et $y \geq dF_{n+1}$ où (F_i) est la suite de Fibonacci. Inversement, majorer le nombre de pas en fonction de y. Qu'en est-il si on tient compte de la taille des opérandes ? (cf. Demazure ou Childs et Knuth vol. 2 pour une discussion du coût en moyenne)

Correction 4 : Il faut que le quotient vaille 1 à chaque étape. Si $n = 1$ alors $y = d$ et $x \geq 2d$. Par récurrence, si $x = qy + z$ alors $y \geq dF_{n+1}$ et $z \geq dF_n$ et donc $x \geq y + z \geq dF_{n+2}$.
On a $F_{n+1} \geq 2^{\frac{2}{3}(n-1)}$ (par récurrence).
Si l'algorithme fait n pas on a $y \geq F_{n+1}$ et donc $n \leq \frac{3}{2}\log(F_{n+1}) + 1 \leq \frac{3}{2}\log y + 1$.
Si l'on tient compte de la taille des opérandes on $n \leq c(\log x)^2$

Un classique : les tris

Exercice 5 : On veut ranger une suite $(a_1, \ldots, a_n)$ en ordre croissant. On utilise l'algorithme suivant (tri par bulles) : on parcourt la suite en échangeant a_i et a_{i+1} s'ils ne sont pas ordonnés, puis on recommence sur la suite modifiée $(a_1, \ldots, a_{n-1})$. Pourquoi ? Combien de comparaisons fait-on ?
On retient le premier i où l'on a fait un échange et on ne recommence qu'à partir de là. Pourquoi ? Combien de comparaisons fait-on au mieux, au pire ?
Quel sens donner à l'affirmation "le tri par fusion (quicksort) est optimal" ? (cf Knuth vol. 3)

Correction 5 : Après le premier passage le plus grand élément a été mis en a_n. On fait donc $\frac{n(n-1)}{2}$ comparaisons.
Les a_j pour $j < i$ sont ordonnés, il suffit donc de recommencer en comparant a_{i-1} et a_i. Si la suite est déjà ordonnée on fait $n-1$ comparaisons, si elle est en ordre inverse on revient à $\frac{n(n-1)}{2}$ comparaisons.
Il y a $n!$ permutations de la suite $(a_1, \ldots, a_n)$. Un algorithme est optimal si à chaque test il partage cet ensemble en deux sous-ensembles de tailles équivalentes et décide dans quel sous-ensemble se trouve la suite. Il fait donc $\log_2 n!$ tests, soit un $O(nlogn)$ en utilisant la formule de Stirling ($n! \sim \sqrt{2\pi n}(\frac{n}{e})^n$). C'est le cas du tri par fusion.

Références

Ciarlet, *Introduction à l'analyse numérique matricielle et à l'optimisation*, Dunod
Childs, *A concrete introduction to higher algebra*, Springer
Demazure, *Cours d'algèbre*, Cassini
Knuth, *The art of computer programming, vol. 2 semi-numerical algorithms*, Addison-Wesley
Knuth, *The art of computer programming, vol. 3 sorting and searching*, Addison-Wesley
Serre, *Les matrices*, Dunod