Ajustement de courbes

nov. 26, 2018

1 Ajustement d'une courbe

On considère une courbe $y = f(x, \mathbf{p})$ où les $\mathbf{p} = p_1, ..., p_n$ sont des paramètres inconnus. Experimentalement, N mesures ont été effectuées pour des abscisses $x_1, ..., x_N$. Les valeurs mesurées y_i sont entâchées d'un bruit que l'on suppose gaussien de variance σ et non biaisé :

$$y_i = f(x_i, \mathbf{p}) + \epsilon_i$$

L'objectif est de trouver un estimateur des paramètres $\mathbf{p}$ à partir des données (x_i, y_i).

Pour cela, on peut appliquer le principe du maximum de vraissemblance, et on obtient que

$$\hat{\mathbf{p}} = \mathrm{argmin} \sum_{i=1}^{i=N} (y_i - f(x_i, \mathbf{p}))^2$$

Dans le cas général, les différents paramètres $\mathbf{p} = p_1, ..., p_n$ sont corrélés. La matrice de corrélation est donnée par :

$$\mathrm{cov}(\mathbf{p}, \mathbf{p}) = \frac{\sigma^2}{2M}$$

où la matrice M est donnée par :

$$M_{k,l} = \sum_i \left.\frac{\partial f}{\partial p_k}\right|_{x_i} \left.\frac{\partial f}{\partial p_l}\right|_{x_i}$$

2 curve_fit

On va utiliser la fonction `curve_fit` du package `scipy.optimize`. Cette fonction permet aussi d'obtenir l'incertitude des paramètres sous forme d'une matrice de corrélation. Cette fonction s'utilise de la façon suivante

```
p_opt, cor_mat = curve_fit(fonction_de_fit, data_x, data_y, p_ini)
```

où

— `fonction_de_fit(x,p1,p2,...,pn)` est la fonction de fit. Les variables p1, .., pn sont les paramètres de la fonction de fit.

— `data_x` et `data_y` sont les points de mesure.

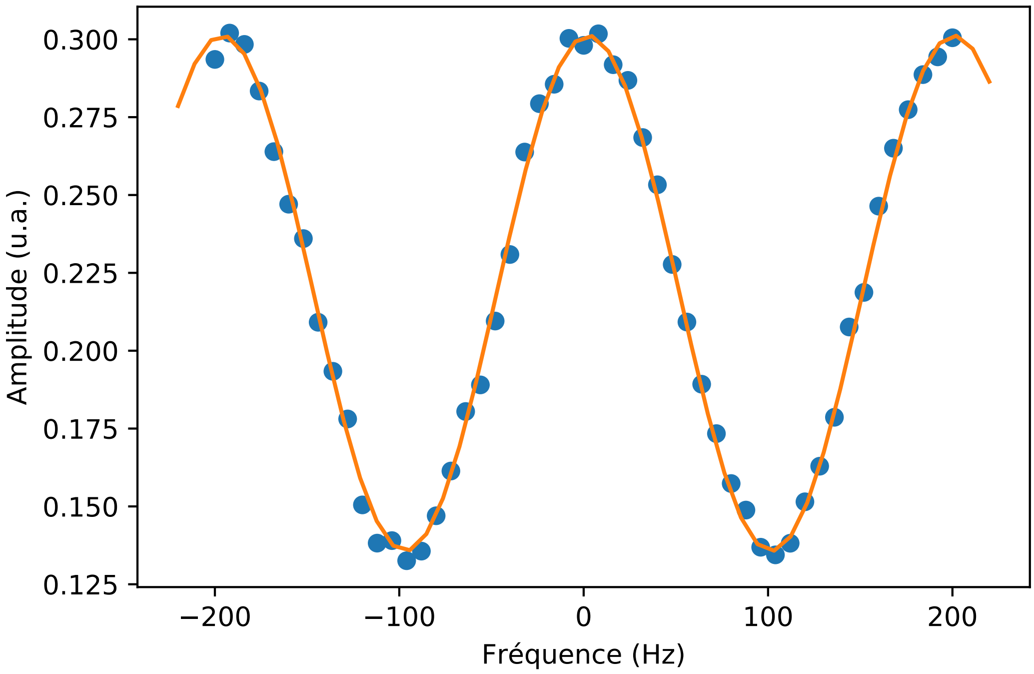
— `p_ini` sont les paramètres initiaux (sous forme d'une liste/tuple/array).

— `p_opt` seront les paramètres optimaux

— `cor_mat` est la matrice de correlation entre les paramètres.

3 Fit de franges d'interférence

On souhaite ajuster les franges d'un interféromètre atomique. Les données sont dans le fichier `data/fit_sinus.dat`. La première colonne du fichiers (axe x) représente une fréquence en Hz. La seconde colonne représente la population mesurée pour une fréquence donnée. L'objectif est de trouver la position de la frange centrale.



On ajustera par une fonction cosinus avec une amplitude, un décalage vertical, une position centrale et une largeur ajustable.

1. Écrivez la fonction de fit qui dépend des paramètres ci dessus. On appellera `frange(x, ...)`. Tracez la courbe pour x entre ± 220 Hz. On prendra un interfrange de 200 Hz.
2. Chargez et tracez les données. On représentera les données par des points (`plot(..., ..., 'o')`).
3. Calculez les paramètres optimaux. Quelle est la position la frange centrale ? Représentez les points et la courbe comme sur la figure ci-dessus.
4. Quelle est l'incertitude sur la position de la frange centrale ?

4 Corrélation entre paramètres

On simule un jeu de données

```
np.random.seed(0)
N = 100
```

```
x = linspace(2000, 2018, N)
y = arange(N)*0.2+45+np.random.normal(size=N)
```

- Tracez et ajustez les données par une une droite $y = ax + b$.
- Quel est l'incertitude sur b ? Qu'en pensez-vous ?
- Calculez la valeur et l'incertitude de votre fit en $x = 2010$.
- Trouvez une fonction de fit plus pertinente pour ce problème.

5 Fit d'une image

Le fichier `data/double_star` contient l'image de 64 par 64 pixels d'une image d'une étoile double. L'objectif de cette partie est d'ajuster cette image par la somme de deux Gaussienne afin d'en déterminer la distance.

L'ajustement d'une image procède de la même manière que l'ajustement d'une courbe. Le tableau xdata sera alors un tableau de taille N par 2 correspondant aux coordonnées des N points de l'image ($N = 64^2$ dans notre exemple) et ydata un tableau de taille N.

Voici l'exemple d'un fit par une gaussienne simple

```
ny, nx = image.shape
X,Y = meshgrid(range(nx), range(ny))
xdata = array([X.flatten(), Y.flatten()]).transpose()

def gauss(xdata, amplitude, center_x, center_y, diameter):
    x = xdata[:,0]
    y = xdata[:,1]
    return amplitude*exp(-((x-center_x)**2 + (y-center_y))/
 ↪diameter**2)

popt, pcov = curve_fit(gauss, xdata, image.flatten(), p0)
```

- Adaptez cet exemple afin de faire un fit par deux gaussiennes

6 Fit par une loi de Poisson

On effectue une expérience de spectroscopie en régime de comptage de photons. La forme de raie est une Lorentzienne :

$$\frac{1}{1 + ((f - f_0)/\Gamma)^2}$$

On va prendre $f_0 = 0$ et $\Gamma = 1$.

Le signal est très faible et on mesure environ 5 coups par seconde au sommet de la courbe. La statistique est une loi de Poisson dont le paramètre est donnée par la forme de raie. La probabilité d'avoir k photons à la fréquence f est donné par :

$$P(k, f) = \frac{\lambda(f)^k}{k!} e^{-\lambda(f)}$$

avec

$$\lambda(f) = \frac{\alpha}{1 + ((f - f_0)/\Gamma)^2}$$

Les mesures ont été prises pour des fréquences f allant de -3Γ à 3Γ avec un pas de $\Gamma/10$.

1. On simule un jeu de donnée à l'aide de la fonction suivante

```python
Gamma = 1
f_0 =  0
amplitude = 5.

f_mesure = linspace(-3, 3, 51)

def lorentz(f, amplitude=amplitude, f_0=f_0, Gamma=Gamma):
    return amplitude/(1+((f-f_0)/Gamma)**2)

def simulate_data():
    return np.random.poisson(lam=lorentz(f_mesure))
```

2. Effectuez un fit par la méthode des moindres carrés.
3. Effectuez un fit en maximisant la fonction de vraissemblance. On utilisera la fonction `minimise` de `scipy.optimize` et la fonction `scipy.stats.poisson.pmf` pour calculer la fonction de vraissemblance). Lisez la documentation de `scipy.stats.poisson.pmf`. Il est nécéssaire de mettre une limite à la fonction minimize car le paramètre λ doit être poisitif

```python
from scipy.optimize import minimize
from scipy.stats import poisson

def log_likelihood(parametres, x, y):
    amplitude, f_0, Gamma = parametres
    # Finissez la fonction en utilisant poisson.pmf

p_0 = np.array([4, .3, .7])
data = simulate_data()
out = minimize(log_likelihood, p_0, args=(f_mesure, data),
                bounds=[(0, None), (None, None), (None, None)])
print(out['x'][1])
```

4. Comparez la variance des deux estimateurs. La variance sera estimée à partir de la variance d'un grand nombre de réalisations (simulation + ajustement).