TP n° 3 – Tracés de fonctions, régressions et courbes de tendance par la méthode des moindres carrés

1 Liminaires

Tous les langages de programmation répandus proposent des fonctions toutes faites pour la plupart des besoins courants, écrites par les concepteurs du langage eux-mêmes, ou par des utilisateurs, puis intégrées au fil du temps dans la distribution du langage. Ces fonctions sont regroupées dans ce que l'on appelle des « bibliothèques » (encore appelées « paquets » ou « modules complémentaires »). Python n'échappe pas à la règle. Parmi les bibliothèques les plus utiles, on peut citer :

- `math` qui contient la plupart des fonctions mathématiques habituellement utilisées en analyse ;
- `numpy` qui fournit des outils variés pour le calcul numérique (scientifique) ;
- `matplotlib.pyplot` qui permet de tracer des graphiques ;
- `os` qui permet de manipuler les répertoires, des fichiers ;
- `random` qui permet de générer des nombres pseudo-aléatoires.

Pour utiliser les fonctions d'une bibliothèque, on commence par l'importer en début de programme. Par exemple, pour importer la bibliothèque `math`, on écrit :

```
import math as ma
```

La commande `as` permet de créer un *alias*, c'est-à-dire renommer la bibliothèque `math` ici en `ma`. Pour utiliser une fonction d'une bibliothèque, il est nécessaire de la préfixer avec l'alias choisi (ou le nom complet de la bibliothèque). Par exemple pour la fonction sinus, on utilisera `ma.sin`. Pour obtenir l'aide associé à une fonction, on fera `help(ma.sin)`. Pour lister les fonctions d'une bibliothèque, on utilisera la fonction `dir`, par exemple :

```
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb',
'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp',
'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma',
'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt',
'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan',
'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin',
'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
```

Mémo – **Fonctions mathématiques**

`import math as ma`

`abs(x)` ..... $|x|$
`ma.pi` ..... π
`ma.sqrt(x)` ..... $\sqrt{x}$
`ma.floor(x)` ..... $\lfloor x \rfloor$
`ma.exp(x)`, `ma.log(x)` ..... $\exp^x$, $\ln(x)$
`10**x`, `ma.log10(x)` ..... 10^x, $\log(x)$
`ma.cos(x)`, `ma.acos(x)` ..... $\cos(x)$, $\text{Arccos}(x)$
`ma.sin(x)`, `ma.asin(x)` ..... $\sin(x)$, $\text{Arcsin}(x)$
`ma.tan(x)`, `ma.atan(x)` ..... $\tan(x)$, $\text{Arctan}(x)$

1.1 Tracé de fonctions

Pour tracer la courbe représentative d'une fonction f sur un intervalle $I = [a, b] \subset \mathbb{R}$, nous allons fabriquer une liste `X=[x1,..., xn]` contenant un certain nombre de réels de cet intervalle, et ensuite fabriquer par compréhension la liste des images `Y=[f(x) for x in X]` associée. Les deux séquences `X` et `Y` ont alors même longueur et on affiche une approximation du graphe de f sur I avec les commandes :

```
import matplotlib.pyplot as plt
plt.figure()
plt.plot(X, Y)
plt.show()
```

La fonction `plot` se charge de placer sur un graphe chacun des points M_i de coordonnées $(x_i, f(x_i))$, puis de le relier à son successeur M_{i+1} par des segments de droite. On obtient ainsi une approximation du graphe de f par une ligne polygonale (ou ligne brisée). Si l'on se donne suffisamment de points, cette ligne qui paraît lisse à l'œil. Tous les logiciels de calcul scientifique fonctionnent de cette manière.

MÉMO – **Tracés et graphiques**

`import matplotlib.pyplot as plt`
`plt.figure()` ..... définition d'une nouvelle figure
`plt.subplot(nlignes, ncolonnes, num)` ..... sous-figure
parcours par ligne avec `num`$\in [\![1, $`nlignes` $\times$ `ncolonnes`$]\!]$
`plt.plot(X,Y)` ..... graphe « classique »
`plt.semilogx(X,Y)` ..... échelle log pour l'axe des abscisses
`plt.semilogy(X,Y)` ..... échelle log pour l'axe des ordonnées
`plt.loglog(X,Y)` ..... échelle log pour les axes
`plt.xlim(xmin,xmax)` ..... limites de l'axe des abscisses
`plt.ylim(ymin,ymay)` ..... limites de l'axe des ordonnées
`plt.axis("equal")` ..... échelle des axes commune
`plt.xlabel("")` ..... légende de l'axe des abscisses
`plt.ylabel("")` ..... légende de l'axe des ordonnées
`plt.title("")` ..... titre du graphe
`plt.legend()` ..... affiche la légende, nom défini pour chaque courbe avec `label=""`
`plt.grid()` ..... affiche une grille
`plt.show()` ..... affichage (du buffer) de la figure

Options graphiques

— trait : `'-'` plein, `':'` pointillés, `'-.'` alterné
— couleur : `'k'` noir, `'r'` rouge, `'g'` vert, `'b'` bleu `'c'` cyan, `'m'` magenta, `'y'` jaune
— marque : `'+'` plus, `'x'` croix, `'*'` étoile, `'o'` rond, `'d'` diamant, `'h'` hexagone

Exercice 3.1

1. Définir une fonction `Discretisation` qui prend en entrée trois arguments `a`, `b` et `n` et qui renvoie une liste de n valeurs équidistantes entre a et b.
2. Définir une fonction `graphe` qui prend comme arguments un nom de fonction `f`, une liste de deux flottants `I` contenant les deux bornes d'un intervalle fermé et un entier `n` correspondant au nombre de points à utiliser pour le graphe de f sur I et qui affiche à l'écran une approximation à n points du graphe de f sur I.
3. Testez votre fonction en traçant, par exemple, la courbe représentative de $x \mapsto 3\sin(x)$ sur $[0, 2\pi]$ avec 1000 points.
4. Définir une fonction `graphes`, adaptée de la fonction `graphe`, qui prend comme arguments le nom d'une famille de fonctions `f`, implémentation de f_k, une séquence des k sous forme de liste `K`, une séquence de 2 flottants `I` contenant les 2 bornes d'un intervalle fermé et un entier `n` correspondant au nombre de points à utiliser pour les graphes de $(f_k)_{k\in K}$ sur I.
5. Testez votre fonction en traçant, par exemple, la famille de courbes représentatives de $f_k : x \mapsto k\sin(x)$ pour $k \in \{1; 2; 5\}$ en précisant la légende de chaque courbe.

1.2 Manipulations de fichiers textes

Copier l'archive `Info-TP03_donnees.zip` dans votre répertoire personnel, la décompresser puis initialiser votre code `python` en définissant le répertoire `donnees` comme répertoire de travail du *shell* avec la commande `chdir` de la bibliothèque `os`, qui prend comme argument l'adresse d'un répertoire sous forme de chaîne de caractères que l'on peut obtenir en cliquant simplement sur la barre d'adresse d'un navigateur de fichiers puis en copiant le contenu avec le combinaison de touches `Ctrl+c` [1] :

```
import os
os.chdir('\\long\\chemin\\donnees')
```

Le contenu de ce répertoire peut être listé avec la commande `os.listdir()`.

MÉMO – **Manipulation de fichiers texte**

`f=open(fichier,'r')` ..... ouvrir un fichier texte en lecture (ou `'w'` écriture, `'a'` ajout)
`f.readline()` ..... lit la ligne suivante
`f.read()` ..... lit tout le fichier si nb de caractères non donné
`f.write(chaine)` ..... écrit dans le fichier
`f.writeline(chaine)` ..... écrit une nouvelle ligne
`f.close()` ..... fermer le fichier

Manipulation de chaînes de caractères
`s.split(x)` ..... découpe la chaîne `s` sur le caractère `x`
`s.strip()` ..... supprime les caractères EOL, EOF
`\n`, `\t` ..... retour à la ligne, tabulation
`int(s)` ..... conversion en entier `int` (rep. exacte de $\mathbb{Z}$)
`float(s)` ..... conversion en flottant `float` (rep. approchée de $\mathbb{R}$)
`str(num)` ..... convertit un nombre en chaîne

Lire un fichier texte de 3 colonnes t, x, y séparées par des tabulations :

```
L=[[],[],[]]
fichier = open('fichier.txt','r')
for ligne in fichier:
    champs=ligne.strip().split('\t')
    for i in range(3):
        L[i].append(float(champs[i]))
fichier.close()
```

Écrire dans un fichier texte

```
fichier = open('fichier.txt','w')
fichier.write(str(nombre)+"chaine"+"\n")
fichier.close()
```

Exercice 3.2 (Lecture de fichiers textes)

1. Lire les données du fichier `mesures2D.txt` (qui contient deux colonnes séparées par une tabulation sous la forme « `x y` ») pour former deux listes de flottants respectivement associées aux deux colonnes du fichier.
2. Adapter ce code pour définir une fonction `Lecture` qui prend comme arguments une chaîne de caractères associée à un nom de fichier, un entier `n` correspondant au nombre de colonne que l'on souhaite récupérer et une chaîne de caractères `sep` correspondant au séparateur de colonne (le plus souvent `\t` ou `;`) et qui renvoie une liste de n listes de flottants respectivement associées aux n colonnes lues du fichier.
3. Testez votre fonction avec les fichiers `mesures2D.txt`, `mesures3D.txt` et `mesures3D2.txt`.

1. Avec Microsoft « fenêtres », il est nécessaire de remplacer chaque *backslash* \ par une paire \\ ou de mettre `r` devant la chaîne de caractères du chemin.

2 Régression et courbes de tendance

Il est de nombreuses situations pratiques où l'on cherche à associer un modèle mathématique à des données expérimentales, par exemple à une série de données représentée par un nuage de points $\{(x_i, y_i), 1 \leqslant i \leqslant m\}$ de $\mathbb{R}^2$. La notion d'association revient à rechercher une courbe passant au mieux par tous les points $M_i\,(x_i, y_i)$. Le plus souvent, la courbe recherchée représente une fonction f, que l'on souhaite la plus simple possible (par exemple affine ou polynomiale). Ainsi, on est amené à rechercher les paramètres $q = (q_1, q_2, \ldots, q_p) \in \mathbb{R}^p$ de la fonction que l'on notera f_q. Une approximation par la méthode des moindres carrés consiste alors à rechercher la fonction f_q minimisant la somme des carrés des écarts entre $f_q(x_i)$ et y_i. Ainsi, on suppose l'existence de q^*, une valeur du paramètre q réalisant le minimum de la fonction coût ou objectif J, c'est-à-dire tel que :

$$q^* \in \mathbb{R}^p \quad \text{et} \quad \forall q \in \mathbb{R}^p, \qquad J(q^*) \leqslant J(q) \qquad \text{avec} \qquad J(q) = \sum_{i=1}^{m} \left(y_i - f_q(x_i)\right)^2$$

Bien que l'on ne sache pas, *a priori*, si ce minimum existe et s'il est atteint en un unique point, on ne traitera dans ce TP que des cas où il existe et où il est unique.

Une régression linéaire consiste en la recherche de la droite d'équation $y = ax + b$ passant au plus près des m points par la méthode des moindres carrés. Dès lors, le problème revient à déterminer le couple $q = (a, b)$ réalisant le minimum de la fonction coût

$$J(a, b) = \sum_{i=1}^{m} \left(y_i - a x_i - b\right)^2$$

La recherche du minimum s'apparente à la recherche d'un point critique [2] et nécessite d'exprimer un système de deux équations associées aux dérivées partielles de la fonction de coût J par rapport aux paramètres a et b :

$$\begin{cases} 0 = \dfrac{\partial J}{\partial a}\,(a, b) = -2 \displaystyle\sum_{i=1}^{m} x_i \;\left(y_i - a\, x_i - b\right) \\ 0 = \dfrac{\partial J}{\partial b}\,(a, b) = -2 \displaystyle\sum_{i=1}^{m} \left(y_i - a x_i - b\right) \end{cases} \iff \begin{cases} a \left(\displaystyle\sum_{i=1}^{m} (x_i)^2\right) + b \left(\displaystyle\sum_{i=1}^{m} x_i\right) = \displaystyle\sum_{i=1}^{m} x_i\, y_i \\ a \left(\displaystyle\sum_{i=1}^{m} x_i\right) + bm = \displaystyle\sum_{i=1}^{m} y_i \end{cases}$$

admettant comme solution :

$$a = \frac{\mathrm{Cov}\,(x, y)}{V(x)} \quad \text{et} \quad b = \bar{y} - a\bar{x}$$

où nous avons introduit les notations de moyenne, de covariance et de variance des séries considérées, respectivement définies comme :

$$\bar{x} = \frac{1}{m} \sum_{i=1}^{m} x_i, \qquad \bar{y} = \frac{1}{m} \sum_{i=1}^{m} y_i, \qquad \mathrm{Cov}\,(x, y) = \left(\frac{1}{m} \sum_{i=1}^{m} x_i\, y_i\right) - \bar{x}\;\bar{y} \quad \text{et} \quad V(x) = \mathrm{Cov}(x, x)$$

Cela prouve que la droite obtenue par régression linéaire admet pour équation $y = a\,(x - \bar{x}) + \bar{y}$ et passe par le point de coordonnées $(\bar{x}, \bar{y})$ qui est l'isobarycentre du nuage de points. Pour évaluer la qualité de l'approximation, on utilise généralement le coefficient de corrélation $R \in [-1; 1]$, défini comme :

$$R = \frac{\mathrm{Cov}(x, y)}{\sqrt{V(x)\,V(y)}}$$

Si $R = 0$ les deux variables ne sont pas corrélées mais si $R = \pm 1$, les deux variables sont linéairement dépendantes. C'est ce qui explique l'usage courant du carré du coefficient de corrélation $R^2 \in [0, 1]$ comme mesure de la qualité de l'approximation.

2. Un point critique est un point annulant les dérivées partielles d'ordre 1. Sous réserve d'hypothèses sur l'ensemble des solutions admissibles et sur J, un minimum est un point critique, mais la réciproque n'est pas toujours vraie. Par exemple, pour une fonction d'une variable, si $J'(q) = 0$, J n'admet pas nécessairement un minimum, mais peut avoir un maximum local ou un point d'inflexion.

Exercice 3.3 (Ajustement affine)
Dans tout ce qui suit, une série de données sera contenue dans une liste de flottants.

1. Définir une fonction `Moyenne` qui prend comme argument une liste `X` et qui renvoie sa moyenne.
2. Définir une fonction `Covariance` qui prend comme arguments deux listes `X` et `Y` de même dimension et qui renvoie leur covariance. En déduire une fonction `Variance`.
3. Définir une fonction `Regression` qui prend comme arguments deux listes `X` et `Y` de mesures et qui renvoie la valeur des deux paramètres (a, b) et la valeur du coefficient de corrélation R.
4. Utiliser la fonction `Lecture` pour former deux listes avec les données du fichier `mesures2D.txt` puis, après appel de la fonction `Regression`, afficher les points avec des cercles rouges (option `marker='o',color='r',linestyle='None'` ou `'or'` de `plot`) et la droite en bleu (option `'-b'`), en faisant figurer dans la légende l'équation de la droite avec la valeur des paramètres a et b et du carré du coefficient de corrélation R^2. Pour arrondir un flottant `x` au 4e chiffre après la virgule, on pourra utiliser la commande `round(x,4)` qui renvoie un flottant, que l'on ne manquera pas de convertir en chaîne de caractères.

Nous venons d'associer une droite de régression de y en x qui minimise la somme des carrés des distances verticales des points à la droite. Nous aurions aussi pu construire la droite de régression de x en y minimisant la somme des carrés des distances horizontales des points à la droite d'équation $y = a'\,x + b'$:

$$J_h(\alpha, \beta) = \sum_{i=1}^{m} (x_i - \alpha y_i - \beta)^2\,, \qquad \text{avec} \quad a' = \frac{1}{\alpha} \quad \text{et} \quad b' = \frac{-\beta}{\alpha}$$

6. Utiliser la fonction `Regression` pour déterminer les coefficients α et β, puis a' et b', d'après les données du fichier `mesures2D.txt`. Afficher sur une même figure les données et les droites de régression de y en x et de x en y (chacune d'une couleur). Que constatez-vous ?

— Complément —
pour les plus rapides

Exercice 3.4 (Ajustements « de tableurs »)
La méthode de régression linéaire peut être étendue à d'autres modèles comme :

- le modèle logarithmique $y = a\ln(x) + b$, en posant $y = ax' + b$ avec $x' = \ln(x)$;
- le modèle exponentiel $y = b \times a^x$, en posant $y' = a'x + b'$ avec $y' = \ln(y)$, $a' = \ln(a)$ et $b' = \ln(b)$;
- le modèle puissance $y = b \times x^a$, en posant $y' = ax' + b'$ avec $y' = \ln(y)$, $x' = \ln(x)$ et $b' = \ln(b)$.

Ce sont les options de courbes de tendance (à deux paramètres $q = (a, b)$) généralement proposées dans les tableurs. On notera toutefois que les approximations avec les modèles exponentiel et puissance minimisent la somme des $(\ln(y_i) - \ln \circ f_q(x_i))^2$ et non la somme des $(y_i - f_q(x_i))^2$, ce qui ne revient pas au même !

1. Définir une fonction `Ajustements` qui prend comme arguments deux listes `X` et `Y` de mesures et une chaîne de caractères associée au type de loi (`"lin"`, `"log"`, `"exp"` ou `"pow"`) pour l'ajustement et qui renvoie la valeur des deux paramètres (a, b) du modèle correspondant et la valeur du coefficient de corrélation R.
2. Définir une procédure `Affichage` qui prend comme arguments deux listes `X` et `Y` de mesures et une chaîne de caractères associée au type de loi (`"lin"`, `"log"`, `"exp"` ou `"pow"`) pour l'ajustement et qui, après appel de la fonction `Ajustements`, génère une figure avec les données sous forme de points, la courbe de tendance en trait plein et en faisant figurer dans la légende l'équation du modèle avec ses paramètres a et b et le carré du coefficient de corrélation R^2.
3. Utiliser cette fonction pour chacune des 3 séries de données associées à un type $\alpha \in \{\text{lin, log, exp, pow}\}$ et contenues dans les fichiers $\alpha\beta$`.txt`, avec $\beta \in [\![1, 3]\!]$. Que remarquez-vous sur la qualité des approximations en fonction de la distribution des données ? Préciser ce que cela peut induire sur les limites d'utilisation de la méthode des moindres carrés.

* *
*