TP : Numérique (révisions)

I) Des matrices

Pour sauver (respectivement charger) le contenu d'un tableau Numpy dans (resp. depuis) un fichier texte :

```
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
>>> np.savetxt('tableau.txt', a)
>>> c = np.loadtxt('tableau.txt')
```

Pour sauver (resp. charger) dans un format binaire compressé et optimisé, on utilise `np.save` (resp. `np.load`). L'extension est `.npy`. Ce format est obligatoire pour les tableaux de dimension 3 ou plus.

Exercice 1

Dans cet exercice, avec Python on pourra utiliser la fonction array de la bibliothèque numpy, ainsi que la fonction `eigvals` de la sous-bibliothèque `numpy.linalg`. (ex : `from numpy import linalg as LA`)

1) Créer deux matrices $R = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ et $S = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ et les faire afficher.

2) Créer une fonction `test`, d'argument `M` , renvoyant la valeur `n` si `M` est une matrice carrée d'ordre `n` (entier naturel non nul), et zéro dans tous les autres cas. Vérifier la fonction test sur `R` et sur `S`.

3) Le fichier `matrice1.txt`, situé sur ma page web, contient un tableau de valeurs flottantes. Lire ce tableau dans le fichier et vérifier qu'il correspond bien à une matrice carrée d'ordre 5 que l'on désignera par `M1`.

4) Déterminer les valeurs propres de la matrice `M1`.

5) Créer une fonction `dansIntervalle`, d'arguments une liste L et deux réels a et b, renvoyant la valeur `True` si tous les éléments de la liste L sont dans l'intervalle $[a, b]$ et `False` sinon. Vérifier que toutes les valeurs propres de la matrice `M1` sont dans l'intervalle $[0, 1]$.

6) Déterminer la matrice de passage P telle que $M = PDP^{-1}$ avec D une matrice diagonale que l'on précisera *(Il existe une commande pour créer une matrice diagonale)*. Vérifiez les calculs. Que constate-t-on ?

Exercice 2

Soit H_n la matrice de Hilbert d'ordre n, de coefficients $h_{ij} = \dfrac{1}{i+j-1}$ pour $1 \leqslant i, j \leqslant n$.

1) Créer une fonction `hilbert` d'argument n qui renvoie la matrice de Hilbert d'ordre n sous forme de tableau numpy.

2) Inverser [1] H_{10} et faire afficher le coefficient en bas à droite. Le résultat attendu (formel) est 44914183600. Que constate-t-on ? Pour $n = 20$, le coefficient correspondant devrait être 48722219250572027160000.

3) Résoudre le système $H_5 X = Y$ avec $Y = \begin{pmatrix} -7.7 \\ -6 \\ 2.1 \\ -0.5 \\ 0 \end{pmatrix}$ puis $Y = \begin{pmatrix} -7.7 \\ -6 \\ 2.1 \\ -0.4 \\ 0 \end{pmatrix}$.

Que peut-on en conclure ?

1. A l'aide de votre pivot de Gauss de l'an dernier, soigneusement rangé, ou bien de la fonction ad hoc de `numpy.linalg`

4) Rappel : dans le cas d'une matrice symétrique réelle A, on appelle conditionnement de A le réel

$$\kappa(A) = \frac{\max\limits_{\lambda \in \mathrm{Sp}\,(A)} |\lambda|}{\min\limits_{\lambda \in \mathrm{Sp}\,(A)} |\lambda|}$$

Si $AX = Y$ et $A(X + \delta X) = Y + \delta Y$, alors $\dfrac{\|\delta X\|}{\|X\|} \leqslant \kappa(A) \dfrac{\|\delta Y\|}{\|Y\|}$.

Donc $\kappa(A)$ permet de mesurer la perturbation sur la solution X lorsqu'on perturbe l'entrée Y (on a un résultat analogue lorsqu'on perturbe A).

L'an dernier, nous l'avions vu dans le cas d'une matrice diagonale. Désormais, nous savons diagonaliser.

Déterminer les valeurs propres maximale et minimale de H_n pour un certain nombre de valeurs de n. Conclure.

5) Le cas des matrices tridiagonales $V_n = \begin{pmatrix} 2 & -1 & & & (0) \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ (0) & & & -1 & 2 \end{pmatrix}$ est plus intéressant en pratique : ce sont des matrices qui servent à résoudre les équations aux dérivées partielles de type $\Delta f = g$.

Les valeurs propres sont $\lambda_k = 2\left(1 + \cos\left(\dfrac{k\pi}{n+1}\right)\right)$ pour $1 \leqslant k \leqslant n$, et un rapide calculs de DL donne

$$\kappa(V_n) \sim \frac{4n^2}{\pi^2}$$