DS n° 05 – Régression linéaire

- Faire tous les exercices dans un fichier NomPrenom.py à sauvegarder,
- mettre en commentaire l'exercice et la question traités (ex : # Exercice 1),
- ne pas oublier pas de commenter ce qui est fait dans votre code (ex : # Je crée une fonction pour calculer la racine d'un nombre),
- il est pas possible de demander un déblocage pour une question, vous auriez alors 0 à cette question.

1 Récupération des données

L'objectif de cette épreuve est de traiter des données concernant l'expérimentation de la décharge d'un condensateur.

Les données de l'expérience sont contenues dans le fichier `donnees.json`. Il est disponible dans le répertoire « /home/eleve/Ressources/PTSI/ », ne doit pas être déplacé mais ouvert directement depuis son emplacement initial.

Les lignes de code suivantes permettent de récupérer ces données.

```
1 import json
2
3 with open('donnees.json', 'r') as openfile:
4     dictionnaire = json.load(openfile)
```

Elles créent un dictionnaire `dictionnaire` dont les clefs sont les suivantes :

- `'titre'` : titre du tracé,
- `'temps'` : instants `t` des mesures,
- `'U'` : tensions `U` aux bornes du condensateur à ces instants,
- `'log(U)'` : `log(U)` à ces instants,
- `'datalog'` : la liste des couples `[t,log(U)]`.

Question 1 Recopier ces lignes et afficher la valeur associée à la clef `'titre'`.

Question 2 Créer `lt`, `lu`, `loglu` et `datalog` contenants les données affectées respectivement aux clefs `'temps'`, `'U'`, `'log(U)'` et `'datalog'` converties au format `array numpy`. Afficher le contenu de `datalog`.

Question 3 Tracer à l'aide de la fonction `scatter` de la bibliothèque `matplotlib.pyplot` :

- `lu` en fonction de `lt`,
- `loglu` en fonction de `lt`.

Question 4 Coder une fonction `tri`, vous pourrez utiliser la méthode de votre choix. Utiliser cette fonction sur la liste `[2,4,3,1,5]`. Afficher le résultat obtenu.

Question 5 A partir de la réponse précédente, proposer une solution afin de trier `datalog` dans le sens des `t` croissants.

2 Régression linéaire

On sait que la tension aux bornes d'un condensateur en phase de décharge peut s'écrire $u(t) = U_0 \cdot e^{-\frac{t}{\tau}}$, ce qui mène à $ln(u(t)) = ln(U_0) - \frac{t}{\tau}$.

Ainsi, le nuage de points `datalog` peut être assimilé à un modèle de droite affine dont l'ordonnée à l'origine est $ln(U_0)$ et la pente est $-\frac{1}{\tau}$.

Question 6 Déterminer et afficher la valeur de $ln(U_0)$, en regardant la première valeur classée dans `datalog`.

Afin de déterminer la pente de la courbe, la suite va consister à soustraire cette valeur à l'ensemble des ordonnées de `datalog` afin de créer `datalog0` qui pourra être modélisée par un modèle linéaire.

Question 7 Déterminer `datalog0` et tracer l'ensemble des points qu'elle contient.

La suite va consister à rechercher la droite la plus « proche » du nuage de points `datalog0`. Pour cela nous allons utiliser la fonction `ecart(liste,pente)` à compléter :

```
def ecart(liste,pente):
    ........
    for i in range(len(liste)):
        .......(liste[i][0]*pente-liste[i][1])**2
    return ......
```

Question 8 Recopier et compléter ce code et déterminer l'écart entre les données `datalog0` et les courbes de pente `-5` et `-10`.

Question 9 Tracer la courbe qui donne le résultat de la fonction `ecart` (en ordonnée) en fonction de la `pente` (en abscisse), avec `pente` $\in [-20, 0]$.

Question 10 Chercher à l'aide d'un code python la valeur de la `pente` pour l'`ecart` minimal. En déduire la valeur de τ.

Question 11 Afin de vérifier vos résultat, les comparer avec ceux obtenus par la fonction `polyfit` de la bibliothèque `numpy`. Vous pourrez utiliser le code suivant à titre d'exemple pour des points répartis autour de la droite d'équation $y = 2 \cdot x + 1$.

```
datalog=np.array(datalog)
x=[0,1,2,3]
y=[0.86,3.3,4.8,6.9]
result=np.polyfit(x,y,1)
print("Résultat polyfit : pente de {:.2f},\
                ordonnée à l'origine {:.2f}".format(result[0],result[1]))
```

Question 12 Afin de visualiser le résultat, superposer le modèle affine ainsi obtenu (par votre script ou avec `polyfot` avec le nuage de points de `datalog`.

FIN

Correction

Question 1

```python
import json

with open('donnees.json', 'r') as openfile:
    dictionnaire = json.load(openfile)

print(dictionnaire['titre'])
```

Question 2

```python
import numpy as np

lt=np.array(dictionnaire['temps'])
lu=np.array(dictionnaire['U'])
loglu=np.array(dictionnaire['log(U)'])
datalog=dictionnaire['datalog']

print(datalog)
```

Question 3

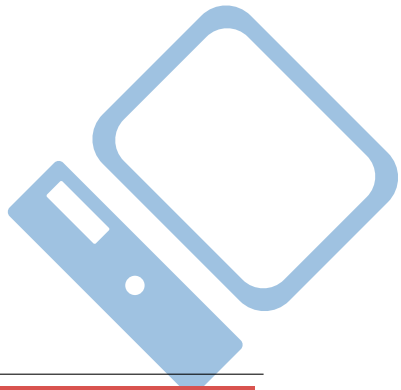
```python
import matplotlib.pyplot as plt

plt.scatter(lt,lu)
plt.show()

plt.scatter(lt,loglu)
plt.show()
```

Question 4

```python
def tri(liste):
    # Parcours de 1 à la taille de la liste
    for i in range(len(liste)-1):
        # Initialiser le min
        min=liste[i]
        jmin=i
        for j in range(i, len(liste)):
            # Chercher le min
            if liste[j]<min:
                jmin=j
                min=liste[j]
```



```
        # Permuter le min et l'élément i
        liste[i],liste[jmin]=liste[jmin],liste[i]


liste=[2,4,3,1,5]
tri(liste)
print(liste)
```

Question 5

On peut directement utiliser la fonction précédente car python compare les listes en suivant l'ordre lexicographique. Ainsi, les premiers éléments de chaque liste sont d'abord comparés entre eux, c'est donc `t` qui sera le critère de tri par défaut.

On peut aussi forcer l'utilisation du premier élément de `liste` pour faire le classement en modifiant la fonction comme suit.

```
def tri2(liste):
    # Parcours de 1 à la taille de la liste
    for i in range(len(liste)-1):
        # Initialiser le min
        min=liste[i][0]
        jmin=i
        for j in range(i, len(liste)):
            # Chercher le min
            if liste[j][0]<min:
                jmin=j
                min=liste[j][0]
        # Permuter le min et l'élément i
        liste[i],liste[jmin]=liste[jmin],liste[i]

print(datalog)
tri2(datalog)
print(datalog)
```

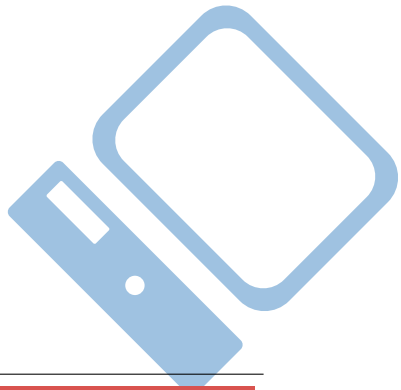
Question 6

```
print("L'ordonnée à l'origine est {:.2f}".format(datalog[0][1]))
```

Question 7

```
datalog0=[]
for t,data in datalog:
    datalog0.append([t,data-datalog[0][1]])

for t,data in datalog0:
    plt.scatter(t,data)
plt.show()
```



Question 8

```
def ecart(liste,pente):
    somme=0
    for i in range(len(liste)):
        somme+=(liste[i][0]*pente-liste[i][1])**2
    return somme

print(ecart(datalog0,-5),ecart(datalog0,-10))
```

Question 9

```
pentes=np.linspace(-20,0,1000)
ecarts=[ecart(datalog0,pente) for pente in pentes]
```

Question 10

```
imin=0
min=ecarts[imin]
for i in range(1,len(ecarts)):
    if ecarts[i]<min:
        imin=i
        min=ecarts[i]
pente=pentes[imin]
print('On trouve une pente de {:.2f}'.format(pente))
print('La valeur de tau est {:.2f}'.format(-1/pente))
```

Question 11

```
datalog=np.array(datalog)
x=datalog[:,0]
y=datalog[:,1]
result=np.polyfit(x,y,1)
print("Résultat script : pente de {:.2f},\
                une ordonnée à l'origine {:.2f}.".format(pente,datalog[0][1]))
print("Résultat polyfit : pente de {:.2f},\
                ordonnée à l'origine {:.2f} => OK.".format(result[0],result[1]))
```

Question 12

```
for x1,y1 in datalog:
    plt.scatter(x1,y1)
plt.plot(x,pente*x+datalog[0][1])
#plt.plot(x,result[0]*x+result[1])
plt.show()
```