Graphiques avec Matplotlib

On importe tout d'abord les modules `pyplot` et `numpy`.

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

Remarque 1 *On lancera ipython en tapant* `ipython --pylab`

1 L'instruction `plot`

Pour tracer "à la main" la courbe $y = f(x)$, on choisit quelques points d'abscisses x_0, x_1, x_2, puis on calcule les ordonnées correspondantes : y_0, y_1, y_2. Ensuite, on joint ces points avec habileté et quelques connaissances sur la fonction (notamment sur la dérivée). Plus le nombre de points est élevé, meilleure est l'allure de la courbe. Matplotlib fait le même raisonnement avec la commande `plot`, l'avantage est qu'il peut utiliser un nombre élevé de points.

L'instruction graphique la plus simple est :

```
plt.plot(x,y)
```

où `x` et `y` sont des vecteurs contenant respectivement les abscisses et les ordonnées des différents points considérés.
Voici, en exemple, comment tracer la fonction $x \to \sin(x)$ sur l'intervalle $[-2\pi, 2\pi]$:

```
x=np.linspace(-2*np.pi,2*np.pi,num=100)
y=np.sin(x)
plt.plot(x,y)
plt.show(block=False)
```

On notera que l'instruction `plt.show(block=False)` permet de récupérer la main sous l'interpréteur ipython par exemple quand le graphique est lancé.

1.1 Couleur et Style

On peut spécifier la couleur et le style de la courbe

```
plt.plot(x,np.sin(x),'r+-.')
```

On touvera toutes les couleurs, styles de traits et marqueurs en tapant `?plt.plot`.

1.2 Epaisseur du trait et du marqueur

```
plt.plot(x,np.sin(x),linewidth=2,markersize=15)
```

2 Embellir son graphique

- Ajouter des légendes

  On peut préciser les données représentrées en abscisses/ordonnées, le titre ou ajouter du texte à des coordonnées précisées:

```
plt.plot(x,np.sin(x))
plt.xlabel('x')                    # l\'egende sur axe des abscisses
plt.ylabel('y')                    # l\'egende sur axe des ordonn\'ees
Iplt.title('La fonction sinus')   # titre
plt.text(0,0,'voici un mot')   # ajout d'un mot
plt.text(0.03,2,'u(a)', fontsize=19)
```

On peut aussi ajouter une légende:

```
plt.plot(x,np.sin(x),label='sin')
plt.plot(x,np.cos(x),marker='v',label='cos')
plt.legend()
 plt.show(block=False)
```

On peut positionner la légende où l'on veut avec l'option `loc`:

```
plt.legend(loc=2)
```

- Changer la taille des caractères

```
plt.xticks(fontsize=17) # Taille des caract\'eres en ordonn\'ee
plt.yticks(fontsize=17) # Taille des caract\'eres en abscisses
plt.legend(prop={'size':15}) # Taille de la l\'egende
```

- Ajuster les axes

```
plt.axis([xmin, xmax, ymin, ymax,]) #ajuste les axes en x et y
plt.xlim(xmin,xmax)  #ajuste l'axe des abscisses
plt.ylim(ymin,ymax)  #ajuste l'axe des ordonn\'ees
plt.axis('equal')  #avoir la meme echelle en x et y
```

3 Gérer plusieurs graphiques

Si on veut observer plusieurs courbes en même temps, trois possibilités s'offrent à nous :

- on peut tracer toutes les courbes sur un même graphique et donc sur une même figure. Dans ce cas, on exécute les instructions `plot` les unes après les autres, les courbes se superposeront (ceci est différent dans Matlab). Exemple:

```
x = np.arange(-np.pi,np.pi,0.01);
plt.plot(x,np.sin(x),'r*-');
plt.plot(x,np.cos(x),'b--');
plt.show()
```

- on peut tracer chaque courbe sur une figure différente. Dans ce cas l'instruction `plt.figure()` permet d'ouvrir une nouvelle fenêtre graphique. Exemple:

```
plt.figure()
plt.plot(x,np.sin(x),'r*-');
plt.figure()
plt.plot(x,np.cos(x),'b--');
plt.show()
```

- on peut subdiviser une figure en une matrice de sous-fenêtres en utilisant la commande `subplot`. Elle s'utilise de la manière suivante :

```
subplot(m,n,p)
```

Cette commande symbolise le découpage de la figure en une matrice m lignes et n colonnes. Le nombre p est compris entre 1 et $n*m$, il désigne le numéro de la sous-fenêtre où l'on veut tracer son graphe. Par exemple, si on veut tracer sin et cos sur une même figure mais sur des graphiques différents :

```
x = np.arange(-np.pi,np.pi,0.01);
plt.subplot(2,1,1)
plt.plot(x,np.sin(x),'b--');
plt.subplot(2,1,2)
plt.plot(x,np.cos(x),'r-.',linewidth=6);
plt.show()
```

4 Sauver une figure sous le format .eps ou .gif

```
plt.savefig('nomfig.eps')
```

5 Légendes en Latex

```
        plt.rc('text', usetex=True)
        nom=r"$\epsilon$="+str(ep)
        plt.title(nom)
```

6 Animation

Une animation est une suite de graphiques affichés à l'écran. En voici un exemple:

```
import numpy as np
import matplotlib.pylab as plt
from matplotlib.animation import FuncAnimation

def animate(i):
        x=np.linspace(0,10,100)
        y=np.sin(x-0.1*i)
        line.set_data(x,y)
        return line,

fig=plt.figure(1)
line,=plt.plot([],[])
plt.axis([0,10,-1,1])

plt.show()
anim=FuncAnimation(fig,animate,)
HTML(anim.to_html5_video())
anim.save('myvideo.mp4', codec='h264') # pour sauvegarder le film
```

7 En 3 dimensions

7.1 Courbe dans l'espace

Le tracé de courbes en 3D se fait à l'aide de la commande `plot` mais il faut préciser auparavant que le graphique se fait en 3D

```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = plt.axes(projection='3d')

t=np.linspace(0,2,num=300)
x=np.sin(20*t)*(1+t**2)
y=np.cos(20*t)*(1+t**2)
z=t

ax.plot(x, y, z)
plt.show()
```

7.2 Surfaces

Pour la représentation de la surface $z = f(x, y)$, on a besoin de connaître les triplets de coordonnées $(X_i, Y_i, Z_{i,j} = f(X_i, Y_j))$ pour un certain nombre de points (X_i, Y_i) $i = 1 \cdots N, j = 1 \cdots M$. On

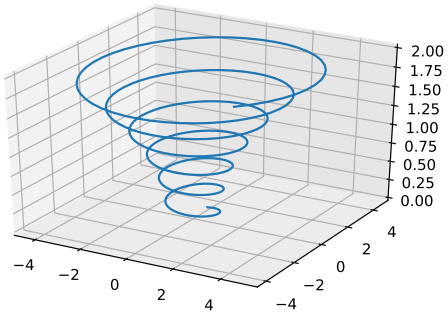

Figure 1: Plot en 3D

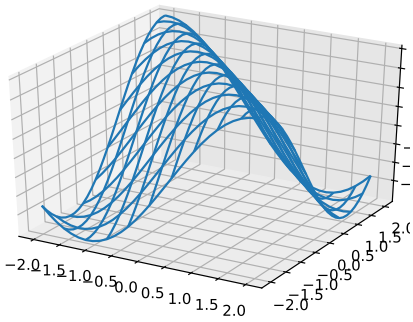

Figure 2: **mesh.**

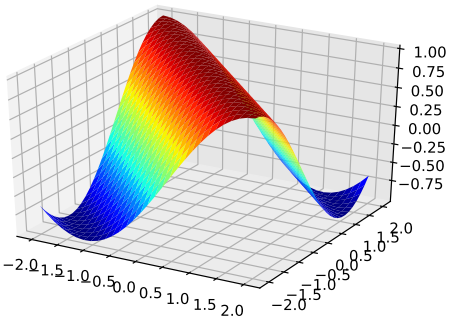

Figure 3: **surf.**

voit que Z a la structure d'une matrice. Matplotlib demande également à ce que X et Y soient des matrices. On construit ces dernières avec l'instruction `np.meshgrid`.

Si $x = (x_0, x_1, \cdots, x_n)$ et $y = (y_0, y_1, \cdots, y_m)$, l'instruction `[X,Y] = np.meshgrid(x,y)` produit les matrices X et Y à m lignes, n colonnes suivantes :

$$X = \begin{pmatrix} x_0 & x_1 & \cdots & x_n \\ x_0 & x_1 & \cdots & x_n \\ \cdots & \vdots & \vdots & \vdots \\ x_0 & x_1 & \cdots & x_n \end{pmatrix}, \qquad Y = \begin{pmatrix} y_0 & y_0 & \cdots & y_0 \\ y_1 & y_1 & \cdots & y_1 \\ \cdots & \vdots & \vdots & \vdots \\ y_m & y_m & \cdots & y_m \end{pmatrix}.$$

```
from mpl_toolkits.mplot3d import Axes3D

x = np.linspace(-2, 2, 30)
y = x.copy()
[X,Y]=np.meshgrid(x,y)
Z = np.cos(X+Y)

fig = plt.figure(1)
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z ,cmap=plt.cm.jet)
plt.show()

fig = plt.figure(2)
ax = plt.axes(projection='3d')
ax.plot_wireframe(X, Y, Z ,cmap=plt.cm.jet, rstride=3, cstride=3)
plt.show()
```

7.3 Lignes de niveau

```
x = np.linspace(-2, 2, 30)
y = x.copy()
[X,Y]=np.meshgrid(x,y)
Z = X**2+Y**2

fig = plt.figure(1)
ax=fig.gca()
CS=ax.contour(X, Y, Z,20)
ax.clabel(CS)
plt.show(block=False)
```

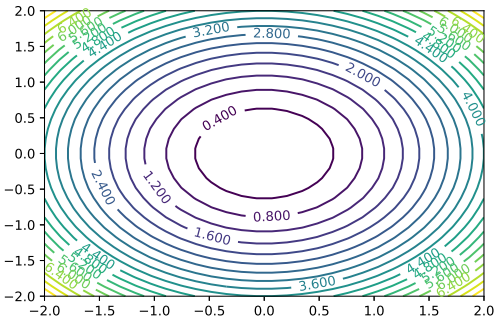

Figure 4: contour