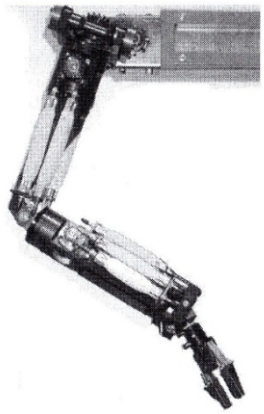
TD4 : Bras artificiel : Correcteur numérique

1- Présentation du système

Mise en situation



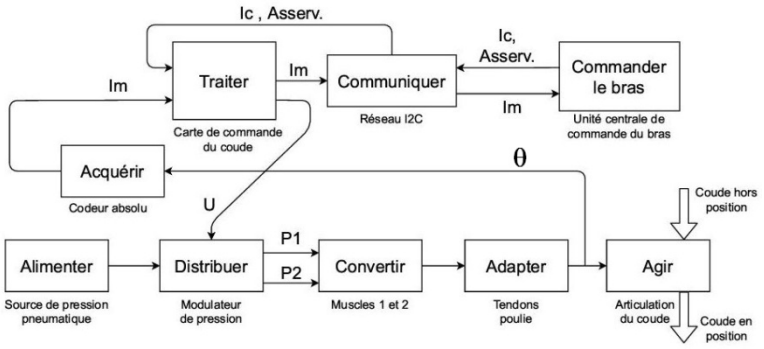
Cette étude concerne un manipulateur développé en laboratoire. Il se caractérise par une structure anthropomorphique à 7 degrés de liberté activés par des paires de muscles montés en opposition.

Un muscle est constitué d'une vessie en caoutchouc emprisonnée dans une tresse de fils. L'angle d'inclinaison de cette tresse permet de convertir le gonflement de la vessie, sous l'effet de la pression, en effort de traction.

La modulation de pression, réalisée à partir d'une tension de commande u(t), permet alors de faire varier l'effort de traction. En associant deux muscles en opposition, on peut ainsi activer une articulation à l'aide d'un fil (tendon) relié aux deux extrémités des muscles et roulant sans glisser sur une poulie.

Structure de l'articulation du coude

La chaine fonctionnelle du coude est décrite par le schéma ci-contre :



On s'intéresse ici à la carte de commande du coude qui assure la commande de l'articulation du coude. Il s'agit d'un microprocesseur dont le code est écrit en Python.

La carte de commande est en communication avec l'unité centrale de commande du bras par un réseau « I2C ».

- ☞ La commande « Acqu_Ic_Ass() » permet d'interroger l'unité centrale qui répond en renvoyant la consigne et si le coude doit être asservi en position. Cette fonction renvoie donc un entier (image de la position de consigne du coude) et un booléen qui indique si le système doit être asservi.
- ☞ La commande « Informer(Im) » permet d'envoyer sur le réseau « I2C » une trame à l'unité centrale qui indique la position réelle mesurée par le codeur absolu.
- ☞ La commande « Acqu_Position() » permet d'interroger le codeur absolu qui renvoie l'image de la position réelle du coude : Im. Cette fonction retourne donc l'entier Im image de θ.
- ☞ La commande « Envoyer_commande(U) » permet d'attribuer à la sortie de la carte de commande reliée au modulateur de pression, une tension U qui assure la commande du modulateur de pression.

Code du microprocesseur

Le code (incomplet) du microprocesseur est donné ci-dessous. On ne donne que la fonction principale qui est exécuté en boucle lorsque la carte de commande est active.

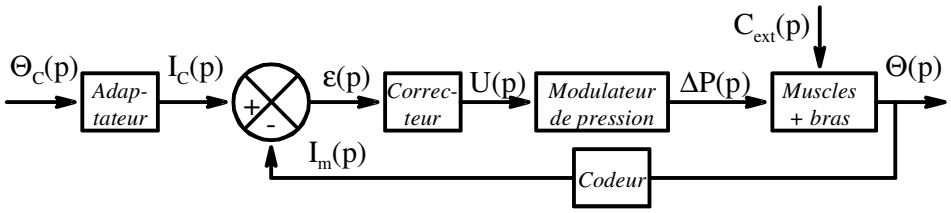
```
def asservissement():
    # Toutes les variables sont globales et sont initialisées à 0
    global Date,Datep,dt,Ic,Im,E,Ep,U1,U1p,U2,U2p,U,Up,M,Mp,Cext,Cons,Te,Tep,Tepp
    Cons=0.0                # Consigne de l'asservissement (Cons)
    Ic,Im=0,0               # Images de la consigne (Ic) et de la réponse (Im)
    E,Ep=0.0,0.0            # Ecart entre les images de la consigne et de la réponse
    U1,U1p=0.0,0.0          # Sortie du 1ier correcteur
    U2,U2p=0.0,0.0          # Sortie du 2nd correcteur
    U,Up=0.0,0.0            # Sortie du 3ième correcteur
    Date,Datep=0.0,0.0      # Dates actuelle et précédentes
    Asservissement = True   # Booléen indiquant si l'articulation doit être asservie
```

```
While Asservissement==True:  # Boucle d'asservissement
    Datep=Date               # Memorisation de la date précédente
    Date=t.time()   # Acquisition de la date de l'horloge de la carte de commande
    dt=Date-Datep   # Calcul de la durée d'échantillonnage
    Ic,Asservissement=Acqu_Ic_Ass()  # Acquisition des consignes
    Im=Acqu_Position()    # Pour mesurer l'image de la réponse à la Date actuelle
    Ep=E                  # Memorisation de l'écart précédent
    E=Ic-Im               # Calcul de l'écart actuel
    Calcul_Commande()     # Fonction à définir suivant le correcteur
    Envoyer_commande(U)   # Envoie de la commande au modulateur de pression
    t.sleep(0.005)        # Temporisation de 5 ms

Envoyer_commande(0)       # Pour annuler la commande
```

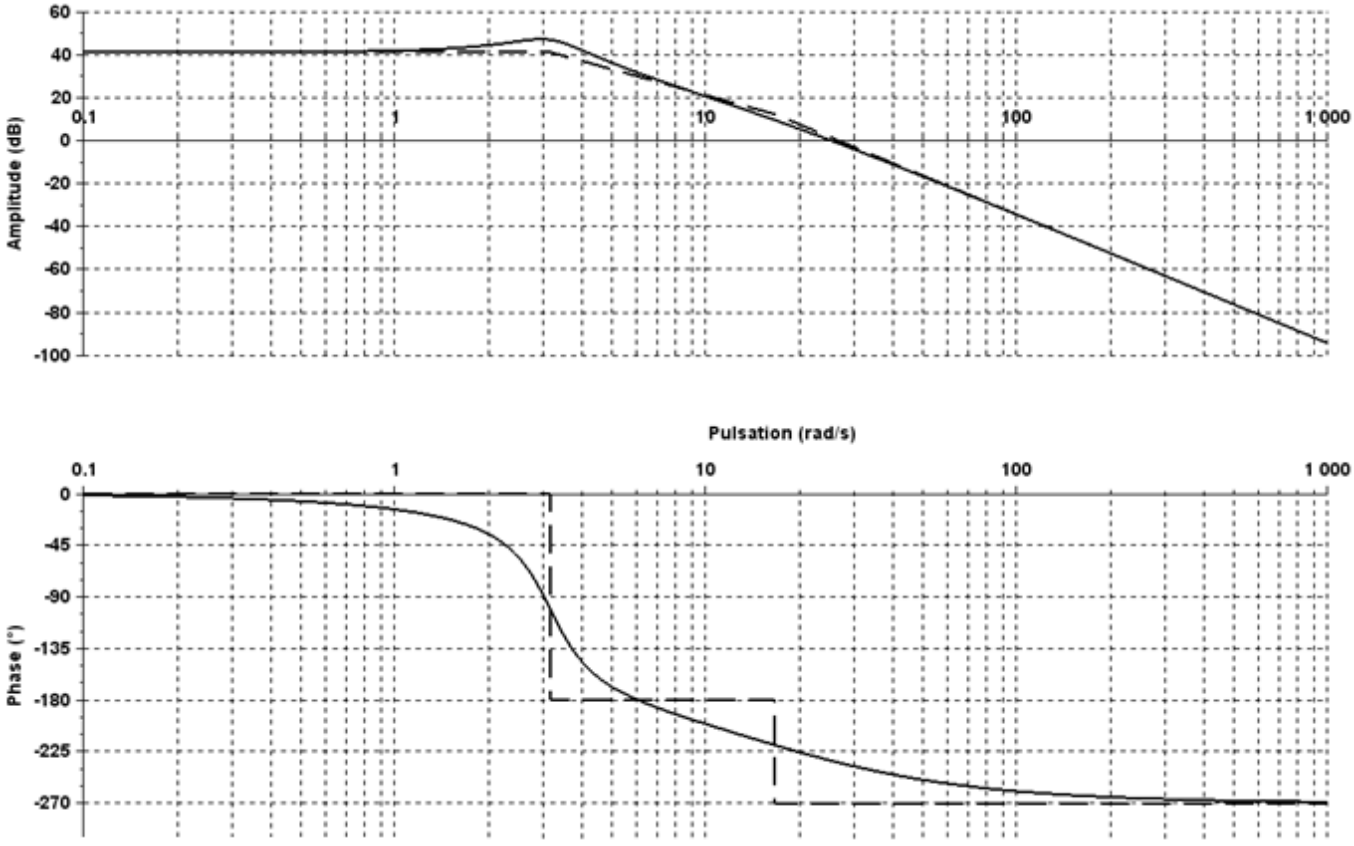
Modélisation de l'asservissement

Une première étude a permis d'obtenir la structure de l'asservissement ci-dessous :



Une seconde étude a permis de déterminer la fonction de transfert non corrigée du système :

$H_{BONC}(p) = \dfrac{114{,}6}{(1 + 0{,}16.p + 0{,}1.p^2).(1 + 0{,}06.p)}$ On a ainsi le diagramme de Bode de cette FTBO



Objectif du problème

L'objectif du problème est de coder le correcteur. C'est-à-dire écrire en langage Python la fonction « Calcul_Commande() » qui permet d'obtenir la tension u(t) à la sortie du correcteur (la variable « U »). Laquelle commande permet de vérifier le cahier des charges :

$\mathbf{M_\varphi \geq 45°}$ $\mathbf{\omega_{0dB} \geq 8\ rad.s^{-1}}$ $\mathbf{t_{5\%} \leq 2\ s}$ $\mathbf{\varepsilon_S = 0°}$ $\mathbf{\varepsilon_t \leq 5° = 8{,}7.10^{-2}\ rad}$.

Travail demandé

A- Correcteur proportionnel

1- On envisage pour commencer un correcteur proportionnel : $C(p) = K_p$. Déterminer la valeur de K_p permettant de garantir une marge de phase de 45°.

2- Compléter le code de la fonction **`Calcul_Commande1()`** (Une seule ligne suffit) qui à partir de l'écart E calcule la commande U (E et U variables globales).

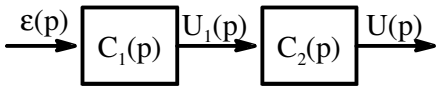
```
def Calcul_Commande1():
  global U, E
```

2- Taper ce code dans le code Python fourni et tester cette correction avec une perturbation de 0 N.m et 5 N.m. Pour cela taper dans le Shell les commandes **`test(calcul_Commande1,0,14,1000)`** et **`test(calcul_Commande1,5,14,1000)`** (14 est la durée de simulation et 1000 le nombre points)

4- Conclure quand au respect des critères de précision et de rapidité du cahier des charges.

B- Correcteur à deux étages

A partir du diagramme de Bode de la FTBO non corrigé et du cahier des charges, une 3ième étude montre qu'ilest envisageable d'utiliser un correcteur à 2 étages $C(p) = C_1(p) . C_2(p)$:



Avec : $C_1(p) = \dfrac{1 + 1{,}17.p}{1 + 0{,}014.p}$

et : $C_2(p) = \dfrac{0{,}016.(1 + 0{,}343.p)}{p}$

5- Ecrire les relations numérique (avec la variable de Laplace) liant les fonctions symboliques, du correcteur : ε(p), $U_1(p)$ et U(p). Passer ces relations dans le domaine temporel. Discrétiser ces relations : $\dfrac{d\,f(t)}{dt} = \dfrac{f(t) - f(t-dt)}{dt}$ et en déduire les expressions numériques de $u_1(t)$ et u(t) en fonction de ε(t), ε(t–dt), $u_1(t)$, $u_1(t-dt)$ et u(t–dt).

6- Compléter le code de la fonction **`Calcul_Commande2()`** qui à partir de l'écart E calcule la commande U (E et U variables globales).

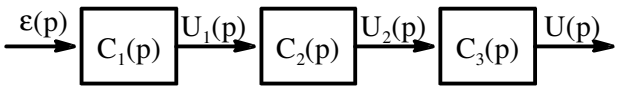
```
def Calcul_Commande2():
  global dt, E, Ep, U1, U1p, U
```

7- Taper ce code dans le code Python fourni et tester cette correction avec une perturbation de 0 N.m et 5 N.m. Pour cela taper dans le Shell les commandes **`test(calcul_Commande2,0,14,1000)`** et **`test(calcul_Commande2,5,14,1000)`**

8- Conclure quand au respect des critères de précision et de rapidité du cahier des charges.

C- Correcteur à trois étages

A partir du diagramme de Bode de la FTBO non corrigé et du cahier des charges, une 3ième étude montre qu'il faut utiliser un correcteur à 3 étages $C(p) = C_1(p) . C_2(p) . C_3(p)$:



Avec : $C_1(p) = C_2(p) = \dfrac{1 + 0{,}26.p}{1 + 0{,}06.p}$

et : $C_3(p) = \dfrac{0{,}034.(1 + 0{,}343.p)}{p}$

9- Reprendre les questions 5, 6 et 7 avec la fonction **`Calcul_Commande3()`** .

```
def Calcul_Commande3():
  global dt, E, Ep, U1, U1p, U2, U2p, U
```

10- Conclure quant au respect des critères du cahier des charges avec cette commande.