# Fiche méthode n° 4 — Optimisation : *curve fitting* avec python

## Principe

On dispose de deux listes de données X = [x1, x2, …, xn] et Y = [y1, y2, …, yn], représentant $n$ valeurs des variables $x$ et $y$.
On se donne une fonction $f$ de la variable $x$, dépendant de paramètres $p_1$, $p_2$, …, $p_k$.
L'optimisation consiste à chercher le jeu de valeurs des paramètres $p_i$ telle que la fonction $y = f(x, p_1, \ldots, p_k)$ s'approche « au mieux » des $n$ points donnés.

## Utilisation de curve_fit de la bibliothèque scipy

La bibliothèque scipy met à notre disposition un outil permettant cette optimisation.

```
from scipy.optimize import curve_fit
```

La syntaxe d'utilisation est

```
curve_fit(f, X, Y, p0 = [p1, ..., pk])
```

où

`f` est la fonction utilisée pour l'optimisation

`X` est la liste des données X = [x1, x2, …, xn]

`Y` est la liste des données Y = [y1, y2, …, yn]

[$p_1$, …, $p_k$ ] est la liste des estimations initiales des paramètres $p_1$, … $p_k$.

➤ on peut ne pas fournir d'estimation initiale des paramètres avec l'option `p0 = None`.

La fonction `curve_fit` retourne deux listes :

— la liste des paramètres $p_k$ qui optimise la fonction par rapport à l'ensemble des données ;

— un tableau qui donne la covariance de la liste des paramètres.

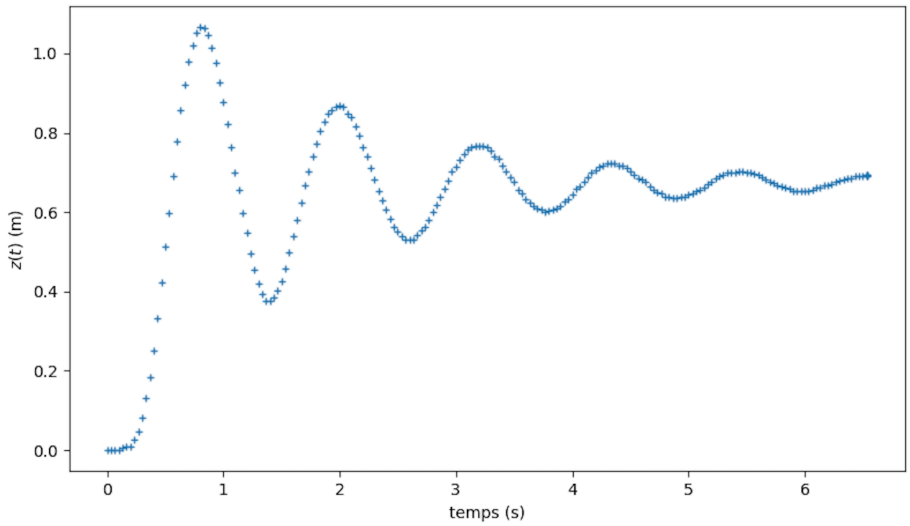
➤ Nous n'utiliserons que la première liste, que l'on appelle usuellement `popt`, la seconde étant appelée `pcov`.

La syntaxe d'appel est donc typiquement

```
popt, pcov = curve_fit(f, X, Y, p0 = [p1, ..., pk])
```

### Exemple

On dispose d'un enregistrement des oscillations amorties d'un ressort :



Les données sont enregistrées dans le fichier `pointage.csv`.
Les instants de pointage sont dans la première colonne, et les altitudes dans la troisième colonne (la seconde donne la position horizontale, qui ne nous intéresse pas ici).

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

fichier = open("pointage.csv","r")
donnees_brutes = fichier.readlines()
fichier.close()

donnees = []
for i in range(len(donnees_brutes)):
    donnees_brutes[i] = donnees_brutes[i][:-1]
    donnees.append(donnees_brutes[i].split(";"))

T,Z = [],[]
for i in range(2,len(donnees_brutes)):
    T.append(float(donnees[i][0].replace(",",".")))
    Z.append(float(donnees[i][2].replace(",",".")))

def modele(t,A,B,w,phi,tau):
    return A+B*np.cos(w*t+phi)*np.exp(-t/tau)

p,pcov = sp.optimize.curve_fit(modele,T,Z,p0 = None)

Z_fit = []
for t in T:
    Z_fit.append(modele(t,p[0],p[1],p[2],p[3],p[4]))

plt.close()
plt.plot(T,Z,linestyle='',marker='+',markersize=4)
plt.plot(T,Z_fit,linewidth=1)
plt.xlabel('temps (s)')
plt.ylabel('$z(t)$ (m)')
plt.show()
```

La liste `p` contient les valeurs optimisées des paramètres `A`, `B`, `w`, `phi`, `tau`.