MATPLOTLIB et NUMPY. RÉSUMÉ DE MÉTHODES UTILES

1 Module matplotlib

C'est un module utilisé pour tracer des courbes. On l'importe (par exemple) grâce à la syntaxe : `import matplotlib.pyplot as plt`. La fonction la plus importante est **plt.plot** :

plt.plot(X, Y)

où X = $[x_0, x_1, ..., x_{n-1}]$ et Y = $[y_0, y_1, ..., y_{n-1}]$ sont des listes ou des vecteurs lignes numpy (voir plus loin). X et Y sont respectivement les abscisses et les ordonnées des points $M_i(x_i, y_i)$ dans le plan (Oxy). Cette fonction trace la courbe reliant les points $M_0 - M_1 - ... - M_{n-1}$ par des segments de droite.

Pour afficher une figure contenant le graphe, il faut écrire :

```
plt.figure()        # Définit une figure
X = ...             # Définition de X
Y = ...             # Definition de Y
plt.plot(X, Y)
plt.show()          # Affiche la figure
```

Pour plus d'informations (gestion des couleurs, forme des points par exemple), voir le document sur matplotlib consultable sur le site de la mp1.

2 Module numpy

Le module **numpy** permet de gérer les matrices numériques. On l'importe (par exemple) en écrivant `import numpy as np`. En voici quelques fonctions et méthodes souvent utilisées. Vous trouverez plus de détails et des informations supplémentaires sur le site de la mp1.

Méthode	**Syntaxe**	**Argument**	**Exemple**	**Résultat**
Création d'un vecteur ligne	A = np.array(L)	L = $[x_0, x_1, ..., x_{n-1}]$	A = np.array([4, -2.3, 5.67])	$(4, -2.3, 5.67)$
Création d'une matrice	A = np.array(L)	L = $[L_0, L_1, ..., L_{n-1}]$ Liste de listes. Chaque liste est une ligne de la matrice	A = np.array([[1, 2], [3, 5]])	$\begin{pmatrix} 1 & 2 \\ 3 & 5 \end{pmatrix}$
Addition d'un scalaire. Ajoute x à chaque élément de A et met le résultat dans B	B = A + x	Pas d'argument	A = $\begin{pmatrix} 1 & -3 \\ 4.2 & 5.7 \end{pmatrix}$ B = A + 2	B = $\begin{pmatrix} 3 & -1 \\ 6.2 & 7.7 \end{pmatrix}$
Multiplication par un scalaire. Multiplie chaque élément de A par x et met le résultat dans B	B = A * x	Pas d'argument	A = $\begin{pmatrix} 1 & -3 \\ 4 & 5 \end{pmatrix}$ B = A * 3	B = $\begin{pmatrix} 3 & -9 \\ 12 & 15 \end{pmatrix}$
Produit de deux matrices	A.dot(B)	Tableau numpy B A.dot(B) équivaut au produit matriciel A B	A = $\begin{pmatrix} 1 & -3 \\ 4 & 2 \end{pmatrix}$ B = $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ C = A.dot(B)	C = $\begin{pmatrix} -5 \\ 8 \end{pmatrix}$

Le module **numpy** contient aussi les méthodes **arange**, **linspace** ainsi que plusieurs fonctions usuelles qui s'appliquent directement sur les matrices numpy :

- A = np.arange(*start*, *stop*, *step*) : crée un tableau numpy unidimensionnel (vecteur ligne) commençant à *start*, finissant à *stop* (exclu) avec un pas égal à *step*. Il s'agit du même type de fonction que range mais adaptée aux objets de numpy et dont les valeurs peuvent être des flottants. Exemple :

  A= np.arange(2.1, 10, 1.3) donne A = (2.1, 3.4, 4.7, 6.0, 7.3, 8.6, 9.9)

- A = np.linspace(*start*, *stop*, N) : crée un vecteur ligne de N flottants régulièrement espacés entre *start* et *stop* inclus tous les deux. Exemple :

  A = np.linspace(3, 5, 5) donne A = (3.0, 3.5, 4.0, 4.5, 5.0)

- Enfin, ce module contient aussi toutes les fonctions usuelles comme **np.cos**, **np.sin**, **np.exp**, **np.log**, etc ... Ces fonctions agissent directement sur tous les éléments d'une matrice A. Par exemple : B = np.cos(A) est la matrice dont les éléments sont les cosinus des éléments de A.