Représentations graphiques avec Python™

Dans ce document, nous utilisons les bibliothèques **numpy** et **matpltotlib.pyplot** : la première permet de faire des calculs numériques et des tableaux de nombres ou « array » ; la seconde regroupe des outils permettant de tracer des graphiques.

Tracer un graphique cartésien avec Python™

- Importation des bibliothèques :

```
import matplotlib.pyplot as plt        #Importe la bibliothèque pour tracer des graphiques
import numpy as np                     #Importe la bibliothèque pour faire divers calculs
```

- Entrée des données expérimentales dans un tableau ou array dans le langage python. Grâce à la bibliothèque **numpy** on crée un **np.array** pour chacune des grandeurs (abscisses et ordonnées).

Par exemple, si l'on souhaite tracer l'évolution d'une grandeur y en fonction d'une grandeur x :

```
x = np.array([rentrer les valeurs de x séparées par des virgules])
#contient les valeurs x auxquelles ont été mesurées celles de y
y = np.array([rentrer les valeurs de y séparées par des virgules])
#contient les valeurs de y correspondantes
```

Il est important de rentrer le même nombre de valeurs de x et de y et dans le bon ordre.

```
plt.figure(figsize=(12,10))     #définition de la taille de la figure
plt.plot(x,y,'b+-',label='fonction')     #Représente y en fonction de x avec des croix bleues reliées de manière continue avec une étiquette pour la courbe
plt.xlabel('x')          #Légende de l'axe des abscisses
plt.ylabel('x')          #Légende de l'axe des ordonnées
plt.title('titre')       #Titre du graphe
plt.legend()             #Affiche l'étiquette de la courbe
plt.grid()               #Affiche le quadrillage
plt.show()               #Affiche le graphique
```

In sérer un curseur

- Nous utilisons ici le widget Cursor de la bibliothèque **matpltotlib.widgets**

```
from matplotlib.widgets import Cursor        #Importe la fonction pour insérer le curseur

cursor = Cursor(plt.gca(), color="black", lw=1)
# définition du curseur (ligne à insérer avant plt.show()
```

Tracer une dérivée avec Python™

- Calcul des grandeurs x_sub et der_y(abscisses et ordonnées).

```
def derivee(x,y):
    return (y[2:]-y[:-2])/(x[2:]-x[:-2])     # Définition de la fonction dérivée
der_y = derivee(x, y)  # Calcul de la fonction dérivée de y
x_sub = x[1:-1] # on redimensionne les valeurs de x pour qu'elles soient de même taille que der_y
```

- Affichage courbe dérivée

```
plt.xlabel("x") #Légende de l'axe des abscisses
plt.ylabel("dérivée") #Légende de l'axe des ordonnées
plt.plot(x_sub, der_y, 'b-',label="dérivée") #Représente la dérivée en fonction de V en bleu de manière continue avec une étiquette pour la courbe
plt.title("évolution de la dérivée") #Titre du graphe
plt.legend()#Affiche l'étiquette de la courbe
plt.grid()                          #Affiche le quadrillage
plt.show()                          #Affiche le graphique
```

Lisser une courbe avec Python™

Nous utilisons ici la fonction make_interp_spline de la bibliothèque **scipy.interpolate**.

```
from scipy.interpolate import make_interp_spline   #Importe la fonction pour faire le lissage
x_new = np.linspace(min(x),max(x),130)
#Définit le nombre de point (130) à tracer pour x
f = make_interp_spline(x,y,k=2)                    #Définit la fonction de lissage d'ordre k = 2
y_smooth = f(x_new)                                #Attribution des ordonnées aux abscisses
plt.plot(x_new,y_smooth,'b-')                      #Représente la fonction lissée
plt.xlabel('x')              #Légende de l'axe des abscisses
plt.ylabel('x')              #Légende de l'axe des ordonnées
plt.title('titre')                  #Titre du graphe
plt.legend()                                       #Affiche l'étiquette de la courbe
plt.grid()                                         #Affiche le quadrillage
plt.show()                                         #Affiche le graphique
```

Trouver le maximum d'une courbe avec Python™

Il faut chercher la valeur maximale de y_max dans le vecteur représentant l'ordonnée, trouver l'indice n_max de ce maximum, attribuer l'abscisse de ce maximum x_max de même indice n_max.

```
y_max=np.amax(y)                    #Donne le maximum d'un vecteur
n_max=np.argmax(y)                  #Donne l'indice d'un élément d'un vecteur
x_max=x_new[n_max]                  #Attribue la valeur d'un élément du vecteur
print('x_max = ',x_max,' et y_max = ',y_max)   #Affiche le résultat
```

Faire une régression linéaire avec Python™

Il y a plusieurs possibilités pour faire une régression linéaire. Nous présentons ici le script permettent de faire une régression linéaire avec la bibliothèque **numpy**.

```
#Renseigner les grandeurs x et y dans des array au préalable.

#Rajout des barres d'incertitude sur la grandeur portée en y

u_y='valeur à fixer'                #Incertitude-type sur la grandeur portée en ordonnée
plt.errorbar(x,y,yerr=u_y,fmt='go')
#Tracé du nuage de points en vert avec les barres d'incertitude en y.

#Régression linéaire avec np.polyfit

p=np.polyfit(x,y,1)
#Régression linéaire de y en fonction de x (équation y = p[0]*x+p[1])
plt.plot(x,np.polyval(p,x))         #Tracé de la droite de régression
plt.xlabel('x')   #Légende de l'axe des abscisses
plt.ylabel('x')   #Légende de l'axe des ordonnées
plt.title('titre')          #Titre du graphe
plt.legend()                        #Affiche l'étiquette de la courbe
plt.grid()                          #Affiche le quadrillage
plt.show()                          #Affiche le graphique
print('pente =',p[0])               #Affiche la pente de la droite
print('y0 =',p[1])                  #Affiche l'ordonnée à l'origine
```

Tracer un histogramme et calculer la moyenne, l'écart type et l'incertitude avec Python™

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

#Renseigner la grandeur x dans un array au préalable.
x=np.array([1.847e-5,1.841e-5,1.864e-5,1.874e-5,1.876e-5,1.871e-5,1.871e-5,1.862e-5,1.860e-5,1.856e-5])
n=len(x)        #Détermine le nombre de valeur du array

# Tracé histogramme
plt.hist(x, bins='rice', histtype = 'step')
plt.show()

# Calcul de la moyenne, de l'écart-type et de l'incertitude-type sur la moyenne
moy=np.mean(x)              #Calcule la moyenne des valeurs du array
print('Moyenne = ', moy)    #Affiche la moyenne des valeurs du array
std = np.std(x,ddof=1)      #Calcule l'écart-type des valeurs du array
print('Écart type = ', std)     # Affiche l'écart-type des valeurs du array
print('Incertitude-type = ', std/np.sqrt(n))    # Calcul et affiche l'écart-type sur la moyenne
```