Séance 05 :

Euler : Ouvre-portail

1. Mise en situation

C.F. sujet.

2. But de l'étude

C.F. sujet.

3. Données de l'étude

C.F. sujet.

4. Travail demandé

1. ***Ecrire une fonction qui prend en argument la vitesse de rotation du bras moteur*** `wm` ***un vecteur*** `Tm` ***contenant des valeurs de*** $\theta_m(t)$ ***et qui renvoie des vecteurs*** `A`, `B`, `Ap`, `Bp` ***et*** `Cp` ***contenant*** $A(\theta_m)$, $B(\theta_m)$, $\dot{A}(\theta_m)$, $\dot{B}(\theta_m)$ ***et*** $\dot{C}(\theta_m)$.
   Je commence par renommer les fonctions *cos* et *sin* :

```python
# Fonctions circulaires
def cos(a):
    return np.cos(a)


def sin(a):
    return np.sin(a)
```

   Ensuite, comme à mon habitude, je crée un dossier `ressources` dans lequel je crée le fichier `data.py` :

```python
a = 100 * 10 ** (-3)
b = 260 * 10 ** (-3)
c = 20 * 10 ** (-3)
d = 324.22 * 10 ** (-3)
l = 280 * 10 ** (-3)
```

   J'importe tout ce dont j'ai besoin au début de mon fichier :

```python
# Imports
import numpy as np
import matplotlib.pylab as plt

# Donnees
from ressources.Data import *
```

   Puis je calcule les données demandés :

```python
def motorSpeed(wm, Tm):
    A = 2 * (a * d - b * c - d * l * cos(Tm) - c * l * sin(Tm))
    B = 2 * (-a * c - b * d - d * l * sin(Tm) + c * l * cos(Tm))
    C = a**2 + b**2 + c**2 + d**2 + 2 * l * (b * sin(Tm) - a * cos(Tm))
    Ap = 2 * l * wm * (d * sin(Tm) - c * cos(Tm))
    Bp = 2 * l * wm * (- d * cos(Tm) - c * sin(Tm))
    Cp = 2 * l * wm * (a * sin(Tm) + b * cos(Tm))

    return (A, B, C, Ap, Bp, Cp)
```

2. ***En utilisant la fonction précédente, écrire une fonction qui prend en argument la vitesse de rotation du bras du moteur*** `wm`***, l'angle de rotation maximal*** `Tmmax` ***du bras du moteur, le nombre*** `N` ***de points désirés et qui renvoie deux vecteurs, l'un,*** `Tm` ***contenant les valeurs de*** $\theta_m(t)$ ***et l'autre,*** `Tpv`***, contenant les valeurs de*** $\dot{\theta}_v(t)$***.***

   D'abord, pour alléger la fonction demandée, je crée une fontion `Tpv` qui calcule $\dot{\theta}_v$:

```
def Tpv(Tm, Tv, wm):
    A, B, C, Ap, Bp, Cp = temporalData(wm, Tm)
    return (Ap * cos(Tv) + Bp * sin(Tv) + Cp) / (A * sin(Tv) - B * cos(Tv))
```

   Puis je calcule les deux vecteurs demandés :

```
def angle(wm, Tmmax, N):
    _Tm = np.linspace(0, Tmmax, N)
    h = Tmmax / (wm * N)
    _Tpv = [Tpv(0, 0, wm)]
    _Tv = [0]
    for i in range(1, N):
        _Tv.append(_Tv[i-1] + _Tpv[i-1] * h)
        _Tpv.append(Tpv(_Tm[i-1], _Tv[i-1], wm))
    return (_Tm, _Tpv)
```

3. ***Tester cette fonction pour une valeur de*** ω_m ***de*** $0,1 rad.s^{-1}$***, et un nombre de point par défaut*** $N = 1000$***. Faire afficher le graphe de*** $\dot{\theta}_v$ ***en fonction de*** θ_m ***et comparer avec les valeurs obtenues expérimentalement :***

$$\begin{cases} \dot{\theta}_{v_{maxi}} &= 0,088 rad.s^{-1} \\ \theta_m &= 0,46 rad \end{cases}$$

   Je crée la petite fonction de conversion de dergé vers radian :

```
def degTorad(deg):
    return deg * np.pi / 180
```

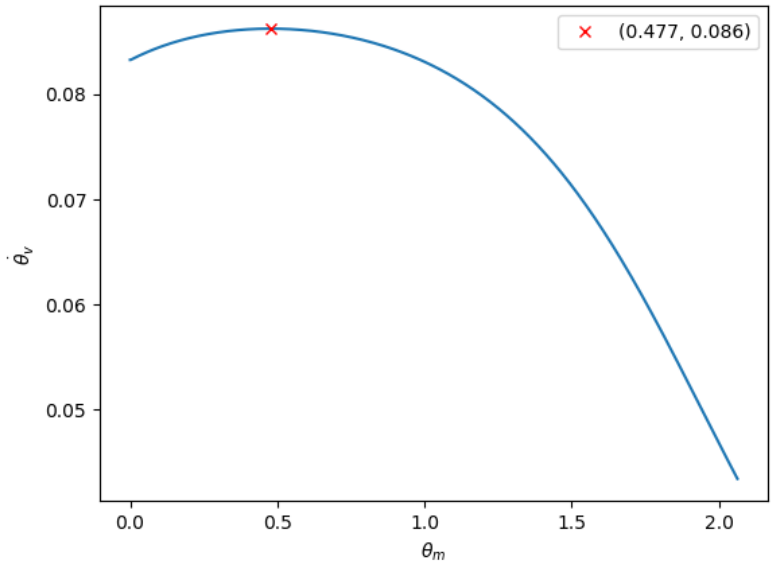
   Puis j'applique le script :

```
# On calcule les coordonnees
G, H = angle(0.1, degTorad(118.2), 1000)

# On cherche le maximum pour le comparer aux valeurs exp.
i = H.index(max(H))

# On trace tout ca
plt.plot(G, H)
plt.xlabel("$\\theta_m$")
plt.ylabel('$\dot{\\theta}_v$')
plt.plot(G[i], H[i], 'rx', label=f'({G[i]:.3f}, {H[i]:.3f})')
plt.legend()
plt.show()
```

   Et j'obtiens :

$$\begin{cases} \dot{\theta}_{v_{maxi}} &\simeq \boxed{0,086}\, rad.s^{-1} \\ \theta_m &\simeq \boxed{0,477}\, rad \end{cases}$$

   ce qui est proche des valeurs expérimentales.



4. ***En utilisant les fonctions précédentes et en faisant évoluer*** ω_m ***sur l'intervalle*** $[0.1, 0.3]$ ***par pas de*** 0.01***, écrire le programme principal qui permet de trouver la valeur maximale de*** ω_m ***vérifiant le critère du cachier des charges (vitesse maximale autorisée par la norme pour le point le plus rapide du portail :*** $0,25m.s^{-1}$***). Rappel :*** $|V_{D,3/0}| = OD.|\dot{\theta}_v|$***.***

   On veut la valeur $|\dot{\theta}_v|$ maximale, tout en respectant $|V_{D,3/0}| \leq 0.25m.s^{-1}$.
   On rajoute la valeur maximale de $L = OD$, de $V_{D,3/0} = 0.25m.s^{-1}$ et de $\theta_{m\_maxi} = 118.2°$ dans le fichier `ressources/Data.py`

```
L = 2.2
Tmmax = 118.2 * np.pi / 180
Vmax = 0.25
```

   Fonction principale :

```
def getMaxSpeed(borneMin, borneMax, pas, N):
    Vd, wm = 0, borneMin

    # Pour sauvegarder les anciennes valeurs
    Vd_o, wm_o = 0, borneMin

    while (Vd < Vmax and wm <= borneMax):

        # On sauvegarde les anciennes valeurs
        # Sinon on fait un tour de trop
        Vd_o, wm_o = Vd, wm

        TpvMax = abs(max(angle(wm, Tmmax, N)[1]))
        Vd = L * TpvMax
        wm += pas

    return (Vd_o, wm_o)
```

   Et, en exécutant :

```
print(getMaxSpeed(0.1, 0.3, 0.01, 1000))
```

   On obtient :

$$\begin{cases} \dot{\theta}_{v_{maxi}} &= \texttt{0.14}\ rad.s^{-1} \\ V_{D,3/0} &= \texttt{0.247}\ m.s^{-1} \end{cases}$$