M2 Agrégation - UE 8.

Quelques commandes Python classiques.

I. Imports classiques

`import numpy as np`
`import matplotlib as plt` ou `import matplotlib.pyplot as plt`

II. Fonctions usuelles

`np.cos, np.sin, np.log, np.exp` etc.
Il est possible de les utiliser sur des vecteurs pour calcul composantes par composantes.

III. Algèbre linéaire

Pour manipuler matrices et vecteurs, utiliser préférentiellement plutôt la structure array de numpy.
Voir l'aide de numpy pour l'utilisation précise de ces routines.

Pour construire

- une matrice : `np.array`
- une subdivision : `np.linspace`, `np.arange`.
- une matrice de zéros : `np.zeros`
- une matrice de 1 : `np.ones`
- la matrice identité : `np.eye`
- matrices diagonales et sur et sous diagonales : `np.diag`

Manipuler les tableaux et les tailles de tableaux :

- nombre d'éléments du tableau A : `np.size(A)`.
- nombre de lignes et nombre de colonnes de A : `np.shape(A)`.
- redimensionner un tableau : `np.reshape`.
- Si A est un array, l'accès à l'élément en ligne i et colonne j se fait par : `A[i,j]`.
- `A[l1:l2,c1:c2]` sous matrice extraite de la ligne l1 à l2-1 et colonnes c1 à c2-1.

Opérations sur les matrices et/ou vecteurs

- multiplication `np.dot(A,B)`. On peut aussi pour multiplier A et B deux matrices, utiliser .dot : `A.dot(B)`.
- Attention les `**` (puissance), `*` (multiplication), `/` (division) effectueront des opérations termes à termes.
- En utilisant numpy.linalg (avec e.g. `import numpy.linalg as alg`), on peut utiliser `alg.matrix_power` pour calculer la puissance d'une matrice. Par exemple pour la puissance 3 de A, on utilise `alg.matrix_power(A,3)`.
- transposée : `np.transpose`.
- inverse d'une matrice : `np.linalg.inv`.
- produit scalaire de deux vecteurs : `np.inner`.
- calculer le déterminant d'une matrice : `np.linalg.det`
- pour résoudre un système linéaire : `np.linalg.solve`
- valeurs propres : `np.linalg.eigvals`
- valeurs propres et vecteurs propres : `np.linalg.eig`
- norme matricielle ou vectorielle : `np.linalg.norm`

IV. Graphes

IV.1. **Graphes 2D de fonctions.**

Exemple :

```
plt.figure(1)
plt.clf()
plt.plot(x,y,'#couleur', LW=#largeurdeligne)
plt.legend(['#legende pour x', '#legende pour y'])
plt.xlabel('#Label pour x')
plt.ylabel('#Label pour y')
plt.title('#Titre pour la figure')
plt.show()
```

avec `#couleur`$\in \{b, r...\}$, `#largeurdeligne`=un chiffre.

On peut utiliser `plt.close('all')` pour fermer toutes les figures. `plt.axis` pour fixer les axes. `plt.subplot` pour faire des subplots.

IV.2. **Tracer un champ de vecteurs**

Avec `import matplotlib.pyplot as plt`.
Exemple :

```
fig=plt.figure(1)
xx=np.arange(-1,1,0.1) # un premier vecteur
yy=np.arange(-1,1,0.1) # un deuxième vecteur
X,Y=np.meshgrid(xx,yy) # crée le maillage associé aux vecteurs x et y choisis
FX=-X#crée la coordonnée x du champ de vecteur voulu.
FY=Y+1#crée la coordonnée y du champ de vecteur voulu.
plt.quiver(X,Y,FX,FY)# crée le graphe du champ de vecteur.
plt.savefig('nomfiguresauvee.png', dpi=fig.dpi)# sauve la figure en un png.
```

IV.3. **Graphes de surface**

Avec `np.meshgrid`
`plot_surface`

Voir le fichier graphes_surfaces.py.

Exemples :

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator, FormatStrFormatter
#Un premier graphe de surface
x=np.arange(-2,2,0.01)
y=np.arange(-1,1,0.05)
x,y=np.meshgrid(x,y)
z=x**2+y**2
fig=plt.figure(1) #Declaration de la figure
plt.clf() #nettoyage de la figure
ax = fig.add_subplot(111, projection='3d') #creation du plot
surf=ax.plot_surface(x,y,z, cmap='hot', linewidth=0, cstride=1, rstride=1, alpha=0.8,
antialiased=False)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title('Graphe de surface')
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
# Parametrage de surfaces
theta=np.arange(-np.pi,np.pi,0.01)
phi=np.arange(0.0,2*np.pi,0.05)
theta,phi=np.meshgrid(theta,phi)
x=np.cos(theta)*np.sin(phi)
```

```
y=np.sin(theta)*np.sin(phi)
z=np.cos(phi)
fig=plt.figure(2) #Declaration de la figure
plt.clf() #nettoyage de la figure
ax = fig.add_subplot(111, projection='3d') #creation du plot
surf=ax.plot_surface(x,y,z, cmap='hot', linewidth=0, cstride=1, rstride=1, alpha=0.8,
antialiased=False)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title('Graphe de surface')
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```