TP 1. Plots

September 14, 2021

1 Rappel

Pour faire des calculs efficacement et afficher des graphiques, on importera toujours les modules numpy et matplotlib via les commandes :

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#Cette dernière commande demande l'incrustation des graphes dans le notebook.
```

2 Tableaux de nombres avec numpy

La bibliothèque numpy fournit une structure de tableau de nombres ndarray (comme à priori les listes) mais la syntaxe d'utilisation est plus simple et les calculs plus efficaces.

2.1 Exécuter et observer l'effet des commandes suivantes.

[ ]:
```
x = [1, 2, 3, 4]
y = [-2, 4, -6, 8]

print(type(x), x, y)

xx = np.array(x)
yy = np.array(y)

print(type(xx), xx, yy)
```

[ ]:
```
s = [a + b for a, b in zip(x, y)]
print(type(s), s)
```

[ ]:
```
ss = xx + yy
print(type(ss), ss)
```

[ ]:
```
p = [a*b for a, b in zip(x, y)]
print(type(p), p)
```

[ ]:
```
pp = xx*yy
print(type(pp), pp)
```

[ ]:
```
c = [a**2 for a in x]
print(type(c), c)
```

[ ]:
```
cc = xx**2
print(type(cc), cc)
```

[ ]:
```
b = [a < b for a, b in zip(x, y)]
print(type(b), b)
```

[ ]:
```
bb = xx < yy
print(type(bb), bb)
```

2.2 Alors ? Que constatez vous ?

2.3 Consulter la documentation des fonctions

- np.ones
- np.zeros
- np.linspace
- np.arange

et créer les tableaux suivants x, y et z suivants :

- x sera de taille 100 et ne contiendra que des 1 ;
- y sera de taille 100 et contiendra les nombres de 1 à 100 ;
- z contiendra 200 nombres en 0 et 1, espacés régulièrement.

2.4 Remarque

La structure s'appelle ndarray car il s'agit en fait d'un tableau multi-dimensionnel, d'où le paramètre size de certaines commandes de création. Voyez plutôt :

[ ]:
```
np.random.random(size=(2,3,4))
```

[ ]:
```
np.ones(shape=(5,5))
```

[ ]:
```
x = np.linspace(0, 1, 9)
print(x.shape, x)
```

[ ]:
```
x.shape = 3 ,3
print(x.shape, x)
```

2.5 Little tip

Si on a deux listes de nombres que l'on veut additionner (ou autres opérations) on commencera par les convertir en tableau numpy.

[ ]:
```
liste_a = [1, 2, 3, 4, 5]
liste_b = [1, 4, 2, 3, 5]
a, b = np.array(liste_a), np.array(liste_b)
resultat = a+b
```

[ ]:
```
resultat
```

3 Premiers graphes

3.1 Exécuter les commandes suivantes.

[ ]:
```
x = np.linspace(0.0, 5.0, 20)
print(x)
```

[ ]:
```
y = np.sin(x)
print(y)
```

[ ]:
```
plt.plot(x, y)
```

On retiendra que l'on passe abcisses puis ordonnées. Comment lisser le graphe ci-dessus ?

Afficher (sur l'intervalle de votre choix) le graphe des fonctions suivantes :

$$f(x) = \cos(x)\sin(x); \quad g(x) = 5\mathrm{e}^{-x/7}; \quad h(x) = 2x(1-x).$$

3.2 Exécuter les commandes suivantes :

[ ]:
```
plt.plot((0, 1), (0,2))
```

[ ]:
```
plt.plot([0,1], [0,1])
```

Qu'en déduisez-vous sur la nature des arguments à passer dans la fonction plot ?

3.3 Quelques options

Il existe tout un tas de commandes pour enjoliver ses graphes.

[ ]:
```
x = np.linspace(-np.pi, np.pi, 200)
y = np.sin(2.0*x) - 2.0*np.cos(3.0*x)

plt.figure(figsize = (8, 5))
#pour gérer la taille de la figure (longueur, largeur)
```

```
plt.plot(x, y, color= "red" , ls =  "--" , lw = 2)
# ls pour line style, style du trait
# lw pour line width, épaisseur du trait
```

[ ]:
```
t = np.linspace(-np.pi, np.pi,50)
x = 2.0*np.cos(t)
y = np.sin(t)

plt.plot(x, y, color= "blue", linestyle = "-", linewidth = 2, marker = "o" )
```

[ ]:
```
t = np.linspace(0.0, 1.5, 100)
x = np.sqrt(t)
y = t**2
z = t**3

plt.figure(figsize = (8, 6))

plt.plot(t, x, linewidth = 2.0, label = r"$y=\sqrt{x}$")

plt.plot(t, t, linewidth = 2.0, label = r"$y=x$")

plt.plot(t, y, linewidth = 2.0, label = r"$y=x^2$")

plt.plot(t, z, linewidth = 2.0, label = r"$y=x^3$")

plt.legend(loc= "best", fontsize = 15)
#pour activer l'affichage des légendes, fontsize pour la taille du titre

plt.title("Graphe des premières puissances", fontsize = 15)
#titre, fontsize pour la taille du titre
```

[ ]:
```
x, y = np.random.random(size = (2, 100))

plt.scatter(x, y)

plt.xlim([0.0, 1.0])
#limites des abscisses
plt.ylim([0.0, 1.0])
#limites des ordonnées
```

3.4 Exercices

En s'inspirant des exemples précédents, tracer les graphes suivants :

1. sur la même figure, $y = x$ en rouge pointillé et $y = 4x(1 - x)$ en bleu continu large ;
2. la courbe paramétrée : $x = cos(t)$, $y = 2sin(2t)$ en trait large, avec les points d'interpolation apparents ;

3. la courbe paramétrée

$$x = \sin(t)\left(\mathrm{e}^{\cos(t)} - 2\cos(4t) - \sin^5\left(\frac{t}{12}\right)\right); \quad y = \cos(t)\left(\mathrm{e}^{\cos(t)} - 2\cos(4t) - \sin^5\left(\frac{t}{12}\right)\right)$$

4. sur la même figure, en rouge, la courbe paramétrée

$$x = \frac{\sqrt{2}}{2} + 0.05\cos(t); \quad y = \frac{\sqrt{2}}{2} + 0.05\sin(t)$$

et en bleu, les 100 premiers points de la suite (x_n, y_n) où

$$x_n = \frac{\sqrt{2}}{2} + \frac{\cos(n/4)}{n+1}; \quad y_n = \frac{\sqrt{2}}{2} + \frac{\sin(n/4)}{n+1}.$$

Limites des abscisses $[0.5, 1]$; limites des ordonnées $[0.5, 1]$; taille de la figure $(7, 7)$;

5. les 200 premiers points $P_n = (x_n, y_n)$ où

$$x_{n+1} = 1.0 - 1.4x_n^2 + y_n; \quad y_{n+1} = 0.3x_n$$

avec des valeurs initiales prises au hasard entre 0 et 1.

On pourra consulter cette gallerie pour plus d'exemples.