REPUPLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Faculté de Technologie

Département de Génie Electrique

Cours

Commande Intelligente

Spécialité :

Master 2 Automatique

Master 2 Commande des Systèmes Electriques

Présenté Par L'enseignant

Dr Samir ZEGHLACHE

CHAPITRE I : COMMANDE PAR LOGIQUE FLOU

CHAPITREII : COMMANDE ET IDENTIFICATION PAR LES RESEAUX DE NEURONE ARTIFICIELS

CHAPITRE III : COMMANDE PAR LES RESEAUX DE NEURONE FLOUE

CHAPITRE IV : OPTIMISATION DE COMMANDES FLOUE PAR LES ALGORITHMES GENETIQUES

Chapitre I

Commande par Logique Flou

I.1. METHODOLOGIE DE LA LOGIQUE FLOUE

I.1.1. Introduction

La logique floue suscite actuellement un intérêt général de la part des chercheurs, des ingénieurs et des industriels, et plus généralement de la part de tous ceux qui éprouvent le besoin de formaliser des méthodes empiriques, de généraliser des modes de raisonnement naturel, d'automatiser la prise de décision dans leur domaine et de construire des systèmes artificiels effectuant les tâches habituellement prises en charge par les humains.

La logique floue est une technique pour le traitement de connaissances imprécises basées, sur des termes linguistiques; elle donne les moyens de convertir une commande linguistique basée sur le raisonnement humain, en une commande automatique, permettant ainsi la commande des systèmes complexes dont les informations sont exprimées d'une façon vague et mal définie.

Dans le domaine du génie électrique, la commande à logique floue a fait l'objet de plusieurs travaux : dans la commande des convertisseurs statiques et dans la commande des machines électriques [1], [2], [3], dans la navigation de robots mobiles [4], [5]. Toutes ces applications ont démontré qu'un régulateur à logique floue est plus robuste qu'un régulateur conventionnel [2], [6].

Les performances que la commande floue peut apporter par comparaison avec les commandes classiques, sont essentiellement dues à la méthode de conception de ces régulateurs. En effet, ces derniers ne nécessitent pas la connaissance des modèles mathématiques du système. Par contre ils ont besoin d'un ensemble de règles basées essentiellement sur les connaissances d'un opérateur qualifié manipulant le système.

Afin de pouvoir appliquer la technique de la logique floue à la commande d'une machine électrique en vitesse et position et à la navigation d'un robot mobile dans un environnement inconnu, nous allons nous intéresser de plus près à cette technique. Dans ce contexte, on se limitera aux propriétés essentielles de la commande par logique floue qui n'utilise qu'une petite partie de toutes les règles existantes de la théorie de la logique floue.

I.1.2. Principe de la logique floue

I.1.3. Principe

Une des caractéristiques du raisonnement humain est qu'il est généralement fondé sur des données imprécises ou même incomplètes. En effet les connaissances dont nous disposons sur un système quelconque sont généralement incertaines ou vagues, soit parce que nous avons un doute sur leur validité ou alors nous éprouvons une difficulté à les exprimer clairement.

- Premier exemple : Soit E l'ensemble des tailles possible et A l'ensemble des grandes tailles. En logique boolienne, on est soit de petite taille, soit de grande taille. Si 170 cm est la frontière entre les deux, on est petit pour x<170 cm et grand pour x>=170. Mais si l'on mesure 169 cm, on est petit !!!...cette discontinuité est totalement absurde (Figure I.1.a). Le plus correcte sera donc de représenter A par un ensemble flou permettant un passage progressif des tailles petites aux tailles grandes (Figure I.1.b).

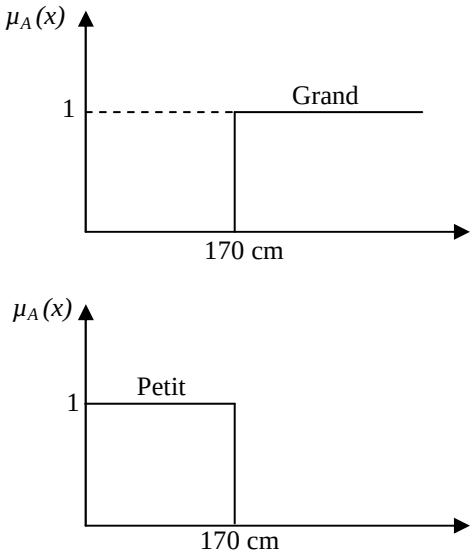


Figure I.1.a : Représentation les 2 variables (grand et petit) par la logique boolienne

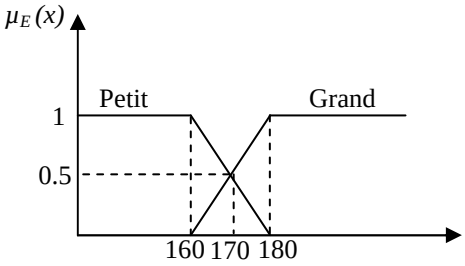


Figure I.1.b : Représentation les 2 variables (grand et petit) par la logique flou

• Deuxième exemple : Une température de 16°C dans la logique classique correspond au seul ensemble "Tiède", alors que dans le modèle flou, elle appartient à la fois aux ensembles "Froid" et "Tiède".

Il est donc nécessaire de penser et de développer un nouveau type de raisonnement : le raisonnement approché, qui permettra de traiter mathématiquement l'imprécis et l'incertain. Le premier à avoir souligné ces possibilités de développement est Lotfi A. Zadeh qui dés 1965 introduit la théorie de la logique floue [7], [8].

C'est une technique pour le traitement de connaissances imprécises et incertaines. Elle permet de prendre en considération des variables linguistiques dont les valeurs sont des mots ou des expressions du langage naturel, telles que rapide, lent, grand, petit, etc....

Un exemple : dans la logique classique, une vitesse peut être qualifiée par les termes « Elevée ». Dans la logique floue, des échelons d'appréciation intermédiaires de la variable vitesse sont possibles. La «Vitesse» devient une variable linguistique dont les valeurs sont par exemple : « Très faible », « Faible », « Moyenne », «Elevée », « Très élevée ».

La logique floue peut être considérée comme une extension de la logique classique ou binaire.

I.2. Les bases théoriques de la logique floue

L'exemple suivant montre où nous voulons en venir avec cette théorie. Nous voulons réaliser une régulation de température sur la base de règles telles que :

1- **Si** la température est basse, **Alors** le chauffage doit être fort,
2- **Si** la température est moyenne **Et** que la pression est haute, **Alors** le chauffage doit être moyen **Et** le volet doit être ouver.
3- **Si** la température est trop élevée **Ou** la pression trop forte, **Alors** le chauffage doit être arrêté **Et** le volet ouvert.

4- **Si** la température est trop élevée **Et** que la pression n'est pas trop haute, **Alors** le chauffage doit être arrêté **Et** le volet en position médiane.
5- et ainsi de suite.. .

On décrit ainsi des situations à l'aide de combinaisons de l'appartenance des valeurs mesurées à un ensemble déterminé de valeurs d'entrées. Chaque situation demande une action déterminée, pour cela on fait appel à un certain nombre d'opération sur les ensembles, que nous allons examiner dans la suite.

Les éléments de base de la logique floue sont les suivants [8]:

* les ensembles flous (*fuzzy sets*) pour la représentation de variables linguistiques ;
* les fonctions d'appartenance (*memberships functions*) qui décrivent le degré d'appartenance de grandeurs physiques (vitesse, courant, température) à un ensemble flou (faible, élevé, chaud) ;
* les opérateurs flous qui permettent l'énonciation de relations logiques entre les assertions floues (conclusion du genre « Si, Alors »);
- l'inférence floue c'est à dire la déduction de nouvelles informations déjà disponibles sur la base des règles linguistiques.

Le premier concept important à expliquer est celui de variable linguistique et de sous-ensemble floue [7].

I.2.1. Variables linguistiques, fonctions d'appartenance

La notion de variable linguistique permet de modéliser les connaissances imprécises ou vagues sur une variable dont la valeur précise est inconnue. Une variable linguistique, ou variable floue, est donc une variable dont les valeurs appartiennent à des ensembles flous pouvant représenter des mots du langage naturel. Ainsi une variable floue peut prendre simultanément plusieurs valeurs linguistiques.

Par exemple la variable « Taille » peut appartenir aux ensembles floues " Petit, Moyen, Grand ".

La variable linguistique peut être représentée par un triplet (x, T(x), U), dans lequel x est le nom de la variable linguistique, T(x) l'ensemble des noms des valeurs linguistiques de x et U l'ensemble de référence (univers de discours).

Par exemple : x = Vitesse est une variable linguistique, son ensemble de valeurs peut être :
T(Vitesse) = [Faible, Moyenne, Elevée,....] où chaque terme dans T(Vitesse) est caractérisé par un ensemble flou dans un univers de discours U = [0, 100] (figure. I-1).

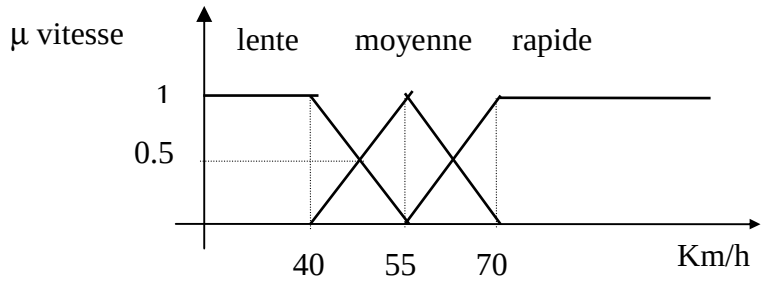
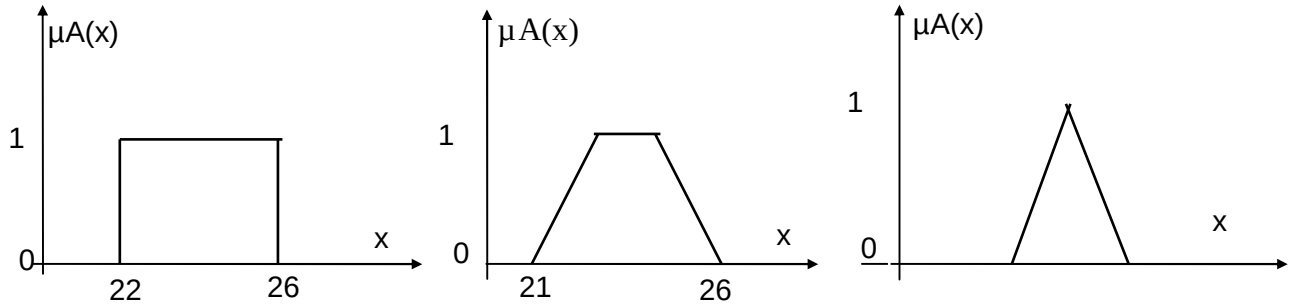


Figure I- 2 Représentation floue de la variable Vitesse

On attribue à chaque valeur de la variable linguistique des fonctions d'appartenance μ, dont la valeur varie entre 0 et 1, en tenant compte de la classification en un certain nombre d'ensembles flous.

Le plus souvent, on utilise pour les fonctions d'appartenance de forme trapézoïdales ou triangulaires, rectangulaires ou de type singleton. Il s'agit des formes les plus simples (figure I-2).



Ensemble net Intervalle flou Nombre flou

Figure I-3 Exemples de fonctions d'appartenance

I.2.2. Univers de discours, ensemble flou, degré d'appartenance

En logique classique, une variable peut prendre deux valeurs possibles : vrai (1) ou faux (0).

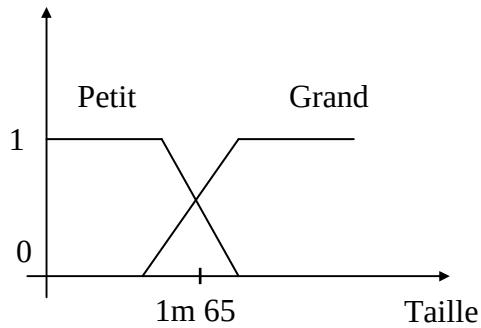
En logique floue, un ensemble flou A d'un univers de discours (ensemble de toutes les valeurs possibles de x) X est défini par une fonction d'appartenance μ_A qui à tout élément x appartenant à X, associe un nombre $\mu_A(x)$ compris entre 0 et 1, qui représente le degré ou facteur d'appartenance de x à l'ensemble flou A

(0 représentant la non - appartenance et 1 l'appartenance totale).

Le concept d'ensemble flou a donc pour objectif de permettre des gradations progressives dans l'appartenance d'un élément à une classe.

L'appellation d'un ensemble flou est d'ordinaire en relation avec la signification que l'on souhaite lui donner et ce mot correspond à la valeur de la variable linguistique x appartenant à X.

Prenons l'exemple de la taille d'une personne classée en deux ensembles flous « Petite » et « Grande » , chaque personne a une taille qui se situe entre les deux (figure I-3).



Si je mesure 1m 65, j'appartiens à la fois à l'ensemble « Petite » et à l'ensemble « Grande » avec des degrés d'appartenance respectifs de 0.5 - Si je mesure 1m 40, je n'appartiens qu'à l'ensemble « Petite » avec un degré 1, - Si je mesure 1m 80, je n'appartiens qu'à l'ensemble « Grande » avec un degré 1.

Figure I-4 Représentation floue de la variable Taille

I.3. Déductions floues (inférences)

Afin de tirer des conclusions, plusieurs valeurs de variables linguistiques sont liées entre elles par des règles. On parle alors de déductions floues ou inférences.

Les règles peuvent alors être exprimées sous la forme générale:

Opération: = **Si** Condition 1 **Et** Condition 1', **Alors** Conséquence 1, ou
Si Condition 2 **E**t Condition 2', **Alors** Conséquence 2, ou
. . .
Si Condition m **Et** Condition m', **Alors** Conséquence m.

Si les conditions, qui sont exprimées par la condition (prémisse), sont vraies, alors l'action spécifiée à la conséquence (conclusion), aura lieu.

Les inférences avec plusieurs règles sont caractérisées par le fait qu'en général plusieurs règles sont (plus ou moins) simultanément vérifiées. L'opération qui doit être effectuée doit tenir compte des différentes conditions et s'obtient par les règles de calcul de la logique floue.

I.4. Opérations en logique floue

Les variables linguistiques sont liées entre elles au niveau des inférences par des opérateurs **Et** ou **Ou**. Il s'agit d'opérateurs de la logique floue qui interviennent sur les fonctions d'appartenance représentant les variables linguistiques.
Les opérateurs les plus importants sont : **l'intersection , l'union , et le complément** .

Nous allons généraliser ces fonctions fondamentales de la théorie des ensembles. On devra retrouver dans le cas classique les fonctions habituelles. Les solutions proposées devront répondre à certaines propriétés (croissance, associativité,...) spécifiques aux fonctions considérées.

I.4.1. Complément d'un ensemble (fonction négation)

a) Critères

Une fonction négation n(x) doit vérifier les conditions suivantes:

- $n(x) : [0, 1] \rightarrow [0, 1]$
- $n(x) \downarrow$ si $x \uparrow$ (croissance)
- $n(1) = 0$ et $n(0) = 1$ (conditions aux limites)

b) Choix

Il existe plusieurs fonctions répondant aux critères énoncés. On a choisi d'adopter la fonction (figure.I-5) :

$$\text{Non } (\mu_A(x) = 1-\mu_A(x)$$

c) Illustration

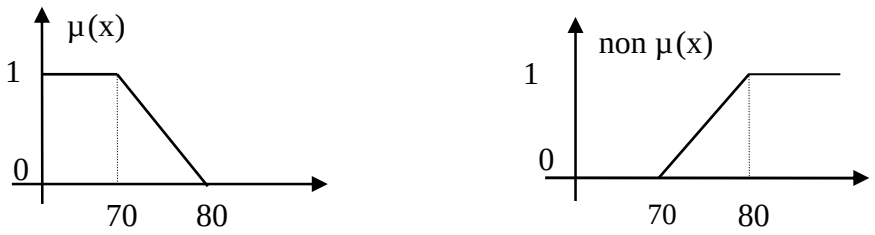


Figure I-5 Fonction d'appartenance de l'opérateur Non

I.4.2. Intersection de deux ensembles (fonction "Et")

a) Critères

Une fonction "Et" associée à deux ensembles d'univers de discours x et y doit remplir les conditions suivantes:

- *Et* (x, y): [0, 1]×[0, 1]→[0, 1]
- *Et* (x, y) = *Et* (y, x) (commutativité)
- si x < y *Et* z < t : *Et* (x, z) < *Et* (y, t) (croissance)
- *Et* (x, *Et* (y, z)) = *Et* (*Et* (x, y), z) (associativité)
- *Et* (0, x) = 0, *Et* (1, x) = x (conditions aux limites)

b) Choix

De nombreuses possibilités existent (voir tableau récapitulatif). La première proposition, due à Zadeh, est encore aujourd'hui souvent utilisée: la fonction minimum. Son interprétation est simple: un élément ne peut pas appartenir plus à l'intersection de deux ensembles qu'à chacun de ceux-ci. Cette fonction est illustrée ci-après (figure I-6).

c) Illustration

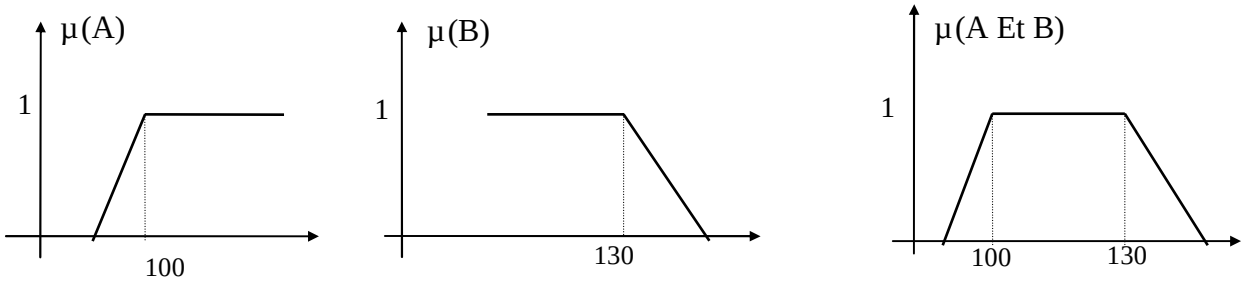


Figure I-6 Fonction d'appartenance de l'opérateur Et

I.4.3. Union de deux ensembles (fonction "Ou")

a) Critères

La fonction doit être telle que:

∗ *Ou* (x, y): [0, 1]×[0, 1]→[0, 1]
∗ *Ou* (x, y) = *Ou* (y, x) (commutativité)
∗ Si x < y *Ou* z < t : *Ou* (x, z) < *Ou* (y, t) (croissance)
∗ *Ou* (x, *Ou* (y, z)) = *Ou* (*Ou* (x, y), z) (associativité)
∗ *Ou* (0, x) = x *Ou* (1, x) = 1 (conditions aux limites)

b) Choix

Plusieurs choix sont également possibles, dont les plus courants sont repris dans le tableau illustratif ci-après. C'est généralement la fonction maximum qu'on sélectionne. Elle est illustrée ci-dessous (figure I-7).

c) Illustration

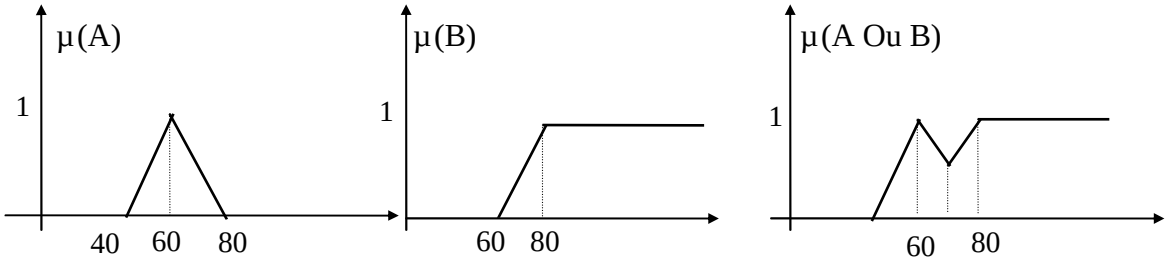


Figure I- 7 Fonction d'appartenance de l'opérateur Ou

I.4.4. Tableau récapitulatif

Appellation	ET	OU	NON
Zadeh	min(x , y)	max(x , y)	1 - x
Probalistique	x y	x + y - x y	1 - x
Lukasiewicz	max (x + y-1, 0)	min(x + y, 1)	1 - x
Hanacher (β>0)	x y/(β+(1- β)(x + y - x y))	(x + y +x y - (1-β) x y) /(1- (1 - β) x y)	1 - x
Weber	x si y = 1, y si x = 1 0 sinon	x si y = 0, y si x = 0 1 sinon	1 - x

I.4.5. Opérateurs Et, Ou réalisés par opérateurs arithmétiques

Les réalisations précédentes donnent dans la plupart des cas des résultats convenables. Cependant pour le réglage et la commande, il peut être judicieux d'utiliser d'autres opérateurs dans le but de les exécuter par microprocesseur [7].

• L'opérateur Et dans ce cas est réalisé par la fonction du produit des fonctions d'appartenances:

$$\mu c(x) = \mu a(x) * \mu b(x) \qquad \text{(I-1)}$$

• L'opérateur Ou est réalisé par la formation de la somme (valeur moyenne) des fonctions d'appartenances:

$$\mu c(x) = [\, \mu a(x) + \mu b(x) \,] / 2 \qquad \text{(I-2)}$$

I.5. Le raisonnement flou

I.5.1 Généralités

Après avoir exposé la répartition des valeurs mesurées en ensembles flous et défini les opérations sur ces ensembles, nous allons maintenant introduire le raisonnement flou et voir comment un régulateur peut être exécuté sur la base des règles logiques.

L'exemple précédent de la commande de la température nous a donné un aperçu du concept du raisonnement utilisé : **Si** les conditions sont remplies, **Alors** la conclusion est validée.

Avec cet unique schéma de raisonnement et les trois opérateurs **Et** , **Ou** et **Non** , nous pouvons déjà prendre un grand nombre de décisions logiques Nous produisons aussi une nouvelle information (une décision) à partir d'informations anciennes.

Le raisonnement flou fait appel à trois notions et étapes fondamentales [6]:

- *l'implication floue,*
- *l'inférence floue,*
- *l'agrégation des règles.*

I.5.2. L'implication floue

L'implication floue donne une information sur le degré de vérité d'une règle floue [7]. En d'autres termes, on quantifie la force de véracité entre la prémisse et la conclusion. Considérons par exemple les deux propositions floues

″ x est A ″
″ y est B ″ Où x et y sont des variables floues et A et B des ensembles flous de l'univers du discours U.

ainsi que la règle floue : **Si** ″ x est A ″ **Alors** ″ y est B ″ .

L'implication floue donne alors le degré de vérité de la règle floue précédente à partir des degrés d'appartenance de x à A (prémisse) et de y à B (conclusion). Il n'existe pas une façon unique de définir l'implication floue.

On notera implication : opérateur *imp* (équivalent à l'opérateur Alors).
Parmi toutes les normes d'implication qui existent celles qui sont les plus utilisées sont :

- La norme Mamdani : $imp\ (\mu_A(x), \mu_B(y)) = \min(\mu_A(x), \mu_B(y))$ (I-3)
- La norme Larsen : $imp\ (\mu_A(x), \mu_B(y)) = (\mu_A(x) \,.\, \mu_B(y))$ (I-4)

I.5.3. L'inférence floue

Le problème tel qu'il se pose en pratique n'est généralement pas de mesurer le degré de véracité d'une implication mais bien de déduire, à l'aide de faits et de diverses règles implicatives, des événements potentiels. En logique classique, un tel raisonnement porte le nom de Modus Ponens (raisonnement par l'affirmation).

Si p $\Rightarrow$ q vrai
Et p vrai
Alors q vrai

Il nous faut le généraliser. Vont intervenir dans cette étape la « fonction d'implication » de l'implication floue, image du lien **Et** les fonctions d'appartenance des sous-ensembles des prémisse(s) et conclusion(s), images de l'importance des événements.

De façon générale, les conditions d'utilisation du Modus Ponens Généralisé sont les suivantes :

	prémisse	***conclusion***
Règle floue :	**Si** x est A	**Alors** y est B
Fait observé:	**Si** x est A'	
---	---	---
Conséquence :		y est B'

A' et B' sont les ensembles flous constatés dans le cas que l'on traite et ne sont pas nécessairement strictement égaux à A et B. B' est l'ensemble flou résultant de A' par l'application de l'implication.

Les informations disponibles pour déterminer la conséquence sont donc d'une part celles relatives à la règle, quantifiées par l'implication floue $\mu_{B/A}(x, y)$, d'autre part celles relatives au fait observé, quantifiées par la fonction d'appartenance $\mu_{A'}$.

I.5.4. Agrégation des règles

Lorsque la base de connaissance comporte plusieurs règles (comme notre exemple de la régulation de température), l'ensemble flou inféré B' est obtenu après une opération appelée agrégation des règles. En d'autres termes l'agrégation des règles utilise la contribution de toutes les règles activées pour en déduire une action de commande floue. Généralement, les règles sont activées en parallèle et sont liées par l'opérateur **Ou** [7].

Nous pouvons considérer que chaque règle donne un avis sur la valeur à attribuer au signal de commande, le poids de chaque avis dépend du degré de vérité de la conclusion.

Dans l'exemple suivant :
Si ″x est A″ **Et** ″y est B″ **Alors** ″z est C″ **Ou**
Si ″x est A'″ **Et** ″y est B'″ **Alors** ″z est C'″ **Ou** .....

L'ensemble flou résultant est obtenu en prenant, pour chaque valeur de sortie z, la valeur maximale des degrés d'appartenance de chaque contribution.(figure I-7).

Dans cet exemple, l'opérateur **«Et »** est représenté par l'opérateur « **min »,** l'implication est représenté par l'opérateur ''**min**''et l'opérateur **«Ou »** est représenté par l'opérateur « **max »** :

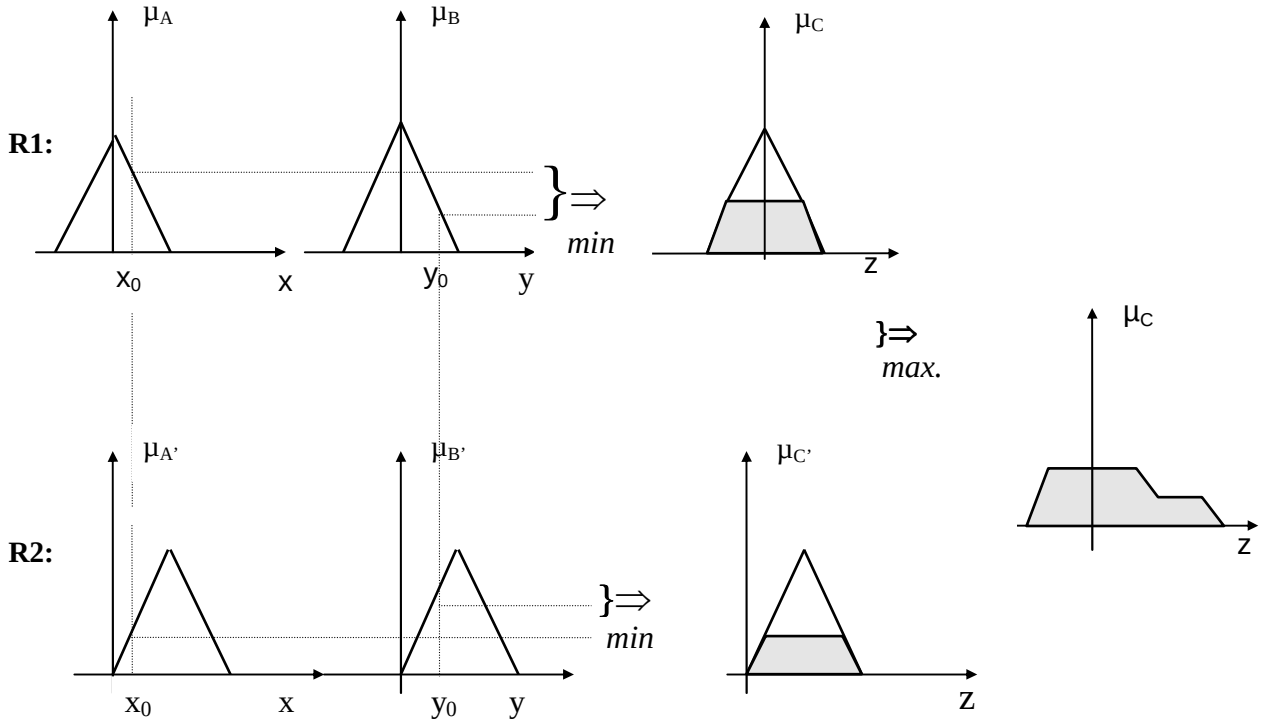


Figure I-8 Principe de l'agrégation de deux règles

I.6. Conclusion

Dans ce chapitre, nous avons vu ce qu'était la logique floue, quels étaient ses intérêts et posé ses bases mathématiques. Le principes de la logique floue, l'utilisation du concept d'ensemble flou peuvent être appliqués à beaucoup de problèmes où, selon la nature de l'information, la manipulation de l'imprécis ou vague est indispensable: (classification, décision multicritère, base de données, commande).

Nous sommes maintenant armés pour voir comment un contrôle de processus peut être réalisé en logique floue. Ce sera l'objet du chapitre suivant.

I.7. PRINCIPE D'UN CONTROLEUR FLOU

I.7.1. Introduction

Dans ce chapitre, nous nous intéressons à l'application de la logique floue dans les systèmes de commande et aux méthodes de conception des contrôleurs flous. Nous appliquons la logique floue à deux domaines précis, celui du contrôle d'une machine électrique et d'un robot mobile. Nous verrons pourquoi le contrôle flou s'est implanté, comment le réaliser et illustrerons ensuite ses principes sur l'exemple de régulation de vitesse et de position d'une machine électrique et à la navigation floue d'un robot mobile dans un environnement inconnu.

Les caractéristiques de chacun des blocs constituant la structure générale d'un régulateur flou sont présentés [1], à savoir la:

- *Fuzzification*
- *Inférence*
- *Défuzzification.*

I.7.2. Les raisons d'un contrôle flou

Actuellement, la technique de la mesure et de la régulation est basé essentiellement sur la connaissance et l'analyse mathématique (équations différentielles, fonctions de transfert,...) du processus.

Le dimensionnement d'un contrôleur conventionnel PID (Proportionnel- Intégral- Dérivée) demande la connaissance précise du modèle du système à contrôler. Les valeurs d'entrée du PID doivent être mesurées le plus exactement possible pour éviter d'entacher d'une erreur l'image de l'état du système qu'elles décrivent.

Un contrôleur flou, lui ne demande aucune de ces deux spécifications. Il n'est pas nécessaire de connaître le modèle analytique du processus pour le concevoir. Le contrôleur flou ne traite pas de relations mathématiques bien définies mais utilise des inférences avec plusieurs règles, se basant sur des variables linguistiques, ces inférences sont alors traitées par les opérateurs de la logique floue.

La connaissance du modèle mathématique du processus n'est pas nécessaire, tout au moins quant aux premières réalisations. C'est l'expérience des opérateurs du procédé ou les connaissances des experts, qui sont prises en compte pour établir la commande floue. Les algorithmes de réglage conventionnels sont alors remplacés par une série de règles linguistiques de la forme **Si**...,**Alors**.… Ainsi, on obtient un algorithme heuristique.

La commande par logique floue peut s'appliquer à tout domaine de la commande traditionnelle. De plus, elle peut opérer lorsque les procédés à commander sont mal connus ou difficiles à décrire précisément, ou lorsque les variables sont évaluées subjectivement et exprimées en langage naturel et non numériquement.

Le réglage par logique floue se prête particulièrement bien à deux domaines d'applications [7]:

- Conception de régulateurs pour des processus mal modélisables .
- Conception de régulateurs non linéaires pour des processus modélisables.

La commande floue est simple à réaliser, flexible et donc facilement adaptable aux conditions de fonctionnement du processus ou à une installation particulière. Elle est robuste face aux perturbations qui peuvent affecter le processus.

Les règles sont faciles à comprendre et à modifier puisqu'elles sont exprimées par des termes de la langue naturelle. Le développement d'un régulateur flou est économique, d'autant plus qu'il existe des logiciels d'application et de plus en plus des composants spécialisés.

Enfin, un contrôleur flou bénéficie d'un aspect d'adaptabilité, de robustesse et de stabilité et plusieurs études et réalisations industrielles ont montré que ce dernier peut donner des meilleures résultats que les régulateurs classiques [7].

I.7.3. Principe d'un contrôleur flou

Un contrôleur flou (figure II-1) ne diffère pas tellement d'un contrôleur traditionnel. On retrouve à chaque fois un bloc de traitement, un bloc d'entrée (quantification, calculs préalables...) et un bloc de sortie (pour la détermination de la commande u à partir de l'incrément du par exemple) [7].

Deux blocs supplémentaires apparaissent dans le cas d'un contrôleur flou: un bloc de fuzzification et un bloc de défuzzifcation . Le bloc de fuzzification constitue l'interface entre le monde physique et celui des sous - ensemble d'inférence (*inférence engine*) et une base de règles (*rules base*). Le rôle de ce dernier bloc sera d'échafauder le raisonnement.

Le bloc de fuzzification convertira les valeurs d'entrées en sous ensembles flous. Le moteur d'inférence activera les règles dont les prémisses seront vérifiées. Chaque règle activée donnera lieu à un sous-ensemble de sortie. Il restera au bloc de défuzzification à agréger ceux-ci et en extraire une action précise et réalisable au niveau de la commande.

I.7.3.1. La fuzzification

Dans les problèmes de commande, les données observées sont habituellement physiques (réelles). Or le traitement de ces données est basé sur la théorie des ensembles flous; ceci nécessite donc une procédure de fuzzification.

L'opération de fuzzification représente le passage des grandeurs réelles (ou physiques) aux valeurs floues. Cette étape nécessite souvent une conversion analogique/digitale, ainsi que le traitement des grandeurs mesurées et leur transformation en variables linguistiques avec la définition des fonctions d'appartenance.

A l'univers de discours d'une entrée X (ensemble des valeurs possibles de x), on associera N sous - ensemble flous notés Ei (valeurs linguistiques). Chacun de ceux-ci sera défini par sa fonction d'appartenance $\mu_{Ei}(x)$, $0 < \mu_{Ei}(x) < 1$.

Le rôle du bloc de fuzzification sera de déterminer pour un x_i donné (variable observée ou mesurée) les degrés d'appartenance de x_i à chacun des sous - ensemble flous E_i.

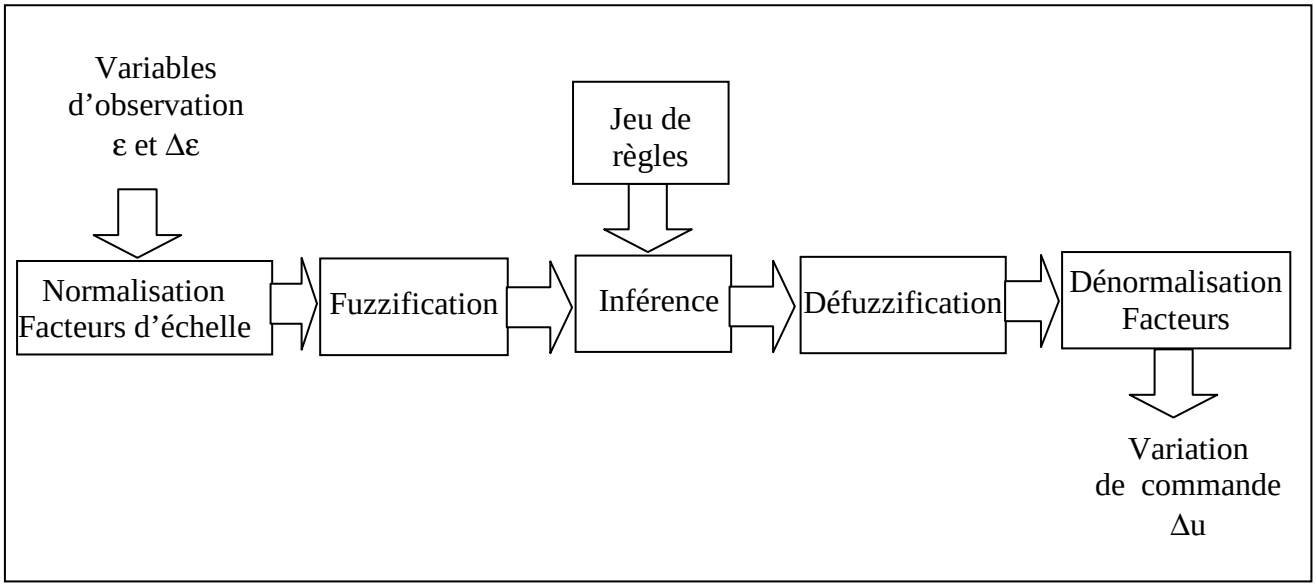


Figure II -1 Configuration interne d'un contrôleur par logique floue.

La fuzzification proprement dite consiste à définir les fonctions d'appartenance pour les différentes variables d'entrée et de sortie. Dans le cas du réglage par logique floue, on utilise en général des formes trapézoïdales et triangulaires pour les fonctions d'appartenance.

Dans ce but, les grandeurs physiques (par exemple l'erreur et la dérivée de la grandeur à réguler) sont réduites à des grandeurs normalisées [1]. On suppose que ces dernières varient normalement dans le domaine : $-1 \leq x \leq 1$.

Ceci pose le problème de choix des facteurs d'échelles. Les gains d'adaptation et de normalisation jouent alors un rôle extrêmement important car ce sont eux qui fixent les performances dynamiques de la commande.

En général, on introduit pour une variable x trois, cinq ou sept ensembles flous, représentés par des fonctions d'appartenance, comme le montre la figure (II-2). Le nombre des ensembles dépend de la résolution et de l'intervention du réglage désiré [7].

Une subdivision plus fine, c'est-à-dire plus de sept ensembles flous, n'apporte en général aucune amélioration du comportement dynamique du système contrôlé par la logique floue. Par contre, un tel choix complique la formation des règles d'inférences et augmente le temps de traitement.

Les différents ensembles flous sont caractérisés par des désignations standards: la signification des symboles est indiquée au tableau (II-1).

Symboles	Significations
NG	Négatif Grand
NM	Négatif Moyen
NP	Négatif Petit
EZ	Environ Zéro
PP	Positif Petit
PM	Positif Moyen
PG	Positif Grand

Tableau II -1 Désignation standard des ensembles flous.

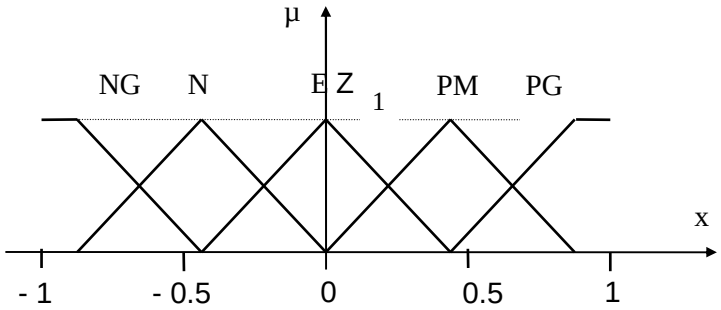


Figure – II - 2 Fuzzification avec cinq fonctions d'appartenance.

I.7.3.2. Inférence

Dans cette partie du régulateur les valeurs des variables linguistiques d'entrée et de sortie sont liées par plusieurs règles qui doivent tenir compte du comportement statique et dynamique du système à régler ainsi que des buts de réglage envisagés en particulier le circuit de réglage doit être stable et bien amorti. La stratégie de réglage dépend essentiellement des inférences adoptées. Il n'est pas possible d'indiquer des règles précises, l'expérience joue ici un rôle important.

Pour exprimer les inférences ils existent plusieurs possibilités à savoir par description linguistique et symbolique, ainsi que par tableaux et matrices d'inférence [7]. Dans ce cours, nous utiliserons cette dernière description.

I.7.3.2.1. Description par matrice d'inférence

Le grand intérêt de cette méthode est la possibilité de regrouper les règles en une matrice. C'est une représentation graphique, à l'intersection d'une colonne et d'une ligne se trouve l'ensemble correspondant de la variable de sortie x_R , défini par une règle d'inférence.

Les variables d'entrées sont liées par l'opérateur **Et**, tandis que les variables de sortie des différentes règles sont à considérer par l'opérateur **Ou**. Cette méthode impose donc une restriction au niveau des règles, dont la condition ne peut contenir que l'opérateur **Et**.

Dans la plupart des contrôles. un nombre de deux variables en entrée est suffisant et on peut donc utiliser un tel tableau. Son implémentation est facile et l'approximation, même grossière, introduite au niveau des valeurs ne perturbe que peu le fonctionnement. De par sa nature, le contrôleur flou tient en effet compte d'une certaine imprécision.

Lorsqu'il y a plus de deux variables, il faut juxtaposer plusieurs matrices. Cependant, ce genre de description devient complexe lorsqu'il y a plus de trois ou quatre variables et si ces dernières sont subdivisées en un nombre élevé d'ensembles.

Si toutes les positions de la matrice d'inférence sont remplies, on parle de règles d'inférence complètes, dans le cas contraire il s'agit de règles d'inférence incomplètes (figure II-3).

X_R		X_1				
		NG	NM	EZ	PM	PG
X_2	NG			PG	PM	
	NM			PM	EZ	NM
	EZ	PG	PM	EZ	NM	NG
	PM	PM	EZ	NM		
	PG		NM	NG		

Figure II -3 Matrice d'inférence incomplètes pour deux variables linguistique x_1 et x_2

I.7.3.2.2. Traitement numérique des inférences

En général deux ou plusieurs règles sont activées en même temps. Une règle d'inférence floue est activée lorsque le facteur d'appartenance lié à la condition de cette règle est non nul. Il existe plusieurs possibilités pour les opérateurs qui s'appliquent aux fonctions d'appartenance. On introduit alors la notion de méthode d'inférence.

En réglage par logique floue, on utilise en général une des méthodes suivantes [7]:

- *Méthode d'inférence Max-Min.*
- *Méthode d'inférence Max-Prod.*
- *Méthode d'inférence Somme-Prod.*

Le nom de la méthode désigne les opérateurs utilisés respectivement pour l'agrégation et l'implication. Le tableau suivant indique la manière de leur utilisation :

Méthodes	Opérateurs sur Prémisses		Opérateur Implication	Opérateur Agrégation
	Ou	**Et**	**Imp**	
Max-min	Max	Min	Min	Max
Max-prod	Max	Min	Prod	Max
Som-prod	Som	Prod	Prod	Som

I.7.3.2.3. Méthode d'inférence somme-produit

La méthode d'inférence somme-produit réalise, au niveau de la condition, l'opérateur **Ou** par la formation de la somme moyenne, tandis que l'opérateur **Et** est réalisé par la formation du produit. La conclusion de chaque règle, précédée par **Alors**, liant le facteur d'appartenance de la condition avec la fonction d'appartenance de la variable de sortie par l'opérateur **Et**, est réalisée par la formation du produit. L'opérateur **Ou** qui lie les différentes règles est réalisé par la formation de la somme moyenne; ainsi s'explique la désignation par *Som-Prod* de cette méthode d'inférence.

La méthode d'inférence *Som-Prod* est représentée graphiquement à la figure (II-4). Sur la figure (II-5), nous avons représenté les deux autres méthodes d'inférence.

La fonction d'appartenance résultante est donnée par :

$$\mu_{RES}(x_R) = [\ \mu_{R1}(x_R) + ...+ \mu_{Rm}(x_R)\ ]/m \qquad \text{(II-1)}$$

Avec: $\mu_{Ri}(x_R) = \mu_{ci} .\mu_{0i}(x_R)$, $i = 1.....,m$

$\mu_{Ri}(x_R)$: La fonction d'appartenance partielle (résultante de la règle i)
μ_{ci} : Facteur d'appartenance de la condition
$\mu_{0i}(x_R)$: Fonction d'appartenance de la conclusion
m : Le nombre de règles intervenant dans l'inférence.

Pour illustrer cette méthode on présente l'exemple suivant ; celle d'un régulateur flou avec deux variables d'entrée (x_1) et (x_2) et une variable de sortie (x_R). Chacune est décomposée en trois ensembles NG, EZ, PG. On suppose que les valeurs sont : $x_1 = 0.44$ et $x_2 = 0.67$

Pour l'inférence :

x_R : = Si (x_1 PG Et x_2 EZ) Alors x_R=EZ Ou
Si (x_1 EZ Ou x_2 NG) Alors x_R=NG

Avec les facteurs d'appartenance $\mu_{PG}(x_1 = 0.44) = 0.67$ et $\mu_{EZ}(x_2 = 0.67) = 0.33$, la première condition prend le facteur d'appartenance $\mu_{Ci} = 0.22$ (produit des deux valeurs).

Pour la condition de la deuxième règle, on a :
$\mu_{EZ}(x_1 = 0.44) = 0.33$ et $\mu_{NG}(x_2 = 0.67) = 0.67$, ce qui donne $\mu_{Ci} = 0.5$
Les fonctions d'appartenance hachurées sur la figure (II-4), s'obtiennent par la formation de la somme (valeur moyenne de μ_{R1} et μ_{R2}).

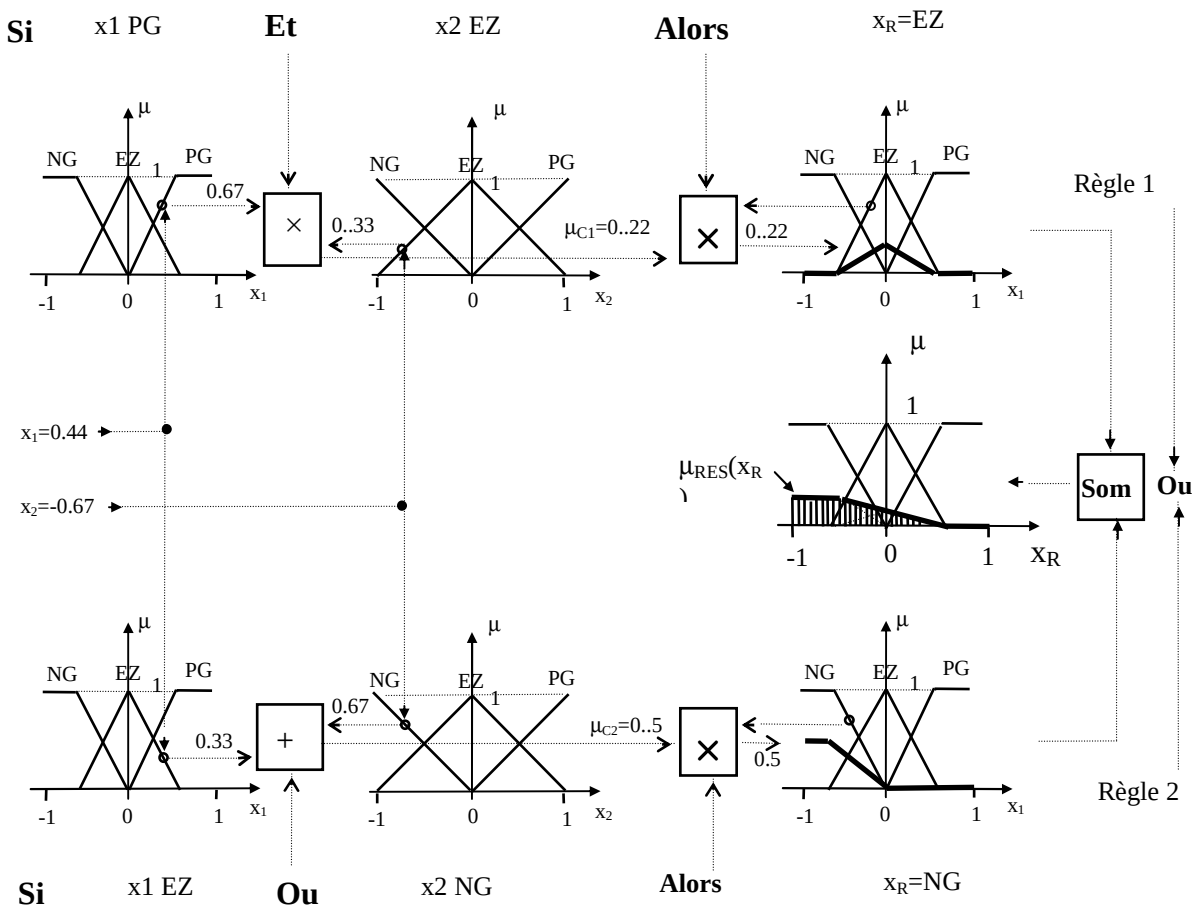
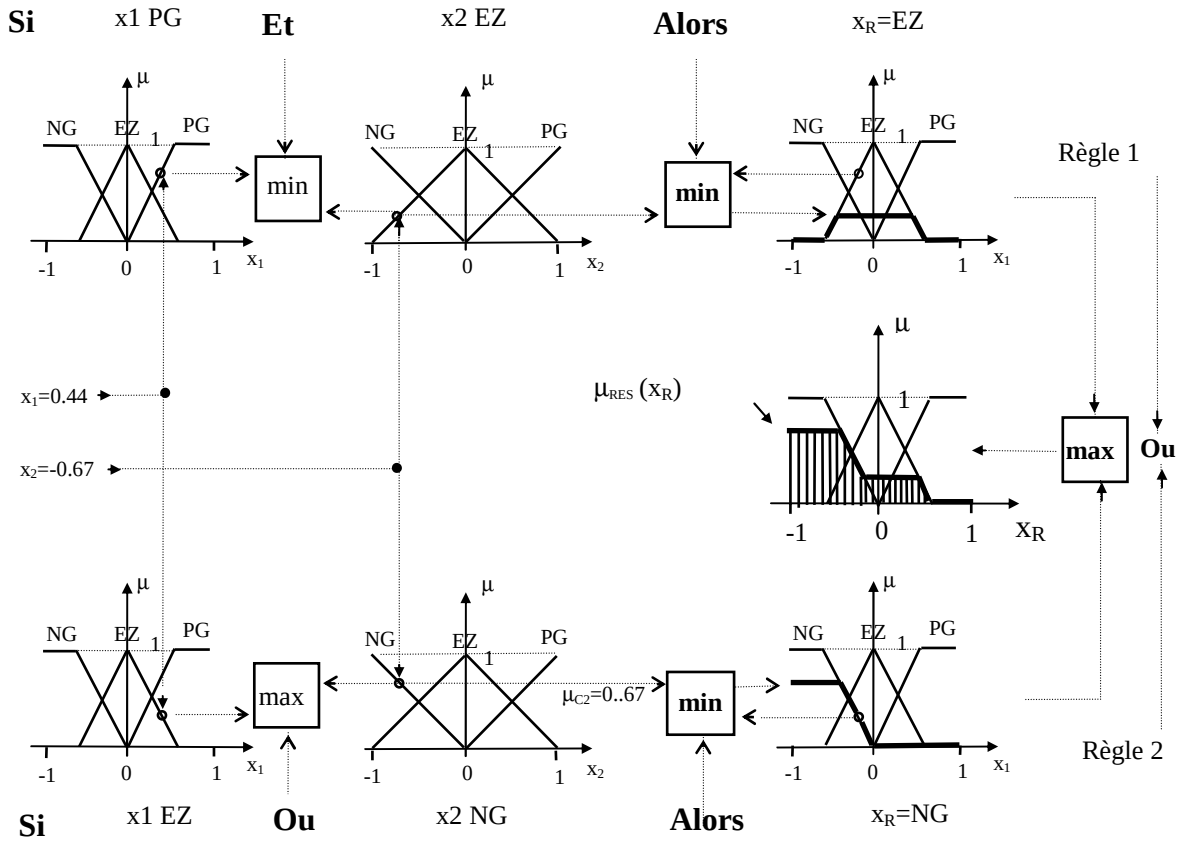


Figure II-4: Méthode d'inférence **Som-Prod** pour deux variables d'entrée et deux règles



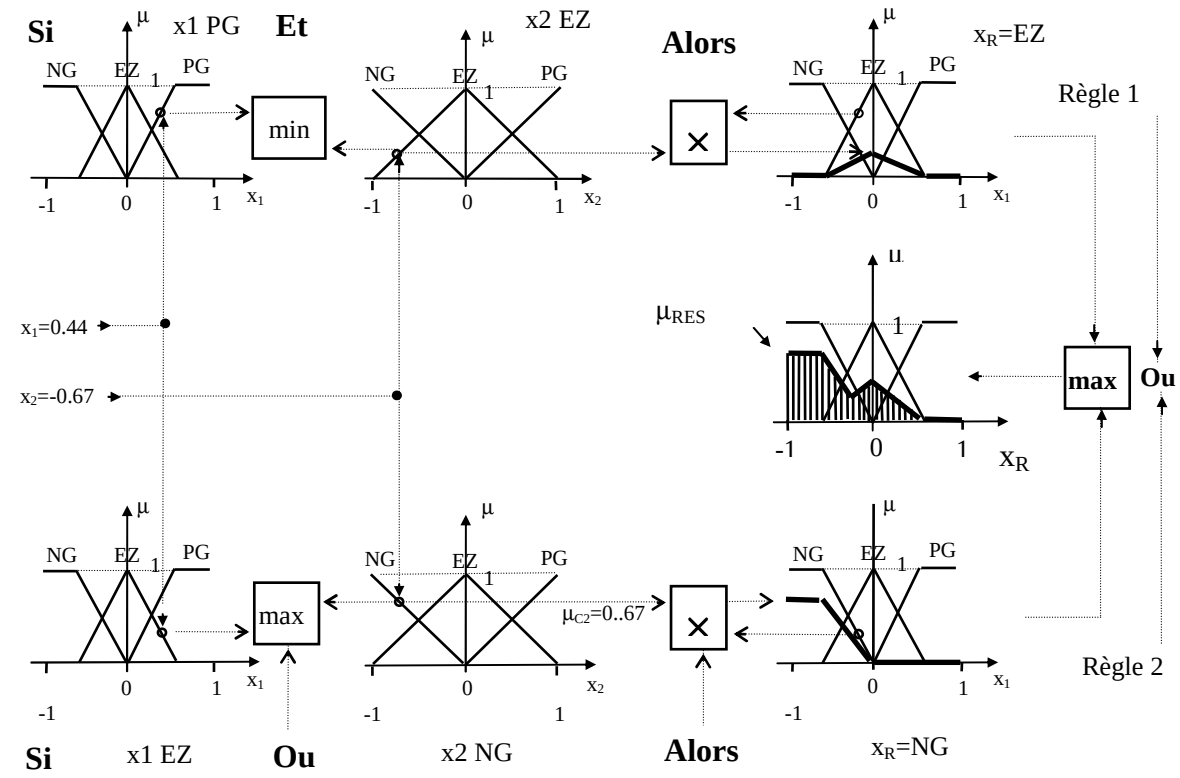


Figure II-5 Principe des méthodes d'inférence **Max-Min** et **Max-Prod**

I.7.4. La défuzzification

La dernière étape du contrôle, appelée défuzzification consiste à définir précisément quelle doit être l'action sur le processus. En effet, le procédé ne peut pas interpréter des ordres du type « Petit » ou « Grand », etc…., on doit lui envoyer une valeur physique.

Les méthodes d'inférences fournissent une fonction d'appartenance résultante $\mu rés(x_R)$ pour la variable de sortie x_R. L'opération de défuzzification permet de calculer à partir de cette dernière la valeur réelle de la variable de sortie à appliquer au processus. On doit souvent prévoir un traitement de signal de sortie et sa conversion numérique - analogique

Le choix d'une méthode de défuzzification est un point très délicat lors de l'élaboration d'une technique de contrôle en logique floue. Celui-ci conditionnera en effet grandement l'évolution dynamique de la commande. On distingue trois méthodes différentes (figure II-6): celle du maximum, celle de la moyenne des maxima et celle du centre de gravité (ou centroïde). Il est toutefois reconnu que la méthode de centre de gravité donne les meilleurs résultats [1], [6].

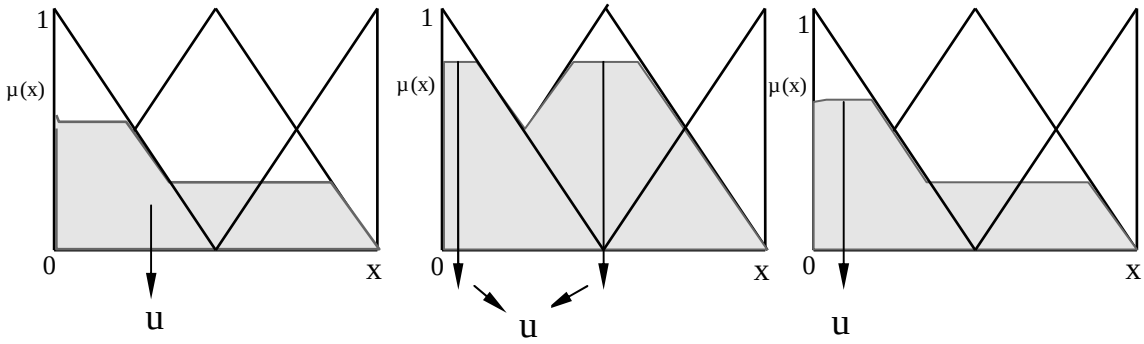


Figure II-6 Principe des différentes méthodes de défuzzification.

Méthode du maxima :

Cette méthode consiste à choisir comme sortie x0 du bloc de déffuzification, une des valeurs possédant la plus grande appartenance au sous-ensemble flou x.

Il se peut que le système possède plusieurs maxima identiques, dans ce cas et afin d'éviter un choix arbitraire, on choisit d'effectuer la moyenne des maxima.

La méthode du maximum à l'avantage d'être simple, rapide et facile. Elle est malheureusement ambiguë et provoque de nombreuses discontinuités.

Méthode de la moyenne des maxima :

Dans le cas ou plusieurs sous-ensembles ont le même maximum, on réalise une commande $u = \frac{\sum u_i}{r}$, u_i étant la commande issue du ieme sous-ensemble flou sélectionnable.

r : nombre de maxima identiques

Les avantages et inconvénients de la méthode de la moyenne des maxima restent grosso modo ceux de la méthode du maximum.

Méthode du centre de gravité (centroide) :

Cette méthode consiste a calculer le centre de gravité de la fonction d'appartenance résultante μrés(x_R). L'abscisse u de ce centre de gravité donne la valeur de commande à appliquer et peut être déterminée par la relation générale suivante :

$$u = \frac{\int_{-1}^{1} x_R \mu_{rés}(x_R) d x_R}{\int_{-1}^{1} \mu_{rés}(x_R) d x_R} \qquad \text{(II-2)}$$

L'intégrale au dénominateur donne la surface, tandis que l'intégrale au numérateur correspond au moment de la surface.

Cette méthode va permettre d'éviter de trop grandes discontinuités et supprimera toute ambiguïté. Elle semble donc optimale mais son implémentation est difficile et surtout coûteuse en calculs. Elle se simplifie notablement lorsqu'on utilise la méthode d'inférence Som-prod ou des singletons pour les fonctions d'appartenance des variables de sortie.

II-4. Contrôleurs flous usuels :

Les contrôleurs flous sont principalement de deux types:

- *Contrôleur flou type Mamdani*
- *Contrôleur flou type Sugeno*

Pour un système à deux variables, les règles floues sont de la forme :

« **SI** x est Ai **Et** y est Bi **ALORS** z est Ci » (II-3)

où Ai et Bi sont des sous-ensembles flous, par contre Ci peut appartenir aussi bien au domaine symbolique (sous-ensemble flou) qu'au domaine numérique.

L'originalité de la méthode de Sugeno réside dans le fait que la conclusion de chaque règle n'appartient pas au domaine symbolique, mais est définie sous forme numérique comme une combinaison linéaire des entrées

.

Selon la méthode de Sugeno, les règles floues, dans le cas de deux variables, s'expriment donc selon la forme suivante :

« **SI** x est Ai **Et** y est Bi **Alors** $z = p_0+p_1.x +p_2.y$ » (II-4)

On parle dans ce cas de contrôleur flou de type Sugeno d'ordre 1.

Dans la suite de ce mémoire, nous n'utiliserons qu'un raisonnement simplifié de Sugeno (contrôleur flou de type Sugeno d'ordre 0) où les règles floues utilisées sont du type :

« **SI** x est Ai **Et** y est Bi **Alors** $z = p_0$ » (II-5)

Dans le contrôleur flou type Sugeno, les étapes d'agrégation et de défuzzification des règles floues se font simultanément et la relation (II.2) devient :

$$u = \frac{\sum \mu_{i\,i}\, z_{i\,i}}{\sum \mu_i} \qquad \text{(II-6)}$$

Cette méthode est plus simple à mettre en œuvre et donne aussi de bons résultats en commande floue que la méthode de Mamdani . Le calcul en temps réel de cette expression ne pose pas de problème.

Une remarque peut être formulée sur le nom donné à cette étape. En effet, elle est appelée « *défuzzification* » alors qu'elle ne manipule aucune donnée floue. Ce choix a été dicté afin d'établir une similitude entre ce type de contrôleur et le contrôleur de type Mamdani où le cheminement « *fuzzification – inférence floue – défuzzification* » a été introduit. A la place de « *défuzzification* », le terme « *agrégation* » aurait été préférable.

I.8. CONCEPTION DES REGULATEURS FLOUS

I.8.1. Exemple 1 : Régulateur flou de vitesse et de position d'une machine électrique

Nous allons maintenant illustrer les principes du contrôleur flou sur l'exemple de la régulation de vitesse et de position d'une machine électrique.

La phase de conception d'un contrôleur flou passe toujours par quatre stades que nous allons détailler successivement.

I.8.1.1 Régulateur flou de vitesse

- **1 ére étape : Choix des entrées et sorties**

Il s'agit de déterminer les caractéristiques fonctionnelles (1) et opérationnelles (2) du contrôleur.

(1)- Il faut d'abord choisir les variables d'entrée et de sortie. Leur choix dépend du contrôle que l'on veut réaliser. Que souhaite-t-on au juste commander ? A l'aide de quels paramètres va-t-on obtenir la commande ?

(2)- Il faudra ensuite se pencher sur le domaine des valeurs que pourront prendre ces variables). On partitionne alors ces domaines en intervalles, auxquels on associe un label descriptif (valeurs linguistique). Cette étape revient à définir les univers des discours des variables d'entrée et de sortie et les diviser en sous-ensembles flous. Cette répartition est intuitif et basé sur l'expérience. On est d'ailleurs généralement amené à l'affiner en cours de conception. Une règle de bonne pratique est de fixer 5 à 9 intervalles par univers de discours. Il faut également prévoir un plus grand nombre de zones à proximité du point de fonctionnement optimal pour en faciliter l'approche régulière [2].

Illustration sur le régulateur de vitesse

Dans le cas de la régulation de vitesse, on a besoin habituellement de l'erreur ($e=\Omega_{r\,ref}-\Omega_r$) et de la dérivée d'erreur (de) et parfois de l'intégration d'erreur :

$$\begin{aligned} e(k) &= \Omega_{rref}(k) - \Omega_r(k) \\ de(k) &= e(k) - e(k-1) \end{aligned} \qquad \text{(III-1)}$$

La sortie du régulateur de vitesse est la valeur du courant de référence iq ou le couple dans le schéma de la commande d'une machine électrique. Si cette sortie est directement appliquée au processus, le contrôleur est alors appelé contrôleur flou de type PD [6] (figure III-1) et. on peut écrire :

iq= Fuzzy (e, de)

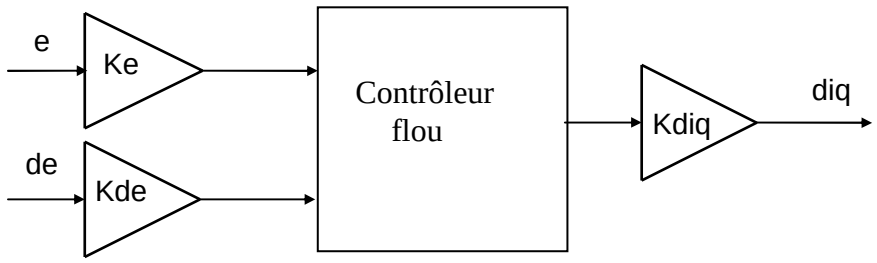


Figure III-1 Schéma de principe d'un contrôleur flou de type PD

Par contre, si la sortie du contrôleur flou est considérée comme un incrément de commande, le contrôleur est appelé contrôleur flou de type PI [6] (figure III-2) et on peut écrire :

diq = F_{uzzy} (e, de) ou encore $iq=F_{uzzy}\left(\int e dt, \int \dot{de}\right)$;

soit $iq(k) = diq(k) + iq(k-1)$ (III-2)

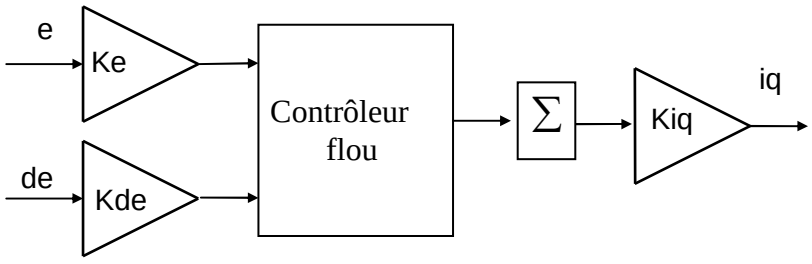


Figure III-2 Schéma de principe d'un contrôleur flou de type PI

Le contrôleur de type PID (figure III-3) peut être obtenu en combinant des contrôleurs flous de type PI et PD de façon suivante :

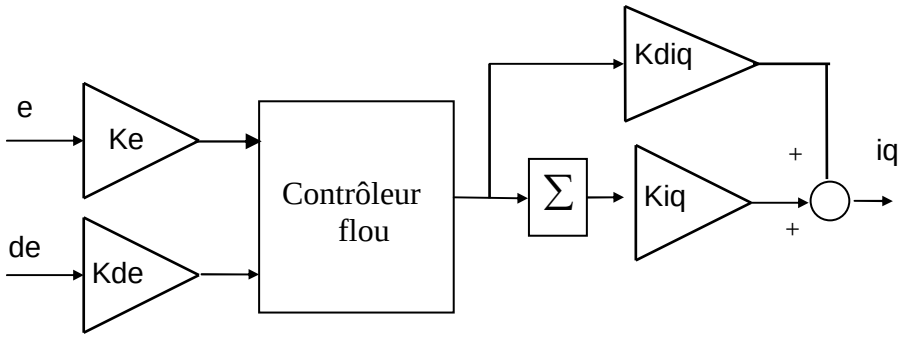


Figure III-3 Schéma de principe Contrôleur flou de type PID

On remarque que cette structure de commande floue de type PID (figure V-9) est en fait une association en série d'un contrôleur flou de base et d'une structure de régulation de type PI, qui, elle, n'est pas floue [6].

Comme les fonctions d'appartenance sont normalisées entre [-1, 1], les variables sont multipliées avec des gains proportionnels. Finalement, la structure du régulateur de vitesse à logique floue est représentée par la figure (III-4).

D'après ce schéma, le système est composé :

- du contrôleur flou composé :
- d'un bloc de calcul de variation de l'erreur au cours du temps (de) ;
- des facteurs d'échelles associés à l'erreur, à sa dérivée et à la commande (diq);
- d'un bloc de fuzzification de l'erreur , de sa variation et de la commande;
- des règles de contrôle flou et d'un moteur d'inférence ;
- d'un bloc de défuzzification utilisé pour la transformation de la commande floue en valeur numérique ;
- d'un bloc intégrateur
- du processus à contrôler.

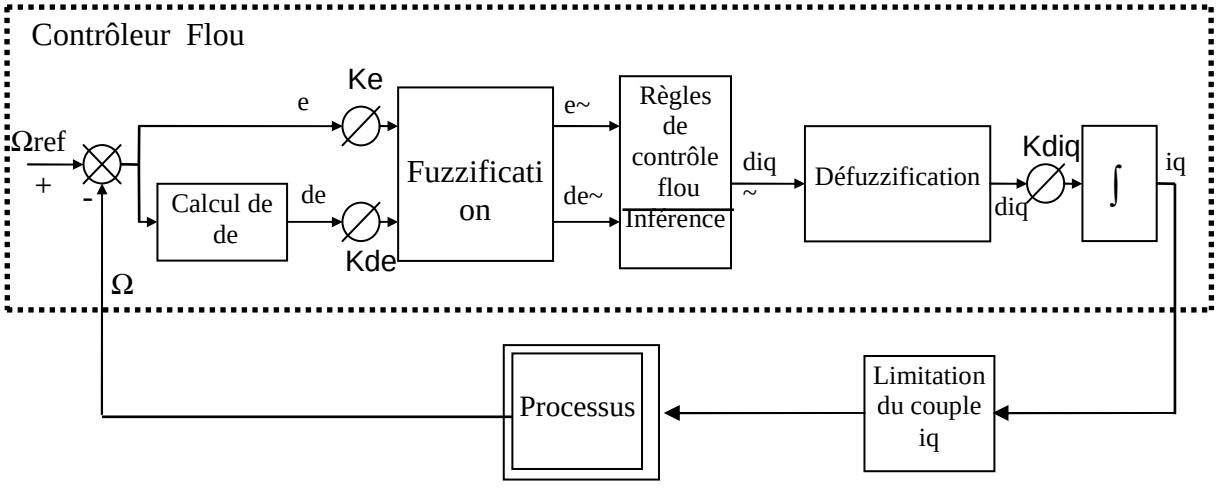


Figure III-4 Structure du régulateur de vitesse à logique floue

- **2 éme étape : Définition des fonctions d'appartenance**

La première étape de conception a permis de cerner au mieux les caractéristiques linguistiques des variables. Il faut maintenant définir complètement les sous-ensembles flous, c'est à dire expliciter leurs fonctions d'appartenance. Une fois encore, l'intuition et l'expérience auront leur rôle à jouer. Quelques principes ressortent de la pratique: choix de fonctions triangulaires ou trapézoïdales, recouvrement d'une fonction de 10 à 50% de l'espace des sous-ensembles voisins, somme des degrés d'une zone de recouvrement égale à 1 (degré maximal d'appartenance) [6], [8].

Illustration sur l'exemple

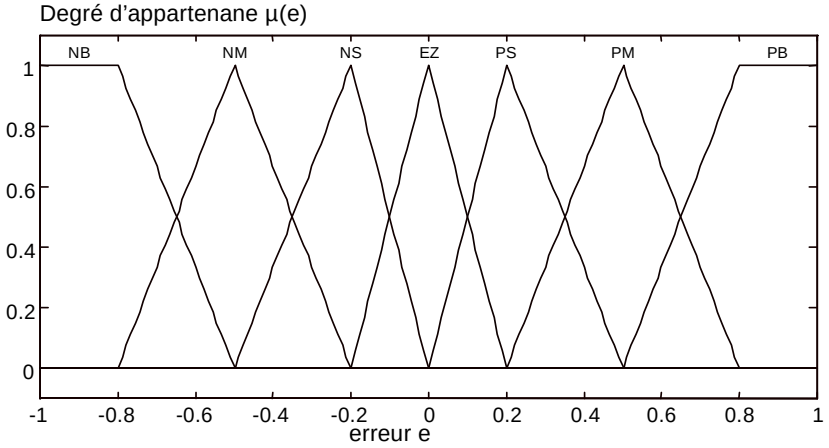
Les fonctions d'appartenance des variables d'entrée sont illustrées par la figure (III-5) avec :

NB : Negative Big (Négative Grand) PB : Postive Big (Positive Grand)
NM : Negative Medium (Négative Moyenne) PM : Postive Medium (Positive Moyenne)
NS : Negative Small (Négative Petit) PS : Postive Small (Positive Petit)
ZE : Zero

On constate que les fonctions d'appartenance de l'erreur ont une forme asymétrique créant une concentration autour de zéro qui améliore la précision près du point de fonctionnement désiré.

Pour la même raison, les formes des fonctions d'appartenance de la variable de sortie sont également asymétriques (figure III-6). Cependant, nous introduisons deux sous-ensembles additionnels compte-tenu de la sensibilité de cette variable [2].

NVB : Negative Very Big (Negative Très Grand)
PVB : Positive Very Big (Positive Très Grand)



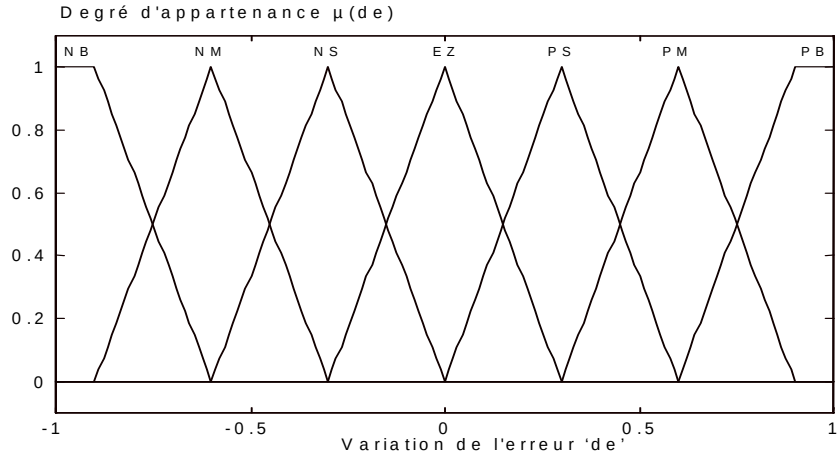


Figure III -5 Fonctions d'appartenance des variables d'entrée

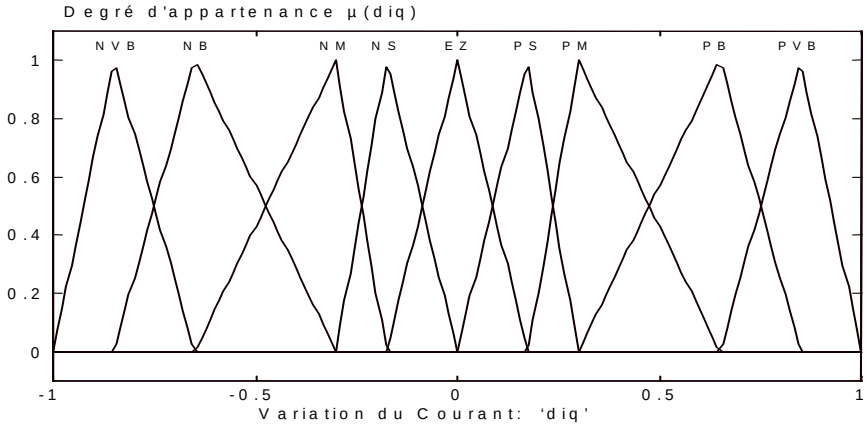


Figure III -6 Fonctions d'appartenance de la variable de sortie

- **3 éme étape : Définition du comportement du contrôleur flou**

Cette étape concerne l'élaboration de la base de règle du contrôleur. C'est de nouveau à un expert à sa connaissance du problème que l'on se fiera le plus souvent. Dans le cadre de la régulation (asservissement), on utilise fréquemment l'erreur (observation) et la variation de l'erreur (dynamique du processus). A partir de ces deux entrées, traduites sous la forme de variables floues, il est possible de déterminer les règles dans le domaine temporel et on peut construire une matrice ***Situation/Action*** reprenant toutes les possibilités linguistiques de celles-ci [1].

Analyse du comportement dynamique - Détermination du jeu de règles

L'analyse temporelle, qui doit conduire à établir les règles du contrôleur flou, peut par exemple consister à considérer la réponse à un échelon d'un processus à piloter en fonction des objectifs que l'on se sera fixé en boucle fermée, et à écrire les règles pour chaque type de comportement du processus :

a)- Pour expliquer la procédure à suivre [6], on considère les neuf points indiqués sur la réponse à un échelon (figure III-7) et, pour chacun de ces points, on explicite l'expertise sous la forme suivante :

* 1 Si e = PB Et de = ZE Alors du = PB (départ, commande importante)
* 2 Si e = PB Et de = NS Alors du = PM (augmentation de la commande pour gagner l'équilibre)
* 3 Si e = PM Et de = NS Alors du = PS (très faible augmentation de u pour ne pas dépasser)
* 4 Si e = PS Et de = NS Alors du = ZE (convergence vers l'équilibre correct)
* 5 Si e = ZE Et de = NS Alors du = NS (freinage du processus)
* 6 Si e = NS Et de = NS Alors du = NM (freinage et inversion de la variation de la commande)
* 7 Si e = NM Et de = ZE Alors du = NM (rappel du processus vers l'équilibre correct)
* 8 Si e = NS Et de = PS Alors du = ZE (convergence vers l'équilibre correct)
* 9 Si e = ZE Et de = ZE Alors du = ZE (équilibre)

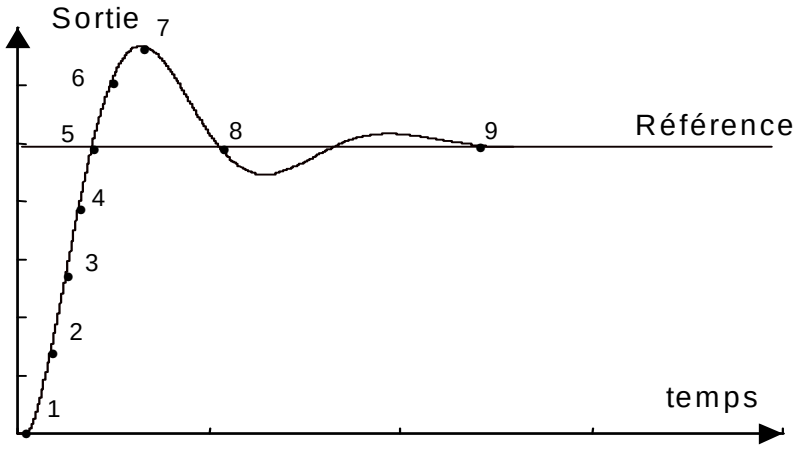


Figure III -7 Ecriture du jeu de règles grâce à une analyse temporelle

En décrivant point par point le comportement du processus et l'action de variation de commande à appliquer, on en déduit la table du contrôleur flou de base (figure III-8) qui correspond en fait à table de règles très connue de Mac Vicar - Whelan [6] :

de \ e	NB	NM	NS	ZE	PS	PM	PB
PB	ZE	PS	PM	PB	PB	PB	PB
PM	NS	ZE	PS	PM	PB	PB	PB
PS	NM	NS	ZE	PS	PM	PB	PB
ZE	NB	NM	NS	ZE	PS	PM	PB
NS	NB	NB	NM	NS	ZE	PS	PM
NM	NB	NB	NB	NM	NS	ZE	PS
NB	NB	NB	NB	NB	NM	NS	ZE

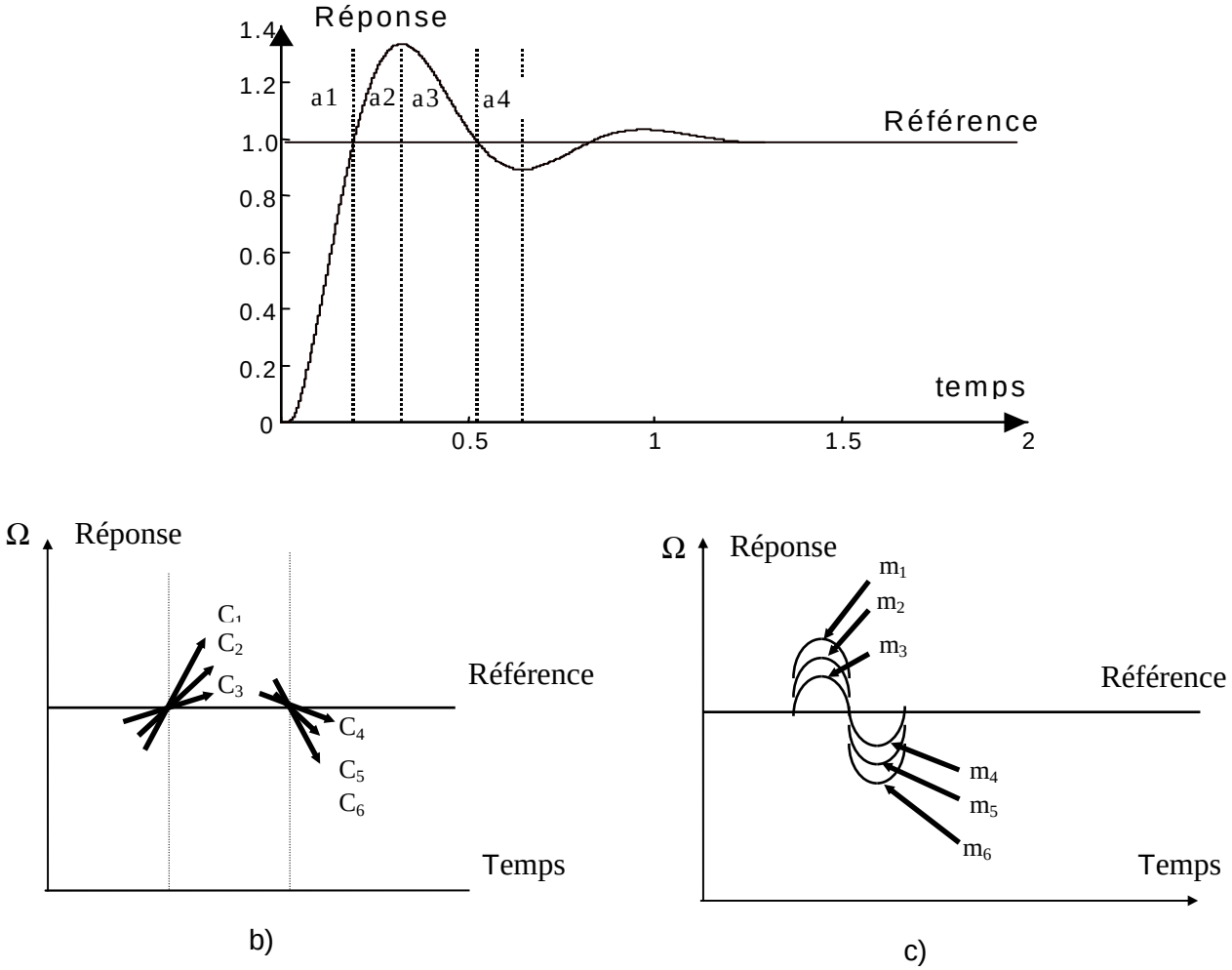
Figure III -8 Table de règles de MacVicar-Whelan

b)- Pour déduire les autres règles, nous procédons à nouveau à une autre expertise [3]. La forme générale de la réponse de vitesse est représentée sur la figure (III-9). Selon l'amplitude de e

et le signe de la variation d'erreur de, la réponse de vitesse est divisée en quatre régions. Les indices utilisés pour identifier chaque région sont définies comme suit :

a_1 : e > 0 et de < 0, a_2 : e < 0 et de < 0,
a_3 : e < 0 et de > 0, a_4 : e > 0 et de > 0,

Pour accroître la résolution de la représentation dynamique, les réponses autour du point de fonctionnement et aux extremums de la figure (III-9-a) sont représentées respectivement sur la figure (III-9-b) et (III-9-c).



Pour identifier la pente de la réponse lors du passage par le point de référence on utilise l'indice c_i défini comme suit :

Figure III -9 Comportement dynamique de la réponse de vitesse

c_1 : (e > 0 → e < 0) et de <<< 0
c_2 : (e > 0 → e < 0) et de << 0
c_3 : (e > 0 → e < 0) et de < 0
c_4 : (e < 0 → e > 0) et de > 0
c_5 : (e < 0 → e > 0) et de >> 0
c_6 : (e < 0 → e > 0) et de >>> 0

Quant à l'indice m_i représentatif du dépassement de la consigne, il est défini par :

m_1 : de ≈ 0 et e <<< 0 m_4 : de ≈ 0 et e > 0
m_2 : de ≈ 0 et e << 0 m_5 : de ≈ 0 et e >> 0
m_3 : de ≈ 0 et e < 0 m_6 : de ≈ 0 et e >>> 0

Les trois types d'indices mentionnés ci-dessous sont combinés ensemble, ceci est représenté sur la table de la figure (III-10).

e / de	NB	NM	NS	ZE	PS	PM	PB
NB	a_2			c_1	a_1		
NM				c_2			
NS				c_3			
ZE	m_1	m_2	m_3	ZE	m_4	m_5	m_6
PS	a_3			c_4	a_4		
PM				c_5			
PB				c_6			

Figure III –10 Règles linguistiques de contrôle

Finalement le tableau de la figure (III-8) est légèrement modifié pour tenir compte de la variable de sortie qui est formée de neuf valeurs floues (figure III-11).

		e						
		NB	**NM**	**NS**	**ZE**	**PS**	**PM**	**PB**
de	**NB**	NVB	NVB	NVB	NB	NM	NS	ZE
	NM	NVB	NVB	NB	NM	NS	ZE	PS
	NS	NVB	NB	NM	NS	ZE	PS	PM
	ZE	NB	NM	NS	ZE	PS	PM	PB
	PS	NM	NS	ZE	PS	PM	PB	PVB
	PM	NS	ZE	PS	PM	PB	PVB	PVB
	PB	ZE	PS	PM	PB	PVB	PVB	PVB

Figure III -11 Base de règles du régulateur I de vitesse

Dans le tableau (figure III-11), chaque élément formalise une règle comme, par exemple :

Si [e(k) est NM] **Et** [de(k) est ZE], **Alors** [diq(k) est NM]

Cet ensemble de règles regroupe toutes les situations possibles du système évaluées par les différentes valeurs attribuées à e et à sa variation de et toutes les valeurs correspondantes de la variation de la commande diq.

Les univers de discours normalisés associés à « e », « de » et à « diq » sont identiques et sont fixés entre –1 et +1. Cette normalisation des variables d'entrée (sortie) nécessite donc l'obtention de facteurs d'échelles respectifs pour chacune d'elles.

L'évaluation des gains proportionnels provient de l'expérience. Pour le gain Ke , par exemple, on peut commencer avec un facteur qui dépend de l'erreur maximale. Effectivement ces valeurs font partie de la procédure d'évaluation par simulation. On a trouvé les valeurs suivantes pour la machine électrique simulée :

Ke = 0.08 Kde = 1.25 Kdiq = 14.9

Dans une deuxième approche d'un régulateur à logique floue, on utilise différentes fonctions d'appartenance pour la variable de sortie (figure III-12):

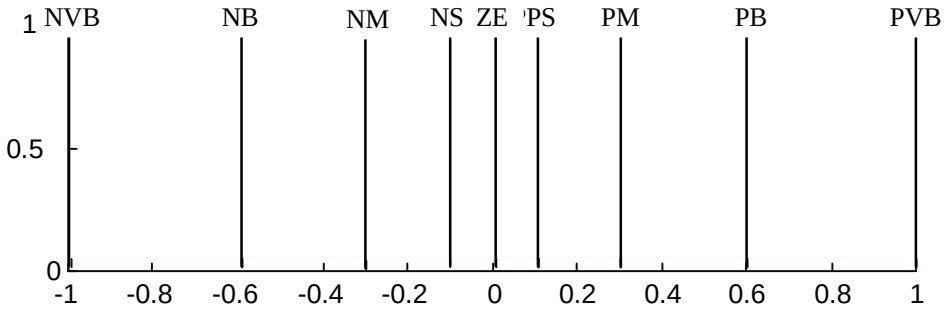


Figure III -12 Fonctions d'appartenance retenues pour la variable de sortie diq(k) (II)

Grâce à cette fonction d'appartenance, appelée «singleton », on tire profit du calcul de la variable de sortie. Dans ces conditions, la formule du centre de gravité se simplifie par :

$$diq_{res} = \frac{\sum_{i=1}^{m} \mu(diq_i) diq_i}{\sum_{i=1}^{m} \mu(diq_i)} \qquad \text{(III-3)}$$

m étant le nombre totale de règles.

Par rapport à la première approche, les règles sont aussi modifiées (figure III-12).

		e						
		NB	**NM**	**NS**	**ZE**	**PS**	**PM**	**PB**
de	**NB**	NVB	NVB	NVB	NB	NM	ZE	ZE
	NM	NVB	NVB	NB	NB	NM	ZE	PS
	NS	NVB	NB	NB	NM	PS	PB	PM
	ZE	NVB	NB	NM	ZE	PM	PB	PB
	PS	NVB	NB	NS	PM	PB	PB	PVB
	PM	NVB	ZE	PM	PB	PB	PVB	PVB
	PB	NVB	ZE	PM	PB	PVB	PVB	PVB

Figure III -12 Base des règles du régulateur modifié

I.8.1.2 Régulateur de position par la logique floue

Pour le régulateur de position on a essayé plusieurs approches, en utilisant comme variable d'entrée l'erreur entre la position réelle et la position de consigne et la variation temporelle de cette

erreur, comme pour la régulation de vitesse. Finalement, il était le plus simple, et avec le meilleur résultat, de choisir un régulateur avec une variable d'entrée et une variable de sortie, notamment l'erreur epos de la position pour l'entrée et la vitesse de référence Ω ref pour la sortie.

L'idée de ce régulateur est justifiée par les principes suivants :

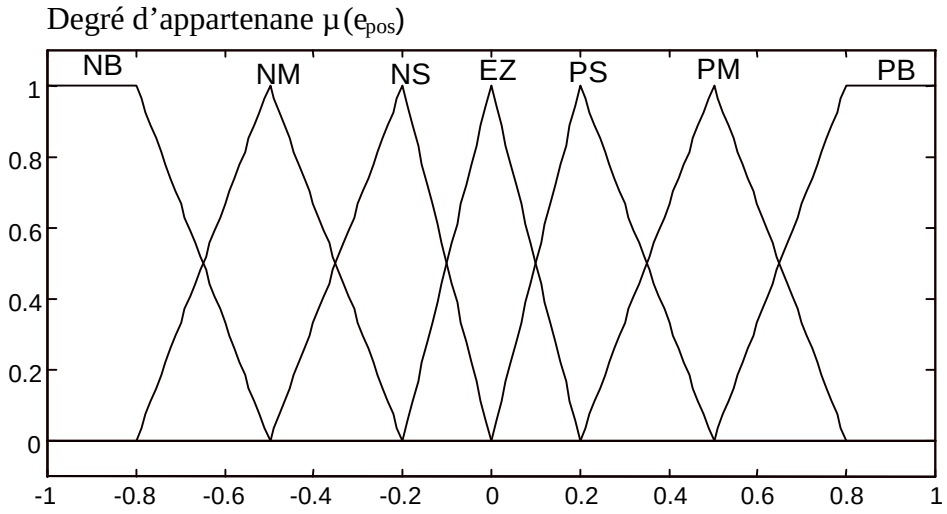
- Si l'erreur est nulle, la vitesse l'est aussi.
- Si l'erreur n'est pas nulle, il faut tourner très rapidement pour éliminer cette erreur.

Enfin, ce but peut être atteint par un couplage à trois états :
- vitesse maximale positive,
- vitesse maximale négative,
- vitesse nulle.

La figure (III-13) illustre les fonctions d'appartenance des deux variables :

On constate, que les sous-ensembles de Ω ref (sauf ZE) sont tous assez loin de zéro. En même temps, les sous-ensembles de l'erreur sont plutôt orientés vers zéro. Ces faits correspondent à la deuxième réflexion.

En conséquence, les règles pour la réalisation sont les suivantes :



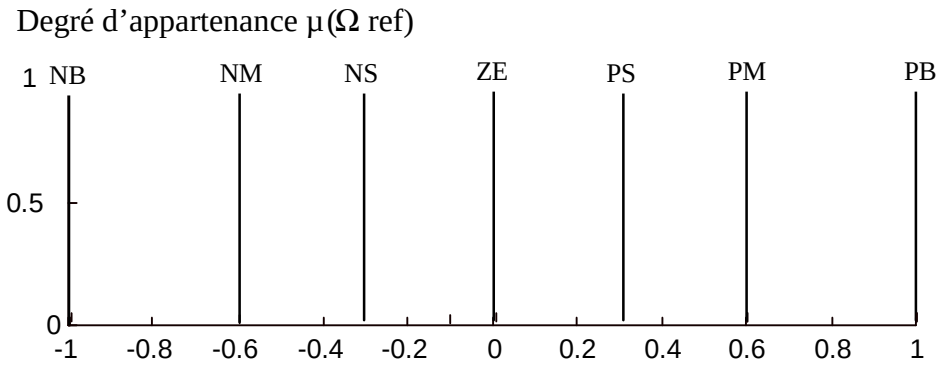


Figure III -13 Fonctions d'appartenance du régulateur de position

I.8.1.3 Résultats expérimentaux

Les régulateurs à logique floue sont réalisés à l'aide du Toolbox « Fuzzy Logic » de Simulink. La figure (III-14) montre l'éditeur de ce module avec ses différentes fenêtres.

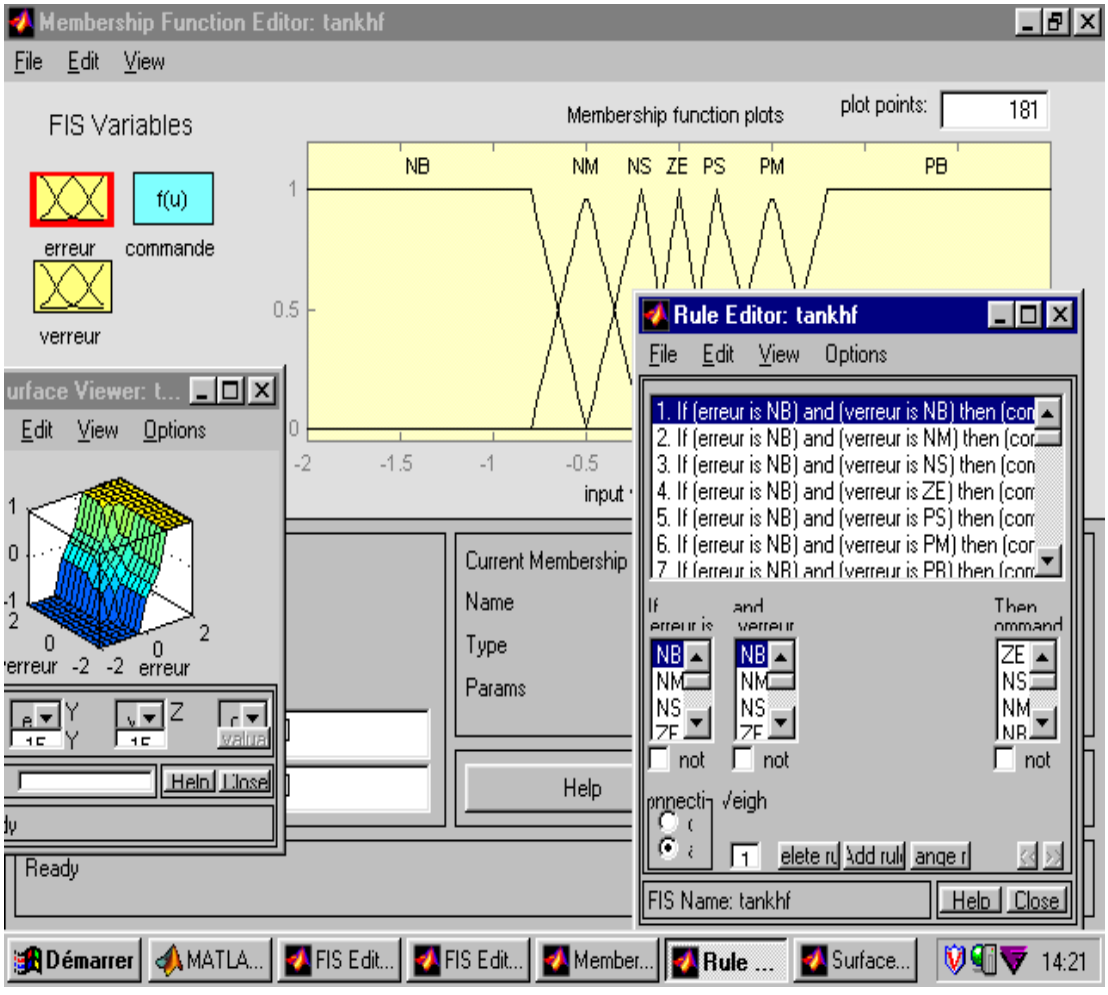


Figure III-14 Fenêtres de l'éditeur du Toolbox Fuzzy de Matlab

Une comparaison des résultats d'expérimentation de la commande de la machine synchrone à aimants permanents par des correcteurs classiques (PI et régulateur d'état) et un correcteur flou a été faite (figures III-15 et III-16). Compte tenu des résultats [11], le PI flou semble pourvoir remplacer le PI conventionnel pour améliorer les performances dynamiques de ce dernier.

Le PI Flou est très peu sensible aux variations des paramètres du système ainsi qu'aux perturbations externes ce qui justifie sa robustesse. Il permet d'obtenir des temps de montée très faibles par rapport au PI classique grâce aux larges domaines physiques de la variation de l'erreur et de la variation de commande.

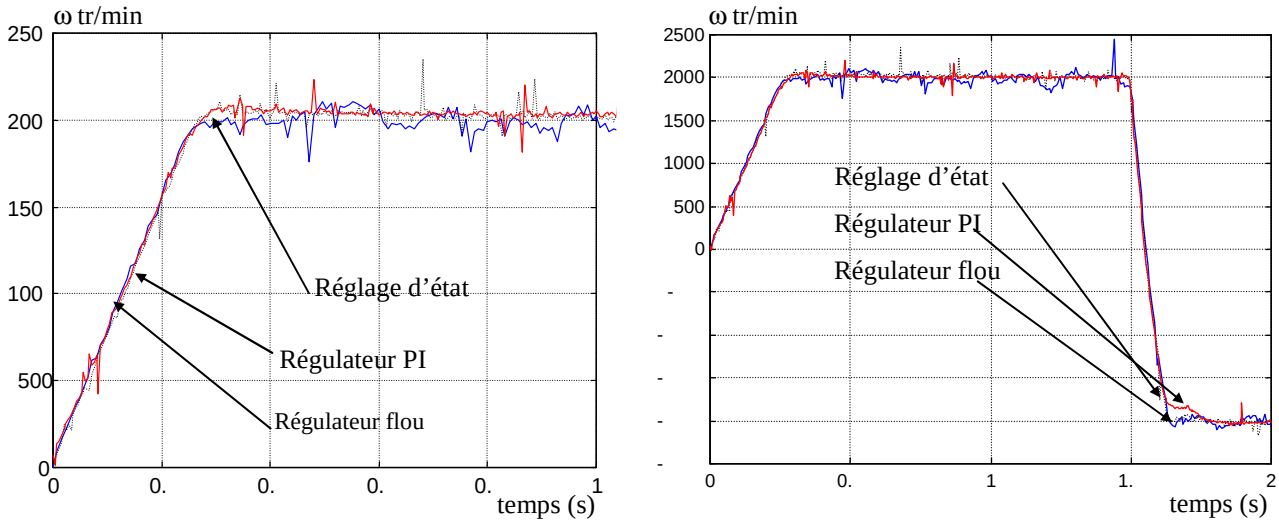


Figure III-15 : Réponses de Vitesse
(Régulateurs PI, d'Etat et Flou)

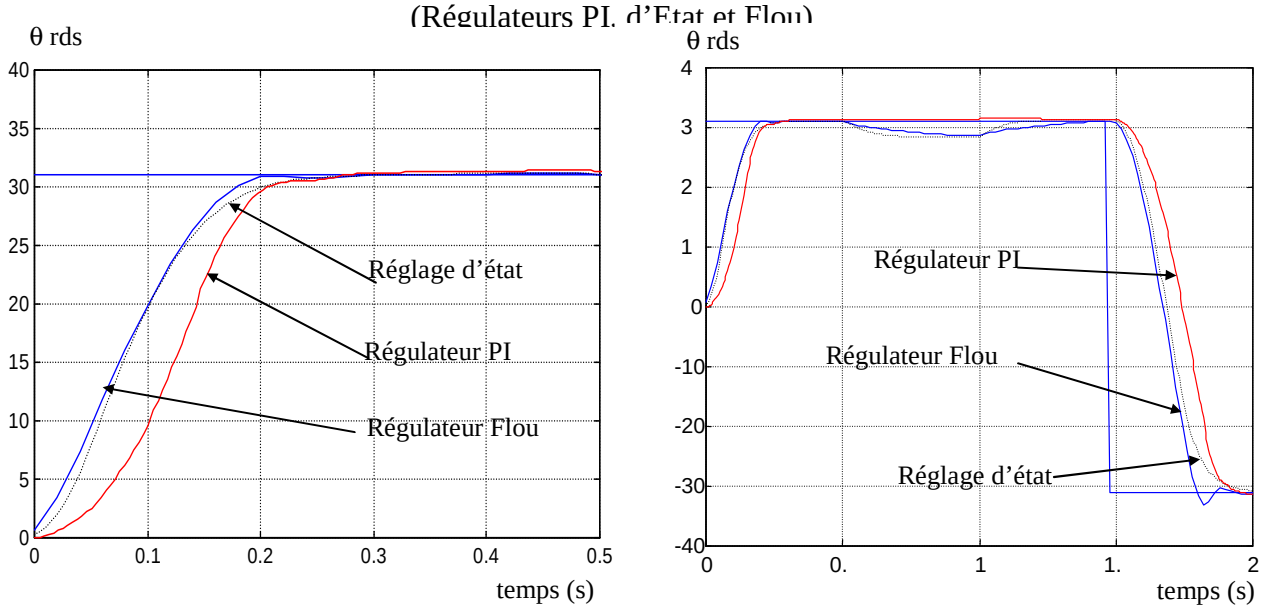


Figure III-16 ***:*** Réponses de Position
(Régulateurs PI, d'Etat et Flou)

I.8.2. Exemple 2 : Navigation réactive d'un robot mobile

Un robot mobile est souvent astreint à suivre une trajectoire de référence, trajectoire pouvant être :

- soit calculée dynamiquement en fonction des obstacles,
- soit matérialisée par une piste tracée sur le sol ou un câble enterré (cas de chariot filoguidé).

Dans le premier cas, le robot doit se localiser par rapport à une trajectoire (immatérielle) de référence ; dans le second, tout changement de trajectoire implique des interventions matérielles parfois lourdes.

Dans la navigation réactive, il n'y a pas de trajectoire de référence : le robot doit se déplacer en évitant de lui même les obstacles qu'il repère par un système de perception (radar, capteurs infra-rouges ou à ultrasons). Ses tâches sont souvent décomposées en évitement d'obstacles et recherche du but.

Pour réaliser la détection d'obstacles, nous avons choisi d'utiliser un ensemble de capteurs à ultrasons qui présentent des caractéristiques intéressantes tant d'un point de vue performance que du point de vue coût ou facilité d'implantation sur un robot mobile.

Le robot Pioneer II supporte une rangée de huit capteurs ultrasonores à travers huit transducteurs [9] placés en son avant. En option, il peut également être équipé d'une autre rangée de huit capteurs ultrasonores placés en son arrière.

Les sonars avants sont positionnés de la manière suivante : un de chaque coté et six de face, séparés de vingt degrés d'intervalle (figure III-17).

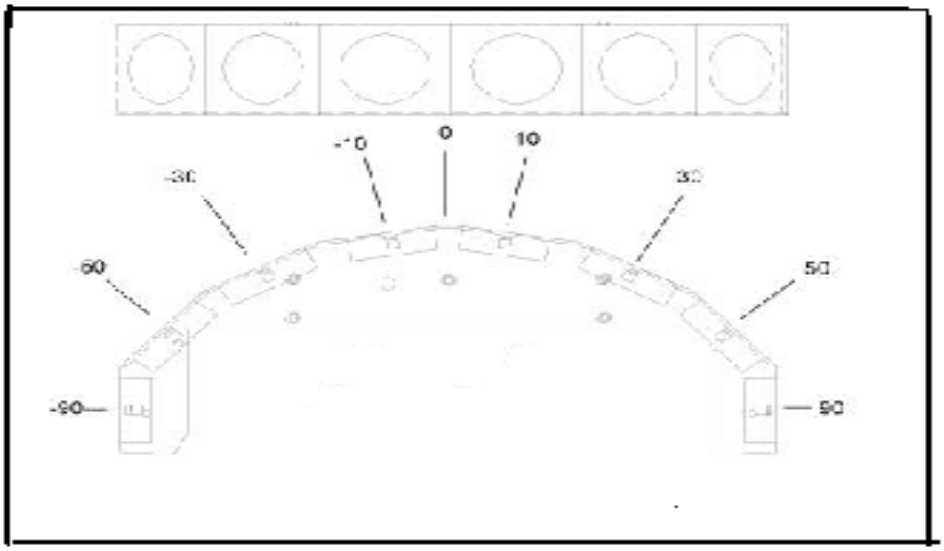


Figure III-17: La disposition des sonars avant du robot Pioneer II.

Le champ de mesure de ces capteurs est compris entre 10 cm de portée minimale à 5m de portée maximale.

Le choix de contrôleur flou pour la navigation d'un robot mobile s'est imposé naturellement :

- les règles de conduite sont génériques, valables pour tous les robots mobiles ;
- elles peuvent être individualisées par apprentissage, pour tenir compte des caractéristiques mécaniques et dynamiques des robots ;
- introduction de connaissances rend le robot immédiatement efficace, dés le début de l'apprentissage ; l'apprentissage sert alors à améliorer la conduite ;
- enfin, le comportement du robot est toujours prévisible puisque, pour une situation donnée, les règles appliquées sont interprétables.

De nombreux auteurs ont proposé des contrôleurs flous pour la navigation réactive, le nombre de règles variant de 8 à 271. Dans ce dernier cas, 243 règles étaient dédiées à l'évitement d'obstacles et 28 à la recherche de l'objectif. Un tel nombre de règles n'est pas très raisonnable : cela nuit à la lisibilité et complique inutilement l'apprentissage.

Dans nos simulations et les expérimentations menées sur un vrai robot (le robot Pioneer II), nous avons utilisé un contrôleur de navigation de type suivant :

- les entrées sont les distances dans les trois directions : gauche, frontale et droite : ces distances sont obtenues par balayage avec un système de perception formé de huit capteurs à ultrasons.
- les sorties sont les commandes de vitesse et de changement de direction du robot : ces deux commandes sont, le cas échéant, transformées en vitesses de rotation pour des roues motrices gauche et droite.

Les distances sont évaluées par rapport aux deux sous-ensembles flous « Proche » et « Loin », représentés à la figure (III-18.) Ces sous-ensembles flous sont définis par deux paramètres :

- "dm", désignant la distance minimum par rapport à un obstacle, cette distance pouvant varier avec la vitesse
- "ds", désignant la distance de sécurité, au delà de laquelle le robot peut circuler à vitesse élevée (ds = 800 mm).

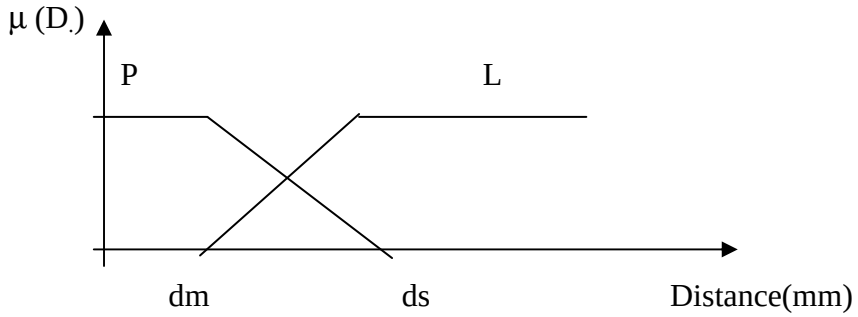


Figure III-18 : Sous-ensembles flous définis pour les variables distances (gauche, face, droite)

Les distances dans les trois directions (à gauche, en face, à droite) permettent de définir huit situations synoptiques de base, notées (Si) avec i=1 à 8) immédiatement interprétables et pour lesquelles la conduite à tenir découle du simple bon sens.

Ainsi, la situation S_1 (des obstacles proches dans les trois directions) correspond à un cul-de-sac : il faut s'arrêter et tourner fortement vers la droite ou la gauche. La situation S_2 correspond à un chemin libre à droite, donc le robot doit tourner à droite. La situation S_7 correspond à un obstacle sur la droite, qu'il faut éviter en tournant légèrement à gauche.

Les angles sont évalués par rapport au sens trigonométrique : positif vers la gauche et négatif sinon.

Ainsi, la politique consistant à se diriger directement vers le point d'arrivée en évitant les obstacles en longeant les murs en tenant sa droite s'exprime symboliquement par les huit règles floues suivantes :

SI	S1 : (P ,P, P)	**Alors**	V est ZR	**Et**	ψ est NG	
SI	S2 : (P ,P, L)	**Alors**	V est ZR	**Et**	ψ est NG	
SI	S3 : (P ,L, P)	**Alors**	V est Vmoy	**Et**	ψ est ZR	
SI	S4 : (P ,L, L)	**Alors**	V est Vmax	**Et**	ψ est NP	(III-5)
SI	S5 : (L ,P, P)	**Alors**	V est ZR	**Et**	ψ est PG	
SI	S6 : (L ,P, L)	**Alors**	V est ZR	**Et**	ψ est PG	
SI	S7 : (L ,L, P)	**Alors**	V est Vmax	**Et**	ψ est PP	
SI	S8 : (L ,L, L)	**Alors**	V est Vmax	**Et**	ψ est ψ (obj)	

Les labels utilisés sont : NG (Négatif-Grand), NP (Négatif-Petit), NM (Négatif-Moyen), ZR (zéro), PP (Positif-Petit), PG (Positif-Grand), Vmoy (Vitesse moyenne) et Vmax (Vitesse maximum).

Les sept premières règles sont utilisées pour l'évitement d'obstacle alors que la huitième règle permet d'atteindre l'objectif (ψ (obj) désigne l'angle qui sépare l'axe du robot et le point d'arrivée).

Les grandeurs de sorties sont la vitesse du robot et la direction qu'il doit prendre. Pour construire un contrôleur flou de type Takagi-Sugeno d'ordre 0, les valeurs symboliques des conclusions sont remplacées par des valeurs numériques Vitj et Phij. Les sorties inférées deviennent :

$$V = \sum \alpha_i . V_i \quad \text{et} \quad \psi = \sum \alpha_i . \psi i \quad \text{pour } i = 1 \text{ à } 8 \qquad \text{(III-6)}$$
$$\text{Car } \sum \alpha_i = 1$$

Où α_i représente la valeur de vérité de la règle i.

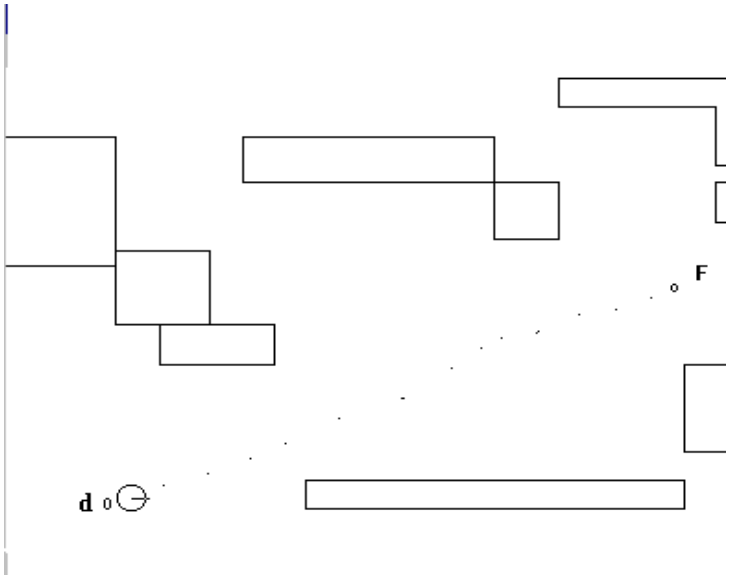
I.9. Résultats du simulateur graphique Pioneer II

Dans cette partie, nous présentons des résultats obtenus par l'interface graphique du simulateur du robot Pioneer II.

Niveau 0 : « Atteindre un point d'arrivée »

Nous rappelons que ce niveau de compétence permet au robot d'atteindre un point, et plus particulièrement le point d'arrivée, lorsque l'environnement est très peu contraint. La figure (III-18) propose des exemples de telles situations. Sur l'exemple (a) de cette figure, le point d'arrivée est situé à l'arrière du robot lorsque celui-ci se trouve dans sa position de départ.

(a)



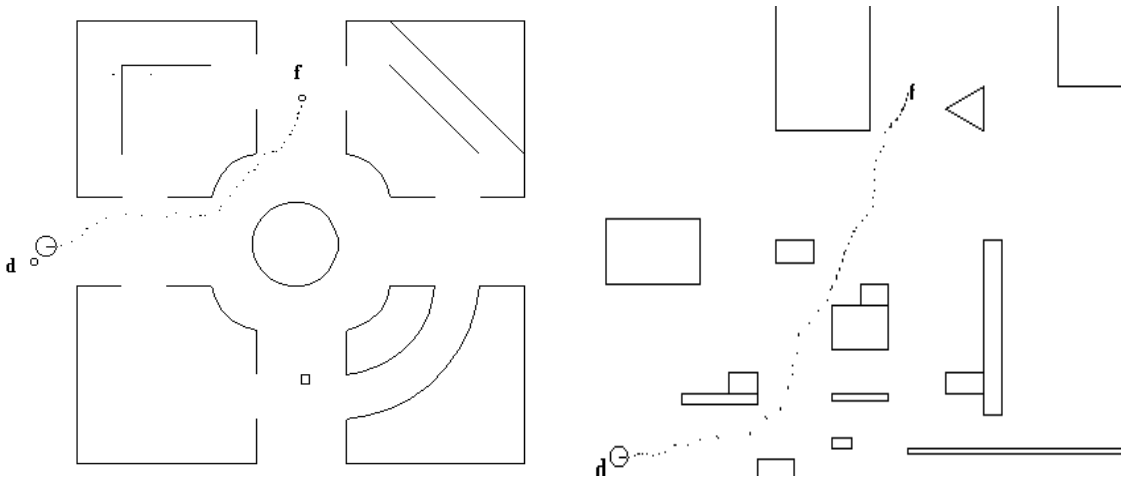
(b)

Figure III.18 : Exemples du niveau Atteindre un point d'arrivée.

Le robot fait demi-tour et s'oriente vers le point cible. Dans la figure (III-18-.b), le robot ne rencontrant aucun obstacle devant lui se dirige directement vers le point d'arrivée.

Niveau 1 : « Evitement des obstacles fixes » :

A ce niveau, les obstacles ne posent pas de limitations majeures à l'évolution du robot. L'opération consiste simplement à faire progresser le robot autour des différents objets situés dans son espace de travail en modifiant constamment la direction suivie par le robot. Des exemples de tels environnements ainsi que les trajectoires correspondantes du robot sont présentés sur la figure (III-19).

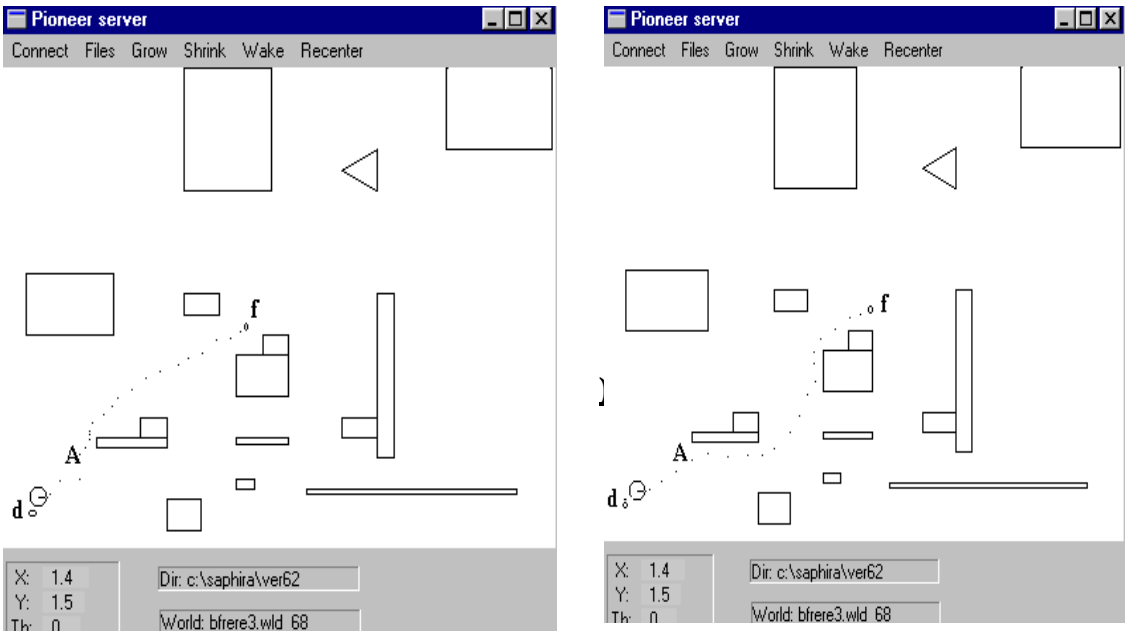


(a) **(b)**

Figure III.19 : Exemples d'évitement d'obstacles fixes

Les solutions proposées assurent la navigation du robot à partir d'informations fournies par les capteurs à ultrasons, l'environnement est à priori inconnu. Pour un même type d'environnements, de faibles variations dans les positions des points de départ et d'arrivée du robot peuvent engendrer d'importants écarts dans les prises de décision. La figure (III-20) illustre cet aspect où, pour un même environnement, deux positions voisines du point d'arrivée sont proposées.

A partir du point A, les trajectoires décrites par le robot s'en trouvent logiquement modifiées. Cette différence provient du choix du couloir libre retenu en en ce point. Dans la figure (III-20-a), le robot décide de suivre le couloir libre situé à gauche du point A. Par contre, le robot choisit de suivre le couloir libre situé à droite de cet obstacle dans la figure (III-20-b).



(a) **(b)**

Figure III-20: Influence de la position du point d'arrivée sur la trajectoire décrite.

Chapitre II

Commande et Identification par les Réseaux de Neurone Artificiels

II.1. Introduction

L'un des défis de l'homme aujourd'hui est de copier la nature et de reproduire des modes de raisonnement et de comportement qui lui sont propre. Les réseaux de neurones, sont nés de cette envie, ils constituent une famille de fonctions non linéaires paramétrées, utilisées dans de nombreux domaines (physique, chimie, biologie, finance, etc.), notamment pour la modélisation de processus et la synthèse de lois de commandes, Ce cours décrit une technique intelligente nouvellement introduite dans le monde de l'automatique. Il s'agit principalement des réseaux de neurones artificiels et les différentes structures qui leurs sont associées ainsi que nous abordons par la suite l'identification et le contrôle de processus par les réseaux de neurones pour la synthèse de lois de commandes.

II.2. Généralités

L'origine des réseaux de neurones vient de l'essai de modélisation mathématique du cerveau humain. Les premiers travaux datent de 1943 et sont l'œuvre de MM. Mac Culloch et Pitts. Ils supposent que l'impulsion nerveuse est le résultat d'un calcul simple effectué par chaque neurone et que la pensée née grâce à l'effet collectif d'un réseau de neurones interconnectés (figure I.1). Ils ont connu des débuts prometteurs vers la fin des années 50, mais le manque d'approfondissement de la théorie a gelé ces travaux jusqu'aux années 80.

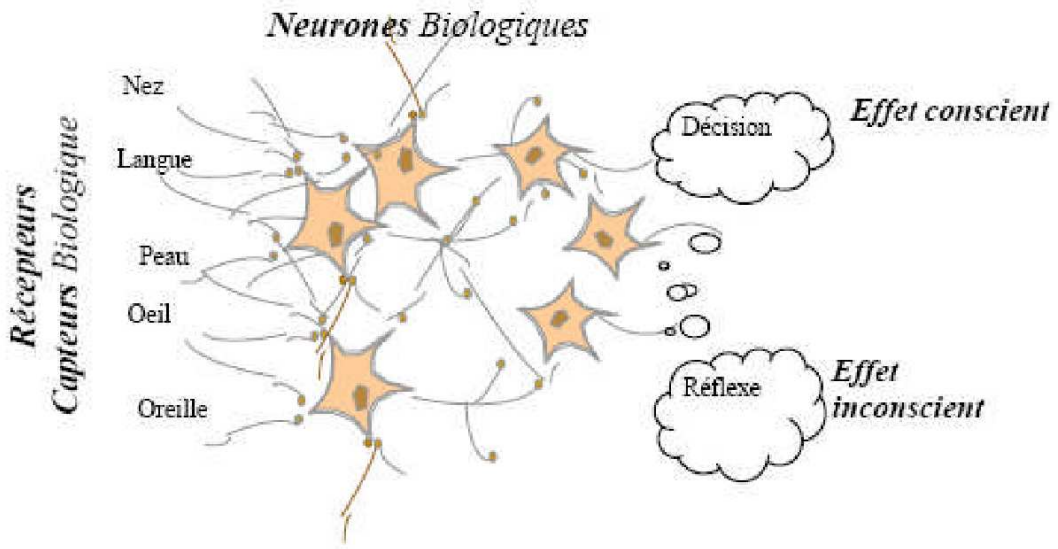


Figure II.1 Structure d'un réseau de neurones biologiques.

Les réseaux de neurones forment une famille de fonctions non linéaires, permettant de construire, par apprentissage, une très large classe de modèles et de contrôleurs. Un réseau de neurones est un système d'opérateurs non linéaires interconnectés, recevant des signaux de l'extérieur par ses entrées, et délivrant des signaux de sortie, qui sont en fait les activités de certains neurones.

II.3. Neurone biologique

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone (unique) pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de

quelques dizaines d'Angstroms (10- 9 m) entre l'axone du neurone afférent et les dendrites (on dit une dendrite) du neurone efférent. La jonction entre deux neurones est appelée la synapse (figure I.2).

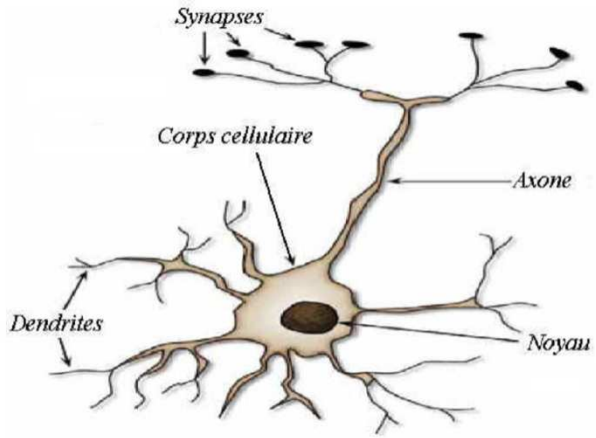


Figure II.2 Schéma simplifié d'un neurone biologique.

Les réseaux de neurones artificiels (Artificial Neural Networks – ANN) constituent une approche fondamentalement nouvelle dans le traitement de l'information. Ce sont des systèmes parallèles, adaptatifs et distribués dont le fonctionnement imite celui des réseaux de neurones biologiques tout en reproduisant leurs caractéristiques de base :

1. La connaissance est acquise par le réseau à travers un processus d'apprentissage ;
2. Les connexions entre neurones, appelées poids synaptiques, sont utilisées pour le stockage de la connaissance.

Du point de vue structurel, un réseau de neurones est d'un certain nombre d'unités de traitement simples appelées neurones formels ou artificiels. Ces derniers sont connectés entre eux de façon à produire la réponse correspondant aux entrées reçues par le réseau. Plusieurs modèles de neurones artificiels ont été développés, s'inspirant du principe de fonctionnement du neurone biologique qui assure essentiellement les fonctions suivantes :

- Réception des signaux provenant des neurones voisins ;
- Intégration de ces signaux ;
- Génération d'une réponse ;
- Transmission de celle-ci à d'autres neurones.

II.4. Eléments de base

II.4.1. Structure de base

Le premier modèle du neurone formel date des années quarante. Il été présenté par Mc Culloch et Pitts. S'inspirant de leurs travaux sur les neurones biologiques ils ont proposés le modèle suivant:

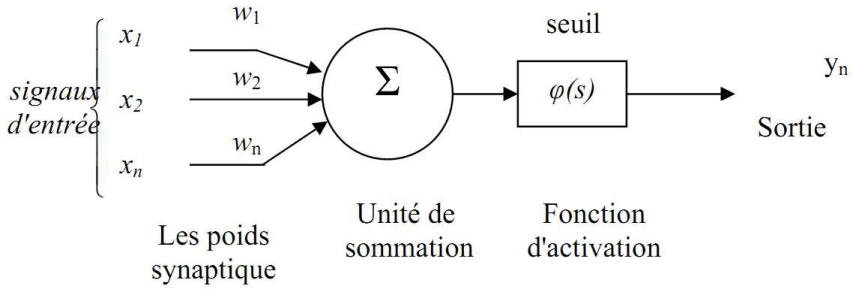


Figure II.3 le neurone formel

Un neurone formel est une fonction algébrique non linéaire et bornée, dont la valeur dépend des paramètres appelés poids synaptiques ou poids des connexions. D'une façon plus générale, un neurone formel est un élément de traitement (opérateur mathématique) possédant n entées (qui sont les neurones externes ou les sorties des autres neurones), et une seule sortie. Ce modèle est décrit mathématiquement par les équations suivantes:

$$s = \sum_{i=1}^{n} w_i x_i$$

$$y_n = \varphi(s)$$

Où x_i, w_i, φ et y_n sont respectivement, les entrées, les poids synaptiques, la fonction d'activation et la sortie du neurone.

II.4.2 Fonction d'activation

Les fonctions d'activations représentent généralement certaines formes de non linéarité. L'une des formes de non linéarité la plus simple, et qui est appropriée aux réseaux discret, est la fonction signe, figure II.3.a. Une autre variante de ce type des non linéarités est la fonction de Heaviside, figure II.3.b. Pour la majorité des algorithmes d'apprentissage il est nécessaire d'utiliser des fonctions sigmoïdes différentiables, telles que la fonction sigmoïde unipolaire, figure II.3.c et la fonction sigmoïde bipolaire, figure II.3.d. La classe, la plus utilisée des fonctions d'activation, dans le domaine de la modélisation et de la commande des systèmes non linéaires est la fonction sigmoïde bipolaire.

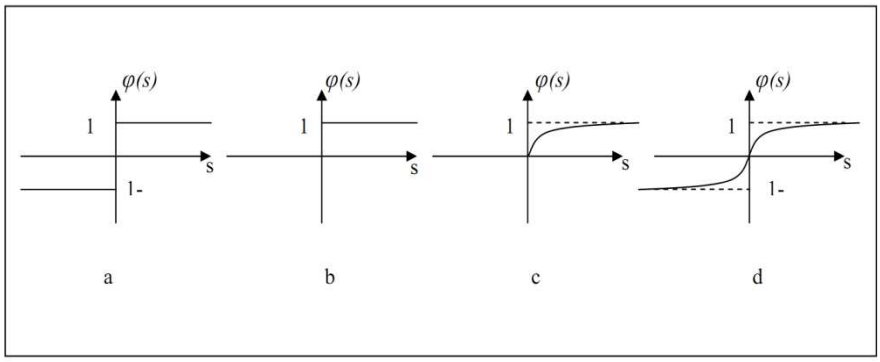


Figure II.4 Différentes fonctions d'activation

Donc un réseau de neurone est un ensemble d'éléments de traitement de l'information, avec une topologie spécifique d'interconnexions (architecteur du réseau) entre ces éléments et une loi d'apprentissage pour adapter les poids de connexions (poids synaptiques), il est caractérisé par un parallélisme à gain très fin et à forte connectivité. Nous entendons par là que dans un réseau de neurones donné, l'information est traitée par grand nombre de processeurs élémentaires très simples, chacun étant relié à d'autres processus. Ce processus très simple est un neurone formel désigné ainsi son fonctionnement s'inspire d'une modélisation des cellules biologiques. Ils sont dotés de deux propriétés importantes qui sont à l'origine de leur intérêt pratique des domaines très divers:

- Capacité d'adaptation ou d'apprentissage qui permet au réseau de tenir compte des nouvelles contraintes ou de nouvelles données du monde extérieur,

- Capacité de généralisation qui est son aptitude de donner une réponse satisfaisante à une entrée qui ne fait pas partie des exemples à partir desquels il apprit.

II.5. L'apprentissage des réseaux de neurones

L'information que peut acquérir un réseau de neurones est représentée dans les poids des connexions entre les neurones. L'apprentissage consiste donc à ajuster ces poids de telle façon que le réseau présente certains comportements désirés. En d'autres termes, l'apprentissage des réseaux de neurones consiste à ajuster les poids synaptiques de telle manière que les sorties du réseau soient aussi proches que possible des sorties désirées. Il existe deux types d'apprentissage:

• L'apprentissage supervisé: pour lequel on dispose de la sortie désirée et qui consiste à ajuster les poids synaptiques de telle sorte à minimiser l'écart entre la sortie désirée et la sortie du réseau,

• L'apprentissage non supervisé: pour lequel le réseau de neurones organise lui-même les entrées qui lui sont présentées de façon à optimiser un critère de performances donné,

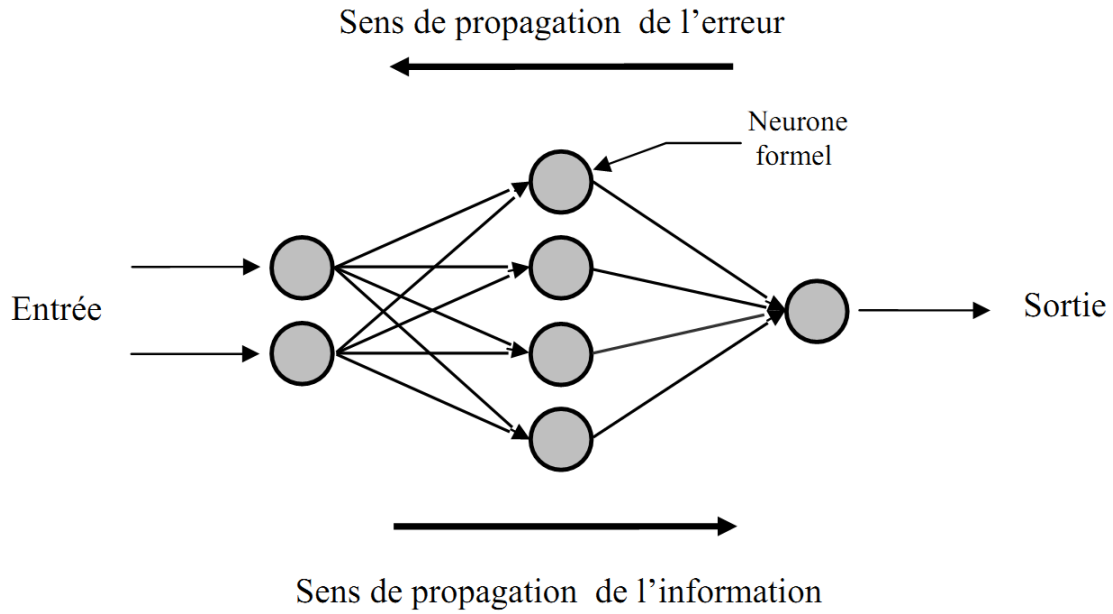


Figure II.5 Un réseau multicouche comportant 2 neurones d'entrée, 4 neurones cachés et un neurone de sortie

II.5 Algorithme d'apprentissage

L'apprentissage des réseaux de neurones, consiste à adapter les poids synaptiques de telle manière que l'erreur entre la sortie du réseau et la sortie désirée soit aussi petite que possible. La plupart des algorithmes d'apprentissage des réseaux de neurones est basée sur les méthodes du gradient: ils cherchent à minimiser un critère de la forme suivante:

$$J = \frac{1}{2}\sum_{q=1}^{m}\left(y_{rq}\left(X_p\right) - y_q^d\left(X_p\right)\right)^2$$

Où y_{rq} et y_q^d sont la sortie du réseau et la sortie désirée pour le vecteur d'entrée Xp. Cette optimisation se fait d'une manière itérative, en modifiant les poids en fonction du gradient de la fonction de coût selon la loi d'adaptation suivante:

$$w_{ji}^l(k+1) = w_{ji}^l(k) - \eta\frac{\partial J}{\partial w_{ji}^l}$$

Où w_{ji}^{l} est le poids de la connexion entre le $j^{ème}$ neurone de la couche *l* et le $i^{ème}$ neurone de la couche *l-1*

$i=1,\ldots,n+1$ la $i^{ème}$ composante du vecteur d'entrée,
$j=1,\ldots,m+1$ la $i^{ème}$ composante du vecteur de sortie,
$l=1,\ldots,L$ l'ordre d'une couche dans le réseau de neurone,

η est une constante positive ($\eta>0$) appelée taux d'apprentissage.
Le gradient est calculé par une méthode spécifique aux réseaux de neurones, dites méthode de rétro propagation. Dans cette méthode il est possible de calculer le gradient de la fonction coût en retropropageant l'erreur commise en sortie vers les couches cachées.

II.6 Identification et commande des systèmes par les réseaux de neurones

Par leur capacité d'approximation universelle, les réseaux de neurones sont bien adaptés pour l'identification et commande des systèmes non linéaires. En effet dans ce cas la fonction commande est une fonction non linéaire, l'objectif est alors d'approximer cette fonction par les RNA (réseaux de neurones artificiels). Cette approximation est réalisée par apprentissage des poids du réseau, l'apprentissage peut se faire hors ligne ou en ligne :

- Dans le cas de hors ligne, l'apprentissage est basé sur un ensemble de données définissant la fonction commande,
- Dans le cas d'en ligne, la mise à jour des poids est essentiellement adaptative.

II.6.1. Identification des processus par réseaux de neurones

Le principe de l'identification par réseau neuronaux consiste à substituer aux modèles paramétriques classiques des modèles neuronaux, c'est-à-dire proposer un modèle établissant une relation entre son entrée et sa sortie et à déterminer, à partir du couple des signaux d'entrée-sortie, le comportement du modèle. Deux raisons importantes nous motivent:

- Prédire le comportement d'un système pour différentes conditions de fonctionnement ;
- Elaborer une loi de commande à appliquer au processus, pour qu'il réalise l'objectif assigné.

Nous citerons deux techniques d'identification à base de réseaux de neurones multicouches : la méthode d'identification directe et la méthode d'identification inverse.

II.6.1.1. Identification directe

La figure II.6 montre le schéma général d'identification directe d'un processus. Sur cette figure, le réseau de neurones identificateur RNI est utilisé en parallèle avec un processus de type boite noire. La sortie du processus, *y*, est comparée avec la sortie du réseau de neurones, $\hat{y}$, puis l'erreur $y-\hat{y}$ est utilisée afin d'affiner les paramètres du système neuronal.

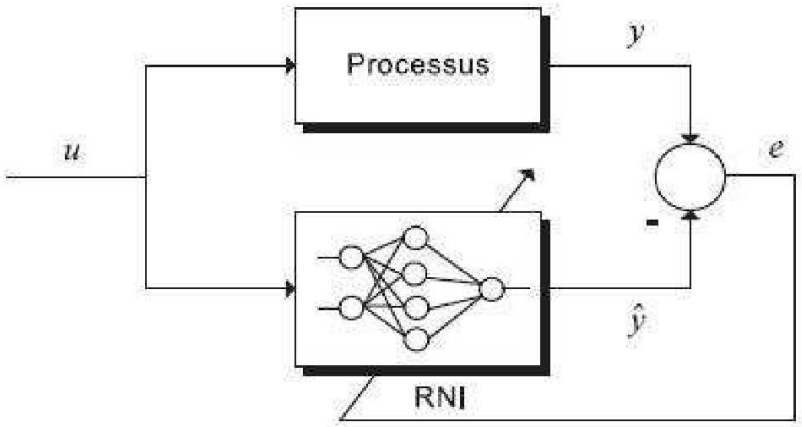


Figure II.6 Schéma d'identification directe d'un processus par réseau de neurones.

Considérons un système non linéaire avec une entrée *u(k)* et une sortie *y(k)*

y(k) peut dépendre de *u(k)* seulement, ou de *u(k)* et les états précédents de y et/ou *u* c'est à dire

y(k)=F(u(k))

Ou

y(k)=F(u(k-1),…,u(k-m),y(k-1),y(k-2),…,y(k-n))

Le principe de l'identification directe par réseau de neurones consiste à construire le réseau (en générale MLP) qui approxime la fonction F, cette approximation se fait par apprentissage hors ligne.
Etant donné la fonction F, il faut construire un RNA pour réaliser l'approximation, on génère une base de données de *N* échantillon *y(k-1), y(k-2),….,y(k-n), u(k-1),…,u(k-m)* (les entrés) et on obtient *y(k)* (la sortie) à partir de F. L'objectif de l'apprentissage est de déterminer la fonction F à partir de ces données. A l'instant t, on donne au RNA :
y(k-1), y(k-2),….,y(k-n), u(k-1),…,u(k-m) et on obtient à la sortie $\hat{y}(k)$. La mise à jour des poids du réseau est basée sur la minimisation de l'erreur entre *y(k)* et $\hat{y}(k)$.

$$J = \frac{1}{2}\sum_{i=1}^{m} y_i - \hat{y}_i$$

II.6.1.2. Identification inverse

Dans cette méthode, l'entrée du processus est comparée avec la sortie de l'identificateur neuronal RNI est la sortie du processus est injectée comme entrée du réseau de neurones (figure II.7). Après un apprentissage hors-ligne du modèle inverse, le RNI peut être configuré afin d'assurer un contrôle directe du processus.

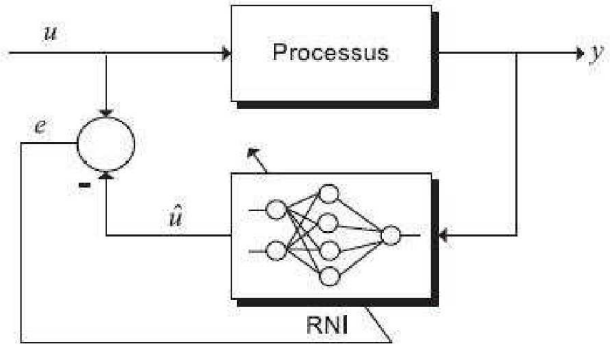


Figure II.7 Schéma d'identification inverse d'un processus avec un réseau de neurones.

I.6.2. Commande des processus par réseaux de neurones

La littérature scientifique fait mention de différentes architectures de commande. Les plus simples se basent sur l'apprentissage d'un contrôleur conventionnel déjà existant, d'autres opèrent un apprentissage hors-ligne du modèle inverse du processus ou d'un modèle de référence et enfin, d'autres travaillent complètement en ligne.

I.6.2.1. Apprentissage d'un contrôleur conventionnel

Un réseau de neurones peut reproduire le comportement d'un contrôleur conventionnel déjà existant (PI, PID, RST, ...) grâce à ses facultés d'apprentissage et d'approximation. Il suffit de le soumettre à un apprentissage hors ligne pendant une phase d'identification directe en considérant que le contrôleur est lui-même un processus. La figure II.7 montre le principe de l'identification directe d'un contrôleur conventionnel.

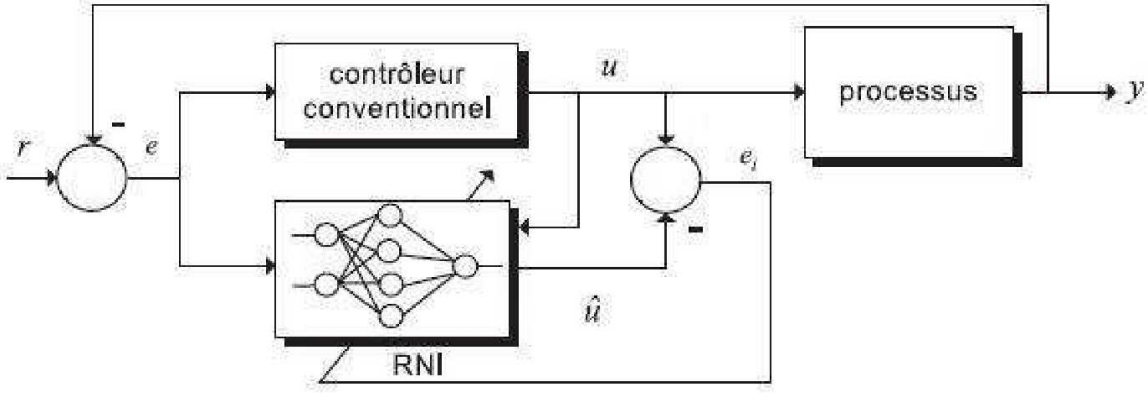


Figure II.7 Schéma d'identification directe d'un contrôleur conventionnel avec un RNI.

Le but de cette architecture n'est pas de perfectionner les performances du contrôleur conventionnel déjà existant, mais de s'affranchir des contraintes d'implémentations matérielles que peuvent nécessiter certains régulateurs. La méthode de régulation de type RST par exemple est reconnue pour ses bonnes performances en commande mais elle pose de sérieux problèmes en intégration numérique.

I.6.2.2. Commande inverse avec apprentissage en ligne

Le principe de cette commande repose sur une d'identification par modèle inverse. La figure II.8 représente le schéma de commande inverse avec un (RNC). Cette architecture reprend le même principe que celui de l'identification inverse montrée dans la figure II.8. En effet, L'entrée de référence *r* est comparée à la sortie y du processus pour former l'erreur de poursuite, $e = r - y$ qui sert

à modifier les paramètres du réseau en ligne. Après avoir appris le modèle inverse, le neuro-contrôleur délivre la sortie *u* du RNC qui est la commande injectée en entrée du processus, l'erreur est alors nulle et la sortie *y* est égale à la référence *r*. Ce principe est identique au RNI de la figure II.6 ou lorsque l'apprentissage du modèle inverse est accompli, la sortie du RNI est égale à l'entrée du processus. L'avantage de la commande inverse avec un RNC est le suivi en temps réel de l'évolution du processus, car l'apprentissage est réalisé en ligne.

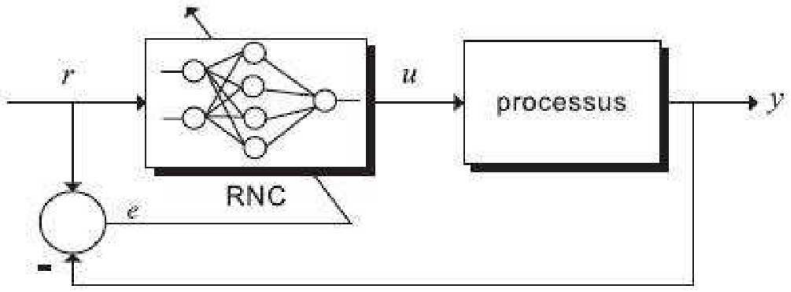


Figure II.8 Schéma de commande inverse avec un RNC.

Comme les performances de cet organe de commande dépendent étroitement de la fidélité du modèle inverse, la stabilité et le niveau de performance ne sera pas garanti dans le cas ou le modèle inverse n'existe pas ou s'il est difficile à trouver.

II.7. Avantages et Inconvénients des réseaux de neurones

II.7.1. Avantages des réseaux de neurones

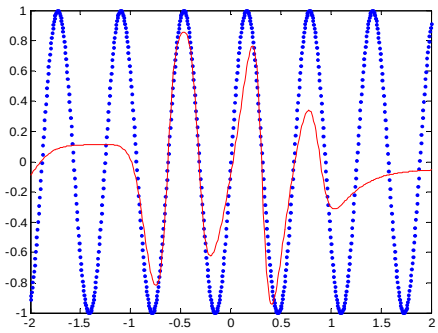
• Capacité de représenter n'importe quelle fonction, linéaire ou pas, simple ou complexe ;
• Faculté d'apprentissage à partir d'exemples représentatifs, par" rétropropagation des erreurs". L'apprentissage (ou construction du modèle) est automatique ;
• Résistance au bruit ou au manque de fiabilité des données ;
• Simple à manier, beaucoup moins de travail personnel à fournir que dans l'analyse statistique classique. Aucune compétence en matis, informatique statistique requise ;
• Comportement moins mauvais en cas de faible quantité de données ;
• Pour l'utilisateur novice, l'idée d'apprentissage est plus simple à comprendre que les complexités des statistiques multivariables.

II.7.2. Inconvénients des réseaux de neurones

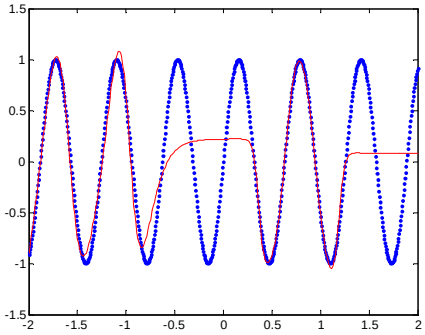
• L'absence de méthode systématique permettant de définir la meilleure topologie du réseau et le nombre de neurones à placer dans la (ou les) couche(s) cachée(s) ;
• Le choix des valeurs initiales des poids du réseau et le réglage du pas d'apprentissage, qui jouent un rôle important dans la vitesse de convergence ;
• Le problème du sur-apprentissage (apprentissage au détriment de la généralisation) ;
• La connaissance acquise par un réseau de neurone est codée par les valeurs des poids synaptiques, les réseaux de neurones sont donc des boites noires où les connaissances sont inintelligibles pour l'utilisateur.

Exemple 1 :

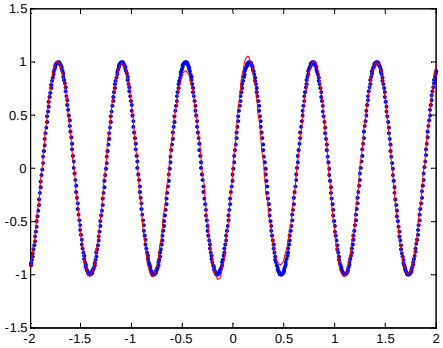
Le but de cet exemple est de mettre en place un réseau de neurones qui contient une couche cachée de 10 neurones et une couche de sortie de 1 neurone, on utilisant la rétropropagation du gradient avec une base d'apprentissage qui contient 70 point, pour élaborer un programme d'identification de la fonction $f(x)=\sin(x) \quad -2 \le x \le 2$.



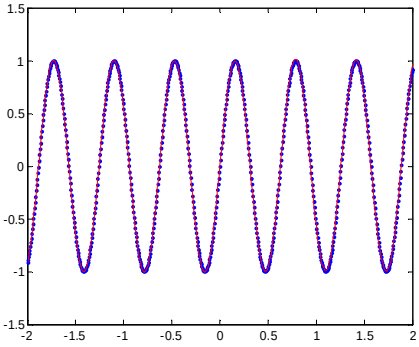
Apprentissage avec 5 itérations



Apprentissage avec 50 itérations



Apprentissage avec 100 itérations



Apprentissage avec 1000 itérations

Exemple 2 :

Soit le système discret donné par la fonction de transfert suivante :

$$H(z)=z^{-1}\frac{b_1+b_2 z^{-1}}{1-a_1 z^{-1}}$$

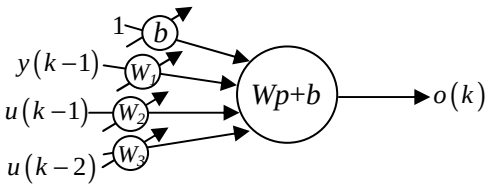
Avec

$b_1=1$, $b_2=-0.8$, $a_1=-0.5$

et l'équation de récurrence liant la sortie $y(k)$ à l'entrée $u(k)$

$$y(k)=a_1\, y(k-1)+b_1\, u(k-1)+b_2\, u(k-2)$$

Le but de cet exercice est l'identification des paramètres (b_1, b_2, a_1) on utilisant une base d'apprentissage qui contient 500 points d'entrés sortie ($u(k)$,$y(k)$) et un résaux de neurone exprimé par la figure suivante :



Les entrés du réseau $p=\left[\, y(k-1) \quad u(k-1) \quad u(k-2) \,\right]$

L'apprentissage du réseau consiste à modifier, a chaque pas les poids et les biais afin de minimiser la somme des carrés des erreurs en sortie.

A chaque pas d'apprentissage, l'erreur en sortie est calculée comme la différence entre la cible recherchée $y(k)$ et la sortie $o(k)$ du réseau.

La quantité à minimiser, a chaque pas d'apprentissage *k* est :

$$J=\frac{1}{2}e^T(k)e(k)=\frac{1}{2}\left(y(k)-o(k)\right)^T\left(y(k)-o(k)\right)$$

L'adaptation des poids se faisant dans le sens inverse du gradient, la matrice de poids de l'étape future (*t*+1) est :

$$W(t+1)=W(t)-\eta\frac{\partial J}{\partial W}$$

$$\frac{\partial J}{\partial W}=\frac{\partial J}{\partial o(k)}\frac{\partial o(k)}{\partial W}$$

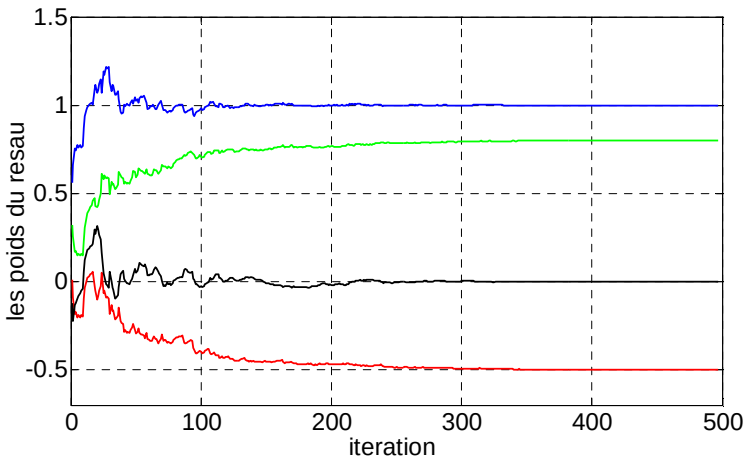
$$\frac{\partial J}{\partial o(k)}=o(k)-y(k)$$

$\frac{\partial o(k)}{\partial W}=\frac{\partial\left(W\ p(k)+b\right)}{\partial W}=p^T(k)$ Avec $p(k)$ désigne les entrés du réseau

$W(t+1)=W(t)+\eta\left(y(k)-o(k)\right)p^T(k)$ Avec η désigne les entrés du réseau

De même, on obtient l'expression de la modification du biais :

$$b(t+1)=b(t)+\eta\left(y(k)-o(k)\right)$$



On remarque que les poids du réseau convergent vers les paramètres (b_1, b_2, a_1) du système

Chapitre III

Commande par les Réseaux de Neurone Floue

III.1 Réseaux neuro-flou

Jusqu'aux années 90, ces deux techniques (neuronal et flou) apparaissaient comme des approches bien distinctes, ayant chacune leurs avantages et leurs inconvénients et leurs domaines spécifiques.

Dans le domaine du contrôle des processus, la commande floue se prêtait bien à un interfaçage avec un opérateur humain en raison de sa capacité à traiter des termes linguistiques. Par contre, il subsistait toujours une certaine part d'arbitraire dans le choix des fonctions d'appartenance et d'imprécision dans l'écriture des règles qui conduisait à une solution qui peut être assez loin de l'optimum.

D'un autre côté, les réseaux neuromimétiques (réseaux multicouches) étaient tout indiqués dans les cas où l'on ne dispose que de données numériques pour élaborer la commande. Ces réseaux apprenaient par l'exemple en minimisant une fonction coût ce qui leur conférait de bonnes qualités de généralisation. Par contre, il était très difficile d'y introduire de la connaissance *a priori* car le contenu du réseau n'était pas interprétable (boîte noire).

Elles présentent également des points communs :

- Les deux méthodes, appliquées au contrôle, présentent de bonnes qualités de robustesse et sont capables d'extrapolation et de généralisation.

- Aucune des deux méthodes ne nécessite de modèle mathématique du système à contrôler et elles peuvent donc toutes deux s'appliquer à des systèmes non linéaires sans complications particulières.

L'idée est donc apparue tout naturellement au début des années 90 de créer un système d'inférence floue optimisé, cherchant à utiliser les méthodes d'apprentissage supervisé, comme par exemple la rétropropagation du gradient, pour optimiser automatiquement certains paramètres d'un système d'inférence floue.

Dans ce qui suit, nous allons présenter trois architectures neuro-flou qui permettent d'optimiser un système d'inférence flou : l'architecture Nomura, l'architecture LSC et l'architecture ANFIS.

III.2 L'architecture Nomura

L'architecture Nomura est une méthode d'autoparamétrage d'un système d'inférence flou dans laquelle est abordé le concept " neuro-flou". Elle fut présentée par les scientifiques japonais Hiroyoshi Nomura [Nomura 91], Isao Hayashi et Noburu Wakami.

Contrairement au perceptron multicouche standard, le réseau n'est pas entièrement connecté et chaque couche du réseau a une fonction particulière, à laquelle correspond une fonction d'activation spécifique.

Ce réseau présente une complète analogie structurelle avec un système d'inférence floue de type "Sugeno" d'ordre zéro (i.e. dans lequel les conclusions des règles sont nettes). Les fonctions d'appartenance sont triangulaires ce qui a conduit au développement d'un algorithme spécifique.

Nous raisonnerons sur un exemple (qui peut facilement se généraliser) de système d'inférence floue à deux entrées e et Δe dont la connaissance s'exprime sous la forme de règles floues de type :

Règle i : Si e est Ai1 et Δe est Ai2 alors u est Wi

Dans lesquelles les conclusions sont des valeurs exactes (non floues), notées Wi. La prise de décision est faite à l'issue des étapes suivantes:

Pour une entrée donnée $(e, \Delta e)$, on calcule les différents degrés d'appartenance aux sous-ensembles flous Ai1 ou Ai2 apparaissant dans la partie prémisse des règles. C'est l'étape de fuzzyfication.

Pour chaque règle, on calcule la valeur de vérité α_i telle que:

α_i =ET((x1 est A1),(x2 est A2))

Enfin, il faut agréger les règles et tirer la conclusion. Les valeurs Wi des parties conclusion des règles sont pondérées par les valeurs de vérité des prémisses:

$$y = \frac{\sum_i \alpha_i \times W_i}{\sum_i \alpha_i} \qquad \text{(III.1)}$$

Chacune de ces étapes sera réalisée par les neurones d'une des couches du système d'inférence floue (figure II.1).

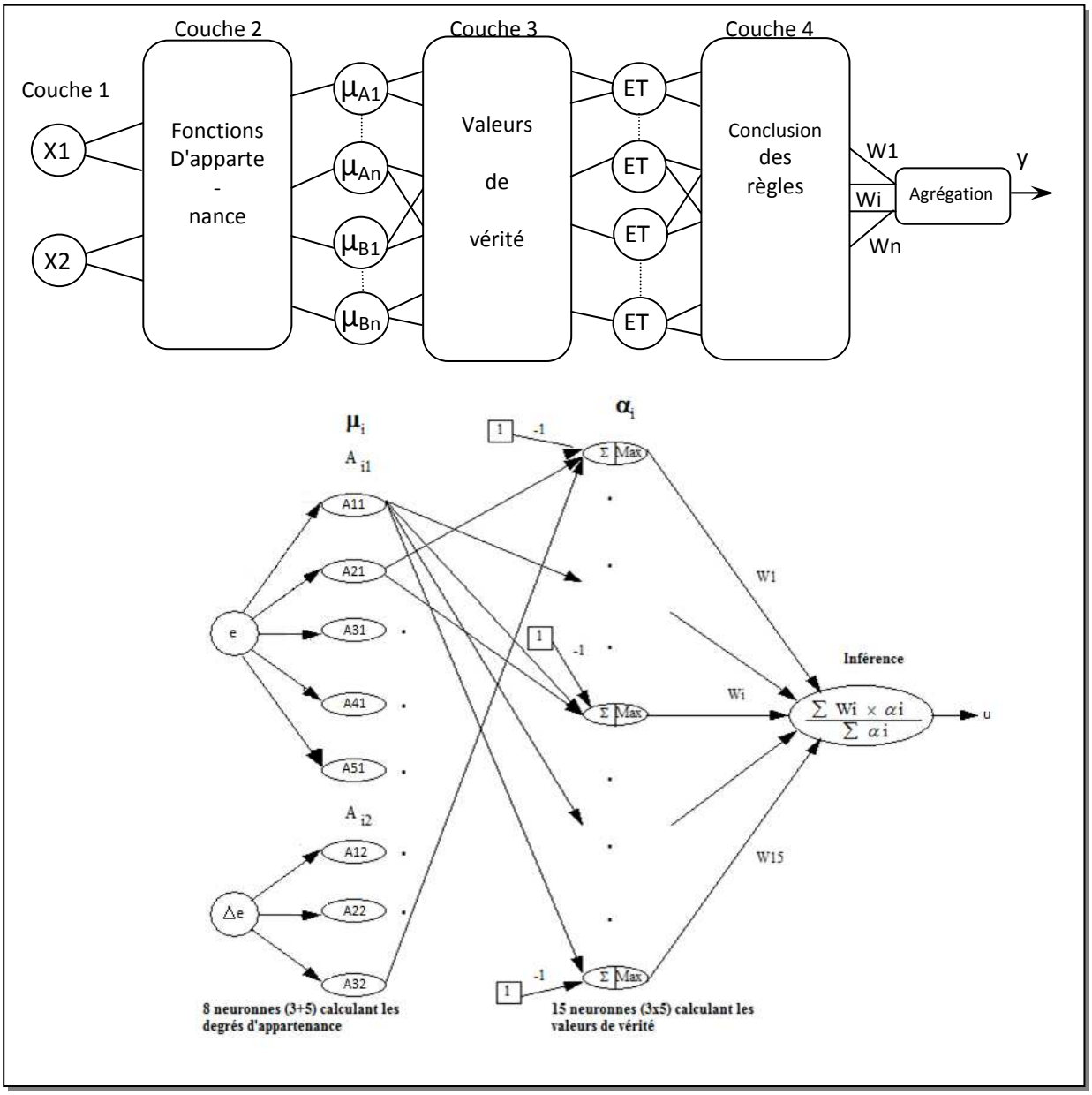


Figure III.1 Représentation du système d'inférence floue

III.2.1 Fonctions des différentes couches

- La première reçoit les entrées du réseau.
- La deuxième couche calcule les degrés d'appartenance des entrées aux diverses fonctions d'appartenance.
- La troisième couche calcule les valeurs de vérité à l'aide des poids entre les deux couches cachées (ils définissent l'opérateur ET choisi).
- La quatrième couche est la couche de sortie. Elle fait l'agrégation des différentes règles et calcule la valeur de sortie. Les poids W_i du réseau entre la troisième et la dernière couche constituent la partie conclusion des règles.

III.2.2 Réalisation des fonctions d'appartenance

Les fonctions d'appartenance de la prémisse proposées par Nomura sont de type triangulaire caractérisées par leur centre a_{ij} pour positionner la fonction sur l'univers du discours, et la largeur b_{ij} pour caractériser son étendue sur cet univers (figure III.6).

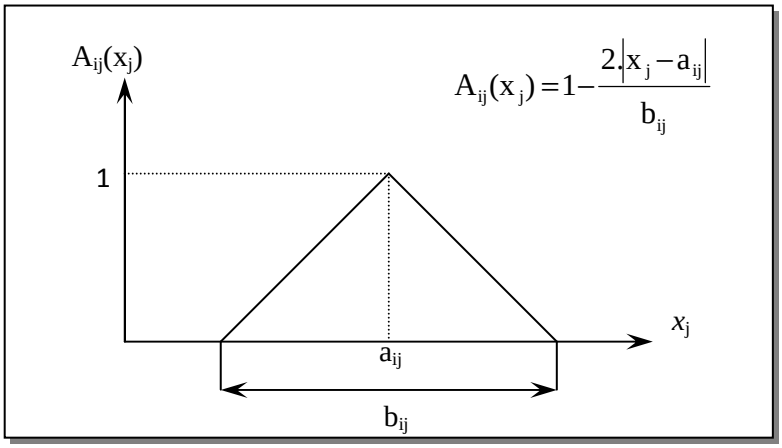


Figure III.2 Fonction d'appartenance utilisée

III.2.3 Calcul des valeurs de vérité α_i

La conjonction floue utilisée dans le réseau de Nomura est celle définie par Lukasiewicz comme suit :

$$ET(X,Y)= \max [0,\ \mu A(x)+\mu B(x)-1] \qquad \text{(III.2)}$$

III.2.4 Algorithme d'apprentissage

C'est une méthode extrêmement générale basée sur la descente du gradient, où chaque paramètre T_i est modifié par rétropropagation en fonction de sa part dans l'erreur globale E :

$$E = \frac{1}{2}(y - y_r)^2 \qquad \text{(III.3)}$$

$$T_i(k+1) = T_i(k) - \eta\ (\partial E / \partial T_i) \qquad \text{(III.4)}$$

$$a_{ij}(t+1) = a_{ij}(t) - K_a \frac{\partial(E)}{\partial(a_{ij})} \qquad \text{(III.5)}$$

$$b_{ij}(t+1) = b_{ij}(t) - K_b \frac{\partial(E)}{\partial(b_{ij})} \qquad \text{(III.6)}$$

$$W_i(t+1) = W_i(t) - K_w \frac{\partial(E)}{\partial(W_i)} \qquad \text{(III.7)}$$

Après dérivation on obtient :

$$a_{ij}(t+1) = a_{ij}(t) - \frac{K_a.\mu_i}{\sum_{i=1}^{n}\mu_i}.(y - y_r) - (W_i(t) - y).\text{sgn}(x_j - a_{ij}(t)).\frac{2}{b_{ij}(t).A_{ij}(x_j)} \quad \text{(III.8)}$$

$$b_{ij}(t+1) = b_{ij}(t) - \frac{K_b.\mu_i}{\sum_{i=1}^{n}\mu_i}.(y - y_r) - (W_i(t) - y).\frac{1 - A_{ij}(x_j)}{A_{ij}(x_j)}.\frac{1}{b_{ij}(t)} \quad \text{(III.9)}$$

$$W_i(t+1) = W_i(t) - \frac{K_w.\alpha_i}{\sum_{i=1}^{n}\mu_i}.(y - y_r) \quad \text{(III.10)}$$

Il y'a d'autre travaux qui utilise le produit pour le calcule le degré de vérité α_i dans l'architecture de Nomura, soit le contrôleur neuro-floue qui contient deux entré (e, Δe) avec deux fonction d'appartenance de type triangle pour chaque entré, et deux sous ensemble floue (small, Large)

Régle R1: If e est Small and Δe is Small then y is b_1 ,
Régle R2: If e est Small and Δe is Large then y is b_2 ,
Régle R3: If e est Large and Δe is Small then y is b_3 ,
Régle R4: If e est Large and Δe is Large then y is b_4 ,

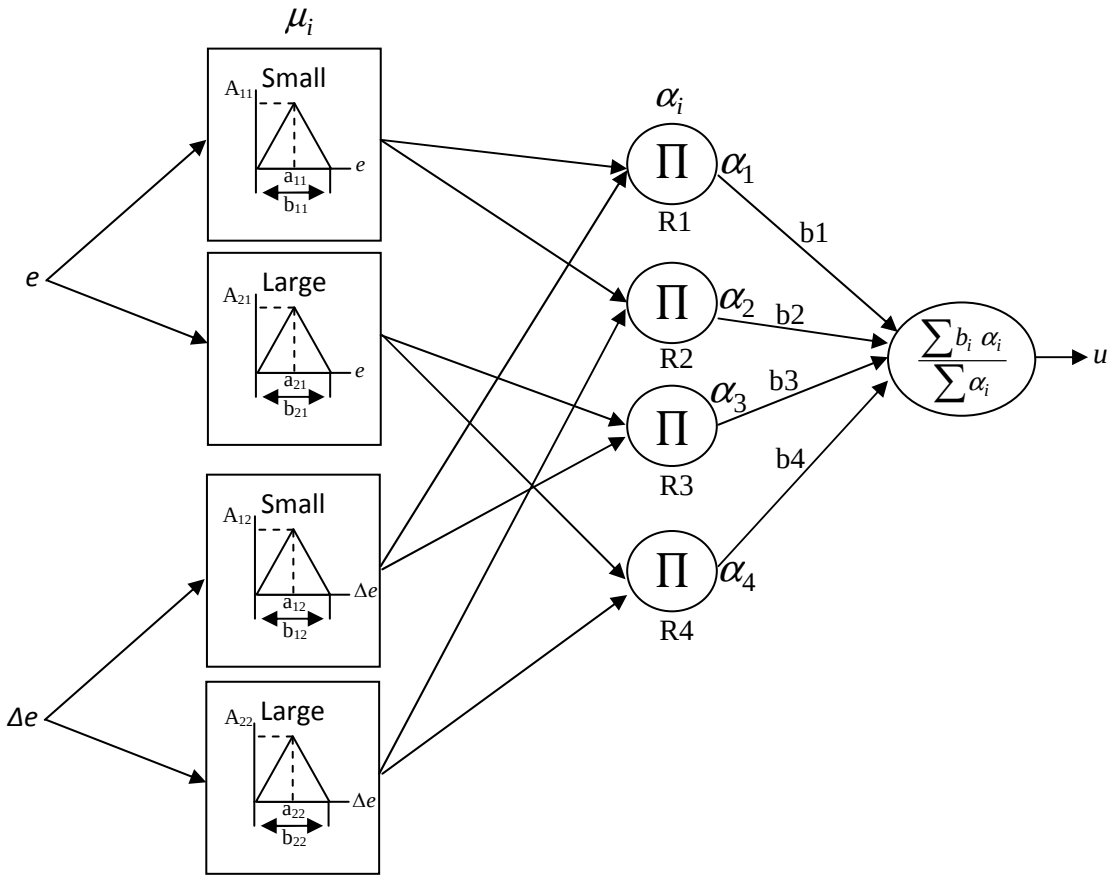


Figure III.3 Représentation du système d'inférence floue

L'adaptation des paramètres des fonctions d'appartenances des prémisses et des conclusions est donnée par les lois suivantes :

$$a_{ij}(t+1) = a_{ij}(t) - K_a \frac{\partial(E)}{\partial(a_{ij})} \quad \text{(III.5)}$$

$$b_{ij}(t+1) = b_{ij}(t) - K_b \frac{\partial(E)}{\partial(b_{ij})} \quad \text{(III.6)}$$

$$b_i(t+1) = b_i(t) - K_w \frac{\partial(E)}{\partial(W_i)} \quad \text{(III.7)}$$

III.3 L'architecture LSC

L'architecture LSC vient du nom du laboratoire Systèmes Complexes de l'Université d'Evry Val d'Essonne qui est à l'origine de sa proposition [Benreguieg 97]. Elle ressemble à l'architecture Nomura, la différence subsiste au niveau de l'algorithme d'apprentissage qui est effectué entièrement en ligne en minimisant une fonction coût *E* pour générer les paramètres w_i de la partie conclusion des règles et les ajuster. Les fonctions d'appartenance de la prémisse proposées par cette architecture sont de type gausienne définies par la relation (III.8):

$$\mu_{ij}(x_i) = \exp\left\{\frac{-(x_i - c_{ij})^2}{2\sigma_{ij}}\right\} \quad \text{(III.8)}$$

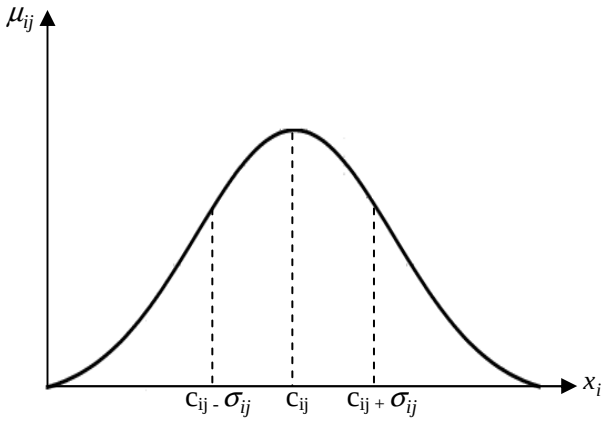


Figure III.4 Représentation du système d'inférence floue

Soit le contrôleur neuro-floue qui contient deux entré (*e*, Δ*e*) avec trois fonction d'appartenance de type gaussienne pour chaque entré, et neuf fonction d'appartenance de type singleton en sortie l'architecture du réseau et présenté par la figure suivante :

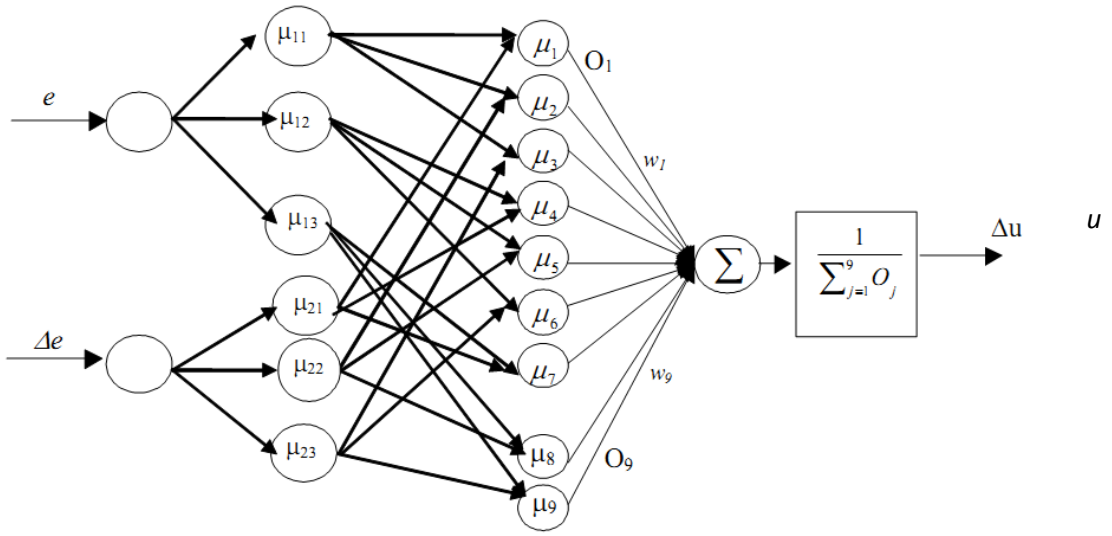


Figure III.5 Représentation du système d'inférence floue

L'apprentissage est basé sur la méthode de la descente du gradient, où chaque paramètre w_i est modifié par rétropropagation en fonction de sa part dans l'erreur globale E :

$$E = \frac{1}{2}(y - y_r) \qquad \text{(III.9)}$$

$$W_i(t+1) = W_i(t) - \frac{K_w.O_i}{\sum_{k=1}^{9} O_k}.(y - y_r) \qquad i = 1,2,....,9 \qquad \text{(III.10)}$$

III.4 L'architecture ANFIS

Cette architecture appartient à une classe de réseaux adaptatifs RBF [Jang 93a] fonctionnellement équivalent à un système d'inférence floue de type Sugeno, connue sous le nom de ANFIS (Adaptive-Network-based Fuzzy Inference System) [Jang93b]. Cette technique applique la méthode des moindres carrée combinée à la rétropropagation du gradient dans l'apprentissage.

Pour simplifier la compréhension et sans perte de généralité, nous considérons un système à deux entrées x_1 et x_2 et une sortie y. Considérons aussi un modèle flou de TS1 de ce système, composé des deux règles suivantes :

Si x_1 est A_1 **et** x_2 est B_1 **alors** $y_1 = F_1(x_1,x_2) = a_1x_1+b_1x_2+c_1$ (III.11)

Si x_1 est A_2 **et** x_2 est B_2 **alors** $y_2 = F_2(x_1,x_2) = a_2x_1+b_2x_2+c_2$ (III.12)

Jang a proposé de représenter cette base de règles par le réseau adaptatif de la figure (III.7)

La méthode ANFIS est basée sur l'utilisation de réseaux multicouches (adaptative networks) où chaque cellule réalise une fonction propre respectant les paramètres qui lui ont

été fournis. L'application de l'exemple précédent peut donc être représentée de la manière suivante (figure III.6):

Le réseau adaptatif ANFIS est un réseau multi-couches dont les connexions ne sont pas pondérées, ou ont toutes un poids de 1. Les nœuds sont de deux types différents selon leur fonctionnalité : les nœuds carrés (adaptatifs) contiennent des paramètres, et les nœuds circulaires (fixes) n'ont pas de paramètres. Toutefois chaque nœud (carré ou circulaire) applique une fonction sur ses signaux d'entrées. La sortie O_i^k du nœud i de la couche k (appelé nœud (i,k)) dépend des signaux provenant de la couche $k–1$ et des paramètres du nœud (i,k), c'est à dire,

$$O_i^k = f(O_1^{k-1}, \ldots, O_{n_{k-1}}^{k-1}, a, b, c, \ldots) \qquad \text{(III.13)}$$

où n_{k-1} est le nombre de nœuds dans la couche $k-1$, et $a, b, c, \ldots$ sont les paramètres du nœud (i,k). Pour un nœud circulaire, ces paramètres n'existent pas.

Dans le réseau de la figure (III.6), les nœuds d'une même couche ont des fonctions issues d'une même famille que nous explicitons ci-dessus :

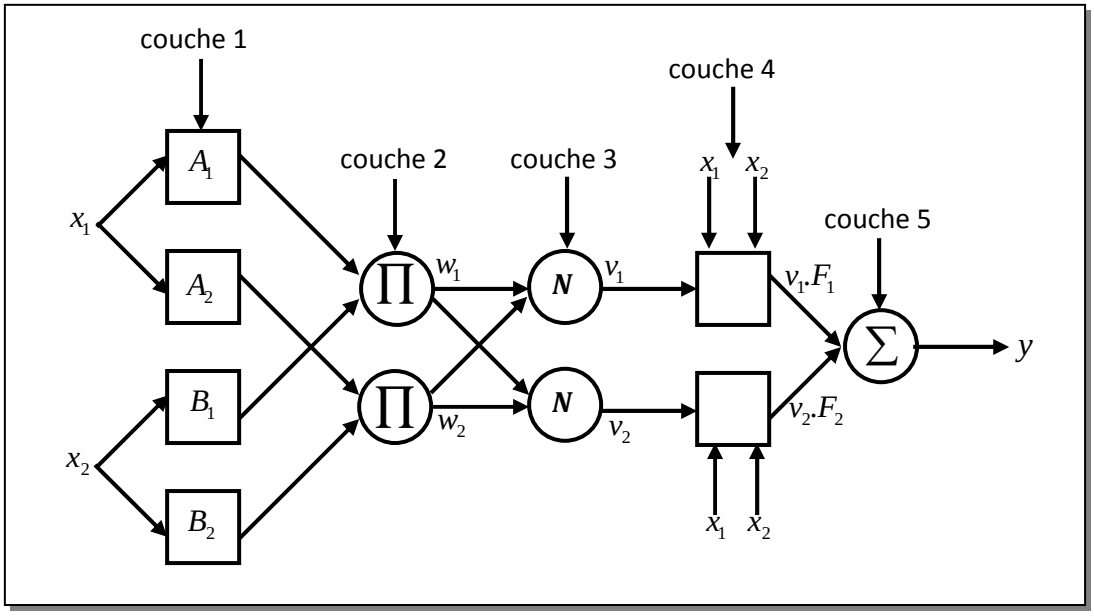


Figure III.6 Réseau ANFIS équivalent au raisonnement flou cité

Couche 1 : Chaque nœud de cette couche est un nœud carré avec une fonction :

$$O_i^1 = \mu_{A_i}(x) \qquad \text{(III.14)}$$

où x est l'entrée du nœud i, et A_i le terme linguistique associé à sa fonction. Donc le résultat que donne cette cellule représente le degré avec lequel x satisfait le qualificatif A_i .En d'autres

termes, O_i^1 est le degré d'appartenance de x à A_i. Les paramètres d'un nœud de cette couche sont ceux de la fonction d'appartenance correspondante. Généralement les fonctions d'appartenance utilisées par la méthode ANFIS sont des gaussiennes ou des fonctions cloches.

Couche 2 : Chaque nœud i de cette couche est un nœud circulaire appelé Π qui engendre en sortie le produit de ses entrées. Ce produit représente le degré d'activation d'une règle :

$$w_i = \mu_{A_i}(x_1).\mu_{B_i}(x_2),\ i=1..2 \qquad \text{(III.15)}$$

Couche 3 : Chaque nœud de cette couche est un nœud circulaire appelé N.

Dans cette couche, les cellules calculent le rapport entre les valeurs de vérité de chaque règle par rapport à la somme de toutes les valeurs de vérité données par chaque règle. Elles estiment donc le poids "normalisé" de chaque règle.

$$v_i = \frac{w_i}{w_1 + w_2} \qquad \text{(III.16)}$$

Couche 4 : Chaque nœud de cette couche est un nœud carré avec une fonction réalisant le calcul :

$$O_i^4 = v_i.F_i = v_i\,(a_i x_1 + b_i x_2 + c_i),\ \ i=1..2 \qquad \text{(III.17)}$$

où v_i est la sortie de la couche 3, et $\{a_i, b_i, c_i\}$ est l'ensemble des paramètres de sortie de la règle i.

Couche 5 : Le seul nœud de cette couche est un nœud circulaire qui effectue la somme des signaux provenant de la couche 4, c'est à dire,

$$O_1^5 = y = \sum_i v_i.F_i \qquad \text{(III.18)}$$

Nous remarquons que la sortie globale du réseau est équivalente à la sotie du modèle de TS1.

La généralisation du réseau à un système à n entrées ne pose aucun problème particulier. Le Nombre de nœuds de la couche 1 est toujours égal au nombre total de termes linguistiques définis. Le Nombre de nœuds des couches 2, 3 et 4 est toujours égal au nombre de règles floues.

La figure III.7 montre un exemple de réseau ANFIS correspondant à un SIF à deux entrées avec 9 règles. A chaque entrée, correspondent trois fonctions d'appartenance ce qui implique que l'espace d'entrée est partagé en 9 sous-espaces flous où chacun couvre une règle floue.

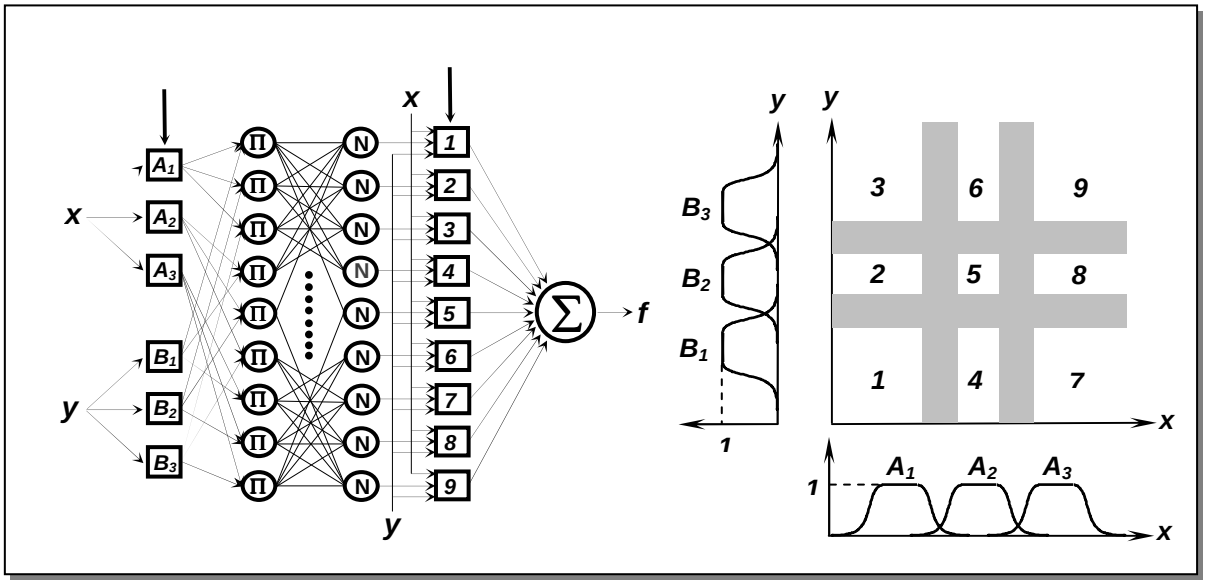


Figure III.6 Une structure ANFIS 2-entrées avec 9 règles

III.4.1 Algorithme d'apprentissage

L'apprentissage à partir d'un ensemble de données concerne l'identification des paramètres des prémisses et des conséquences, la structure du réseau étant fixée. L'algorithme d'apprentissage commence par construire un réseau initial, en suite nous appliquons une méthode d'apprentissage par rétro-propagation de l'erreur. Jang a proposé d'utiliser une règle hybride d'apprentissage qui combine un algorithme de descente de gradient avec une estimation par moindres carrés (MC).

Dans l'architecture ANFIS, l'apprentissage est basé sur une méthode hybride qui applique la méthode des moindres carrée combinée à la rétropropagation du gradient tout au long de l'apprentissage. Les paramètres des conclusions des règles qui sont linéaires par rapport à la sortie du système d'inférence floue sont ajustés par la méthode des moindres carrées dans le sens "*forward*" du réseau, puis vient l'ajustement des paramètres des prémisses des règles par la méthode du gradient dans le sens "*backward*" du réseau, et ce, pour chaque itération de l'algorithme. Le tableau III.1 nous résume les différentes étapes de l'algorithme :

-	Forward (une fois)	Backward (une fois)
Paramètres de la prémisse	Fixés	Descente du gradient
Paramètres de la conclusion	Moindres carrées	Fixés

Tablaux.III.1 La méthode hybride utilisé dans ANFIS

L'identification des paramètres non-linéraires de la partie prémisse des règles floues avec des fonctions d'appartenance de type gaussienne définies par la relation (III.18):

$$\mu_{A_i^j}(x_i) = \exp\left\{-0.5\left(v_i{}^j\left(x_i - c_i{}^j\right)\right)^2\right\} \quad \text{Avec } v_i{}^j = \frac{1}{\sigma_i{}^j} \tag{III.18}$$

où *c* est la moyenne, *v* est l'inverse de la variance (cette dernière est adoptée afin d'éviter la division par zéro lors de son adaptation) s'effectue comme suit:

Considérons l'ensemble des paramètres des prémisses caractérisé par le vecteur θ, et soit un ensemble de données entrée-sortie $(\underline{x}(k), y_d(k))$. Notre objectif est de trouver les valeurs du vecteur θ pour que la sortie du système flou approche le mieux possible la sortie désirée $y_d(k)$, en minimisant le critère :

$$J = \frac{1}{2}\sum\left(f(\underline{x}(k), \underline{\theta}) - y_d(k)\right)^2 \tag{III.20}$$

L'algorithme de rétro-propagation est donné par:

$$\underline{\theta}(k+1) = \underline{\theta}(k) - \lambda\frac{\partial J}{\partial \theta} \tag{III.21}$$

avec λ est le gain d'apprentissage.

Cet algorithme est souvent appelé algorithme de rétro-propagation du gradient, utilisé dans les réseaux de neurones. Cependant, ce dernier est connu par ses inconvénients d'être souvent lent et peut être piégé par des minima locaux lors de l'apprentissage.

Dans un système flou de type Sugeno, nous avons $\theta = [c \;\; v]^T$, où [c, v] sont les paramètres des prémisses, nous avons donc:

$$\frac{\partial J}{\partial \theta} = \left[\frac{\partial J}{\partial c} \;\; \frac{\partial J}{\partial v}\right]^T \tag{III.22}$$

En utilisant l'expression de la sortie du système flou, il vient:

$$\frac{\partial J}{\partial c_i^j} = \sum_{k \in I_{c_i^j}} \frac{\partial J}{\partial \mu_k}\frac{\partial \mu_k}{\partial \mu_{A_i^j}}\frac{\partial \mu_{A_i^l}}{\partial c_i^j} \tag{III.23}$$

où $I_{c_i^j}$: est l'ensemble des indices (*l*) des règles floues (R_l) dont lesquelles apparaît l'ensemble flou A_i^j.

$$\frac{\partial J}{\partial \mu_k} = \frac{\mu_k(y_k - y_d)}{\sum_{l=1}^{M}\mu_l} \tag{III.24}$$

$$\frac{\partial \mu_k}{\partial \mu_{A_i^j}} = \prod_{\substack{l=1 \\ l \neq i}}^{n} \mu_{A_l^{kl}}(x_l) \tag{III.24}$$

Pour une fonction d'appartenance gaussienne équation (III.18), on a :

$$\frac{\partial \mu_{A_i^j}}{\partial c_i^j} = v_i^{j\,2}(x_i - c_i^j)\mu_{A_i^j}(x_i) \tag{III.25}$$

Après calcul, il vient :

$$\frac{\partial J}{\partial c_i^j} = \frac{v_i^{j\,2}(x_i - c_i^j)\sum_{k \in I_{c_i^j}} \mu_k (y_k - y_d)}{\sum_{l=1}^{M} \mu_l} \tag{III.26}$$

De la même manière, on trouve :

$$\frac{\partial J}{\partial v_i^j} = \frac{-v_i^j (x_i - c_i^j)^2 \sum_{k \in I_{c_i^j}} \mu_k (y_k - y_d)}{\sum_{l=1}^{M} \mu_l} \tag{III.27}$$

Chapitre IV

Optimisation de Commandes Floue par les Algorithmes Génétiques

IV.1 Algorithme génétique adopté pour l'optimisation d'un SIF

Pour décrire le processus d'optimisation de l'AG, appliqué en commande, considérant le schéma bloc fonctionnel donné dans la figure III.1 où (e) et (Δe) désignent respectivement l'erreur et la variation de l'erreur.

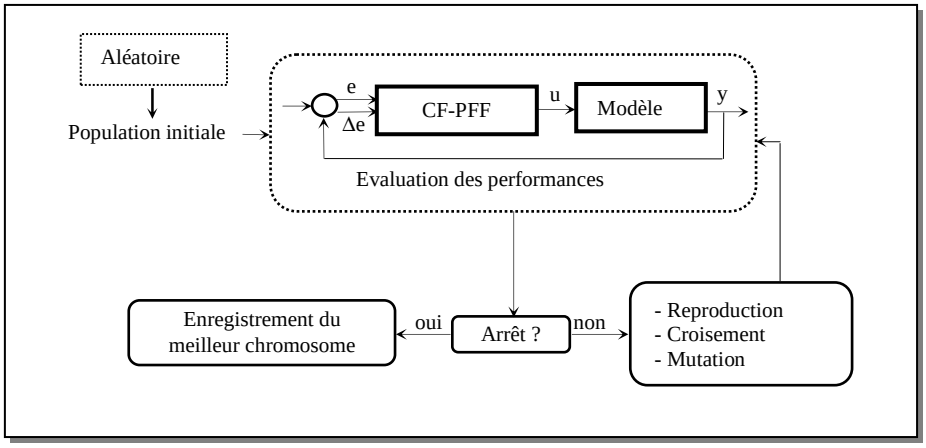


Figure III.1 *Schéma bloc fonctionnel montrant le processus d'optimisation par AG*

Les paramètres déterminés par le processus d'exploration, et qui garantissent un contrôle optimal, sont :

- d'une part, les paramètres des prémisses,
- et d'autre part, les paramètres des conclusions des règles.

Par l'utilisation de partitions floues fortes, il suffit seulement de coder les valeurs modales des fonctions d'appartenance, ce qui réduit considérablement l'espace de recherche.

IV.2 Représentation des solutions (codage)

Dans le mécanisme de l'AG, les paramètres à optimiser sont codés pour former le chromosome. Il existe deux types de codage dans la littérature, le codage réel et le codage binaire, nous avons adopté ce dernier.

Dans le codage binaire des paramètres, le nombre de bit pour coder chaque paramètre dépend de la marge et de la précision voulue.

Pour un SIF avec n entrées et m_i triangles par entrée, i = 1 à n, un individu de la population est codé par :

$A =$	$C_{1,1} \dots C_{1,m1}$	$C_{2,1} \dots C_{2,m2}$	…	$C_{n,1} \dots C_{n,mn}$	$W_{11} \dots W_{nmn}$

Les $(C_{i,j})_{j=1}^{m_i}$ représentent les valeurs modales des triangles définis sur le domaine de l'entrée i. De plus, nous imposons la contrainte :

$$(\forall i)(C_{i,1} < C_{i,2} < \dots < C_{i,m_i}) \qquad \text{(III.1)}$$

Par contre, les W_{ij} représentent les conclusions des règles (dans notre cas, des singletons).

IV.3 Opérateurs génétiques

Au début du processus d'optimisation, la population initiale comporte un ensemble de chromosomes qui sont dispersés partout dans l'espace de recherche. La population initiale

peut être aléatoirement produite ou peut être en partie fournie par l'utilisateur. Cependant, dans toutes nos simulations, la population se compose de 200 chromosomes qui sont tous tirés aléatoirement au début.

La sélection par la méthode de "*la roue biaisée*" est utilisée pour déterminer les membres de la nouvelle population [Goldgerg 94]. Une fois que la nouvelle population est construite, le croisement est effectué et ensuite est suivi par l'opération de mutation.

IV.4 Résultats de simulation

Dans ce paragraphe nous examinons les performances de la méthode présentée ci-dessus en l'appliquant pour la conception d'un contrôleur, de type Sugeno d'ordre zéro, du pendule inverse. Le contrôleur flou de type Sugeno à déterminer a une structure à 2 entrées(trois fonctions d'appartenance pour l'erreur (e) et trois fonctions d'appartenance pour la variation de l'erreur (Δe)) et une sortie u(t), avec 9 règles. Les fonctions d'appartenance sont de type triangulaire formant une partition floue forte. L'objectif global du système de commande est de réduire au minimum l'erreur à chaque instant k entre la réponse réelle du système et le point de consigne, l'indice de performance, F est choisi comme suit:

$$F = \sum_{k=1} e^2(k) \qquad \text{(III.2)}$$

IV.4.1 Optimisation des paramètres du contrôleur

Les paramètres introduits dans l'AG sont représentés comme suit:

- Les valeurs modales des triangles respectivement C_{11}, C_{12}, C_{13} pour l'erreur (e) et C_{21},C_{22},C_{23} pour la variation de l'erreur (Δe), sont choisis entre [-1 1], et chacun est codé sur une chaîne de 8 bits, tout en respectant la contrainte (III.1) lors du processus d'optimisation,
- Les conclusions des règles W_{11} … W_{33} sont choisies entre [-1 1], et chacun est codé sur une chaîne de 8 bits,
- La population contient 200 chromosomes (chacun de longueur de 120 bits) et le nombre maximal de génération est 100,
- La valeur du critère de performance F pour la solution obtenue est égale à 277.9652.

Le tableau (III.1 (a),(b)) nous donne les paramètres optimaux obtenus par l'AG après 40 générations:

Fcts d'appartenance	**Les points modaux des fonctions d'appartenance**		
e	-0.7433 **(C_{11})**	-0.4450 **(C_{12})**	0.5651 **(C_{13})**
Δe	-0.6173 **(C_{21})**	-0.1023 **(C_{22})**	0.8750 **(C_{23})**

(a)

Les conclusions des règles (w_{ij})								
W_{11}	W_{12}	W_{13}	W_{21}	W_{22}	W_{23}	W_{31}	W_{32}	W_{33}
-0.1042	-0.7482	-0.0281	-0.6063	0.2547	0.1254	-0.1463	0.9015	0.2321

(b)

TAB.III.1 P*aramètres optimaux obtenus par l'AG*

La figure III.2 représente les fonctions d'appartenance trouvées après optimisation :

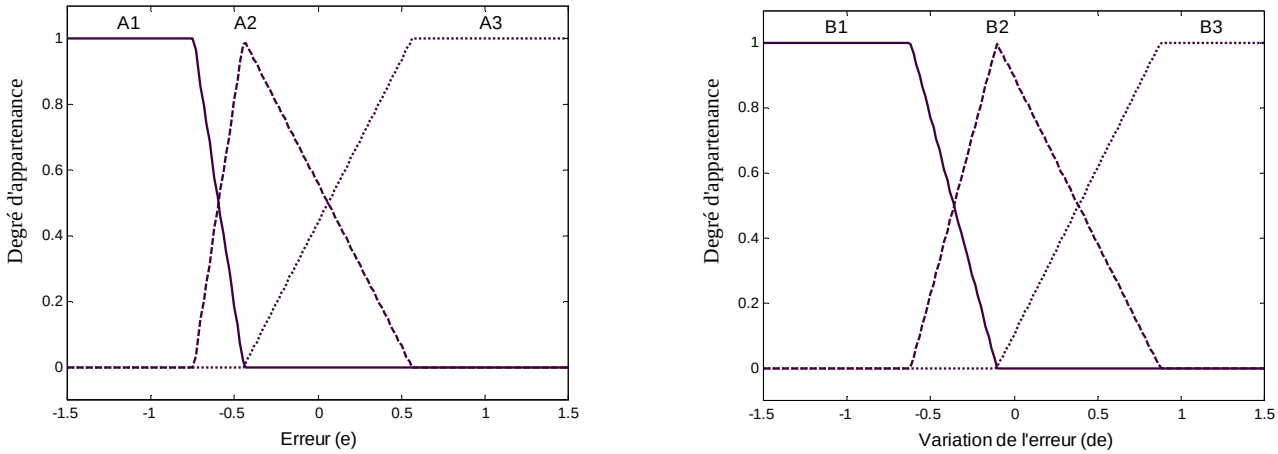
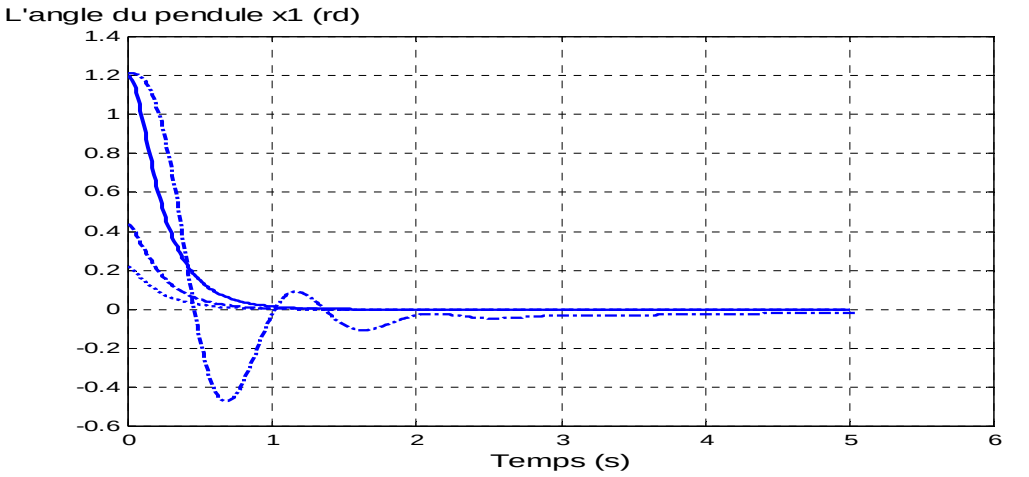
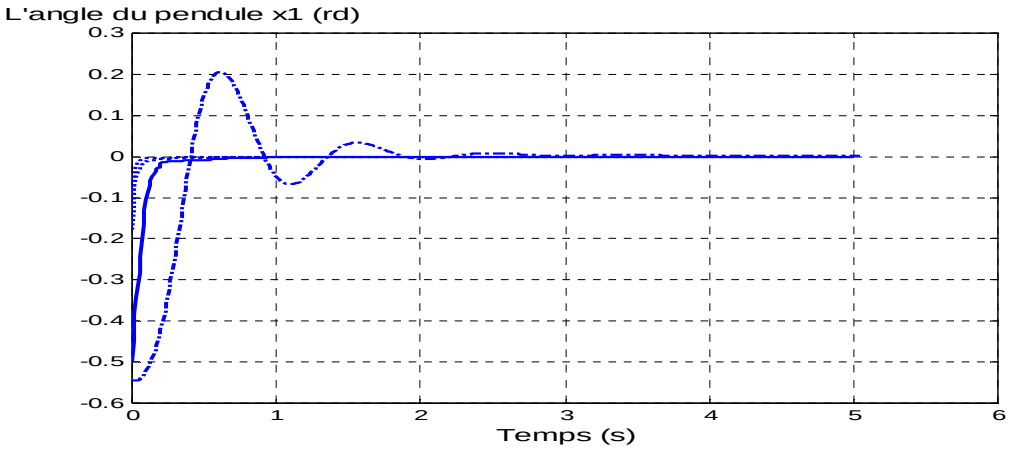


Figure III.2 *Disposition et forme des fonctions d'appartenance après optimisation*





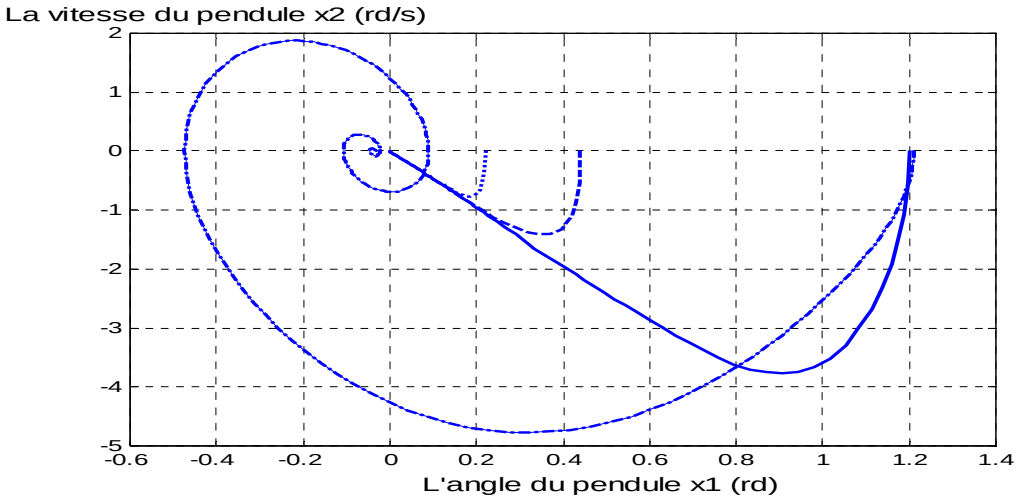
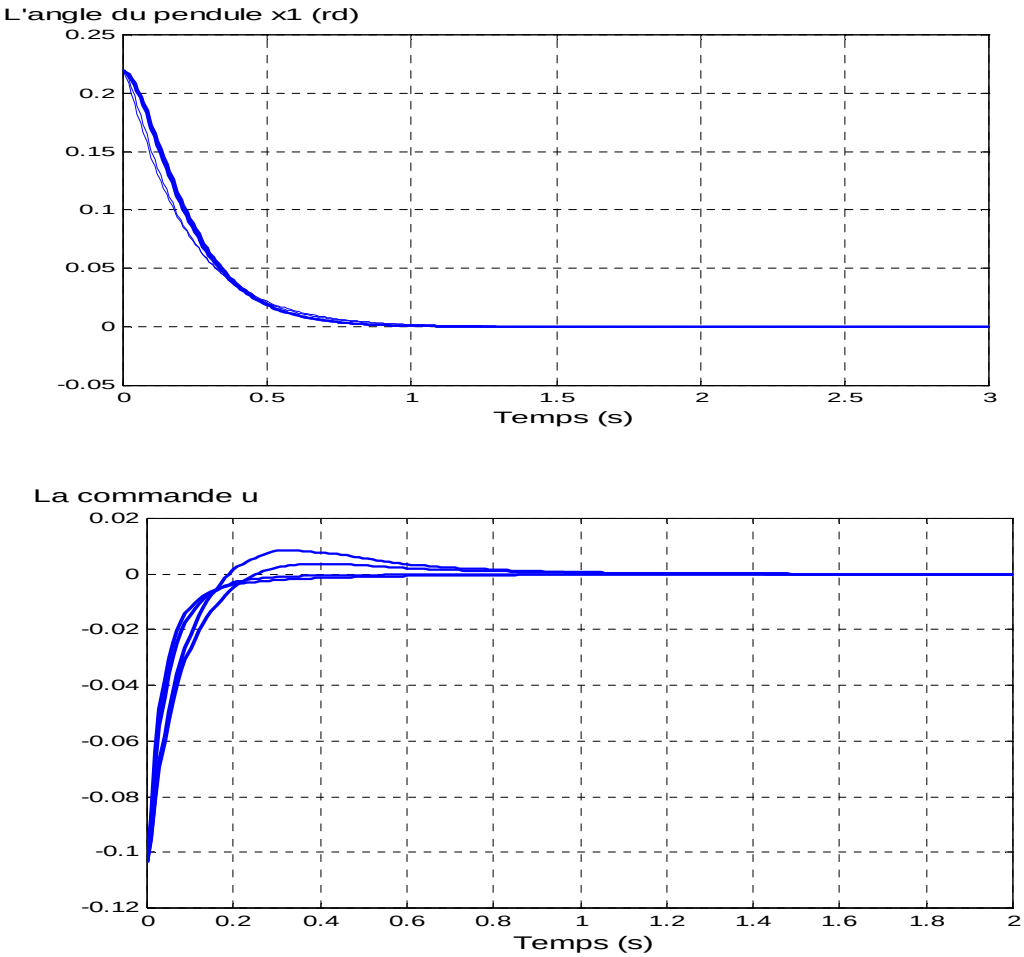


Figure III.3 Les réponses du pendule soumis au contrôleur optimal obtenues pour les paramètres nominaux du système et différentes conditions initiales



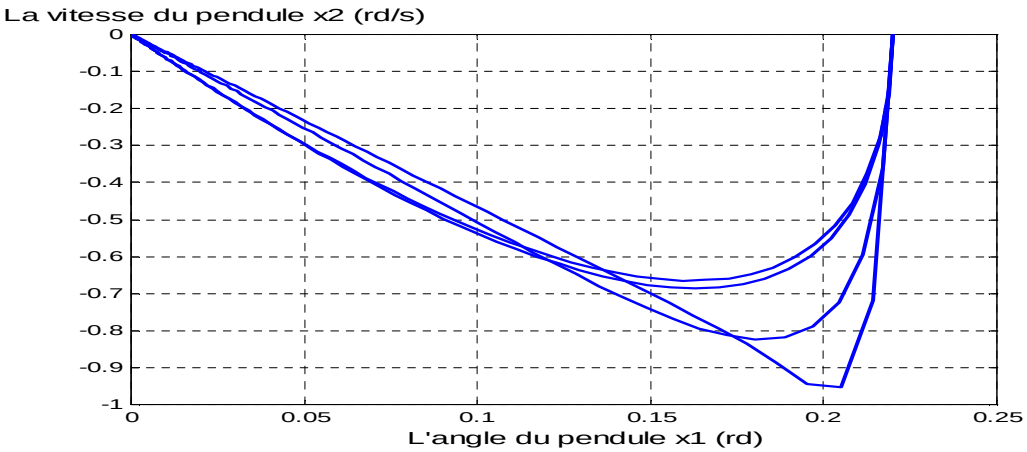


Figure III.4 *réponses du pendule soumis au contrôleur optimal obtenues pour une variation de 0% à 100% de la demi-longueur du segment (l)*

La figure (III.3) montre la performance du contrôleur flou optimisé pour le contrôle du pendule inverse avec des conditions initiales différentes (0.22 rad, 0 rad/s), (0.44 rad, 0 rad/s), (1.2 rads, 0 rad/s). Nous remarquons que le contrôleur arrive toujours à stabiliser le pendule en un temps de réponse d'environ 1.25 s pour atteindre l'état d'équilibre, et il n'échoue qu'à partir d'un angle qui dépasse la valeur de 1.2 rads.

Pour voir la réaction du contrôleur aux changements des paramètres du système, nous avons procédé à une variation de 0% à 100% du paramètre *l* du système. La figure (III.4) montre que le contrôleur arrive toujours à stabiliser le système avec un temps de réponse d'environ de 1.2 s pour atteindre l'état d'équilibre. Pour une variation de plus de 100% le contrôleur commence à perdre ses caractéristiques.

Bibliographie

BIBLIOGRAPHIES

[1] R.KETATA

" Méthodologies de régulation numérique incluant la logique floue "-Thèse de Docteur L.A.A.S Toulouse – 1992

[2] M. BAUER

" Application de contrôleurs à logique floue pour la commande vectorielle des machines à induction: commande en vitesse et en position "

Rapport de stage ingénieur- ESIM-22 Février 1995

[3] A.IBALIDEN

" Implantation d'un régulateur de type flou sur des commandes d'onduleurs pilotant des machines alternatives : application à la détermination des correcteurs "- Rapport d'activités 1994-1995

[4] B. BEAUFRERE

" Application de la logique floue à la planification de trajectoires de robots mobiles dans des environnements inconnus "- Thèse de Docteur de l'université de Poitiers, Décembre 1994.

[5] P. Y. GLORONNEC

" Algorithme d'apprentissage pour systèmes d'inférences floues " - INSA de Rennes (IRISIA)- 1999.

[6] B. DEMAYA

" Commande floue des systèmes à dynamiques complexes- Application à la commande d'un moteur thermique "- Thèse de Docteur LAAS Toulouse - 17 Octobre 1994

[7] H. BUHLER

" Réglage par logique floue " -Presse polytechniques et universitaires Romandes –1994

[8] K. ZINSER, R. SCHREIBER

" La logique floue: une nouveauté prometteuse de la technique d'automatisation "- La Technique moderne- N° 1-2- 1994

[9]M.RACHDI , DJ.MAIZI

"Implantation sur DSP de contrôleurs flous pour la commande vectorielle du modèle analogique de la machine asynchrone " - Projet de fin d'études ingénieur - EMP 1997

[10] F.AMRANE , N.GOULMANE

"Commande par logique floue d'un moteur à courant continu "- Projet de fin d'études ingénieur - EMP 1997

[11] K.BENMANSOUR
« Etude et implantation de contrôleurs robustes et flous d'une machine synchrone à aimants permanents avec pilotage vectoriel », Thèse de magister- Novembre 1999