

VIII – La bibliothèque numpy

1/ Généralités

Cette bibliothèque contient une collection d'outils qui vont vite s'avérer indispensables pour le calcul scientifique :

- des fonctions numériques de référence ;
- des fonctions numériques plus évoluées ;
- des nouveaux types (nombres, vecteurs, tableaux) ;
- des fonctions sur ces nouveaux types.

Les fonctions de la bibliothèque numpy, **contrairement à celles de la bibliothèque math**, acceptent les listes en argument. Elles s'appliquent directement aux éléments successifs de celles-ci, sans qu'il faille recourir à une compréhension de liste.

Exemple : avec la fonction `cos` de numpy, on peut taper `cos(L)` pour `[cos(e) for e in L]`.

Cependant, pour toute autre fonction, la compréhension de listes restera incontournable.

2/ Fonctions usuelles

Trigonométrie Circulaire	Trigonométrie Hyperbolique	Exponentielles et Logarithmes	Nombres Complexes
<code>sin</code>	<code>sinh</code>	<code>exp</code>	<code>real</code>
<code>cos</code>	<code>cosh</code>	<code>exp2</code>	<code>imag</code>
<code>tan</code>	<code>tanh</code>	<code>log</code>	<code>conj</code>
<code>arcsin</code>	<code>arcsinh</code>	<code>log2</code>	<code>abs</code>
<code>arccos</code>	<code>arccosh</code>	<code>log10</code>	<code>angle</code>
<code>arctan</code>	<code>arctanh</code>		<code>sqrt</code>

Remarques :

- ❶ La fonction `log` représente ici le logarithme népérien et n'accepte pas la base comme second argument optionnel, contrairement à la fonction de la bibliothèque `math`.
- ❷ Si a est un complexe, `sqrt(a)` renvoie la solution de $z^2 = a$ de partie réelle positive.

<code>rint</code>	arrondit à l'entier le plus proche
<code>ceil</code>	arrondit à l'entier suivant (partie entière + 1)
<code>floor</code>	arrondit à l'entier précédent (partie entière)

3/ Les tableaux

Pour représenter les vecteurs et les matrices, on va privilégier le type `array`. Il permet de créer facilement des tableaux de toutes dimensions et dispose de toutes les fonctions utiles pour les manipuler (somme, produit scalaire, produit, etc...).

Exemple : le vecteur `v = array([1,0,2])` et la matrice `A = array([[1,2],[3,4]])`.

Les vecteurs

<code>array([x1,x2,...,xn])</code>	crée le vecteur $[x_1, x_2, \dots, x_n]$
<code>zeros(n) / zeros(n,int)</code>	crée un vecteur contenant n zéros flottants / entiers
<code>linspace(a,b,n)</code>	vecteurs de n réels uniformément répartis de a à b
<code>vdot(vect1,vect2)</code>	produit scalaire de deux vecteurs de même dimension
<code>cross(vect1,vect2)</code>	produit vectoriel de deux vecteurs de l'espace

- On accède aux coordonnées des vecteurs à l'aide des indices, comme pour les listes.
- Les fonctions de `numpy` et les opérations de base (+, *, **, /) s'appliquent terme à terme aux coordonnées des vecteurs.

Les matrices

<code>array([L1,L2,...,Ln])</code>	crée la matrice de lignes $L_1, L_2, \dots, L_n$
<code>zeros((n,p))</code>	crée une matrice nulle de n lignes et p colonnes
<code>diag(v)</code>	crée une matrice diagonale avec les coefficients de v
<code>diag(v,k)</code>	même chose en décalant la diagonale de k cases
<code>transpose(A), A.T</code>	renvoie la transposée de la matrice A (c'est un couple)
<code>shape(A), A.shape</code>	renvoie le format de la matrice A
<code>trace(A)</code>	renvoie la trace de la matrice A
<code>dot(A,B)</code>	renvoie le produit matriciel $A B$
<code>savetxt('fichier.txt',M)</code>	sauvegarde un tableau dans un fichier
<code>M=loadtxt('fichier.txt')</code>	lit un tableau stocké dans un fichier

Remarques :

- ❶ Les matrices et les vecteurs se parcourent comme des listes ou des listes de listes.
- ❷ On accède au terme d'indice i de la ligne d'indice j de A via `A[i,j]` ou `A[i][j]`.
- ❸ Il est souvent pratique de créer une matrice nulle de bon format, dont on modifie par la suite les coefficients (même idée que pour les suites récurrentes).
- ❹ Attention, **les coefficients restent tous du même type** `int` ou `float` lors des calculs.

Les fonctions suivantes se trouvent dans la bibliothèque `numpy.linalg`

<code>inv(A)</code>	inverse la matrice A
<code>solve(A,v)</code>	résout le système linéaire $Ax = v$
<code>det(A)</code>	calcule le déterminant de A

Exercices

Exercice 1.

- 1) Représenter les fonctions circulaires et leurs réciproques.
- 2) Représenter les fonctions hyperboliques et leurs réciproques.
- 3) Représenter les fonctions $\exp_a : x \mapsto a^x$ pour $a = 0.3, 0.5, 2$ et 3 .

Exercice 2.

- 1) Créer une fonction `Produit_Scalaire` qui prend pour arguments deux vecteurs de l'espace (ce sont des listes) et qui retourne leur produit scalaire.
On transformera les vecteurs en objets du type `array` pour utiliser la fonction `vdot()`.
- 2) Créer une fonction `Angle` qui mesure l'angle entre deux vecteurs de l'espace, en radians.
- 3) Modifier la fonction précédente pour qu'elle donne les angles en degrés.

Exercice 3. Soient $\vec{u} \begin{pmatrix} 2 \\ 1 \end{pmatrix}$, $\vec{v} \begin{pmatrix} 1 \\ -2 \end{pmatrix}$, $A \begin{pmatrix} 0 & 2 \\ 1 & 4 \\ 2 & 3 \end{pmatrix}$. Calculer $\vec{u} \cdot \vec{v}$, $A\vec{u}$, $A\vec{v}$ et $(A\vec{u}) \cdot (A\vec{v})$.

Exercice 4.

- 1) Créer une fonction `EquationPlan` qui :
 - prend pour arguments les coordonnées de trois points A, B, C ;
 - vérifie que les points ne sont pas alignés ;
 - détermine alors une équation du plan (ABC)
- 2) On appliquera cette fonction aux points $A(1, 2, -1)$, $B(-1, 1, 0)$ et $C(0, 1, 2)$.

Exercice 5.

- 1) Créer une fonction `VolumeTetra` qui prend pour arguments les coordonnées spatiales de quatre points A, B, C, D et qui renvoie le volume du tétraèdre $ABCD$.
- 2) On appliquera cette fonction aux points $A(1, 2, 3)$, $B(3, -1, -1)$, $C(2, 0, 1)$ et $D(1, 1, 2)$.

Exercice 6.

Construire le triangle de Pascal à un ordre quelconque.

Exercice 7.

Écrire « à la main » une fonction qui renvoie la transposée d'une matrice.

Exercice 8.

Renvoyer la décomposition d'une matrice carrée sous la forme de la somme d'une matrice symétrique et d'une matrice antisymétrique.

Exercice 9.

Vérifier que les matrices suivantes sont inversibles, puis calculer leurs inverses :

$$A = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 2 & 0 \\ 3 & 0 & 2 \end{pmatrix}$$

Exercice 10.

Résoudre les systèmes linéaires $\begin{cases} x + y + z = 3 \\ x + 2y + 3z = 6 \\ 2x + y - z = 2 \end{cases}$ et $\begin{cases} x + y + (2a - 1)z = 1 \\ ax + y + z = 1 \\ x + ay + z = 3(a + 1) \end{cases}$

Pour le second, créer d'abord une fonction qui renvoie la solution pour chaque valeur de a , puis représenter x , y et z en fonction de a .