5.4 Correction numérique des systèmes asservis

Sommaire

1.Analyser : Chaîne de commande d'un système asservi 2

2. Implémentation d'un correcteur numérique 2

   2.1 Tracé de la FTBF avec $\boldsymbol{C(p) = 1}$ 3

   2.2 Tracé de la FTBF avec un correcteur PI 4

3. Analyser : Effets des différents types de correcteurs 4

1.Analyser : Chaîne de commande d'un système asservi

Q1. Recopier et remplir le schéma de commande suivant avec les éléments suivants :

- CAN ;
- CNA ;
- Capteur ;
- PI.

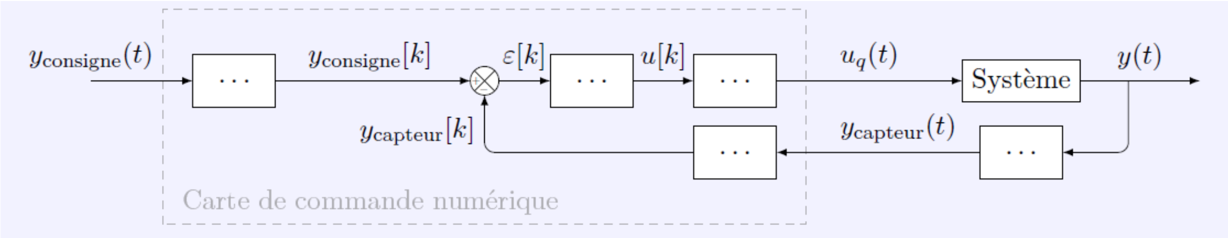


Figure 1 : schéma blocs d'un asservissement de position avec correcteur numérique

Dans la suite de ce document, on confondra signal échantillonné et signal numérique, en faisant l'hypothèse que la mémoire allouée pour la variable mesurée est infinie. Cette hypothèse est souvent prise en compte, ce qui n'empêche pas d'étudier à part l'impact d'une taille de mémoire finie sur la résolution de mesure.

2. Implémentation d'un correcteur numérique

On considère le système en boucle fermée à retour unitaire possédant un correcteur PID numérique suivant (*cf. Figure 2*) :

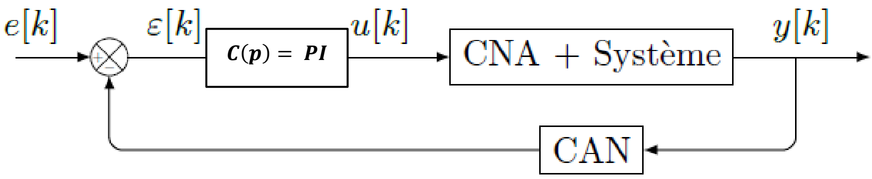


Figure 2 : Système du premier ordre en boucle fermée avec correcteur PI en commande numérique

<u>Objectif</u> : L'objectif de cette partie est d'implémenter un correcteur PI par des équations de récurrence sous Python.

Il va donc falloir définir une fonction PI, qui prend pour variable d'entrée l'écart numérique $\varepsilon[k]$ et renvoie la consigne de commande numérique $u[k]$.

2.1 Tracé de la FTBF non corrigée : $C(p) = 1$

Informatique : Traitement de données sous Python

Q2. Recopier les lignes de code suivantes en prenant soin de remettre en place l'indentation.

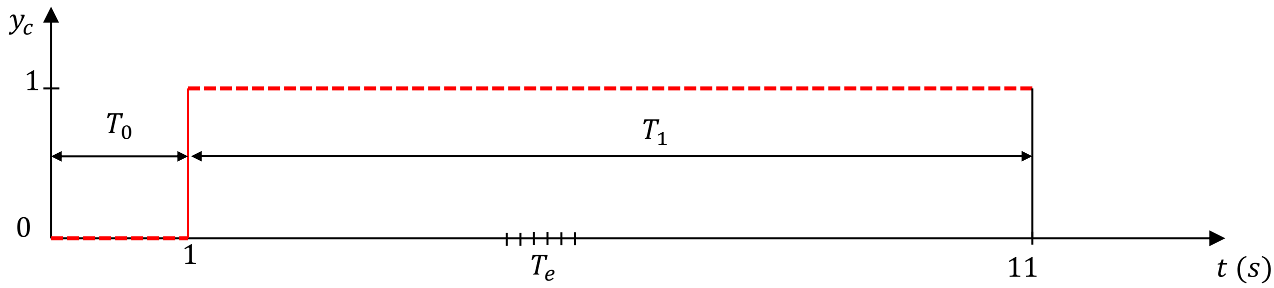
Lignes de code (fichier texte disponible avec le Td : *54_lignes_Q2.txt*

```
import math as ma
import matplotlib . pyplot as plt
import numpy as np
# Conditions initiales
Te = 1e -1
K = 1
tau = 3
y0 = 0
Kp = 0 # gain du correcteur proportionnel
Ki = 0 # gain du correcteur intégral


# Initialisation du temps et de la consigne d'entrée Q3 et Q4
# A COMPLETER
#t =
#yc =

def quantification (s1,s2,t):
S1 =[s1 [0]]
S2 =[s2 [0]]
T=[t[0]]
for k in range (1, len (s1)):
S1. append (s1[k -1])
S1. append (s1[k])
S2. append (s2[k -1])
S2. append (s2[k])
T. extend ([t[k],t[k]])
return (S1,S2,T)

def BF_1Ordre_z (K,tau ,T,y0 ,yc):
y=[ y0]
eps =[ yc [0]]
for k in range (len(yc) -1):
eps . append (yc[k+1] -y[k])
y. append (y[k]* ma.exp(-T/tau) + K* eps[k]*(1 - ma. exp(-T/ tau)))
return (y)

yc_BF ,y_BF , t_BF = quantification (yc , BF_1Ordre_z (K,tau ,Te ,y0 ,yc),t)

# Affichage des courbes Q5
# A COMPLETER
```

Q3. Définir un vecteur (*de type : np.array*) temps t de onze secondes, sachant que le temps d'échantillonnage est de :

$$T_e = 1\,e^{-1}\,s$$

Q4. Définir yc un vecteur (*de type : np.array*) représentant le signal de consigne tel que ce soit un échelon unitaire commençant à l'instant $t = 1s$.



yc sera construit à partir de deux vecteurs mis bout à bout ($concaténation^*$) :

- Le premier vecteur sera représenté par un tableau unidimensionnel de $zéros^*$ dont le nombre sera calculé à partir de la période T_0 échantillonnée à T_e.
- Le deuxième vecteur sera représenté par un tableau unidimensionnel de uns^* dont le nombre sera calculé à partir de la période T_1 échantillonnée à T_e.

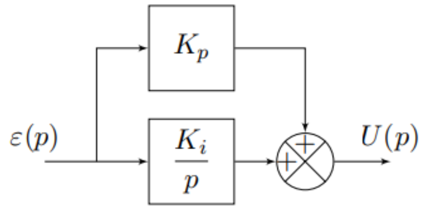
Q5. Tracer le signal de sortie y et la consigne d'entrée yc en fonction du temps t. Ajouter des « *label* » afin de pouvoir identifier les courbes.

(*) : voir les possibilités offertes par la librairie numpy

2.2 Tracé de la FTBF avec un correcteur PI

<u>Informatique</u> : Implémenter un correcteur PI numériquement.

On utilise un correcteur PI défini par le schéma bloc suivant :



$$C(p) = \frac{U(p)}{\varepsilon(p)} = K_p + \frac{K_i}{p}$$

On utilisera les coefficients de correcteur suivants :

- $Kp = 8$;
- $Ki = 1$;

Q6. Retrouver l'équation de récurrence de ce correcteur :

$$u_k = \cdots$$

(Prendre exemple sur le cours).

Pour les questions suivantes s'aider des lignes de code données ci-dessous.

Lignes de code correcteur PI (fichier texte disponible avec le Td : *54_lignes_PI.txt*)	
1 **def** PI (...) : # Q7 2 . 3 . 4 . 5 **return** (...) # retourne le vecteur de commande numérique du système 6 7 **def** BF_1Ordre_PI_z (K,tau ,Te ,y0 ,yc ,Kp ,Ti): 8 y=[y0] # sortie du système (voir figure 2 du TD) 9 eps =[yc[0] -y[0]] # epsilon sortie du comparateur 10 u=[Kp*eps [0]] # sortie du correcteur (voir figure 2 du TD) 11	12 13 y. append (y [0]* ma.exp(-Te/tau) + K*u [0]*(1 - ma. exp(-Te/ tau))); 14 **for** k **in range** (1, **len** (yc) -1): 15 eps . append (yc[k]-y[k]) 16 ... = PI (...) 17 y. append (y[k]* ma.exp(-Te/tau) + K*u[k]*(1 - ma. exp(-Te/ tau))) 18 **return** (y) 19 20 yc_PI ,y_PI , t_PI = quantification (yc , BF_1Ordre_PI_z (K,tau ,Te ,y0 ,yc ,Kp ,Ti),t) 21 22 # Affichage des courbes #Q9 23 # A COMPLETER

Q7. Définir une fonction PI, utilisant l'équation de récurrence du correcteur proportionnel intégral définit à la question précédente, qui devra retourner le vecteur u représentant la consigne de commande numérique.

<u>On notera</u> :

- eps_k : la variable représentant ε_k
- eps_km1 : la variable représentant ε_{k-1}

Q8. Compléter une nouvelle fonction $BF\_1Ordre\_PI\_z$ permettant d'implémenter le correcteur.

Q9. Afficher ensuite la consigne, la FTBF non corrigée et la FTBF corrigée sur le même graphique avec un correction Proportionnelle seule et une correction Proportionnelle Intégrale (Ajouter des « *label* » afin de pouvoir identifier les courbes.

3. Analyser : Effets des différents types de correcteurs

Q10. Observer et discuter des effets sur les performances :

- d'un correcteur proportionnel P : $Kp = 8$;
- d'un correcteur proportionnel intégral PI : $Kp = 8$ et $Ki = 1$;