Master EAI 2ème année
- Modélisation et commande floue -
Travaux pratiques
Année universitaire 2008-2009

Analyse et réglages d'un régulateur FLOU

Objectif du TP :

- Modéliser et identifier un processus
- Développer un régulateur flou
- Implémenter le contrôleur sur le système modélisé
- Tester le régulateur sur le processus réel
- Comparer le comportement en simulation et en conditions réelles

Avant de commencer :

Toutes les commandes Matlab seront inscrites dans un fichier de commande (script) que vous exécuterez.
Les modèles construits sous Simulink seront simulés via Matlab et non directement dans Simulink.
Commentez chaque calcul de vos scripts (une ligne de commentaire (en vert) commençant par %).

1. *CONTEXTE ET CAHIER DES CHARGES*

On souhaite commander la puissance de chauffe d'une maison pour maintenir une température intérieure moyenne égale à la consigne souhaitée.
On réalisera dans un premier une identification du processus à partir de données réelles. Il s'agira ensuite de développer un régulateur FLOU pour contrôler la température intérieure moyenne.
On souhaite que le suivi de consigne échelon se fasse sans erreur statique, avec un temps de réponse à 5% égal à 3600s, en régime nominal. On souhaite de plus, que le système soit suffisamment robuste pour fonctionner aux 3 régimes de fonctionnement.

2. DESCRIPTION DU SYSTEME EXPERIMENTAL

La maquette possède une résistance de chauffe permettant de chauffer l'intérieur de la maison et des capteurs de températures répartis dans l'ensemble de la maison. On considère que la température moyenne à un instant donné dans la maquette est égale à la moyenne des différentes températures relevées à l'intérieur de la maquette à ce même instant.

La puissance de chauffe peut être contrôlée dans Matlab en utilisant la fonction Envoyer_commande. Cette fonction prend comme paramètre la valeur de la puissance « Puissance » à appliquer sur la résistance. La variable de sortie *'retour'* permet de savoir si l'exécution a fonctionné (1) ou non (0).

```
function Envoyer_commande(Puissance)
```

- Puissance : Puissance de chauffe à appliquer en W

Le relevé des températures de la journée actuelle peut être effectué grâce à la fonction *'Acquisition_Temp'* qui retourne un vecteur pour la date d'acquisition et un vecteur pour chacune des 7 températures.

```
function [Date Text T1 T2 T3 T4 T5 T6 T7] = Acquisition_Temp()
```

- Date_Num_Temp : Instant de l'acquisition en serial
- Text : Température extérieure en °C
- T1 : Température position 1
- T2 : Température position 2
- T3 : Température position 3
- T4 : Température position 4
- T5 : Température position 5
- T6 : Température position 6
- T7 : Température position 7

Le relevé des puissances appliquées de la journée actuelle peut être effectué grâce à la fonction *'Acquisition_Com'* qui retourne un vecteur pour la puissance.

```
function [Date_Num_Com Commande] = Acquisition_Com()
```

- Date_Num_Com : Instant de l'acquisition en serial
- Commande : Puissance de chauffe en W

3. MODELISATION ET IDENTIFICATION DU SYSTEME

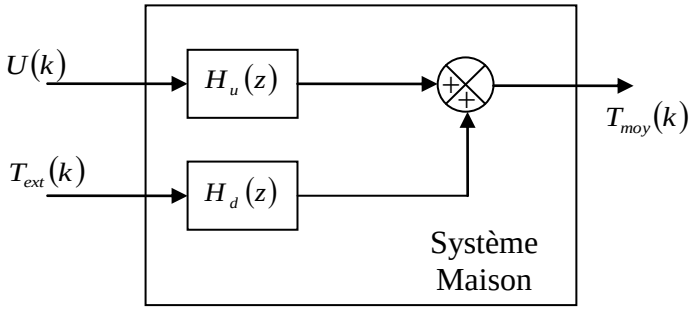
On souhaite obtenir un modèle linéaire de la partie opérative, c'est-à-dire de la variation de la température dans la pièce en fonction de la puissance de chauffe et de la température extérieure. On identifiera le système sous la forme échantillonnée suivante :

$$T_{moy}(z) = H_u(z) \cdot U(z) + H_d(z) \cdot T_{ext}(z)$$

Avec :

- T_{moy} la température moyenne de la pièce
- T_{ext} la température extérieure
- U : la puissance de chauffe

- $H_u(z)$: la fonction de transfert entre U et T_{moy}.
- $H_d(z)$: la fonction de transfert entre T_{ext} et T_{moy}.



Des essais expérimentaux ont été réalisés en appliquant des variations de type échelon sur la puissance de chauffe. Les différentes températures
A partir des relevés expérimentaux des différentes températures fournies dans le fichier *'Temperature.mat'* et à partir des valeurs de la puissance appliquée fournies dans le fichier *'Commande.mat'*. L'import de ces données se fait par la commande :

```
load Temperature;
load Commande ;
```

La première créée dans le workspace les températures et les instants des relevés :

- Date_temp_id : Instant de l'acquisition en HH:MM:SS
- T0_id : Température extérieure en °C
- T1_id : Température du plafond central en °C
- T2_id : Température au Sud Ouest en °C
- T3_id : Température au Sud Est en °C
- T4_id : Température au Milieu Ouest en °C
- T5_id : Température au Milieu Est en °C
- T6_id : Température au Nord Ouest en °C
- T7_id : Température au Nord Est en °C

La seconde créée dans le workspace les températures et les instants des relevés :

- Date_com_id : Instant de l'acquisition en HH:MM:SS
- U_id : Puissance en W

Utilisez ces données pour retrouver les deux fonctions de transfert $H_u(z)$ et $H_d(z)$.

4. ***DESIGN DU CONTROLEUR FLOU***

A partir du modèle thermique nous pouvons synthétiser un régulateur flou
Le régulateur possède 1 entrée : l'erreur de poursuite $\varepsilon(k)$ définie comme étant :

$$\varepsilon(k) = T_{cons}(k) - T_{moy}(k)$$

$t_e = 60s$ est la période d'échantillonnage et k est l'indice temporel.

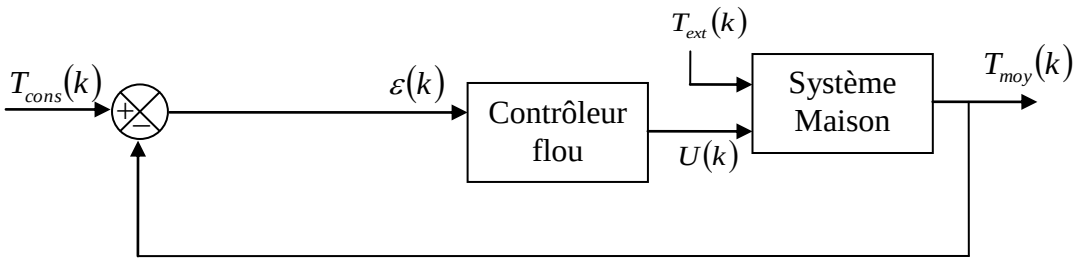


Figure 2 : Synoptique du système piloté par un contrôleur flou de type SISO

4.1. On définit pour l'entrée, dénommée erreur, 3 ensembles flous avec des fonctions d'appartenances de type gaussienne sur l'univers de discours [-10 ;10].

4.2. Les fonctions d'appartenance pour la variable erreur seront nommées : NEGATIVE, NULLE, et POSITIVE. Pour la variable de sortie, nommée commande, les fonctions d'appartenance seront nommées : GN, N, Z, P et GP (Grande Négative, Négative, Zéro, Positive, et Grande Positive).

4.3. Définir les 3 règles nécessaires au bon fonctionnement du régulateur.

5. ***CREATION DU CONTRÔLEUR FLOU DANS MATLAB***

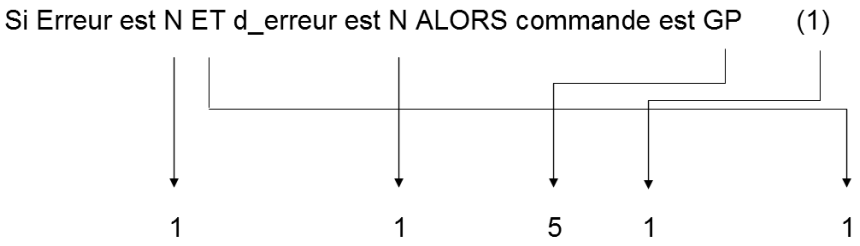
5.1. Fuzzification des variables d'entrée et de sortie

La création d'un système flou se fait à l'aide de la commande *newfis* qui accepte jusqu'à 7 arguments.
Dans la fenêtre de Matlab, tapez help *newfis*.

5.1.1. Créer ce régulateur, que l'on nommera *sys_flou*, à l'aide des fonctions *addvar* et *addmf*.
5.1.2. Tracer les fonctions d'appartenance de la valeur d'entrée du système *sys_flou* à l'aide de la commande *plotmf*.
5.1.3. De même pour la variable de sortie.

5.2. Edition des règles floues

Pour un système flou possédant *m* entrées et *n* sorties, l'ensemble des règles floues est défini par une matrice de règles possédant autant de lignes que d'ensembles flous de chacune des entrées et (*m+n+2*) colonnes.
L'exemple d'une règle, d'un système flou à deux entrées et une sortie, est rappelée ci-après.
Si l'erreur est négative (1er ensemble nommé N) **ET** la dérivée de l'erreur est négative (1er ensemble nommé N) **ALORS** la commande est GP (5ème ensemble nommé GP). Cette règle est pondérée par 1.
Le dernier chiffre symbolise l'opérateur liant les clauses des 2 prémisses (1 pour ET, 2 pour OU).



5.2.1. Créer la matrice de règles.
5.2.2. La commande *showrule(sys_flou)* affiche les règles de façon normalisées.
5.2.3. A l'aide de la commande *getfis*, afficher les caractéristiques :
- des entrées
- de la sortie
- du système flou
- du 1er ensemble flou de la 1ère entrée

5.2.4. Afficher à l'aide de la commande *showfis* les caractéristiques de l'ensemble du système flou.
5.2.5. Afficher la liste des règles avec la commande *showrule*.
5.2.6. Générer la surface de la variable de sortie en fonction des entrées. On utilisera pour cela la commande *surfview*.
5.2.7. Représenter graphiquement, à l'aide de la commande *plotfis*, le système.

5.3. Défuzzification

La commande *ruleview* affiche la fenêtre *Rule Viewer* dans laquelle on peut observer la défuzzification par la méthode max-min.
A l'aide de la souris on peut choisir des valeurs quelconques pour chacune des entrées et observer la fonction d'appartenance de la variable de sortie obtenue par la méthode d'inférence max-min. Par défaut, la défuzzification est réalisée par la méthode du calcul du centre de gravité.

Que donne la commande issue du régulateur flou pour une valeur 0,4545 de l'erreur. Ce résultat peut également être obtenu par la commande *evalfis*.

6. UTILISATION DU REGULATEUR FLOU DANS UNE LOI DE COMMANDE

Pour utiliser ce régulateur, il faut définir :
- le signal de consigne
- l'initialisation de la commande
- la sortie du processus

6.1. Le signal de consigne

La variable r sera définie comme étant la consigne et de la forme :

```
r=(0:60)/12;
triangle = [r fliplr(r)];
carre = [zeros(1,100) 5*ones(1,100) zeros(1,100)];
triangle = triangle+sin(0.2*(0:length(triangle)-1));
r=2+[triangle carre];
```

6.2. Initialisation de la commande

```
u = ones(1,2);
y = r;
err = zeros(1:2);
%gain multiplicatif de la sortie du régulateur flou
Gain = 1.8;
```

6.3. lecture de la sortie du processus

```
%la commande à appliquer
x = [err(i) d_err(i)];
du(i) = evalfis(x,sys_flou);
u(i) = u(i-1)+Gain*du(i)
```

6.3.1. Le processus est régit par l'équation suivante :

$$T_{moy}(z) = H_u(z) \cdot U(z) + H_d(z) \cdot T_{ext}(z)$$

Définir l'équation de la sortie du processus.

6.3.2. On définira ensuite l'erreur et la dérivée de l'erreur comme suit :

```
%erreur et dérivée de l'erreur
err(i) = y(i)-r(i+1);
d_err(i) = err(i)-err(i-1);
```

6.3.3. Et la commande à appliquer :

```
%la commande à appliquer
x = [err(i) d_err(i)];
du(i) = evalfis(x,sys_flou);
u(i) = u(i-1)+Gain*du(i)
```

7. ***TESTS EN CONDITIONS REELLES***

A l'aide de la fonction *'Envoyer_commande'*, *'Acquisition_Temp'* et *'Acquisition_Com'* appliquer votre régulateur sur le système réel.
Afficher en temps réel les températures et les commandes sur un graphe pour suivre l'évolution du système.

Expliquer quelles corrections peuvent être apportées au régulateur pour obtenir des résultats plus proche du cahier des charges.

8. ***OPTIMISATION DU CONTROLEUR FLOU***

Le régulateur possède cette fois ci 2 entrées : l'erreur de poursuite $\varepsilon(k)$ et sa variation $\Delta\varepsilon(k)$, avec :

$$\varepsilon(k) = T_{cons}(k) - T_{moy}(k)$$

$$\Delta\varepsilon(k) = \frac{\varepsilon(k) - \varepsilon(k-1)}{t_e}$$

$t_e = 60s$ est la période d'échantillonnage et k est l'indice temporel.

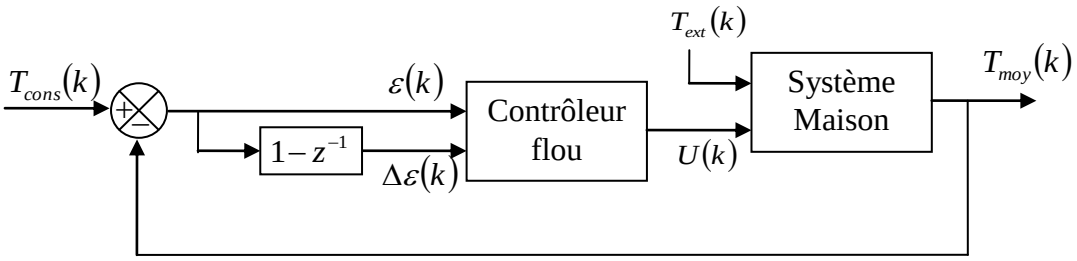


Figure 2 : Synoptique du système piloté par un contrôleur flou de type MISO

Il suffit de compléter le contrôleur précédent en lui ajoutant une entrée et les 6 règles manquantes. La 2ème variable, d_erreur, aura le même univers de discours que la 1ère variable.

Tester en simulation et en conditions réelles.
Analyser et Conclure.

9. CONCLUSION

Afficher l'ensemble des résultats (consigne, commande et sortie du processus) des tests en simulation et sur la maquette.
Commenter et analyser les performances des différents régulateurs flous.

Expliquer quelles seraient les corrections à apporter pour rendre le régulateur plus dynamique.