REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
وزارة التعليم العالي و البحث العلمي

UNIVERSITE SAAD DAHLEB DE BLIDA
جامعة سعد دحلب البليدة

FACULTE DES SCIENCES
كلية العلوم
DEPARTEMENT DE MATHEMATIQUES
دائرة الرياضيات

Mémoire de Magister

EN
MATHEMATIQUES APPLIQUEES

OPTION
MODELISATION MATHEMATIQUE
ET TECHNIQUES DE DECISION

THEME:

Contrôle flou adaptatif par les Réseaux de neurones

Présenté par
SELLALI – HEBBACHE Annisa

Rapporteur : Mr A. Guessoum **Prof** Université de Blida

Présenté devant le jury composé de :

Président	**: Mr F. Hannane**	**M.C**	Université de Blida
Rapporteur	**: Mr A. Guessoum**	**Prof**	Université de Blida
Examinateurs:	**Mr M. Blidia**	**M.C**	Université de Blida
	Mr H. Azzoune	**Dr**	U.S.T.H.B

Date de soutenance : 2001/2002

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
وزارة التعليم العالي و البحث العلمي

UNIVERSITE SAAD DAHLEB DE BLIDA
جامعة سعد دحلب البليدة

FACULTE DES SCIENCES
كلية العلوم
DEPARTEMENT DE MATHEMATIQUES
دائرة الرياضيات

Mémoire de Magister

EN
MATHEMATIQUES APPLIQUEES

OPTION
MODELISATION MATHEMATIQUE
ET TECHNIQUES DE DECISION

THEME:

Contrôle flou adaptatif par les Réseaux de neurones

Présenté par
SELLALI – HEBBACHE Anissa
Sous la direction de
Mr A . GUESSOUM

2001/2002

ملخص

إن إشتراك المنطق الغامض و شبكات الخلايا العصبية تبين جد مفيد في مراقبة الأنظمة الصناعية المركبة، التي يصعب الحصول على وجود أنظمة رياضية متقنة.

هذا الاشتراك يمتاز بقدرة خلق نموذج رياضي للغة الطبيعية للمنطق الغامض، و ذلك باستعمال قدرة التعلم لشبكات الخلايا العصبية.

هذه المذكرة تتطرق إلى إنشاء مراقب بالعصب الغامض.

لتحسين الوسائط لدالات الانتماء، نستعمل طريقة خاصة للانتشار العكسي بالتدرج

Résumé

L'association réseaux de neurones-logique floue s'est révélée très fructueuse dans le contrôle des processus industriels complexes pour lesquels une modélisation mathématique rigoureuse est difficile à établir.

Elle a pour avantage de combiner les capacités de modélisation des langages naturels de la logique floue aux capacités d'apprentissage des réseaux de neurones évitant ainsi toute notion de subjectivité liée aux informations émanant d'opérateurs humains.

Le travail présenté consiste en la conception et la réalisation d'un contrôle neuro flou.

Ce contrôleur sera implementé sous forme de SIFANE (Système d'inférence floue à architecture neuronale).

Utilisant les informations d'entrée et de sortie du système, les fonctions d'appartenances associées aux variables linguistiques seront ajustées à l'aide d'une forme spéciale de la rétro propagation du gradient.

Abstract

Neural network-fuzzy logic association has recently found various applications in industry for complex systems that are not easily modelised.

Neuro fussy logic controlers provides a feasible alternative since they can easily capture the approximate, qualitative aspects of human knowledge and reasoning, particulary with the capabilities of learning of neural network.

The goal of this work is to presente a control strategy called ANFIS based of neural network and fussy logic.

To ajusted the parameters of this controler, we apply a special form of the back propagation type gradient descent.

Remerciements :

Je tiens tout d'abord à remercier le professeur A. Guessoum, mon directeur de thèse, pour m'avoir accueilli dans son laboratoire et pour m'avoir orientée par ses suggestions et ses critiques constructives.

Je remercie également messieurs les membres du jury, pour avoir accepté d'examiner et d'apporter une appréciation sur mon travail.

Ma gratitude va à toutes les personnes, collègues, amis et famille pour leur appui moral durant cette période ; qu'elles trouvent ici l'expression de mes remerciements les plus chaleureux !

Un grand merci à l'équipe du centre de calcul de la faculté des sciences de l'université de Blida pour leur appui matériel et leur totale disponibilité .

Préface

Au fur et à mesure que la complexité d'un système augmente, notre habilité à formuler de manière précise et significative son comportement, diminue jusqu'à une limite, au delà de laquelle la précision et la signification deviennent des caractéristiques pratiquement mutuellement exclusives.

Lotfi Zadeh

Table des Matières

Introduction Générale

Le suivi du fonctionnement des systèmes complexes est un enjeu industriel majeur. Le mauvais fonctionnement, voire l'arrêt d'une machine coûtent en effet très chers. Il importe donc d'en effectuer le diagnostic le plus tôt possible afin de prendre les décisions qui s'imposent et d'en limiter ainsi les conséquences. De plus pour des raisons de sécurité et de coût, certains états dangereux ou non désirés ne peuvent pas être observés.

L'automatique fournit une large famille de méthodes pour synthètiser des lois de commande lorsque l'on dispose d'un modèle suffisamment précis du processus, généralement un système d'équations différentielles reliant les entrées aux sorties surtout lorsque les modèles sont linéaires.

Lorsqu'une modélisation mathématique rigoureuse du système n'existe pas ou est trop complexe mais qu'une conduite de processus est réalisée par un être humain, expert dans le domaine, il serait intéressant d'envisager un moyen d'exploiter correctement le savoir humain pour automatiser certaines tâches. On est donc confronté à la nécessité de modéliser les informations fournies par un opérateur humain dans un langage naturel, phénomène qui est en rupture avec la tradition des scientifiques, lesquels se préoccuppent essentiellement de la modélisation de l'univers physique.

Le problème de représentation et d'utilisation de ce genre de connaissances sont au centre d'une discipline relativement nouvelle qu'on appelle intelligence artificielle. Cette discipline a eu un impact très limité sur les applications industrielles parce

qu'elle a mis l'accent de façon exclusive sur le traitement symbolique de la connaissance, par opposition à la modélisation numérique utilisée traditionnellement dans les sciences de l'ingénieur.

Plus récemment, on assiste à un retour du numérique dans les problèmes d'intelligence artificielle avec l'utilisation conjointe de la logique floue et des réseaux de neurones.

Introduite en 1965 par LOTFI ZADEH, professeur renommé d'automatique et théorie des systèmes, la logique floue n'a intéressée qu'un nombre restreint de chercheurs. Il a fallu attendre les années 80 pour voir apparaitre au Japon son utilisation dans des domaines trés diversifiés.

L'effervescence japonaise en la matière s'est essentiellement manifestée dans le domaine de la commande floue puis élargissant l'application à d'autres secteurs : electroménager, transports, audivisuel, engins mobiles ,...

Les applications developpées ont montré que l'association de la logique floue-réseaux de neurones.s'est révélée trés fructieuse dans le contrôle des processus industriels.

Le travail présenté consiste en la conception d'un contrôleur neuroflou. Ce contrôleur sera implémenté sous forme d'un SIFANE (Système d'inférence floue à architecture neuronale).

Ce mémoire est organisé de la manière suivante :

Le chapitre 1 présente les élémerts fondamentaux de l'approche floue au point de vue des concepts et opérations de base.

Dans le chapitre 2 nous aborderons la commande floue.

Le chapitre 3 sera consacré à l'étude théorique ces réseaux de neurones.

Dans le chapitre 4 sera présenté la modélisation neuronale d'un système d'inférence floue.

Le chapitre 5 sera consacré à l'expérimentation et à la validation du modèle conçu.

Une bibliographie est également proposée.

Chapitre 1

La logique floue : Notions de base

1.1 Introduction

La théorie des ensembles flous est une théorie mathématique dont l'objectif est de modéliser des notions vagues du langage naturel pour pallier à l'inadaptation de la théorie des ensembles classiques dans ce domaine.

La caractéristique fondamentale d'un ensemble classique est le passage brusque de l'appartenance à la non appartenance. Le concept d'appartenance est modélisé par une fonction φ_A appelée fonction caractéristique de l'ensemble A.

La fonction φ_A définie sur un ensemble de référence U et à valeur dans $\{0,1\}$ est telle que pour tout $x \in U$

$$\varphi_A(x) = \begin{cases} 1 & \text{si} \quad x \in A \\ 0 & \text{si} \quad x \notin A \end{cases}$$

Exemple :

U désigne une communauté donnée et A est un sous ensemble de U défini par la caractéristique d'être "vieux" pour tout $x \in U$

$$\varphi_A(x) = \begin{cases} 1 & \text{si } x \text{ est vieux} \\ 0 & \text{si } x \text{ est jeune} \end{cases}$$

La fonction φ_A signifie que tout membre de la communauté U est soit vieux soit jeune, excluant par conséquent tout cas intermédiaire qui supposerait l'existence d'éléments x sur la frontière de A (individus d'age moyen).

La théorie des ensembles flous se propose de généraliser cette fonction d'appartenance pour des catégories vagues.

1.2 Définition d'un ensemble flou

Soit U un ensemble de référence (ou univers de discours). On définit un ensemble flou A dans U par la donnée d'une application μ_A de U dans l'intervalle réel $[0, 1]$:

A tout élément $x \in U$, on associe une valeur $\mu_A(x)$ telle que :

$$\mu_A : U \to [0, 1] \qquad 0 \leqslant \mu_A(x) \leqslant 1 \tag{1.1}$$

μ_A est appelé fonction d'appartenance de l'ensemble flou A. Elle n'est pas nécessairement égale à 0 ou 1. Elle est à priori quelconque et désigne le degré d'appartenance de x à l'ensemble A.

On peut distinguer trois cas :

$\mu_A(x) = 0$: $x \notin A$, c'est à dire que l'élément x ne satisfait pas du tout la propriété vague sous entendu par A

$\mu_A(x) = 1$: $x \in A$, c'est à dire que x satisfait pleinement la propriété vague sous entendu par A.

$0 < \mu_A(x) < 1$: dans ce cas, le degré d'appartenance de $\mu_A(x)$ est une valeur intermédiaire entre 0 et 1, c'est à dire que x ne satisfait la propriété vague définie par A que partiellement à un certain degré $\mu_A(x)$

Un ensemble flou A est complétement défini par l'ensemble des couples $(x_i, \mu_A(x_i))$ noté $\mu_A(x_i)/x_i$.

Selon que le référentiel est fini ou infni, ou plus précisément selon que le support de l'ensemble flou A est fini ou infini, on a deux modes de présentation [9] :

Cas fini :

$$A = \sum_{i=1}^{N} \mu_A(x_i)/x_i \quad \text{ou} \quad \mu_A(x_1)/x_1 + ... + \mu_A(x_N)/x_N \tag{1.2}$$

Cette somme (+) n'est qu'un *ou* logique et $\mu_A(x_i)/x_i$ représente le couple $(\mu_A(x_i), x_i)$

Cas infini : on note

$$A = \int_U \mu_A(x)/x \tag{1.3}$$

Exemple 1 :

Soient les ensembles flous PETIT, MOYEN et GRAND définis de la manière suivante :

$Petit = \{1/0, 1/0.1, 0.9/0.2, 0.8/0.3\}$

$Moyen = \{0.5/0.3, 0.9/0.4, 1/0.5, 0.9/0.6\}$

$Grand = \{0.5/0.6, 0.8/0.7, 1/0.9, 1/1\}$

On voit que 0 satisfait pleinement la propriété "petit" ($\mu_{petit}(0) = 1$) et 0.9 satisfait la propriété "petit" à un degré égal à 0.2 ($\mu_{petit}(0.9) = 0.2$) .

Exemple 2 :

Soit $U = \mathbb{R}$ et

$$\mu_A(x) = \begin{cases} 1 - \sqrt{\frac{|x-6|}{3}} & \text{si} \quad 3 \leqslant x \leqslant 9 \\ 0 & \text{sinon} \end{cases}$$

alors

$$A = \int_{\mathbb{R}} \mu_A(x)/x$$

est l'ensemble flou représentant les nombres réels approximativement égaux à 6.

Exemple 3 :

Soit l'univers U des températures de 0 à 45°

$U = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45\}$

On peut définir les notions de $Froid$ et de $Chaud$ respectivement par les ensemble flous suivants :

$Froid = \{1/0, 0.9/5, 0.6/10, 0.5/15, 0.3/20, 0.2/25, 0.1/30, 0/35, 0/40, 0/45\}$

$chaud = \{0/0, 0/5, 0.1/10, 0.5/15, 0.7/20, 0.8/25, 0.9/30, 1/35, 1/40, 1/45\}$

Les fonctions d'appartenances $\mu_{Froid}(x)$ et $\mu_{Chaud}(x)$ représentées sur la fig 1.2 sont à comparer avec les fonctions caractéristiques des sous-ensembles A_0 et B_0 représentées respectivement par la figure 1.1 où A_0 est le sous-ensemble des températures froides et B_0 est le sous-ensemble des températures chaudes avec par exemple

$A_0 = \{0, 5, 10, 15, 20, 25\}$

$B_0 = \{25, 30, 35, 40, 45\}$

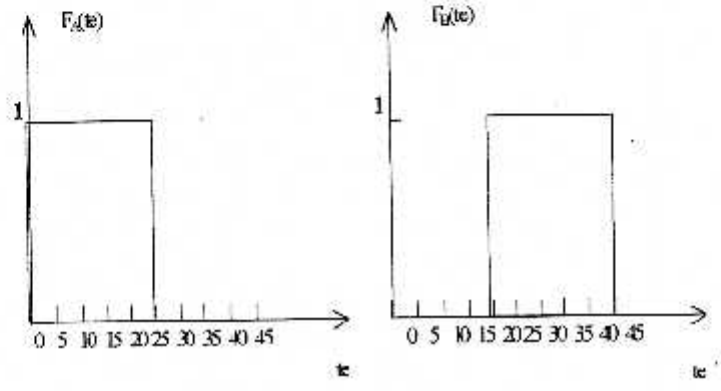

Fig 1.1 : Fonctions caractéristiques des sous ensembles A_0 et B_0

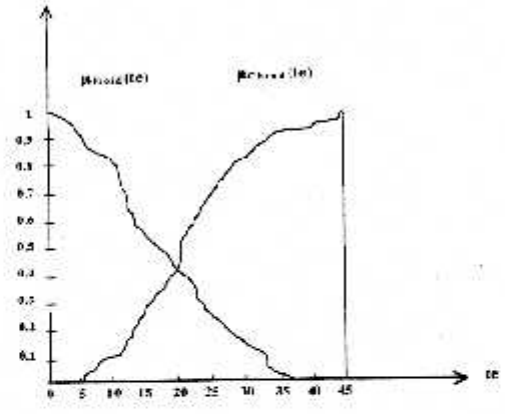

Fig 1.2 : Fonctions d'appartenances des ensembles flous A (chaud) et B (froid)

1.3 Degré d'appartenance et degré de probabilité

il s'agit dans les deux cas d'un nombre réel compris entre $[0, 1]$ mais la notion de degré d'appartenance et la notion de probabilité sont différentes par nature.

Une probabilité est associée inévitablement à la notion d'événement. C'est une mesure associée à l'occurence d'un événement : il y a donc incertitude et attente de voir celui ci se réaliser.

Par contre un degré d'appartenance constitue une mesure d'incertitude par rapport à une notion vague [20]. Cette mesure indique que selon l'univers de discours considéré, et selon les conventions adoptées, une propriété est satisfaite à des degrés divers.

Dire que la probabilité que T soit une température froide est $= 0.5$ veut dire que T à 50% de chance d'appartenir à l'ensemble A_0 de l'exemple précédent alors que $\mu_{Froid}(T)$ veut dire que T appartient à l'ensemble flou " froid" mais qu'elle ne vérifie cette propriété que partiellement à un degré 0.5.

1.4 Opérations sur les ensembles flous

Les opérations sur les ensembles flous sont en général des extensions des opérations bien connues sur les ensembles classiques (égalité, intersection, réunion, complément) [53].

1.4.1 Egalité

Soient deux sous-ensembles flous A et B dans un univers de discours U. On dit que A et B sont égaux et on écrit $A = B$, si leur fonction d'appartenance ont la même valeur en tout x de U.

$$\mu_A(x) = \mu_B(x) \qquad \forall x \in U. \tag{1.4}$$

1.4.2 Inclusion

Un ensemble flou A est un sous-ensemble d'un ensemble flou B si

$$\forall x \in U \qquad \mu_A(x) \leqslant \mu_B(x) \tag{1.5}$$

On dit que A est inclus dans B et on écrit $A \subseteq B$

1.4.3 Réunion

La réunion de deux ensemble flous A et B d'un même référentiel U est un ensemble flou noté $A \cup B$ et dont la fonction d'appartenance est :

$$\mu_{A \cup B}(x) = \max\left(\mu_A(x), \mu_B(x)\right) \qquad \forall x \in U \tag{1.6}$$

1.4.4 Intersection

L'intersection de deux ensembles A et B de même référentiel U est l'ensemble flou noté $A \cap B$ et dont la fonction d'appartenance est :

$$\mu_{A \cap B}(x) = \min\left(\mu_A(x), \mu_B(x)\right) \qquad \forall x \in U \tag{1.7}$$

1.4.5 Complément

Le complément d'un ensemble flou A dans un univers U est ensemble flou $\overline{A}$ dont la fonction d'appartenance est

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \tag{1.8}$$

Le principe du tiers exclu selon le quel il n'y a pas de catégorie intermédiaire entre A et $\overline{A}$ et le principe de non contradiction selon lequel on ne peut pas avoir A et $\overline{A}$ simultanément ne sont pas vérifiés :

$$\begin{aligned} A \cap \overline{A} &\neq \phi \\ A \cup \overline{A} &\neq U \end{aligned} \tag{1.9}$$

En définitive, le complément flou est plus nuancé que le complément ordinaire, car il nous indique qu'il n'y a pas de frontière franche entre une notion vague et son contraire. Ainsi par exemple une température peut être considérée comme froide à un certain degré et chaude à un autre.

1.5 Normes et Conormes triangulaires

L'intersection et la réunion de deux ensemble flous ont été définies par plusieurs fonctions appelées normes et conormes triangulaires ou encore t-normes ou t-conormes. Elles sont destinées à représenter différentes formes de disjonction et conjonction utilisées en logique floue [19].

1.5.1 Normes triangulaires ou t-normes

Une fonction $t(x, y)$ à valeurs dans l'intervalle $[0, 1]$ et dont les variables x et y appartiennent également au même intervalle est une norme triangulaire ou une t-norme si et seulement si elle satisfait les conditions suivantes:

•Elle est commutative et associative .

•Elle est non décroissante par rapport à chacune des variables x et y. Autrement dit, si $x \leqslant x_1 et y \leqslant y_1 alors\ t(x, y) \leqslant t(x_1, y_1)$;

•Elle admet un élément neutre qui est la valeur 1

$$\forall x \in [0, 1], \quad t(x, 1) = x$$

1.5.2 Conormes triangulaires ou t-conormes

Une fonction ou opération $s(x, y)$ de deux variables $x, y \in [0, 1]$, à valeurs dans l'intervalle $[0, 1]$ est une conorme triangulaire ou t-conorme, si et seulement si elle satisfait les conditions suivantes :

- Elle est commutative et associative;
- Elle est non décroissante par rapport à chacune des deux variables x, y;
- Elle admet un élément neutre qui est la valeur 0

$$\forall x \in [0, 1], \quad S(x, 0) = x$$

1.5.3 Variable linguistique

Une variable linguistique est une variable dont les valeurs ne sont pas numériques, mais plutôt symboliques, en termes d'expression du langage naturel [20].

Elle est définie par le triplet (X, U, T_X) où X désigne le nom de la variable, U le référentiel et T_X l'ensemble fini ou non des valeurs linguistiques de la variable X appelées termes.

La nature de la variable X dépend de celle des éléments de l'univers U, ainsi l'âge d'un arbre ne serait pas représenté avec les mêmes valeurs que l'age d'un être humain.

Exemple :

X désigne la taille d'un être humain. L'univers des tailles en cm est :

$U = \{80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200\}$

L'ensemble T_X est constitué par l'ensemble des termes (très petit, petit, moyen, grand, très grand) qu'on peut associer respectivement à des ensemble flous A_1, A_2, A_3, A_4, A_5.

1.6 Conclusion

Dans ce chapitre nous avons présenté les éléments de base de la logique floue.Le lecteur souhaitant avoir une connaissance théorique plus approfondie sur cette dernière, de ces conditions d'utilisation, de ces avantages et inconvenients, pourra se référer à la bibliographie [12],[32],[53].

Nous tenterons de montrer dans le chapitre suivant comment ces notions sont

utilisées comme modèle de représentation du langage et du raisonnement humains.Nous y aborderons la commande floue qui s'avère être le champ d'application le plus actif à travers le monde.

Chapitre 2

La commande floue

2.1 Introduction

2.1.1 Rappel d'un automatisme

Un automatisme est un dispositif qui gère les informations de fonctionnement d'une machine, d'un appareil, d'une installation ou de tout processus complexe dans le but d'en assurer la commande [53]. Selon les cas, cette commande peut être partiellement ou complètement indépendante de l'intervention humaine.

2.1.2 Structure générale d'un automatisme

Un automatisme est schématiquement constitué de deux parties : une partie opérative correspondant à l'objet commandé (machine, processus....) et une partie commande qui gère les informations de fonctionnement de cet objet.Le lien entre

ces deux parties est réalisé au moyen de deux interfaces : une interface d'action qui transmet les ordres de commande à la partie opérative par l'intermédiaire d'organes appelés actionneurs (moteurs, vannes,...) et une interface d'observation constituée d'organes appelés capteurs qui fournissent un compte rendu à la partie commande sur l'état de l'objet commandé. La partie commande peut être munie d'une interface de dialogue avec le monde extérieur (opérateur humain) [53].

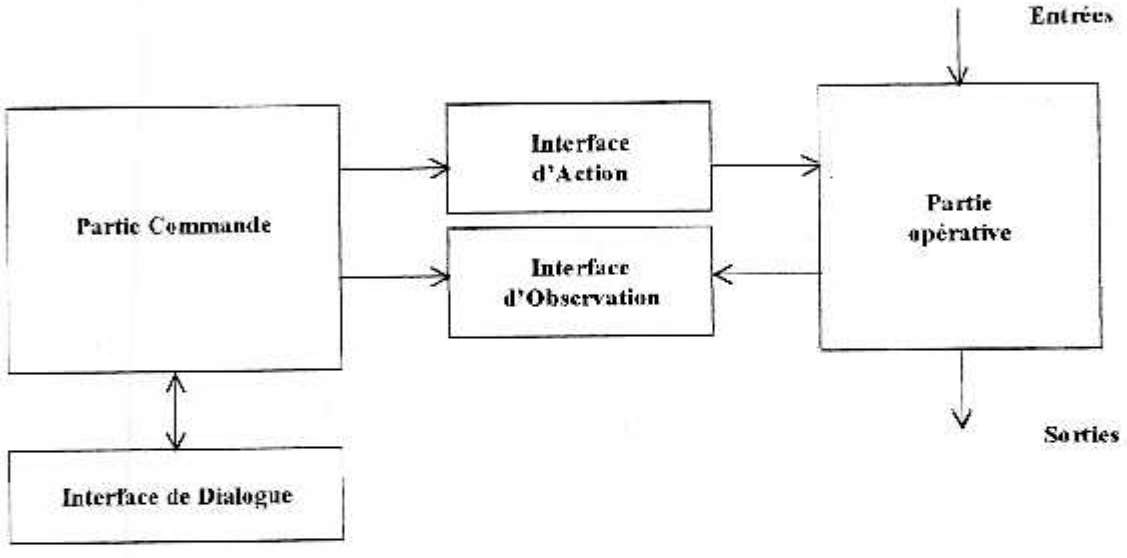

Fig 2.1 : Structure d'un automatisme

Par ailleurs, les paramètres de sortie ne peuvent être maintenus en général constants du fait de diverses perturbations, et même si les ordres d'entrées sont constants.Par conséquent, une régulation par rétroaction est nécessaire pour corriger l'écart des paramètres de sortie par rapport à une référence appelée consigne.La structure normale d'un automatisme est donc en boucle d'asservissement comme

l'indique le schéma ci dessous :

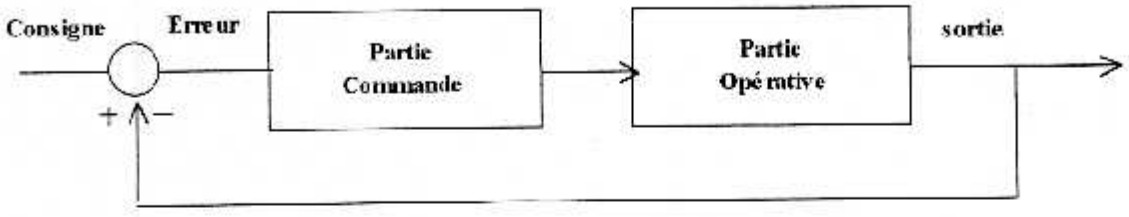

Fig 2.2 : structure en boucle d'un automatisme

2.2 La commande floue

2.2.1 La commande de processus

On appelle processus dynamique une entité pouvant être de nature variée (physique, économique,...) susceptible d'évoluer en fonction du temps.

On suppose que le comportement en fonction du temps n'est pas satisfaisant vis à vis d'un certain nombre de critères. L'objectif de la commande est d'agir sur le processus pour atteindre ou se rapprocher d'un comportement désiré.

Un processus est en liaison avec l'environnement par un ensemble de variables d'entrées, un ensemble de variables de sorties et un ensemble de grandeurs appelées perturbations.

Résoudre le problème de commande consiste à calculer l'évolution des variables

de commande et appliquer les résultats de calcul au système pour obtenir de celui ci un comportement désiré.

La loi d'évolution des entrées et appelée loi de commande. Le contrôleur est l'organe chargé d'engendrer cette commande.

On parle de commande floue lorsque le contrôleur est réalisé en logique floue. Dans ce cas, on synthètise la loi de commande en codant la connaissance fournie en langage naturel par des experts qui contrôlent le processus.

2.2.2 Structure d'un contrôleur flou

Un contrôleur flou est un système à base de connaissances [32]. Il utilise une expertise sous forme de règles dont la forme est pour un contrôleur à deux entrées et une sortie, la suivante :

Si (X_1 *est* A_1) et (X_2 *est* A_2) alors (Y *est* B).

L'expression (X_1 *est* A_1) et (X_2 *est* A_2) est appelée prémisse de la règle.

(Y *est* B) est appelée conclusion de la règle.

X_1, X_2, Y représentent les variables physiques de processus à commander.

A_1 , A_2 , B sont des valeurs linguistiques prises respectivement par les variables X_1, X_2, Y.

Les différentes valeurs linguistiques que pourront prendre les variables X_1, X_2 et Y sont représentées par des ensembles flous caractérisés par

leurs fonctions d'appartenances.La plupart des fonctions d'appartenances rencontrées en commande sont de forme triangulaire, trapèzoïdale, ou des singletons.Bien

qu'il soit possible d'utiliser d'autres courbes (courbe en cloche,) ces fonctions sont préférées pour des raisons de facilité d'implémentation.

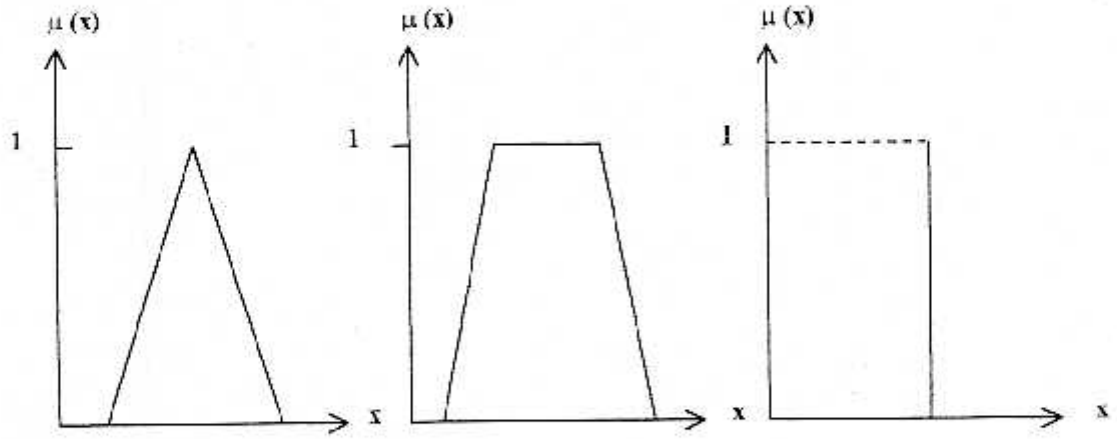

Fig 2.3 : Formes courantes des fonctions d'appartenances

Les fonctions d'appartenances d'une variable donnée doivent assurer un recouvrement flou de l'univers de discours de cette variable. Selon le domaine couvert chaque fonction d'appartenance doit avoir au plus deux points d'intersections : le premier avec la courbe située immédiatement à gauche et l'autre avec la courbe située immédiatement à droite.

On distingue deux types de règles selon que la conclusion est symbolique dans la méthode de Mamdani ou qu'elle est numérique, comme la méthode de Sugeno dans laquelle les conclusions sont des polynômes représentant une combinaison linéaire des entrées.

Deux exemples respectifs sont :

- Si ε est grande alors u est moyenne

- Si ε est grande alors $u = 3 * \varepsilon + 2$

Dans les contrôleurs flous les plus répandus, il n'y a pas d'enchaînement des règles, toutes les règles activables sont activées. Une règle est activable dés que sa prémisse est caractérisée par un degré d'appartenance non nul.

2.2.3 Les différentes étapes de traitement de l'expertise dans un contrôleur flou

En règle générale , un contrôleur flou est composé de trois modules :

-La fuzzification

-L'inférence

-La défuzzification

Pour la plupart des étapes, il existe plusieurs possibilités théoriques [32].Nous nous limiterons dans ce chapitre aux choix les plus courants .

la fuzzification La fuzzification désigne l'opération de projection des variables physiques réelles sur les ensembles flous caractérisant les valeurs linguistiques prise par ces variables.La fuzzification d'une mesure précise d'une variable réelle permet de caractériser le degré avec lequel la mesure appartient à un ensemble flou donné.

Exemple :

Pour la règle suivante : Si ε est grande alors u est moyenne, la fuzzification permet de calculer le degré d'appartenance de ε à l'ensemble flou "grande".

Opérations de base sur les ensembles flous Les prémisses des règles comportent en général plusieurs clauses liées par des connecteurs ET, OU et NON .En pratique, pour les opérations de conjonction et disjonction, on souvent recours aux opérateurs Min et Max.

$$\begin{aligned} \mu_{A \text{ et } B}(x,y) &= \min\left(\mu_A(x), \mu_B(y)\right) \\ \mu_{A \text{ ou } B}(x,y) &= \max\left(\mu_A(x), \mu_B(y)\right) \end{aligned} \tag{2.1}$$

Quant à la négation $\bar{A}$ d'un ensemble flou A elle est caractérisée par:

$$1 - \mu_A(x). \tag{2.2}$$

L'implication floue L'implication floue est un opérateur qui permet d'évaluer un degré de vérité d'une règle R de la forme si $(x \ est \ A)$ alors $(y \ est \ B)$.

A partir des valeurs de la prémisse d'une part et celles de la conclusion d' autre part

$$\mu_R(x,y) = I\left(\mu_A(x), \mu_B(y)\right) \tag{2.3}$$

Il existe plusieurs possibilités. Les opérateurs les plus utilisés en pratique sont les implications de Mamdani et de Larsen.

* Implication de Mamdani: $\mu_M(x,y) = \min\left(\mu_A(x), \mu_B(y)\right)$

* Implication de Larsen : $\mu_L(x,y) = \mu_A(x) * \mu_B(y)$.

Inférence floue En logique classique, le modus ponens permet à partir de la règle

Si x est A alors y est B

et du fait que (X *est* A)de conclure que (Y *est* B). Lotfi Zadeh a étendu ce principe au cas flou, principe qu'on appelle le modus ponens généralisé

Fait	X *est* A	X *est* A'
Règle	si (X *est* A) alors (Y *est* B)	si (X *est* A) alors (Y *est* B)
Déduction	Y *est* B	y *est* B'

A partir de la règle si$X$ *est* A alors (Y *est* B), on déduit un nouveau B' qui est caractérisé par un ensemble flou dont la fonction d'appartenance est :

$$\mu_{B'}(y) = \sup_{x \in X} \min\left(\mu_A(x), \mu_B(y)\right) \tag{2.4}$$

Agrégateur des règles Avec les implications usuellement utilisées en contrôle flou, l'agrégation des règles s'effectue à l'aide d'une conorme triangulaire, généralement l'opérateur max. Ceci revient à considérer que les règles sont liées par un opérateur OU.

$$\mu_{B'}(y) = \max_{i=1,\ldots,m} \mu_{B'_i}(y) \tag{2.5}$$

où m est le nombre de règles activées parmi les N règles de la base des règles.

La défuzzification Le résultat de l'agrégation est la fonction d'appartenance d'un ensemble flou qui caractérise la décision de commande. Les actionneurs actuels, utilisés dans les boucles de commande ne s'accommodant pas à ce genre de décision,

il convient de transformer ce résultat en une grandeur de commande précise : C'est le but de la défuzzification. Les méthodes les plus couramment utilisées sont :

- la méthode de centre gravité qui donne :

$$u = \frac{\int_y y\mu_{B'}(y)dy}{\int_y \mu_{B'}(y)dy} \tag{2.6}$$

- La méthode du maximum, conduisant à :

$$u = \max_{y \in Y} \mu_{B'}(y) \tag{2.7}$$

- La méthode des moyennes des maximums, fournissant :

$$u = \sum_i y_j / J \tag{2.8}$$

où les y_j sont les valeurs telles que $\mu_{B'}(y_j)$ est maximum et J le nombre de maximums.

2.2.4 Un exemple de raisonnement flou en commande

Pour illustrer ces différentes étapes nous présentons la méthode de Mamdani qui repose sur l'utilisation de l'opérateur min pour la combinaison des prémisses et pour l'implication. L'agrégation des règles est réalisée par l'opérateur max, la défuzzification est effectuée par la méthode du centre de gravité. La figure 2.4 illustre le principe de cette méthode pour les deux règles suivantes :

R_1 :si ε est $ZERO$ et $\Delta\varepsilon$ est $ZERO$ alors u est$ZERO$

R_2 :si ε est $Positive-Petit$ et $\Delta\varepsilon$ est $Positive-Petit$ alors u est $Positive-Petit$

où ε désigne l'erreur et $\Delta\varepsilon$ la variation de l'erreur

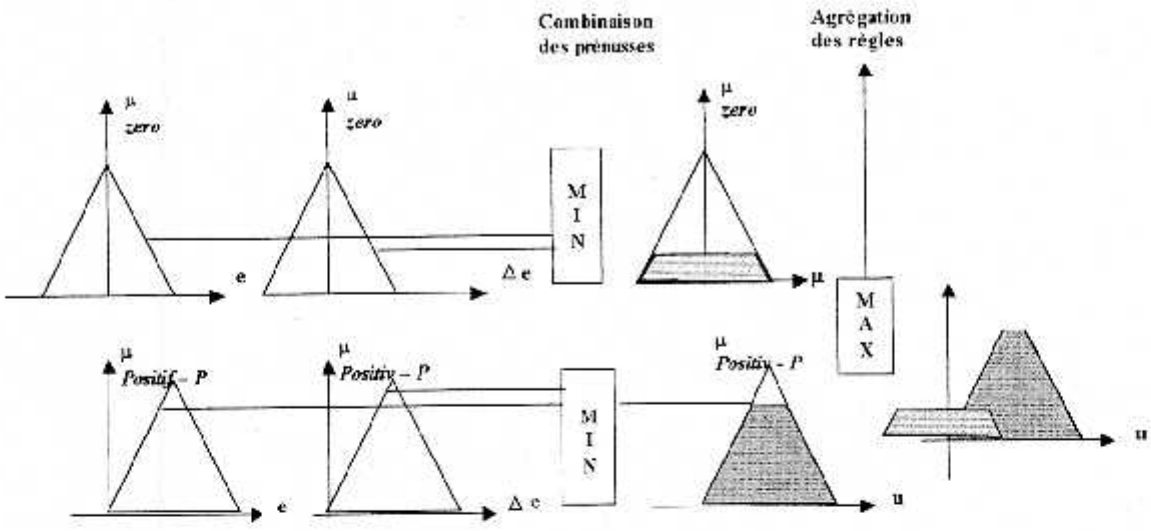

Fig 2.4 :La méthode de Mamdani

2.2.5 Analogie avec les contrôleurs classiques

Selon que la commande concerne la commande ou sa variation ,on peut établir une analogie avec les contrôleurs classiques de type PD et PI.

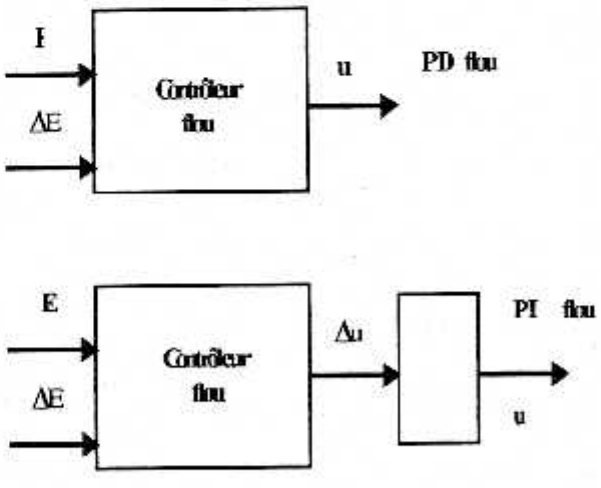

Fig 2.5 :Equivalents flous des contrôleurs usuels

2.2.6 le contrôleur flou placé dans une boucle de contrôle

Le contrôleur flou s'insère dans une boucle de contrôle représentée par la figure 2.6.

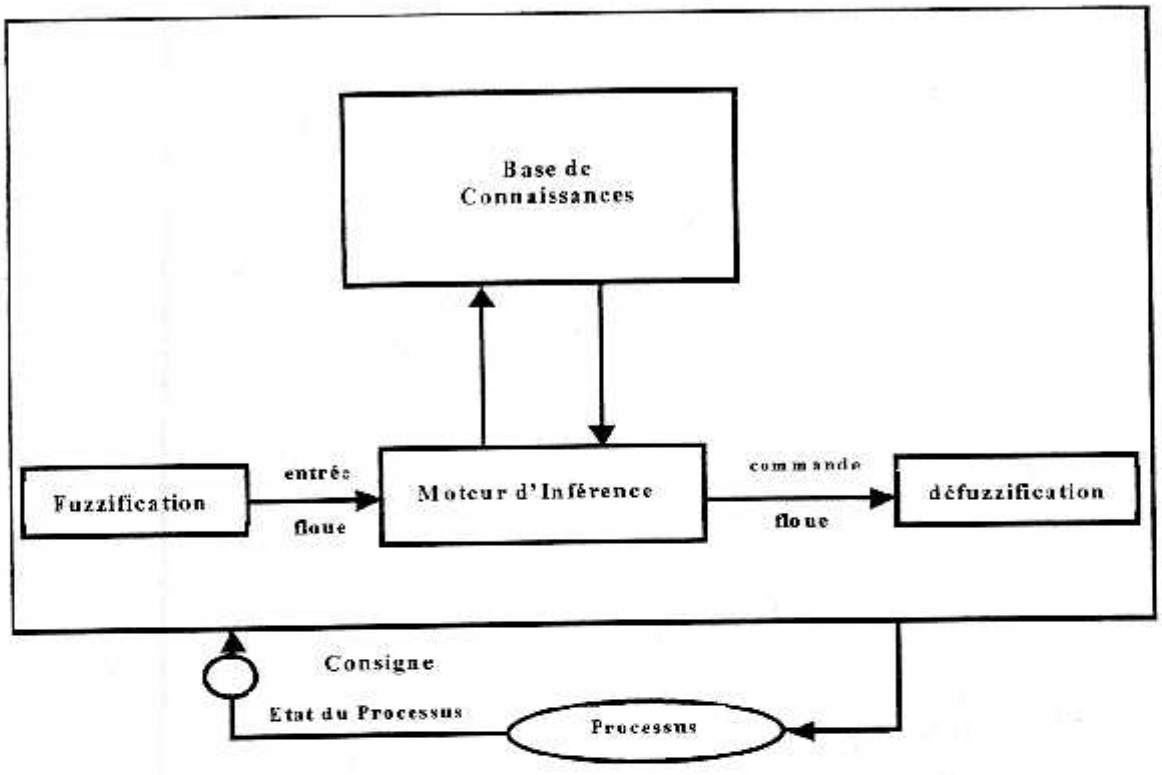

Fig 2.6 : Contrôleur flou placé dans une boucle de contrôle

La base de connaissances comprend une connaissance du domaine d'application et les buts de contrôle.

Elle est composée :

- d'une base de données fournissant des informations nécessaires utilisées pour définir l les règles de contrôle linguistiques et la manipulation des données floues pour le contrôleur.

- d'une base de règle caractérisant les buts de la politique de contrôle des experts du domaine.

Le moteur d'inférence est le noyau du contrôleur flou. Il simule la prise de décision de l'être humain en se basant sur les concepts flous et les règles d'inférences floues.

La construction d'une base des règles nécessite :

- un choix approprié des variables d'états du processus comme entrées et des variables de contrôle comme sorties.

- Un choix de l'ensemble des termes linguistiques associés à chaque variable.

- La dérivation des règles de contrôle.

- Les types de règles de contrôle.

2.3 Conclusion

Même si elle peut apparaître, à premiere vue, comme une démarche très heuristique, la commande floue repose sur une théorie et une méthodologie.

Elle n'est pas une alternative à l'automatique classique, elle est son complément dans des situations où un modèle mathématique n'existe pas ou est difficile à réaliser, mais où une expertise humaine est accessible.

Elle a pour avantage aussi par rapport à une approche classique d'intelligence artificielle, d'éviter tout effet de "seuillage" ce qui est essentiel dans les problèmes de commande. Cette élimination de l'effet de seuillage est due aux techniques de

défuzzification utilisées en commande.

Si la commande floue permet d'aborder des applications difficiles, elle présente aussi des limitations et des faiblesses. Une des faiblesses, générale aux démarches de l'intelligence artificielle, des systèmes experts, ou des systèmes à base de connaissances, réside dans le fait que le succès dépend de la phase d'acquisition de connaissances.L'intégration de techniques neuronales auxquelles nous consacrons le chapitre suivant dans les systèmes flous, permet une acquisition plus ou moins automatique des connaissances ou tout au moins une adaptativité du système, qui viendra corriger une absence d'information partielle.

Chapitre 3

Présentation théoriques des réseaux de neurones

3.1 Introduction

Dans l'état actuel de leur développement, les réseaux de neurones peuvent être décrits comme une tentative d'imiter le mode de fonctionnement du cerveau . Parler de réseau de neurones nécessite l'acquisition d'un vocabulaire emprunté au biologiste. C'est pour cela,qu' avant d'aborder la modélisation, il est impératif, de se familiariser avec ce vocabulaire.

3.1.1 Le neurone

Le neurone [18] (cellule nerveuse) se compose de trois parties : (Fig. 3.1).

- Le corps cellulaire : c'est le centre de la synthèse biologique. Il contient le

noyau.

- Les dendrites : Elles sont ramifiées et offrent une grande surface de réception.

- L'axone : C'est une excroissance qui véhicule le potentiel d'action. Il se divise à son extrémité pour distribuer les signaux dans plusieurs directions.

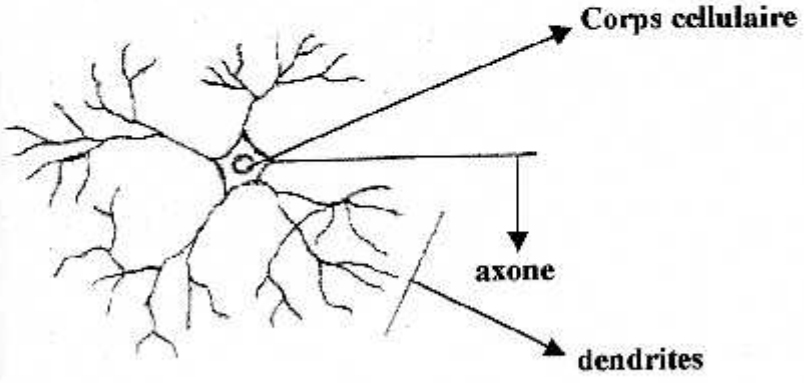

Fig 3.1 : Le neurone biologique

Communication

Les cellules nerveuses communiquent chimiquement au niveau des synapses. La cellule présynaptique (terminaison axonale d'un neurone A) est séparée de la cellule postsynaptique (dendrite d'un neurone B) par la fente synaptique.

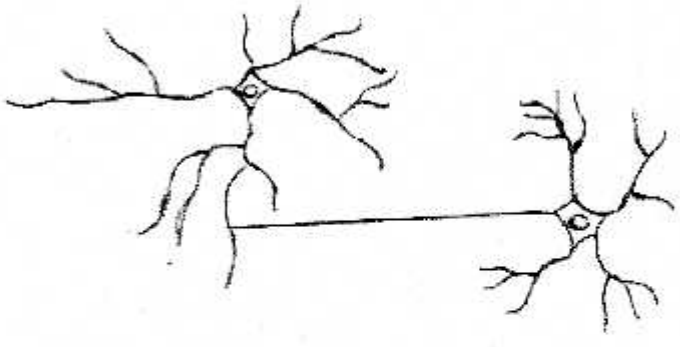

Fig 3.2 : Schéma d'une synapse

Le potentiel d'action

Toutes les cellules sont entourées d'une membrane plasmique. La différence de tension qui existe de part et d'autre de la membrane plasmique est le potentiel de la membrane. La charge est véhiculée de part et d'autre par des ions de sodium et de potassium à travers des canaux particuliers. Les canaux convertissent les signaux chimiques extra-cellulaires en signaux électriques : c'est le potentiel d'action.

3.1.2 Propriétés

L'adaptation

Un neurone stimulé continuellement pendant une longue période devient peu à peu moins sensible à un stimulus constant. C'est le phénomène d'adaptation qui permet à un neurone de réagir avec sensibilité à un changement en dépit d'une stimulation stable élevée.

L'influence mutuelle

Deux neurones voisins de quelques millimètres peuvent s'influencer. Les signaux d'une telle distance peuvent se propager passivement, c'est à dire, provoquer une atténuation.

Indépendance

Des parties distinctes de l'arbre dendritiques peuvent se comporter comme des voies plus ou moins indépendantes dans l'acheminnement de l'information et de la communication.

Plasticité

Dans le système nerveux central et périphérique, le motif des connexions synaptiques parait être plastique : l'expérience peut le façonner en stimulant ou réprimant l'activité électrique.

3.2 Modélisation

3.2.1 Le neurone formel

Le premier modèle de neurone formel a été présenté par Mac Culloch et Pitts en 1943 à l'université de Chicago.

Le modèle de Mac Culloch et Pitts

Une neurone formel (Fig 3.3) fait la somme pondérée des potentiels d'action qui lui parviennent puis s'active suivant la valeur de cette somme pondérée . Si la somme dépasse un certain seuil, le neurone est activé et transmet une réponse (sous forme de potentiel d'action) dont la valeur est celle de son activation. Si le neurone n'est pas activé, il ne transmet rien [18].

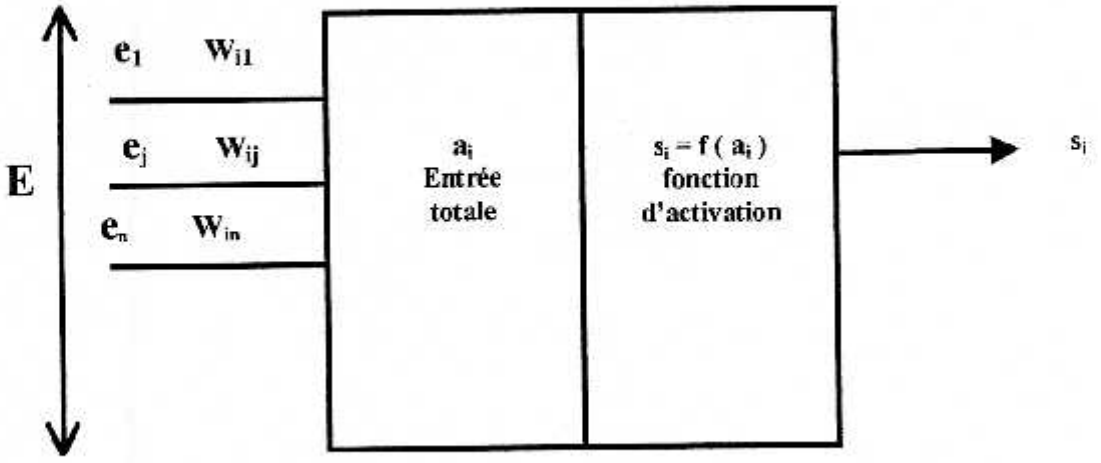

Fig 3.3 : Le neurone formel

Modélisation générale

D'une façon plus générale un neurone est un automate défini par :

a- La nature des entrées :

* binaire $(-1, +1)$ ou $(0, 1)$

* réelle.

b- La fonction d'entrée totale : qui définit le prétraitement effectuée sur les entrées:

Elle peut être:

* une fonction booléenne des entrées

* la somme pondérée des entrées

$$a^i = \sum_j w^{ij} e^j \tag{3.1}$$

* la somme pondérée des entrées seuillées

$$a^i = \sum_j w^{ij} e^j - \theta \tag{3.2}$$

* une fonction polynomiale des entrées :

exemple : $a^i = w^{01}e_1^2 + w^{11}e^1e^2 + w^{01}e^2 + w^{00}$

c- La fonction d'activation (Fig 3.4) du neurone qui définit son état interne en fonction de l'entrée totale :

Elle peut être :

* La fonction de Heaviside

$$\begin{cases} si \quad a^i > 0 \quad s^i = 1 \\ \text{sinon} \quad\quad\quad s^i = 0 \end{cases} \tag{3.3}$$

* La fonction signe

$$\begin{cases} si \quad a^i > 0 \quad s^i = 1 \\ sinon \quad\quad\quad s^i = -1 \end{cases} \tag{3.4}$$

* La fonction linéaire par morceaux ("Saturation")

$$\begin{cases} si\ a_i < s & s_i = s \\ a_i \in [s, S] & s_i = a_i \\ a_i > S & s_i = S \end{cases} \tag{3.5}$$

* La fonction sigmoïde

$$s_i = m(e^{ks} - 1)e^{ka} + 1 \tag{3.6}$$

* La fonction stochastique

$$\begin{cases} s_i = 1 \quad avec\ la\ probabilité \dfrac{1}{1 + \frac{\epsilon^{-a_i}}{T}} \\ s_i = 0 \quad sinon \end{cases} \tag{3.7}$$

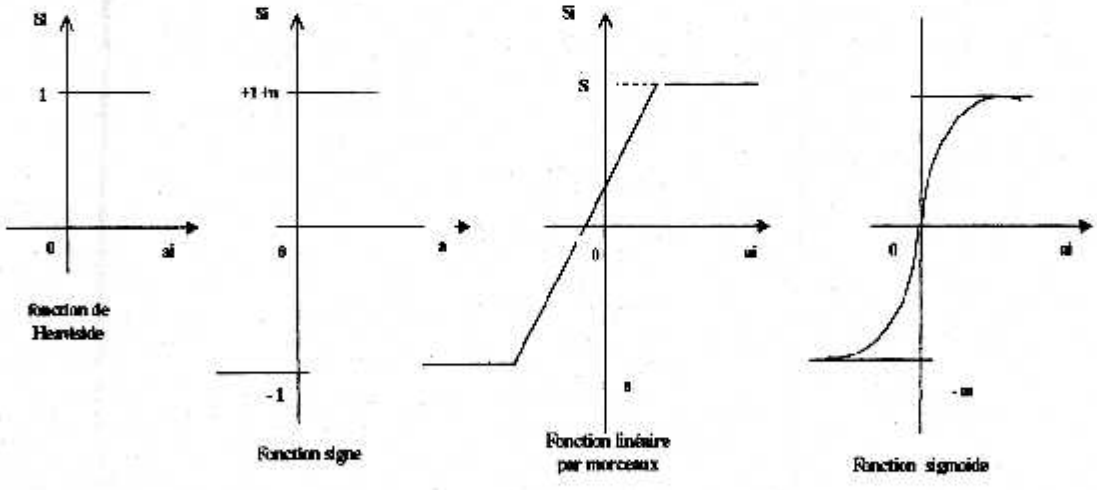

Fig 3.4 :Les fonctions d'activations

d- La fonction de sortie qui calcule la sortie du neurone en fonction de son état d'activation

e- La sortie du neurone.

3.2.2 Le réseau neuronal

C'est un réseau d'automates du type ci-dessus [30] [44] .

Il est spécifié par :

* Le nombre d'automates

* Le type d'automate

* L'architecture du réseau

* Le poids des connexions w_{ij}

* La règle d'évolution (dynamique des états)

* La règle d'apprentissage

3.2.3 La règle d'évolution

Le réseau neuronal est un ensemble de neurones formels reliés par des connexions. Le poids affecté à ces connexions correspond à la plasticité du neurone.

En 1985 Fogelman. Soulié a défini différentes dynamiques discrètes ou processus d'itération.

L'itération parallèle

Tous les automates réevaluent leur état périodiquement et simultanément .A partir d'un état initial $x(0) = x^0$ on a :

$$\forall t > 0,\ x(t) = F(x(t-1)). \tag{3.8}$$

L'itération séquentielle

Les automates changent d'état successivement selon un ordre déterminé . Pour chaque classe de permutation π de $\{1, ..., n\}$ on a :

$$\forall t > 0,\ x(t) = F^{\pi}(x(t-1)) \tag{3.9}$$

L'itération bloc séquentielle

Elle est définie pour chaque partition ordonnée

L'itération aléatoire

chaque automate réevalue son état à des intervalles de temps aléatoires.Ces modes peuvent être mixés dans le cas des réseaux multicouches, par exemple, parallèle à l'interieur d'une couche et séquentiel pour le passage d'une couche à une autre.

3.2.4 L'apprentissage

C'est la phase pendant laquelle la connaissance est encodée dans les connexions [23]. L'apprentissage peut être supervisé ou non.

Apprentissage supervisé

La présentation d'un prototype en entrée donne lieu à une réponse du réseau, déterminée par la règle d'évolution du réseau.

La supervision de l'apprentissage s'effectue par comparaison de ce résultat avec le résultat souhaité. L'apprentissage supervisé proprement dit consiste à minimiser une fonction de coût en modifiant les poids.

Apprentissage non supervisé

Il y a absence de professeur. C'est la structure du réseau qui crée des groupements. Le réseau se relaxe jusqu'à un état stable. En fin d'apprentissage, une cellule (ou groupements de cellules) est plus active que les autres et correspond à une classe.

Apprentissage par renforcement

L'apprentissage par renforcement trouve son opportunité lorsque l'on ne dispose pas des données nécessaires à l'apprentissage supervisé, ou que celles ci soient coûteuses à obtenir, mais que l'on dispose d'une information permettant de donner une certaine évaluation des solutions obtenues. le système est informé sur l'action choisie et son effet. Cette action est à renforcer si elle conduit à une amélioration des performances

du système entrainé.

Les différentes règles d'apprentissage

Les règles les plus utilisées sont :

a- La règle de Hebb

Si deux neurones connectés entre eux sont activés au même instant, la connexion qui les relie doit être renforcée .

Si à un instant t le réseau soumis à un stimulus, provoque l'activation s_i du neurone i,et l'activation s_j du neurone on a :

$$w_{ij} \leftarrow w_{ij} + k s_i s_j \tag{3.10}$$

où k est le paramètre d'intensité d'apprentissage .

b- La règle du perceptron

C'est une procédure erreur/correction dont il existe de nombreuses variantes :

e_j :stimulus d'entrée

D_i : sortie désirée

S_i :sortie calculée

$$w_{ij} \leftarrow w_{ij} + (s_i - S_i)e_j \tag{3.11}$$

c- La règle de Widrow et Hoff

La correction porte sur la différence entre la sortie désirée et la somme pondérée des entrées non seuillées

$$ej wij \leftarrow wij + k(si - \sum_j wij)ej \tag{3.12}$$

d- La règle stochastique

$$wij \leftarrow wij + \beta \left(pij - p'ij\right) \tag{3.13}$$

avec

pij : probabilité que i et j soient activées en mode forcé

$p'ij$: probabilité que i et j soient activées en mode libre

e- La rétropropagation du gradient

Elle consiste en une rétropropagation du côut quadratique

$$c(w) = \|S - D\|^2 \tag{3.14}$$

$$uij \leftarrow wij - \Gamma disi \tag{3.15}$$

Γ : vitesse d'apprentissage.

A partir de ces définitions, plusieurs modèles ont été proposés : Le perceptron, le modèle de Hopfield, le modèle de Kohonen et le modèle multicouches. Nous n'examinerons pas le modèle de Hopfield ni celui de Kohonen car cela sortirait du cadre de notre travail. Nous nous interesserons par contre au perceptron car ce fût le premier grand modèle en évolution permanente et au modèle multicouches puisqu'il est le modèle le mieux adapté à la commande de processus.

3.3 Le perceptron

3.3.1 Description

Le perceptron (Fig 3.5) est un système adaptatif proposé par Rosenblatt en 1961.

Il se compose de trois types de cellules :

- Les cellules d'entrée qui constituent la rétine
- Les cellules d'apprentissage
- Les cellules de sortie .

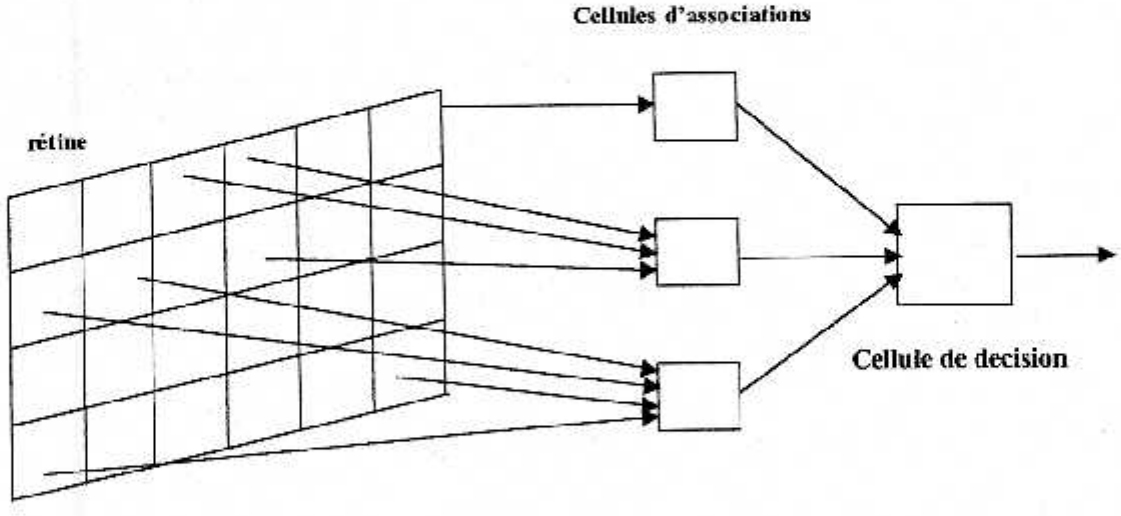

Fig 3.5 :Le perceptron simple

3.3.2 La règle de perceptron

L'apprentissage du perceptron est un apprentissage supervisé qui se fait par erreur/correction suivant le tableau ci dessous :

Sortie désirée	Sortie calculée	Poids
1	1	w_i inchangé
0	0	w_i inchangé
0	1	si $e_i > 0 \Rightarrow w_i \downarrow$
		si $e_i < 0 \Rightarrow w_i \uparrow$
1	0	si $e_i > 0 \Rightarrow w_i \uparrow$
		si $e_i < 0 \Rightarrow w_i \downarrow$

$$w_i \leftarrow w_i + k(d - s)e_i \tag{3.16}$$

3.4 Le réseau multicouches

Les réseaux à couches actuels (Fig 3.6) sont directement issus du perceptron [44]. Vu la diversité des applications déjà opérationnelles, ce sont probablement les plus populaires.

3.4.1 Le neurone

C'est un neurone de type classique qui applique une fonction non linéaire à la somme pondérée des entrées. Généralement on choisit la fonction sigmoïde.

3.4.2 Le réseau

Le réseau à couches est composé de trois types de cellules :

- Les cellules d'entrée, l'état est fixé par l'extérieur
- Les cellules de sortie, sans intéraction avec l'extérieur
- Les cellules de sortie.

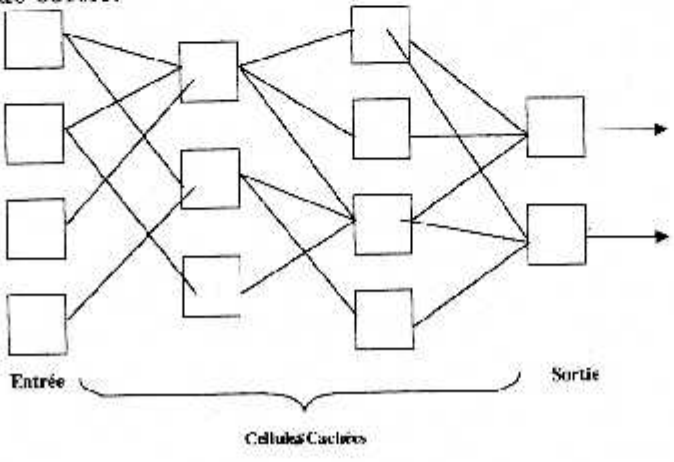

Fig 3.6 :Le réseau multicouches

La règle d'évolution :

$$si = f(ai)) \qquad \text{avec} \qquad ai = \sum_j wij\, sj \tag{3.17}$$

3.4.3 L'apprentissage

Dés 1961, Rosenblatt a proposé un perceptron à couches et Minsky et Papert en 1969 ont montré que la solution de certains problèmes passait par la multiplication des couches.

Malheureusement à cette époque, aucune règle d'apprentissage ne pouvait fonctionner sur plusieurs couches. Ce sont des équipes en France (Fougelman Soulie, Gallinarri, Lecun) et aux USA (Rumelhart, Hilton et Williams) qui ont débloqué la situation grâce à l'algorithme de la rétro-propagation de gradient [45].

Principe de l'algorithme de la rétropropagation du gradient

Le principe consiste à présenter en forçant les cellules d'entrée un vecteur d'entrée E^h parmi les p prototypes. La propagation en vertu de la règle $s^i = f(a^i))$ avec $a^i = \sum_j w_{ij} s^j$ donne la sortie calculée S^h. Un vecteur de sortie D^h est présenté sur les cellules de sortie. L'algorithme de la rétropropagation du gradient minimise une fonction de côut quadratique :

$$c(w) = \frac{1}{p} \sum_h C^h \tag{3.18}$$

avec

$$C^h = \left\| S^h - D^h \right\|^2 \tag{3.19}$$

$$C^h = \sum \left(S_q^h - D_q^h \right)^2 \tag{3.20}$$

q est un indice de sortie.

Cette erreur calculée sur les sorties va être propagée de manière rétrograde sur les couches en amont.

Modification des poids Les poids sont modifiés de la manière suivante :

$$w_{ij}(h) = w_{ij}(h-1) - \Gamma(h)\frac{\partial C^h}{\partial w_{ij}} \tag{3.21}$$

où Γ designe la constante d'apprentissage.

Or

$$\frac{\partial C^h}{\partial w_{ij}} = \frac{\partial C^h}{\partial a_i}\frac{\partial a_i}{\partial w_{ij}} \tag{3.22}$$

avec a_i entrée totale : $a_i = \sum_j w_{ij}s_j$ et $s_i = f(a_i)$

donc

$$\frac{\partial C^h}{\partial w_{ij}} = \frac{\partial C^h}{\partial a_i}s_j \tag{3.23}$$

on note $d_i = \frac{\partial C^h}{\partial a_i}$ donc

$$\frac{\partial C^h}{\partial w_{ij}} = d_i s_j \tag{3.24}$$

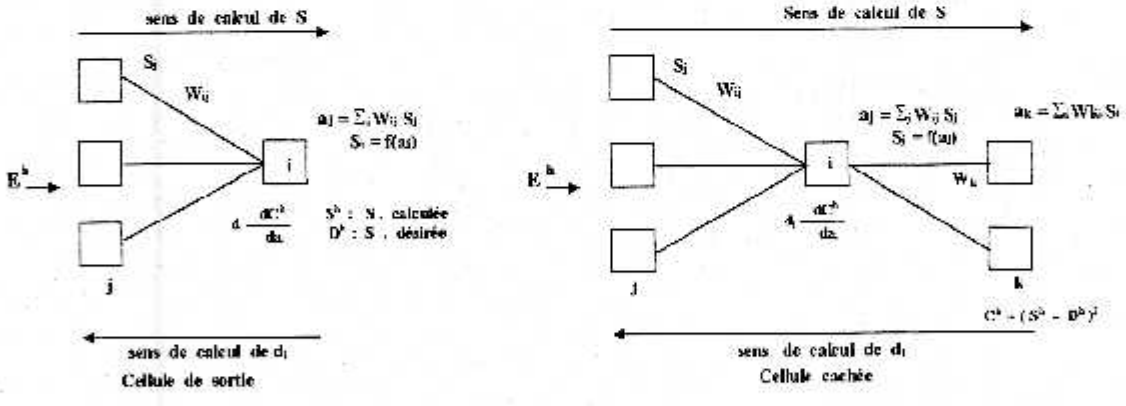

fig 3.7 :Chaines des dépendances dans un réseau à couches

* si i est une cellule de sortie :

$$C^h = \sum_i \left(S_i^h - D_i^h\right)^2 \tag{3.25}$$

donc

$$d_i = 2\left(S_i^h - D_i^h\right)\frac{\partial S_i^h}{\partial a_i} \tag{3.26}$$

$$d_i = 2\left(S_i^h - D_i^h\right) f'(a_i) \tag{3.27}$$

*si i est une cellule cachée suivie de cellule k ($w_{ik} \neq 0$ et dk connu)

$$d_i = \sum_k \frac{\partial C^h}{\partial a_k}\frac{\partial a_k}{\partial a_i} \tag{3.28}$$

$$d_i = \sum_k d_k \frac{\partial a_k}{\partial a_i} \qquad \text{or} \qquad a_k = \sum_i w_{ki} s_i \tag{3.29}$$

$$d_i = \sum_k d_k \frac{\partial a_k}{\partial s_i}\frac{\partial s_i}{\partial a_i} \tag{3.30}$$

$$d_i = \sum_k d_k w_{ki} f'(a_i) \tag{3.31}$$

Par conséquent :

$$w_{ij}(h) = w_{ij}(h-1) - \Gamma(h) d_i S_j^h \tag{3.32}$$

Algorithme de la rétropropagation du gradient :

Debut

Pour chaque ensemble E^h

faire

- propager les états

  $s^i = f(a^i)$;

- Comparer la sortie calculée S^h

  et la sortie désirée D^h

  en deduire l'erreur

  $\Delta_i^h = (S_i^h - D_i^h)$

- rétropropager l'erreur

  si $i \in$ *couche de sortie*

  alors $d^i = 2\Delta_i^h f'(a^i)$

  sinon $d^i = (\sum_k w^{ki}(h) d^k) f'(a^i)$

  fsi;

- Modifier les poids

  $wij(h) = wij(h-1) - \Gamma(h) d^i S_j^h$

Fait;

Fin.

Discussion de l'algorithme [13]

* L'apprentissage nécessite plusieurs passes de la base d'exemples .

* Test d'arrêt :

En pratique, il est difficile d'accomplir la terminaison automatique de l'algorithme.On utilise diverses conditions d'arrêt :

a- Un seuil sur la fonction de coût au dessous duquel on arrête l'apprentissage

b- Un seuil sur le taux de succés sur la base d'exemples au dessus duquel on arrête l'apprentissage

c- Un nombre d'itérations préfixé.

*Architecture du réseau :

En général, on ne connait pas la taille du réseau pour un problème donné.Si le nombre de couches cachées ou le nombre de neurones est trop petit, le réseau n'apprend pas le problème, s'il est par contre trop grand, il n'est pas capable de généraliser.

Une méthode consiste à démarrer avec un réseau trés petit, puis ajouter les neurones jusqu'a atteindre la bonne performance.Une autre consiste à démarrer avec un réseau large, puis éliminer les neurones en plus.

3.5 Intérêt des réseaux neuronaux

3.5.1 Le parallèlisme

Les réseaux de neurones se classent dans les architectures cellulaires décentralisées, c'est à dire que l'on est en présence d'unités de traitement relativement simples opérant en même temps.

3.5.2 La tolérance aux pannes

Le parallèlisme intrinsèque des réseaux neuronaux est d'autant plus interessant qu'il est associé à la distribution des connaissances. La mémoire est distribuée sur tout le réseau, ce qui offre une grande tolérance aux pannes.

Un neurone absent ou une connexion détruite ne géne quasiment pas le fonctionnement de l'ensemble. Cela n'implique pas la perte d'informations mais simplement modifie la carte de répartition de la connaissance.

3.5.3 Capacité d'apprentissage

La connaissance n'est plus introduite par des règles ou par modélisation mais, par la présentation d'une succession d'exemples. Cela donne une grande adaptabilité au réseau et lui permet de tenir compte de contraintes exterieures nouvelles (par exemple du bruit). Cette adaptabilité se caractérise dans certains réseaux par leur capacité d'auto-organisation qui assure leur stabilité en tant que système dynamique.

3.5.4 Propriété de généralisation

cette propriété de généralisation semble intéressante pour le développement des systèmes experts. On est face à un système qui permet à partir d'exemples apprendre et retrouver des règles sous-jacentes ou mimer des comportements qui permettent de résoudre un problème.

3.5.5 Conclusion

Une solution avec les réseaux de neurones doit être envisagée à chaque fois que pour une application on dispose d'une grande quantité d'exemples, avec les réponses attendues, sans pour autant avoir une modélisation solide pour obtenir des réponses.

Il est à noter que le choix des paramètres (nombre de couches, nombres de neurones, conditions initiales des poids) agissant sur le système n'est pas chose aisée.

Pour tirer le plus grand profit des réseaux de neurones, une technique consiste à y adjoindre les propriétés intéressantes de la logique floue donnant ainsi naissance aux systèmes neuro flous auxquels sera consacré le chapitre suivant.

Chapitre 4

Modélisation neuronale d'un système d'inférence floue

4.1 Introduction

La justification d'associer la logique floue aux réseaux s'explique par la volonté de juxtaposer les propriétés intéressantes de l'imprécis du vague ou de l'incertain des systèmes flous à des propriétés d'apprentissage des réseaux de neurones. Cet apprentissage est d'autant plus facilité que l'on part d'une information structurée contenue dans des règles de productions.

4.2 Méthodes neuro-floues

Dans la littérature, l'utilisation conjointe ces méthodes neuronales et floues a été proposée sous diverses formes allant de la fusion des deux types de méthodes à l'utilisation séquentielle de l'une et de l'autre [12].

Le premier type d'association, le plus répandu est représenté dans le cas où un système d'inférence floue est mis sous la forme d'un réseau multicouches dans lequel les poids synaptiques correspondent aux paramètres du système, l'architecture du réseau dépendant des types de règles, des méthodes d'inférences, d'agrégation et de déffuzification choisis (fig4.1).

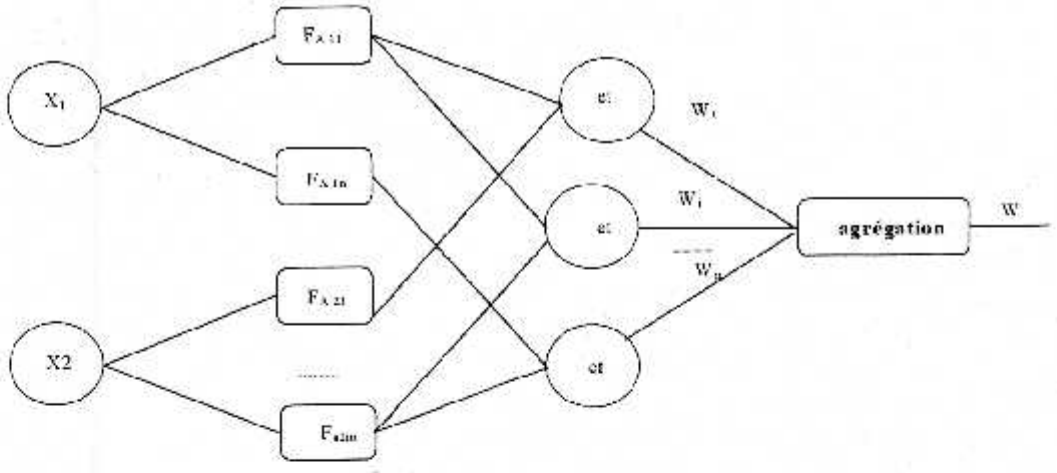

Fig 4.1 : Exemple de réseau neuronal pour la commande floue

Les fonctions d'appartenances intervenant dans les règles sont considérées comme des paramètres, ajustés par l'intermédiaire des poids entrant dans la première couche cachée, les conclusions des règles sont également des paramètres ajustables par l'intermédiaire des poids associés à la dernière couche.

Le second type d'association entre réseaux de neurones et systèmes flous correspond à l'utilisation des réseaux de neurones pour remplacer chacune des composantes d'un sytème d'inférence floue. De tels réseaux servent à l'apprentissage des fonctions d'appartenances, au calcul d'inférence et à la réalisation de la phase d'agrégation et de défuzzification.

Le troisieme grand type d'association correspond à l'utilisation des réseaux de neurones et des systèmes flous organisés en série ou en parallèle. Cette approche est à envisager lorsque le problème nécessite un traitement à differents niveaux. On peut tout d'abord construire un réseau de neurones qui fonctionne en amont d'un système d' inférence floue (Fig 4.2) où les variables d'entrées d'un système flou sont par exemple déterminées à partir de la sortie d'un réseau lorsqu'elles ne sont pas mesurables directement.

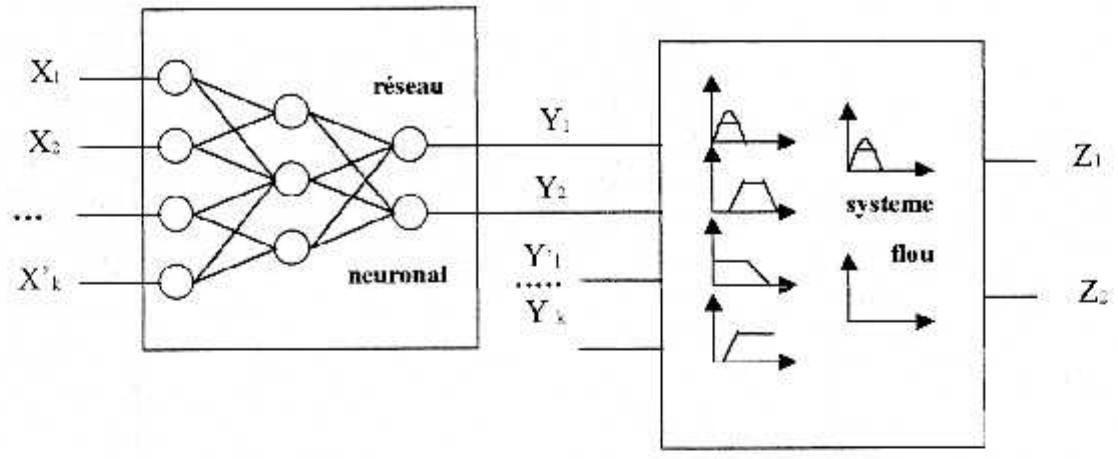

Fig 4.2 : Exemple d'association en série d'un réseau neuronal et d'un système d'inférence floue

On peut aussi avoir recours à un réseau de neurones qui fonctionne en aval d'un système flou (Fig 4.3) par exemple dans le but d'ajuster les sorties d'un système de commande floue à de nouvelles connaissance obtenues, les variables d'entrée étant l'ensemble de celles du système flou et des nouvelles obtenues, les variables de sortie étant les erreurs sur les variables de sortie du système flou .

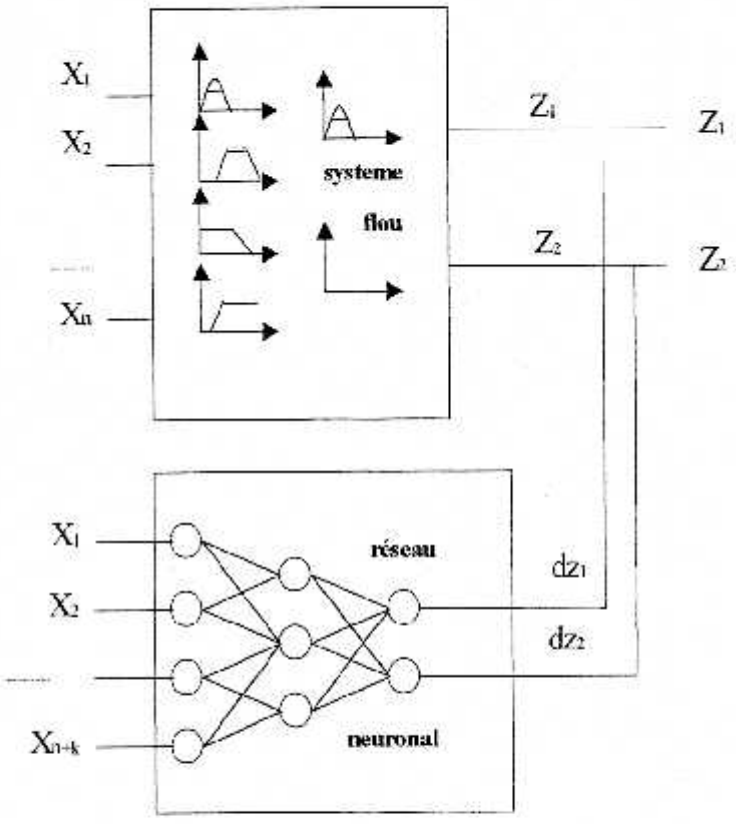

Fig 4.3 : Exemple d'association en parallèle d'un réseau neuronal et d'un système d'inférence floue

4.2.1 modélisation neuronale d'un système d'inférence floue

La formalisation des différentes étapes de processus de contrôle flou amène à observer un système d'inférence floue comme une fonction de deux familles de paramètres $P = \{P_1, P_2\}$ où P_1 est caractérisé par le choix du raisonnement flou, soit ces différents opérateurs (T-normes et T-conormes) et P_2 par la définition des fonctions d'appartenances. La modélisation floue consiste à déterminer les paramètres optimaux correspondant à P^* soit P_1^* et P_2^* .

L'apprentissage d'un contrôleur nécessite d'une part de déterminer la forme des ensembles flous associés aux prémisses et conclusions du contrôleur, et d'autre part de déterminer quelles conclusions doivent être associées à quelle prémisses.

Il s'agit donc d'extraire, d'une part, le sens numérique absolu des symboles employés en prémisses et conclusions (partitionnement flou des référentiels d'entrée et de sortie).et d'autre part la sémantique relationnelle liant ces symboles.

Une architecture neuro-floue à cinq couches suffit à mettre en oeuvre un contrôleur flou [24]. Les cellules composant cette architecture sont appelées neurones flous car elles ne réalisent pas une classique somme pondérée mais effectue plutôt les opérations spécifiques telles que : calcul d'appariement entre entrées et prémisses, propagation d'inférence entre prémisses et conclusions et défuzzification .

Les paramètres de cette architecture (forme des ensembles flous des prémisses et conclusions, matrice de connexions entre prémisses et conclusions) sont les poids synaptiques inter couches.

4.2.2 Conception d'un contrôleur neuroflou

Un contrôleur neuroflou peut être considéré comme la mise en place d'un algorithme de commande d'un processus déterminé.Cet algorithme décrit par des régles de commande floues est implémenté comme une relation floue entre les prémisses et les conclusions [53].C'est à partir de ces régles, des fonctions d'appartenances, des méthodes d'apprentissages et des données réelles (faits observés ou mesurés) que le contrôleur déduit par inférence la ou les sorties correspondantes.

L'organigramme de la figure 4-4 décrit les démarches principales à suivre lors de la conception d'un contrôleur flou.

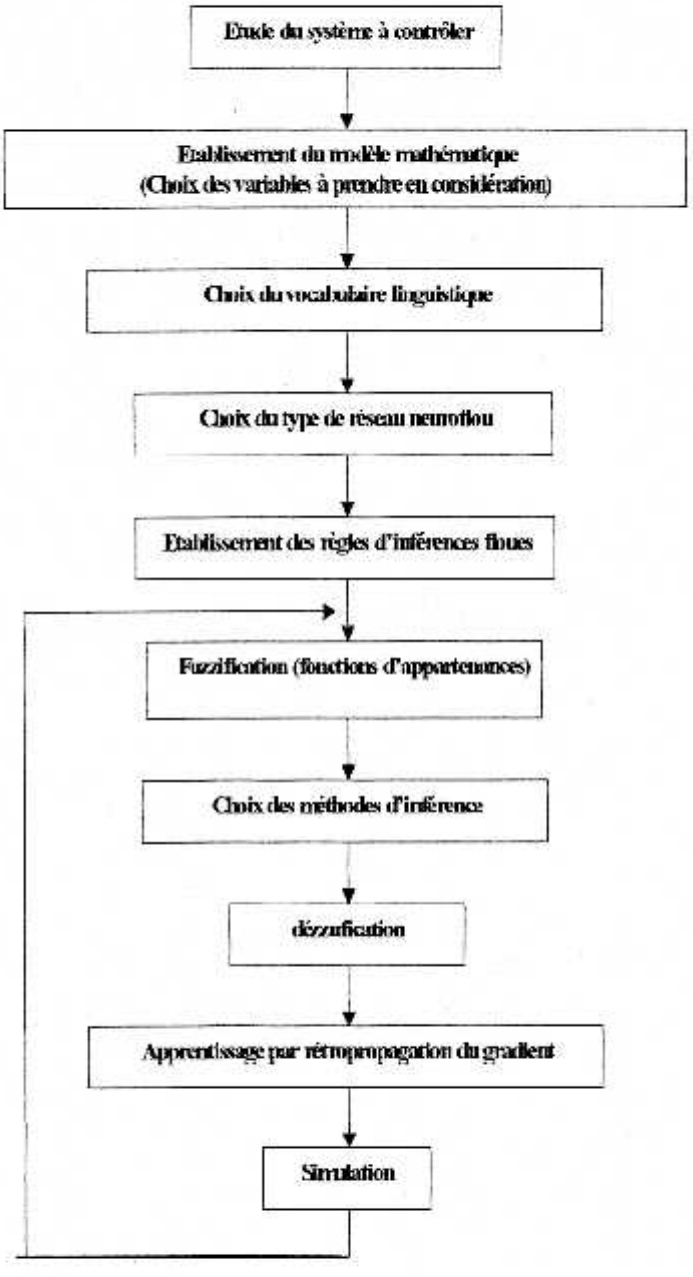

Fig 4.4 : Etapes de conception d'un contrôleur flou

4.3 présentation de la méthode d'apprentissage

4.3.1 Introduction

L'utilisation des capacités d'apprentissage des réseaux de neurones pour l'identification des paramètres optimaux d'un système d'inférence floue a été étudiée par plusieurs auteurs et différentes architectures ont été proposées dans ce cadre. La performance des contrôleurs flous est liée à deux facteurs importants : la solidité des techniques d'acquisition des connaissances et la disponibilité d'experts humains.

Ces deux facteurs nous ont conduit à opter dans notre étude pour l'architecture neuro-floue dénotée SIFANE [27] (système d'inférence floue à architecture neuronale) où les contrôleurs sont dotés d'une capacité d'auto-apprentissage pour atteindre les objectifs de contrôle d'une manière optimale.Cette architecture est construite autour du concept de réseaux adaptatifs dont nous rappelons les caractéristiques avant de présenter un SIFANE .

4.3.2 Les réseaux adaptatifs

Un réseau adaptatif est (Fig4.5) est un réseau multicouches avec rétroaction dans lequel chaque neurone exécute une fonction particulière (fonction d'activation) sur les signaux d'entrées en utilisant un ensemble de paramètres spécifiques à ce neurone [44]. La forme des fonctions d'activations peut varier d'un neurone à un autre du processus que le réseau adaptatif est désigné pour implémenter.

Pour refléter les capacités d'adaptativité de ces réseaux, nous utiliserons des

cercles et des carrés pour représenter les neurones.

Un neurone représenté par un carré est un neurone adaptatif c.à.d possèdant des paramètres susceptibles d'être modifiés tandis qu'un neurone représenté par un cercle (neurone fixe) n'en a pas. L'ensemble des paramètres d'un réseaux adaptatif est l'union des ensemble de paramètres de chaque neurone adaptatif.

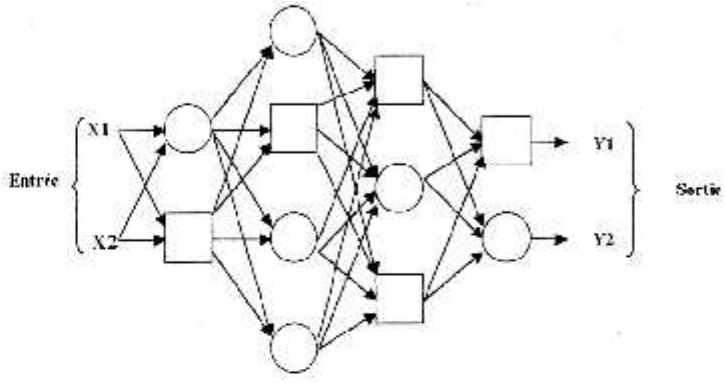

Fig 4.5 : Réseau adaptatif

Pour suivre un modèle d'entrée-sortie désiré, ces paramètres sont mis à jour en accord avec des exemples d'apprentissages présentés au réseau et une procédure de descente du gradient introduite au chapitre 3 .

Supposons que le réseau adaptatif considéré possède L couches et que la kième couche possède k neurones.

On peu noter le neurone qui se trouve à la ième position de la kième couche par (k,i) et sa fonction d'activation par O_i^k. Comme la sortie du neurone dépend des signaux d'entrée et d'un ensemble de paramètres on a :

$$O_i^k = O_i^k\left(O_i^{k-1}, ..., O_{k-1}^{k-1}, a, b, c\right) \tag{4.1}$$

où a, b, c sont les paramètres de ce neurone.

Si P exemples d'apprentissages sont présentés au réseau, on peut définir l'erreur quadratique par :

$$E_p = \sum_{m=1}^{L} \left(T_{m,p} - O_{m,p}^{L}\right)^2 \tag{4.2}$$

où $T_{m,p}$ est le m^{ieme} composant du vecteur des sortie désirée et $O_{m,p}^{L}$ est le m^{ieme} composant du vecteur de sortie calculée. On en déduit une erreur globale :

$$E = \sum_{p=1}^{P} E_p \tag{4.3}$$

Pour développer une procédure de mise à jour en implémentant une descente du gradient sur l'erreur globale E à travers l'espace des paramètres, nous avons à calculer l'erreur $\frac{\partial E}{\partial O}$ pour chaque exemple p et pour chaque sortie du neurone O

a- Le signal d'erreur pour un neurone appartenant à la couche de sortie L est calculé de la manière suivante :

$$\frac{\partial E_p}{\partial O_{i,p}^{L}} = -2\left(T_{i,p} - O_{i,p}^{L}\right) \tag{4.4}$$

b- Le point essentiel consiste à savoir calculer l'erreur pour les neurones des couches cachées. La méthode est basée sur la propagation des erreurs de sortie au sens rétrograde des neurones de sorties vers les neurones d'entrées couche par couche, ce qui assigne à chaque neurone le but de minimiser les erreurs propagées d'où :

Pour un neurone i de la k^{ieme} couche cachée, le signal d'erreur peut être calculé par :

$$\frac{\partial E_p}{\partial O_{i,p}^{k}} = \sum_{m=1}^{k+1} \frac{\partial E_p}{\partial O_{m,p}^{k}} \frac{\partial O^{k+1}}{\partial O_{i,p}^{k}} \qquad \text{où} \qquad 1 \leqslant k \leqslant L-1 \tag{4.5}$$

L'erreur est exprimée sous la forme de combinaison linéaire des erreurs des neurones de la couche suivante. Par conséquent pour tout $1 \leqslant k \leqslant L$et $1 \leqslant i \leqslant k$, on peut calculer $\frac{\partial E_p}{\partial O_{i,p}^k}$ par (4.5) et (4.6).

c- si α est un paramètre de réseau adaptatif considéré on a :

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha} \tag{4.6}$$

où S est ensemble des neurones dont la sortie dépend directement de α, d'où l'erreur globale E en fonction de α :

$$\frac{\partial E_p}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha} \tag{4.7}$$

Par conséquent la mise à jour pour un paramètre α est :

$$\alpha = -\eta \frac{\partial E}{\partial \alpha} \tag{4.8}$$

où η est la constante de propagation du paramètre α.

En pratique η est exprimée par :

$$\eta = \frac{S}{\sqrt{\sum_\alpha \left(\frac{\partial E}{\partial \alpha}\right)^2}} \tag{4.9}$$

où S est la longueur de chaque transition de gradient (Step size).

4.3.3 Présentation d'un SIFANE

Dans ce qui suit, pour des raisons de simplicité nous supposons que le système d'inférence floue possède deux entrées x et y et une sortie z et que la base des

règles contient deux règles de type Takagi et Sugeno où la sorte est une combinaison linéaire des entrées[27].

L'architecture correspondante est représentée par la figure 4.6

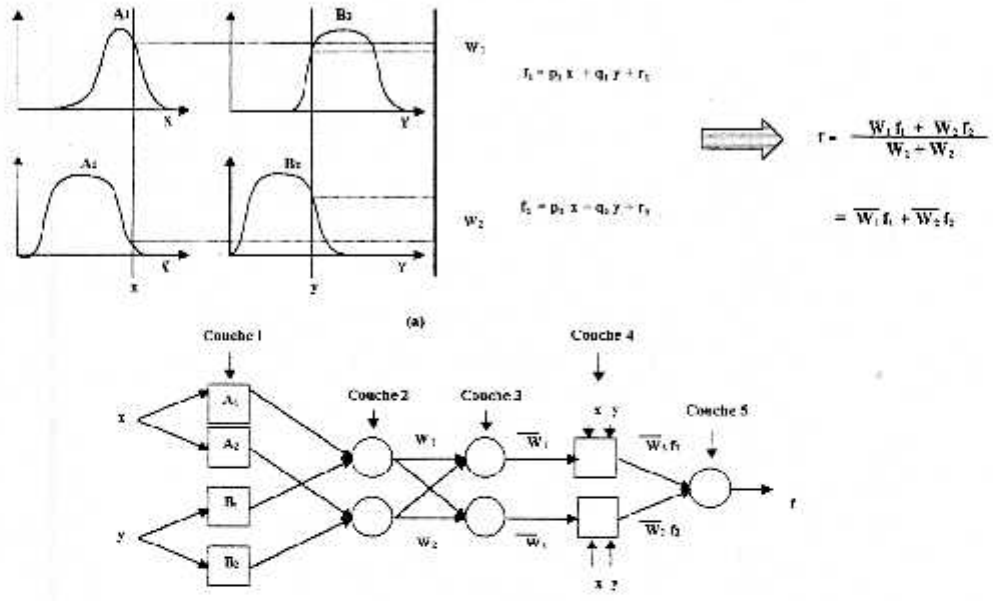

Fig 4.6 : (a) :système d'inférence flou .(b) :SIFANE équivalent

Chaque couche de ce réseau réalise une fonction particulière du système

- Couche 1: Chaque neurone de cette couche calcule le degré d'appartenance aux différents sous ensembles flous sur les domaines de chaque entrée.

Il a pour fonction d'activation :

$$O_i^1 = \mu_{A_i}(x) \tag{4.10}$$

où x est une variable d'entrée et A_1 un label linguistique que x peut prendre. Le neurone de cette couche calcule le degré avec lequel x satisfait A_1. Il a une entrée et il distribue sa sortie à toutes les règles utilisant le clause."x est A_i". En pratique

on choisit :

$$\mu_{A_i}(x) = \frac{1}{1 + \left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}} \tag{4.11}$$

où a_i, b_i et c_i sont les paramètres associés à cette fonction. L'ajustement pendant l'apprentissage des paramètres entraîne une variation de la forme de la fonction d'appartenance associée au label linguistique.Le nombre de neurones de cette couche est égal au nombre de fonctions d'appartenances des deux entrées.

- Couche 2: C'est la couche de conjonction entre tous les antécédents dans une règle.Chaque neurone effectue le produit des degrés d'appartenance de la partie prémisse des règles fournissant ainsi le poids de la règle .

Pour une règle k :

Si x est A_1 et y est A_2 alors $Z = ...$

$$\mu_k = \mu_{A_1}(x).\mu_{A_2}(x) \tag{4.12}$$

Le nombre de neurone de cette couche est égale au nombre de règles. On peut utiliser un autre opérateur pour la conjonction .

- Couche 3: Le neurone de cette couche calcule le rapport entre le poids de la ième règle et la norme des poids des autres règles. En d'autres termes, les neurones de cette couche calcule le facteur de normalisation de chaque règle k :

$$\overline{\mu_k} = \frac{\mu_k}{\sum_{i=1}^{n} \mu_i} \tag{4.13}$$

- Couche 4: Chaque neurone i de cette couche est connecté à un seul neurone i

de la couche précédente ; il calcule la sortie de la règle (k) par relation :

$$O_i^4 = \overline{\mu_k}.\left(\overset{k}{\alpha 0} + \overset{k}{\alpha 1}x + \overset{k}{\alpha 2}y\right) \tag{4.14}$$

où $\overline{\mu_k}$ est la sortie de la couche 3 et $\left\{\overset{k}{\alpha 0}, \overset{k}{\alpha 1}, \overset{k}{\alpha 2}\right\}$ sont les paramètres de conséquences.

- Couche 5: Il a un seul neurone dans cette couche, il effectue une sommation des sorties de la couche 4 selon :

$$\mu^* = \sum_{i=1}^{n} \overline{\mu_i}\mu_i \tag{4.15}$$

Il caractérise la procédure de défuzzification.

Tel qu'il a été modélisé, on peut établir une écivalence entre un système d'inférence flou et un réseau de neurones à cinq couches .Les paramètres ce cette architecture seront ajustés par une approche basée sur la descente du gradient .

4.4 La rétropropagation temporelle

Dans ce paragraphe, nous décrivons d'une manière détaillée la méthode d'apprentissage adoptée tout en essayant d'apporter une justification à ce choix durant l'étude.

Pour contrôler la trajectoire induite par le système dynamique, l'idée de base est d'implémenter le contrôleur flou et le système à travers différentes phases comme un réseau adaptatif. La succession des blocs (contrôleur neuro flou-système) qu'on appelera SAN (Stage adaptative network) à des instants donnés définira le réseau adaptatif correspondant à la trajectoire. L'algorithme de la rétropropagation du gradient sera appliqué à travers ces différentes phases et une correction de la matrice synaptique sera effectuée à chaque itération. Cette méthode d'apprentissage est appelée Méthode de Rétropropagation Temporelle [26],[27].

4.4.1 Implémentation d'un SAN

Comme nous l'avons mentionné, un SAN (Fig 4.7) est constitué de deux sous réseaux, l'un correspondant au contrôleur neuroflou qu'on appelera par la suite CNF et l'autre correpondant au système à contrôler qu'on appelera RESSYS.

Nous supposons que les variables d'états sont accessibles avec précision et que le passage à travers le controleur est trés court.

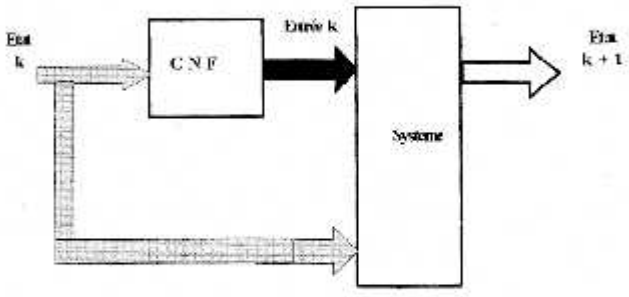

Fig 4.7 : Représentation d'un SAN

Le point du départ pour l'implémentation d'un SAN est de définir les modèles mathématiques du contrôleur neuroflou et du système :

le contrôleur neuro-flou

Le CNF sera implémenté sous forme d'un réseau adaptatif. Pour un système à p entrées, deux possibilités s'offrent à nous, la première est de l'implémenter sous forme de p SIFANES tandis que la seconde est de l'implémenter sous forme d'un SIFANE avec des règles à conséquences mutiples. Nous avons opté pour la première possibilité pour des raisons de simplicité.

Deux cas vont alors se présenter :

* Une expertise humaine existe :

Cette expertise sera transformée en un ensemble de règles floues (*si..alors*) et les paramètres correspondant (qui caractérise les fonctions d'appartenances) peuvent être utilisés comme paramètres initiaux dans la procédure d'apprentissage. La coopération entre cette méthode d'apprentissage et l'expertise sera avantageuse dans

la mesure où les connaissances fournies par l'expert peuvent guider l'apprentissage dans le sens où le point de départ dans l'espace des paramètres n'est pas éloigné du point optimal et que la procédure d'apprentissage peut affiner ces connaissances pour atteindre une performance meilleure.

* L'expertise humaine est absente :

le nombre de règles sera décidé par essai et erreur.A priori, on pourrait être effrayé par le nombre important de règles que ce principe risque d'engendrer.Or comme la conséquence d'un SIFANE est une représentation directe de l'action, en pratique cela ne nécessite pas un nombre élevé de règles.

Le système

Le système sera implémenté sous forme d'un réseau adaptatif. Toute fonction d'approximation susceptible de représenter au mieux le comportement des entrés-sorties du système peut être choisie grâce à la flexibilité qui caractérise ce type de réseaux.

Dans ce cas le système sera conçu comme un ensemble de n neurones, chacun d'eux utilisant une équation différentielle de premier ordre pour obtenir la variable d'état à l'instant suivant connaissant la variable à l'instant t.

Par ailleurs, si les équations d'état du système sont un ensemble d'équations différentielles de premier ordre

$$\dot{\vec{x}}(t) = \vec{f}(\vec{x}(t), \vec{in}(t), t) \tag{4.16}$$

où $x(t)$ est le vecteur composé de variable d'états au temps t et le vecteur d'entrée au

système, on peut employer une approximation linéaire pour déterminer les équations comme suit :

$$\overrightarrow{x}(h*k+h) = h\overrightarrow{f}(\overrightarrow{x}(h*k), \overrightarrow{in}(h*k), h*k) + \overrightarrow{x}(h*k) \qquad (4.17)$$

où k est un entier et h le temps de l'échantillonnage.

Il est à noter que si le temps h d'échantillonnage est grand l'approximation linéaire n'est plus une estimation valable pour l'état suivant.

Par conséquent un SAN sera considéré comme un réseau adaptatif constitué de deux sous réseaux, le CNF et le système.

4.4.2 Réseau adaptatif correspondant à la trajectoire

Connaissant l'état du système au temps $t = h*k$, le CNF doit générer une entrée au système qui évoluera vers l'état suivant au temps $(k+1)*h$.

En répétant ce procédé à partir de $t = 0$, nous obtiendrons la trajectoire déterminée par l'état initial et les paramètres de CNF. Le diagramme de transition représenté par la fig 4.8 est lui même un réseau adaptatif constitué de m SAN's.

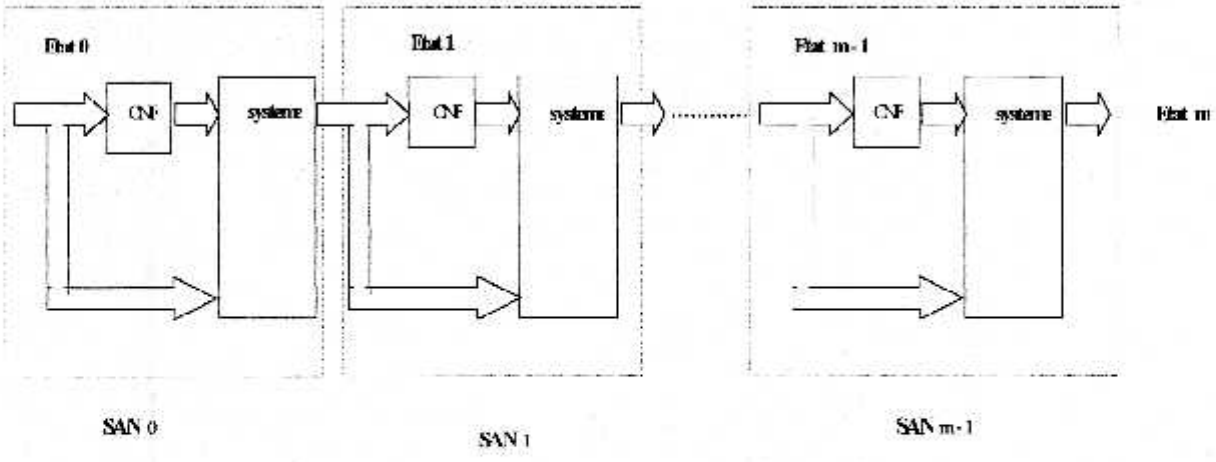

Fig 4.8 : Diagramme de transition

On peut donc appliquer une rétropropagation du gradient pour minimiser la différence entre les sorties du réseau adaptatif et les sorties désirées.

Les entrées du réseau adaptatif correspondant à la trajectoire sont l'état initial du système à $t = 0$; les sorties du réseau sont l'état de la trajectoire de $t = h$ jusqua $m * h$.

Les paramètres à ajuster appartiennent aux CNF (Fig 4.9)

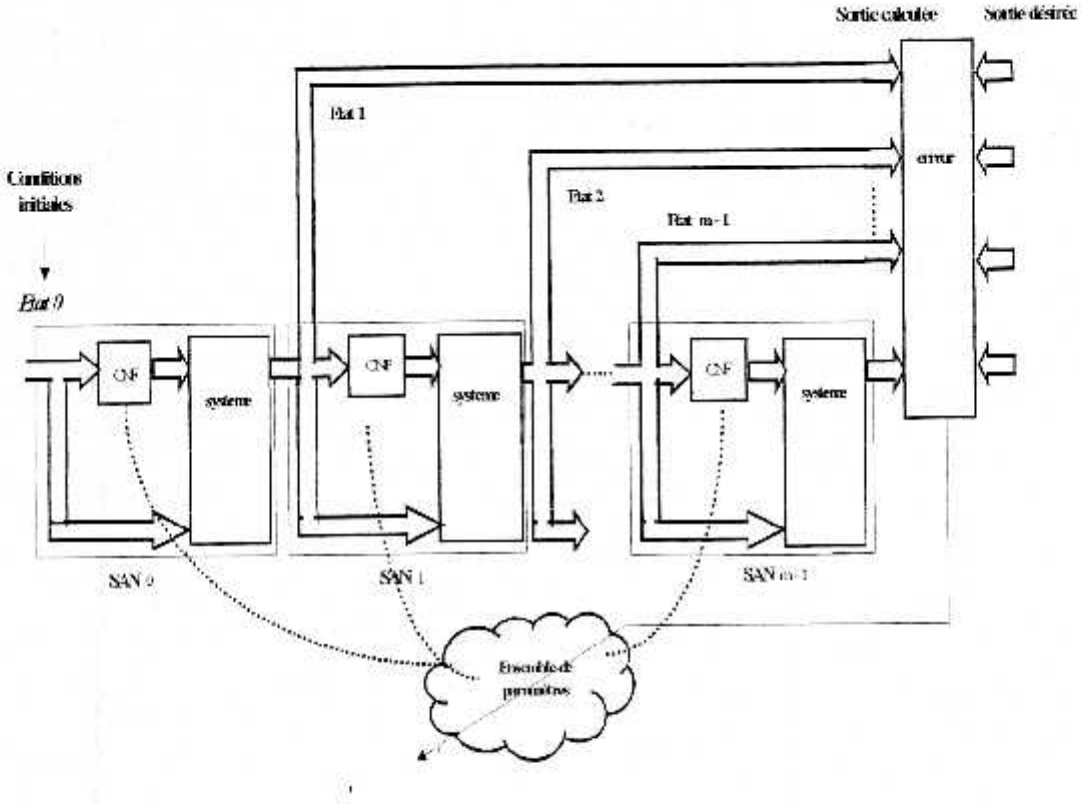

Fig 4.9 : Réseau adaptatif correspondant à la trajectoire

Chaque entrée de la base d'apprentissage est de la forme (état initial,trajectoire désirée)et la mesure de l'erreur à minimiser est :

$E = \sum_{k=1}^{m} \| \overrightarrow{x} (h * k) - \overrightarrow{xd} (h * k) \|$

où xd est la trajectoire désirée.

Avec des modifications mineures, la mesure de l'erreur peut être révisée comme

$$E = \sum_{k=1}^{m} \| \overrightarrow{x} (h * k) - \overrightarrow{xd} (h * k) \|^2 + \lambda \sum_{k=0}^{m-1} \left\| \overrightarrow{in} (h * k) \right\|^2 \qquad (4.18)$$

où $\overrightarrow{in}(n * k)$ est la sortie du CNF au temps $t - n * k$. Avec une sélection de λ, un compromis entre l'erreur de la trajectoire et l'effort de contrôle peut être obtenu.

4.5 Conclusion

Dans ce chapitre nous venons de voir une description détaillée d'une méthodologie de contrôle basée sur la rétropropagation temporelle du gradient.

Grâce à sa flexibilité et la liberté qu'elle offre pour la représentation mathématique des modèles, on peut l'inscrire dans la large panoplie des contrôleurs neuro-flous. Il faut tout de même signaler que les algorithmes d'apprentissage pour les systèmes d'inférence floue ne se réduisent pas uniquement à des adaptations de la rétropropagation du gradient pour des variantes des systèmes de Takaji-Sugeno. Les autres variétés d'apprentissages présentent chacun leur intérêt, et il est donc important de déterminer les algorithmes d'optimisation appropriés.

Chapitre 5

Expérimentation et validation

Afin d'expérimenter l'approche présentée, nous proposons d'étudier le problème classique de système dynamique non linéaire contrôle intelligent, celui du pendule invers [8].

5.1 Position du problème

Le pendule inversé est constitué d'un chariot sur lequel repose sans frottements un balai comme l'illustre la fig 5.1 :

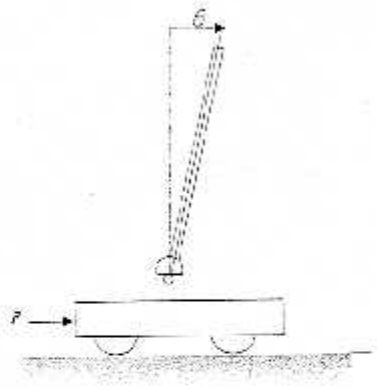

Fig 5.1 : Pendule inversé

La dynamique du pendule inversé est caractérisée par quatre variables d'états :

θ : angle du balai.

$\dot{\theta}$: vitesse angulaire.

Z : position du chariot.

$\dot{Z}$: vitesse du chariot.

Le comportement de ces variables d'état est gouverné par le système d' équations différentielles de second ordre suivant :

$$\begin{aligned} \ddot{\theta} &= \frac{g * \sin\theta + \cos\theta * \left(\frac{-F - m*l*\theta^2*\sin\theta}{m_c+m}\right)}{l * \frac{3}{4} - \frac{m*\cos^2\theta}{m_c+m}} \\ \ddot{Z} &= \frac{F + m*l*(\theta^2 + \sin\theta - \ddot{\theta} * \cos\theta)}{m_c + m} \end{aligned} \tag{5.1}$$

où g est l'accélération de la pesanteur est égale à $9,8m/s^2$, mc la masse du chariot est égale à $1.0kg$, m la masse du balai est égale à $0.1kg$, l la longueur du balai et F la force à appliquer pour maintenir le chariot en équilibre.

Dans notre simulation, nous ne considérons que les variables d'états θ et $\dot{\theta}$.

Contrôler le pendule inversé consiste à appliquer une force au chariot de manière

à ce que l'angle entre le balai et la verticale soit nul ainsi que la vitesse angulaire du balai.

5.2 configuration du réseau pour le problème du pendule inversé

Comme nous l'avons mentionné dans le chapitre 4, toute fonction succeptible de représenter au mieux le comportement des entrées et sorties du système peut être choisie.Nous avons opté pour une approximation linéaire par la méthode d'Euler [24], [27] :

$$\begin{aligned} x_1(t+h) &= h\,\dot{x_1}\,(t) + x_1(t) \\ x_2(t+h) &= h\,\dot{x_2}\,(t) + x_2(t) \end{aligned} \tag{5.2}$$

Avec $x_1(.) = \theta(.)$ et $x_2(.) = \dot{\theta}\,(.)$.

Le sous réseau RESSYS sera par conséquent constitué de deux neurones dont les fonctions d'activation sont les deux équations (5.2)

5.2.1 Le CNF

Notre problème étant de réduire en minimum le nombre de paramètres à ajuster, le CNF sera implémenté dans un premier temps comme un SIFANE à deux entrées caractérisées chacune par deux fonctions d'appartenances.

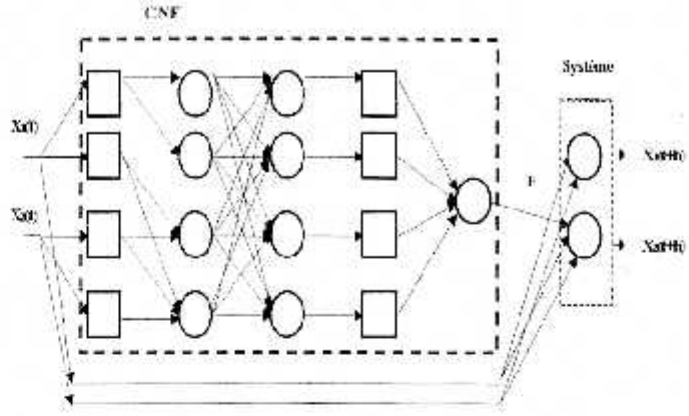

Fig 5.2 : SAN pour la simulation

C'est donc un contrôleur neuro-flou de quatre règles de type Sugeno.

5.2.2 Quantification des références d 'entrées et de sortie

Les valeurs de θ et $\dot{\theta}$ sont supposées varier sur la plage $[-20\ 20]$ et $[-50\ 50]$ respectivement. La force F évolue sur l'intervalle $[-15\ 15]$.

5.2.3 Constitution de la base d'exemples

L'apprentissage supervisé des paramètres du réseau nécessite une base d'exemples (entrée x, commande désirée).

N'ayant à priori aucune connaissances sur le système, les paramètres ont été initialisés d'une manière subjective.Les paramètres des conséquences ont tous été initialisés à zéro, ce qui signfie que le contrôle est nul initialement.Les ensembles flous intervenant dans les prémisses sont caractérisés par des fonctions d'appartenances de type Bell Shaped (en cloche).

Les différentes valeurs linguistiques prise par l'entrée floue ANGLE correspondant à θ sont PETIT et GRAND caractérisées respectivement par $\{20, 2, -20\}$ et $\{20, 2, 20\}$ et les valeurs linguistiques prise par l'entrée floue VITANG correspondant à $\dot{\theta}$ sont PETITE et GRANDE caractérisées respectivement par $\{50, 2, -50\}$ et $\{50, 2, 50\}$.

5.2.4 L'apprentissage

Pour induire la trajectoire, nous employons 100 SAN's, chacun d'eux correspondant à un temps de transition de 10ms.Un SAN étant constitué de deux sous réseaux, leur apprentissage se fera séparément La technique d'apprentissage est la rétropropagation du gradient de l'erreur.La tâche de RESSYS est de simuler la dynamique du système.Ses entrées sont donc l'état du système à un instant donné et la force à appliquer au chariot et sa sortie est l'état du système à l'instant suivant.L'apprentissage se fait en appliquant une force aléatoire au chariot. Une fois l'apprentissage de RESYS réalisé, celui-ci est relié au contrôleur. L'entrée du contrôleur est l'état du système, sa sortie est la force qu'on doit appliquer au chariot.Le rôle du contrôleur est donc d'apprendre la loi de commande (force) et.son apprentissage est par conséquent guidé par celui de RESSYS car il n'existe aucun professeur qui lui apprend la commande(force).

Le pas utilisé est donc $h = 10ms$ et la trajectoire correspond à l'intervalle de temps de $t = 0$ à $t = 1s$.

Il faut remarquer que si h est trop petit, un nombre plus élevé de SAN's sera em-

ployé, ce qui augmenterai le temps des signaux de propagation et par conséquent retardera le processus d'apprentissage et si il est par contre trop grand, l'approximation linéaire utilisée pour implémenter le système ne sera plus valable.

La base d'apprentissage est constituée des couples (condition initiale, trajectoire désirée)

Les conditions initiales sont un vecteur de deux éléments correspondant aux conditions initiales du balai (angle et la vitesse angulaire).

La trajectoire césirée est un vecteur de 100 éléments qui contient la valeur désirée de l'angle à chaque phase.

Dans notre simulation, deux entrées ont été utilisées :

Les conditions initiales (10 0) et (-10 0) respectivement et la trajectoire désirée est toujours le vecteur nul.

En résumé, le réseau doit apprendre à balancer le balai à partir des conditions initiales -10 ou -10 et parvenir à l'objectif de contrôle d'une manière optimale en minimisant l'erreur

$$E = \sum_{k=1}^{100} \theta^2 (h * k) + \lambda \sum_{k=0}^{99} f^2 (h * k) \tag{5.3}$$

où $f(h*k)$ est la sortie du contrôleur c'est a dire la force et λ l'effort de contrôle.

5.3 Simulation

5.3.1 Introduction

Etant donnée la nature du thème à traiter, nous avons eu recours au logiciel MATLAB car il intègre parfaitement les fonctions appropriées au traitement dans ce contexte.

C'est est un langage de programmation qui à l'origine a été developpé pour le calcul matriciel .Aujourd'hui il offre bien d'autres possibilités comme la visualisation graphique des données, le traitement du signal, l'analyse numérique et possède des librairies pour répondre à des besoins plus spécifiques comme les réseaux de neurones et la logique floue. Il représente un système interactif de calcul numérique et de visualisation graphique 2D et 3D.

5.3.2 Résultats

Les figures 5.3 a et 5.3 b représentent les fonctions d'appartenances initiales de l'angle et de la vitesse angulaire tandis que les figures 5.3 c et 5.3 d représentent les fonctions d'appartenances finales .

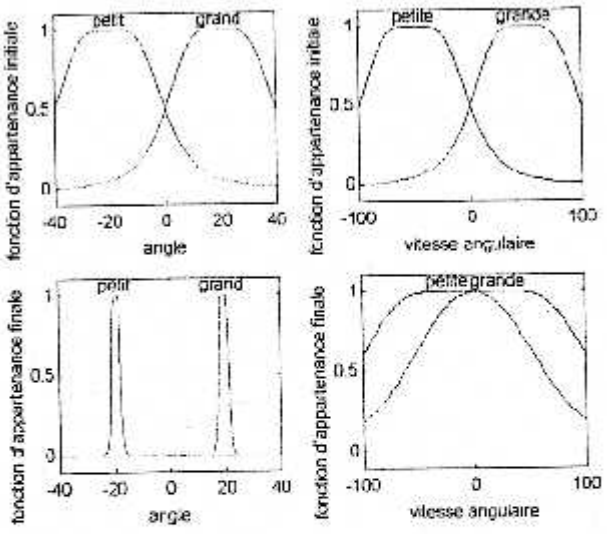

Fig 5.3 :Fonctions d'appartenance initiales et finales

La surface d'action de contrôle initiale est représentée par la figure 5.4 et le diagramme d'inférence initial par la figure 5.5 tandis que le diagramme d'inférence final et la surface d'action de contrôle finale sont représentés respectivement par les figure 5.6 et 5.7.Ces figures ont été obtenues grâce aux commandes *gensurf* et *ruleview*.

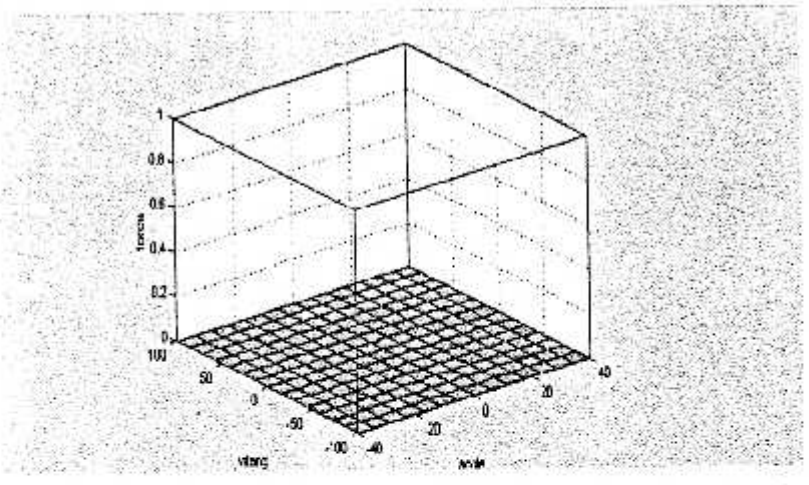

Fig 5.4 : Surface d'action de contrôle initiale

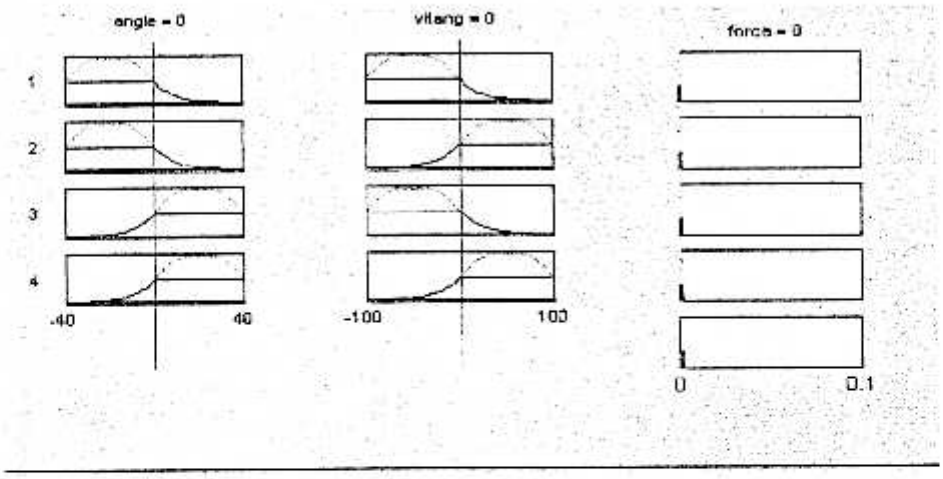

Fig 5.5 :Diagramme d'inférence initial

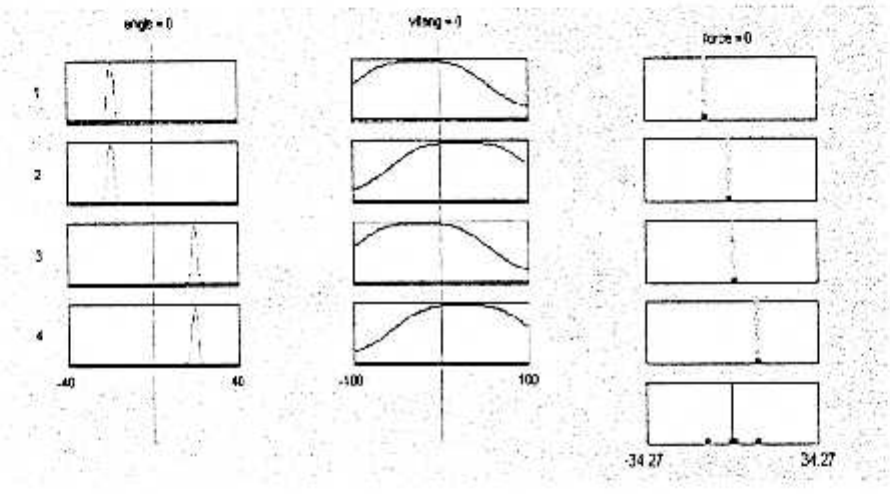

Fig 5.6 : Diagramme d'inférence final

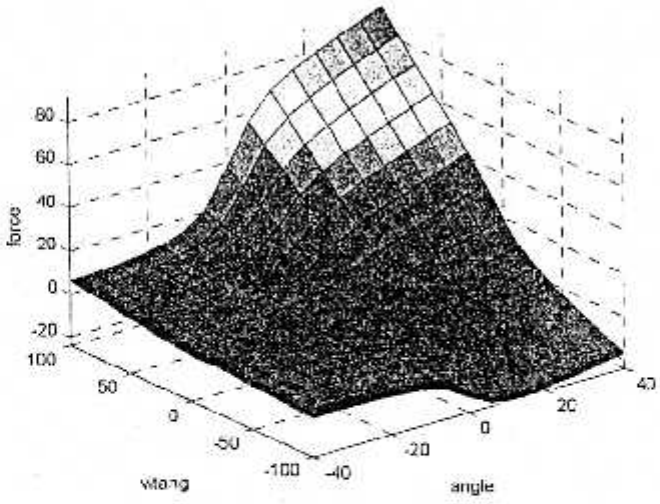

Fig 5.7 : Surface d'action de contrôle finale

La base des règles initiale est la suivante :

si θ est $Petit$ et $\dot{\theta}$ est $Petite$ alors $force = 0$

si θ est $Petit$ et $\dot{\theta}$ est $Grande$ alors $force = 0$

si θ est $Grand$ et $\dot{\theta}$ est $Petite$ alors $force = 0$

si θ est $Grand$ et $\dot{\theta}$ est $Grande$ alors $force = 0$.

La base des régles finale est la suivante :

si θ est $Petit$ et $\dot{\theta}$ est $Petite$ alors $force = 0.05493 * \theta + 0.05372 * \dot{\theta} + 0.005432$

si θ est $Petit$ et $\dot{\theta}$ est $Grande$ alors $force = 0.1198 * \theta + 0.1195 * \dot{\theta} + 0.001187$

si θ est $Grand$ et $\dot{\theta}$ est $Petite$ alors $force = 0.3134 * \theta + 0.3121 * \dot{\theta} + 0.03132$

si θ est $Grand$ et $\dot{\theta}$ est $Grande$ alors $force = 0.6833 * \theta + 0.6831 * \dot{\theta} + 0.06833$.

5.3.3 Interprétations

Nous remarquons que la différence entre la fonction d'appartenance initiale de l'angle et sa fonction d'appartenance finale est trés apparente et qu'il n'existe aucune fonction d'appartenance qui couvre l'intervalle [-10 10] de l'angle θ rendant l'interprétation des régles linguistiques difficiles.Or , comme notre contrôleur a été conçu comme un approximateur de fonction, l'intérprétabilité linguistiques ne constitue pas l'une de ces prérogatives majeures...Pour permettre une interprétation plus facile des règles linguistiques, nous avons augmenté le nombre de paramètres du contrôleur le dotant ainsi de plus de degrés de liberté..Chaque variable linguistique est caractérisée par trois valeurs linguistiques au lieu de deux . La figure montre que la différence entre les fonctions d'appartenances initiales et finales du contrôleur flou à 9 règles est moins apparente .et que les fonctions d'appartenances finales couvrent tout l'intervalle .

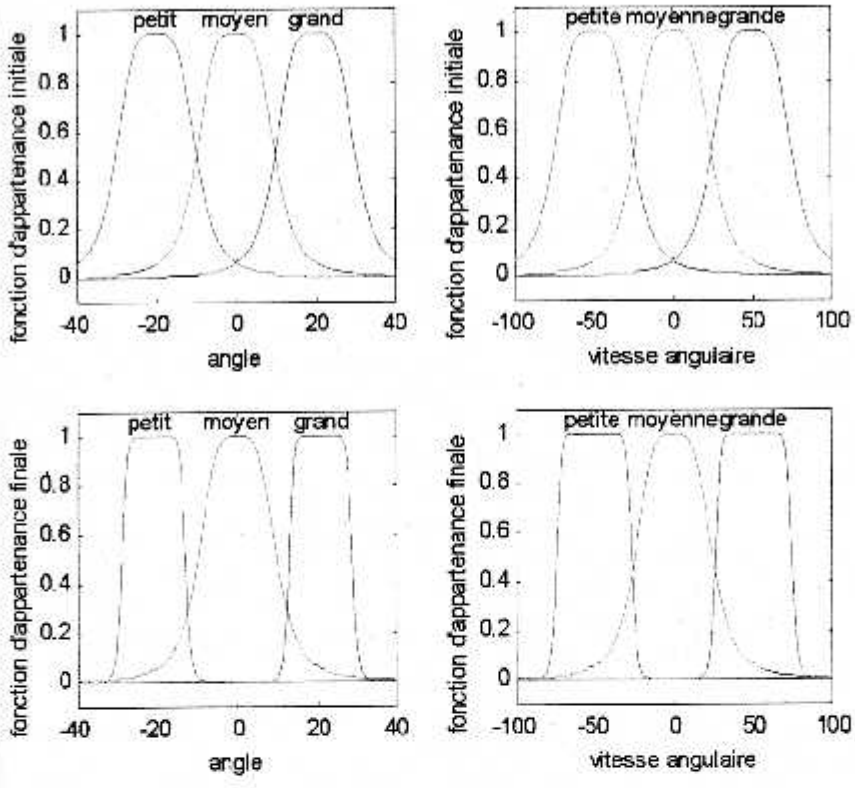

Fig 5.8 : (a),(b):Fonctions d'appartenances initiales,(c),(d):fonctions d'appartenances finales pour un système d'inférence flou à 9 règles

La généralisation est une des propriétés les plus importantes des réseaux de neurones .Elle représente la capacité du réseau à répondre correctement aux entrées qui ne lui ont pas été présentés lors de l'apprentissage .Nous avons utilisé d'autres conditions initiales que celles utilisées durant la phase d'apprentissage pour savoir comment le contrôleur allait réagir .

La base des règles obtenue est la suivante :

si θ est *Petit* et $\dot{\theta}$ est *Petite* alors $force = 0.03902 * \theta + 0.03906 * \dot{\theta} + 0.0039594$

si θ est *Petit* et $\dot{\theta}$ est *Grande* alors $force = 0.09899 * \theta + 0.09906 * \dot{\theta} + 0.002618$

si θ est Grand et $\dot{\theta}$ est $Petite$ alors $force = 0.2856 * \theta + 0.2858 * \dot{\theta} + 0.02531$

si θ est Grand et $\dot{\theta}$ est Grand$e$ alors $force = 0.6713 * \theta + 0.6715 * \dot{\theta} + 0.05587$.

La figure 5.9 représente la position de l'angle obtenue grâce à la base d'apprentissage (rouge) et la position de l'angle obtenue avec d'autres conditions initiales (jaune).

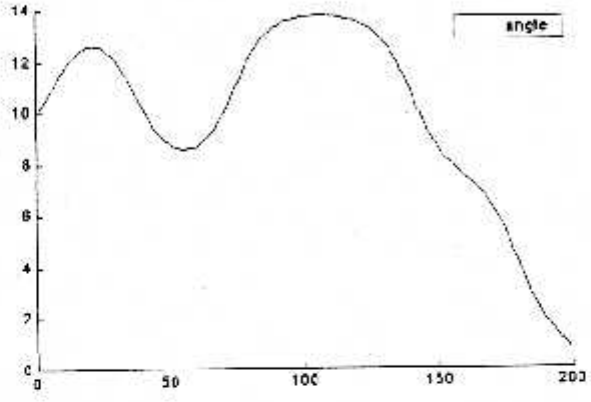

Fig 5.9 : courbes comparatives de la phase apprentissage et de la phase de généralisation

Conclusion générale

Parmi les nombreuses applications de théorie des ensembles flous, la commande floue est le domaine qui suscite le plus de curiosité, le plus de controverse et finalement s'avère être le champ d'application le plus actif dans le monde .

Dans ce contexte, nous avons présenté une technique de contrôle basée sur la fusion de la logique floue et des réseaux de neurones .

Tout d'abord, nous avons présenté les éléments de base de la logique floue.Les notions développées ont été appuyées par des exemples d'illustrations apportant une idée précise quant à la supériorité et la puissance de la logique floue par rapport aux méthodes classiques (ensembles ordinaires, logique classique, théorie des probabilités) de représentation de l'information .

Nous avons ensuite examiné les différents niveaux d'intervention du flou en commande.De cette étude il est ressorti que les contrôleurs flous utilisent un raisonnement développé à partir de connaissances exprimées sous forme de règles linguistiques .On peut distinguer plusieurs étapes dans le traitement de ces règles.Pour la plupart des étapes, il existe plusieurs possibilités théoriques, ces dernières n'étant évidemment pas indépendantes entre elles ni de l'application considérée.

Notre choix s'est porté sur les règles de type Takagi -Sugeno dont la forme est la suivante:

si x_1 est A_1.....et x_n est A_n alors $y = f(x_1, x_2, ... x_n)$ où f est une fonction affine.

Ce choix est justifié par le fait que ces types de règles se prêtent bien à des méthodes d'optimisation utilisant la descente du gradient .

L'acquisition des connaissances dans un cadre flou pose de réelles difficultées .Les paramètres du contrôleur sont généralement déterminé suite à une expertise humaine, des notions de subjectivité y sont généralement attachées et aucune garantie d'optimalité n'est énoncée .C'est pour cela que nous avons avons utilisé une démarche permettant l'extraction automatique des connaissances en utilisant les capacités d'apprentissage des réseaux de neurones.

La robustesse est probablement l'une des propriétés les plus intéressantes de la commande floue, nous l'avons constaté aprés avoir modifiées les conditions inititiales dans la phase de généralisation.

Dans un soucis de clarté et de simplicité, notre étude s'est restreinte au cas MISO (Multi Inputs Single Output) .L'extension au cas MIMO (Multi Inputs Multi Outputs) dans le cas où un processus multivariable pourrait être découplé s'effectuerai à l'aide de plusieurs contrôleurs monovariables et l'on est donc ramené au cas précédent .Dans le cas contraire, la synthèse des lois de commande floue est beaucoup plus délicate et reste un des problèmes ouverts de la commande floue .

De façon plus générale, les ensembles flous peuvent intervenir efficacement dans la modélisation d'un système complexe et leur utilité ne se restreint pas uniquement à la commande de processus .Tous les grands domaines techniques peuvent tirer avantage de son utilisation que ce soient la reconnaissance des formes, les bases de données, le traitement d'images, la gestion financière pour ne citer que les plus

[1] : F. Alche Buc, «Modèles neuronaux et algorithmes constructifs d'apprentissage des régles de decision », Thèse de doctorat Paris 11, 1993.

[2] : S. Altug, « A mutual update training algorithm for fuzzy logic control/ decision network (FALCON) », IEEE Trans. On Neural Network, vol 10 n°1, pp 196-198, 1999.

[3] : C.W. Anderson, « Learning and solving with multilayer connexionist systems », Phd Thesis, University of Massachusets, 1986.

[4] : C.W. Anderson, « Strategy learning with multilayer connexionist representation », Tech.Rep.TR87-509.3, GTE Laboratories, Inc, May 1998 .

[5] : R.C. Atkinson, G.H.Bower, and E.J.Crothers, « An introduction to mathematical learning theory », New York.Wiley, 1965.

[6] : A. Barto, R. Sulton, C.Anderson . « Neuronlike elements can solve difficult learning control problems », IEEE Trans. On SMC, vol 13, Sept. 83.

[7] : A.Barto and M.I. Jordan, « Gradient following without backpropagation in layered networks », in Proc.IEEE, First Annual Conf. Neural Network, San Diego, CA, pp 629-636, 1987.

[8] : H.R. Berendji, « learning and tuning fuzzy logic controllers through reinforcements », IEEE Trans. On Neural Network, vol 3 n°5 pp 724-740, 1992.

[9] : H.R. Berendji, « Fuzzy logic controllers, in an Introduction to fuzzy logic applications in Intelligent Systems », Eds.Boston :Kluwer Academic, pp 69-96, 1991.

[10] : H.R. Berendji, « In Introduction for designing fuzzy controllers using neural network, Int.J.Approximate reasonning », Vol 6 n° 2, pp 267-292, Feb 1992.

[11] : H.R. Berendji, « Q-learning of fuzzy rules », Proc. Of IJCAI'93, Chambéry 1993.

[12] : B.Bouchon Meunier, «La logique floue et ses applications », 1995.

[13] : B. Bourret - J.Reggia - M.Samuelides, «Réseaux neuronaux, une approche connexionniste de l'intelligence artificielle », Ed.Teknea, 1991.

[14] : Braae, «fuzzy relations in a control setting », Kybernetes, vol 7, pp 185-188, 1978.

[15] : Buckley, « Solving systems of linear fuzzy equations », Fuzzy sets and Systems 43, pp 33-43, 1991.

[16] : L. Chang, L. Zadeh, «On fuzzy mapping and control », IEEE Trans. On systems ; vol 2, n°1, 1972.

[17] : G. Chyan Hwang and Shih-Chang Lin, « A stability approach to fuzzy control design for non linear systems », Fuzzy sets and systems 48, pp 279-287, North Holland, 1992.

[18] : E. Davalo, P. Naim, «Des réseaux de neurones », Eyrolles Ed, 1990.

[19] : D.Driankov, H. Heillendroorn, M. Reinfranb, «An introduction to fuzzy control », 1993.

[20] : D. Dubois, H.Prade, «Théorie des possibillités », Ed. Masson, 1988.

[21] : L.A. Feldkamp, Puskorins G., «Architecture and training of a Hybrid Neural-Fuzzy System », Proc. Of Iuzuka'99, Iuzuka, Japan 1992.

[22] : H.Ferreny, M. Ghallab, «Intelligence artificielle », Ed. Hermes 1990.

[23] : P.Y.Glorennec, «Fuzzy Q-learning and Dynamical Fuzzy Q-learning », Proc.of FUZZ-IEEE' 94,Orlando, juin 1994.

[24] : P.Y. Glorennec, « Learning algorihms for neuro fuzzy networks », fuzzy systems, RC PRS, 1993.

[25] : S.Horikawa, T. Furuhashi, Y. Uchikawa, « On Fuzzy modeling using fuzzy neural networks with the back propagation algorithm », IEEE Trans. On Neural Networks, vol 3 n°5, pp 801-806, 1992.

[26] : J.S.Jang, «Fuzzy modeling using generalised neural networks and Kalman filter algorithm », Proc. Ninth National Conf. Artificial Intelligence (AAAI-91), pp 762-767, Juillet 1991.

[27] : J.S. Jang, «self learning Fuzzy controllers based on temporal back propagation », IEEE Trans. On Neural Network, vol 3 n°5 pp 714-723, 1992.