L1-S2 : UE Méthodes numériques

SEANCE 7

Ajustement de données

30 Mars 2022

Introduction

En science, on est souvent amené à comparer des données expérimentales à un modèle mathématique.

- On mesure par exemple dans une expérience deux grandeurs x et y.
- Les résultats des mesures notés x_i, y_i sont entachés d'incertitudes. Par ailleurs il y a des fluctuations (aléatoires ?) dans l'expérience que l'on maîtrise mal.
- On voudrait savoir si un lien (une loi) existe entre les mesures de y et les mesures de x.

Introduction

On peut alors se poser les questions suivantes :

- Les grandeurs x et y sont-elles indépendantes ?
- Si non, quelle relation mathématique modélise le mieux le nuage de points (x_i, y_i) ? Ce modèle permettra de prévoir la valeur de y pour une valeur quelconque de x.
- Un travail théorique propose que la relation entre x et y est de la forme $y = f(x)$. Est-ce bien le cas ?
- On sait que le lien entre x et y est bien de la forme $y = f(x)$ aux incertitudes près. On veut alors en déduire une **estimation (valeur et incertitude)** des paramètres intervenant dans la fonction f. Par exemple : si $f(x)$ est une fonction linéaire, on veut connaître le coefficient directeur.

Introduction

Rappels :

- moyenne de x : $\bar{x} = \frac{1}{n} \sum_i x_i$
- variance de x : $var(x) = \frac{1}{n} \sum_i (x_i - \bar{x})^2$
- écart-type de x : $\sigma_x = \sqrt{var(x)}$
- covariance de x et y : $cov(x, y) = \frac{1}{n} \sum_i (x_i - \bar{x})(y_i - \bar{y})$

Exemple simple

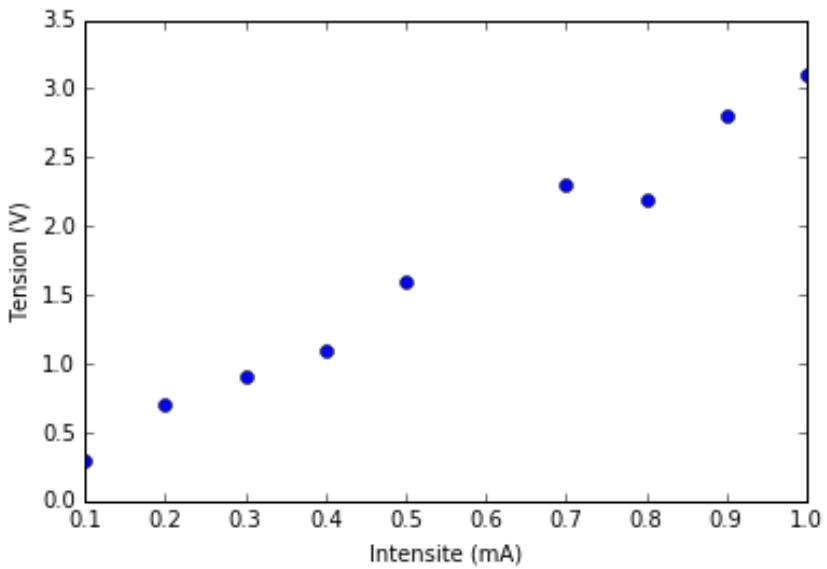
Au cours d'un TP on a mesuré avec un voltmètre la tension U aux bornes d'un dipôle, ainsi que l'intensité I le traversant. Le dispositif expérimental permettait de faire varier l'intensité. On a obtenu le tableau de valeurs suivant.

I(mA)	0.1	0.2	0.3	0.4	0.5	0.7	0.8	0.9	1
U(V)	0.3	0.7	0.9	1.1	1.6	2.3	2.2	2.8	3.1

Exemple simple

On commence par représenter sur un graphique U en fonction de I.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.8, 0.9, 1 ])
5 y = np.array([0.3, 0.7, 0.9, 1.1, 1.6, 2.3, 2.2, 2.8, 3.1 ])
6
7 plt.plot(x,y,'o')
8 plt.xlabel('Intensite (mA)')
9 plt.ylabel('Tension (V)')
```



Exemple simple

Visuellement, les points semblent se répartir autour d'une droite d'équation $U = aI + b$.

- Nous formulons ainsi l'hypothèse que la relation entre U et I est $U = aI + b$ à des fluctuations aléatoires près. Ces fluctuations sont reliées aux incertitudes sur les mesures.
- Quelle est alors la "meilleure" droite modélisant les points expérimentaux ?
- **C'est celle qui minimise la somme des différences entre les points expérimentaux et les points de la droite modèle.**

Méthode des moindres carrés

Si f est de la forme :

$$f(x) = ax + b \tag{1}$$

On cherche les valeurs de a et b qui minimisent la somme χ^2 des écarts quadratiques entre points expérimentaux y_i et valeurs données par le modèle $f(x_i)$:

$$\chi^2 = \sum_i (y_i - f(x_i))^2 = \sum_i (y_i - ax_i - b)^2 = \sum_i r_i^2(x_i) \tag{2}$$

$r_i(x_i)$ est appelé résidu. χ^2 est appelé "Chi deux" ou "Chi carré" : c'est la somme des résidus au carré.

Recherche de l'optimum : régression linéaire

Les valeurs de a et b qui minimisent le χ^2 satisfont les équations :

$$\frac{\partial \chi^2}{\partial b} = \sum_i -2(y_i - ax_i - b) = 0$$

$$\frac{\partial \chi^2}{\partial a} = \sum_i -2x_i(y_i - ax_i - b) = 0$$

En divisant par n la première équation, on trouve en terme des $\bar{x}$ et $\bar{y}$:

$$\bar{y} - a\bar{x} = b$$

$$\sum_i -2x_i[y_i - \bar{y} + a(\bar{x} - x_i)] = 0$$

Si on remarque que $\sum_i \bar{x}(\bar{x} - x_i)$ et $\sum_i \bar{x}(y_i - \bar{y}) = 0$, on trouve

$$\sum_i x_i(y_i - \bar{y}) = a \sum_i x_i(\bar{x} - x_i)$$

$$\sum_i (x_i - \bar{x})(y_i - \bar{y}) = a \sum_i (\bar{x} - x_i)^2$$

Régression linéaire : résultat

Et donc, dans le cadre d'une régression affine, le coefficient directeur a et l'ordonnée à l'origine b d'une fonction affine $f(x) = ax + b$ minimisant la fonction $\chi^2(a, b)$ sont :

$$\boxed{a = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x_i})^2} = \frac{cov(x, y)}{var(x)}}, \qquad \boxed{b = \bar{y} - a\bar{x}} \qquad (3)$$

Si le modèle mathématique utilisé est une combinaison linéaire de paramètres (par exemple $f(x) = ax + b$), la régression linéaire est une technique permettant d'obtenir une formule analytique pour chacun des paramètres.

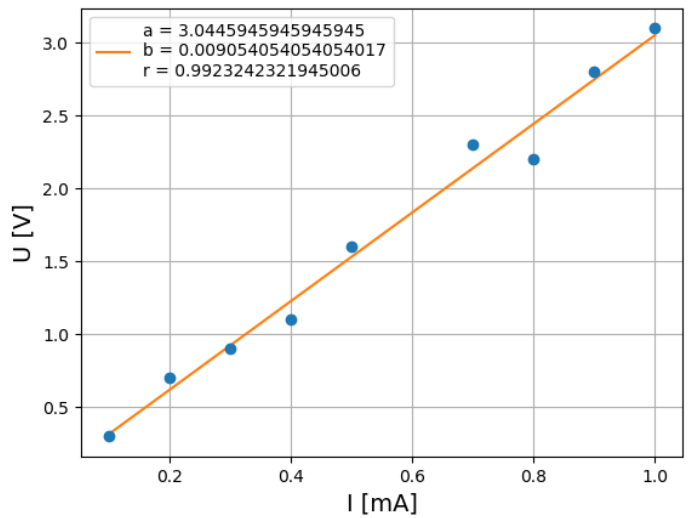
Remarque 1 : on voit que si $cov(x, y)$=0, le coefficient directeur a est nul. Les grandeurs y et x sont alors linéairement indépendantes.

Remarque 2 : estimer un paramètre par une formule analytique est toujours beaucoup plus rapide que de l'estimer en minimisant un χ^2.

Régression linéaire : résultat

Avec Python : pour réaliser une régression linéaire sur un jeu de données, on peut utiliser la fonction `linregress` de la bibliothèque `scipy.stats`.

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from scipy import stats
4
5  x=np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.8, 0.9, 1])
6  y=np.array([0.3, 0.7, 0.9, 1.1, 1.6, 2.3, 2.2, 2.8, 3.1])
7
8  a,b,r,p,stderr = stats.linregress(x, y)
9
10 plt.figure()
11 plt.plot(x, y ,marker='o', linestyle='None')
12 plt.plot(x, a*x+b, label="a = "+str(a)+"\nb = "+str(b)+"\nr =
       "+str(r))
13 plt.xlabel('I [mA]', fontsize=14)
14 plt.ylabel('U [V]', fontsize=14)
15 plt.grid()
16 plt.legend()
17 plt.show()
```



Régression linéaire : résultat

Signification des paramètres de sortie de linregress `a, b, R, p,`

`stderr = stats.linregress(x,y)`

- a est le coefficient directeur de la régression linéaire
- b est l'ordonnée à l'origine de la régression linéaire
- $R = \frac{cov(f(x),y)}{\sqrt{var(f(x))var(y)}}$ est le coefficient de corrélation **R**. On l'utilise beaucoup sous la forme R^2. Quand $R = 0$, les variables x et y sont décorrélées.
- p donne la probabilité que la distribution ainsi obtenue soit le fruit du hasard (peu utilisé).
- *stderr* est l'incertitude sur le coefficient directeur.

Régression linéaire : résultat

Coefficient de détermination R^2
Le paramètre R^2 est défini de la manière suivante :

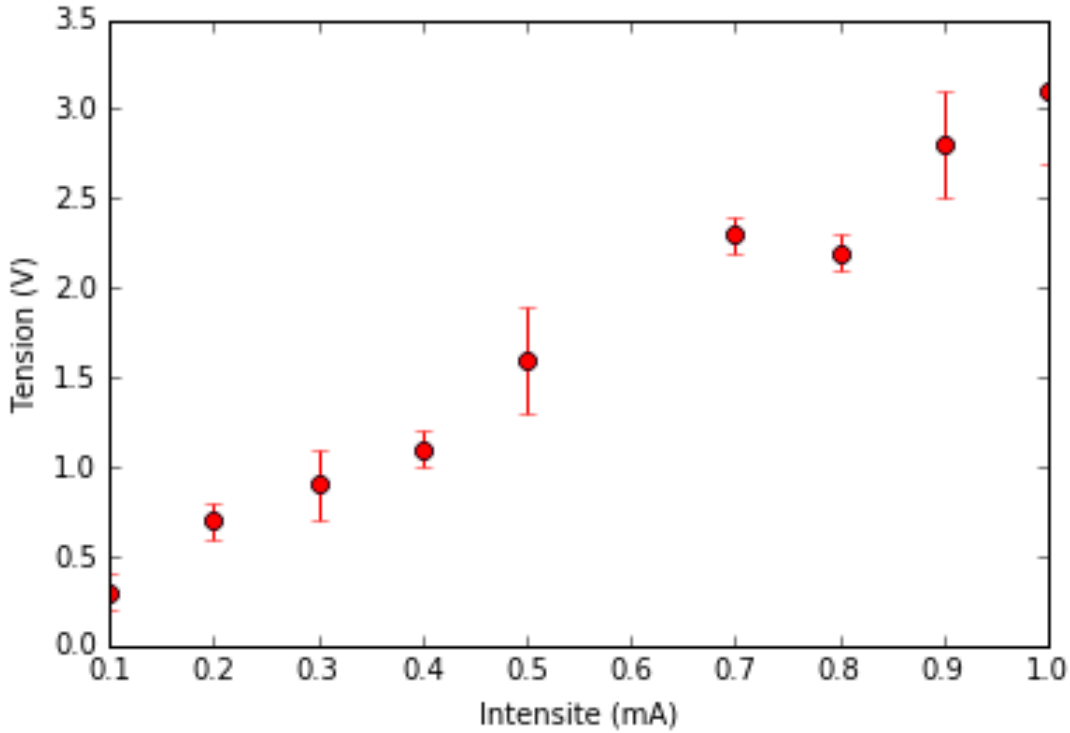
$$R^2 = 1 - \frac{\sum_i (y_i - f(x_i))^2}{\sum_i (y_i - \bar{y})^2} \tag{4}$$

Signification de R^2 :
Ce nombre compare la somme des résidus au carré à la variance de la série de valeurs y_i.
Ce nombre nous permet de répondre à la question : "dans quelle mesure les variations de y sont-elles explicables par les variations de x" ?

- R^2 est compris entre 0 et 1.
- S'il est proche de 0, cela signifie que y n'a aucun lien avec x : ce sont des variables indépendantes, x ne permet pas d'expliquer y.
- Plus il est proche de 1, plus les variations de y_i sont explicables par une loi de type $y_i = ax_i + b$.
- Plus il est proche de 1, plus notre régression linéaire est "bonne".

Prise en compte des incertitudes

Souvent il faut prendre en compte le fait que les mesures expérimentales y_i sont affectées d'une incertitude σ_i que l'on a pu estimer (incertitude de lecture du voltmètre, ...).



Que devient notre régression linéaire dans ces conditions ?

Prise en compte des incertitudes

Rappel

Le module matplotlib contient une fonction **errorbar()** pour tracer des points de mesures avec leurs **incertitudes**.

```
1  import numpy
2  import matplotlib.pyplot as plt
3
4  x=numpy.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.8, 0.9, 1])
5  y=numpy.array([0.3, 0.7, 0.9, 1.1, 1.6, 2.3, 2.2, 2.8, 3.1])
6  sigma_y=numpy.array([0.1, 0.1, 0.1, 0.2, 0.2, 0.1, 0.2, 0.2,
       0.3])
7
8  plt.figure()
9  plt.errorbar(x ,y, yerr=sigma_y, marker='o', color='r',
       linestyle='None')
10 plt.xlabel('x')
11 plt.ylabel('y')
12 plt.show()
```

Prise en compte des incertitudes

Prise en compte des incertitudes dans la régression linéaire.

- On cherche à nouveau les valeurs de a et b qui minimisent la somme des écarts entre points expérimentaux y_i et valeurs données par le modèle $f(x_i) = ax_i + b$.
- Toutefois les points associés à de fortes incertitudes seront moins pris en compte : on pondère leur contribution au χ^2 par un coefficient w_i d'autant plus faible que l'incertitude σ_i est grande.
- On choisit $w_i = \frac{1}{\sigma_i^2}$

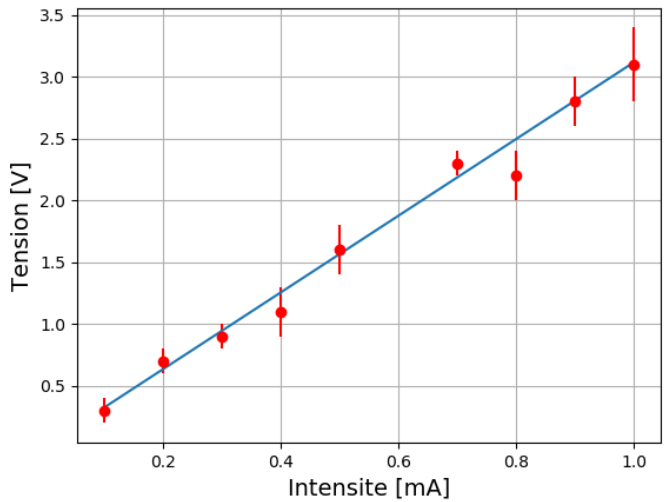
On cherche donc a et b qui minimisent la quantité suivante :

$$\chi^2 = \sum_i \frac{(y_i - f(x_i))^2}{\sigma_i^2} = \sum_i w_i r_i^2(x_i) \tag{5}$$

Prise en compte des incertitudes

La bibliothèque scipy.optimize comprend des fonctions d'optimisation que l'on peut utiliser pour ajuster des modèles quelconques à des données avec leurs incertitudes.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

def fit_func(x, a, b):
    return a*x + b

x = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.8, 0.9, 1 ])
y = np.array([0.3, 0.7, 0.9, 1.1, 1.6, 2.3, 2.2, 2.8, 3.1 ])
sigma_y = np.array([0.1, 0.1, 0.1, 0.2, 0.2, 0.1, 0.2, 0.2,
      0.3])

a0=0.0
b0=0.0
p0=[a0,b0] #initialisation de la regression

params,cov = curve_fit(fit_func, x, y, p0, sigma_y)

[a, b] = params

plt.figure()
plt.errorbar(x, y, yerr=sigma_y, marker='o', color='r',
      linestyle='None')
plt.plot(x, fit_func(x,a,b), label="a="+str(a)+"\nb="+str(b))
plt.xlabel('Intensite [mA]', fontsize=14)
plt.ylabel('Tension [V]', fontsize=14)
plt.grid()
plt.legend()
plt.show()
```



Prise en compte des incertitudes

Utilisation de la fonction `curve_fit`

```
1 params,cov=curve_fit(fit_func,x,y,p0=None,sigma=None,...)
```

Paramètres d'entrée :

- `fit_func` est le nom de la fonction servant de modèle.
- `x,y` sont des tableaux `np.array` contenant les valeurs expérimentales x_i et y_i
- p0 est un tableau contenant les valeurs initiales des paramètres de la fonction (dans l'exemple ci-dessus a et b) pour amorcer l'algorithme itératif de moindres carrés (optionnel mais parfois nécessaire pour des régressions compliquées).
- `sigma` est un tableau `np.array` contenant les valeurs des incertitudes σ_{y_i} sur les valeurs y_i (argument optionnel)

Prise en compte des incertitudes

Utilisation de la fonction `curve_fit`

```
1 params,cov=curve_fit(fit_func,x,y,p0=None,sigma=None,...)
```

Paramètres de sortie :

- `params` est un tableau 1D contenant les résultats de la minimisation des moindres carrés : ici $params[0] = a$ et $params[1] = b$.
- `cov` est un tableau 2D. C'est la matrice de covariance. **Ses termes diagonaux contiennent les incertitudes au carré sur les paramètres de la régression :** $cov[0,0] = \sigma_a^2$ et $cov[1,1] = \sigma_b^2$

Remarque : Il existe également la méthode `np.polyfit` qui ajuste des fonctions polynomiales par régression linéaire en prenant en compte les incertitudes.

Le modèle décrit-il suffisamment les données ?

Écarts modèle-points expérimentaux – Notion de χ^2 réduit
Après minimisation des moindres carrés, les résidus ne sont pas complètement nuls. Si la régression a convergé et que les incertitudes ont été évaluées "honnêtement", résidus et incertitudes doivent être du même ordre de grandeur, et la moyenne des résidus doit être nulle.

Pour évaluer si le modèle s'ajuste bien aux données, étant données les incertitudes de mesure, on calcule **le χ^2 réduit** :

$$\chi^2_\nu = \frac{1}{\nu} \sum_i \frac{(y_i - f(x_i))^2}{\sigma_i^2} = \frac{1}{\nu} \sum_i w_i r_i^2(x_i) \tag{6}$$

où $\nu = N - k$, N le nombre de points expérimentaux et k le nombre de paramètres de la fonction modèle ($k = 2$ pour une droite) :

- si la valeur de χ^2_ν est beaucoup plus grande que 1, le modèle n'est pas pertinent ou on a sous-estimé les incertitudes
- si χ^2_ν <0.1, les incertitudes ont été surestimées.

Généralisation : ajustement par un modèle non-linéaire quelconque

Tout ce que nous avons vu jusqu'à présent pour un modèle linéaire se généralise au cas d'un modèle $y = f(x)$ quelconque.

- Notre modèle est de la forme : $y = f(x, \beta_1, \beta_2...)$, où $\beta_1, \beta_2...$ sont des paramètres (dans le cas linéaire il s'agissait de $\beta_1 = a$ et $\beta_2 = b$).
- On cherche les valeurs des paramètres $\beta_1, \beta_2...$ qui minimisent la somme des écarts entre points expérimentaux y_i et valeurs données par le modèle $f(x_i, \beta_1, \beta_2...)$.
- On pondère la contribution des points par un coefficient $w_i = \frac{1}{\sigma_i^2}$

On cherche donc les valeurs des paramètres $\beta_1, \beta_2...$ qui minimisent :

$$\chi^2 = \sum_i \frac{(y_i - f(x_i, \beta_1, \beta_2...))^2}{\sigma_i^2} \tag{7}$$

Généralisation : ajustement par un modèle non-linéaire quelconque

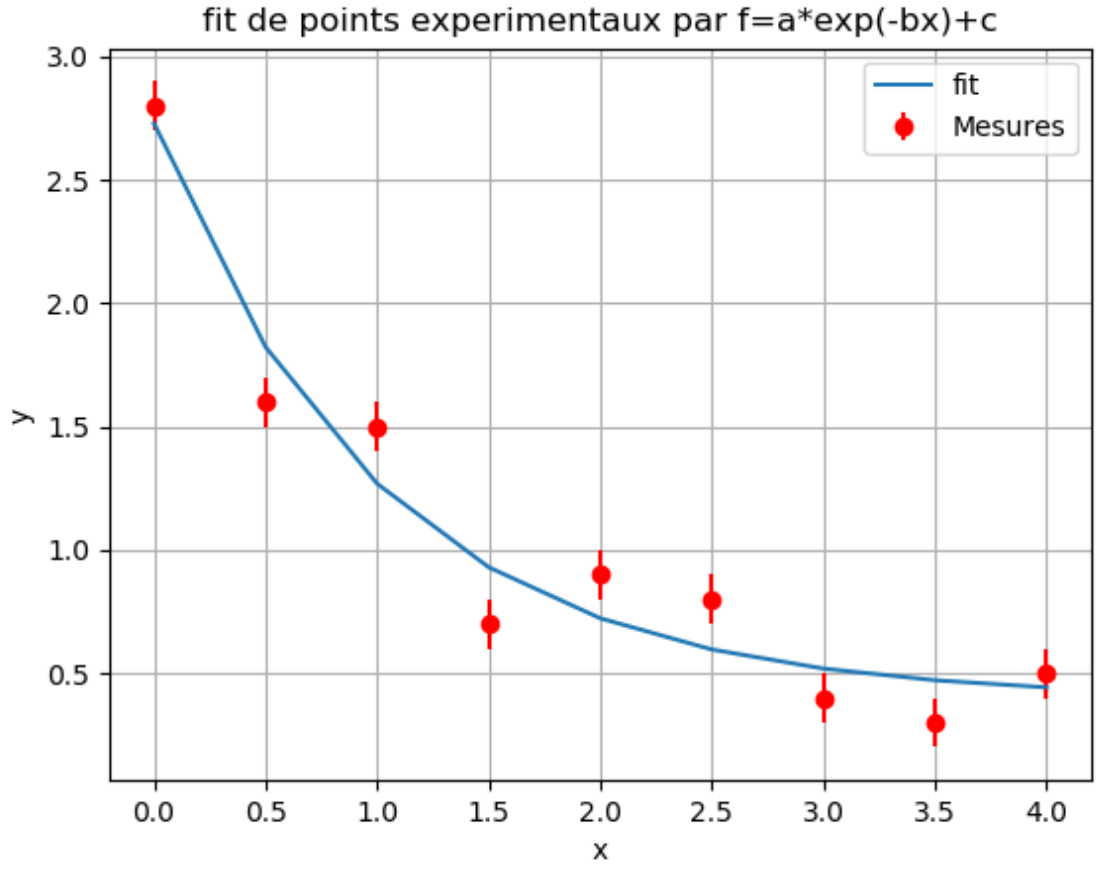
Exemple : ajustement de points expérimentaux par une fonction : $f(x) = ae^{-bx} + c$

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit


def func(x, a, b, c):
    return a * np.exp(-b * x) + c


xdata = np.array([ 0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0])
ydata = np.array([2.8, 1.6, 1.5, 0.7, 0.9, 0.8, 0.4, 0.3, 0.5])
sigma = np.array([0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1])
param0 = np.array([0.0, 0.0, 0.0]) #initial guess

popt,cov = curve_fit(func, xdata, ydata, param0, sigma)

print('\tValeur de a :',round(popt[0],1),'+/-:',round(np.sqrt(cov[0,0]),1),' unite')
print('\tValeur de b :',round(popt[1],1),'+/-:',round(np.sqrt(cov[1,1]),1),' unite')
print('\tValeur de c :',round(popt[2],1),'+/-:',round(np.sqrt(cov[2,2]),1),' unite')

plt.figure()
plt.errorbar(xdata,ydata,yerr=sigma,marker='o',color='r',label='Mesures',linestyle='None')
plt.plot(xdata,func(xdata,popt[0],popt[1],popt[2]),label='fit')
plt.xlabel('x')
plt.ylabel('y')
plt.title('fit de points experimentaux par f=a*exp(-bx)+c')
plt.legend()
plt.grid()
plt.show()
```

Généralisation : ajustement par un modèle non-linéaire quelconque

Out[0]:

```
Valeur de a : 2.3 +/-: 0.2  unite
Valeur de b : 1.0 +/-: 0.3  unite
Valeur de c : 0.4 +/-: 0.2  unite
```



À vos TPs !

1. Ouvrir un terminal :
   - soit sur https ://jupyterhub.ijclab.in2p3.fr
   - soit sur un ordinateur du 336
2. Télécharger la séance d'aujourd'hui :

```
methnum fetch L1/Seance7 TONGROUPE
```

   en remplaçant TONGROUPE par ton numéro de groupe.
3. Sur un ordinateur du bâtiment 336 uniquement, lancer le jupyter :

```
methnum jupyter notebook
```

4. Pour soumettre la séance, dans le terminal taper :

```
methnum submit L1/Seance7 TONGROUPE
```