Exploitation de données

I Préparation

I.1 Bibliothèques indispensables

```
import numpy as np
from scipy.optimize import curve_fit

import matplotlib.pyplot as plt
```

I.2 Gestion des incertitudes

Cette bibliothèque n'est pas toujours installée, on pourra s'en passer.

```
from uncertainties import ufloat
from uncertainties.umath import *  # sin(), etc.
```

I.3 Graphique

```
%matplotlib notebook
```

La ligne précédente ne doit apparaître que dans les notebooks `Jupyter`, pas dans un fichier python.

Toutes les courbes seront affichées sur le même graphique.

```
fig,ax = plt.subplots()
```

II Saisie des données

On entre les mesures sous la forme $[x, \Delta_x, y, \Delta_y]$.

```
donnees = np.array([[.5,.1,1.255,.1],
                    [.387,.15,1.25,.15],
                    [.24,.07,1.189,.07],
                    [.136,.2,1.124,.2],
                    [.04,.01,.783,.05],
                    [.011,0.005,.402,0.05]])
```

On extrait les abscisses, les ordonnées, les incertitudes, les valeurs extrêmes, le nombre de données

```
n = len(donnees)

x = donnees[:,0] #abscisses des points exp
y = donnees[:,2] #ordonnées des points exp
Deltax =  (donnees[:,1])#incertitudes-types sur x
Deltay = (donnees[:,3]) #incertitudes-types sur y

xmin = np.amin(donnees[:,0])
xmax = np.amax(donnees[:,0])
ymin = np.amin(donnees[:,2])
ymax = np.amax(donnees[:,2])
```

III Tracé de la courbe

- Ajuster les chaînes `abscisse`, `ordonnée`, `titre`, et éventuellement la position de la légende `loc`
- Choisir le style de courbe par la chaîne `fmt` : ici `bo` utilisera des cercles bleus. D'autres exemples sont disponibles sur les cheatsheets matplotlib

```
ax.cla()
ax.set_xlabel('abscisse')
ax.set_ylabel('ordonnée')
```

```
ax.set_title('titre')
ax.errorbar(x,y,yerr=Deltay,xerr=Deltax,fmt='bo ',label='données',capsize=3)
ax.legend(loc='best')
```

IV Ajustement numérique

IV.1 Calcul

On définit la fonction d'ajustement, `p` est un array contenant les paramètres.

```
def func(x, a, b):
  # a et b seront les paramètres à ajuster
  return a * x / (b + x)
```

L'array `p0`, facultatif, contient les valeurs initiales des paramètres pour l'ajustement. Il est possible, si les données sont très proches du modèle, que l'ajustement réussisse sans valeurs pertinentes pour `p0`.

```
estimations_initiales = ([1.2, 0.03]) #qu'on peut omettre
```

La fonction `curve_fit` recherche les valeurs optimales de `p`, stockées dans l'array `parametres`. Leurs incertitudes `std_parametres` sont calculées à partir de la matrice de covariance `pcov`

```
parametres, pcov = curve_fit(func, x, y, p0=estimations_initiales,sigma=Deltay)
std_parametres = np.sqrt(np.diag(pcov))
nombre_params= len(parametres) # nombre de paramètres
```

Si la bibliothèque `uncertainties` est disponible, on rassemble les valeurs des `parametres` et leurs incertitudes dans `resultats`.

```
resultats = [ufloat(parametres[i],std_parametres[i]) for i in range(nombre_params)]
```

IV.2 Affichage des résultats

- Sur la console

```
for i in range(nombre_params):
  print('p{0} = {1:.3e}'.format(i, resultats[i]))
```

- On prépare la légende de la figure

```
legende = ''
for i in range(nombre_params):
    legende+='p{0} = {1:.3e}\n'.format(i, resultats[i])
```

- On rajoute l'ajustement à la courbe

```
xfit = np.linspace(xmin,xmax)
yfit = func(xfit, *parametres[:])
ax.plot(xfit,yfit,'r',label='fit')
ax.text(.9,.1,legende,transform=ax.transAxes,horizontalalignment='right', verticalalignment
```

- On peut sauvegarder directement le graphique depuis sa «fenêtre» ou en utilisant la ligne suivante :

```
# fig.savefig('courbe.png')
```