Graphiques - Matplotlib

1. Introduction

En science, la nécessité de tracer des graphiques est omniprésente que ce soit pour analyser l'évolution d'une fonction, traduire un fichier de résultats d'une mesure, traiter une image, réaliser une animation,...
Dans ce but, ce cours présente certaines fonctionnalités de la bibliothèque Matplotlib de Python.

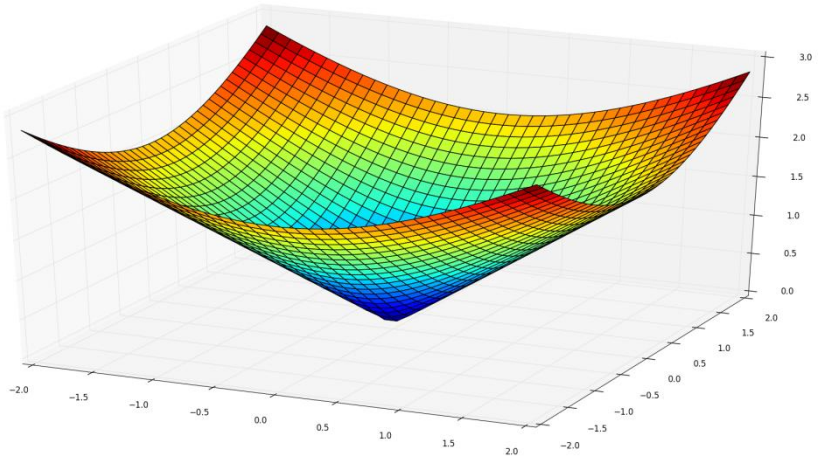
2. Matplotlib

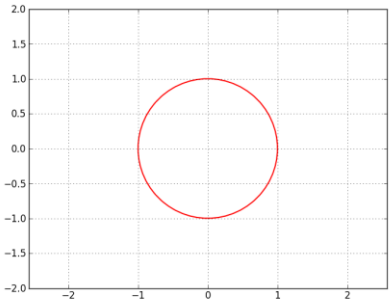
Une courbe s'obtient par la donnée de deux listes de même taille X et Y. Un couple (X[i],Y[i]) correspond à un point de la courbe qui est d'autant mieux définie que les points sont *rapprochés*.
L'annexe à la fin de ce cours synthétise les fonctionnalités de base de Matplotlib nécessaires au tracé d'une courbe 2D et à son *habillage*. L'aide de Python est toujours à votre disposition pour en savoir plus sur une de ces fonctions.

3. Extensions

3.1 Courbe en 3D

Le module `mpl_toolkits.mplot3d` est une extension de Matplolib qui permet de tracer des courbes en 3D. Le scrypt et la figure ci-dessous en donnent un exemple :





```
# Importation des modules
from mpl_toolkits.mplot3d import *
from pylab import *
fig1 = figure(1)
ax = Axes3D(fig1)

# Définition du maillage et de la fonction
x,y = linspace(-2,2,40),linspace(-2,2,40)
X,Y = meshgrid(x,y)
R2 = X**2 + Y**2
Z = sqrt(R2)

# Tracé en 3D
ax.plot_surface(X,Y,Z, rstride=1,cstride=1,cmap=cm.jet)

# Coupe
fig2 = figure(2)
contour(X, Y, Z, levels=[1.5], colors='red',linewidths=1.5)
axis("equal")
grid()
show()
```

3.2 Animation

La bibliothèque Matplotlib comporte un module Animation. La fonction `FuncAnimation` appelle la fonction `init`() qui sert à créer l'arrière de l'animation qui sera présent sur chaque image et la fonction `animate`() qui met à jour la courbe pour chaque image.

```
from pylab import *
from matplotlib import animation

# Définition des éléments que l'animation doit modifier
fig = figure()
ax = axes(xlim=(0, 30), ylim=(0, 1))
ttl = ax.text(15,0.5, '')
line, = plot([], [])

# Fonction d'initialisation
def init():
    ttl.set_text('')
    line.set_data([], [])
    return line,ttl

# Fonction modifiant les éléments
def animate(i):
    tau=[5,4,3,2,1,0.5]
    x = linspace(0, 30, 100)
    y = 1-exp(-x/tau[i])
    ttl.set_text('tau='+str(tau[i])+' s')
    line.set_data(x, y)
    return line,ttl

# Animation avec les arguments frames (nbr appels) et interval (durée en ms entre deux modifications)
anim = animation.FuncAnimation(fig, animate, init_func=init, frames=6, interval=1000)

plt.grid()
plt.show()
```

3.3 Image

Une image peut être considérée comme un tableau 2D (matrice) où chaque cellule (*pixel*) est affectée d'une couleur. La procédure `imshow()` permet d'afficher un tableau comme une image. Les couleurs sont fixées par la palette standard `colormap` selon la valeur de la cellule.

Le scrypt suivant affiche l'image correspondante à une matrice 4×4 dont les coefficients sont définis aléatoirement. La procédure `imshow()` dispose d'une option d'interpolation qui permet de définir la transition entre les couleurs de cellules adjacentes.

```
from pylab import *
A = rand(4,4)
figure(1)
imshow(A, interpolation='nearest')
figure(2)
imshow(A)
show()
```

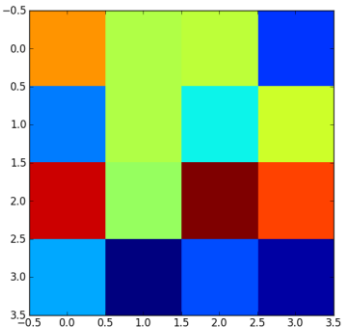


Figure 1

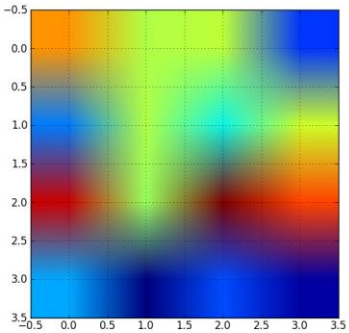


Figure 2

Les codes barres (1D ou 2D) sont des traductions graphiques d'informations codées en binaire. Le scrypt ci-dessous affiche l'image correspondante à une matrice 30×30 dont les coefficients valent aléatoirement 0 ou 1. L'option `cmap=cm.gray` sélectionne la palette de niveau de gris dans laquelle 0 correspond à noir et 1 à blanc.

```
from pylab import *
# Création de la matrice aléatoire
A =random.randint(2,size=(30,30))
# Affichage de la matrice en tant qu'image
figure(1)
imshow(A, interpolation='nearest',cmap=cm.gray)
show()
```

ANNEXE

Les directives ci-dessous sont accessibles après avoir importé le module `pyplot` de `matplotlib` ou directement `pylab` qui importe automatiquement `numpy` et `pyplot`.

Nom	Description
`plot(x,y, color = 'red', linestyle='--', linewidth = '2')`	Trace la courbe y=f(x) où x et y sont des listes de même dimension. `Color` fixe la couleur de la courbe, `linestyle` le type de tracé et `linewidth` la largeur du trait. Styles de ligne possibles : : = pointillés -- = ligne en tirets -. = ligne tiret-point
`plot(x,y,'o')`	Trace les points de coordonnées (x,y) en les repérant par un marqueur (les points ne sont pas reliés). Styles de marqueurs possibles : o = boulette ^ = triangle . = point x = croix s = carré (square) + = plus * = étoile
`show()`	Affiche la courbe dans une figure (à placer en fin de programme une fois la courbe totalement paramétrée)

`figure (figsize = (L,H))`	Dimensionne la fenêtre graphique pour l'affichage (L : largeur, H : hauteur)
`Xlim(min,max)`	Borne les valeurs d'abscisses
`Ylim(min,max)`	Borne les valeurs d'ordonnées
`xlabel('...')`	Affiche un titre pour l'axe des abscisses
`ylabel('...')`	Affiche un titre pour l'axe des ordonnées
`title('...')`	Affiche un titre pour la courbe
`legend`	Affiche une légende pour la courbe
`Xticks(liste1,liste2)`	Fixe l'emplacement `(liste1)` et le texte `(liste2)` des graduations de l'axe des abscisses
`yticks(liste1,liste2)`	Fixe l'emplacement `(liste1)` et le texte `(liste2)` des graduations de l'axe des ordonnées
`text(x,y,'texte')`	Affiche « texte » au point de coordonnées (x, y)
`grid()`	Affiche une grille sur la figure

Fonctions Numpy également utiles :

Nom	Description
`linspace(min, max, nbr)`	Retourne un tableau de `nbr` valeurs régulièrement réparties entre `min` et `max`.
`arange (min, max, pas)`	Retourne un tableau de valeurs régulièrement réparties d'un `pas` entre `min` et `max`.
`meshgrid (x, y)`	Crée un tableau 2D (*grille ou maillage*) à partir des tableaux 1D x et y

Illustration de la fonction `meshgrid` :

```
>>> x,y = linspace(-2,2,3),linspace(-2,2,3)
>>> X,Y = meshgrid(x, y)
>>> X
array([[-2.,  0.,  2.],
       [-2.,  0.,  2.],
       [-2.,  0.,  2.]])
>>> Y
array([[-2., -2., -2.],
       [ 0.,  0.,  0.],
       [ 2.,  2.,  2.]])
```