TP1

Mathématiques de la décision

L'objectif de cette séance est de réviser les fonctions de bases de la librairie numpy de Python liées au calcul matriciel, de les utiliser dans le cadre de la résolution de systèmes linéaires : $Ax = b$, où $A \in \mathcal{M}_{m,n}$, $b \in \mathbb{R}^n$ et $x \in \mathbb{R}^m$ désigne l'inconnue et d'implémenter l'algorithme du pivot.

1. Calcul matriciel avec Python et numpy

Python est devenu un standard aussi bien dans le monde académique (recherche, enseignement, lycée, etc.) que dans le monde industriel. C'est un langage de programmation simple d'accès (au moins en surface) et d'une redoutable efficacité. Il est libre et s'utilise sur toutes les plateformes (Linux, Mac OSX, Windows). De plus, pour le calcul scientifique, on dispose de la librairie numpy qui permet de rendre encore plus facile toutes les opérations de bases que l'on peut vouloir faire dans ce contexte (algèbre linéaire, optimisation, statistique, etc.). Ici, il s'agit de se familiariser avec la manipulation de matrices et quelques opérations d'algèbre linéaire. Comme tout apprentissage d'un langage de programmation, il faut utiliser la gigantesque source d'information qu'est internet. Le site officiel de numpy est très bien fait : NumPy.

1.1. **Écrire un script python.** Python peut s'utiliser en mode console. L'interpréteur interactif permet d'écrire et d'exécuter du code Python à la volée, de faire des tests rapides, d'obtenir facilement des informations sur une fonction ou un module. . .Ça peut être un bon moyen de faire des testes. Cependant, en utilisant l'interpréteur interactif, nous ne pouvons pas garder l'historique de nos commande pour résoudre tel ou tel problème.
Les commentaires.
Un commentaire est un texte ajouté au code source d'un programme servant à décrire le code source, facilitant sa compréhension par les humains. Il est donc séparé du reste du code grâce à une syntaxe particulière, ce qui fait qu'en général, le commentaire est ignoré par le compilateur ou l'interpréteur du langage concerné. Pour pouvoir (re)comprendre rapidement son code, le réutiliser plus tard et le partager avec d'autres humains, il est nécessaire de commenter correctement son code. Ici, la syntaxe particulière est le # en début de ligne qui fera que l'interpréteur python ignorera ce qui se trouve derrière ce caractère.

1.2. **Librairies numpy.** La librairie numpy contient des fonctions essentielles pour traiter les tableaux, les matrices et les opérations de type algèbre linéaire avec Python. Dans cette section, je mets un très rapide listing des différentes fonctions que nous allons utiliser tout au long de ce TP.
Nous importons la librairie numpy avec la ligne suivante :

```
1 import numpy as np
```

qui nous permet de faire référence à numpy par simplement np. **Les tableaux**.
Pour définir des matrices, ou tableaux, nous utiliserons numpy.array(), c'est-à-dire la fonction array() de la librairie numpy. Si nous voulons définir un tableau d'une ligne contenant les valeurs [1 ; 2 ; 3 ; 4], on écrira alors :

```
3
4  import numpy as np
5
6  A = np.array([1,2,3,4])
```

Pour une matrice de deux lignes par exemple on fera :

```
3
4  import numpy as np
5
6  A = np.array([[1,2,3,4],[5,6,7,8]])
```

Voici quelques opérations de bases sur les array :

```
3
4  import numpy as np
5
6  # tableaux
7  A = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]])
8  # copy de tableau
9  B = A.copy()
10 # sans le .copy(), si on modifie la variable C, on modifie aussi A
11 C = A
12 # accès à un élément
13 print("A[0,3] = ", A[0,3])
14 # accès à une ligne entière
15 D = A[1,:]
16 # accès à seulement quelques éléments d'une colonne
17 # indice départ : indice arrivé : step
18 E = A[1:4:2, 2]
19
20 # affichage
21 print("A = ", A)
```

Listes des fonctions utiles.
Voici une listes des fonctions qui peuvent vous être utiles pour ce TP (liste non exhaustive bien entendu).

```
type()

np.vstack()
np.hstack()
np.concatenate()
np.dot()
np.sum()
np.diag()
np.eye()
np.arange()
np.ones()
np.shape()
np.ndarray

np.linalg.eig()
np.linalg.inv()
np.linalg.matrix_rank()
np.linalg.solve()
```

1.3. **Structure de contrôle. boucles et fonctions.**

L'indentation est fondamentale en Python. Les blocs sont identifiés par l'indentation, au lieu d'accolades comme en C ou C++ ; ou de begin ... end comme en Pascal ou en Ruby. Une augmentation de l'indentation marque le début d'un bloc, et une réduction de l'indentation marque la fin du bloc courant.

Boucle for

```
import numpy as np

# boucle: i variant de 0 à 9
for i in np.arange(10):
    print("i = ", i)
```

Structures de Test if/else/elif

```
if mois == 'Décembre':
    print 'Joyeux Noel'
elif mois == 'Janvier':
    print 'Bonne année'
else :
    print 'rien ce mois'
```

Définition de fonction. On peut définir des fonctions avec python. Voici la syntaxe :

```
def nom_fonction(liste de paramètres):
      bloc d'instructions
```

Un exemple :

```
def sommeEntiers(stop):
    i = 0
    somme = 0
    while i < stop:
        i = i + 1
        somme = somme + i
    return somme
```

2. Exercices

Exercice 1. *(Prise en main)* On considère la matrice $A \in \mathcal{M}_{3,4}(\mathbb{R})$ définie par :

$$A = \begin{pmatrix} 4 & 6 & -2 & 3 \\ 2 & -1 & 0 & 1 \\ -7 & 0 & 1 & 12 \end{pmatrix}.$$

(1) Définir la matrice A comme un np.array().

(2) Modifier la matrice A pour que ses deux premières lignes soient multipliées par 2 et que sa dernière ligne soit divisée par 3.

(3) Créer une nouvelle matrice B définie par

$$B = \begin{pmatrix} 4 & 5 & 6 \\ 5 & 10 & 15 \\ 1 & 1 & 1 \end{pmatrix}$$

en utilisant le fait que les lignes 1 et 2 sont composées des éléments successifs de deux suites arithmétiques (voir fonction np.arange et np.ones()).

(4) Créer la matrice $C \in \mathcal{M}_{3,3}(\mathbb{R})$ extraite de A telles que pour $1 \leqslant i, j \leqslant 3$, et $c_{ij} = a_{ij}$.

(5) Réaliser le produit matriciel D de B et A (np.dot()).

(6) Calculer la somme des éléments de la matrice B et le vecteur colonne $Y \in \mathbb{R}^3$ tel que pour $1 \leqslant i \leqslant 3$, $y_i = \sum_{j=1}^{4} d_{ij}$ (np.sum()).

Exercice 2. Nous allons ici explorer quelques commandes très utiles en algèbre linéaire. On considère la matrice $A \in \mathcal{M}_{4,4}(\mathbb{R})$ définie par :

$$A = \begin{pmatrix} 4 & 5 & 6 & -1 \\ 5 & 10 & 15 & 2 \\ 6 & 15 & 1 & 14 \\ -1 & 2 & 4 & -2 \end{pmatrix}.$$

(1) Pourquoi A est-elle diagonalisable ? À l'aide de la fonction np.linalg.eig(), calculer avec Python ses valeurs propres et donner une base de vecteurs propres.

(2) Calculer de deux manières l'inverse de A en utilisant le résultat précédent et la fonction np.linalg.inv. Comparer les résultats obtenus (vous pourrez regarder la librairie matplotlib pour faire des tracés avec Python).

Exercice 3. Soit la matrice $A \in \mathcal{M}_{3,5}(\mathbb{R})$ définie par :

$$A = \begin{pmatrix} 1 & -1 & 2 & 1 & 2 \\ -1 & 2 & 3 & -4 & 1 \\ 0 & -1 & 1 & 0 & 0 \end{pmatrix}.$$

(1) Quel est le rang de la matrice A ? Utiliser la fonction np.linalg.matrix_rank afin de retrouver ce résultat.

(2) En définissant
$A = \begin{pmatrix} 1 & -1 & 2 \\ -1 & 2 & 3 \\ 0 & -1 & 1 \end{pmatrix}$ et $b = \begin{pmatrix} 3 \\ -7 \\ 1 \end{pmatrix}$,
résoudre à l'aide de la fonction np.linalg.solve() le système $Ax = b$.

Exercice 4. Matrice augmenté

(1) Coder la matrice $A = \begin{pmatrix} 4 & 4 & 8 \\ 1 & -3 & 5 \\ 3 & 7 & 3 \end{pmatrix}$, le vecteur $B = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$ et la matrice augmentée $C \coloneqq [A|B]$ (la ligne verticale de séparation n'est pas nécessaire).

(2) Faire afficher A, B et C.

3. Systèmes linéaire et pivot de Gauss

On va s'intéresser à l'implémentation dans cet environnement de la méthode du pivot. Il s'agit d'éviter de recourir aux boites noires que constituent les fonctions préprogrammées et surtout de prendre en main Python et d'apprécier la simplicité de l'implémentation des opérations associées à la méthode.

Exercice 5. On s'intéresse au système de 5 équations linéaires à 7 inconnues $(x_i)_{1 \leqslant i \leqslant 7}$,

$$\begin{cases} 4x_1 + 2x_2 + x_3 + 4x_4 + 5x_5 + 6x_6 + 7x_7 = 1 \\ x_1 + 4x_2 - 6x_3 + 2x_4 - 2x_5 \quad\quad - 2x_7 = 2 \\ x_1 - 2x_2 + 3x_3 + 3x_4 - 5x_5 + 6x_6 + x_7 = -10 \\ x_1 - x_2 + x_3 - x_4 + 2x_5 - 3x_6 - x_7 = -2 \\ x_1 + x_2 + \quad\quad 8x_4 + 2x_5 + 3x_6 + 4x_7 = 3 \end{cases}$$

(1) Reformuler le problème sous la forme $Ax = b$.

(2) Définir la matrice $G \in \mathcal{M}_{5,8}(\mathbb{R})$ obtenue par adjonction du vecteur colonne b à la matrice A, c'est-à-dire pour $1 \leqslant j \leqslant 7$ et $1 \leqslant i \leqslant 5$, $G_{ij} = A_{ij}$ et pour $1 \leqslant i \leqslant 5$ on a $G_{i8} = b_i$.
C'est à la matrice G que l'on va appliquer la méthode du pivot de Gauss.
Soit L_i le vecteur ligne associé à la i-ème ligne de G pour tout $1 \leqslant i \leqslant 5$, on adopte alors la notation suivante :

$$G = (L_1, ..., L_5)^\top.$$

(3) Créer la fonction permL qui prend en argument une matrice G, une ligne i et une colonne j, et qui renvoie l'opération $L_i \leftrightarrow L_j$ sur G.

(4) Créer la fonction ajoutS qui prend en argument une matrice G, une ligne i, une colonne j et un scalaire a, et qui renvoie l'opération $L_i \leftarrow L_i + aL_j$ sur G.

(5) Créer la fonction multiS qui prend en argument une matrice G, une ligne i et un scalaire a, et qui renvoie l'opération $L_i \leftarrow aL_i$ sur G.

(6) Tester les 3 fonctions en affichant le résultat.

Description de l'algorithme.

(a): Première itération du pivot. Si $a_{11} \neq 0$,
$\rightarrow L_1 \leftarrow L_1/a_{11}$;
$\rightarrow L_i \leftarrow a_{i1}L_1$ **pour** $2 \leqslant i \leqslant 5$.

(b): Deuxième itération du pivot. Si $a_{22} \neq 0$,
$\rightarrow L_2 \leftarrow L_2/a22$;
$\rightarrow L_i \leftarrow a_{i2}L_2$ **pour** $3 \leqslant i \leqslant 5$.

(c): etc.

Algorithme du pivot de Gauss

(7) Créer la fonction pivotGauss qui échelonne-réduit une matrice $\overline{C} = (a_{i,j})_{i,j} \in \mathcal{M}_{n,m}(\mathbb{R})$ **par cet algorithme :**

Début
$p = 1$
Pour j **de** 1 **à** m
$k = \mathbf{argmax}_{p \leqslant i \leqslant n} |a_{i,j}|$
Si $a_{k,j} \neq 0$
$L_k \leftarrow \frac{1}{a_{k,j}} L_k$
Si $k \neq p$ **:** $L_k \leftrightarrow L_p$
Pour i **de** 1 **à** n
Si $i \neq p$ **:** $L_i \leftarrow L_i - a_{i,j}L_p$
Fin Pour
$p \leftarrow p + 1$
Fin Si
Fin Pour
Fin

(8) Tester la fonction pivotGauss sur la matrice G **en affichant le résultat.**