TP 09

Lecture de fichier et tracé de courbes

Savoirs et compétences :

- ❒ *Th. 8 : Tris.*

Proposition de corrigé

Activité 1 – Mise en situation et généralité

Activité 2 – Tracé de fonctions simples

En préambule :

```
import matplotlib.pyplot as plt
from numpy import exp, linspace, pi, sin, cos
from math import floor
```

Question 1

x est une liste python, donc de type `list`.

Question 2

Python représente une fonction comme une ligne brisée. On indique les coordonnées des extrémités des segments en passant en argument la liste des abscisses et celle des ordonnées à la fonction plot.

Question 3

Le tracé s'arrete au point d'abscisse 9,5. C'est bien le dernier élément de x.

Question 4

```
def ex_sin(nom\_de\_fichier):
    """Trace la courbe du sinus sur [0,10] et l'enregistre dans nom\_de\_fichier"""
    x = linspace(0,10,200)
    y = [sin(t) for t in x]

    plt.clf()
    plt.plot(x,y,label='sin(x)')
    plt.xlabel('x')
    plt.legend(loc=0)
    plt.title('Tracé du sinus sur [0,10]')
    plt.savefig(nom\_de\_fichier)
```

Question 5

```
def transitoire(A,nom\_de\_fichier):
    """Trace les graphes de t->A(1-exp(t/tau)) pour tau = 2,4,6,8 sur [0,10].
       Les sauvegarde dans nom\_de\_fichier."""
    x = linspace(0,10,200)
    tau = [0.5,1,2,4,8]
    style = ['g-','b-','r-','b--','r--']
    plt.clf()
    for k in range(5):
```

```
        y = [A*(1-exp(-t/tau[k])) for t in x]
        plt.plot(x,y,style[k],label='$\\tau='+str(tau[k])+'$')
    plt.xlabel('$t$')
    plt.ylabel('$'+str(A)+'\\times\\exp(-t / \\tau)$')
    plt.title('Régime transitoire, A={}'.format(A))
    plt.axis([0,10,0,A])
    plt.legend(loc=0)
    plt.savefig(nom\_de\_fichier)
    return None
```

Activité 3 – Exploiter les données enregistrées sur le système Comax

Question 6 *Après avoir observé la structure du fichier CoMax.txt, exécuter le fichier lire_comax.py permettant d'extraire sous deux listes distinctes* `temps` *et* `q_exp` *les instants de prise d'échantillonnage et les positions codeur correspondantes en tops (nombre de points). Convertir ces deux listes en tableaux Numpy (*`import numpy as np`*), grâce à la commande* `liste=np.array(liste)`.

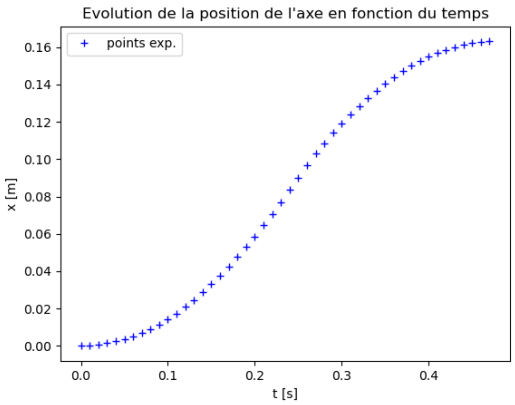
```
import numpy as np
temps = np.array(temps)
q_exp = np.array(q_exp)
```

Question 7 *Générer alors un tableau des positions verticales de l'axe nommé* `X_exp`, *qui a une condition initiale nulle sur la position.*

```
X_exp = (q_exp-q_exp[0])*(3.41/1e6)
```

Question 8 *Tracer l'évolution des positions mesurées expérimentales en fonction du temps, avec des croix bleues (+). On légendera correctement les axes, et on indiquera une légende du type : "points mesurés"*

```
plt.figure(1)
plt.plot(temps,X_exp,'b+',label="points exp.")
plt.legend()
plt.xlabel('t [s]'); plt.ylabel('x [mm]')
plt.title("Evolution de la position de l'axe ↙
    en fonction du temps")
plt.savefig('tp04_durif_q08.png')
```



Question 9 *Trouver les expressions littérales de t_1, t_2, X_1 et de X_2 en fonction de A_{cmax} et de V_{max}.*

```
"""Donnees"""
Acmax = 2.83; Vmax = 0.68
t1 = Vmax/Acmax; t2 = 2*t1
X1 = Vmax**2/(2*Acmax)
```

Question 10 *Concevoir deux fonctions* `Loi_Vitesse` *et* `Loi_position` *prenant en argument un instant t et permettant de retourner la vitesse, respectivement la position, à cet instant. (A noter que l'on pourra introduire des variables globales, comme t_1, t_2, etc.)*

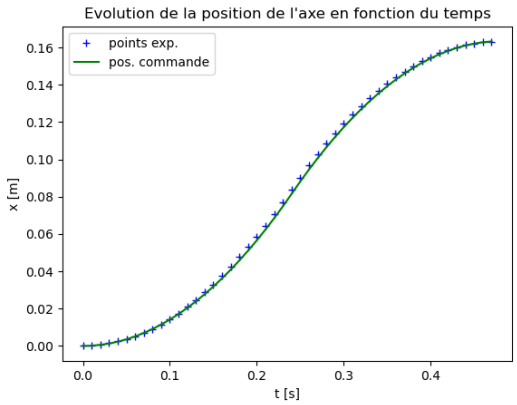
```
def Loi_Vitesse(t):
    global t1, Vmax, Acmax
    if 0<=t<t1:
        v=Acmax*t
    elif t1<=t<=t2:
        v=Vmax - Acmax*(t-t1)
    else:
        print("erreur")
    return v
```

```
def Loi_Position(t):
    global t1, Vmax, Acmax
    if 0<=t<t1:
        x=(Acmax/2)*t**2
    elif t1<=t<=t2:
        x=X1 + Vmax*(t-t1) - (Acmax/2)*(t-t1)↙
            **2
    else:
        print("erreur")
    return x
```

Question 11 *Construire deux tableaux* `X_th` *et* `V_th` *où sont stockées les positions théoriques commandées, respectivement vitesses théoriques aux instants définis dans le tableau* `temps`*. Superposer la courbe d'évolution de la position théorique sur les points expérimentaux, obtenus précédemment (tracé en vert, ligne continue, avec légende explicite). Vous produirez alors une image, que vous enverrez à votre enseignant et que vous appellerez tp04_nom1_nom2_Q11.png.*

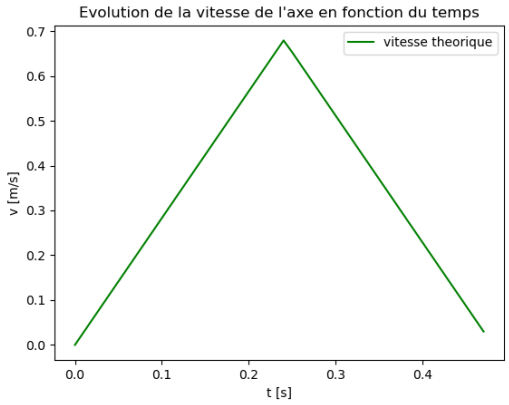
```
n = len(temps)
X_th = np.zeros(n); V_th = np.zeros(n)
for i in range(n):
    X_th[i] = Loi_Position(temps[i])
    V_th[i] = Loi_Vitesse(temps[i])


plt.plot(temps,X_th,'g',label="pos. commande")
plt.legend()
plt.savefig('tp04_durif_q11.png')
```



Question 12 *Sur une nouvelle figure tracer en vert, trait continu, l'évolution de la vitesse théorique. Vous produirez alors une image, que vous enverrez à votre enseignant et que vous appellerez tp04_nom1_nom2_Q12.png.*

```
plt.figure(2)
plt.plot(temps,V_th,'g',label="vitesse ↙
    theorique")
plt.legend()
plt.xlabel('t [s]'); plt.ylabel('v [m/s]')
plt.title("Evolution de la vitesse de l'axe en↙
     fonction du temps")
plt.savefig('tp04_durif_q12.png')
```



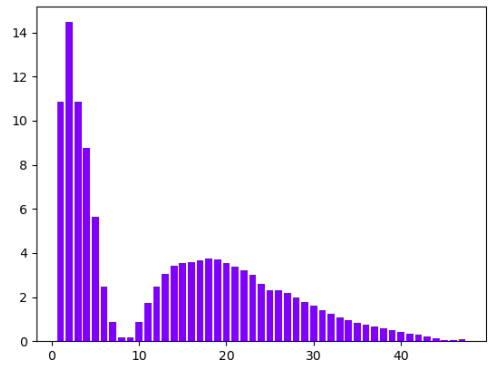
Question 13 *Concevoir une fonction* `Calcul_ecarts` *prenant en arguments deux tableaux à une dimension et retournant un tableau* `Delta_X` *de même dimension où sont stockés les écarts relatifs entre chacune des valeurs des deux tableaux spécifiés en arguments d'entrée.*

```
def Calcul_ecarts(T1,T2):
    return 100*abs((T1-T2)/T2)

Delta_X = Calcul_ecarts(X_exp[1:n],X_th[1:n])
```

Question 14 *Tracer un histogramme montrant l'évolution des écarts relatifs en position en fonction du numéro de la mesure (on utilisera* `plt.bar` *et* `plt.subplot`*). On n'évaluera pas l'écart relatif sur la première valeur (nulle). Vous produirez alors une image, que vous enverrez à votre enseignant et que vous appellerez tp04_nom1_nom2_Q14.png.*

```
plt.figure(3)
plt.bar([i for i in range(1,n)],Delta_X,color
    =[0.5,0,1])
plt.savefig('tp04_durif_q14.png')
```



Question 15 *Écrire une fonction* `Calculs_stats` *permettant, à partir d'un tableau* `T` *passé en argument, de retourner un tuple de 3 valeurs : moyenne, médiane et écart type. Indication : On pourra utiliser les fonctions de la bibliothèque numpy (*`np.sum(T)`*, pour faire la somme de tous les éléments du tableau* `T`*,* `np.sort(T)`*, pour trier le tableau* `T` *dans l'ordre croissant. Appliquer le résultat au tableau* `Delta_X`*.*

```
def Calculs_stats(T):
    n = np.shape(T)[0]
    #Moyenne :
    moy_T = np.sum(T)/n
    #Médianne :
    T_trie = np.sort(T)
    if n==0:
        med_T = 0
    elif n%2==0:
        med_T = (T_trie[n//2]+T_trie[n//2-1])/2
    else:
        med_T = T_trie[n//2]
    #Ecart type :
    sigma_T = np.sqrt(np.sum((T - moy_T)**2)/n)

    return (moy_T, med_T, sigma_T)

(a,b,c) = Calculs_stats(Delta_X)
```

Question 16 *Comparer les résultats en utilisant les fonctions suivantes :* `np.mean`*, pour la moyenne ;* `np.median`*, pour la médiane et* `np.std`*, pour l'écart type*

```
print(np.mean(Delta_X))
print(np.median(Delta_X))
print(np.std(Delta_X))
```