Magistère de mathématiques

TP1 – Initiation à Python

Python est un langage informatique qui a la particularité d'avoir des commandes relativement simples et énormément de bibliothèques qui permettent une programmation plus fluide. Python est un langage beaucoup utilisé à la fois dans des entreprises (Google, NASA,...) à la fois dans la conception de logiciels libres (Blender, LibreOffice,...).

1 Utilisation du shell (interpréteur Python)

On peut directement utiliser Python comme une super-calculatrice opérant directement sur des matrices de flottants, les nombres réels ou complexes étant considérés comme des matrices de flottants de taille 1×1. En important auparavant les bibliothèques `numpy` et `math` (`import numpy as np` et `from math import *`), tester quelques commandes dans le shell :

Définition de vecteurs et matrices

```
x = np.array([1, 2, 3])
y = np.array
    ([[4],[5],[6]])
Y = np.array([4,5,6]).T
YY=np.array([[4,5,6]]).T
z = np.matrix([[1j,
    2],[4, pi]])
Z = np.array([[1j,
    2],[4, pi]])
t = np.arange(1, 5)
u = np.linspace(1, 5, 5)
```

Matrices particulières

```
id = np.eye(5, 5)
zz = np.zeros((5, 3))
zzz = np.zeros_like(y)
un = np.ones((6, 2))
```

Extraction de coefficients

```
x[1]
z[0, 1]
x[1:3]
z[0:1, 0:2]
u[t]
z[t[0:2], x[0]]
z[t[0:2]-1, x[0]]
```

Opérations d'algèbre linéaire

```
z**2
z * np.eye(2)
z.T
x + y.T
z * y[0:2]
np.linalg.solve(z, y
    [0:2])
```

Opérations coefficient par coefficient

```
Z**2
Z * np.eye(2)
x**2
x / y.T
1 / x
```

Attention!!!
On a utilisé parfois l'objet `np.matrix`, attention, il ne se comporte pas de la même manière que l'objet `np.array`. Dans tous les TP, on utilisera l'objet `np.array` afin de bien distinguer le produit terme à terme `A*B` et le produit matriciel `A.dot(B)` (que l'on peut utiliser pour `B` une matrice ou `B` un vecteur).

Exercice 1. ***Définition de matrices***

a. Avec deux boucles sur i et j (voir fiche mémo), définir la matrice P de taille 5×5 dont les coefficients sont $P_{i,j} = \cos(i\pi/5) + \sin(j\pi/7)$, pour $0 \leqslant i, j \leqslant 4$.

b. En mettant à profit sa structure très particulière, comme somme de deux matrices de rang 1, redéfinir la matrice P sans utiliser aucune boucle.

c. À l'aide de la commande `np.diag` (voir la fiche mémo), construire la matrice L de taille 10×10 dont les coefficients sont

$$\forall i, j \in \{1, \dots, 10\},\ \ell_{ij} = 2, \ \text{ si } i = j, \ \text{ et } \ell_{i,j} = -1 \ \text{ si } i - j = \pm 1.$$

2 Programmation avec Python

Il est possible de regrouper des instructions Python dans des fichiers Python (extension : .py). Il faut ensuite exécuter les fichiers .py dans un shell afin de pouvoir utiliser les fonctions qu'on a définies dans le fichier.

Utilisation de fonctions

Créer le fichier "mafonction.py" contenant :

```
def f(x):
  return(x**3)
```

On peut utiliser directement la fonction `f` dans un shell Python, après avoir exécuter le fichier "mafonction.py" dans le shell :

```
f(4)
f([1,2,3])
```

Exécution de scripts

Créer le fichier "script.py" contenant la suite d'instructions suivante

```
def f(x):
  return(x**3)

n = 10
A = np.zeros((n,3))
for i in range (n):
  x = np.array([i,i+1,i
     +2])
  A[i,:] = f(x)

print(A)
```

Son exécution dans le shell renverra directement la matrice `A` car on a utilisé la fonction `print()`.

Recommandations

• Dans un fichier .py avec plusieurs fonctions, nous recommandons de tester toutes les fonctions séparément.

• Il vaut mieux encapsuler des lignes de script dans des fonctions afin de pouvoir les exécuter quand on le veut. En effet, sinon après plusieurs exécutions du code, on aura créé plusieurs fois les mêmes objets.

3 Outils graphiques (bibliothèque "matplotlib.pyplot")

Il faut importer la bibliothèque `matplotlib.pyplot` en mettant au début du code la commande `import matplotlib.pyplot as plt`.

Graphes de fonctions

```
x = np.linspace(0,4,100)
y = np.sin(x)
z = np.cos(x)
plt.plot(x,y)
plt.plot(x,z)
plt.show()
```

Textes et légendes

```
plt.plot(x,y,label="sin"
    )
plt.plot(x,z,label="cos"
    )
plt.legend()
plt.title("graphes
    trigonometriques")
plt.show()
```

Partitionnement de la fenêtre

```
plt.subplot(121)
plt.plot(x,y)
plt.title("sinus")
plt.subplot(122)
plt.plot(x,z)
plt.title("cosinus")
plt.show()
```

Motifs et couleurs

```
plt.plot(x,y,'o')
```

pour des points

```
plt.plot(x,y,'--',color
    = "red")
```

pour des pointillés en rouge

```
plt.plot(x,y,'o--')
```

pour des points liés avec des pointillés

Exercice 2. *Précision machine*

Soit $(u_n)_{n\in\mathbb{N}}$ la suite de terme général $u_n = 1 + 2^{-n}$.

a. Calculer numériquement les valeurs $(u_n)_{0\leq n\leq 60}$.

b. À partir de ces valeurs, obtenir numériquement celles de la suite $v_n = u_n - 1$, $0 \leq n \leq 60$. Qu'observez-vous ?

c. Sur une figure, tracer les points de coordonnées (n, w_n) pour $0 \leq n \leq 60$ où $w_n = -\log_2(u_n - 1 + 2^{-100})$.

d. Comment interpréter vos observations ?

Exercice 3. *Quotient de Rayleigh d'une matrice hermitienne*

a. Programmer la fonction `Rayleigh(A,x)` qui prend en argument une matrice $A \in \mathcal{H}_n(\mathbb{C})$ hermitienne et un vecteur non-nul $x \in \mathbb{C}^n$, et renvoie le réel $R_A(x) = \langle Ax, x\rangle / \langle x, x\rangle$ où $\langle\cdot,\cdot\rangle$ est le produit scalaire hermitien canonique.

b. On considère les matrices

$$A = \begin{pmatrix} 2 & 1 \\ 1 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & i \\ -i & -3 \end{pmatrix}.$$

Avec la commande `np.linalg.eig`, déterminer les valeurs propres et vecteurs propres de A et B.

c. Pour tout $\theta \in \mathbb{R}$, on pose $x(\theta) = (\cos\theta, \sin\theta)^T \in \mathbb{R}^2$. Tracer la courbe de la quantité $R_A(x(\theta))$ en fonction de $\theta \in [0, 2\pi]$. Comparer aux valeurs propres de A.

d. Pour tout $\theta \in \mathbb{R}$, et tout $\phi \in \mathbb{R}$, on pose $z(\theta, \phi) = (\cos(\theta)e^{i\phi}, \sin(\theta))$. Considérant successivement différentes valeurs de $\phi \in [0, 2\pi]$, tracer la courbe de la quantité $R_A(z(\theta, \phi))$ en fonction de $\theta \in [0, 2\pi]$. Comparer aux valeurs propres de B.

Exercice 4. *Interpolation polynomiale*

a. Examiner les programmes proposés dans la fiche mémo pour le problème d'interpolation polynomiale.

b. Programmer une version de ce même problème d'interpolation utilisant la représentation du polynôme recherché dans la base canonique de $\mathbb{R}_n[X]$ et de ce fait, l'inversion d'un système linéaire de Vandermonde $Vx = y$ où $V_{i,j} = x_i^j$ pour $0 \leq i \leq n$ et $0 \leq j \leq n$.

c. Comparer vos résultats à ceux obtenus par la méthode de Newton (utilisée dans la fiche mémo).

d. Procéder de même avec la base de Lagrange.