M1 IM-MF/MPA
Université Côte d'Azur

POO
2023-24
TP2 (Python 2)

TP2 POO - Python

simon.girel@univ-cotedazur.fr, gilles.scarella@univ-cotedazur.fr

1 Loi de Poisson

Soit X une variable aléatoire suivant une loi de Poisson de paramètre λ, on a alors, pour tout entier $k \geq 0$

$$\mathbb{P}(X = k) = \frac{\lambda^k}{k!} \exp(-\lambda)$$

- Dans le fichier *exo1.py*, définir une fonction à deux arguments, λ et n, qui renvoie un tableau *numpy* de taille n contenant les valeurs de $\mathbb{P}(X = k)$ pour $k = 0$ à $n-1$.
  Indication: *numpy.math.factorial(k)* renvoie $k!$.
- Pour $\lambda = 11$ et $n = 25$, représenter graphiquement avec *matplotlib*, le résultat de l'appel à la fonction précédente, en fonction de k=0 à $n-1$. Les points doivent être reliés entre eux et apparaître aussi avec un marqueur (diamant), le tout avec la couleur rouge.
- Sur la même figure, représenter le même type de résultat que précédemment, pour la même valeur de n et pour $\lambda = 7$; on utilisera le marqueur 'o' et la couleur verte.

  Même chose pour $\lambda = 5$, on utilisera le marqueur '*' et la couleur noire.

  Ajouter une légende à la figure obtenue, le xlabel 'k' et le ylabel 'P'.

2 Jeu de données *iris*

iris est un jeu de données classique donnant des mesures concernant la fleur iris. Ces données se trouvent dans le fichier *iris.txt* disponible dans la page Moodle du cours, sous *Python/TP2* (ou bien au lien https://rb.gy/pc4wnq)

Télécharger le fichier *iris.txt* dans votre dossier courant. Ce fichier contient des données de mesures pour la longueur de sépale, largeur de sépale, longueur de pétale et largeur de pétale.

- Dans un premier temps, pour chacune de ces 4 quantités, **afficher le nuage de points** correspondant (1 seul par figure, vous aurez donc 4 figures) Sur chaque figure respective, superposer au nuage de points une ligne horizontale de couleur jaune, dont l'ordonnée est égale à la **moyenne** du nuage de points, ainsi qu'une seconde ligne horizontale de couleur magenta, dont l'ordonnée est égale à la **médiane** du nuage de points,

On utilisera les styles d'affichage suivants, pour les nuages de points

$Sepal.Length$	'bo' (ronds bleus)
$Sepal.Width$	'rd' (diamants rouges)
$Petal.Length$	'g*' (étoiles vertes)
$Petal.Width$	'k.' (points noirs)

- Dans un second temps, **afficher** pour chacune des quantités, **l'histogramme avec 20 bins (voir cours 2 page 46)**.

3 Représentation de fonctions mathématiques

On considère la fonction u suivante et sa dérivée, définies à partir de deux paramètres τ_m et τ_s différents, vérifiant pour tout $t \geq 0$

$$\begin{aligned} u(t) &= \frac{1}{(1-\tau_s/\tau_m)}\left(\exp(-\frac{t}{\tau_m}) - \exp(-\frac{t}{\tau_s})\right) \quad \forall t \geq 0 \\ \frac{du}{dt}(t) &= \frac{1}{(1-\tau_s/\tau_m)}\left(-\frac{1}{\tau_m}\exp(-\frac{t}{\tau_m}) + \frac{1}{\tau_s}\exp(-\frac{t}{\tau_s})\right) \quad \forall t \geq 0 \end{aligned}$$

Représenter avec un subplot de matplotlib (voir cours 2, pages 37-38) les deux courbes représentant u et sa dérivée pour 8 couples différents de (τ_m, τ_s):

τ_m	0.1	1	1	1	1	1.1	2	4
τ_s	1	1.1	2	4	0.1	1	1	1

Indications:

- On considère le domaine de définition $x \in [0, x_{max}]$ pour $x_{max} = 13$.
- On utilisera **numpy.arange** ou **numpy.linspace** pour discrétiser l'intervalle $[0, x_{max}]$ (avec suffisamment de points)
- **Calculer y_{max} comme la valeur maximum de u, pour tous les couples (τ_m, τ_s) du tableau**
- Toutes les figures doivent être affichées pour les mêmes axes, pour les abscisses (càd $[0, x_{max}]$) et les ordonnées $[-0.25, y_{max}]$.
- On soignera la légende, la figure 1 donne un exemple d'affichage.

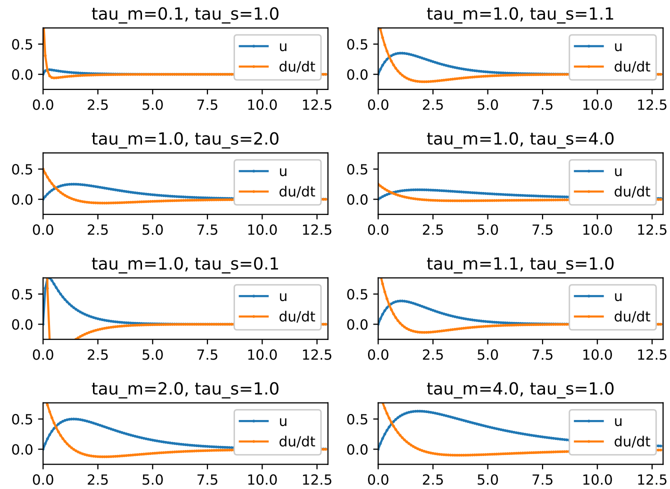

Figure 1: Exemple de figure à obtenir pour l'exo 2

4 Droite des moindres carrés

On suppose qu'il existe une relation entre deux quantités physiques X et Y. Le tableau 1 donne un ensemble de mesures - $(x_i)_{1\leq i\leq N}$ étant un échantillon de X (idem pour $(y_i)_{1\leq i\leq N}$ par rapport à Y).

(x_i)	1	5	10	15	20	25	30	35	40	45	50
(y_i)	4.1	26.4	39.9	56.5	80.1	112.2	120.8	138.0	160.4	167	200.2

Table 1: Données de l'exercice 4

On veut calculer la droite des moindres carrés représentant la dépendance de Y selon X. On veut donc calculer les réels α et β minimisant

$$\|y - MV\|_2^2 = \sum_i (y_i - \alpha - \beta\, x_i)^2$$

avec

$$V = \begin{pmatrix} \alpha \\ \beta \end{pmatrix},\ M = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix} \text{ et } y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

Ecrire un script python *lsq_line.py* qui

- affiche, avec matplotlib, le nuage des points (x_i, y_i) dans une figure

- calcule α et β. Pour rappel, l'estimateur des moindres carrés V est l'unique solution de
$$M^t MV = M^t y.$$
- affiche à l'écran le résultat sous la forme suivante (on remplacera alpha et beta par les valeurs calculées)

```
La droite des moindres carres a pour equation y = alpha + beta*x
```

- affiche la droite des moindres carrés, dans la même figure contenant le nuage de points

5 Maillage triangulaire d'un rectangle [exercice facultatif]

On considère un rectangle allant du point $(0,0)$ au point (ℓ_x, ℓ_y) qu'on va découper en triangles. (n_x+1) points seront utilisés en x, (n_y+1) en y, on suppose avoir un pas de discrétisation h tel que $\ell_x = n_x\, h$ et $\ell_y = n_y\, h$.

On manipulera deux matrices

- la matrice M contenant les coordonnées des points du maillage. Elle sera de taille $((n_x+1)(n_y+1), 2)$.
- la matrice C définissant la connectivité du maillage. Elle est de taille $(2n_x n_y, 3)$.

Dans le fichier maillage.py, **créer une fonction python *plot_triangle* qui affiche un triangle** avec matplotlib, cette courte fonction a pour arguments un tableau numpy 2d correspondant à M et ℓ_c, une ligne de C (càd un tableau numpy 1d). Elle permet d'afficher le triangle correspondant à la ligne ℓ_c de C et utilisant les coordonnées M.

Dans le fichier maillage.py, **créer une fonction python *coord***, ayant pour arguments n_x, n_y et h (le pas de discrétisation) qui implémente la définition de la matrice M.

Indication: la numérotation des nœuds du maillage doit suivre un ordre. Par exemple, on commencera par les points de coordonnées $(0, j\,h)$ $(j = 0, .., n_y)$, puis $(h, j\,h)$ ... $(n_x\,h, j\,h)$.

Dans le fichier maillage.py, **créer une fonction python *connec***, ayant pour arguments n_x et n_y, qui définit la connectivité du maillage triangulaire (à savoir la matrice C).

Indication: la ligne i de C contient trois entiers i_0, i_1, i_2 qui correspondent aux numéros i_0, i_1, i_2 des nœuds composant le triangle i. $M[i_0, :]$ contient les coordonnées du nœud i_0, $M[i_1, :]$ contient les coordonnées du nœud i_1 ...

Enfin, **créer une fonction python *plotmesh***, ayant pour arguments n_x, n_y et h, utilisant les fonctions *coord*, *connec* et *plot_triangle*, affichant le maillage triangulaire du rectangle avec matplotlib.