MATPLOLIB

TP

Le but de ce TP est d'apprendre à utiliser la librairie Matplotlib.

Jy Plantec

SOMMAIRE

MATPLOTLIB

Le module Matplotlib est chargé de tracer les courbes :

```
>>> import matplotlib.pyplot as plt
```

L'objet n'est pas ici de donner une vue complète de toutes les possibilités de Matplotlib. Pour plus de détails, on se réfèrera à la documentation en ligne. On pourra également consulter le site **http://scipy-lectures.github.io/intro/matplotlib/matplotlib.html** qui donne de nombreux exemples.

D'une manière générale les fonctions **`plt.plot`** prennent en arguments des vecteur/matrice, bref des tableaux de points du plan. Selon les options, ces points du plan sont reliés entre eux de façon ordonnée par des segments : le résultat est une courbe.

Commençons par la fonction sinus.

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(-5,5,100)
plt.plot(x,np.sin(x)) # on utilise la fonction sinus de Numpy
plt.ylabel('fonction sinus')
plt.xlabel("l'axe des abcisses")
plt.show()
```

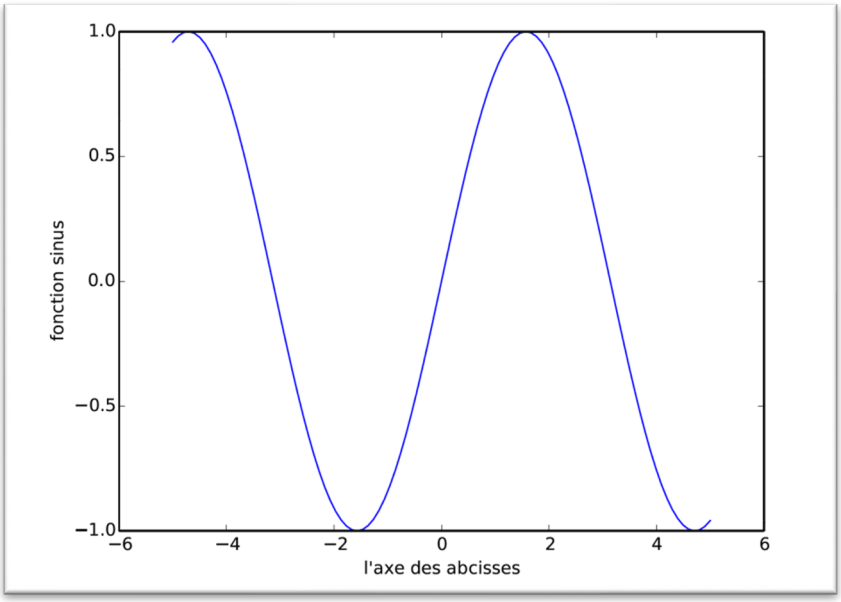

Figure 1 : Graphe de la fonction sinus

Après la commande **`plt.show`**, une fenêtre s'ouvre comme la *figure 1*. Il est possible d'utiliser des menus au bas de cette fenêtre : zoomer, déplacer la figure, etc et surtout sauvegarder dans un format PNG, PDF, EPS, etc.

D'autres commandes sont utiles.

`plt.clf()` efface la fenêtre graphique

`plt.savefig()` sauvegarde le graphique.
Par exemple `plt.savefig("mongraphe.png")` sauve sous le nom "mon- graphe.png" le graphique. Par défaut le format est PNG. Il est possible d'augmenter la résolution, la couleur de fond, l'orientation, la taille (a0, a1, lettertype, etc) et aussi le format de l'image. Si aucun format n'est spécifié, le format est celui de l'extension dans "nomfigure.ext" (où "ext" est "eps", "png", "pdf", "ps" ou "svg"). Il est toujours conseillé de mettre une extension aux noms de fichier ; si vous y tenez **`plt.savefig('toto',format='pdf')`** sauvegardera l'image sous le nom "toto" (sans extension !) au format "pdf".

En mode interactif Python, une caractéristique est le mode interactif de cette fenêtre graphique. Si vous avez tapé l'exemple précédent, et si cette fenêtre n'a pas été fermée alors la commande **`plt.xlabel("ce que vous voulez")`** modifiera l'étiquette sous l'axe des abcisses. Si vous fermez la fenêtre alors la commande **`plt.xlabel("ce que vous voulez")`** se contentera de faire afficher une fenêtre graphique avec axe des abcisses, des ordonnées allant de 0 à 1 et une étiquette "ce que vous voulez" sous l'axe des abcisses. L'équivalent "non violent" de fermer la fenêtre est la commande **`plt.close()`**.

L'inconvénient, une fois un premier graphique fait, est le ré-affichage ou l'actualisation de cette fenêtre graphique au fur et à mesure des commandes graphiques : lenteur éventuelle si le graphique comporte beaucoup de données. Il est donc indispensable de pouvoir suspendre ce mode interactif. Heureusement tout est prévu !

`plt.isinteractive()` Retourne `True` ou `False` selon que la fenêtre graphique est interactive ou non.
`plt.ioff()` Coupe le mode interactif.
`plt.ion()` Passe en mode interactif.
`plt.draw()` Force l'affichage (le "retraçage") de la figure.

Ainsi une fois la première figure faite pour revenir à l'état initial, les deux commandes **`plt.close()`** et **`plt.ioff()`** suffisent.

Pour connaître toutes les options, le mieux est de se référer à la documentation de Matplotlib. Voyons ici quelques-unes d'entre elles :

- Bornes : spécifier un rectangle de représentation, ce qui permet un zoom, d'éviter les grandes valeurs des fonctions par exemple, se fait via la commande **`plt.axis([xmin,xmax,ymin,ymax])`**

- Couleur du trait : pour changer la couleur du tracé une lettre g vert (green), **`r`** rouge (red), **`k`** noir, **`b`** bleu, **`c`** cyan, **`m`** magenta, **`y`** jaune (yellow), **`w`** blanc (white). **`plt.plot(np.sin(x),`**'r') tracera notre courbe sinus en rouge. Les nuances de gris sont obtenues via color='(un flottant entre 0 et 1)'. Enfin pour avoir encore plus de couleurs, la séquence **`color='#eeefff'`** donnera la couleur attendue en hexadécimal. On peut également manipuler les couleurs RGB par **`color=( R, G, B)`** avec trois paramètres compris entre 0 et 1.

- Symboles : mettre des symboles aux points tracés se fait via l'option marker. Les possibilités sont nom- breuses parmi **[+ | * | , | . | 1 | 2 | 3 | 4 | < | > | D | H | ^ | _ | d | h | o | p | s | v | x | | | TICKUP | TICKDOWN | TICKLEFT | TICKRIGHT | None |].**

- Style du trait : pointillés, absences de trait, etc se décident avec linestyle. Au choix - ligne continue,

- **--** tirets, **-.** points-tirets, **:** pointillés, sachant que 'None', '', **' '** donnent "rien-du-tout". Plutôt que **linestyle, ls**(plus court) fait le même travail.

- Épaisseur du trait **: linewidth=flottant** (comme **linewidth=2**) donne un trait, pointillé (tout ce qui est défini par style du trait) d'épaiseur "flottant" en points. Il est possible d'utiliser **lw**en lieu et place de **linewidth**.

- Taille des symboles (markers) : **markersize=flottant** comme pour l'épaisseur du trait. D'autres para- mètres sont modifiables **markeredgecolor** la couleur du trait du pourtour du marker, **markerfacecolor** la couleur de l'intérieur (si le marker possède un intérieur comme 'o'), **markeredgsize=flottant** l'épaisseur du trait du pourtour du marker. Remarquez que si la couleur n'est pas spécifiée pour chaque nouvel appel la couleur des "markers" change de façon cyclique.

- Étiquettes sur l'axe des abcisses/ordonnées : Matplotlib décide tout seul des graduations sur les axes. Tout ceci se modifie **via plt.xticks(tf), plt.yticks(tl)** où **tf** et **tl** sont des vecteurs de flottants ordonnés de façon croissante.

- Ajouter un titre : **plt.title("Mon titre")**

- Légendes : il faut définir les labels des légendes au moment du tracé. Voir l'exemple ci-après.

```
>>> x=np.linspace(-np.pi,np.pi,100)
>>> plt.plot(x,np.sin(x),color="red",  linewidth=2.5, linestyle="-",
label="sinus")
>>> plt.plot(x,np.cos(x),color="blue",  linewidth=2.5, linestyle="-",
label="cosinus")
>>> plt.legend(loc='upper left')
>>> plt.axis([-np.pi,np.pi,-1,1])
>>> plt.yticks([-1, 0, +1],
   [r'$-1$', r'$0$', r'$+1$'])
>>> plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
   [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
```

EXERCICE 1

Réaliser le graphe de la fonction $y(t) = v_0 t - \frac{1}{2} g t^2$ pour $v_0 = 10$, $g = 9.81$, et $t \in [0, 2v_0/g]$. Le label sur l'axe des x devra être "temps (s)" et le label sur l'axe des y "hauteur (m)".

EXERCICE 2

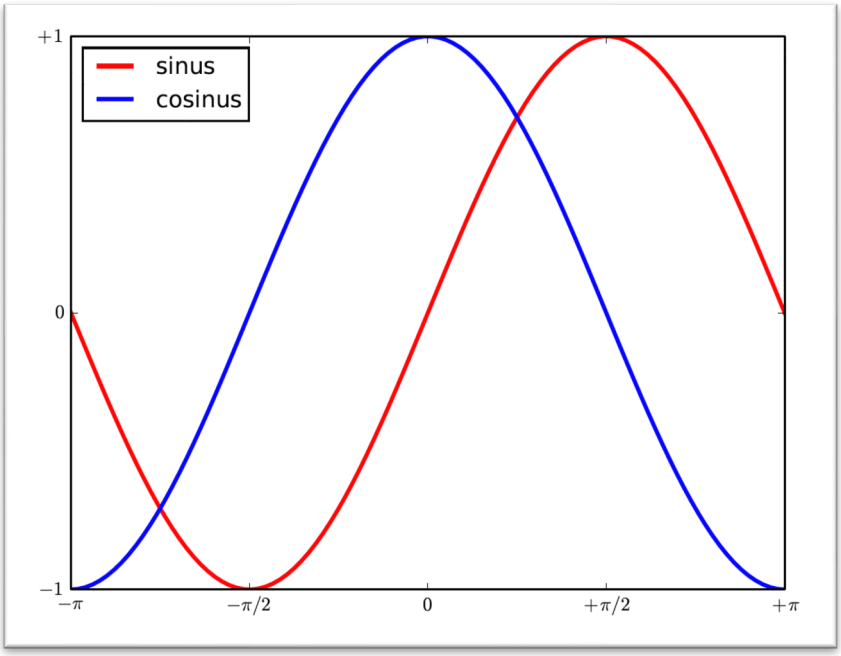

Figure 2 : Graphe des fonctions sinus et cosinus

Faire un programme qui lit un ensemble de valeurs v_0 en ligne de commandes et trace les courbes correspondantes $y(t) = v_0 t - \frac{1}{2} g t^2$ pour $g = 9.81$ dans la même fenêtre graphique. On prendra $t \in [0, 2v_0/g]$ pour chaque courbe, de telle sorte que on devra fabriquer un vecteur t différent pour chaque courbe.

EXERCICE 3

La fonction

$$f(x,t) = e^{-(x-3t)^2} \sin(3\pi(x - t))$$

décrit pour une valeur fixe de t une onde localisée en espace. Faire un programme qui visualise cette fonction comme une fonction de x dans l'intervalle $x \in [-4, 4]$ pour $t = 0$.

EXERCICE 4

La fonction sinus peut être approchée par un polynôme grâce à la formule.

$$\sin x \approx S(x;n) = \sum_{j=0}^{n} (-1)^j \frac{x^{2j+1}}{(2j+1)!}$$

L'erreur dans l'approximation $S(x;\ n)$ décroit quand n augmente et à la limite on a $lim_{n\to\infty}\ S(x;\ n)\ =\ sin\ x$. Le but de cet exercice est de visualiser la qualité de diverses approximations $S(x;\ n)$ quand n croit.

La première partie de l'exercice est d'écrire une fonction Python **S(x,n)** qui calcule $S(x;\ n)$.

La partie suivante est de tracer simultanément la fonction $sin\ x$ sur $[0, 4\pi]$ avec ses approximations $S(x;\ 1), S(x;\ 2), S(x;\ 3), S(x;\ 6)$ et $S(x;\ 12)$.