Factorisation LU

AnaNum – Chapitre 2

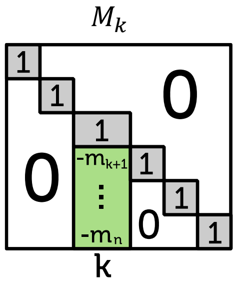
I. Principe

$$Ax = b \quad \text{et} \quad \underset{\substack{L\,:\,\text{tri. inf. unit.}\\ U\,:\,\text{tri. sup.}}}{A = LU} \quad \Rightarrow \quad L\,\underbrace{Ux}_{c} = b \quad \underline{Lc = b} \quad \underline{Lx = c}$$

But : plus rapide

II. Transformation de Gauss

1. Maths



$$\underbrace{M_{n-1} \dots M_1}_{M} A = U \quad \Leftrightarrow \quad A = \underbrace{L_1 \dots L_{n-1}}_{M^{-1}=L} U = LU$$

$$\tau_k^T = \left(\underbrace{0, \dots, 0}_{k \text{ fois}}, \frac{a_{k+1,k}}{a_{kk}}, \dots, \frac{a_{nk}}{a_{kk}}\right) \qquad M_k = I - \tau_k e_k^T \qquad L_k = I + \tau_k e_k^T$$

2. Optimisations

a. Calcul de L

$$L_k L_{k'} = I + \tau_k e_k^T + \tau_{k'} e_{k'}^T \qquad k < k' \qquad \Rightarrow$$

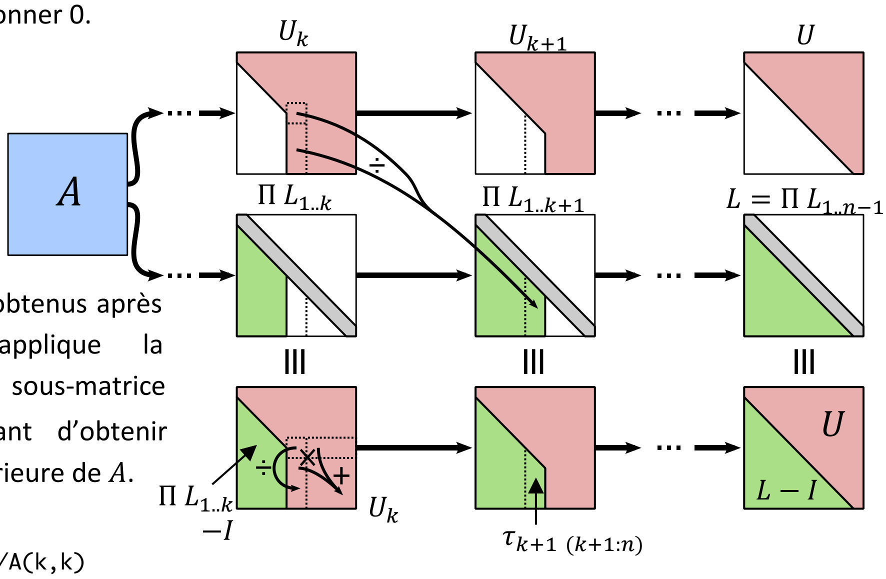
On a juste à recopier les colonnes.

b. Calcul de la transformation de Gauss (U)

$$M_k A = A - \tau_k A_{k,:} \quad \Rightarrow \quad (M_k A)_{(k+1:n,k+1:n)} = A_{(k+1:n,k+1:n)} - \tau_k(k+1:n)A(k, k+1:n)$$

Pas besoin de calculer M_k. Et on peut faire le calcul que sur la sous matrice $A(k+1:n, k+1:n)$ car les colonnes précédentes vont donner 0.

3. Algorithme



Pour chaque k de 1 à $n-1$, on va modifier A afin de stocker dans la partie inférieure, sur la colonne k les valeurs de L donc de τ_k (au lieu d'y stocker les 0 obtenus après triangularisation), et on applique la transformation de Gauss à la sous-matrice $A(k+1:n, k+1:n)$ permettant d'obtenir également U dans la partie supérieure de A.

```
pour k = 1 à n-1 faire
        A(k+1:n,k) = A(k+1:n,k)/A(k,k)
        A(k+1:n,k+1:n) = A(k+1:n,k+1:n) - A(k+1:n,k) × A(k,k+1:n)
finpour
```

(avec $\underbrace{A(k+1:n,k)}_{\tau_{k+1}(k+1:n)}$ dans les deux lignes)

```
pour k = 1 à n-1 faire
        pour i = k+1 à n faire
                A(i,k) = A(i,k)/A(k,k)      // τ_{k+1}(i)
        finpour
        pour i = k+1 à n faire
                pour j = k+1 à n faire
                        A(i,j) = A(i,j) - A(i,k) × A(k,j)     // U_k = M_k U_{k-1}
                finpour
        finpour
finpour
```

4. Condition d'existence

Si à l'étape $a_{kk} = 0$, on ne pas faire la factorisation LU.

La factorisation LU existe ⇔ $\forall\, k \in [\![1; n-1]\!], \det\big(A(1\!:\!l, 1\!:\!k)\big) \neq 0$

III. Factorisation $PA = LU$

1. Principe

On constate que plus le pivot (a_{kk}) est grand, plus l'algorithme est stable numériquement. A chaque étape, on permute donc la ligne k avec la ligne restante ayant le plus grand pivot possible.

$$Ax = b \quad \text{et} \quad \underset{\substack{L\text{ : tri. inf. unit.}\\ U\text{ : tri. sup.}}}{PA = LU} \quad \Rightarrow \quad L\,\underbrace{Ux}_{c} = Pb \qquad \underline{Lc = Pb} \qquad \underline{Lx = c}$$

2. Maths

$$M_{n-1}P^{(n-1)}M_{n-2}P^{(n-2)} \dots M_k P^{(k)} \dots M_1 P^{(1)} A = U$$

$$\Leftrightarrow \quad \underbrace{\widetilde{M}_{n-1} \dots \widetilde{M}_1}_{\widetilde{M}}\ \underbrace{P^{(n-1)} \dots P^{(1)}}_{P} A = U \quad \Leftrightarrow \quad PA = \underbrace{\tilde{L}_1 \dots \tilde{L}_{n-1}}_{\tilde{L} = \widetilde{M}^{-1}} U$$

$$\widetilde{M}_k = P^{(n-1)} \dots P^{(k+1)} M_k P^{(k+1)} \dots P^{(n-1)} \qquad\qquad P_{ij} M_k P_{ij} = I - \tau_k^{(ij)} e_k^T$$

3. Algorithme

```
pour k = 1 à n-1 faire
        i = max |A(i,k)|
        p(k) = i
        A(k,:) ↔ A(i,:)       // permet de permuter U et les τ_k pour avoir les τ̃_k donc L̃

        A(k+1:n,k) = A(k+1:n,k)/A(k,k)
        A(k+1:n,k+1:n) = A(k+1:n,k+1:n) - A(k+1:n,k) × A(k,k+1:n)
finpour
```

4. Inversion de matrice

$$AA^{-1} = I \quad \Rightarrow \quad AA^{-1}(:, j) = e_j \quad \Rightarrow \quad \text{On résout } n \text{ systèmes}$$