

TD – Algèbre linéaire (Partie II)

Polynôme caractéristique.

Décompositions LU et de Cholesky.

Introduction et objectifs

Dans cette seconde partie du TD consacré à l'algèbre linéaire, nous abordons dans un premier temps la méthode de Leverrier-Faddeev-Souriau pour le calcul efficace des coefficients du polynôme caractéristique d'une matrice carrée. Nous introduisons ensuite le thème de la décomposition (ou factorisation) matricielle à travers les décompositions LU et de Cholesky.

Polynôme caractéristique.

La méthode de LEVERRIER-FADDEEV-SOURIAU.

Partie mathématique

On considère une matrice A dans $\mathcal{M}_n(\mathbb{K})$ où $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$ et n est un entier naturel non nul.

On en cherche le polynôme caractéristique $\chi_A(X) = \det(X I_n - A) = \sum_{k=0}^n a_k X^k$ avec $a_n = 1$.

L'idée centrale de la méthode présentée ici consiste à utiliser la matrice complémentaire de

$X I_n - A$ que nous notons $C(X) = \sum_{k=0}^n X^k C_k$ (pour rappel, $C(X)$ est la transposée de la comatrice de $X I_n - A$: $C(X) = {}^t \text{com}(X I_n - A)$ et les C_k sont des matrices de $\mathcal{M}_n(\mathbb{K})$).

1. Que peut-on dire du produit $(X I_n - A).C(X)$?
2. En convenant que $C_{-1} = 0$, montrer que $\sum_{k=0}^n X^k (C_{k-1} - A C_k) = \left(\sum_{k=0}^n a_k X^k \right) I_n$ puis en déduire une relation entre C_{k-1} , C_k et a_k .
3. Montrer que $\chi_A'(X) = \text{Tr}(C(X))$ puis en déduire : $\forall k \in \llbracket 0; n-1 \rrbracket, \text{Tr}(C_k) = (k+1)a_{k+1}$.
4. Montrer que $\forall k \in \llbracket 0; n-1 \rrbracket, a_k = -\frac{\text{Tr}(A C_k)}{n-k}$.

Partie informatique

On a $a_n = 1$ et $C_n = 0$. On va calculer $a_{n-1}, a_{n-2}, \dots, a_0$ à l'aide de $C_{n-1}, C_{n-2}, \dots, C_0$ grâce à :

$$\begin{cases} C_k = A C_{k+1} + a_{k+1} I_n \\ a_k = -\frac{\text{Tr}(A C_k)}{n-k} \end{cases}$$

pour k allant de $n-1$ à 0 .

Exercice N°1 – Mise en œuvre

Ecrire une fonction Python `PolyCar` recevant en argument un `array A` dont on vérifiera qu'il correspond à une matrice carrée et qui renvoie une liste `L` contenant, dans cet ordre, les coefficients $a_0, a_1, \dots, a_{n-1}, a_n$.

On utilisera des fonctions du module `linalg` de la bibliothèque `numpy`.

On testera le programme avec les matrices suivantes :

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \text{ avec : } \chi_A(X) = X^2 - 5X - 2.$$

$$A = \begin{pmatrix} 1 & -3 & 11 & -8 \\ 0 & 1 & 7 & 23 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & 2 \end{pmatrix} \text{ avec : } \chi_A(X) = X^4 - 6X^3 + 13X^2 - 12X + 4.$$

$$A = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \text{ avec } \chi_A(X) = X^n - X^{n-1}.$$

$$A = \begin{pmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{pmatrix} \text{ avec } \chi_A(X) = X^n - nX^{n-1}.$$

Exercice N°2 – Complexité

Evaluer la complexité de la méthode précédente (on évaluera séparément le nombre d'additions, le nombre de multiplications et le nombre de divisions).

Exercice N°3 – Déterminant

La méthode précédente permet de calculer très économiquement le déterminant d'une matrice carrée.

A partir de la fonction `PolyCar`, écrire une fonction Python `DetLFS` recevant en argument un `array A` dont on vérifiera qu'il correspond à une matrice carrée et qui en renvoie le déterminant.

Décomposition LU

On considère une matrice $M \in GL_n(\mathbb{K})$ avec $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$.

On rappelle qu'une telle matrice admet une décomposition LU si, et seulement si, les $n-1$ mineurs diagonaux principaux $\Delta_1, \Delta_2, \Delta_3, \dots, \Delta_{n-2}$ et Δ_{n-1} sont non nuls. La décomposition est alors unique.

Validation de l'existence d'une décomposition LU

Exercice N°4

Ecrire une fonction Python `ValidLU` qui recevra comme arguments un tableau numpy `M` et renverra un booléen selon que la matrice `M`, supposée inversible, admet ou non une décomposition LU.

La fonction devra vérifier que les mineurs diagonaux principaux $\Delta_1, \Delta_2, \Delta_3, \dots, \Delta_{n-2}$ et Δ_{n-1} sont non nuls.

Pour les calculs des déterminants, on utilisera la fonction `det` du module `numpy.linalg`.

Décomposition LU par identification

On rappelle que dans ce cas, on construit alternativement les lignes et les colonnes des matrices `L` et `U` à partir des formules :

$$\text{Pour tout } i \text{ dans } \llbracket 1; n-1 \rrbracket : \\ \forall j \in \llbracket i; n \rrbracket, u_{ij} = m_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \text{ puis } \forall j \in \llbracket i+1; n \rrbracket, l_{ji} = \frac{1}{u_{ii}} \left(m_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki} \right)$$

$$\text{Et on calcule enfin : } u_{n,n} = m_{n,n} - \sum_{k=1}^{n-1} l_{n,k} u_{k,n}.$$

Exercice N°5 - Mise en œuvre.

Ecrire une fonction Python `DecompoLU_Ident` qui recevra comme arguments un tableau numpy `M` et renverra un tuple de deux tableaux `L` et `U` correspondant à la décomposition LU de la matrice `M`, supposée inversible.

La fonction devra vérifier que la matrice `M` est inversible. Si la matrice `M` n'est pas carrée ou inversible, la fonction devra renvoyer un message adapté.

On validera le code à l'aide des situations suivantes :

$$\begin{aligned} M = \begin{pmatrix} 3 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} & \text{ qui donne : } L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/3 & 2/3 & 1 \end{pmatrix} \text{ et } U = \begin{pmatrix} 3 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & -2/3 \end{pmatrix} \\ M = \begin{pmatrix} 2 & 1 & -1 & 1 \\ 0 & -1 & 2 & 2 \\ 4 & 3 & -3 & -3 \\ 2 & 3 & -6 & 3 \end{pmatrix} & \text{ qui donne : } L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & -1 & 1 & 0 \\ 1 & -2 & -1 & 1 \end{pmatrix} \text{ et } U = \begin{pmatrix} 2 & 1 & -1 & 1 \\ 0 & -1 & 2 & 2 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 3 \end{pmatrix} \end{aligned}$$

Factorisation de Cholesky

Présentation

Une matrice symétrique définie positive (SDP) `M` peut être écrite comme produit d'une matrice triangulaire inférieure (`L`, cette lettre correspondant à l'anglais « lower ») et de sa transposée (`L'`, qui est donc triangulaire supérieure) : $M = L \times L'$. C'est la factorisation de Cholesky.

En imposant aux termes diagonaux d'être strictement positifs (ils sont tous non nuls puisque le déterminant de `M`, qui est strictement positif (`M` étant SDP), est le carré du produit des coefficients diagonaux de `L` (ou de `L'` puisqu'ils sont identiques)), cette factorisation est unique. La matrice `L` (ou sa transposée) alors obtenue peut être considérée comme une racine carrée de la matrice `M`.

Pour montrer que $M \in \mathcal{M}_n(\mathbb{R})$ est SDP, on peut :

- Calculer les valeurs propres de `M` et montrer qu'elles sont toutes strictement positives. On prendra garde aux situations où toutes ou partie de ces valeurs propres sont proches de 0, les approximations de calcul pouvant conclure à la nullité erronée d'une ou plusieurs valeurs propres non nulles.

- Utiliser le critère de Sylvester :

$$M \text{ est SDP} \Leftrightarrow \forall p \in \llbracket 1; n \rrbracket, \Delta_p = \det M_p > 0 \text{ où } M_p = (m_{ij})_{(i,j) \in \llbracket 1; p \rrbracket^2}$$

Cette méthode offre l'avantage de pouvoir être mise en œuvre facilement.

Exercice N°6 – Mise en œuvre

Ecrire une fonction `Cholesky` qui reçoit en argument une matrice M de $\mathcal{M}_n(\mathbb{R})$ et qui :

- Vérifie que la matrice M est bien SDP.
- Renvoie la matrice triangulaire inférieure $L = (l_{ij})_{(i,j) \in \llbracket 1; n \rrbracket^2}$ telle que :

$$M = L \times {}^tL \text{ et } \forall k \in \llbracket 1; n \rrbracket, l_{kk} > 0$$

On calcule les coefficients de L colonne par colonne et on vérifiera que l'on a, pour toute matrice $M \in \mathcal{M}_n(\mathbb{R})$ SDP d'ordre supérieur ou égal à 2 :

- $l_{11} = \sqrt{m_{11}}$ et $\forall i \in \llbracket 2; n \rrbracket, l_{i1} = \frac{m_{i1}}{l_{11}}$.
- Puis, pour tout entier $k \in \llbracket 2; n \rrbracket$:

$$l_{kk} = \sqrt{m_{kk} - l_{k1}^2 - l_{k2}^2 - \dots - l_{k,k-1}^2} = \sqrt{m_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$$

$$\forall i \in \llbracket k+1; n \rrbracket, l_{ik} = \frac{1}{l_{kk}} (m_{ik} - l_{k1}l_{i1} - l_{k2}l_{i2} - \dots - l_{k,k-1}l_{i,k-1}) = \frac{1}{l_{kk}} \left(m_{ik} - \sum_{j=1}^{k-1} l_{kj}l_{ij} \right)$$

On testera la fonction avec les matrices suivantes :

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} \quad \begin{pmatrix} 4 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 4 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$