Cours 9 : Calcul scientifique. Tracé de courbes avec `numpy` et `matplotlib`

MPSI - Lycée Thiers

2014/2015

Modules scientifiques pour le tracé de courbes

Les modules à utiliser pour le tracé de courbes sous python :

1. numpy : Outils pour créer, manipuler, et appliquer de nombreuses opérations sur des tableaux de nombres.
   http://docs.scipy.org/doc/numpy/reference/ (en anglais).
2. matplotlib.pyplot : Permet le tracé de graphes de fonctions.
   http://matplotlib.org/users/pyplot_tutorial.html (en anglais).

Le module numpy

• Le module numpy permet de créer, de manipuler, des tableaux homogènes non-redimensionnables de nombres et de leur appliquer des opérations mathématiques courantes.

• **La fonction** array() permet de créer un tableau, ou array, à partir d'un tableau python c'est à dire d'une liste de listes ou tuples de nombres :

```
>>> import numpy as np
>>> A = np.array([1, 2, 3, 4])
>>> A
array([1, 2, 3, 4])
>>> A[0] , A[-1]
 (1, 4)
```

• Toutes les opérations sur les listes et séquences python, en dehors de celles qui redimensionnent sont possible sur un tableau numpy.

• **La fonction** arange() crée un tableau de façon assez analogue à la fonction range(), à ceci près que les coefficients ne sont pas forcément entiers :

```
>>> v = np.arange(0, 1.5, 0.5)
>>> v
array([ 0., 0.5, 1. ])
>>> 2*v
array([ 0., 1., 2. ])
>>> (2*v) ** 2 + 100
array([ 100., 101., 104. ])
```

On peut appliquer sur les tableaux fonctions et opérations mathématiques, comprises terme à terme.

Le module numpy

• **La fonction** `linspace(a, b, n)` crée le tableau des n valeurs régulièrement espacées prises entre a et b, tous deux inclus.

C'est à dire le tableau de la subdivision régulière de l'intervalle $[a, b]$ par n points (ou par $n-1$ segments). Exemple :

```
>>> v = np.linspace(0,1,10)
>>> v
array([ 0. , 0.11111111, 0.22222222, 0.33333333, 0.44444444,
    0.55555556, 0.66666667, 0.77777778, 0.88888889, 1. ])
```

• `numpy` contient aussi toutes les fonctions mathématiques (aussi présentes dans `math`) :

```
>>> np.cos(v)
array([ 1. , 0.99383351, 0.97541009, 0.94495695, 0.90284967,
    0.84960756, 0.78588726, 0.71247462, 0.63027505, 0.54030231])
```

• Tableaux (uni-dimensionnels) sous `numpy` :

`array(liste)`	crée un tableau à partir d'une liste ou séquence `liste`
`arange(a,b,k)`	crée le tableau de tous les `a+k.N` entre `a` (inclu) et `b` (exclu).
`linspace(a,b,n)`	crée le tableau des `n` valeurs régulièrement espacées entre `a` et `b` (inclus)
`zeros(p)`	crée un tableau de taille `p` rempli de zéros
`mean()`	retourne la moyenne d'un tableau
`size()`	retourne le nombre d'éléments d'un tableau

Le module matplotlib

• Pour le simple tracé de courbes nous n'utiliserons que le sous-module pyplot, importé, avec alias, à l'aide de la commande :

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

cf. documentation à : http://www.matplotlib.org.

• Les fonctions essentielles de pyplot sont :

1. plot() pour le tracé de points, de courbes, et
2. show() pour afficher le graphique créé.

• Utiliser plot() avec :

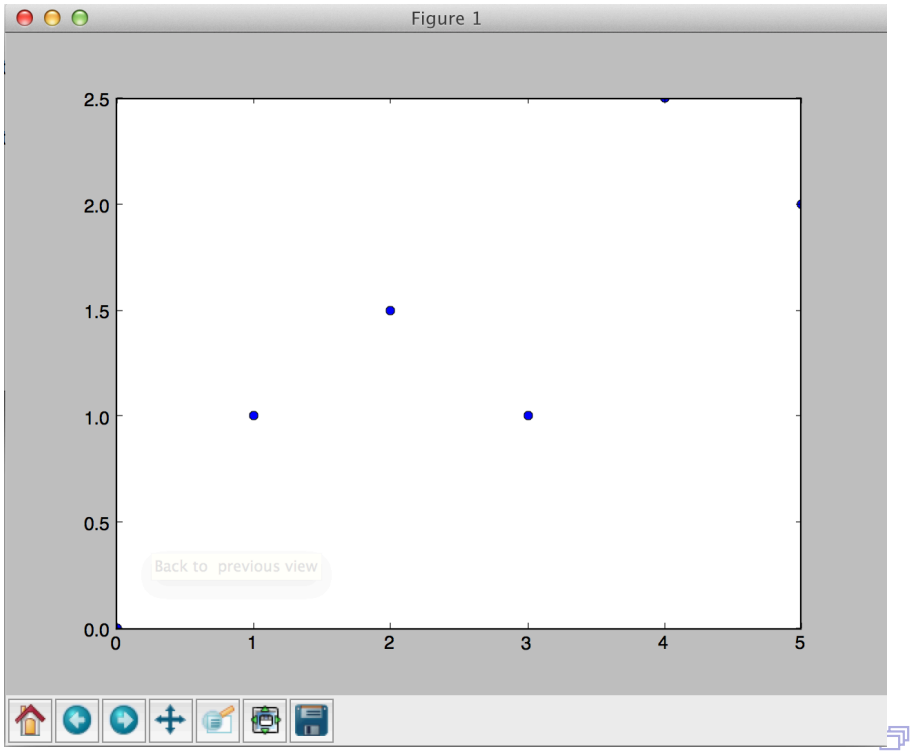
1. en 1^{er} argument la liste des abscisses,
2. en 2^{eme} argument la liste des ordonnées,
3. en 3^{eme} argument (optionnel) le motif des points :
   - 3.1 '.' pour un petit point,
   - 3.2 'o' pour un gros point,
   - 3.3 '+' pour une croix,
   - 3.4 '*' pour une étoile,
   - 3.5 '-' points reliés par des segments
   - 3.6 '--' points reliés par des segments en pointillés
   - 3.7 '-o' gros points reliés par des segments (on peut combiner les options)
   - 3.8 'b', 'r', 'g', 'y' pour de la couleur (bleu, rouge, vert, jaune, etc...)
   - 3.9 cf. http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot.

Le module matplotlib

- Exemple : pour le tracé d'un nuage de points

```
>>> import matplotlib.pyplot as plt
>>> abs = [0, 1, 2, 3, 4, 5]
>>> ord = [0, 1, 1.5, 1, 2.5, 2]
>>> plt.plot(abs, ord, 'o')
[<matplotlib.lines.Line2D object at 0x10c6610d0>]
>>> plt.show()
```

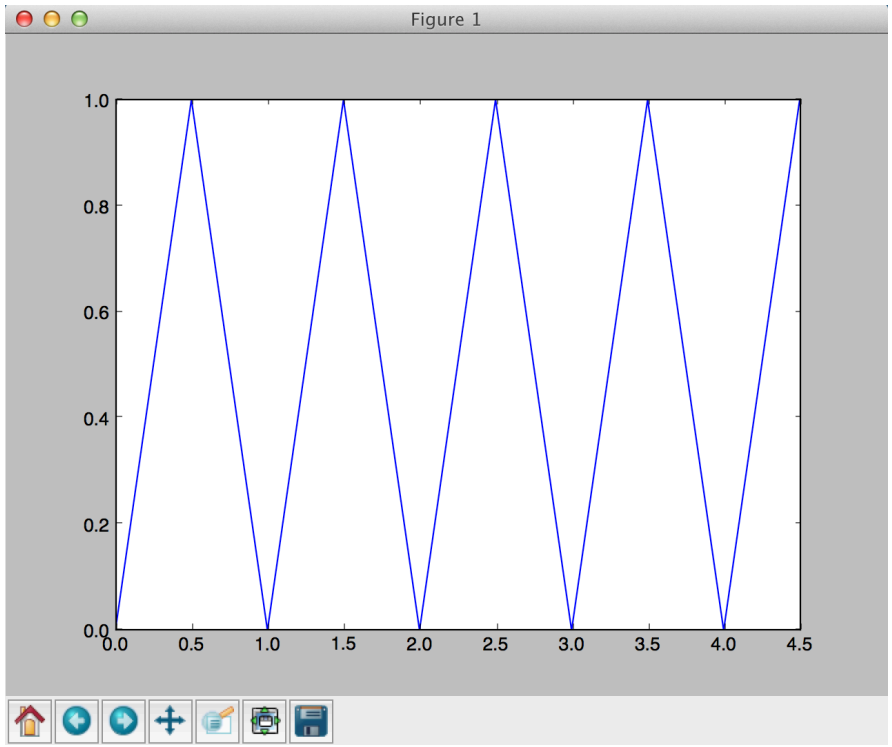
produit un graphique (au format .png) :



Le module matplotlib

- Exemple : pour le tracé d'une ligne brisée :

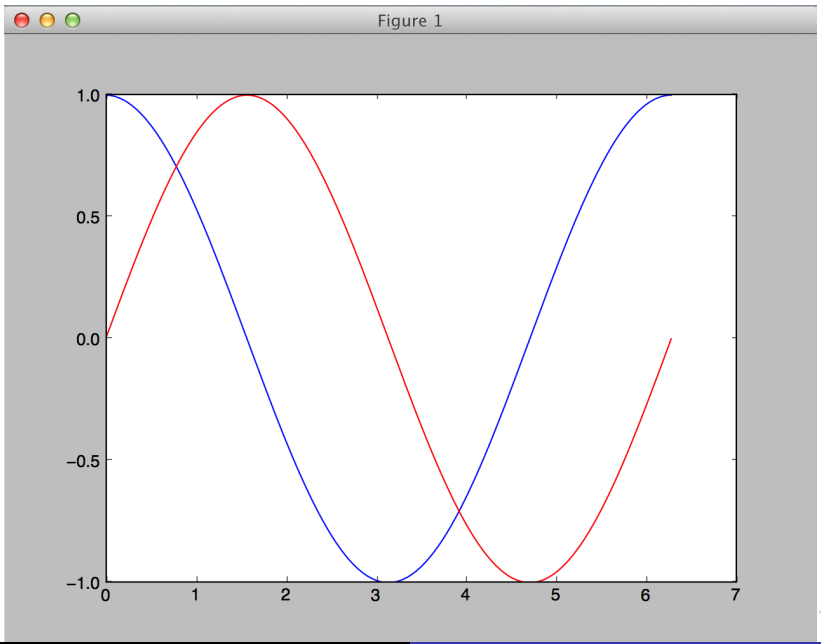
```
>>> import matplotlib.pyplot as plt
>>> abs = [n/2 for n in range(10)]
>>> ord = [n % 2 for n in range(10)]
>>> plt.plot(abs,ord,'-b')
[<matplotlib.lines.Line2D object at 0x10dd1fa10>]
>>> plt.show()
```



Le module matplotlib

- Exemple : pour le tracé de courbes représentatives de fonctions réelles :

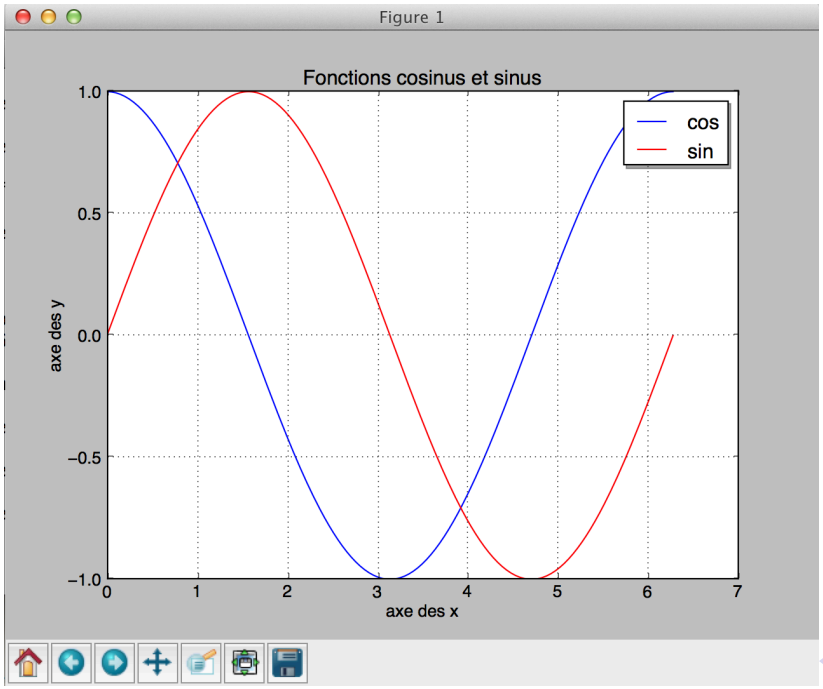
```
>>> import matplotlib.pyplot as plt
>>> import numpy as np          # pour linspace() et les fonctions mathématiques
>>> X = np.linspace(0, 2*np.pi, 255)        # X = 255 pts régulièrement espacés
>>> Ycos = np.cos(X)                        # image directe de X par cos
>>> Ysin = np.sin(X)                        # image directe de X par sin
>>> plt.plot(X,Ycos,'b')                     # tracé de la courbe de cos en bleu
[<matplotlib.lines.Line2D object at 0x10d5e2b50>]
>>> plt.plot(X,Ysin,'r')                     # tracé de la courbe de sin en rouge
[<matplotlib.lines.Line2D object at 0x1073aad90>]
>>> plt.show()
```



Le module matplotlib

• On améliore le tracé en remplissant quelques options avant de la sauvegarder (au format .png dans le répertoire utilisateur).

```
>>> plt.plot(X, Ycos, 'b', X, Ysin, 'r')    # Tracé simultané des 2 courbes
>>> plt.grid(True)                  # Affiche la grille
>>> plt.legend(('cos','sin'), 'upper right', shadow = True)    # Légende
>>> plt.xlabel('axe des x')         # Label de l'axe des abscisses
>>> plt.ylabel('axe des y')         # Label de l'axe des ordonnées
>>> plt.title('Fonctions cosinus et sinus')    # Titre
>>> plt.savefig('ExempleTrace')     # sauvegarde du fichier ExempleTrace.png
>>> plt.show()
```



Le module matplotlib

- On peut tout aussi bien tracer des courbes paramétrées.

```
>>> T = np.linspace(0,2*np.pi,255)        # paramètre t
>>> X = T * np.cos(T)                     # x(t) = t.cos(t)
>>> Y = T * np.sin(T)                     # y(t) = t.sin(t)
>>> plt.plot(X,Y,'b')                     # Tracé de la courbe paramétrée {(x(t),y(t))}
[<matplotlib.lines.Line2D object at 0x10c044ed0>]
>>> plt.show()
```