Ajustement de courbes de tendance

UE 1.1 - Analyse de données spatiales

Seuls les exercices 1, 2, 6 seront traités lors des séances encadrées. La correction de l'ensemble des exercices sera mis à disposition à l'issue de la séance.

Avant-propos

Dans tous ces exercices, les fichiers de données sont organisés en colonnes, séparées par des espaces. L'ordre des données contenues est indiqué dans l'entête. On considère le fichier `data.txt` de la forme :

data.txt

```
# t[s] x
1 2.345
2 3.456
3 4.567
```

La première colonne désigne la variable t (entre crochets son unité, la seconde), la seconde la variable x.

On utilise la fonction `loadtxt` de `numpy` pour charger les données :

Python

```python
import numpy as np
data = np.loadtxt('data.txt')
t = data[:,0:1]
x = data[:,1:2]
```

Ici :

- Les lignes débutant par le caractère `'#'` sont ignorées par défaut lors de l'appel de `loadtxt`.
- La mention des colonnes de début et de fin de la plage de données extraites de `data` et affectées aux variables `t` et `x` permet d'obtenir des objets `ndarray` de dimension 2 qui seront plus faciles à manipuler par la suite.

On peut alors visualiser le nuage de points :

Python

```python
import matplotlib.pyplot as plt
plt.plot(t,x,'ob')
```

Pour la réalisation du TP, nous vous conseillons d'implémenter préalablement une fonction `Python` permettant un ajustement par moindres-carrés, prenant en entrée le vecteur des observations et la matrice modèle et renvoyant le vecteur inconnu :

```
Python
def ls(Y, A):
  ⋮
  return Xhat
```

Enfin, citons quelques opérateurs et fonctions utiles de la librairie `numpy` :

- L'opérateur `.T` pour le calcul de la transposée.
- L'opérateur `@` pour le calcul d'un produit matriciel.
- La fonction `inv` de la librairie `numpy.linalg` permet le calcul de l'inverse dune matrice.
- Les fonctions `hstack` et `vstack` pour le cumul horizontal et vertical de tableau `numpy`.
- Les fonctions `ones` et `zeros` pour la création de tableau de 1 ou de 0.
- La fonction `delete` pour supprimer un élément d'un tableau `numpy`.
- Les fonctions `argmax` et `argmin` pour accéder à l'identifiant de la valeur maximale ou minimale d'un tableau `numpy`.
- L'attribut `shape` pour accéder aux dimensions d'un tableau `numpy`.

Exercices

1. **Choix de la régression**
   On considère le nuage de point (x_i, y_i) du fichier `choix_regression.txt` (figure 1).

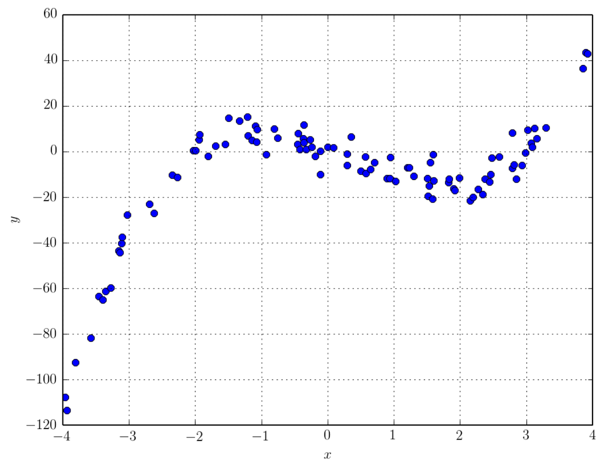

Figure 1 – Nuage de points pour le choix d'une régression.

   (a) Ajuster par moindres carrés à ce nuage les modèles suivants :
      i. $y(x) = ax + b$
      ii. $y(x) = ax^2 + bx + c$
      iii. $y(x) = ax^3 + bx^2 + cx + d$

      Visualiser et commenter les résultats.

(b) Calculer la moyenne et l'écart-type du vecteur des résidus ($\hat{V} = Y - A\hat{X}$) obtenu pour chaque ajustement [1].
Conclure.

2. **Impact des valeurs aberrantes**
On considère le nuage de point (x_i, y_i) du fichier `points_aberrants.txt` (figure 2).

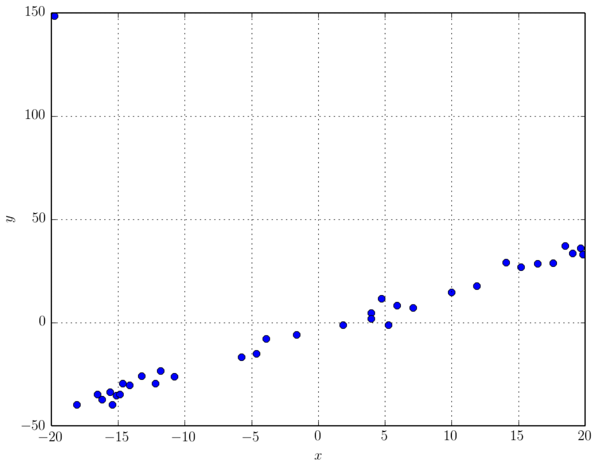

FIGURE 2 – Nuage de points pour le choix d'une régression.

(a) Ajuster par moindres carrés à ce nuage le modèle suivant $y(x) = ax+b$. Visualiser le résultat.

(b) Calculer et visualiser le vecteur des résidus de l'ajustement, $\hat{V}$ donné par : $\hat{V} = A\hat{X} - Y$. Calculer sa moyenne et son écart-type [2].

(c) Identifier la valeur maximale des résidus [3]. Ajuster à nouveau par moindres carrés le modèle $y(x) = ax + b$ à ce nuage en ayant préalablement retiré le point identifié [4]. Visualiser le résultat.

(d) Calculer la moyenne et l'écart-type du nouveau vecteur des résidus [5].

(e) Conclure.

3. **Profil de température atmosphérique**
Dans le fichier `temperature.txt`, on dispose de n mesures de température T_i (K) à différentes altitudes z_i (m) obtenues à l'aide d'un radiosondage. On peut approcher son évolution par la droite :

$$T(z) = T_0 + \Gamma \times z \tag{1}$$

Déterminer les coefficients T_0 et Γ. Visualiser le résultat.

1. On pourra utiliser les méthodes `mean()` et `std()` de `numpy`.
2. On pourra utiliser les méthodes `mean()` et `std()` de `numpy`.
3. On pourra utiliser les fonctions `argmin` ou `argmax` de `numpy`.
4. On pourra utiliser la fonction `delete` de `numpy`.
5. On pourra utiliser les méthodes `mean()` et `std()` de `numpy`.

4. **Chute libre d'une bille**
   Dans le fichier `chute_bille.txt`, on dispose des mesures de hauteur z_i (m) à différents instants t_i (s) correspondant à la chute d'une sphère métallique. En supposant une chute sans frottement, l'équation de la chute s'écrit sous la forme :
$$z(t) = -\frac{1}{2}g \times t^2 + v_0 \times t + z_0 \tag{2}$$
   Déterminer les coefficients g, v_0 et z_0. Visualiser le résultat.
5. **Célérité du son dans l'eau**
   Il existe une formulation approchée permettant de calculer la célérité du son dans l'eau, c en fonction de la température, t (en °C), la salinité S (en %) et la profondeur d (en m) :
$$c(t, S, d) = c_0 + a_1 \cdot t + a_2 \cdot t^2 + a_3 \cdot t^3 + a_4 \cdot S + a_5 \cdot S \cdot t + a_6 \cdot d \tag{3}$$
   Dans le fichier `celerite_eau.txt`, on dispose de n mesures de célérité, température, salinité et profondeur.
   Déterminer les coefficients de la formule de célérité. Visualiser le résultat.
6. **Courbe paramétrée**
   Dans le fichier `courbe_parametree.txt`, on dispose d'un ensemble de n triplets (θ_i, x_i, y_i) correspondant à la numérisation d'une courbe paramétrée de la forme :
$$\left\{ \begin{array}{lcl} x(\theta) & = & a_3 \sin^3\theta \\ y(\theta) & = & a_0 + a_1\cos\theta + a_2\cos^2\theta + a_3\cos^3\theta + a_4\cos^4\theta \end{array} \right. \tag{4}$$
   Déterminer les coefficients a_0 à a_4.
7. **Décharge d'un condensateur**
   Le fichier `decharge.txt` contient 50 mesures de tension u_i à différents instants t_i aux bornes d'un condensateur d'un circuit électrique « RC ». On sait que l'équation de la décharge s'écrit sous la forme :
$$u(t) = u_0 \times \exp\left(-\frac{t}{\tau}\right) \tag{5}$$
   Où :
   - u_0 est la tension aux bornes du condensateur avant décharge.
   - τ est la constante de temps du circuit RC ($\tau = R \times C$).

   Déterminer les coefficients u_0 et τ. Visualiser le résultat.
8. **Données marégraphiques**.
   Dans le fichier `maree_brest.txt`, on dispose de n mesures de hauteur d'eau h_i en fonction du temps t_i acquises par le marégraphe de Brest durant l'année 2010. Ces mesures de hauteur d'eau peuvent se décomposer comme la somme d'onde d'amplitude, de pulsation et de phase différentes.
   Chacune des observations du marégraphe peut ainsi se modéliser sous la forme

suivante :

$$h(t) = h_0 + \sum_{i=1}^{m} A_i \cos(\omega_i t + \Phi_i) \tag{6}$$

Où :

- h_0 est le niveau moyen, inconnu ;
- A_i est l'amplitude de l'onde i, inconnue ;
- ω_i est la pulsation de l'onde i, connue ;
- Φ_i est la phase de l'onde i, inconnue.

En pratique, un développement à l'ordre $m = 8$ est suffisant pour décrire la série de marée. Les périodes de ces 8 premières ondes sont connues, elles valent :

i	Nom de l'onde	Période [h]
1	M2	12,42
2	N2	12,66
3	S2	12,00
4	K2	11,97
5	O1	25,82
6	K1	23,93
7	P1	24,07
8	Q1	26,87

(a) Déterminer les 17 coefficients h_0, A_i et Φ_i. Visualiser le résultat.

(b) Visualiser les résidus de l'ajustement. Commenter les résultats