TP3 – Décomposition LUP

Projet de programmation M1

14 Octobre 2014

Afin d'effectuer rapidement certaines opérations sur les matrices, il est souvent intéressant d'avoir une représentation factorisée de la matrice, où chaque facteur a une propriété particulière qui peut être exploitée efficacement. Un exemple notoire est la décomposition LUP : le but est de trouver, étant donné une matrice carré A, une matrice de permutation P telle que PA s'écrive sous la forme $L \times U$ où L est une matrice triangulaire inférieure avec des 1 sur la diagonale, U une matrice triangulaire supérieure. On se propose dans ce TP d'implémenter en C une méthode qui étant donné une matrice A inversible, calcule une décomposition LUP de A. Nous explorerons aussi quelques applications de cette décomposition : la résolution de systèmes linéaires, l'inversion de matrices et le calcul de déterminant.

On va commencer par montrer que toute matrice inversible admet une décomposition LUP, ce qui nous donnera une idée générale de l'algorithme à utiliser. Nous détaillerons ensuite son implémentation.

1 Une démonstration par récurrence

Soit A une matrice $n \times n$ inversible.

1. Justifiez qu'il existe une matrice de permutation Q telle que :

$$QA = \begin{pmatrix} a & w \\ v & A' \end{pmatrix}$$

avec $a \neq 0$, v une matrice colonne de taille $n-1$, w une matrice ligne de taille $n-1$ et A' une matrice $(n-1) \times (n-1)$.

2. En remarquant que

$$QA = \begin{pmatrix} 1 & 0 \\ v/a & I_{n-1} \end{pmatrix} \times \begin{pmatrix} a & w \\ 0 & A' - vw/a \end{pmatrix}$$

justifiez qu'on peut trouver par récurrence une matrice de permutation P', une matrice triangulaire inférieure avec des 1 sur la diagonale L' et une matrice triangulaire supérieure U' telles que :

$$P'(A' - vw/a) = L'U'$$

3. Vérifiez que

$$\begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} QA = \begin{pmatrix} 1 & 0 \\ P'v/a & L' \end{pmatrix} \times \begin{pmatrix} a & w \\ 0 & U' \end{pmatrix}$$

et conclure.

2 Implémentation

On pourrait utiliser la démonstration précédente pour implémenter directement un algorithme récursif pour trouver une décomposition LUP. Cependant, suivre naïvement la récurrence de la question précédente, donne un algorithme inefficace, puisqu'il faut copier les matrices pour l'appel récursif puis les réécrire pour calculer L, U ou P. On va montrer ici comment on peut implémenter cet algorithme en utilisant les opérations du pivot de Gauss.

Pour $k \geq 0$, on construit U_k, L_k deux matrices $n \times n$ et une permutation π_k de $[n]$ telles que $P_{\pi_k} A = L_k U_k$ où $P_{\pi_k}[i, j] = 1$ si et seulement si $j = \pi_k(i)$.

On suppose que L_k est de la forme suivante :

$$L_k = \begin{pmatrix} L_k^1 & 0 \\ L_k^2 & I_{n-k} \end{pmatrix}$$

où L_k^1 est une matrice $k \times k$ triangulaire inférieure avec des 1 sur la diagonale et U_k est de la forme :

$$U_k = \begin{pmatrix} U_k^1 & U_k^2 \\ 0 & C_k \end{pmatrix}$$

où U_k^1 est une matrice $k \times k$ triangulaire supérieure et C_k une matrice $k \times k$. Notre but est d'adapter un peu la démonstration de la partie précédente pour calculer L_{k+1} et U_{k+1} de façon à ce qu'on ait désormais $k+1$ colonnes ou lignes qui soient de la bonne forme.

Comme précédemment, C_k est inversible, donc on peut trouver une matrice de permutation Q telle que

$$QC_k = \begin{pmatrix} a & w \\ v & C' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/a & I_{n-k-1} \end{pmatrix} \times \begin{pmatrix} a & w \\ 0 & C' - vw/a \end{pmatrix} = D_1 \times D_2$$

1. Montrer que
$$(\begin{pmatrix} 1 & 0 \\ 0 & Q \end{pmatrix} P_{\pi_k})A = \begin{pmatrix} L_k^1 & 0 \\ QL_k^2 & D_1 \end{pmatrix} \times \begin{pmatrix} U_k^1 & U_k^2 \\ 0 & D_2 \end{pmatrix}$$
On définit désormais $L_{k+1} = \begin{pmatrix} L_k^1 & 0 \\ QL_k^2 & D_1 \end{pmatrix}$, $U_{k+1} = \begin{pmatrix} U_k^1 & U_k^2 \\ 0 & D_2 \end{pmatrix}$ et π_{k+1} comme étant la permutation représentée par la matrice $\begin{pmatrix} 1 & 0 \\ 0 & Q \end{pmatrix} P_{\pi_k}$
2. On commence avec $L_0 = I_n$ et $U_0 = A$. Montrer alors que si on applique itérativement la définition ci-dessus, on trouve $L_n U_n = P_{\pi_n} A$.
3. Écrire une fonction
```
decLUP(int n, float A[n][n], float L[n][n], float U[n][n], int p[n])
```
qui calcule la décomposition LUP de A et la stocke dans L, U et p (on représentera la matrice de la permutation π par un tableau d'entier tel que $p[i] = \pi(i)$).
4. (bonus, à faire tout à la fin) On peut remarquer que l'on peut stocker toute l'information nécessaire pour L_k et U_k dans une seule matrice $n \times n$. Modifier votre algorithme pour écrire une fonction
```
decLUP(int n, float A[n][n], int p[n])
```
qui met A sous la forme
$$\begin{pmatrix} & U \\ L & \end{pmatrix}$$
sans utiliser de mémoire supplémentaire (on dit que le programme travaille *en place*).

3 Applications

1. Écrire une fonction qui, étant donnée une matrice triangulaire supérieure A et un vecteur b, résoud le système $Ax = b$. Combien d'opérations arithmétiques sont-elles nécessaires ?
2. De même si A est triangulaire inférieure.
3. En déduire une fonction qui étant donné une décomposition LUP d'une matrice A, résoud le système $AX = b$. Sans compter le calcul de la décomposition LUP, combien d'opérations sont nécessaires pour résoudre ce système ? Comparez avec la méthode habituelle du pivot de Gauss.
4. En déduire une méthode d'inversion de matrice. Implémentez-la.