Rapport TP3 Automatique

Commande numérique : PI numérique

Vitrat Romain Bourlot Xavier

4AE-SE, Binôme 2

30 mai 2020

Table des matières

1 Introduction

Le but de ce TP est d'effectuer une régulation d'un niveau d'eau par PI numérique. On commande une pompe qui remplit un réservoir fuyant, le but étant d'asservir la hauteur d'eau dans le réservoir. Le schéma bloc de la manipulation est présenté figure 7.

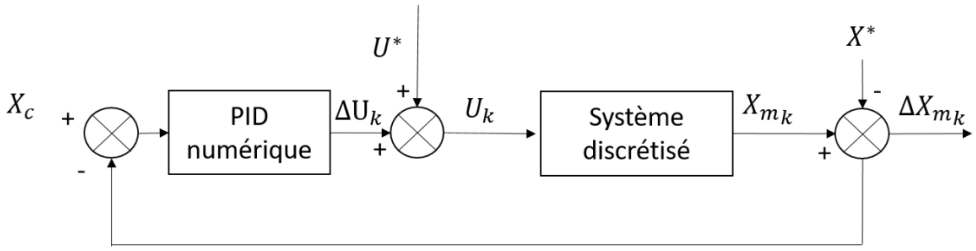


FIGURE 1 – Schéma bloc du système

La simulation est effectuée avec Matlab et la régulation est faite sous XPC Target.

2 Identification du processus

Le processus au complet étant non linéaire (du fait de la fuite d'eau turbulente, en $\sqrt{X}$, X la hauteur d'eau dans le réservoir), on linéarise le modèle autour d'un point de fonctionnement, ici $U^* = 5V$, U étant la tension de commande en entrée de la pompe.
La fonction de transfert autour du point de fonctionnement (U^*, X_m^*) est alors :

$$G(s) = \frac{\Delta X_m(s)}{\Delta U(s)} = \frac{K}{1+Ts} \tag{1}$$

Pour déterminer les paramètres K et T du système, on étudie la réponse à des échelons de position.
Pour cela, on évalue d'abord le comportement statique au point de fonctionnement. Ainsi, on mesure pour $U^* = 5V$ une sortie $X_m^* = 4.333V$.
On a tenté de réaliser un échelon de 1V, mais le système sature (réservoir en limite de capacité) et les résultats ne sont alors pas exploitables. Un échelon de 0.5V permet d'obtenir la réponse suivante :

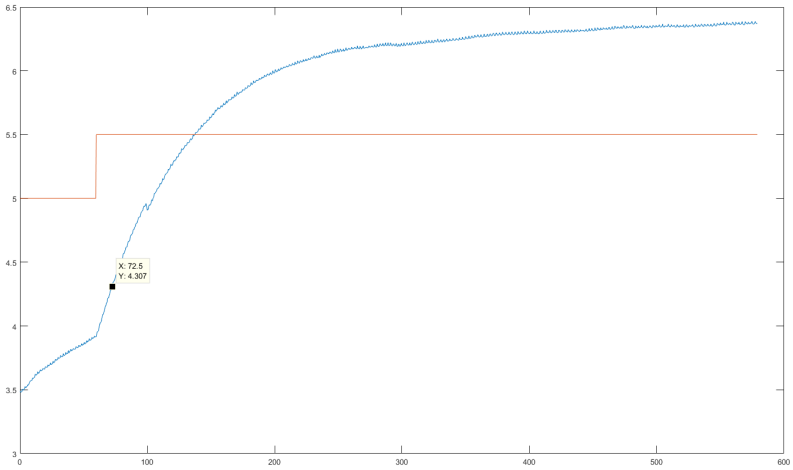


FIGURE 2 – Réponse à un échelon de $0.5V$, en bleu la sortie, en orange l'entrée du système

On peut alors mesurer sur cette réponse le gain du système et le temps de réponse :

$$\begin{cases} K = \frac{X_{mfinal} - X_m^*}{\Delta U} = \frac{6.667 - 4.333}{0.5} = 4.068 \\ T = temps(63\% X_{mfinal}) = temps(63\% 2.034) = 93.5s. \end{cases} \tag{2}$$

Une des difficultés rencontrées lors de ces manipulations a été le temps de réponse très long du système, qui nécessitait de longues acquisitions et d'avoir un régime permanent bien établi avant de commencer un échelon (ce qui n'est pas le cas sur la Fig.2).

3 Mise en place d'un correcteur proportionnel

3.1 Etude théorique

On discrétise le système en utilisant la discrétisation avant, Te est choisi dans un premier temps à 4s.

$$G(z) = \frac{K}{1 + T(\frac{z-1}{T_e})} = \frac{K}{1 - \frac{T}{T_e} + \frac{T}{T_e}z} \tag{3}$$

d'où le le système en boucle fermée :

$$G_{BF}(z) = \frac{K_p K}{1 - \frac{T}{T_e} + \frac{T}{T_e}z + K_p K} \tag{4}$$

On cherche le gain K_p critique d'auto-oscillation. On a

$$K_p = \frac{1}{K}(\frac{2T}{T_e} - 1)$$
soit pour $T_e = 4s$
$$K_p = \approx 11.3$$

3.2 Simulation

Dans le but de paramétrer un correcteur optimum en stabilité et rapidité, nous avons utilisé l'outil sisotool. Celui-ci nous a permis de dégager un Kp optimal en jouant sur la rapidité du système (plus proche du centre du cercle unitaire).

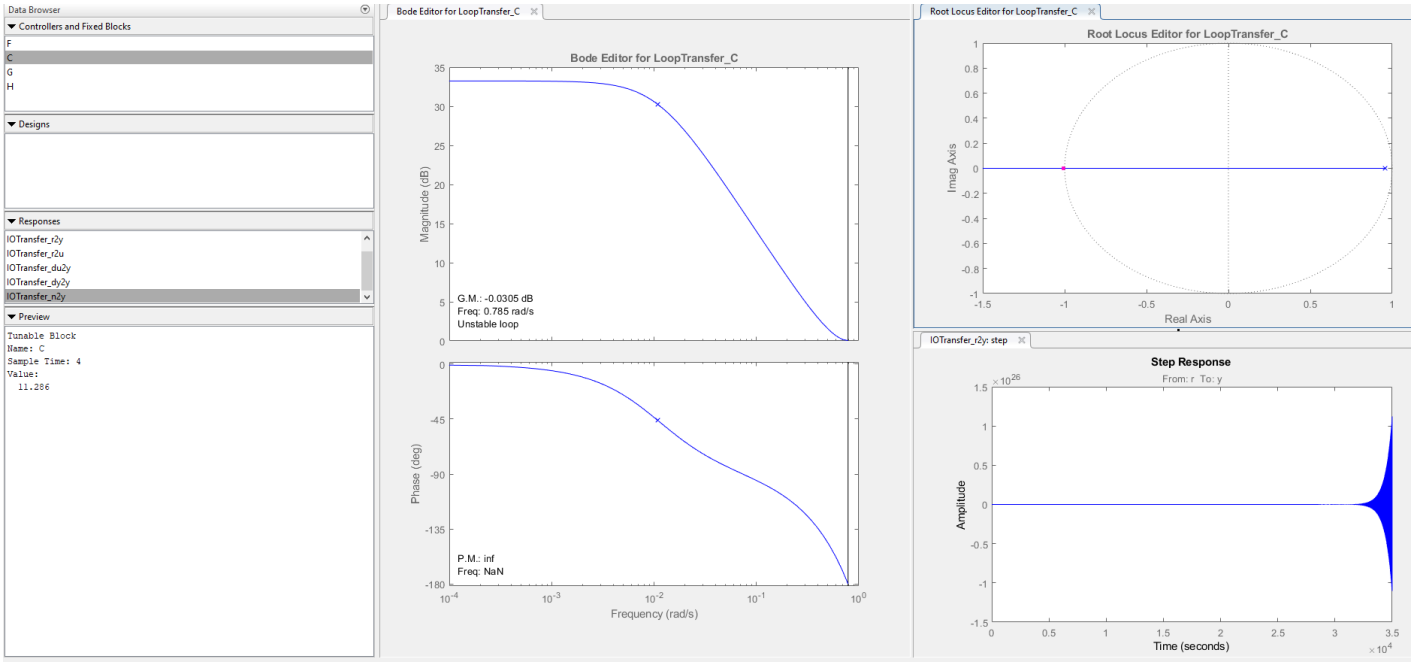


FIGURE 3 – Résultats des simulations sur l'outil sisotool, ($Kp = 11.2$)

Le système Fig.3 est en limite d'instabilité. Cela est en adéquation avec la première approche théorique, qui nous donnait un $Kplim$ de 11.3.

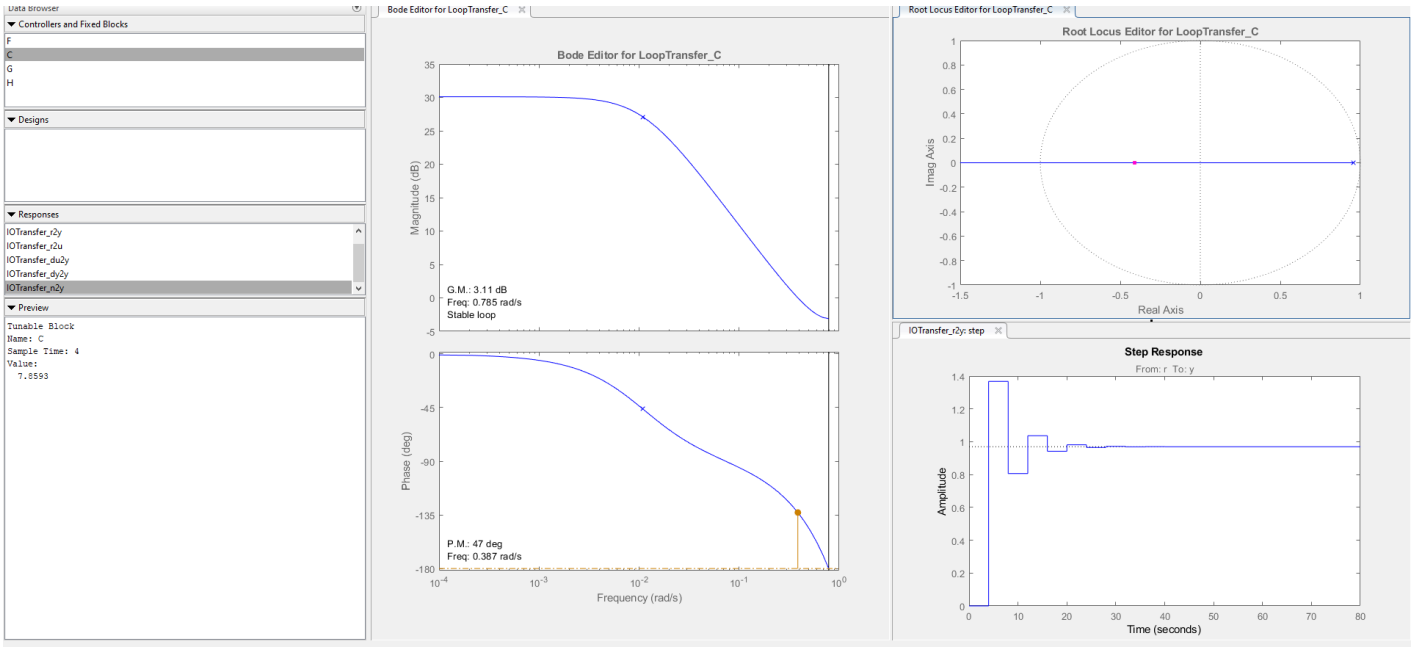


FIGURE 4 – Résultats des simulations sur l'outil sisotool, avec un $Kp = 7.8593$, meilleur compromis de stabilité et de rapidité.

On en dégage : $Kp \approx 7.9$.

3.3 Réalisation

L'interface avec le procédé est réalisée par un cible sous Simulink Real Time équipée d'une carte d'entrée/sortie NI. Le schéma Simulink correspondant est le suivant :

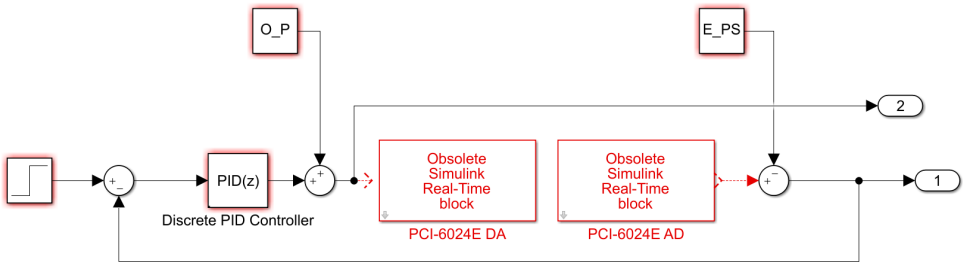
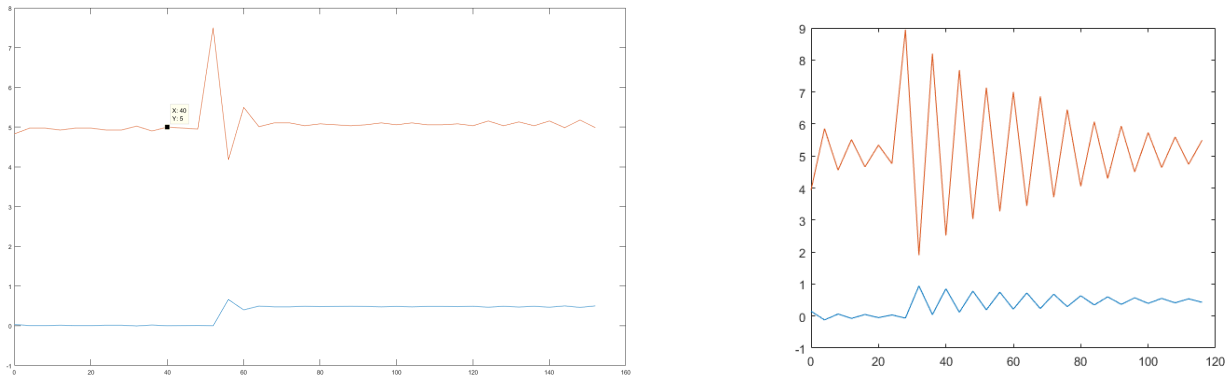


FIGURE 5 – Schéma Simulink de la cible

On applique sur le système réel notre correcteur proportionnel, avec différentes valeurs de K_p.



(a) $K_p = 5$

(b) $K_p = 7$

FIGURE 6 – Réponse à une échelon de 0.5V pour $Te = 4s$ (en orange la commande et en bleu la sortie)

On observe bien que plus K_p augmente plus le système approche de l'instabilité. Lorsque $K_p = K_{plim}$ le système oscille. L'erreur théorique en régime permanent vaut : $\epsilon = \frac{K_p K}{1+K_p K}$.
Les valeurs mesurées en réel sont comparées aux valeurs théoriques dans la table 1.
En raison du temps de réponse du système et de la faible fréquence d'échantillonage, il est difficile d'obtenir des valeurs fiables d'erreur en régime permanent.

K_p	$\epsilon_{theorique}$	ϵ_{exp}	$\Delta\epsilon(\%)$
5	0.9535	≈ 0.94	1.4
7	0.966	≈ 0.96	0.62
10	0.976	N/A	?
50	instable	instable	?

TABLE 1 – Comparaison des erreurs théoriques et expérimentales en régime permanent

4 Mise en place d'un correcteur PI

Afin de corriger cette erreur statique on se propose d'ajouter une composante intégrale au correcteur.

4.1 Etude théorique

Toujours en se basant sur la forme continue des correcteurs, on se propose d'étudier l'influence du correcteur proportionnel intégral $Ci(s) = Kp + \frac{Ki}{s}$. Toujours en utilisant une discrétisation avant, on obtient le correcteur suivant :

$$\begin{cases} Ci(z) = \dfrac{Kp*(z+\dfrac{Ki*Te-Kp}{Kp})}{z-1} = \dfrac{Kpi*(z+zpi)}{z-1} \end{cases} \quad Avec \begin{cases} \text{zpi}=\dfrac{Ki*Te-Kp}{Kp} \\ \text{Kpi=Kp.} \end{cases} \tag{5}$$

4.2 Simulation

On recommence la démarche précédente appliquée au correcteur PI. En utilisant sisotool, on détermine donc les meilleures valeurs des coefficients Kpi et zpi. Il y a cette fois possibilité de jouer avec les pôles en Bf et le zéro du correcteur, en plus du gain. Sachant que le pôle est fixé à 1 par définition, celui-ci est immuable.

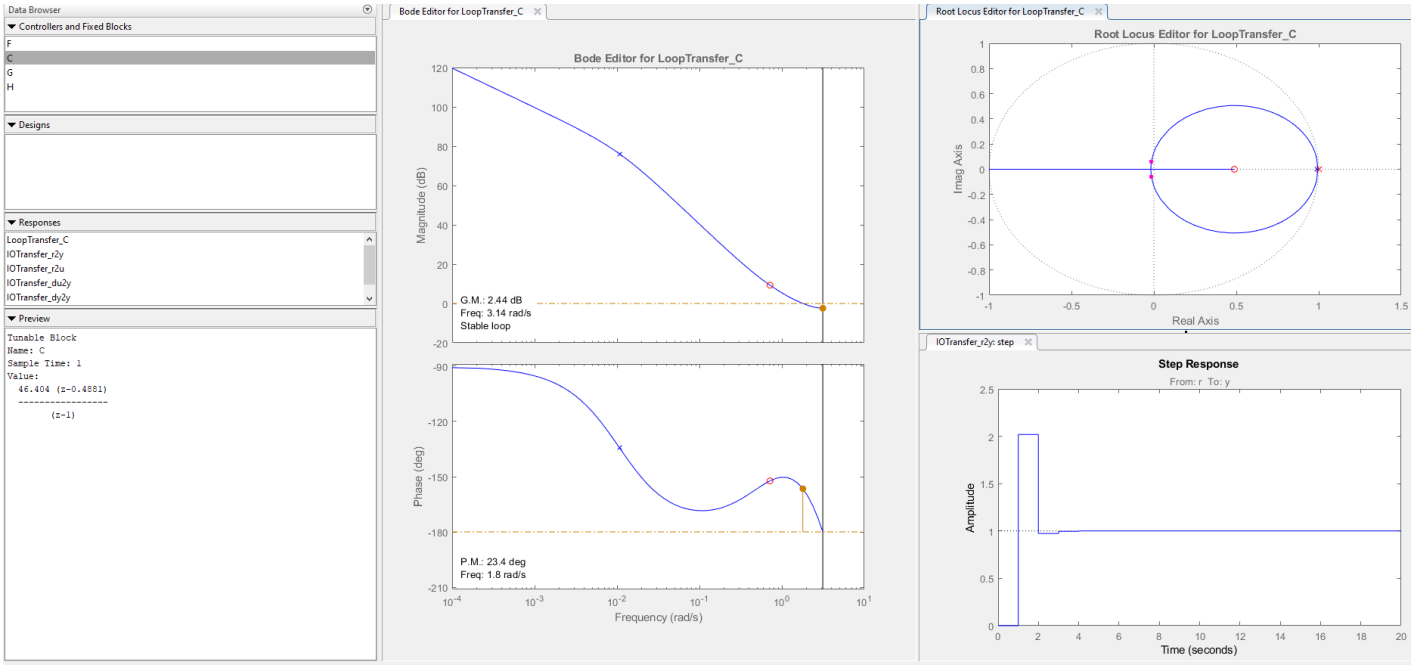


FIGURE 7 – Résultats des simulations du correcteur PI

Comme on peut l'observer sur le lieu des racines, on a rapproché les pôles en boucle fermée du centre du cercle. Ralentissant ceux-ci, les oscillations sont plus modérées, et on obtient une erreur statique nulle. En conclusion, on obtient : $zpi = 0.4881$. $Kpi = 46.404$.

4.3 Réalisation

La mise en oeuvre en réel n'a pas donnée de résultats stables par manque de temps, et n'est donc pas présentée ici.

5 Conclusion

Partant initialement d'un système non linéaire, nous l'avons linéarisé autour d'un point de fonctionnement nous permettant d'appliquer nos méthodes linéaires. Malgré cela, nous avons constaté que s'éloigner du point de fonctionnement était très vite préjudiciable, la nature du système nous permet de le commander que par "petits bouts". En partant de cette idée, nous avons pu voir l'influence d'un correcteur P sur la stabilité du système. Le gain proportionnel accélère le système mais il est apparu nécessaire de le limiter pour rester stable. Le correcteur PI permettait, lui, de corriger les problèmes d'erreur statique propre au système. Grâce à l'outil sisotool, il était très facile de gérer les paramètres du système que sont sa rapidité, sa stabilité et son caractère oscillatoire. Enfin, nous avons constaté la grande influence du temps d'échantillonage sur le système, les 4 secondes initialement choisies n'étaient pas suffisantes pour assurer la précision du système.

A Annexes

A.1 Code Matlab

```
clear all
clc
close all
% Variables :
Te=4;
stepTime=25;
O_P=5;
Final=O_P+1;
Ki=1;
Kd=0;
E_PS=5;
T=93.5;
K=4.068;
delt=0.5;
Kp_crit=-(1/K-2*T/(K*Te));
Kp=1;;
gbo=tf([K*Te],[T Te-T],Te);
pid=tf([Kp Ki*Te-Kp],[1 -1],Te);
Kp=47.585;
Ki=(0.4769+Kp)/Te;

sisotool(gbo);
sisotool(gbo,pid);

setxpcenv('CCompiler','VisualC','CompilerPath',...
    'C:\Program Files (x86)\Microsoft Visual Studio 10.0',...
    'HostTargetComm','TcpIp','TcpIpTargetAddress','10.1.5.244'...
    ,'TcpIpTargetPort','22222'...
    ,'TcpIpSubNetMask','255.255.255.0'...
   ,'TcpIpGateway', '10.1.5.254'...
    ,'TcpIpTargetDriver',...
    'Auto','TcpIpTargetBusType','PCI');

rtwbuild('tp3_sim');

+tg;

y=tg.OutputLog;
x=tg.TimeLog;
```