Modélisation numérique en physique

Séquence 4 : Ajuster un modèle

Diapositives : Diata Traore

Cours et calepins : https://phys-mod.github.io/source/intro.html

Objectifs de la séquence :

- *Créer et manipuler*
- *les dictionnaires.*
- *l'objet DataFrame du module Pandas.*
- *les objets Series et Index du module pandas.*
- *Décrire et utiliser les opérateurs de comparaisons sur les booléens.*
- *Extraire et filtrer des tableaux Numpy et des DataFrame en utilisant des conditions.*
- *Ecrire la syntaxe et créer des structures conditionnelles et itératives (if, for et while)*
- *Ecrire la syntaxe d'une fonction Python, décrire le fonctionnement des arguments d'entrée et de sortie, et créer des fonctions simples.*
- *Utiliser l'entrée et la sortie standard, écrire et lire des fichiers avec un format défini.*

\+ Ajuster un modèle

Objectif #6 : utiliser l'entrée et la sortie standard, écrire et lire des fichiers avec un format défini

☆ Vérifier le contenu de son dossier de travail sous unix : ! ls

☆☆ Ouvrir le fichier essai.txt en mode lecture :

```
file = open( 'nom' , mode = 'r', encoding= 'uft-8')
```

☆☆ Ouvrir un fichier en mode écriture :

```
file = open( 'nom', mode = 'w', encoding= 'uft-8')
```

☆☆ Fermer un fichier :

```

```

☆☆ Ouvrir un fichier en mode écriture avec le *Gestionnaire de fichiers :*

```
with open('nom' ,mode ='w' , encoding='uft-8') as file  :
   …
```

Objectif #6 : utiliser l'entrée et la sortie standard, écrire et lire des fichiers avec un format défini

☆ Ajouter du texte à un fichier déjà existant :

```
with open('essai.txt', mode='a', encoding='uft-8') as file :
    file.write('Mon texte \n Retour à la ligne.")
```

☆ Ecraser un fichier avec un texte :

```
with open('essai.txt', mode='w', encoding='uft-8') as file :
    file.write('Mon texte \n Retour à la ligne.")
```

☆☆ Fonction permettant de lire tout ce qui est écrit au clavier (str) :

```
chaine = input("Entrer votre message avec input: \n")
```

Objectif #6 : utiliser l'entrée et la sortie standard, écrire et lire des fichiers avec un format défini

☆☆ Importer des données depuis un fichier .dat :

```
import numpy as np

data = np.loadtxt ('exoplanets.dat')
print(data[1:20][:])
```

☆☆ Importer des données en attribuant chaque colonne à un tableau 1D :

```
a, p, mp, me = np.loadtxt ('exoplanets.dat',unpack = True)
```

```
# demi-gd axe [UA]      période [j]     masse [m_jupiter]       masse étoile [m_solaire]
1.60044 672.1   11.1883 1.2
0.0761616       7.008151        0.00752338      1.2
0.519687        124.9144        0.0254263       1.2
0.736177        210.60697       0.217563        1.2
0.0530508       4.072   1.32811 1.2
1.25186 471.6   7.12332 1.17
0.996347        331.60059       0.937417        1.2
2.33328 1187.3  11.6809 1.191
0.0465021       3.35524 0.685857        1.191
0.113014        12.7204 0.0899033       1.19
0.034488        2.143634        0.984236        1.19
1.11234 392.6   1.2654  1.19
0.0702822       6.2385  0.0223552       1.19
0.0454854       3.24674 1.02109 1.19
2.60488 1405    4.7347  1.19
0.0499354       3.735433        0.508547        1.19
0.047858        3.509267        0.46004 1.187
```

exoplanets.dat

```
[[7.6161600e-02 7.0081510e+00 7.5233800e-03 1.2000000e+00]
 [5.1968700e-01 1.2491440e+02 2.5426300e-02 1.2000000e+00]
 [7.3617700e-01 2.1060697e+02 2.1756300e-01 1.2000000e+00]
 [5.3050800e-02 4.0720000e+00 1.3281100e+00 1.2000000e+00]
 [1.2518600e+00 4.7160000e+02 7.1233200e+00 1.1700000e+00]
 [9.9634700e-01 3.3160059e+02 9.3741700e-01 1.2000000e+00]
 [2.3332800e+00 1.1873000e+03 1.1680900e+01 1.1910000e+00]
 [4.6502100e-02 3.3552400e+00 6.8585700e-01 1.1910000e+00]
 [1.1301400e-01 1.2720400e+01 8.9903300e-02 1.1900000e+00]
 [3.4488000e-02 2.1436340e+00 9.8423600e-01 1.1900000e+00]
 [1.1123400e+00 3.9260000e+02 1.2654000e+00 1.1900000e+00]
 [7.0282200e-02 6.2385000e+00 2.2355200e-02 1.1900000e+00]
 [4.5485400e-02 3.2467400e+00 1.0210900e+00 1.1900000e+00]
 [2.6048800e+00 1.4050000e+03 4.7347000e+00 1.1900000e+00]
 [4.9935400e-02 3.7354330e+00 5.0854700e-01 1.1900000e+00]
 [4.7858000e-02 3.5092670e+00 4.6004000e-01 1.1870000e+00]
 [3.6515500e-02 2.3412127e+00 1.0910000e+00 1.1840000e+00]
 [4.3920800e-02 3.0925140e+00 9.0207500e-01 1.1810000e+00]
 [4.9943600e-02 3.7521000e+00 5.4279700e-01 1.1800000e+00]]
```

Objectif #6 : utiliser l'entrée et la sortie standard, écrire et lire des fichiers avec un format défini

☆☆ Sauvegarder un tableau Numpy dans un fichier de données :

```
np.savetxt ('exoplanets-SI.dat', np.c_ [a_SI, p_SI, mp_SI, me_SI ], fmt = '%10.3e')
```

```
# change units to SI
a_SI = a * au2m
p_SI = p * day2seconds
mp_SI = mp * M_jupiter
me_SI = me * M_sun
```

```
2.394e+11  5.807e+07  2.124e+28  2.387e+30
1.139e+10  6.055e+05  1.428e+25  2.387e+30
7.774e+10  1.079e+07  4.826e+25  2.387e+30
1.101e+11  1.820e+07  4.129e+26  2.387e+30
7.936e+09  3.518e+05  2.521e+27  2.387e+30
1.873e+11  4.075e+07  1.352e+28  2.327e+30
1.491e+11  2.865e+07  1.779e+27  2.387e+30
3.491e+11  1.026e+08  2.217e+28  2.369e+30
6.957e+09  2.899e+05  1.302e+27  2.369e+30
1.691e+10  1.099e+06  1.706e+26  2.367e+30
5.159e+09  1.852e+05  1.868e+27  2.367e+30
1.664e+11  3.392e+07  2.402e+27  2.367e+30
1.051e+10  5.390e+05  4.243e+25  2.367e+30
6.805e+09  2.805e+05  1.938e+27  2.367e+30
3.897e+11  1.214e+08  8.986e+27  2.367e+30
7.470e+09  3.227e+05  9.652e+26  2.367e+30
7.159e+09  3.032e+05  8.732e+26  2.361e+30
5.463e+09  2.023e+05  2.071e+27  2.355e+30
6.570e+09  2.672e+05  1.712e+27  2.349e+30
7.471e+09  3.242e+05  1.030e+27  2.347e+30
7.018e+09  2.951e+05  5.620e+26  2.347e+30
```

exoplanets-SI.dat

Objectif #6 : utiliser l'entrée et la sortie standard, écrire et lire des fichiers avec un format défini

☆☆ Sauvegarder un tableau Numpy dans un fichier de données :

```
np. … ('exoplanets-SI.dat', np. … [a_SI, p_SI, mp_SI, me_SI ], fmt = '%10.3e')
```

☆ Format des données : `fmt = '%10.3e'`

`10` = nombre total de caractères (signes compris)

`3` = nombre de chiffres après le point décimal

`e` = type d'affichage exponentiel

```
# change units to SI
a_SI = a * au2m
p_SI = p * day2seconds
mp_SI = mp * M_jupiter
me_SI = me * M_sun
```

```
2.394e+11  5.807e+07  2.124e+28  2.387e+30
1.139e+10  6.055e+05  1.428e+25  2.387e+30
7.774e+10  1.079e+07  4.826e+25  2.387e+30
1.101e+11  1.820e+07  4.129e+26  2.387e+30
7.936e+09  3.518e+05  2.521e+27  2.387e+30
1.873e+11  4.075e+07  1.352e+28  2.327e+30
1.491e+11  2.865e+07  1.779e+27  2.387e+30
3.491e+11  1.026e+08  2.217e+28  2.369e+30
6.957e+09  2.899e+05  1.302e+27  2.369e+30
1.691e+10  1.099e+06  1.706e+26  2.367e+30
5.159e+09  1.852e+05  1.868e+27  2.367e+30
1.664e+11  3.392e+07  2.402e+27  2.367e+30
1.051e+10  5.390e+05  4.243e+25  2.367e+30
6.805e+09  2.805e+05  1.938e+27  2.367e+30
3.897e+11  1.214e+08  8.986e+27  2.367e+30
7.470e+09  3.227e+05  9.652e+26  2.367e+30
7.159e+09  3.032e+05  8.732e+26  2.361e+30
5.463e+09  2.023e+05  2.071e+27  2.355e+30
6.570e+09  2.672e+05  1.712e+27  2.349e+30
7.471e+09  3.242e+05  1.030e+27  2.347e+30
7.018e+09  2.951e+05  5.620e+26  2.347e+30
```

exoplanets-SI.dat

Cours : Ajuster un modèle aux données

Paramètres du modèle

Un modèle	Des données expérimentales
Modèle de l'accélération constante : $$v(t; v_0, g) = v_0 - gt$$	

i	0	1	2	3	4	5
t_i	0.0	0.1	0.2	0.3	0.4	0.5
v_i	0.338	-1.509	-5.301	-4.404	-6.967	-6.229

paramètres à ajuster aux données expérimentales

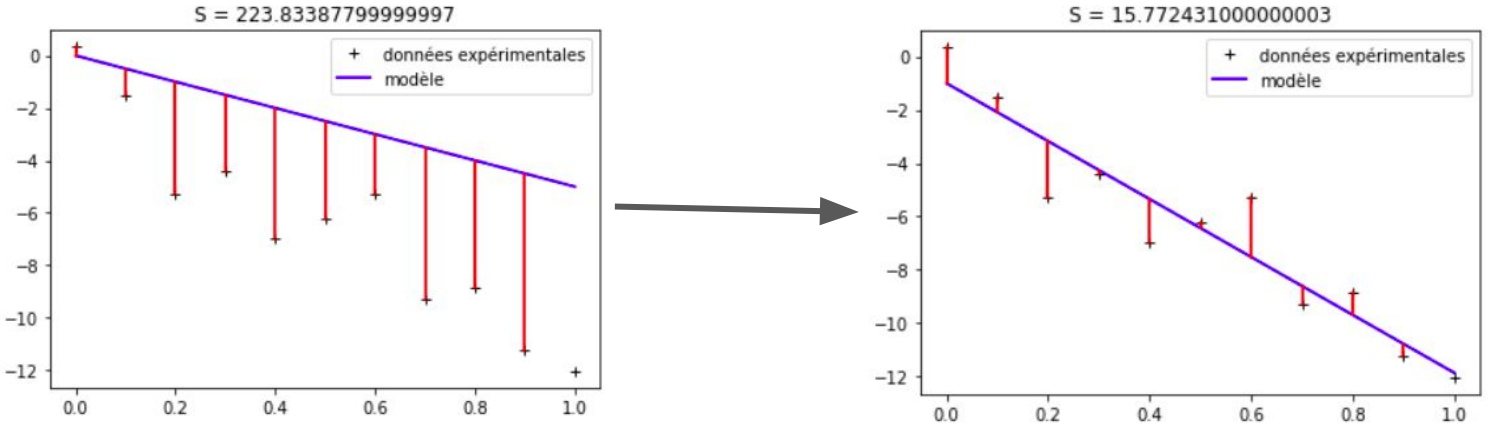
Ajuster les paramètres aux données expérimentales :

- A la main
- Pour un modèle linéaire : la régression linéaire
- (Pour un modèle non linéaire : l'algorithme de Gauss-Newton, …)
- Utilisation de la fonction `curve_fit` de `scipy.optimize`

Obtenir le minimum de $$S(a,b) = \sum_{i=0}^{N-1} [y_i - (a + bx)]^2$$

$$\frac{\partial S}{\partial a}(a_m, b_m) = 0$$

$$\frac{\partial S}{\partial b}(a_m, b_m) = 0$$



La fonction curve_fit

```
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
-------------------------------------------------------------------------

# Données expérimentale :
x_exp = [...]
y_exp = [...]
-------------------------------------------------------------------------
# Définition du modèle
def modele(x, a, b, ...) :
    return a + b*x + …

# Ajustement du modèle sur les données expérimentales :
solution = curve_fit (modele, x_exp, y_exp)
a, b, … = solution [0]

x_mod = [...]
y_mod = modele(x_mod, a, b, ...)
-------------------------------------------------------------------------

# Représenter le modèle et les données sur le même graphique
plt.plot(x_exp, y_exp, '+', label = 'données expérimentales')
plt.plot(x_mod, y_mod, '-', label = 'modèle')
```

La fonction curve_fit

```
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
-------------------------------------------------------------------------

# Données expérimentale :
x_exp = [...]
y_exp = [...]
-------------------------------------------------------------------------
# Définition du modèle :
def modele(x, a, b, ...) :
    return a + b*x + …

# Ajustement du modèle sur les données expérimentales :
solution = curve_fit (modele, x_exp, y_exp)
a, b, … = solution [0]

x_mod = [...]
y_mod = modele(x_mod, a, b, ...)
-------------------------------------------------------------------------

# Représenter le modèle et les données sur le même graphique
plt.plot(x_exp, y_exp, '+', label = 'données expérimentales')
plt.plot(x_mod, y_mod, '-', label = 'modèle')
```

Remarque :
`solution[1]` comporte des données permettant d'évaluer la qualité de votre ajustement.

Echéances

- Date limite de dépôt de la **1ère chance** pour le **MP3** : dimanche 27/02
- Date limite de dépôt de la **2nd chance** pour le **MP1** : dimanche 20/02
- Date limite de dépôt de la **2nd chance** pour le **MP2** : dimanche 27/02

A préparer pour la prochaine séance

- Faire le calepin de cours “Dérivation et intégration numérique”