Régulateur PI(D) - BE noté

13 et 14 novembre 2019

Ce document est un synthèse très succinte du rôle et de l'implémentation d'un PID numérique.

1 Modalités

Ce BE se fait sous Matlab en binôme. Un rapport de synthèse est à rendre et sera évalué pour la note de la matière.

Critères d'évaluation :

— Toute recopie du sujet entraine des points en moins
— Toute information répondant aux questions apporte des points en plus.
— Il s'agit d'une synthèse : 5 pages A4 max.

Rapport à rendre avant le 29/11/2018 12h30 sous moodle
(https ://moodle.ensta-bretagne.fr/course/view.php ?id=969)

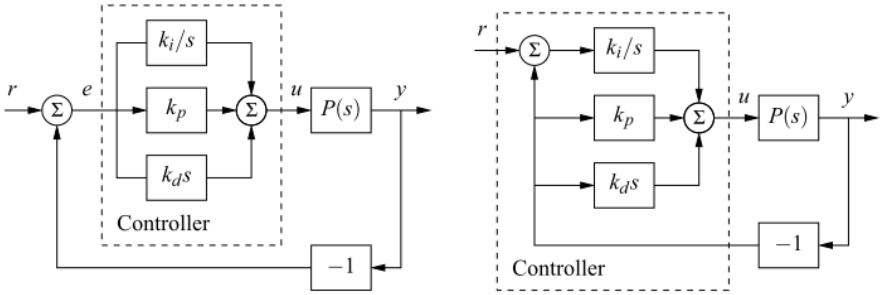
2 Introduction

Les régulateurs PID répondent à plus du 90% des besoins industriels et le nombre de régulateurs installés dans une usine pétrolière, par exemple, se compte par milliers. Malheureusement, malgré l'expérience acquise au fill des ans, les valeurs choisies pour les paramètres P, I et D ne sont pas toujours satisfaisantes, ni adaptées au processus à régler. L'histoire des régulateurs est déjà longue et il peut être intéressant de rappeler quelques étapes importantes. Les premiers régulateurs de type centrifuge apparaissent vers 1750 pour régler la vitesse des moulins à vent, suivi en 1788 du fameux contrôleur de vitesse d'une machine à vapeur de James Watt. En 1942, Ziegler et Nichols ont proposé deux démarches permettant de trouver facilement les paramètres optimums pour une installation donnée. Avec le temps, les propositions de Ziegler et Nichols ont été adaptées ou modifiées selon les besoins. En 1963, Horowitz a ajouté un degré de liberté supplémentaire au régulateur PID afinn de mieux contrôler les dépassements obtenus lors d'une réponse indicielle. Ce nouveau degré de liberté consiste, en particulier, à ne réinjecter vers le terme proportionnel qu'une partie du signal de sortie. Au début des années 1990 et dans le but de fournir des règles d'ajustement simples mais plus performantes que celles de Ziegler-Nichols, Åström et ses collaborateurs ont analysé le comportement dynamique d'un grand nombre de processus. Cette analyse a conduit à l'établissement de tableaux servant aux calculs des paramètres P, I et D à partir de mesures simples.

3 Description

Un régulateur PID remplit essentiellement trois fonctions :

1. Il fournit un signal de commande $u(t)$ en tenant compte de l'évolution du signal de sortie $y(t)$ par rapport à la consigne $w(t)$.
2. Il élimine l'erreur statique grâce au terme intégrateur.
3. Il anticipe les variations de la sortie grâce au terme dérivateur.



(a) PID using error feedback (b) PID using two degrees of freedom

3.1 Le terme proportionnel

Il est simplement implémenté en remplaçant les variables continues par leur version échantillonnéă :

$$P(t_k) = k_p(r(t_k) - y(t_k)$$

Avec h comme pas d'intégration, $t_{k+1} = t_{k+h}$.

3.2 Le terme intégral

Il est calculé itérativement par a méthode des rectangles et en prenant en compte les aspects anti-windup (voir plus loin pour le rappel)ă :

$$I(t_{k+1}) = I(t_k) + k_i h e(t_k) + \frac{h}{T_t}(sat(v) - v)$$

avec

$$T_t = \frac{h}{k_t}$$

3.3 le terme dérivateur

L'inconvénient majeur de l'implémentation d'une fonction de dérivation, c'est le gain important en hautes fréquence, ainsi le bruit (généralement haute fréquence) sera fortement amplifié. Pour pallier ce problème, on utilise une dérivée filtrée :

$$\frac{k_d s}{1 + sTf}$$

avec

$$T_f = \frac{k_d}{k_p}.\frac{1}{N}$$

et N compris entre 2 et 20.

Le terme dérivateur, noté D, est donné par l'équation différentielle ordinaire suivante :

$$T_f \frac{dD}{dt} + D = -k_d \frac{dy}{dt}$$

Une approximation (discrétisation) possible est la suivante :

$$T_f \frac{D(t_k) - D(t_{k-1})}{h} + D(t_k) = -k_d \frac{y(t_k) - y(t_{k-1}}{h}$$

Ce qui se traduit par la relation de récurrence suivante :

$$D(t_k) = \frac{T_f}{T_f + h} - \frac{k_d}{T_f + h}(y(t_k) - y(t_{k-1}))$$

Remarque : on notera que le choix de ce schéma d'intégration fait que l'équation aux différences est stable.

3.4 Influence des coefficients

Il s'agit ici d'évaluer l'influence des paramètres de réglage d'un PID pour un système d'ordre 3. La fonction de transfert d'un tel système est donnée par :

$$P(s) = \frac{1}{(s+1)^3}$$

En utilisant la forme suivante du régulateur PID :

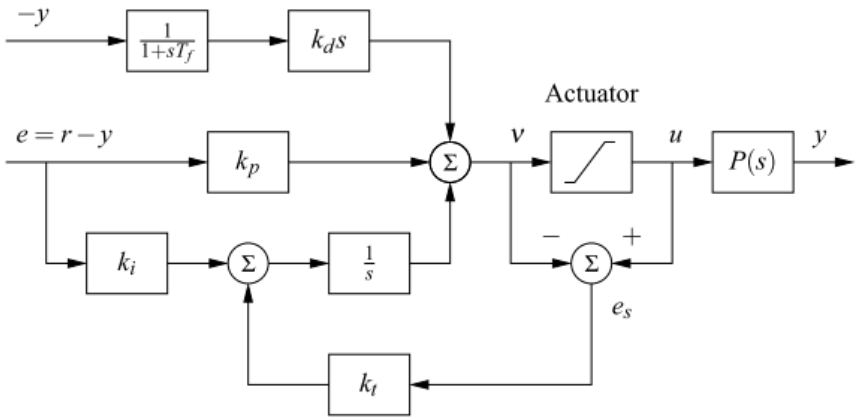
$$C(s) = \frac{k_p s + k_i + k_d s^2}{s}$$

1. Vérifier que le comportement temporel de la boucle fermée est cohérent de celui vu en cours en réglant successivement :
   — le gain k_p (proportionnel), vérifier et tracer les évolutions de comportement
   — le gain k_i (intégral), vérifier et tracer la disparition de l'erreur statique
   — le gain k_d (dérivée), vérifier la stabilisation du comportement
   Pour cela on pourra utiliser les fonctions `tf` et `step`.
2. Donner les diagrammes fréquentiels correspondants (`bode, nyquist, margin`) de la boucle ouverte et analyser les évolutions des marges de gain et de phase.
3. Déduire de ces diagrammes les relations qui existent entre les marges, les fréquences de coupures et le gain statique de la boucle ouverte et le comportement temporel de la boucle fermée.
4. Comparer les réponses fréquentielles des boucles ouverte et fermées

Fonctions à utiliser : `tf, step, bode, nyquist`

3.5 Anti-windup

La première non linéarité qu'il faut prendre en compte, c'est la saturation des actionneurs ; en effet, un moteur a une vitesse limite, une vanne ne peut pas être plus que ouverte ou fermée, etc... Quand ces saturations sont atteintes, le système fonctionne comme s'il était en boucle ouverte car la commande est indépendante du feedback mis en place. Le terme intégral va aussi donner des valeurs importantes car l'erreur reste non nulle et continue à être intégré. La commande reste donc saturée pendant une longue période. Il existe de nombreuses méthodes pour éviter ce phénomène de windup. Une des méthodes consiste en la réinjection de l ?erreur de saturation dans l'intégrateur.



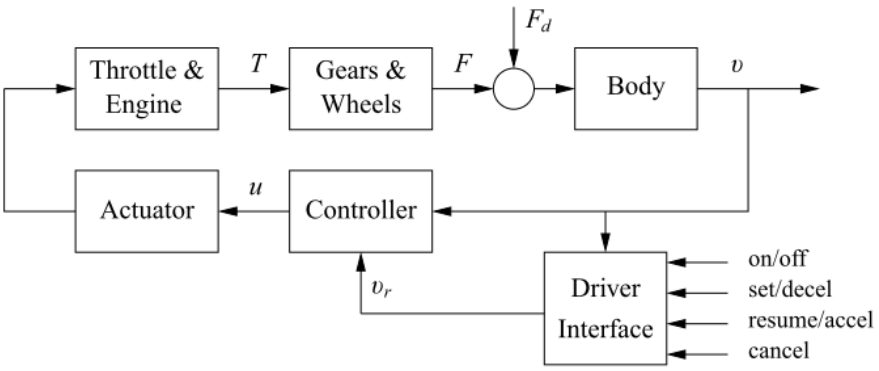
4 Pseudo code

Le pseudo code correspondant à l'implémentation d'un PID numérique est donné ici. Il faudra l'adpater au langage de programmation utilisé dans le système à commander.

```
% Precompute controller coefficients
bi=ki*h
ad=Tf/(Tf+h)
bd=kd/(Tf+h)
br=h/Tt
% Control algorithm - main loop
while (running)
{
        r=adin(ch1)                     % read setpoint from ch1
        y=adin(ch2)                     % read process variable from ch2
        P=kp*(r-y)                      % compute proportional part
        D=ad*D-bd*(y-yold)              % update derivative part
        v=P+I+D                         % compute temporary output
        u=sat(v,ulow,uhigh)             % simulate actuator saturation
        daout(ch1)                      % set analog output ch1
        I=I+bi*(r-y)+br*(u-v)   % update integral
        yold=y                  % update old process output
        sleep(h)                        % wait until next update interval
}
```

5 Mise en place d'un régulateur de vitesse de voiture

Le régulateur de vitesse a été introduit commercialement en 1958 comme une option sur la Chrysler Imperial. Le fonctionnement de base d'un régulateur de vitesse est de détecter la vitesse du véhicule, comparer cette vitesse à une référence voulue, et ensuite accélérer ou décélérer le véhicule selon les besoins. Un algorithme de contrôle simple pour contrôler la vitesse est d'utiliser une rétroaction Proportionnelle Intégrale (PI). Dans cet algorithme, on choisit le flux de carburant vers le moteur sur la base à la fois de l'erreur entre la vitesse actuelle et désirée, et l'intégrale de cette erreur.



5.1 Modèle dynamique

Il s'agit du modèle donné en cours (voir planches) et il est donné dans le fichier `cruisedyn.m`. Il s'agit du modèle non linéaire du véhicule.

5.2 Modèle linéaire

Le modèle linéaire est obtenu par le fichier `cruise_lin.m`. Il s'agit ici d'expliquer ce que signifie les lignes de code de ce fichier en rapport à une linéarisation autour de (v_e, u_e).

L'équation principale de la dynamique devient :

$$\frac{d(v - v_e)}{dt} = a(v - v_e) - b_g(\theta - \theta_e) + b(u - u_e)$$

avec

$$\begin{aligned} a &= \frac{u_e \alpha_n^2 T(\alpha_n v_e) - \rho C_d A v_e}{m} \\ b_g &= g\cos(\theta_e) \\ b &= \frac{\alpha_n T(\alpha_n v_e)}{m} \end{aligned}$$

1. Donner une interprétation du code.
2. En utilisant les fonctions `tf` ou `ss`, créer un système linéaire qui permet de tracer la réponse fréquentielle de ce système.

5.3 Boucle de régulation

1. A partir du fichier `cruise_pictrl.m`, implémenter avec l'un et/ou l'autre des modèles, une boucle de régulation PI avec les spécifications suivantes :
   — Dépassement $D < 1m.s^{-1}$ en réponse à un échelon d'amplitude unitaire.
   — Robustesse à une variation de masse de 25% et une pente de 4°.
2. Comparer le fonctionnement avec et sans anti-wind-up.
3. Quel est l'intérêt de l'intégrateur dans cette régulation ? Argumenter votre réponse avec des courbes temporelles.
4. vérifier le lien entre les performances temporelles et la réponse fréquentielle (en boucle ouverte)