Travaux pratiques n°9

Résolution de l'équation matricielle $AX = B$

F. Cuvellier, J. Tanoh

M.P.S.I. 1, 2018–2019

On se propose de résoudre l'équation matricielle $AX = B$. D'une façon générale, une matrice M sera un tableau de tableaux, tous de même longueur, mais une matrice colonne pourra aussi être vue comme un simple tableau de nombres.
On notera qu'en Python, l'indexation de chaque liste commence par 0, contrairement à l'usage courant en mathématiques. Par ailleurs, on n'oubliera pas que le résultat d'un calcul peut être nul en théorie et non nul en pratique (pour un logiciel comme Python).

1 Algorithme du pivot

1.1 Copies de collections

Dans les exercices suivants, on sera amené à modifier des tableaux. Du fait qu'en Python l'affectation se fonde sur la référence aux objets, si deux références `A` et `B` portent sur le même objet, toute modification de `B` concerne aussi l'objet `A`.
Par exemple, si `A` est le tableau `[0, 1, 2]`, à la suite des opérations `B = A` et `B[2] = 3`, l'objet `A` est `[0, 1, 3]`.
Pour effectuer une copie d'un tableau, on peut se servir de l'opérateur d'extraction (*slice* en anglais) : `B = A[:]`. Dans ce cas, `B` se réfère à un autre objet, et une modification sur lui ne concernera pas l'objet auquel se réfère `A`.
Pour effectuer une copie d'une collection, on peut se servir des fonctions `copy.copy()` et `copy.deepcopy()` (du module `copy`). La première effectue une copie élément par élément de la collection passée en argument, la seconde effectue une copie de la collection, mais aussi une de toute collection qui en serait un élément ou un élément d'élément, etc.

Exercice 1. Dans chacune des questions suivantes, on copie un objet `old` : `new = old` (par exemple), et on modifie sa copie. À la fin, on compare `old` et `new`. On doit obtenir des objets différents. (L'objet `old` ne doit pas être modifié.)

1. À partir de `old = [2, 3, 5, 7]`, effectuer la modification sur la copie `new[3] = 8`.
2. À partir de `old = [2, 3, 5, [7, 11]]`, effectuer la modification `new[3][0] = 8`.

1.2 Systèmes de Cramer triangulaires

Exercice 2.

1. Donner un algorithme pour la résolution de l'équation matricielle $TX = B$, où T est une matrice triangulaire supérieure dont les éléments diagonaux sont tous non nuls.
2. Le réaliser en Python. (La fonction obtenue ne s'assurera pas que la matrice T est triangulaire.)

1.3 Opérations élémentaires sur les lignes d'une matrice

Exercice 3. Donner des fonctions `typeI()`, `typeII()`, `typeIII()` qui effectuent les opérations élémentaires de type I ($L_h \leftarrow L_h + \alpha L_k$, où h et k sont des indices distincts), de type II ($L_h \leftarrow \alpha L_h$, où α désigne un nombre inversible) et de type III ($L_h \leftrightarrow L_k$, où h et k sont des indices distincts).

1.4 Recherche d'un pivot

Exercice 4. Donner une fonction `pivot()`, à deux paramètres `A` et `k`, qui recherche le pivot de plus grand module dans la colonne d'indice `k` de la matrice `A`. Elle devra aussi donner l'indice de ligne de ce pivot.

1.5 Systèmes de Cramer

Exercice 5. Donner une fonction qui réalise la méthode du pivot pour les systèmes de Cramer.

1.6 Systèmes quelconques

Exercice 6. Donner une fonction qui réalise la méthode du pivot pour les équations matricielles $AX = B$ quelconques. Elle retournera une solution particulière et une base du sous-espace vectoriel des solutions de l'équation homogène associée, ou l'ensemble vide, selon que le système est compatible ou non.

1.7 Application au calcul de l'inverse d'une matrice carrée

Exercice 7. Donner une fonction qui résolve l'équation $AX = I_n$.

Exercice 8. Pour les nombres entiers naturels n plus petits que 9, inverser la matrice de Hilbert d'ordre n, c'est-à-dire la matrice carrée d'ordre n dont l'élément d'indice (h, k) est $1/(h + k - 1)$ $(1 \leqslant h, k \leqslant n)$. Vérifier le résultat.

1.8 Usage du module NumPy

Le module `numpy.linalg` dispose de fonctions pour traiter les questions d'Algèbre linéaire. Par exemple, la fonction `solve()` permet de résoudre les équations matricielles.

2 Autres méthodes

2.1 Méthode de point fixe

L'inverse d'une matrice diagonale dont les éléments diagonaux sont tous non nuls se calcule aisément. On peut généraliser cette situation.
Une matrice carrée d'ordre n dont le module de chaque élément diagonal est strictement plus grand que la somme des modules des autres éléments de la ligne est inversible.
Considérons le cas d'une telle matrice A. Soient D la matrice diagonale dont les éléments diagonaux sont ceux de A et H la matrice $A - D$. Pour résoudre l'équation $AX = B$, on se sert du fait que toute suite $(X_n)_{n\in\mathbf{N}}$ telle que, pour tout nombre entier naturel n,

$$X_{n+1} = D^{-1}(-HX_n + B)$$

est convergente vers la solution X_∞ de l'équation $AX = B$. Plus précisément, en notant de façon générale $\|M\|$ le plus grand des modules des éléments de M, on peut établir la relation

$$\|X_\infty - X_n\| \leqslant \frac{R^n}{1 - R}\|X_1 - X_0\|,$$

où R désigne le nombre strictement plus petit que 1

$$\max_{1\leqslant k\leqslant n} \frac{1}{a_{kk}} \sum_{h\neq k} |a_{hk}|.$$

Exercice 9.

1. Écrire un algorithme fondé sur ces formules.
2. Quelle est sa complexité ?
3. Le réaliser en Python.

2.2 Calcul du polynôme minimal d'une matrice carrée

Étant donné un endomorphisme u d'un espace vectoriel de dimension finie d, pour tout élément $\vec{x}$ de E, la suite $(u^n(\vec{x}))_{n\in\mathbf{N}}$ d'éléments de E est liée. Il existe donc un plus petit nombre entier naturel n tel que $u^n(\vec{x})$ soit une combinaison linéaire des vecteurs $u^k(\vec{x})$ $(0 \leqslant k \leqslant n-1)$. Le polynôme unitaire f de plus bas degré tel que $f(u)(\vec{x})$ s'appelle le *polynôme minimal* de $\vec{x}$. On le note $p_{\vec{x}}$.
On peut démontrer que toute base de E possède un vecteur $\vec{a}$ tel que $p_{\vec{a}}(u) = 0$. Comme le polynôme minimal de tout vecteur de E divise tout polynôme annulateur de l'endomorphisme u, on obtient ainsi un procédé de calcul du *polynôme minimal* de u, c'est-à-dire du polynôme unitaire de plus bas degré qu'annule u.

Exercice 10.

1. Donner un algorithme de calcul du polynôme minimal d'un vecteur colonne pour une matrice carrée donnée.
2. Le réaliser en Python.
3. Donner un algorithme de calcul du polynôme minimal d'une matrice carrée. Évaluer sa complexité.
4. Le réaliser en Python.
5. En déduire un algorithme de calcul de l'inverse d'une matrice carrée inversible.