Fiche : Tracer des courbes

Le module `numpy` contient des outils permettant de générer et de manipuler des tableaux de nombres.

```
import numpy as np
```

Générer des listes d'abscisses

`L1 = list(range(n))` .....
affecte à la variable `L1` la liste d'entiers `[0,1,2,...,n-1]`

`L2 = np.linspace(a,b,n)` .....
`L2` contient un tableau contenant n valeurs appartenant à l'intervalle $[a;b]$: n représente le nombre de points utilisés pour subdiviser de manière régulière l'intervalle $[a;b]$ en $(n-1)$ intervalles de même longueur.

`L2 = np.arange(a,b,p)` .....
`L3` contient un tableau contenant des valeurs appartenant à l'intervalle $[a;b]$: p représente le pas donc l'écart entre deux valeurs consécutives du tableau.

Le module `matplotlib.pyplot` contient des outils permettant de tracer des graphiques.

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

Placer des points ou relier des points sur un graphique

`plt.plot(a,b,'o')` .....
génère le point de coordonnées $(a;b)$.

`plt.plot([x1,x2],[y1,y2],'o')` .....
génère les points de coordonnées $(x_1;y_1)$ et $(x_2;y_2)$.

`plt.plot([x1,x2],[y1,y2])` .....
génère la droite reliant les points de coordonnées $(x_1;y_1)$ et $(x_2;y_2)$.

`plt.plot(L1,L2)` .....
génère la ligne brisée reliant les n points définis par les listes de taille n L_1 et L_2 (abscisses et ordonnées).

`plt.show()` .....
permet de visualiser les points/lignes générés dans une fenêtre graphique.

Remarque : On retient que pour tracer un segment, il suffit de donner les coordonnées de deux points à relier.

Exemple : `plt.plot([-5,5],[0,0])` ..... Génère l'axe des abscisses sur l'intervalle [-5,5] puisque relie les points (-5,0) et (5,0)

Pour tracer une courbe, on construit donc en fait un nuage de points (reliés entre eux) : on commence toujours par générer une liste (ou un tableau) contenant des valeurs d'abscisses et un tableau contenant les ordonnées correspondantes.

Tracer la courbe représentant une fonction f

`X=np.linspace(a,b,n)`
`Y=[np.log(t) for t in X]` ou `Y=np.log(X)`
Y est une liste contenant les images par f des réels listés dans le tableau X.
`plt.plot(X,Y)`
`plt.show()`

Attention, pour « appliquer » une fonction mathématique à un tableau (X), il faut que celle-ci provienne du module `numpy` et non du module `math`.

On peut ajouter un troisième argument (optionnel) à la fonction `plot` pour préciser

- le symbole utilisé pour les points

  `'.'` : petits points `'h'` : hexagones `'x'` : croix x

  `'o'` : gros points `'+'` : croix + `'*'` : étoiles

- la couleur

  `'r'` : rouge `'g'` : vert `'m'` : magenta (rose foncé)

  `'b'` : bleu `'c'` : cyan (bleu clair) `'k'` : noir

- le type de ligne

  `'-'` : ligne pleine `'--'` : ligne discontinue (tirets) `'-.'` : ligne discontinue (alternance points/tirets)

- le nom que l'on souhaite donner à la courbe si on décide d'afficher une légende

  `label='courbe 1'` assigne le nom 'courbe 1' à la figure générée par l'instruction plot associée

On peut aussi ajouter du texte sur le graphique à une certaine position :

`plt.text(x,y,'blabla',color='b')`
insère le texte « blabla » (en bleu) au niveau du point de coordonnées (x, y).

Il peut enfin être agréable de modifier les paramètres de la fenêtre graphique ou d'ajouter une légende.

Quelques options d'habillage du graphique

`plt.axis('equal')` ou `plt.axis([xmin,xmax,ymin,ymax])`
rend le repère orthonormé ou fixe les bornes des axes.
`plt.grid()`
affiche un quadrillage (peut faciliter les lectures graphiques).
`plt.xlabel('blabla')`
affiche « blabla » sous l'axe des abscisses.
`plt.ylabel('blabla')`
affiche « blabla » à côté de l'axe des ordonnées.
`plt.title('blabla')`
affiche « blabla » comme titre au-dessus du graphique.
`plt.legend()`
genère une légende à partir des étiquettes précédemment définies