République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud MAMMERI, Tizi-Ouzou

Faculté de Génie Electrique et d'Informatique
Département d'Automatique

Mémoire de Fin d'Etudes

En vue de l'obtention du diplôme

De Master en Automatique

Thème

Contribution à la conception et l'implémentation des régulateurs à base de la logique floue sur les APIs (S7-200)

Proposé et dirigé par : M[r] HADDOUCHE

Présenté par :

TAHENNI Hamza
ZADOUD Mohamed Amziane

Soutenu le : 10 / 0 7 /2011

Promotion 2011

Ce travail a été préparé au laboratoire d'automates programmable et de régulation industrielle.

REMERCIEMENTS

Nous remercions dieu de nous avoir donné la force et la volonté pour mener à bien notre travail.

Nous tenons à exprimer nos vifs remerciements pour Monsieur HADDOUCHE, pour nous avoir proposé ce sujet, pour son aide, conseils et son soutient constant

tout au long de la réalisation de ce mémoire.

Nous remercions aussi Monsieur MAIDI, Monsieur CHARIF, pour leur aide précieuse et pour la documentation qu'ils ont mis à notre disposition.

Nous remercions également les membres de jury qui nous ferons l'honneur d'évaluer ce modeste travail.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail, trouvent ici, l'expression de notre profonde gratitude.

Chapitre I

Contrôle à base de la logique floue

Chapitre II

Identification d'un four électrique et régulation de température

Chapitre III

Techniques de programmation des Automates Programmable Industriels (APIs)

Chapitre IV

Implémentation d'un régulateur à base de logique floue sur API

Introduction Générale

Tandis que les techniques modernes de commande ont fait une incursion modeste en applications pratiques, la technique de commande par logique floue avait rapidement gagné de popularité parmi les ingénieurs d'application. Cette popularité accrue peut être attribuée au fait que la logique floue fournit un outil puissant qui permet à des ingénieurs d'incorporer, facilement, le raisonnement humain dans les algorithmes de commande. Par opposition à la théorie de commande moderne, la conception de stratégie de commande par logique floue n'est pas basée sur un modèle mathématique du processus à contrôler mais sur l'expérience humaine par rapport à ce processus. Un contrôleur flou est conçu en utilisant le raisonnement humain basé sur des instruments de logique floue et programmé dans la langue de la logique floue (les fonctions d'appartenance, les règles et l'interprétation des règles) [6].

Il est intéressant de noter que le succès de la commande par logique floue est en grande partie dû à la réussite de ses nombreuses applications industrielles. Les intérêts industriels pour la commande basée sur logique floue, comme à été démontrée par de nombreux travaux de recherche cités dans la littérature, a créé une prise de conscience de son importance croissante au sein de la communauté des chercheurs. Cet intérêt donné a logique floue à commencé tôt dans les années 90, où le Laboratoire Recherche sur la commande appliquée à Cleveland State University. Soutenu par des associés industriels, le laboratoire a lancé un programme de recherche pour étudier le rôle et l'apport du contrôle par la logique floue dans l'industrie. La question primaire qui été posée à ce moment là est : 'Que peut ramener de plus le contrôle par logique floue par rapport aux techniques de commande classiques ?'. Les résultats de recherches au cours des dernières années ont été rapportés dans [13-17] (13-17 dans 6).

Actuellement, les contrôleurs programmables logiques (Automates Programmables Industriels : API) occupent la première place dans l'industrie. Ils ont montrés leur capacité de piloter des chaînes de production très complexes et des machines très précises. La régulation au sein de ces contrôleurs est basée sur les techniques de commande classique, à savoir les régulateurs PID classiques [10]. Avec l'émergence des nouvelles techniques de commandes basées sur l'intelligence artificielle, les automaticiens commencent à s'intéresser à l'intégration de ces nouvelles techniques dans les contrôleurs logiques programmables (APIs). C'est dans ce contexte que s'inscrit notre projet de fin d'étude. Par conséquent, Notre objectif est de répondre à l'hypothèse suivante :

Peut-on intégrer les techniques de commande basées sur la logique floue sur un API ?

Pour étudier cette hypothèse, nous avons envisagé d'une part, de concevoir un régulateur PID floue pour la régulation de température dans un four électrique a fin de comprendre l'approche de contrôle basé sur la logique floue et d'autre part, de trouver un moyen de matérialiser ce régulateur basé sur la logique floue et de l'intégrer sur un API afin de réguler la température du four électrique en question.

Notre travail est organisé comme suit : Après l'introduction générale, dans le premier chapitre nous présentons le contexte général de la logique floue et les différents types de contrôleurs flous. Le deuxième chapitre sera consacré à l'identification d'un four électrique ainsi qu'à l'élaboration de régulateurs, PID classique et PID flou, pour le contrôle du four. Dans le troisième Chapitre nous donnons une vue générale sur les APIs, tout en se focalisant sur les techniques de programmation, et un exemple d'application de régulation PID pour le four. Le contenu du quatrième et dernier chapitre portera sur l'implémentation d'un régulateur flou sur un API S7-200 de type SIEMENS et nous terminons par une conclusion générale.

Chapitre I

Contrôle à base de la logique floue

I.1. Introduction

La logique floue est une théorie qui a connu un grand engouement depuis que Zadeh a introduit le concept de sous-ensembles flous en 1965. Elle trouve notamment sa place dans le domaine de la commande pour une large gamme de systèmes et plus généralement en génie électrique. Elle présente en effet l'avantage d'utiliser des règles linguistiques simples permettant de traduire facilement le savoir faire d'un expert pour répondre à une problématique spécifique.

Dans ce sens, des correcteurs à base de logique floue améliorent de façon globale aussi bien les performances dynamiques que la robustesse des systèmes commandés, en s'appuyant sur la connaissance de ceux-ci.

Cependant, une difficulté inhérente au contrôle flou doit être soulignée : un correcteur flou, même pour une structure très simple, se voit constitué d'un nombre important de paramètres. Ce constat permet d'expliquer la délicatesse du réglage de ce type de commande. Il est bien entendu toujours possible de la régler par tâtonnements, mais cette procédure est longue et ne garantit pas qu'un jeu de paramètres exploitable sera obtenu à l'issue de la procédure et ne fixe aucune limite prévisible à la durée de l'optimisation [1].

Dans cette partie, nous rappellerons le principe de logique floue et la structure des correcteurs flous.

I.2 Contrôle flou [1]

Depuis une vingtaine d'années, la commande floue connaît un intérêt croissant. L'un des principaux intérêts de ces commandes à base de logique floue consiste à pouvoir faire passer relativement simplement par l'intermédiaire de règles linguistiques, l'expertise que l'on peut avoir du processus vers le contrôleur. Il est ainsi possible de transformer le savoir de l'expert en règles simples que le contrôleur peut mettre en œuvre. Une facilité d'implantation des solutions pour des problèmes complexes est alors associée à une robustesse vis à vis des incertitudes et la possibilité d'intégration du savoir de l'expert.

Du point de vue historique, les prémisses de la logique floue, visant à traiter la notion d'incertitude, datent des années 30. Il faudra cependant attendre que Zadeh introduise le concept de sous-ensembles flous, en 1965, pour assister aux premières grandes avancées dans le domaine. Par la suite, en 1974, Mamdani introduisait la commande floue pour la régulation de processus industriel.

Enfin, dans les années 80, la commande floue connaît un essor considérable au Japon, notamment grâce aux travaux de Sugeno pour se répandre ensuite dans le monde entier.

I.2.1 Logique floue et ensembles flous [1]

La notion de logique floue permet d'étendre la notion de logique classique, associée aux variables booléennes ne prenant que deux valeurs 0 et 1. Il est alors possible d'associer à des variables des coefficients d'appartenance à des sous-ensembles flous prenant des valeurs dans l'intervalle [0, 1] et quantifiant l'incertitude sur la variable. Un événement certain pour la variable se traduira par un coefficient d'appartenance au sous-ensemble flou, c'est-à-dire à la propriété, égal à 1 alors que la valeur sera inferieure à 1 en présence d'incertitudes. Il vient alors qu'une valeur nulle pour un coefficient d'appartenance indique que la possibilité d'appartenance au sous-ensemble sélectionné de la variable représentant la grandeur concernée est complètement rejetée.
L'univers de discours d'une variable donnée, c'est-à-dire son domaine de variation, peut alors être divisé en plusieurs sous-ensembles au moyen de fonctions d'appartenance, FA, comme illustré dans la figure **I.1** avec des fonctions d'appartenance triangulaires.

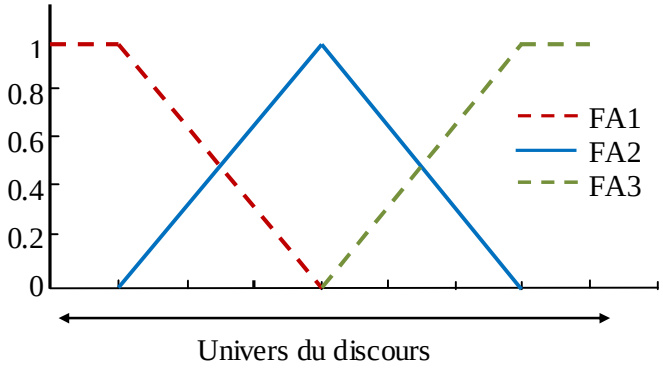


Figure I.1 Exemple de fonctions d'appartenance

Pour chaque valeur de la variable considérée, des degrés d'appartenance à chacun des sous-ensembles flous vont être définis. La fonction d'appartenance FA1, se rapporte alors au sous ensemble flou SF1, etc.

La problématique sera donc de choisir le nombre et le type de sous-ensembles flous pour chacune des variables devant être traitée.

Cette notion peut s'appliquer à de nombreux problèmes différents et notamment à la commande floue pour générer des correcteurs de type PID non-linéaires.

I.2.2 Principe d'une commande floue [1]

La structure d'une commande floue, présentée dans la figure **I.2**, peut être décomposée en trois grands modules.

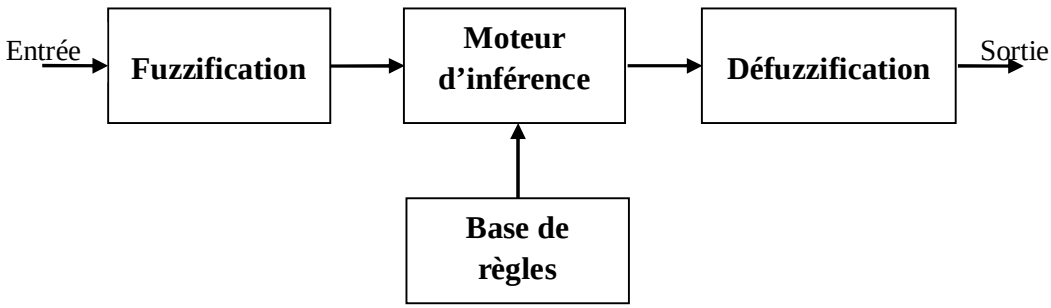


Figure I.2 Structure générale d'une commande floue

Le premier de ces modules traite les entrées du système : c'est la fuzzification. Il permet d'associer à chacune des entrées réelles, par le biais de fonctions d'appartenances, un degré d'appartenance pour chacun des sous-ensembles flous définis sur l'ensemble du discours.

Le deuxième module est constitué du moteur d'inférence et de la base de règles.
Celle-ci est constituée de règles de type : "Si..., Alors..." et va permettre de passer des degrés d'appartenance des grandeurs d'entrées aux degrés d'appartenance aux sous-ensembles flous de la grandeur de commande. Le moteur d'inférence, lui, va permettre de générer une conclusion à partir des entrées et des règles actives. Il calcule alors les degrés d'appartenance aux sous-ensembles flous correspondant à la commande du système.

Enfin, le dernier module, l'interface de défuzzification, va permettre de transformer les degrés d'appartenance des sous-ensembles flous de commande en grandeur numérique. C'est la transformation inverse du module de fuzzification. A partir de cette structure, différents types de correcteurs flous vont alors pouvoir être définis.

I.2.3 Moteur d'inférences floues [1]

La présente section a pour but de présenter plus en détails la structure retenue pour les correcteurs flous de type PI et PID. Ces deux correcteurs utilisent le même moteur flou dont la structure de type PD est représentée figure **I.3**.

Deux entrées sont traitées, l'erreur **e** et la dérivée de l'erreur **de** pour une unique commande **SPD**. Les deux entrées sont normalisées au moyen de gains de normalisation, **gem** pour l'erreur et **gdem** pour la dérivée de l'erreur. Un gain de dé normalisation, **gm**, est affecté sur la sortie. L'univers du discours pour le moteur flou est ainsi ramené sur l'intervalle [−1, +1]. Les facteurs de normalisation permettent ainsi de définir le domaine de variation normalisé des entrées et le gain de dénormalisation définit le gain en sortie du correcteur flou de type PD. Ces éléments permettent d'agir de façon globale sur la surface de commande en élargissant ou réduisant l'univers du discours des grandeurs de commande.

En ce qui concerne le module de fuzzification, il existe de nombreux types de fonctions d'appartenance comme par exemple des fonctions de type triangle, trapèze, gaussienne pour n'en citer que quelques unes. Celles-ci vont être définies sur l'univers du discours normalisé afin de donner les degrés d'appartenance aux sous ensembles flous en entrée. L'influence des

positions des fonctions d'appartenance va également être traduite par une action globale sur la surface de commande.

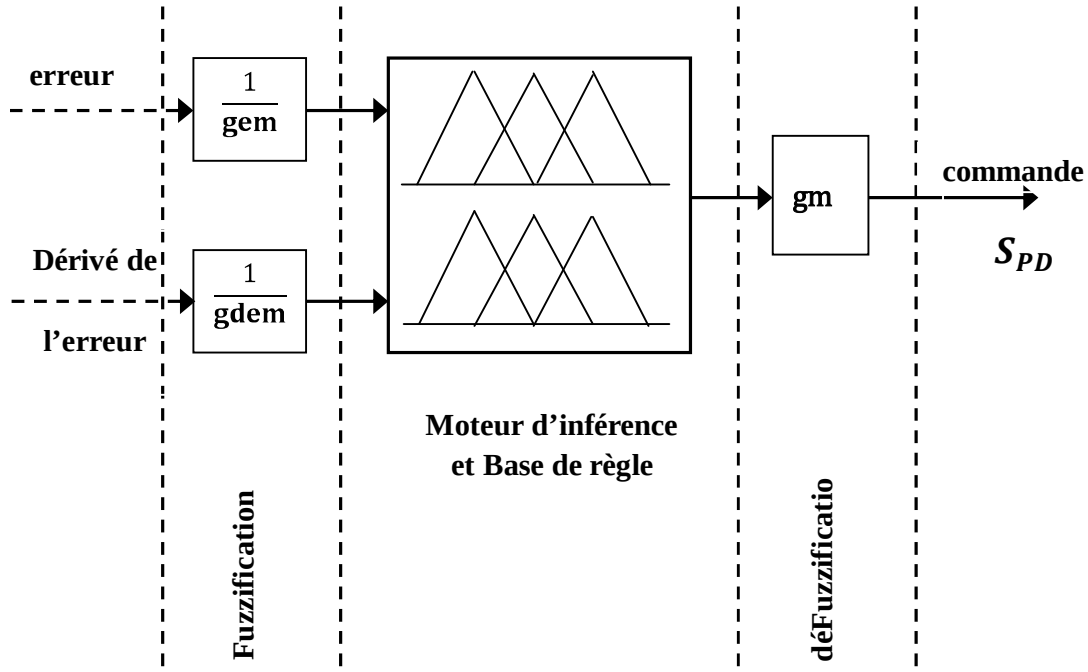


Figure I.3 Structure du correcteur flou de type PD

Souvent, dans le domaine de la commande, elles seront positionnées de façon à obtenir une action réactive lorsque la valeur de la grandeur régulée est éloignée de la référence mais un gain moindre autour de celle-ci.

Pour le deuxième module, la base de règles floues va caractériser les relations entre les classes d'événements possibles en entrée et les commandes correspondantes. Ainsi, pour chaque combinaison des sous-ensembles flous activée en entrée, la base de règles associe un sous-ensemble flou de sortie. La base de règles possède alors une influence locale sur la surface de commande. La modification d'une règle permet d'adapter précisément la commande par rapport à une contrainte particulière.

Enfin, plusieurs méthodes permettent de réaliser l'étape de défuziffication. La méthode du centre de gravité est l'un des moyens les plus simples et les plus utilisés. Elle consiste à rechercher le centre de gravité d'un système de sous ensembles flous dont les poids sont leurs coefficients d'appartenance. La sélection des sous-ensembles flous de commandes activés au moyen de degrés d'appartenance conduit alors par cette méthode à la définition d'une grandeur de commande réelle.

Il est important de remarquer qu'il existe une certaine dualité entre une action sur les fonctions d'appartenance et les gains de normalisation et dénormalisation, chacun de ces éléments agissant globalement sur la surface de commande. En effet, en fonction de la valeur attribuée au coefficient de normalisation, une même position d'une fonction d'appartenance activera un même sous-ensemble flou pour un état différent du système. Cette propriété sera largement utilisée dans le cadre de la définition de réglages "simples" de correcteurs flous, notamment pour les réglages préétablis définis dans [1] présentés plus loin, les positions des fonctions d'appartenance restant fixes et les gains de normalisation étant calculés en fonction du système commandé.

I.2.4 Choix des éléments du moteur d'inférences floues [1]

La nécessité de simplifier le réglage des commandes floues utilisées conduit à réaliser certains choix pour la structure du correcteur. La présente section a pour but de présenter ceux-ci ([1]).

Le premier élément est le choix de la nature des fonctions d'appartenance en entrée. Afin de faciliter les réglages du contrôleur flou, nous utiliserons des formes triangulaires, ce qui permet de traiter très simplement des fonctions linéaires par morceaux en entrée. Les fonctions d'appartenance sont placées de telle manière qu'à tout moment il n'y ait que deux fonctions d'appartenances activées pour chaque entrée. Ce choix apporte plusieurs avantages. Tout d'abord, en limitant les interactions entre les paramètres, la commande est ainsi considérablement simplifiée. De plus, une action très localisée sur la surface de commande est ainsi rendue possible.

Enfin, limitant le nombre de fonctions actives simultanément, le temps de calcul nécessaire au traitement flou sur le calculateur est également réduit, en vue de possible une implantation sur microcontrôleur.

Ayant choisi le type de fonction d'appartenance en entrée, il faut maintenant déterminer leur nombre, c'est-à-dire la couverture de l'univers du discours. Plus ce nombre sera important, plus le nombre de sous-ensembles flous sera conséquent, et plus la sensibilité de la commande floue augmentera. Cependant, une telle augmentation se traduit aussi par un nombre de paramètres à régler de plus en plus important, ce qui peut s'avérer problématique en termes de temps et difficulté de réglage. Nous fixons alors à sept le nombre de fonctions

d'appartenance, afin d'obtenir un bon compromis entre la sensibilité de la commande et la difficulté de réglage.

Celles-ci sont représentées dans la figure **I.4** et notées : NB (Négative Big), NS (Négative Small), NV S (Negative Very Small), Z (Zero), PV S (Positive Very Small), PS (Positive Small) et PB (Positive Big). La grandeur μ(x) définit le degré d'appartenance au sous-ensemble flou considéré. Afin de garantir une réponse du système identique pour les sollicitations positive et négatives, une symétrie par rapport à zéro est mise en place pour les fonctions d'appartenance, ce qui conduit à fixer la fonction d'appartenance centrale.

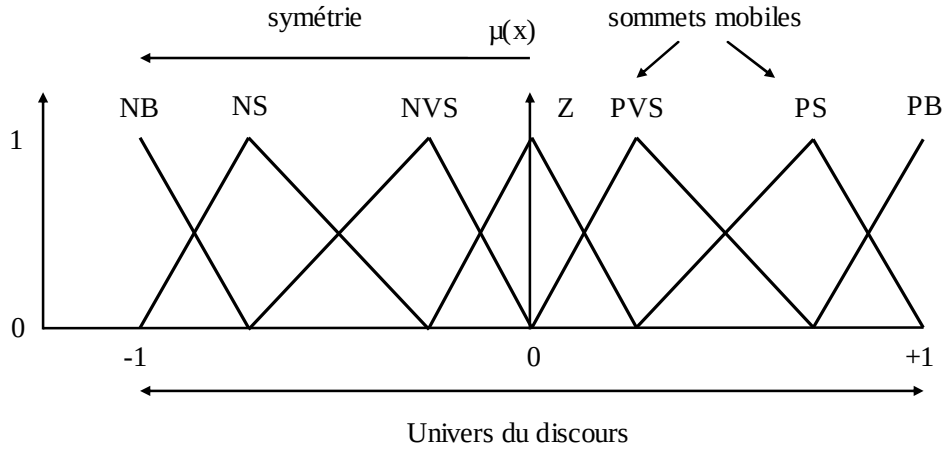


Figure I.4 Fonctions d'appartenance sur les entrées

De plus, afin de pouvoir agir sur l'ensemble de l'univers du discours sélectionné par les gains de normalisation, les fonctions d'appartenance extrêmes (NB et PB) sont fixées aux limites de l'intervalle [−1, +1]. Ceci se traduit alors par deux degrés de liberté pour chaque entrée pour régler le contrôleur flou, correspondant aux deux fonctions d'appartenance intermédiaires mobiles, **PS** et **PV S**. Il y a alors deux sommets à régler par entrée soit 4 paramètres pour les fonctions d'appartenance en entrée. Les notations retenues, pour les sommets des fonctions d'appartenance, seront par exemple **PV Se** pour l'erreur et **PV Sde** pour la dérivée de l'erreur.

Dans l'optique du réglage simple d'une commande floue, il est préférable de fixer les règles de la table de règles car régler séparément chacune des règles augmenterait de façon considérable le nombre de paramètres à régler. De plus, leur influence sur la surface de commande étant locale, un tel choix ne pénalise pas fortement le comportement global. Pour ce faire une table anti diagonale classique, tableau **I.1** va être utilisée.

		de						
		NB	***NS***	***NVS***	***Z***	***NVS***	***PS***	***PB***
e	***NB***	***NB***	***NB***	***NB***	***NB***	***NS***	***NVS***	***Z***
	NS	***NB***	***NB***	***NB***	***NS***	***NVS***	***Z***	***PVS***
	NVS	***NB***	***NB***	***NS***	***NVS***	***Z***	***PVS***	***PS***
	Z	***NB***	***NS***	***NVS***	***Z***	***PVS***	***PS***	***PB***
	NVS	***NS***	***NVS***	***Z***	***PVS***	***PS***	***PB***	***PB***
	PS	***NVS***	***Z***	***PVS***	***PS***	***PB***	***PB***	***PB***
	PB	***Z***	***PVS***	***PS***	***PB***	***PB***	***PB***	***PB***

Tableau I.1 Tableau de règles anti diagonale

Cette table donne l'ensemble des règles linguistiques et se lit, par exemple :

Si e est **PS** et de est **PV S alors uf** est **PB**

La perception humaine de la commande du procédé est ainsi traduite, c'est-à-dire que lorsque la valeur du signal est éloignée de la référence et qu'elle continue à s'en éloigner, un très fort gain va être appliqué au système. Au contraire, au voisinage de la référence, le gain sera moindre. Il est donc aisé d'introduire la non-linéarité de la commande. Le tableau de règles permet d'agir très localement sur la surface de commande et donc une variation de l'un de ses paramètres n'aura qu'une répercussion locale sur la réponse globale. De plus, le nombre de paramètres à régler est ici très important (7×7 = 49 paramètres). Le correcteur sera donc réglé par rapport aux degrés de liberté sur les fonctions d'appartenance d'entrées et les singletons de sortie, qui ont une influence plus globale, ce qui permet de limiter le nombre de variables du correcteur.

Pour la défuzzification, le contrôleur flou retenu est de type Sugeno, c'est-à-dire qu'il possède des singletons en guise de fonctions d'appartenance de commande, notés par exemple **PV Ss**, comme indiqué figure **I.5**.

La défuzzification par la méthode du centre de gravité pour un correcteur flou de type Sugeno revient alors à effectuer une moyenne de valeurs pondérées ce qui est plus simple et donc plus rapide à calculer sur le calculateur que la plupart des autres méthodes de défuzzification. De façon similaire aux fonctions d'appartenance en entrée, les singletons extrêmes sont fixés

ainsi que le singleton central et une symétrie par rapport à zéro est appliquée. Ce choix conduit alors à deux singletons mobiles à régler.

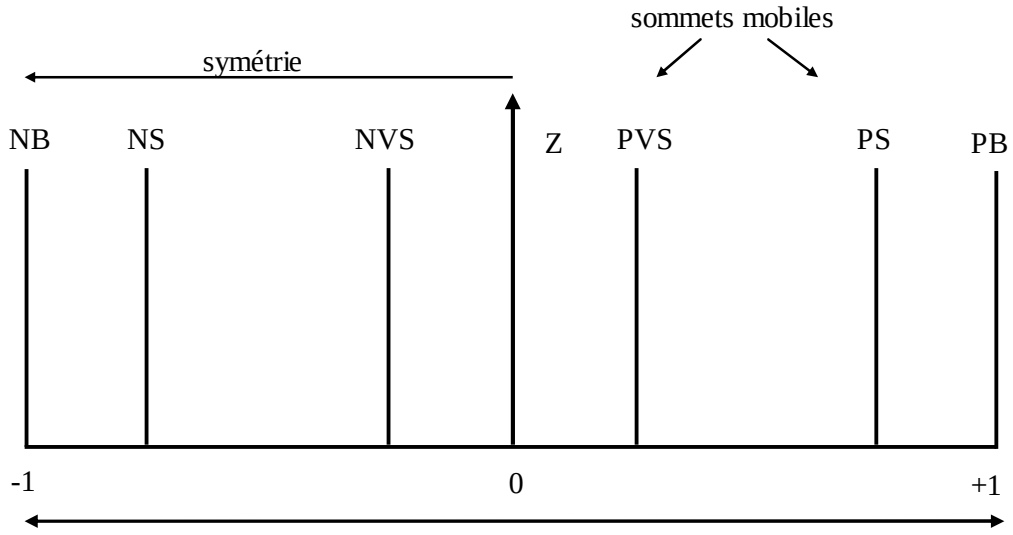


Figure I.5 Fonctions d'appartenance sur la commande

Le recensement de l'ensemble des paramètres du moteur flou donne alors :

- deux gains de normalisation en entrée (**gem** et **gdem**) et un gain de dénormalisation pour la sortie (**gm**).
- 7 fonctions d'appartenance pour chaque entrée.
- 49 paramètres dans la table de règles,
- 7 singletons de sortie.

Avec les choix que nous avons effectués, c'est-à-dire de fixer les coefficients de la table de règles et la plupart des fonctions d'appartenance en entrées et des singletons en sortie, plus les symétries, le problème de réglage se ramène à seulement 9 paramètres à régler sur les 73 initiaux. Il sera ainsi beaucoup plus aisé de régler le correcteur flou. Néanmoins, régler 9 paramètres en interactions entre eux demeure un réel problème auquel nous apportons des solutions.

I.2.5 Correcteurs flous de type PI et PID [1]

La présente section a pour but de finaliser la description des correcteurs à base de logique floue. A partir du moteur flou précédemment décrit, deux correcteurs, de type PI d'une part et de type PID d'autre part, vont pouvoir être construits.

La première étape consiste à définir la base de chacun de ces correcteurs, c'est-à-dire le moteur flou. La sortie de celui-ci, notée SPD, peut être définie à partir des fonctions kp et kd, qui sont respectivement les gains de l'action proportionnelle et dérivée du moteur flou, variant selon le point de fonctionnement, c'est-à-dire des valeurs de e et de, équation 1.1 :

$$S_{PD} = K_p(e, de).e + K_d(e, de).de \qquad (\mathbf{I.1})$$

Cette équation (**I.1**) peut être réécrite (**I.2**) en définissant des fonctions **k1** et **k2** de **e** et **de** :

$$S_{PD} = gm\left(\frac{1}{gem}.K_1(e, de) + \frac{1}{gdem}.\frac{de(t)}{dt}.K_2(e, de)\right) \qquad (\mathbf{I.2})$$

À partir de cette description du moteur flou, il devient aisé de construire les correcteurs.

I.2.5.A Correcteurs flous de type PI

Pour réaliser un correcteur de type PI, il suffit d'intégrer la sortie du moteur flou comme indiqué figure **I.6**.

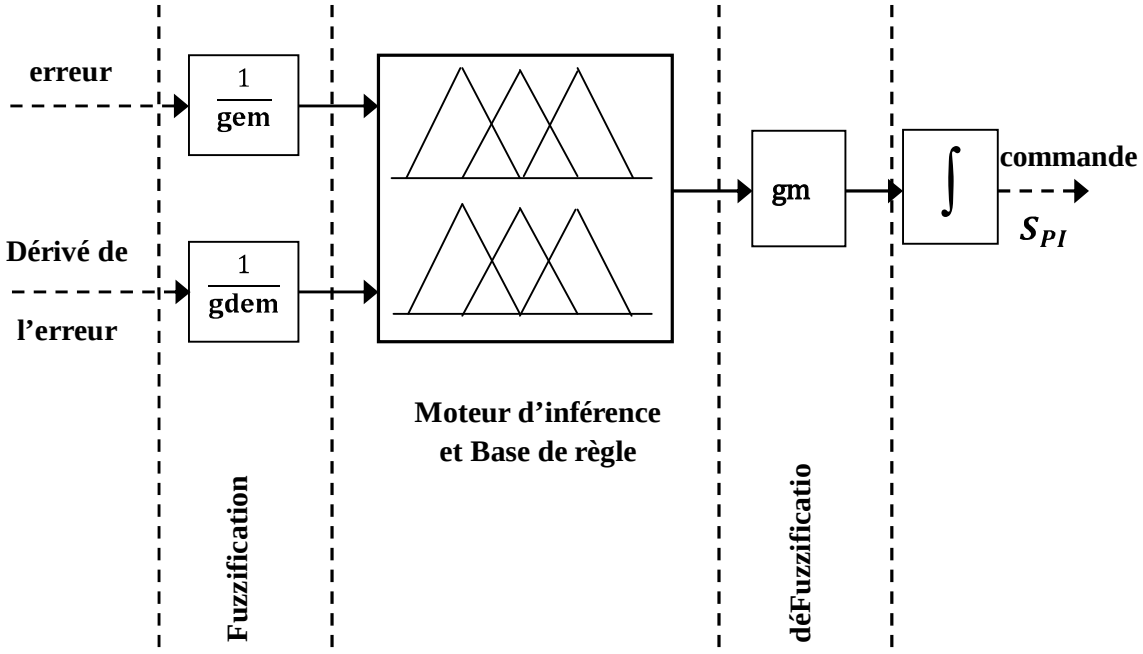


Figure I.6 Structure du correcteur flou de type PI

En notant SPI la sortie du contrôleur flou de type PI, il vient, équation **I.3** :

$$S_{PI} = \int S_{PD} \qquad (\mathbf{I.3})$$

En utilisant l'´equation **1.2**, il vient :

$$S_{PI} = \frac{gm}{gem}.\int K_1(e,de).e(t).dt + \frac{gm}{gdem}.\int K_2(e,de)\frac{de(t)}{dt}.dt \qquad (\mathbf{I.4})$$

I.2.5.B Correcteurs flous de type PID

Pour réaliser un correcteur de type PID, une partie intégrale va être ajoutée en parallèle au moteur flou, représentée figure **I.7**.

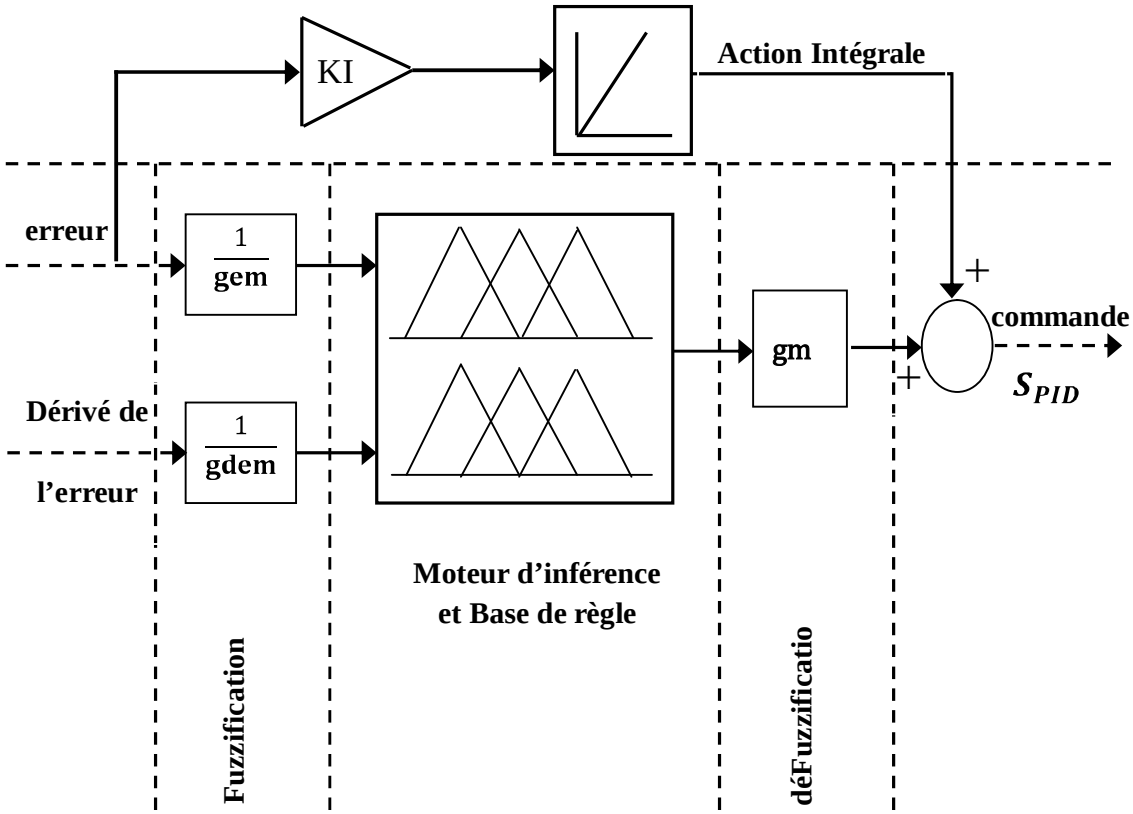


Figure I.7 Structure du correcteur flou de type PID

L'avantage de ce type de structure, basée sur le moteur flou proportionnel dérivé, est qu'il est possible de construire un correcteur de type PID sans avoir à calculer la dérivée seconde de l'erreur, qui risquerait d'amplifier de façon trop importante le bruit de mesure.

En notant **SPID** la sortie du contrôleur flou de type PID, il vient, équation **I.5** :

$$S_{PID} = Ki \int e(t).dt + S_{PD} \qquad (\mathbf{I.5})$$

Soit :

$$S_{PID} = Ki. \int e(t).dt + gm\left(\frac{1}{gem}.e(t).k_1(e,de) + \frac{1}{gdem}.\frac{de(t)}{dt}.k_2(e,de).dt\right) \qquad (\mathbf{I.6})$$

I.3 Conclusion

Dans ce chapitre, nous avons présenté les concepts de la logique floue, une description générale des constituants d'un système flou. Ce qui a permis de voir que la logique floue sert à représenter des connaissances incertaines et imprécises. Ainsi, le système flou sert à prendre une décision pouvant, en un certain sens, s'approcher du raisonnement humain. D'autre part, cette décision peut être prise sans toutefois avoir à estimer les entrées et les sorties de manière précise, mais plutôt qu'à partir de prédicats vagues.

Enfin, nous avons présenté un bref aperçu de la structure de base des contrôleurs flous (PI et PID). Un exemple d'illustration complétant ce chapitre sera abordé dans le chapitre suivant.

Chapitre II

Identification d'un four électrique et régulation de température

II.1. Introduction

La fonction de transfert réelle d'un procédé industriel est pratiquement impossible à déterminer. Il est alors nécessaire d'utiliser un modèle qui soit le plus représentatif possible de ce procédé. Pour cela, identifier un procédé est de rechercher, à partir des données expérimentales enregistrées au cours de fonctionnement, les paramètres qui caractérisent son fonctionnement.

On utilise des méthodes d'identification qui permettent de trouver un modèle de comportement traduisant, le plus fidèlement possible, le comportement du procédé autour d'un point de fonctionnement.
La connaissance des paramètres caractéristiques d'un procédé peut-être utile en particulier dans les domaines suivants:

- Réglage des actions dans les boucles de régulation.
- Choix des modes de régulation.

II.2 Présentation du four électrique [2]

La figure **II.1** présente un four électrique. Ce dernier est de forme cubique et contient un élément chauffant, un ventilateur pour le refroidissement et un thermocouple pour la mesure de température.

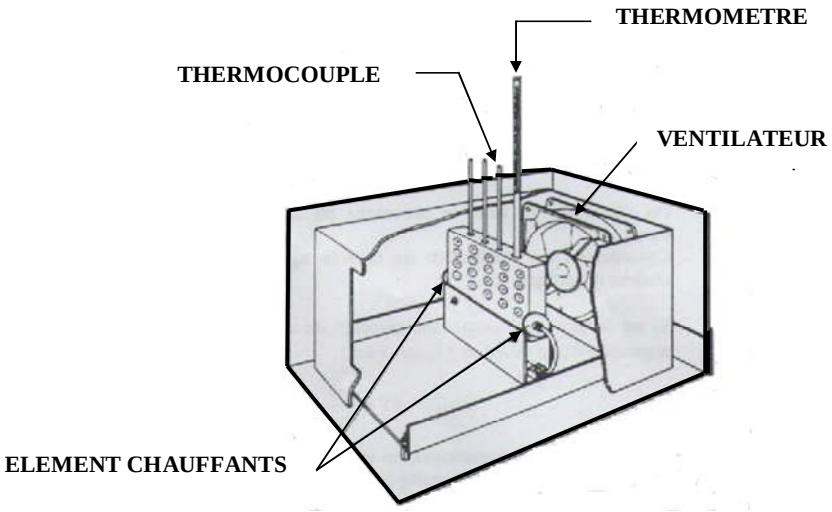


Figure II.1 Représentation du four

Les principaux éléments constituant du four sont :

II.2.1 Le ventilateur

Il permet de refroidir rapidement la plaque d'aluminium, afin de pouvoir réaliser les expériences suivantes après peu de temps.

II.2.2 Thermomètre

C'est un thermomètre au mercure, qu'on emploiera pour donner la température de référence. On insérera le thermomètre en un point de mesure spécial se trouvant sur la plaque d'aluminium.

II.2.3 Elément chauffant

L'élément chauffant est un actionneur (résistance électrique) qui s'échauffe lorsqu'il reçoit une puissance électrique.

II.2.4 Thermocouple [3]

Un thermocouple est un capteur qui mesure la température. Il se compose de deux métaux de natures différentes reliés à une extrémité. Il est créé dès lors que deux métaux différents entrent en contact, ce qui produit une faible tension en circuit ouvert au point de contact, qui varie en fonction de la température. Cette tension thermoélectrique est connue sous le nom de tension de Seebeck, d'après Thomas Seebeck qui l'a découverte en 1821. La tension n'est pas linéaire en fonction de la température. Cependant, pour de petites variations de température, la tension est approximativement linéaire, soit :

$$\Delta V \approx S\Delta T \qquad (\mathbf{II.1})$$

où ΔV est la variation de la tension, S est le coefficient de Seebeck et ΔT la variation de la température.

Les thermocouples sont disponibles dans différentes combinaisons de métaux ou de calibrages pour accommoder diverses utilisations. Les trois calibrages les plus communs sont K, T et J; le type K étant le plus populaire grâce à sa large palette de température et son prix peu élevé. Il existe aussi des calibrages à hautes températures, R, S, B, G, C, et D qui offrent une performance jusqu'à 2320°C. Ceux-ci sont fabriqués à partir de métaux précieux tel le platine/rhodium et le tungstène/rhénium qui sont par conséquent relativement chers.

Dans l'expérience sur le four, on a utilisé le type J qui est composé du Fer et du Constantan (alliage nickel+cuivre) comme représenter dans la figure ci-dessous. Il se caractérise par son bon fonctionnement dans le vide et dans une plage de température de 0 à 800 °C, mais n'est pas recommandé pour les basses températures, à cause de problèmes d'oxydation du fer et de l'azote.

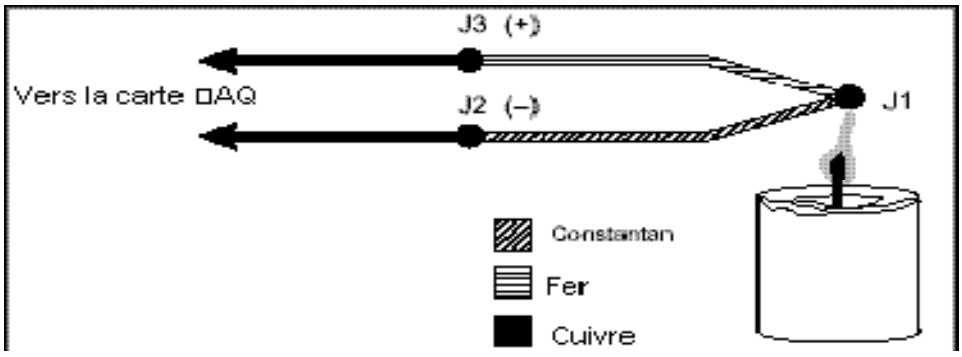


Figure II.2 Thermocouple de type J

La figure **II.2** représente un thermocouple de type J situé dans la flamme d'une bougie à une certaine température que nous souhaitons mesurer. Les deux fils du thermocouple sont connectés aux fils de connexion en cuivre d'une carte DAQ (acquisition de données). Le circuit contient trois jonctions métalliques différentes : J1, J2 et J3. J1 génère une tension de Seebeck proportionnelle à la température de la flamme de la bougie. J2 et J3 possèdent chacune leur propre coefficient de Seebeck et génèrent leur propre tension thermoélectrique proportionnelle à la température, au niveau des terminaux DAQ. Afin de déterminer la contribution en tension de J1, il faut connaître les températures des jonctions J2 et J3, ainsi que leurs relations tension/température. On peut ensuite soustraire les contributions des thermocouples parasites en J2 et J3 de la tension mesurée.

La figure **II.3** représente l'image réelle du thermocouple qu'on a utilisé pendant notre expérience au laboratoire :

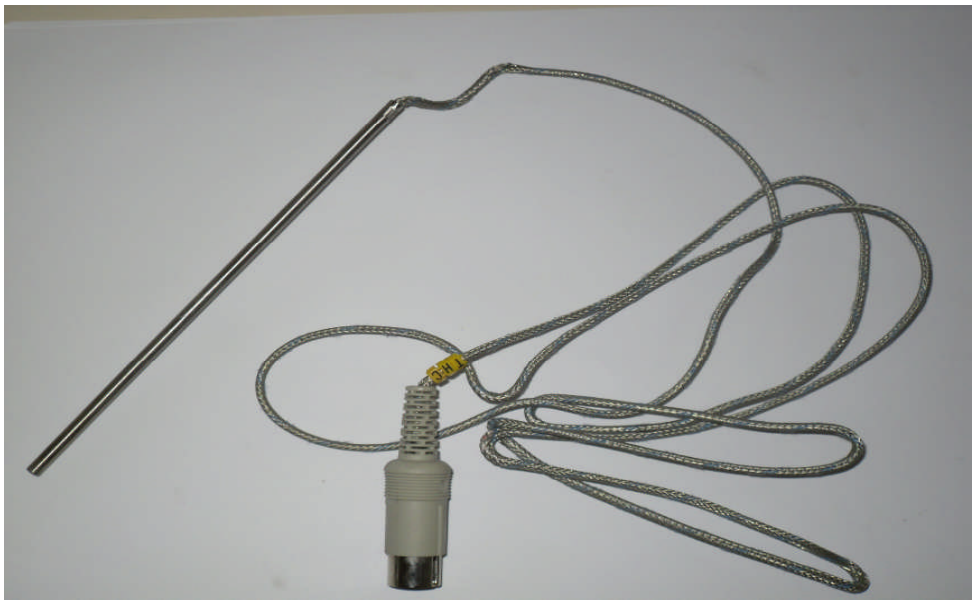


Figure II.3 Thermocouple de type J

Les principales caractéristiques de ce thermocouple sont :

- Constante de transduction : 53 µV/°C.
- Erreur : +/-2.2°C sur la plage de 0-270°C.
  +/- 0.75 % sur la plage de 270-760°C.

- Protection contre les éventuels agents atmosphériques au moyen d'un revêtement métallique.

II.3 Schéma de câblage

La famille S7-200 est constituée de micro-automates programmables utilisables dans des applications d'automatisation variées. Son dessin compact, son faible prix et son important jeu d'opérations en font une solution idéale pour la commande de petites applications. En outre, le large choix de tailles et de tensions de CPU ainsi que les multiples options de programmation disponibles offrent la souplesse nécessaire pour résoudre des problèmes d'automatisation.

La figure **II.4** présente le câblage entre le four, l'automate et le PC

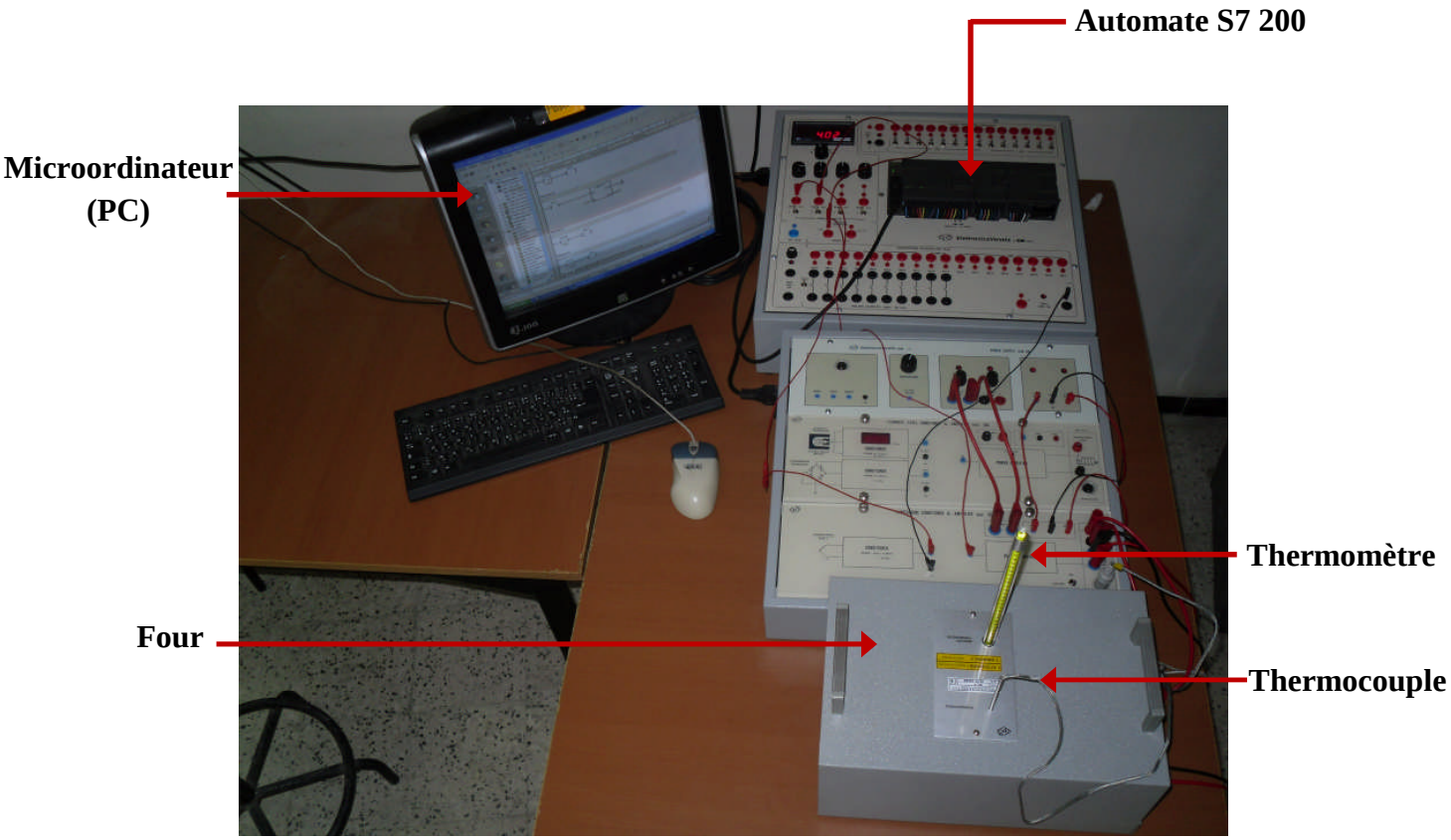


Figure II.4 Schéma de câblage automate, four et PC.

II.4 Identification en boucle ouverte [4]

II.4.1 Méthode de mesure

On envoie un signal d'entrée ***Y***(t) connu (figure **II.5**) et on enregistre le signal de sortie ***X***(t) qui est analysé par la suite.

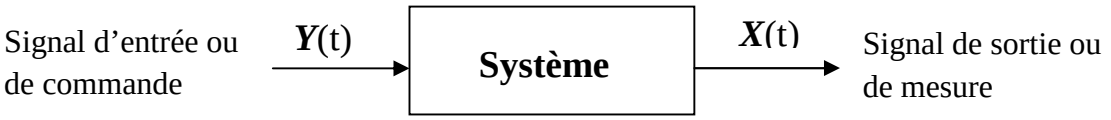


Figure II.5 système en chaine ouverte

Aux transformées de Laplace $\boldsymbol{Y}(\mathrm{p})$ et $\boldsymbol{X}(\mathrm{p})$ des fonctions $y(\mathrm{t})$ et $x(\mathrm{t})$, petites variations de Y(t) et X(t), on associe la fonction de transfert H(p) du système en chaine ouverte:

$$H(p) = \frac{\mathrm{X(P)}}{\mathrm{Y(P)}}$$

II.4.2 Exemple de signaux

1) signaux d'entrées

a)Echelon d'amplitude A

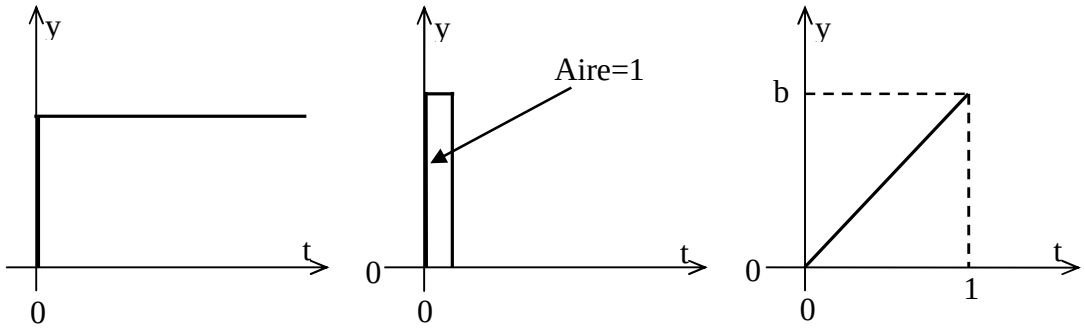
$$y(\mathrm{t}) = \mathrm{A}.u(\mathrm{t}) \longrightarrow Y(\mathrm{p}) = \frac{A}{p} \qquad (\mathbf{II.2})$$

b) Impulsion unitaire

$$y(\mathrm{t}) = \delta(t) \longrightarrow Y(\mathrm{p}) = 1 \qquad (\mathbf{II.3})$$

c) Rampe de pente b

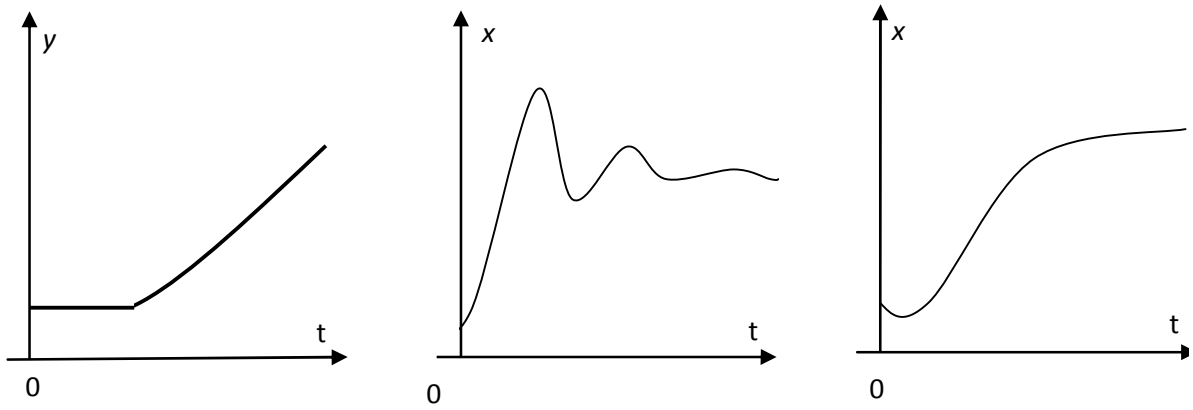
$$y(t) = b.t.u(t) \longrightarrow y(t) = \frac{b}{p^2} \qquad (\mathbf{II.4})$$



a)Echelon d'amplitude A b) Impulsion unitaire c) Rampe de pente b

Figure II.6 Signaux d'entrée y(t) les plus utilisés

2) Signaux de sortie



a) Courbe «intégratrice » b) Courbe « avec oscillations » c) Courbe «en S »

Figure II.7 Signal de sortie d'un système : courbes usuelles

II.4.3 Identification du four électrique

L'identification du four consiste à exciter le système par un signal constant (consigne). La réponse du système est enregistrée dans le tableau **II.1** de données.

L'allure de la courbe obtenue suggère la méthode à appliquer pour identifier le procédé. Le meilleur modèle est celui dont la réponse est la plus proche de la courbe obtenue expérimentalement. Pour notre identification nous avons choisi un échelon d'amplitude A=150°C (6volts) comme entrée et nous avons eu quelques valeurs dans le tableau **II.1.**

Nombres d'échantillons	temps (min)	La valeur de la mesure X(°C)
1	0,5	26,41
10	5	57,79
20	10	86,66
30	15	112,72
40	20	133,05
50	25	148,81
60	30	161,26
70	35	171,25
80	40	179,24
90	45	185,67
100	50	191,15
110	55	195,67
120	60	199,33
130	65	202,45
140	70	205,10
150	75	207,26
160	80	209,17
170	85	210,72
180	90	212,10
190	95	213,31
200	100	214,29
210	105	214,29
220	110	215,54

Tableau II.1 Tableau de mesures

La figure **II.8** présente l'entrée (consigne) ainsi que la sortie (réponse du système)

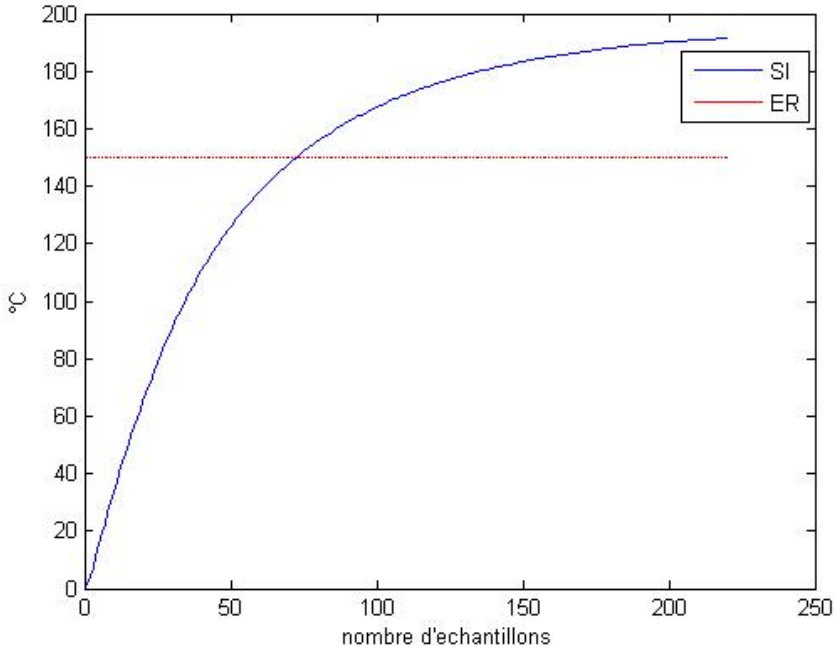


Figure II.8 Réponse du four à une entrée échelon

II.5 Analyse de la courbe

La forme obtenue de la réponse du système est en « s ». Parmi les méthodes d'identification qui existent pour l'analyse de différentes formes de réponse tel que en « s », « intégratrice » et avec « oscillation », on a choisit la méthode de STREJC et BROÏDA.

II.5.1 Méthode de STREJC [4]

La méthode de STREJC consiste à construire un modèle mathématique pour le processus dont la fonction de transfert est donnée par :

$$H(p) = \frac{X(p)}{Y(p)} = \frac{G_s}{(\theta p + 1)^n} \qquad (\mathbf{II.5})$$

Il s'agit de déterminer les valeurs de la constante de temps θ, Gs et n. Pour avoir ces valeurs on trace la tangente au point d'inflexion (figure **II.9**) puis on mesure les durées T_u et T_a. La

constante de temps θ et l'ordre n sont déterminés à partir du rapport $\frac{T_u}{T_a}$ (voir l'annexe 1). La variation ΔX est mesurée directement sur le graphe.

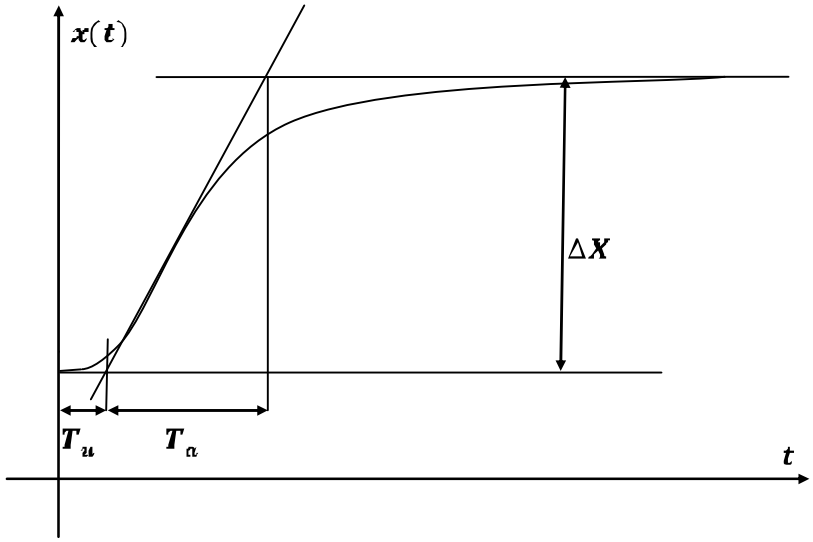


Figure II.9 courbe en « s » par la méthode de STRAJC

G_s est le gain statique du procédé tel que :

$$G_s = \frac{\Delta X}{A} \qquad (\mathbf{II.6})$$

Il n'y a pas d'intégration dans $H(p)$, le procédé est dit naturellement stable ou autoréglant.

Analyse de la courbe obtenue par la méthode de STREJC :

$$\Delta X = X_{max} - X_{min} \qquad X_{max} = 215℃ \qquad X_{min} = 24.2℃$$

Les résultats trouvés sont illustrés dans le tableau **III.2** ci-contre

$\Delta X(℃)$	$T_u(s)$	$T_a(s)$	G_s	$\theta(s)$	n
190,8	30	1410	1.27	900	1

Tableau II.2 Les résultats d'identification par la méthode de STREJC

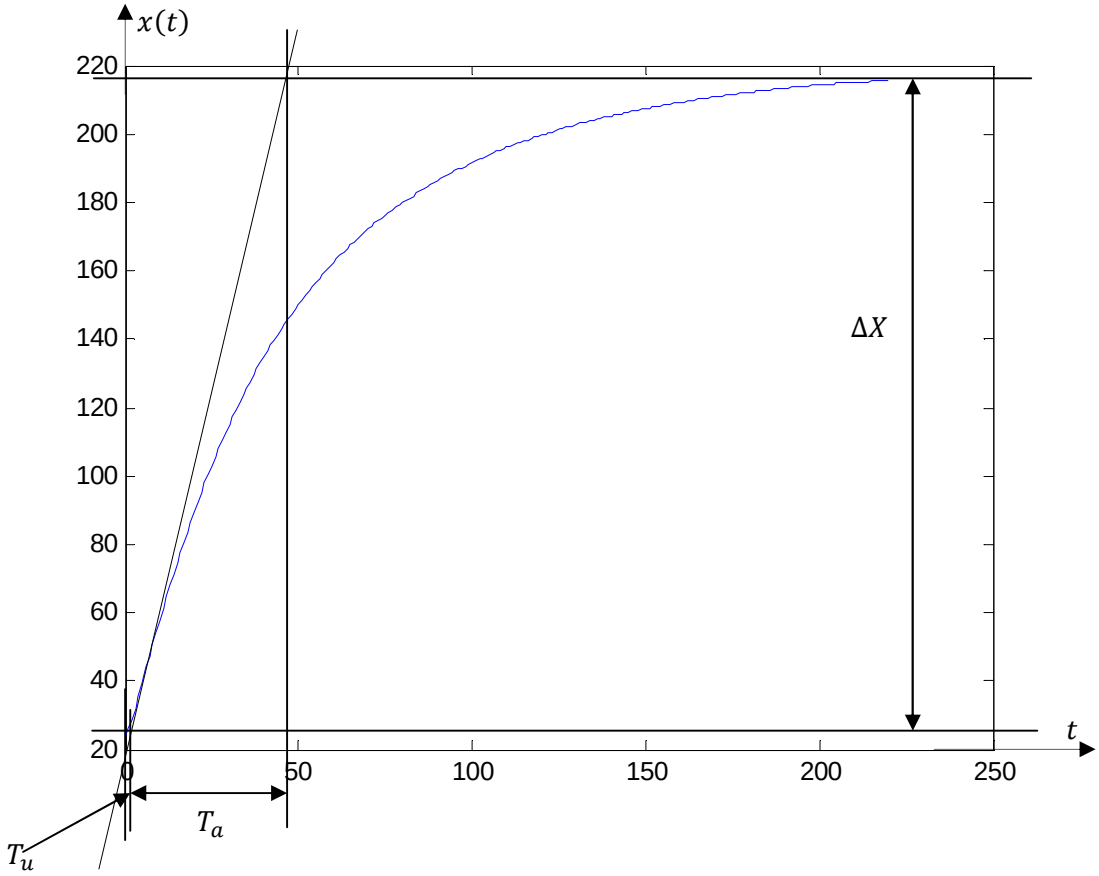


Figure II.10 Réponse du four analysée par la méthode de STTREJC

La fonction de transfert du système obtenue par cette méthode au point du fonctionnement est $H_s(p)$:

$$H_s(p) = \frac{1{,}27}{(900p + 1)} \qquad (\mathbf{II.7})$$

II.5.2 Méthode de BROÏDA [4][5]

La méthode de BROÏDA consiste à construire un modèle mathématique pour un système, dont la fonction de transfert est donnée par :

$$H(p) = \frac{G_s e^{-\tau p}}{(\theta p + 1)} \qquad (\mathbf{II.8})$$

Pour avoir les valeurs de la constante de temps θ, GS *et* et du temps mort τ : on mesure t_1 à

$0{,}28\Delta X$ et t_2 à $0{,}40\Delta X$ (figure **II.11**). La variation ΔX est mesurée directement sur le graphe puis on calcule $\boldsymbol{\theta} = 5{,}5(t_2 - t_1)$ et $\boldsymbol{\tau} = 2{,}8t_1 - 1.8t_2$

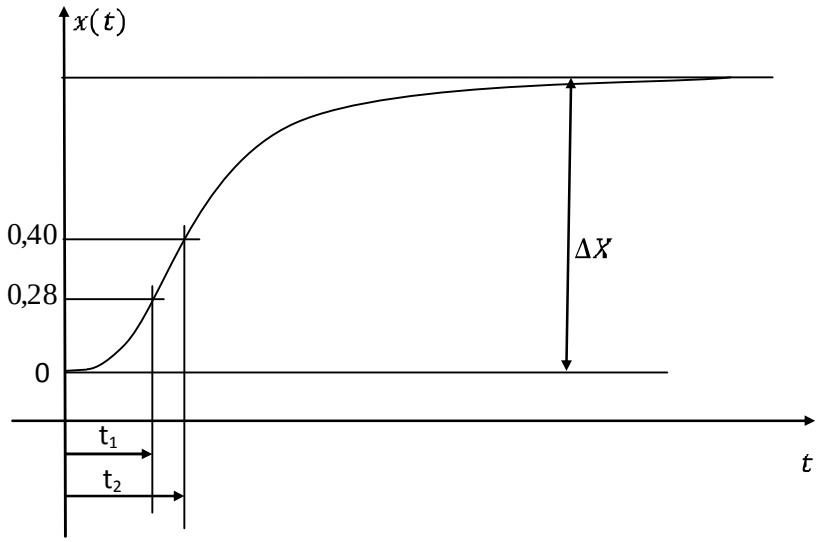


Figure II.11 Courbe en « s » analysée par la méthode de BROÏDA

G_s est le gain statique du procédé tel que :

$$G_s = \frac{\Delta X}{A} \qquad (\mathbf{II.9})$$

Il n'y a pas d'intégration dans $H(p)$, le procédé est dit naturellement stable ou autoréglant.

Analyse de la courbe obtenue par la méthode de BROÏDA :

Les résultats trouvés sont illustrés dans le tableau ci-dessous

ΔX(°C)	$t_1(s)$	$t_2(s)$	G_s	$\tau(s)$	$\theta(s)$
190,8	465	660	1.27	114	1072

Tableau II.3 Les résultats d'identification par la méthode de BROÏDA

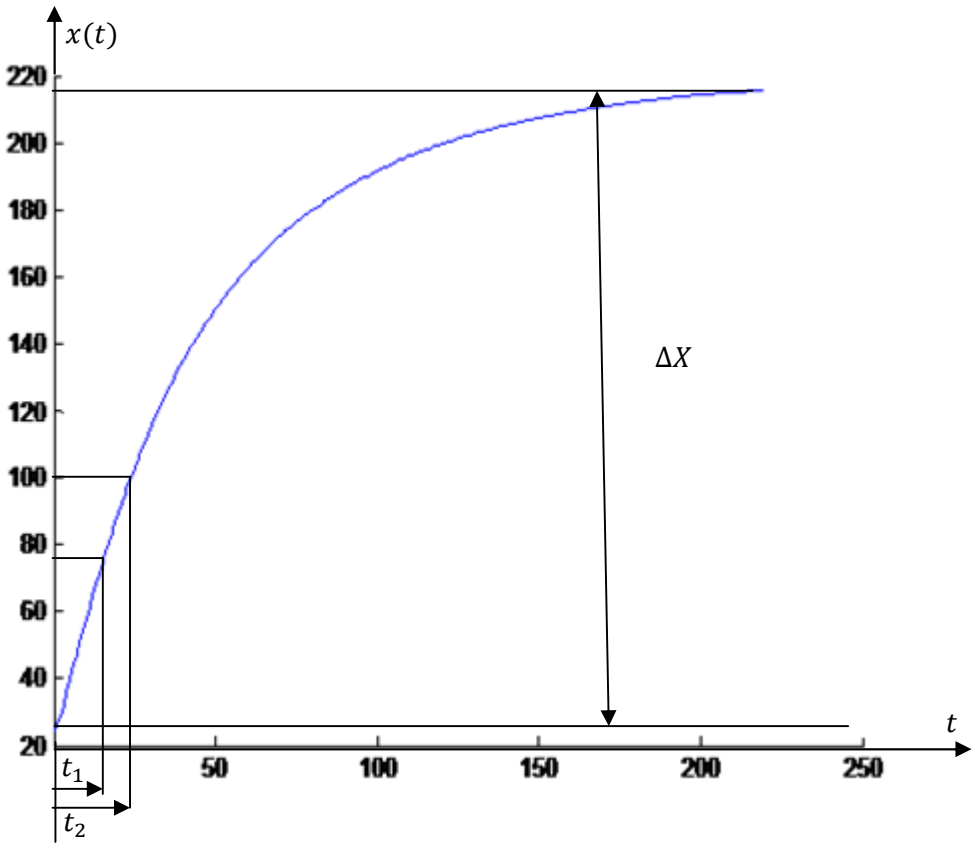


Figure II.12 Réponse du four analysée par la méthode de BROÏDA

La fonction de transfert du système obtenue par cette méthode au point du fonctionnement est $H_b(p)$:

$$H_b(p) = \frac{1{,}27e^{-114p}}{(1072p + 1)} \qquad (\mathbf{II.10})$$

Remarque II.1 : Les deux méthodes d'identification utilisées dans cette partie ont données deux modèles identique du coté degré (ordre du système) avec une nuance différence entre les paramètres.

Simulation des modèles obtenus

H_s: Réponse du modèle obtenu par la méthode de Strejc.

H_b: Réponse du modèle obtenu par la méthode de Broïda.

RP: Réponse du four.

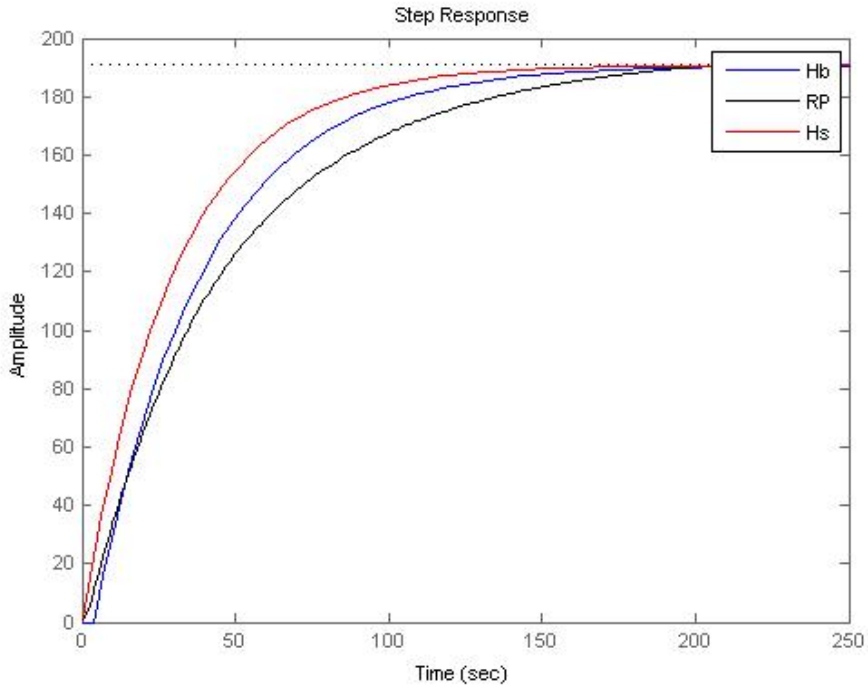


Figure II.13 Réponses des systèmes identifiés avec celle du procédé à une entrée échelon

Hb : Réponse du modèle identifié par la méthode de BROÏDA

Hs : Réponse du modèle identifier par la méthode de STREJC

RP : Réponse du procédé (four)

Ces étapes permettent de déduire que le comportement du processus est du premier ordre. On choisit comme fonction de transfert H_s pour répondre à l'objectif du premier chapitre.

Pour la conception d'un contrôleur basé sur le modèle choisit (H_s) dont la structure est présentée par la figure **II.14**

II.6 Elaboration d'un régulateur PID flou

Schémas de simulation

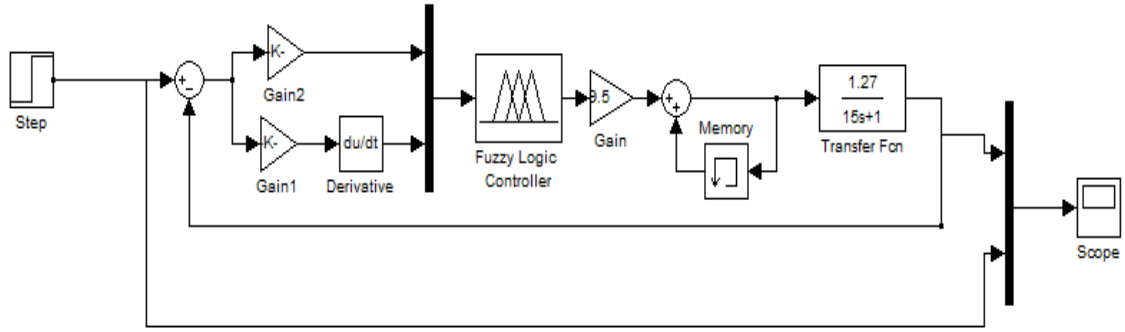


Figure II.14 Structure d'un PID flou.

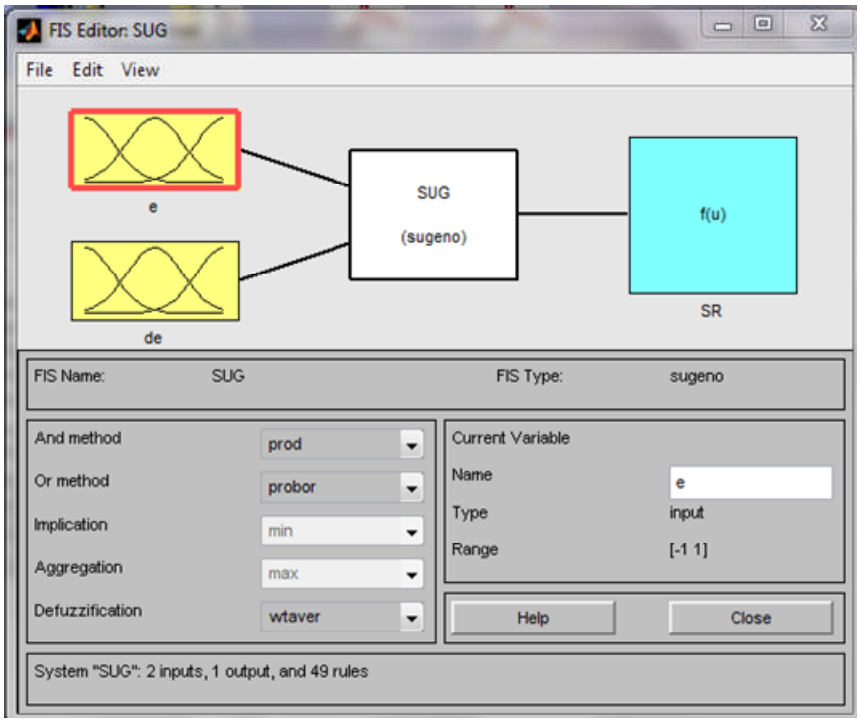


Figure II.15 Moteur d'inférence

On choisit trois fonctions d'appartenances pour chaque entrée, puis on construit la table de règle qui contient neuf règles.

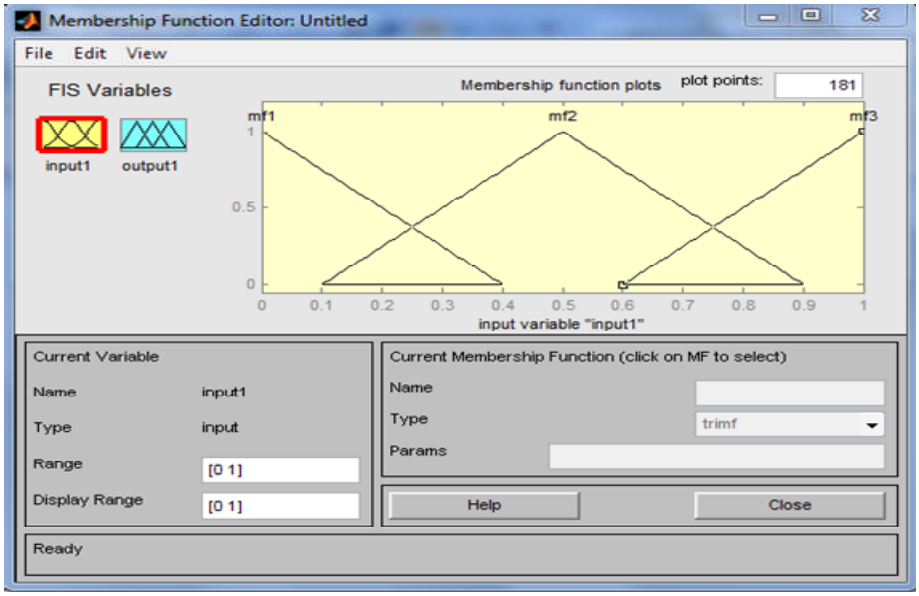


Figure II.16 Fonction d'appartenance

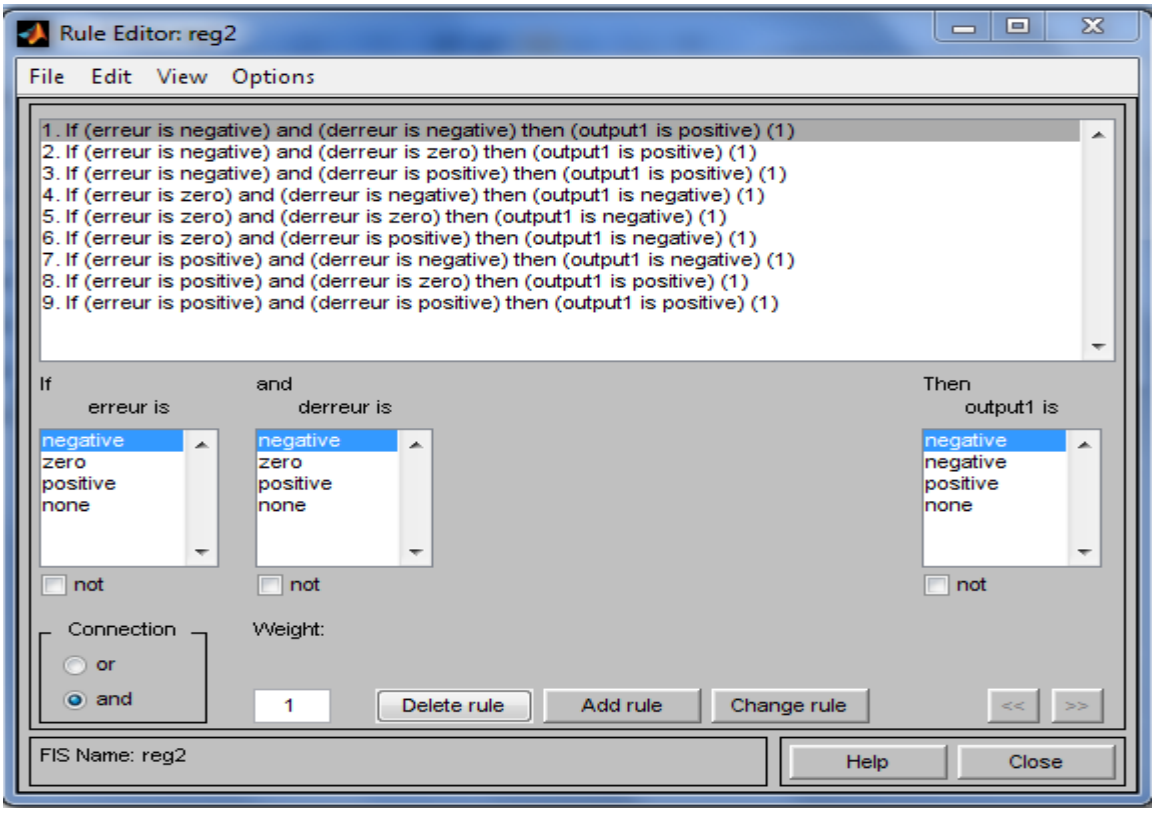


Figure II.17 Table de règles

Le choix des paramètres du correcteur se fait par tâtonnement, tout en cherchant d'obtenir de meilleures performances en boucle fermée.

Gain de normalisation de l'erreur : 0.01.

Gain de normalisation de la dérivée de l'erreur : 0.025

Gain de dénormalisation : 9.5

Résultat de simulation

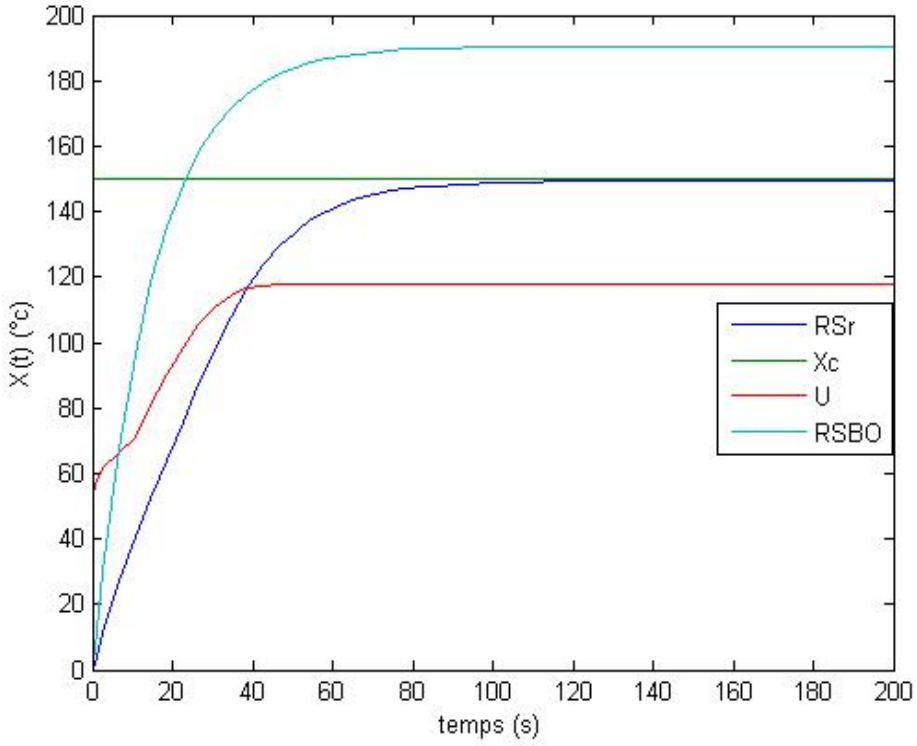


Figure II.18 résultat de simulation pour un régulateur flou

RSr : Réponse du système régulé

RSBO : Réponse du système en boucle ouvert

Xc : Consigne

U : Commande

II.7 Elaboration d'un régulateur PID classique

Schéma de simulation

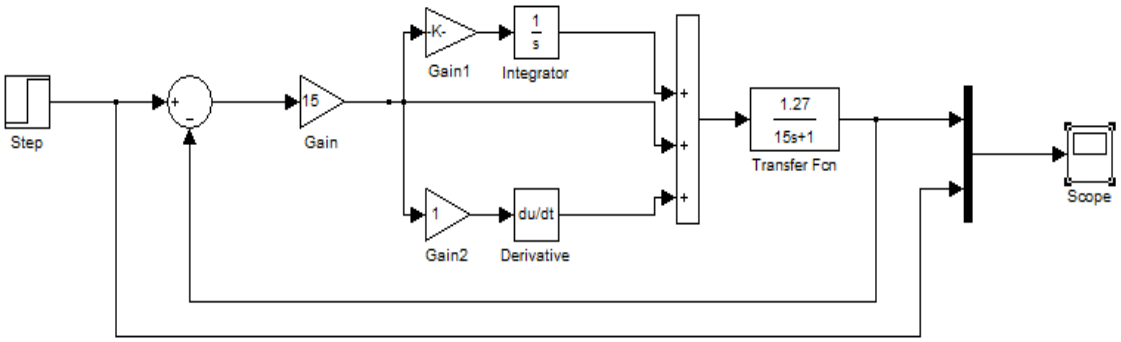


Figure II.19 schéma de simulation pour un PID classique

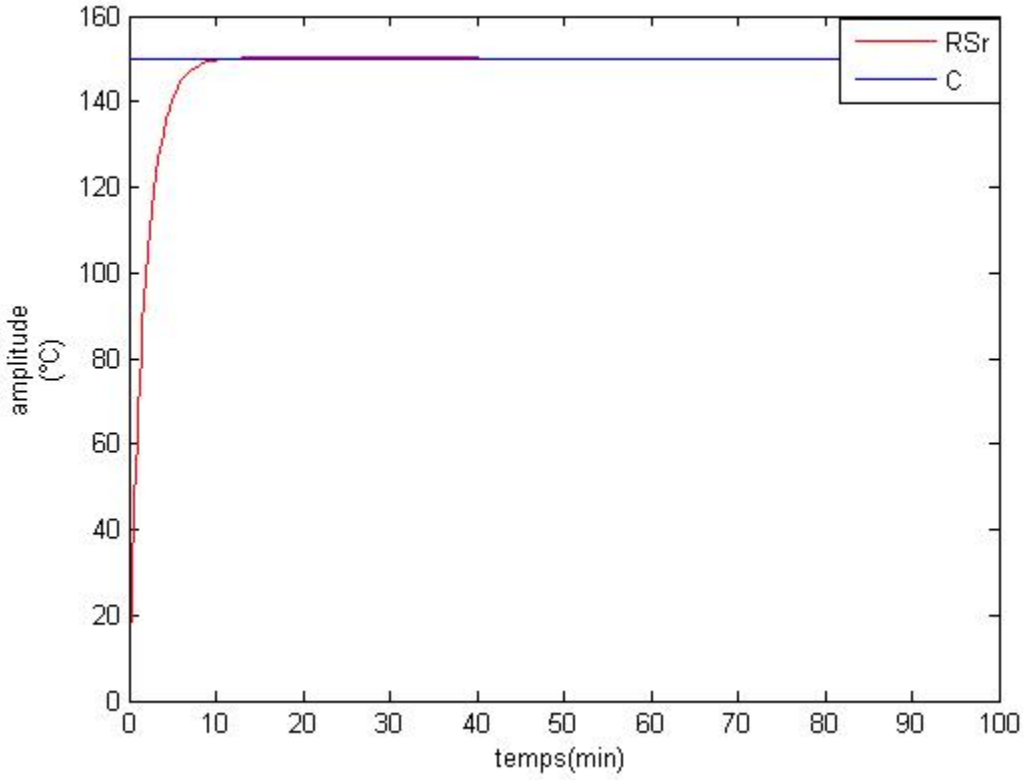


Figure II.20 Résultat de simulation pour un régulateur PID classique

Le temps de réponse est de 5.34 minutes. L'écart statique est nul.

II.8 Conclusion

Dans ce chapitre une étude restreinte sur l'identification des procédés a été présentée pour l'identification du modèle mathématique (fonction de transfert) du four, puis une simulation sur Matlab des modèles obtenu, inséré dans une boucle de régulation par la logique floue et par un PID classique. Enfin, on a montré que la logique floue peut être utilisée pour la commande des systèmes.

Le chapitre suivant sera consacré pour un outil très important dans l'industrie garantissant une certaine flexibilité qui est l'automate programmable industriel (API).

Chapitre III

Techniques de programmation des Automates Programmable Industriels (APIs)

III.1 Introduction

Les automates programmables sont apparus aux USA vers 1969. Ils étaient destinés à l'origine à automatiser les chaînes de montages automobiles. C'est en 1971 Qu'ils firent leur apparition en France. Ils sont de plus en plus employés dans toutes les industries.

L'automatisation par automate programmable industriel consiste à simplifier la logique de l'automatisation, réaliser des fonctions plus complexes, utiliser moins d'appareils de commande, modifier et améliorer le système juste en modifiant le programme, câblage moins encombrant et moins de risques, puis la fiabilité est meilleure et possible de diagnostiquer les pannes pour éviter les temps d'arrêt.

III.2 Place des automates programmables dans l'industrie

Parmi les systèmes de traitement de l'information, les API occupent une place de choix. Les équipements notés « commande » sont souvent des automates. Remplaçant initialement des ensembles en technologie câblée (relais électromagnétiques ou statiques, composants pneumatiques), ils constituent de plus en plus un maillon fiable et efficace entre le calculateur, qui a plutôt un rôle de gestion et l'appareillage de terrain (capteurs et actionneurs).

III.3 Définition d'un automate programmable industriel

L'automate programmable industriel (API) est un appareil électronique qui comporte un processeur pour l'exécution d'un programme stocké dans une mémoire programmable par un utilisateur automaticien à l'aide d'un langage adapté, pour le stockage interne des instructions composant les fonctions d'automatisme comme par exemple :

- Logique séquentielle et combinatoire ;
- Temporisation, comptage, décomptage, comparaison ;
- Calcul arithmétique ;
- Réglage, asservissement, régulation, etc, pour commander, mesurer et contrôler au moyen d'entrées et de sorties (logiques, numériques ou analogiques) différentes sortes de machines ou de processus, en environnement industriel.

III.4 Rôle des APIs sur les systèmes industriels

Les systèmes industriels de production sont de plus en plus automatisés et demande pour cela l'emploi d'Automate Programmable Industrielle (API). Ils permettent de commander des systèmes plus ou moins complexe (de la petite emballeuse, à la chaîne de montage automobile).

Ils permettent une grande souplesse et des avantages non négligeable tel que:

- L'exploitation de la machine ;
- Performance (vitesse de production) ;
- La modification d'utilisation (programme) ;
- Gain de place ;
- Le dépannage plus rapide ;
- Des installations de plus en plus complexes.

III.5 Architecture des automates [7]

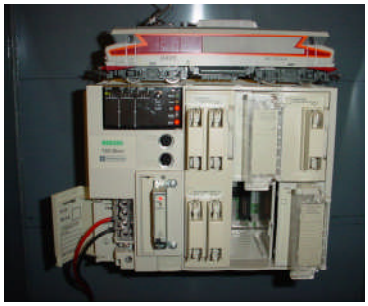
III.5.1 Aspect extérieur

Les automates peuvent être de type compact ou modulaire **:**
Pour le type compact, on distinguera les modules de programmation (LOGO de Siemens, ZELIO de Schneider, MILLENIUM de Crouzet ...).

Il intègre le processeur, l'alimentation, les entrées et les sorties. Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage rapide, E/S analogiques ...) et recevoir des extensions en nombre limité.

Ces automates, de fonctionnement simple, sont généralement destinés au contrôle de petits automatismes.

Automate compact (Allen-bradley)



Automate CROUZET Millénium II

Automate SCHNEIDER Micro

Figure III.1 automates de type compact.

Pour le type modulaire, le processeur, l'alimentation et les interfaces d'entrées / sorties résident dans des unités séparées (modules) et sont fixées sur un ou plusieurs racks contenant le "fond de panier" (bus plus connecteurs).

Ces automates sont intégrés dans les automatismes complexes où puissance, capacité de traitement et flexibilité sont nécessaires.

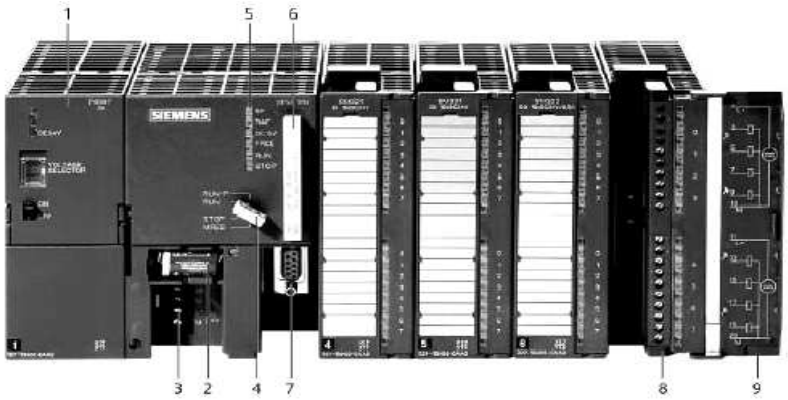


Figure III.2 automates de type Modulaire

1 Module d'alimentation
2 Pile de sauvegarde
3 Connexion au 24V cc
4 Commutateur de mode (à clé)
5 LED de signalisation d'état et de défauts
6 Carte mémoire
7 Interface multipoint
8 Connecteur frontal
9 Volet en face avant

III.5.2 Structure interne

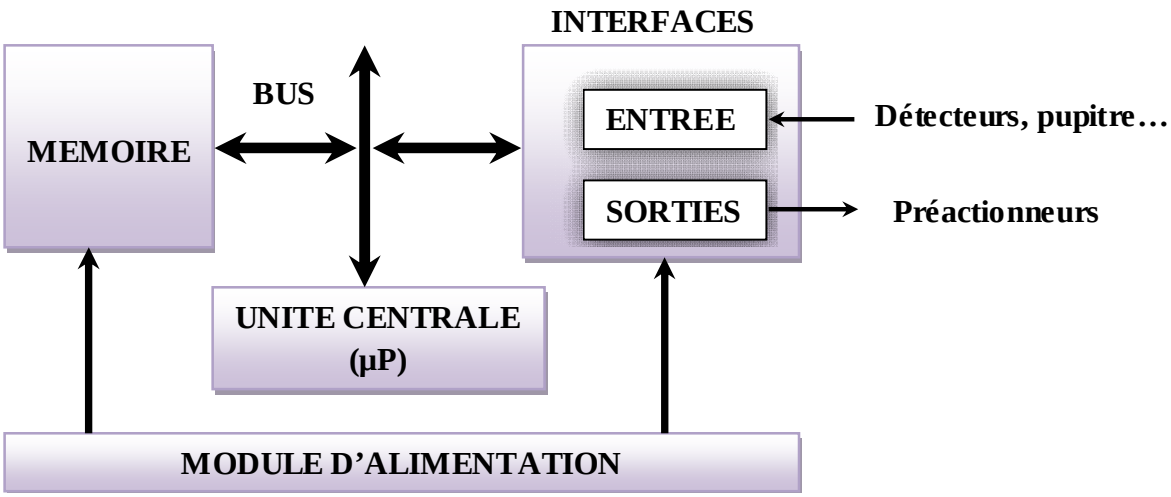


Figure III.3 Structure interne des automates.

- Module d'alimentation : ce module doit fournir l'énergie nécessaire au fonctionnement correct de l'ensemble de l'automate. Il sera dimensionné en fonction des consommations des différentes parties ;
- Unité centrale : elle traite les données et réalise toutes les fonctions logiques, arithmétiques et de traitement numérique (transfert, comptage, temporisation ...). Elle contient en mémoire le programme et élabore donc les ordres de commande. Son cœur est composé d'un microcontrôleur alimenté en 5 volts ;
- Les bus interne : ce sont des circuits chargés d'adapter en tension et en courant les signaux entre l'unité centrale et les entrées-sorties. Ils assurent en outre un isolement entre les entrées-sorties et l'unité centrale ;
- Mémoires : Elles permettent de stocker le système d'exploitation (ROM ou PROM), le programme (EEPROM) et les données système lors du fonctionnement (RAM). Cette

dernière est généralement secourue par pile ou batterie. On peut, en règle générale, augmenter la capacité mémoire par adjonction de barrettes mémoires type PCMCIA ;

- Interfaces d'entrées / sorties :
  - Interfaces d'entrées : ce sont des circuits spécialisés capables de recevoir en toute sécurité pour l'automate les signaux issus des capteurs. Elles peuvent être logiques (T.O.R.), analogiques, ou numériques.
  - Interfaces de sorties : ce sont des circuits spécialisés capables de contrôler en toute sécurité pour l'automate les circuits extérieurs. Elles peuvent être logiques (T.O.R.), analogiques, ou numériques.

III.6 Structure d'un système automatisé [7]

Tout système automatisé peut se décomposer selon le schéma ci-dessous :

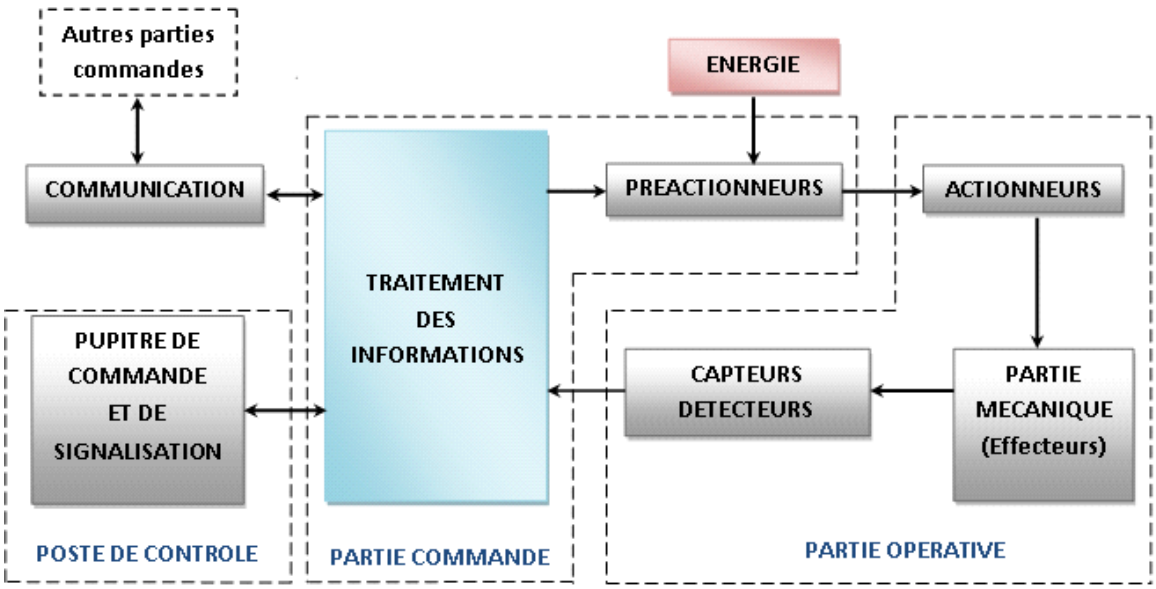


Figure III.4 Structure d'un système automatisé.

III.6.1 Partie opérative

Elle agit sur la matière d'œuvre afin de lui donner sa valeur ajoutée.

Les actionneurs (moteurs, vérins) agissent sur la partie mécanique du système qui agit à son tour sur la matière d'œuvre. Les capteurs / détecteurs permettent d'acquérir les divers états du système.

III.6.2 Partie commande

Elle donne les ordres de fonctionnement à la partie opérative.

Les pré-actionneurs permettent de commander les actionneurs ; ils assurent le transfert d'énergie entre la source de puissance (réseau électrique, pneumatique …) et les actionneurs.

Exemple : contacteur, distributeur …

Ces pré-actionneurs sont commandés à leur tour par le bloc traitement des informations. Celui-ci reçoit les consignes du pupitre de commande (opérateur) et les informations de la partie opérative transmises par les capteurs / détecteurs.

En fonction de ces consignes et de son programme de gestion des tâches (implanté dans un automate programmable ou réalisé par des relais (on parle de logique câblée)), elle va commander les préactionneurs et renvoyer des informations au pupitre de signalisation ou à d'autres systèmes de commande et/ou de supervision en utilisant un réseau et un protocole de communication.

III.6.3 Poste de contrôle

Il est Composé des pupitres de commande et de signalisation**,** il permet à l'opérateur de commander le système (marche, arrêt, départ cycle …).

Il permet également de visualiser les différents états du système à l'aide de voyants, de terminal de dialogue ou d'interface homme-machine (IHM).

III.7 Domaines d'emploi des automates

On utilise un API dans tous les secteurs industriels pour la commande des machines (convoyage, emballage ...) ou des chaînes de production (automobile, agroalimentaire ...) ou il peut également assurer des fonctions de régulation de processus (métallurgie, chimie ...).
Il est de plus en plus utilisé dans le domaine du bâtiment (tertiaire et industriel) pour le contrôle du chauffage, de l'éclairage, de la sécurité ou des alarmes.

III.8 Nature des informations traitées par l'automate [8]

Les informations peuvent être de type :

III.8.1 Tout ou rien (T.O.R.)

L'information ne peut prendre que deux états (vrai/faux, 0 ou 1 …). C'est le type d'information délivrée par un détecteur, un bouton poussoir…

III.8.2 Analogique

L'information est continue et peut prendre une valeur comprise dans une plage bien déterminée. C'est le type d'information délivrée par un capteur (pression, température …)

III.8.3 Numérique

L'information est contenue dans des mots codés sous forme binaire ou bien hexadécimale. C'est le type d'information délivrée par un ordinateur ou un module intelligent.

III.9 Programmation [8]

C'est un des atouts majeurs des APIs puisqu'elle permet une multitude de traitements des informations reçues sans toucher à la configuration matérielle. Certaines modifications peuvent même s'effectuer alors que l'automate est en marche. Il faut toutefois comprendre le fonctionnement du processeur, qui impose certaines contraintes et choisir le langage le plus approprié dans le cadre du problème à résoudre.

Elle peut s'effectuer de trois manières différentes

- Sur l'A.P.I. lui-même à l'aide de touches ;
- Avec une console de programmation reliée par un câble spécifique à l'API ;
- Avec un PC et un logiciel approprié.

III.9.1 Les différents langages utilisés pour la programmation

III.9.1.A Liste d'instructions (IL)

C'est un langage textuel, qui rappelle par certains aspects l'assembleur employé pour la programmation des microprocesseurs. Une instruction débute sur une ligne, comporte un opérateur, un ou plusieurs opérandes. On peut introduire des étiquettes et des commentaires. La structure des champs est donc la suivante :

ETIQUETTE (facultatif)	**OPERATEUR**	**OPERANDE(S)**	**COMMENTAIRE (facultatif)**

Ce langage n'est pas celui jugé le plus pratique par la plupart des utilisateurs, car le passage d'une analyse souvent semi-graphique à la programmation n'est pas toujours aisé, pas plus que la mise au point du programme.

III.9.1.B Langage littéral structuré (ST)

Issu de l'informatique, il présente des analogies avec le PASCAL. Il offre des possibilités telles que :

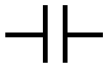
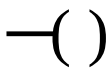
```
SI

   ALORS
   AUTREMENT

FIN SI
```

Il utilise des expressions, assemblage ordonné d'opérateurs et d'opérandes, avec des priorités, des parenthèses, etc.

Ce langage facilite donc la mise en œuvre d'algorithmes complexes comportant beaucoup de traitement numérique. Il permet aussi des passerelles vers des blocs extérieurs, ainsi que des traitements particuliers (chaînes de caractères). En contrepartie, il est moins commode pour la mise au point de fonctions booléennes.

III.9.1.C Langage à contacts (LD)

Il utilise désormais systématiquement une forme graphique, alors c'est un langage graphique d'où son appellation de schéma à contacts (il est encore appelé diagramme à relais). Il traduit directement l'équation en un schéma électrique avec des symboles particuliers :

- Contact à fermeture : 
- Contact à ouverture : 
- Bobine : 

L'application est représentée par un ou plusieurs réseaux. Un réseau est formé d'éléments graphiques, et éventuellement de blocs fonctionnels, connectés entre eux, partant d'une barre d'alimentation à gauche, et se terminant par une barre à droite (facultative).

Ce langage est très efficace pour des systèmes combinatoires (où les sorties ne dépendent que des entrées). Il est très populaire dans le monde industriel, et reste la référence aux États-Unis par exemple. Il est plus lourd à manipuler dans le cas de fonctions séquentielles (où intervient implicitement le temps, les sorties dépendant des entrées mais aussi de leur état précédent).

III.9.1.D Diagramme fonctionnel en séquences (SFC)

Il s'agit de ce que l'on dénomme couramment les langages « GRAFCET ». Langage graphique ou des fonctions sont représentées par des rectangles avec les entrées à gauche et les sorties à droites. Les blocs sont programmés (bibliothèque) ou programmables.
Utilisé par les automaticiens.

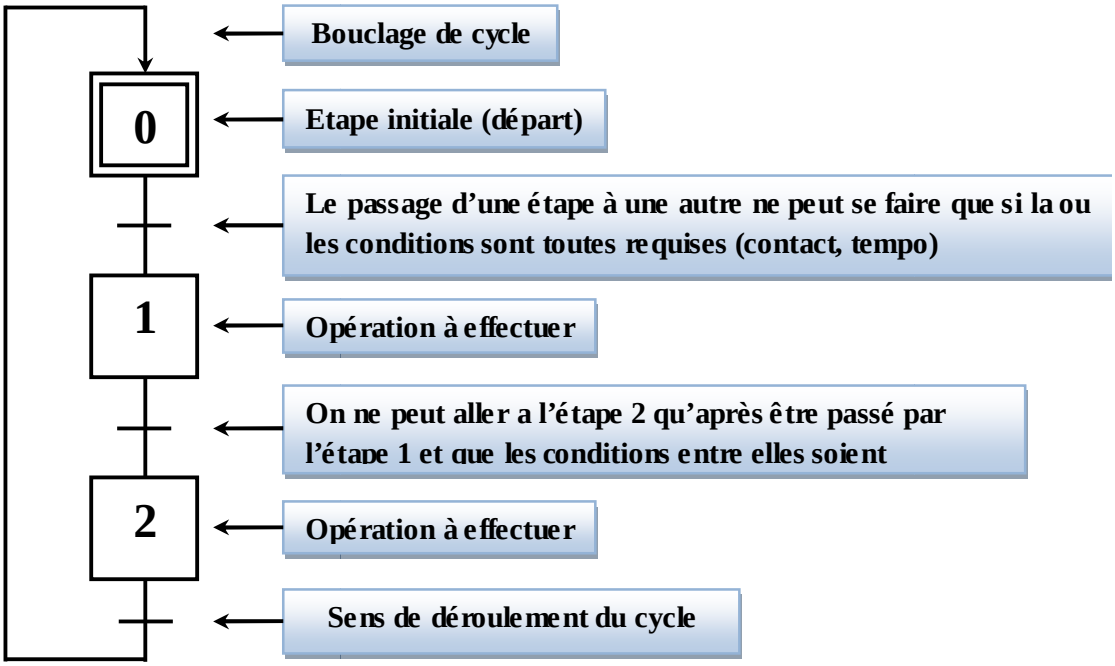
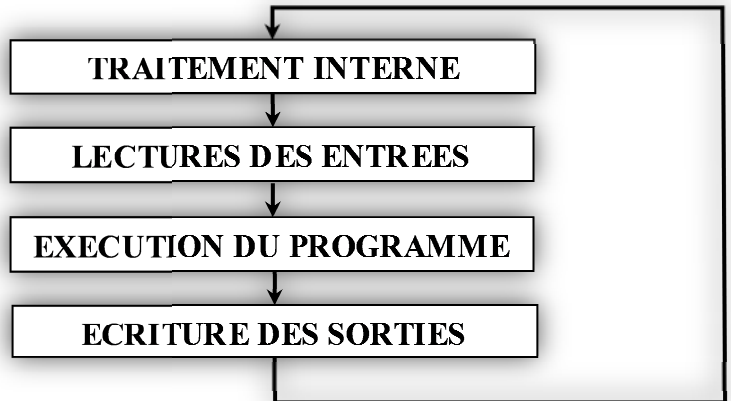


Figure III.5 Langage GRAFCET

III.10 Traitement du programme automate



- Traitement interne : L'automate effectue des opérations de contrôle et met à jour certains paramètres systèmes (détection des passages en RUN / STOP, mises à jour des valeurs de l'horodateur, ...) ;
- Lecture des entrées : L'automate lit les entrées (de façon synchrone) et les recopie dans la mémoire image des entrées ;
- Exécution du programme : L'automate exécute le programme instruction par instruction et écrit les sorties dans la mémoire image des sorties.

- Ecriture des sorties : L'automate bascule les différentes sorties (de façon synchrone) aux positions définies dans la mémoire image des sorties.

Ces quatre opérations sont effectuées continuellement par l'automate (fonctionnement cyclique).

On appelle scrutation l'ensemble des quatre opérations réalisées par l'automate et le temps de scrutation est le temps mis par l'automate pour traiter la même partie de programme. Ce temps est de l'ordre de la dizaine de millisecondes pour les applications standards.

III.11 Technique de programmation sur automate

Afin de programmer ou d'implémenter des algorithmes sur un automate, une connaissance préalable est nécessaire sur les outils de base qui existe dans la bibliothèque de chaque automate.
La richesse des automates en module (blocs) fonctionnels permet de développer plusieurs programmes visant un même objectif .parmi ces blocs on trouve :

- Temporisateurs.
- Comparateurs.
- Blocs de transfert de données vers les sorties ou vers les cases mémoires.
- Blocs de régulation : PID.
- Bascules.
- Opérateurs (logique, mathématique)…etc.

Ces outils seront d'apport considérable pour l'implémentation des régulateurs à base de logique floue sur les automates programmable industriels.

III.12 Capacité d'un automate

Elle est déterminée par le nombre de ses entrées, de ses sorties, ainsi que sa capacité mémoire nécessaire à stocker le programme dans l'Unité Centrale.

Le tout détermine bien évidemment son prix qui peut varier d'environ 80 euros pour un modèle 8 entrées/4 sorties à plus de 15 000 euros pour un de 1000 entrées/sorties.

III.12.1 Choix d'un automate programmable industriel

Le choix d'un automate programmable est en premier lieu le choix d'une société ou d'un groupe et les contacts commerciaux et expériences vécues sont déjà un point de départ.

Voici quelques critères essentiels du choix d'un automate programmable industriel :

- Les compétences/expériences de l'équipe d'automaticiens en mise en œuvre et en programmation de la gamme d'automate.
- La qualité du service après-vente.
- Les capacités de traitement du processeur (vitesse, données, opérations, temps réel...).
- Le type des entrées/sorties nécessaire.
- Le nombre d'entrées/sorties nécessaire.

III.12.2 Mise en œuvre

A partir d'un problème d'automatisme donné, dans lequel on a défini les commandes, les capteurs et le processus à réaliser, il faut :

- Etablir le GRAFCET (ou l'organigramme, le schéma à contact, logigramme, équations logiques...).
- Ecrire le programme (écriture des instructions).
- Rentrer le programme à l'aide de la console de programmation.
- Transférer le programme dans l'unité centrale de l'automate.
- Tester à vide (mise au point du programme, simulation).
- Raccorder l'automate à la machine.

III.13 Avantages et Inconvénients

III.13.1 Les Avantages

- Simplification du câblage.
- Modifications du programme faciles à effectuer par rapport à une logique câblée.
- Enormes possibilités d'exploitation.
- Fiabilité professionnelle.
- Possibilité de gérer des tâches parallèles.

- Réalisable sur des microprocesseurs peu puissants.
- Simplicité de la programmation : Pour des applications très simples, il existe des langages ne nécessitant quasiment aucune connaissance en programmation comme exemple le langage « CONTACT ».

III.13.2 Les inconvénients

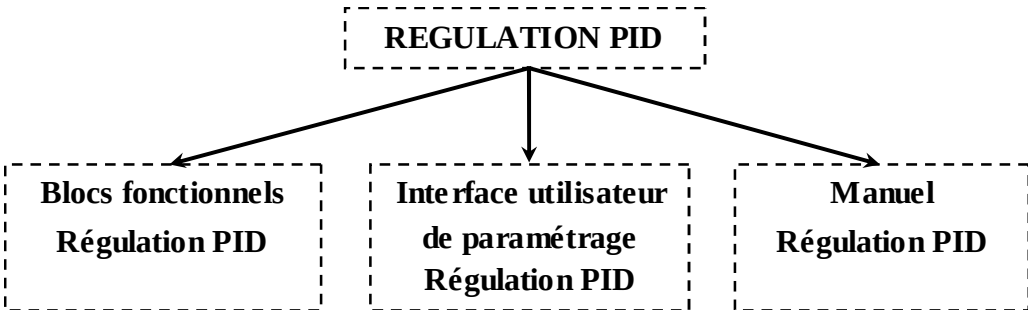
- Le coût élevé du matériel, principalement avec les systèmes hydrauliques.
- La maintenance doit être structurée.

III.14 La régulation et les APIs [9]

La régulation (ou asservissement) consiste à agir de façon à ce que une mesure soit égale à une consigne. Si l'on cherche à atteindre une consigne (de position ou de température), on parlera de poursuite ou asservissement ; si l'on cherche à éliminer des perturbations pour qu'une valeur reste constante, on parlera de régulation. Les régulateurs n'étaient pas intégrés dans l'unité centrale des Automates Programmables Industriels, mais se présentaient sous forme de modules autonomes gérant leur environnement propre (acquisition, calcul, commande…).

De plus en plus, les automates intègrent les régulateurs au sein de l'unité centrale. Soit sous la forme de module autonome émulant un régulateur externe au sein de l'UC (évitant ainsi la redondance de câblage qu'impose l'utilisation de régulateur externe), soit sous la forme de blocs primitifs intégrables au sein du code au même titre qu'un bloc temporisateur.

III.14.1 Régulation PID [11]



Le progiciel « Régulation PID » se compose des éléments suivants :

- Les blocs fonctionnels.
- L'interface utilisateur de paramétrage pour la configuration des blocs de régulation.
- Le manuel consacré pour l'essentiel à la description des blocs fonctionnels.

Les FB de régulation proposent une régulation purement logicielle, c'est-à-dire qu'un bloc contient toutes les fonctions du régulateur. Les données nécessaires au calcul cyclique sont stockées dans des blocs de données associés, les blocs de données d'instance, ce qui permet aux FB de les appeler plusieurs fois.

La performance de régulation et donc la rapidité du traitement dépendant uniquement de la puissance de la CPU utilisée. Pour une CPU donnée, il faut trouver un compromis entre le nombre de régulateurs et la fréquence de traitement de chacun d'eux. Plus les boucles de régulation seront rapides, c'est-à-dire plus souvent les grandeurs réglantes sont à calculer par unité de temps, moins vous pourrez installer de régulateur. Il n'y a pas de restriction concernant le type de processus à régler. Les blocs fonctionnels s'appliquent aux systèmes inertiels (températures, niveaux de remplissages, etc.) comme aux systèmes très rapides (débit, vitesse de rotation, etc.).

Remarque 1 : Le comportement statique (gain) et les propriétés dynamiques (retard, temps mort, constante d'intégration, etc) du système régulé exercent une influence décisive sur la conception du régulateur et sur le dimensionnement de ses paramètres statiques (influence P) et dynamiques (influence I et D). Il est par conséquent indispensable de connaître exactement le type et les caractéristiques du système à réguler.

III.14.2 Paramétrage du régulateur PID

Les principaux paramètres du régulateur PID dont on aura besoin sont cités dans les deux tableaux suivants :

III.14.2.A Paramètres d'entrée

Le tableau_1 suivant représente quelques paramètres d'entrée de réglage du régulateur PID :

Paramètre	Type de données	Description
COM_RST	BOOL	**COMPLETE RESTART /** Démarrage Le bloc renferme un sous-programme de démarrage qui est exécuté quand cette entrée est à 1.
P_SEL	BOOL	**PROPORTIONAL ACTION ON /** Activation de l'action proportionnelle Dans l'algorithme PID, il est possible d'activer et de désactiver séparément chacune des actions. L'action P est active quand cette entrée est à 1.
I_SEL	BOOL	**INTEGRAL ACTION ON /** Activation de l'action par intégration Dans l'algorithme PID, il est possible d'activer et de désactiver séparément chacune des actions. L'action I est active quand cette entrée est à 1.
D_SEL	BOOL	**DERIVATIVE ACTION ON /** Activation de l'action par dérivation Dans l'algorithme PID, il est possible d'activer et de désactiver séparément chacune des actions. L'action D est active quand cette entrée est à 1.
CYCLE	TIME	**SAMPLE TIME /** Période d'échantillonnage Le temps s'écoulant entre les appels de bloc doit être constant. Il est indiqué au niveau de cette entrée.
SP_INT	REAL	**INTERNAL SETPOINT /** Consigne interne Cette entrée sert à introduire une valeur de consigne.
PV_PER	WORD	**PROCESS VARIABLE PERIPHERIE /** Mesure de périphérie La mesure en format de périphérie est appliquée au régulateur par cette entrée.
GAIN	REAL	**PROPORTIONAL GAIN /** Coefficient d'action proportionnelle Cette entrée indique le gain du régulateur.
TI	TIME	**RESET TIME /** Temps d'intégration Cette entrée détermine la réponse temporelle de l'intégrateur.
TD	TIME	**DERIVATIVE TIME /** Temps de dérivation Cette entrée détermine la réponse temporelle de l'unité de dérivation.
PV_FAC	REAL	**PROCESS VARIABLE FACTOR /** Facteur de mesure Cette entrée est multipliée par la mesure. Elle sert à adapter l'étendue de valeur de mesure.
PV_OFF	REAL	**PROCESS VARIABLE OFFSET /** Décalage de mesure Cette entrée est ajoutée à la mesure. Elle sert à adapter l'étendue de valeur de mesure.

Tableau III.1 Paramètres d'entrée

III.14.2.B Paramètres de sortie

Le tableau_2 montre quelques paramètres de sortie :

Paramètre	Type de données	Description
LMN_PER	WORD	**MANIPULATED VALUE PERIPHERY /** Valeur de réglage de périphérie Cette sortie fournit la valeur de réglage en format de périphérie.
PV	REAL	**PROCESS VARIABLE /** Mesure Cette sortie donne la mesure effective.
ER	REAL	**ERROR SIGNAL /** Signal d'erreur Cette sortie donne le signal d'erreur effectif.
LMN_P	REAL	**PROPORTIONALITY COMPONENT /** Composante P Cette sortie contient la composante proportionnelle de la grandeur de réglage.
LMN_I	REAL	**INTEGRAL COMPONENT /** Composante I Cette sortie contient la composante intégrale de la grandeur de réglage.
LMN_D	REAL	**DERIVATIVE COMPONENT /** Composante D Cette sortie contient la composante différentielle de la grandeur de réglage.
QLMN_HLM	BOOL	**HIGH LIMIT OF MANIPULATED VALUE REACHED /** Grandeur de réglage à la limite supérieure La valeur de réglage est toujours bornée à une limite supérieure et à une limite inférieure. Cette sortie signale le dépassement de la limite supérieure.
QLMN_LLM	BOOL	**LOW LIMIT OF MANIPULATED VALUE REACHED /** Grandeur de réglage à la limite inférieure La valeur de réglage est toujours bornée à une limite supérieure et à une limite inférieure. Cette sortie signale le dépassement de la limite inférieure.

Tableau III.2 Paramètres de sortie

III.14.3 Régulation de la température du four électrique par automate

Cet exemple est une application d'une commande classique en utilisant le bloc de régulation PID qui existe sur API. On choisit les paramètres des actions du régulateur à partir du système identifié (**II.1**) puis une configuration du bloc.

$$H_b(p) = \frac{1{,}27e^{-114p}}{(1072p+1)} \qquad (\mathbf{II.10})$$

La figure **III.6** montre le réglage des paramètres du PID sur l'automate.

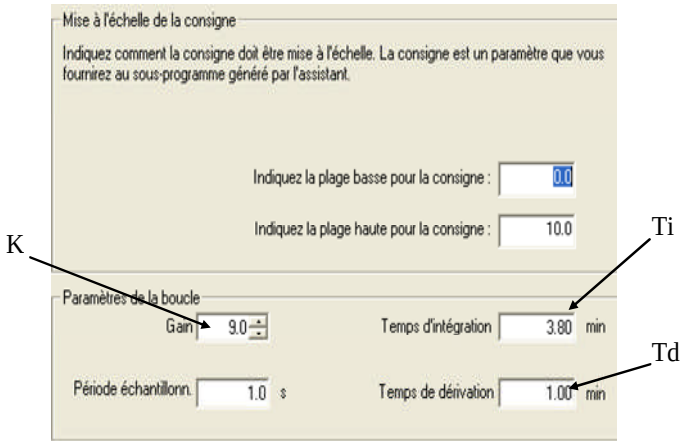


Figure III.6 Paramètres du PID

Le résultat obtenu est présenté dans la figure **II.7**.

La consigne est un échelon d'amplitude de $150℃$.

Le temps est représenté en échantillon, chaque échantillon est de $10s$.

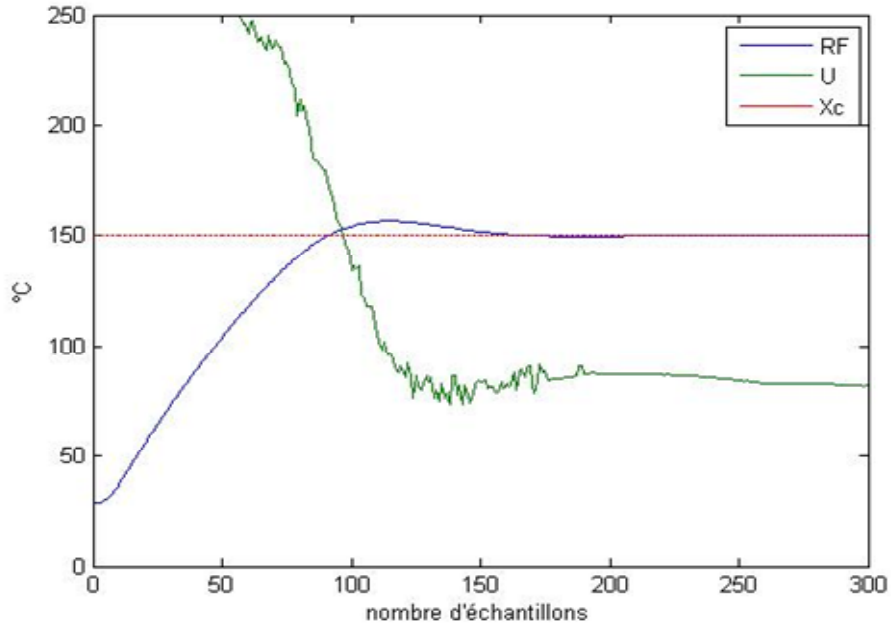


Figure III.7 Réponse du four régulé par automate

RF : Réponse du four

U: Commande du four

Xc : Consigne

On remarque que l'écart statique est nul, un dépassement de 4% avec un temps de réponse de 13.5 minute.

III.15 Conclusion

Dans ce chapitre nous avons étudié les techniques de programmation des automates programmables industriels APIs. Il est à signaler que les différentes opérations possible à effectuées (addition, soustraction, multiplication, …) et les différents blocs disponibles (conversion, inversion, comparaison, …) peuvent être les outils de base pour une implémentation d'un régulateur sur un API.

Bien que, l'implémentation d'une boucle de régulation classique (PID classique) sur un automate programmable industriel (API) consiste à configurer un bloc régulateur PID. L'implémentation d'une boucle de régulation floue implique l'utilisation des composants de base. La méthodologie d'implémentation sera développée dans le prochain chapitre.

Chapitre IV

Implémentation d'un régulateur à base de logique floue sur API

IV.1 Introduction

La logique floue permet d'obtenir une loi de commande souvent efficace, sans devoir faire appel à des développements théoriques importants. D'après les applications effectuées, les contrôleurs flous ont surtout démontré des performances plus robustes, par rapport aux autres techniques de commandes traditionnelles.

Il existe, certes, des logiciels professionnels pour la mise en œuvre des systèmes flous, par exemple Matlab, mais rien qui ne préconise une approche pédagogique.

Dans ce chapitre nous développons s la méthodologie de l'implémentation d'un contrôleur à base de la logique floue dans un contexte d'automate programmable industriel. Nous visons à présentons les techniques d'application issues de cette nouvelle technologie pour la conduite des procédés.

IV.2 Etablissement de la table des règles

Les opérations nécessaires pour effectuer des inférences floues sont initialement conclues à partir des caractéristiques du système (le four électrique) comme l'évolution de la sortie, la rapidité et la réponse aux différentes commandes. Ces inférences floues permettent de prendre les décisions nécessaires répondant aux objectifs visés par l'expert (atteindre une référence en un temps réduit avec un écart minimal).

Les étapes que nous avons suivies pour l'obtention de la table de règles sont :

a) Première étape

Cette étape consiste à choisir une consigne. L'expérience acquise précédemment sur la manipulation du système (four) a permis de recueillir quelques informations sur l'évolution de la réponse de ce dernier et l'influence de quelques commandes. Pour cet effet On a considéré une même consigne qui est de 150 °C (6Volts).

b) Deuxième étape

Les deux entrées (prémisses) sur lesquelles le traitement flou s'effectuera sont l'erreur et sa dérivée, on définit les fonctions d'appartenances pour chaque entrée comme suit :

- Erreur : si la sortie est inférieur à la consigne on dit que l'erreur est positive et dans le cas contraire négative. On définit le degré du signe (grand négative, grand positive,…) et on élabore huit fonctions d'appartenances selon les intervalles suivants :

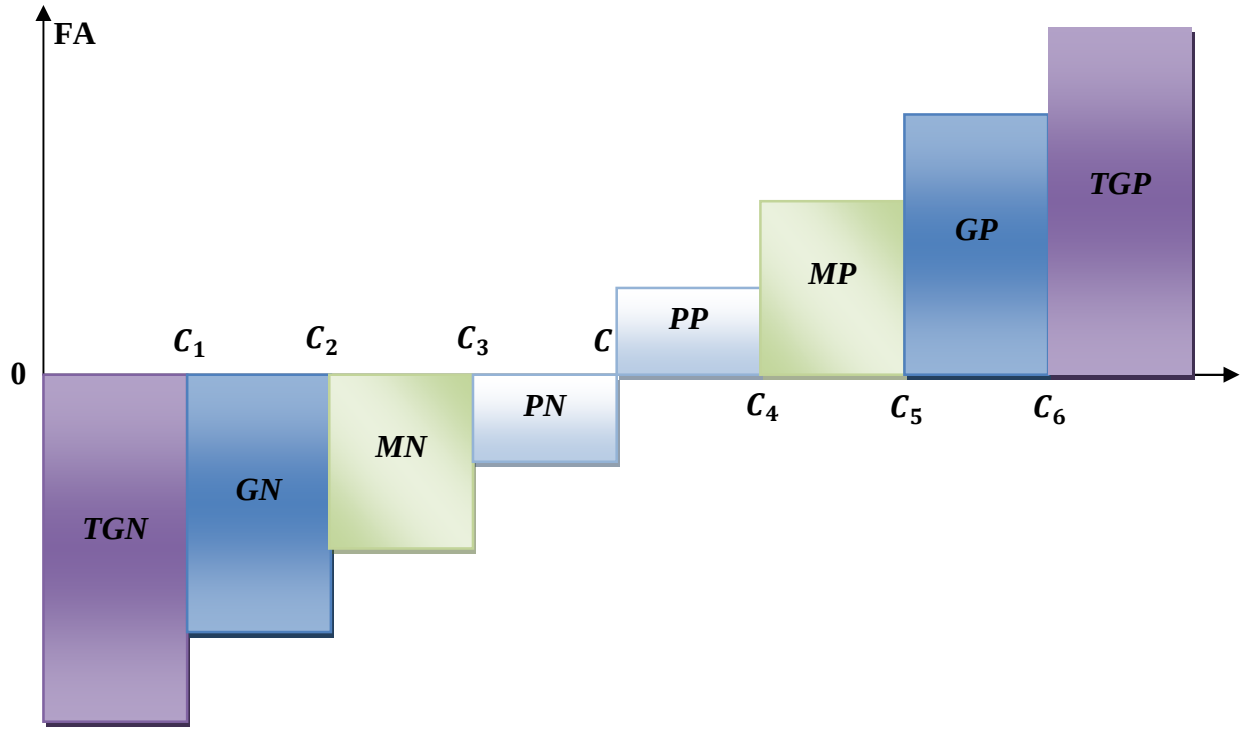


Figure IV.1 Fonctions d'appartenance de l'erreur

Les limites des intervalles choisis sont :

$\boldsymbol{C_1 = 5\ volt}$ $\boldsymbol{C_4 = 6.2\ volt}$

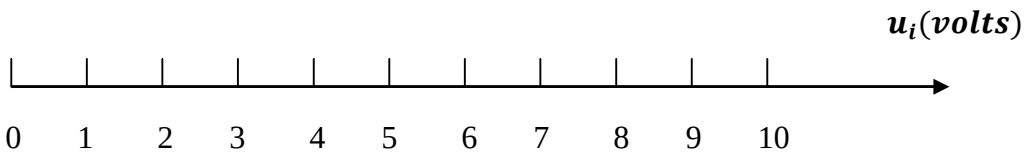
$\boldsymbol{C_2 = 5.5\ volt}$ $\boldsymbol{C_5 = 6.5\ volt}$

$\boldsymbol{C_3 = 5.8\ volt}$ $\boldsymbol{C_6 = 7\ volt}$

$\boldsymbol{C = 6\ volt}$ (Consigne)

- Dérivée de l'erreur **:** si la variation de la température évolue dans le sens croissant alors on dit que la dérivée de l'erreur est positive, par contre dans le cas où la variation est dans le sens décroissant, on dit que la dérivée de l'erreur est négative.

- la sortie (conclusion) :c'est des singletons (constantes)



Les valeurs de commandes à utiliser

$u_1 = 1\ volt$ $u_7 = 4.5\ volt$

$u_2 = 1.5\ volt$ $u_8 = 7\ volt$

$u_3 = 2\ volt$ $u_9 = 8\ volt$

$u_4 = 2.5\ volt$ $u_{10} = 8.8\ volt$

$u_5 = 3\ volt$ $u_{11} = 9.8\ volt$

c) Troisième étape

Une fois les fonctions d'appartenance convenablement ajustées et à partir des informations précédentes, on construit la table de règles présentée dans le tableau **IV.1**

de \ e	*TGN*	*GN*	*MN*	*PN*	*PP*	*MP*	*GP*	*TGP*
N	u_{11}	u_{10}	u_9	u_7	u_6	u_5	u_3	u_1
P	u_{11}	u_9	u_8	u_6	u_5	u_4	u_2	u_1

Tableau IV.1 Table de règles

Cette table se lit comme suit :

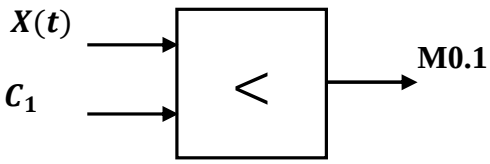
Si e est *TGN* et de est *N* alors $u_i = u_{11}$.

IV.3 L'implémentation

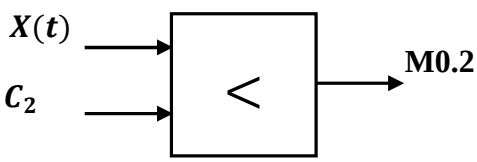
L'implémentation de cette table sur l'automate se fait par l'utilisation des blocs comme les comparateurs, les temporisateurs…etc.

Pour avoir les fonctions d'appartenance de l'erreur on utilise un comparateur entre la sortie et les différentes limites des intervalles, si la comparaison est vraie alors une sortie est activée indiquant l'appartenance de l'erreur.

Exemple de comparateur :



Comparateur

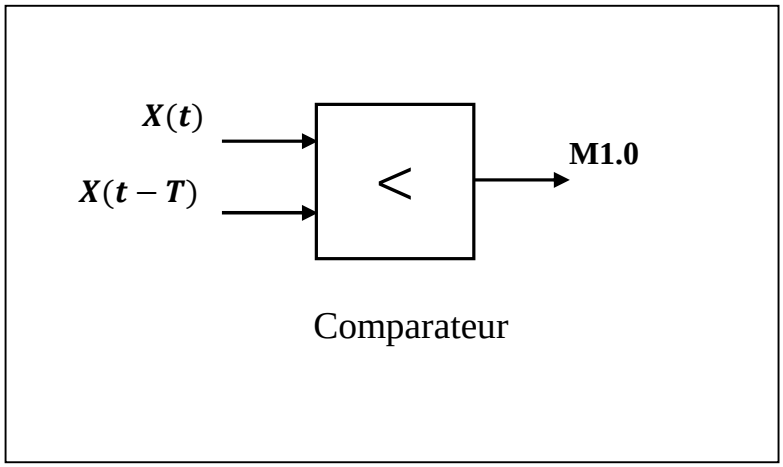


Comparateur

Si **MO.1** est mis à **1** alors on dit que l $\boldsymbol{e}$ est ***TGN***

Si **M0.2** est mis à **1** et **M0.1** est mis a **0** alors $\boldsymbol{e}$ est ***GN***

Pour avoir les fonctions d'appartenance de la dérivée de l'erreur on utilise un comparateur entre $\boldsymbol{X}$**(t)** et $\boldsymbol{X(t-T)}$ tel que $\boldsymbol{T = 15}$ secondes.

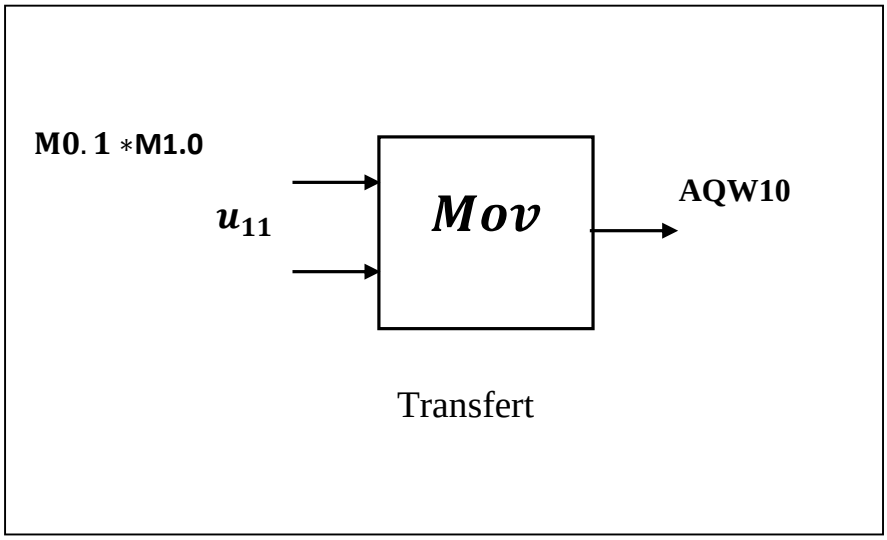


Si **M1.0** est mis à **1** alors on dit que la $\boldsymbol{de}$ est **N**

Si **M1.0** est mis à **0** alors on dit que la $\boldsymbol{de}$ est ***P***

Les commandes vont être sélectionnées et transférées vers une sortie de l'automate de la façon suivante :

Si **MO.1** est à **1** et **M1.0** est mis à **1** alors charger $\boldsymbol{u_{11}}$ dans **AQW 10**



IV.4 Résultat de la simulation

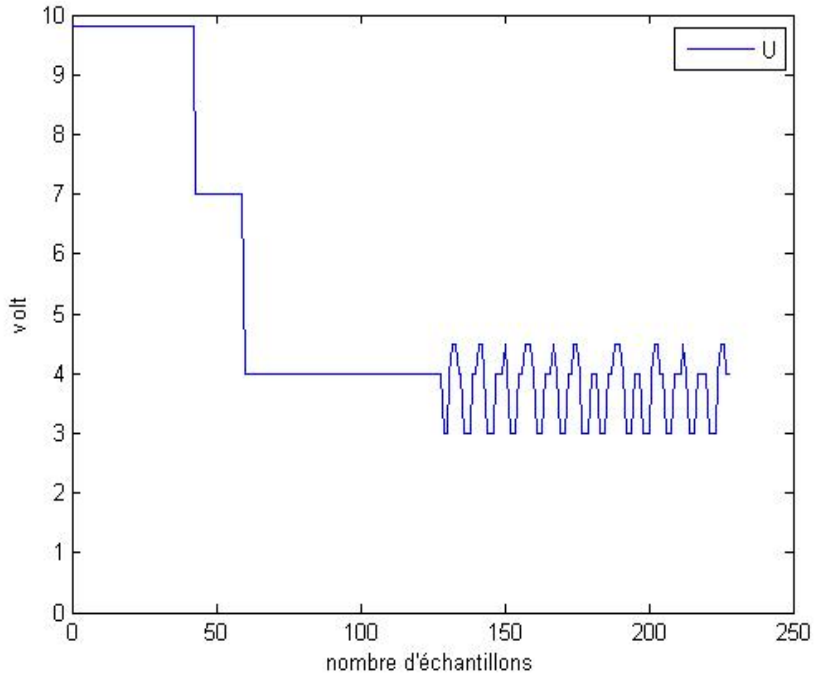


Figure IV.2 Signal de commande appliqué au four

One remarque une variation très importante de la commande en régime transitoire, par contre elle se stabilise autour de la valeur de 4 Volts en régime permanant. Les petites variations sont dues à la sensibilité de cette commande aux petites variations de la sortie du système.

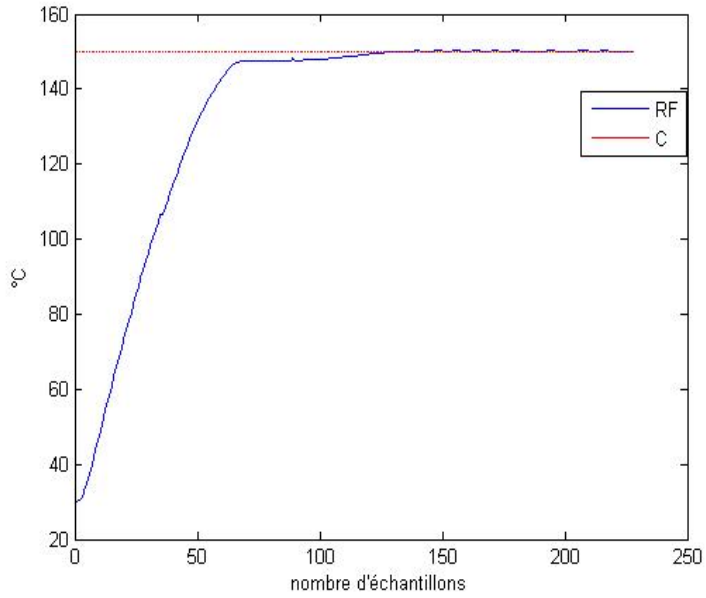


Figure IV.3 Contrôle flou de la température dans le four

Remarque :

Chaque intervalle entre deux échantillons, sur la figure, correspond à un temps de 15 secondes.

La courbe obtenue après l'implémentation de la commande floue (table des règles) dans l'automate montre que le contrôleur flou implémenté a permet de répondue à l'objectif visé, à savoir le contrôle de la température dans le four électrique. Les résultats obtenus sont satisfaisant puisque l'écart statique nul avec un temps de réponse de 14.5 minute.

IV.5 Conclusion

Dans ce chapitre, nous avons présentés une initiation à l'application de la logique floue sur l'automate S7 200, on a fait une interprétation des concepts de la logique floue en langage programmable (LADDER) en utilisant des outils de programmation fourni dans la bibliothèque du logiciel de programmation des APIs.

Un programme traduisant ce raisonnement humain a été implémenté pour le control de la température d'un four ce qui nous a permis d'obtenir des résultats satisfaisants avec des performances considérablement bonnes.

Conclusion Générale

Le but de ce projet était d'élaborer une technique d'implémentation et d'application de la logique floue dans un contexte d'automate programmable. Pour atteindre cet objectif, nous avons choisi le control de température d'un four électrique par API S7200. Ce choix est justifié par le fait que ce matériel existe au sein du laboratoire d'automatique industrielle.

Notre démarche nous a obligés à faire une identification du modèle (fonction de transfert) d'un four électrique a fin de faire des études théorique et la simulation du modèle obtenu à coïncider avec les résultats d'identification obtenus.

La conception du contrôleur flou sur un API S7 200 a permit de réaliser une application pour la régulation de la température du four électrique et les résultats obtenus été satisfaisants.

Nous pensons que l'optimisation du contrôleur flou conçu permettra d'obtenir des résultats intéressants ce qui donne à l'implémentation des contrôleurs flous sur les APIs une importance majeur.

Notre travail à démontrer que la logique floue peut être implémenté dans les automates, toutefois un bloc fonctionnel traitant cette technologie reste à développer et amélioré sur les APIs afin de donner une autre dimension à l'utilisation de la logique floue.

Références bibliographiques

[1] FAUCHER, J., « thèse : les plans d'expériences pour le réglage de commande à base de logique floue », Institut National Polytechnique de Toulouse, Septembre 2006.

[2] Manuel, « contrôle processus avec API [PLC] ».
Electronica Vaneta & NEL, ITALY.

[3] FERGUSON T., « Tutorial : Mesurer des températures par thermocouples ».
National Instruments, Note d'application 043, Février 2001.

[4] PROUVOST.P., « contrôle régulation ».
Nathan Technique, Paris, 1997.

[5] FLAUS.J.M., « La régulation industrielle : régulation PID, prédictifs et flous »
Hermes Science, Paris, 1994, 2000.

[6] ZHIQIANG, G. et TRAUTZCH A., « A Stable Self-Tuning Fuzzy Logic Control System for Industrial Temperature Regulation »
Department of Electrical Engineering**,** Cleveland State University.

[7] GONZAGA, A., « Les automates programmable industriels : Cours ».

[8] MAIDI, A., « cours d'automatismes industriels »
UMMTO, 2005,2006.

[9] VITTE, E., « Conseils en automatisme, technique et innovation (PEC) **»**
Schneider Electric, Inter Sections, juin2004.

[10] GHARIEB, W., « Software Quality in Ladder Programming**»**
Computer and Systems Engineering Department, Faculty of Engineering Ain Shams University, Cairo-EGYPT.

[11] Manuel « Logiciel de base pour S7_300/400, régulation PID »
Siemens, Simatic, 2001.

[12] James DAWSON, **«**Fuzzy Logic Control of Linear Systems with Variable Time Delay»
M.S. Thesis, Cleveland State University, Juin 1994.

[13] Thomas A. TRAUZTCH, «Self-Tuning Temperature Control Using Fuzzy Logic»
M.S. Thesis, Department of Electrical Engineering, Cleveland State University, Juin 1996.

[14] DAVID J. Elliott, «Fuzzy Logic Positional Servo Motor Control DevelopmentPlatform»
M.S. Thesis, Department of Electrical Engineering, Cleveland State University, Juin 1997.

[15] WILFRED Nonnenmacher, «Fuzzy Logic Position Control of a Servo Motor»,
M.S. Thesis, Department of Electrical Engineering, Cleveland State University, Avril 1997.

[16] NITIN, Dhayagude, ZHIQIANG Gao et Fouad MRAD, **«**Fuzzy Logic Control of Automated Screw Fastening», *Journal Robotics and Computer Aided Manufacturing*, Vol. 12, No.3 pp. 235-242, 1996

[17] K. PASSINO et S. YURKOVICH, «Fuzzy Control»
Addison-Wesley, 1998.