RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
ÉCOLE NATIONALE POLYTECHNIQUE

Département de Génie électrique
Option : Automatique

Projet de Fin d'Etudes

THÈME

Commande par systèmes flous à structure auto adaptative

Proposé et dirigé par :
Mr H.CHEKIREB

Présenté par :
LALEG Nazim
TALBI Ahmed

Promotion Juin 2003

Laboratoire de commande des processus
10, Avenue Hassen Badi, El-Harrache, Alger.

REMERCIMENTS

Les travaux réalisés dans ce projet de fin d'études se sont déroulés au sein du laboratoire de commande des processus de l'Ecole Nationale Polytechnique.

Nous remercions Monsieur H. CHEKIREB, notre promoteur, pour le temps précieux qu'il nous a consacré afin de nous diriger et nous conseiller dans notre travail.

Nous remercions aussi les membres du jury, à savoir Mr TADJINE et Mr BERKOUKE, pour avoir bien voulu juger notre travail.

Nous remercions également tous les enseignants du département de Génie Electrique.

DEDICACES

Je dédie cet humble travail à :

Mes très chers parents dont la tendresse, l'amour et les encouragements ont été l'essence de ma réussite dans mes études.

Mes très chers frères et sœurs.

Mon très cher ami et binôme, NAZIM, ainsi qu'a toute sa famille.

Mes très chers amis et compagnons sans oublier ceux de demain.

Tous les membres de ma grande famille sans oublier mes adorables petits neveux.

Ahmed

Ce travail aussi humble que soit il est dédié à :

Mes très chers parents sans qui je n'aurais pu surmonter toutes ces épreuves.

Ma sœur MIMI ainsi qu'à mon frère AZIZ pour leur soutient.

A mes grands mères, et à la mémoire de mon regretté grand père.

A mon complice dans les études mon binôme Ahmed ainsi qu'à toute sa famille.

A ma tante Fadhila, mes oncles et leurs épouses et mes cousins.

A tous mes amis de Tadmait et de polytech.

Nazim

SOMMAIRE

Chapitre I

Rappels sur la logique floue et les réseaux adaptatifs

I.1. PARTIE 1 : GENERALITES SUR LA LOGIQUE FLOUE

I.1.1. INTRODUCTION

La logique floue, considérée comme une science du raisonnement, a commencé avec les travaux du philosophe ARISTOTE. Plus tard après une période de stagnation, le mathématicien G.BOOLE introduit, au 19ème siècle, la logique à forme symbolique et mathématique. Cette logique, dite booléenne ou binaire, admet comme propriété la non-contradiction, ce qui signifie qu'une proposition est vraie ou fausse. Or, pour être complété, la description d'une réalité nécessite une forme de logique souple affirmant qu'une proposition peut être à peu près fausse ou à peu près vrai.

En 1965, le concept d'ensemble flou a été introduit par l'automaticien **L. ZADEH** [15]. L'idée de base a pris naissance lorsqu'on a constaté la difficulté de programmer un automate en vue de la réalisation d'une tache jugée simple à réaliser par un être humain. Le mode de raisonnement humain et le moyen de formaliser la connaissance humaine dans un langage accessible à une machine constituent les deux principaux sujets de réflexion qui ont mené à l'apparition de la logique floue.

L'application de la logique floue paraît intéressante dans le cas de la mise en œuvre de méthodes récentes dont les données sont subjectives et qui se basent sur des connaissances exprimées par une sémantique (traitement du sens des mots) d'une autre manière, on peut dire que les règles floues sont une représentation des connaissances humaines, exprimées en langage naturel, à l'aide de mots vagues mal définis « flous » [1].

Les premières applications dans le domaine de la commande ont vu le jour en 1974 avec les travaux de **MAMDANI** et **ASSILIAN**. Depuis, l'application de la logique floue pour la commande des procédés industriels ne cesse de se généraliser. Cette nouvelle stratégie de commande repose sur le remplacement des algorithmes conventionnels par des règles introduisant des variables linguistiques ayant la forme suivante :

$$\text{Si (X est A), ... Alors (Y est B)} \qquad \textbf{(I.1)}$$

L'application de la logique floue pour la commande offre des avantages multiples, en particulier, dans le cas d'un processus complexe difficile à modéliser ou dans le cas de la synthèse d'un régulateur pour un procédé non linéaire.

La structure générale d'une commande par logique floue dite commande floue, est schématisée dans la **FigureI.1.** suivante :

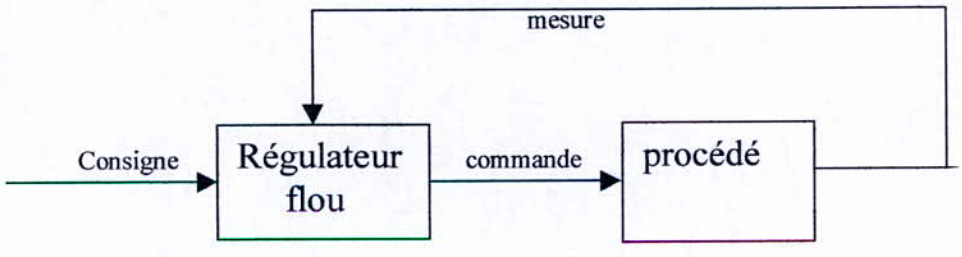


Figure.I.1. Structure générale d'une commande floue

I.1.2. ENSEMBLES FLOUS

Un ensemble classique est un ensemble possédant des limites et des frontières. Si on définit par exemple l'ensemble A tel que :

$$A = \{x \mid x > 6\} \qquad (I.2)$$

Alors, il est clair que la limite de l'ensemble A est 6. Tout élément x supérieur à 6 appartient à l'ensemble A. Si x est inférieur à 6, il ne fait pas partie de A.

Contrairement aux ensembles classiques, les ensembles flous tel que leur nom l'indique, sont des ensembles sans limite ni frontière. La transition de «l'appartenance à un ensemble » vers la « non appartenance à un ensemble » s'effectue d'une manière graduelle, et elle est caractérisée par des fonctions d'appartenance qui donnent aux ensembles flous une certaine flexibilité dans la modélisation utilisant des expressions linguistiques tel que «l'eau est chaude », «la température est élevée ».

ZADEH a mentionné en 1965 dans sa publication intitulée « Ensembles flous » [15] que les ensembles imprécis jouent un rôle important dans la pensée humaine, particulièrement dans le domaine de la modélisation et de la communication. Il est à noter que le concept « flou » ne provient pas du caractère aléatoire des constituants de l'ensemble mais plutôt de l'imprécision et de la nature des pensées et des concepts abstraits.

I.1.2.1. ENSEMBLES FLOUS ET FONCTIONS D'APPARTENANCE

Supposons que X représente un certains nombre d'objets x, alors l'ensemble flou A inclus dans X est définit par un ensemble de paires ordonnées :

$$A = \{(x, \mu_A(x)) \mid x \in X\} \qquad (I.3)$$

Où $\mu_A(x)$ est appelée fonction d'appartenance de x dans A, notée MF (membership function). La fonction d'appartenance lie chaque élément de X à une valeur continue comprise entre 0 et 1.

Evidemment, la définition de l'ensemble flou n'est qu'une simple extension de la définition d'un ensemble classique à la différence que la fonction caractéristique des ensembles flous prend des valeurs continues comprises entre 0 et 1. Si la valeur de la fonction d'appartenance $\mu_A(x)$ prend des valeurs égales à 0 ou 1, alors l'ensemble A sera un ensemble classique et $\mu_A(x)$ une fonction caractéristique de A.

Généralement X est appelé « l'univers de discours », ou tout simplement « l'univers », il peut contenir des éléments discrets ou des valeurs continues.

I.1.2.2. OPERATIONS DANS LES ENSEMBLES FLOUS

a. Inclusion et sous ensembles

L'ensemble flou A est inclus dans l'ensemble flou B (ou bien A est un sous ensemble de B) si et seulement si $\mu_A(x) \leq \mu_B(x)$ pour tout x. on note alors :

$$A \subseteq B \Leftrightarrow \mu_A(x) \leq \mu_B(x) \qquad \textbf{(I.4)}$$

b. Union (disjonction)

L'union entre deux ensembles flous A et B est un ensemble flou noté C = A ∪ B, ou C =A **OR** B. La fonction d'appartenance de C est donnée par :

$$\mu_C(x) = max(\mu_A(x), \mu_B(x)) = \mu_A(x) \ \text{V} \ \mu_B(x) \qquad (I.5)$$

La définition la plus commune de l'union telle qu'elle a été indiquée par ZADEH [15] est le plus petit ensemble flou contenant les deux ensembles A et B. Inversement, si D est un ensemble flou quelconque contenant les deux ensembles A et B alors il contient A ∪ B.

c. Intersection

L'intersection entre deux ensembles flous peut être définit de la même manière que l'union. L'intersection de deux ensembles flous A et B est un ensemble flou C, noté C = A ∩ B, ou bien C = A **and** B, dont la fonction d'appartenance est donnée par :

$$\mu_C(x) = min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x) \qquad (I.6)$$

Comme dans le cas de l'union, il est évident que l'intersection de A et B est le plus grand ensemble flou inclus dans A et B. ce qui revient finalement à la définition de l'intersection d'ensembles non flous.

d. Complémentarité

Le complément d'un ensemble flou A, noté $\overline{A}$ (ou bien : –A, NOT A), est définit par :

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \qquad (I.7)$$

La figure I.2 représente les principales opérations vues précédemment.

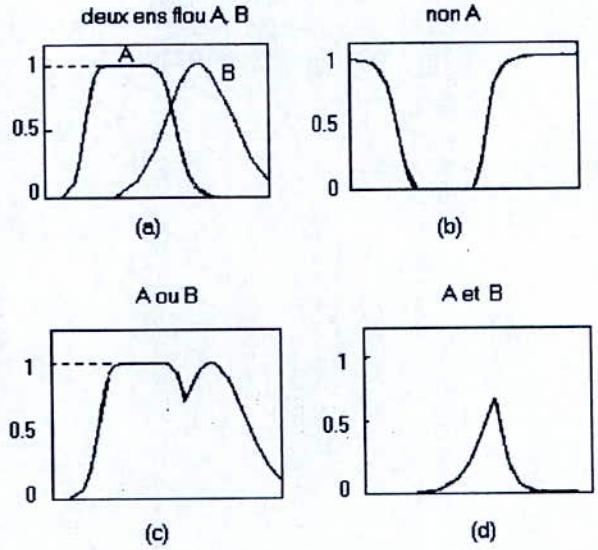


Figure I.2. Opérations dans les ensembles flous : a)-deux ensembles flous A et B, b)- $\overline{A}$ c)- A U B, d)- A∩B

Il est à noter qu'il est possible de trouver dans la littérature d'autres définitions relatives à ces opérations proposées sous le nom de « T-norm » et « T-conorm ».

A l'exception du MIN et MAX, aucune autre opération ne satisfait la propriété de distribution suivante :

$$\mu_{A\cup(B\cap C)}(x) = \mu_{(A\cup B)\cap(A\cup C)}(x)$$
$$\mu_{A\cap(B\cup C)}(x) = \mu_{(A\cap B)\cup(A\cap C)}(x) \qquad \text{(I.8)}$$

Cependant, MIN et MAX engendrent de difficultés lors de l'analyse des systèmes FIS. C'est pour cela qu'il est préférable d'utiliser le AND et OR définis en probabilité :

$$\mu_{(A\cap B)}(x) = \mu_A(x)\,\mu_B(x)$$
$$\mu_{(A\cup B)}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\,\mu_B(x) \qquad \text{(I.9)}$$

I.1.2.3. Exemples de fonctions d'appartenance

Dans ce qui suit, nous allons présenter différentes classes de fonctions paramétriques communément utilisées pour définir les fonctions d'appartenance. Ces dernières jouent un rôle très important dans les systèmes FIS adaptatifs.

a. Fonctions d'appartenance triangulaires

La fonction d'appartenance triangulaire est définit par trois paramètres {a, b, c} qui définissent les coordonnées des trois sommets du triangle:

$$triangle(x;a,b,c) = max\,(min\,(\frac{x-a}{b-a}, \frac{c-x}{c-b}), 0). \qquad \text{(I.10)}$$

La figure I.3.a illustre un exemple d'une MF triangulaire définit par *triangle(x ; 20, 60, 80).*

b. Fonctions d'appartenance trapézoïdales

La fonction d'appartenance trapézoïdale est caractérisée par quatre paramètres {a, b, c, d} comme suit :

$$trapézoïde\,(x;a,b,c,d) = max\,(min\,(\frac{x-a}{b-a},1,\frac{d-x}{d-c}),0). \qquad (I.11)$$

La **figure I.3.b** illustre un exemple d'une MF trapézoïdale définit par *Trapézoïde (x ; 10, 20, 60, 95).* Il est clair que la MF triangulaire n'est qu'un cas particulier de la MF trapézoïdale.

Grâce à la simplicité de leurs formules et à leur efficacité de calcul, les MF triangulaires et trapézoïdales ont été très utilisées, spécialement dans les implémentations en temps réel. Cependant, étant donné que ces MF sont composées de lignes droites, elles ne sont pas lisses. D'autres MFs définies par des fonctions lisses et non linéaires ont été alors introduites.

c. Fonctions d'appartenance gaussiennes

Elles sont définies par deux paramètres {σ, c} comme suit :

$$guassien(x,\sigma) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})} \qquad (I.12)$$

Où c représente la valeur moyenne de la MF et σ sa variance. La **figure I.3.c** représente un exemple de MF gaussienne définie par *gaussienne(x ; 20, 50).*

d. Fonctions d'appartenance en cloche

Les fonctions d'appartenance en cloche sont spécifiées par trois paramètres {a, b, c} tel que :

$$Bell(x\,;\,a,\,b,\,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \qquad (I.13)$$

Où le paramètre b est généralement positif. Cette MF est une généralisation directe de la distribution de CAUCHY utilisée dans la théorie des probabilités. La **figure.I.3.d** illustre un exemple d'une MF en cloche donnée par *bell(x ; 20, 4, 50).*

Un choix approprié des paramètres {a, b, c} permet d'aboutir à des MFs en cloche désirées. On peut faire varier la valeur moyenne et la variance de la MF en faisant varier les paramètres c et a. Le paramètre b permet de contrôler la pente au niveau des points de croisement. La **figure.I.4** montre la signification physique de chaque paramètre dans une cloche.

e. Fonctions d'appartenance sigmoïdale

Elle est définie par :

$$Sigmoïde(x\ ;\ a,\ c) = \frac{1}{1+\exp[-a(x-c)]} \qquad (I.14)$$

Où a contrôle la pente au niveau du point de croisement $x=c$.

Etant donné qu'une MF sigmoïdale dépend du signe du paramètre a, elle est fondamentale pour la représentation des concepts tels que « très large » ou « très négative ». Les fonctions sigmoidales de ce type sont très utilisées dans les fonctions d'activation des ANNs.

Il est à noter que les fonctions d'appartenances introduites dans ce chapitre sont sûrement les plus utilisées mais il est possible de définir une fonction d'appartenance spécifique à une application donnée. Il est aussi évident que n'importe quelle fonction de distribution continue de probabilité peut être utilisée comme MF.

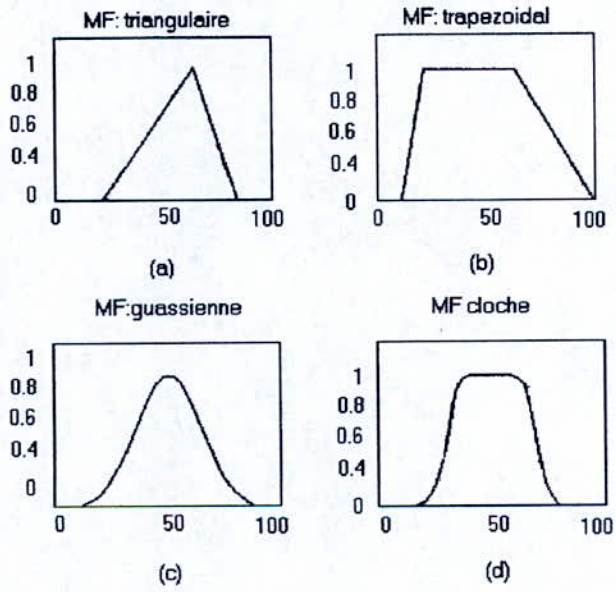


Figure I.3 : Exemples de classes de fonctions d'appartenance

(a) triangle(x ; 20, 60, 80)
(b) trapézoïde(x ; 10, 20, 60, 95)
(c) gaussienne (x ; 20, 50)
(d) belt(x ; 20, 4, 50)

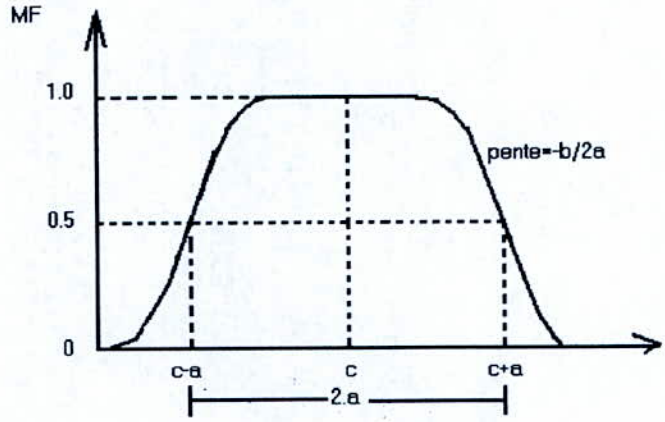


Figure I.4 : Sens physique des paramètres pour une fonction en cloche.

I.1.3. REGLES FLOUES

Les règles floues appelées aussi, les règles «si,..., alors», ou encore implications floues, sont de la forme suivantes :

Si x est A alors y est B

Où A et B sont des valeurs linguistiques définies par des ensembles flous dans les univers de discours X et Y respectivement. Souvent, l'expression « *x est A* » est appelée antécédent alors que « y est B » est dite conséquence. Des exemples des règles «si,..., alors» sont utilisées dans les expressions linguistiques, comme par exemple :

- Si la pression est élevée alors le volume est petit.
- Si la route est glissante alors la conduite est dangereuse.
- Si la tomate st rouge alors elle est mure.

Avant d'appliquer les règles floues «Si,...,alors» pour la modélisation et l'analyse d'un Système, il faut d'abord formaliser ce que veulent dire les expressions «Si x est A alors Y est B », ce qui est des fois noté A → B. L'expression décrit la relation entre deux variables x et y ; ceci suggère que la règle floue «si,..., alors» soit définit comme étant une relation R floue, et binaire dans l'espace produit X x Y. Il est à noter que la relation binaire R est une extension du produit cartésien classique, où chaque élément $(x, y) \in X \times Y$ est associé au degrés d'activation noté $\mu_R(x,y)$. Alternativement la relation binaire floue R peur être définit par l'ensemble flou de l'univers X x Y. Cet ensemble flou est caractérisé par une fonction d'appartenance à deux variables $\mu_R(x,y)$.

I.1.4. RAISONNEMENT FLOU

Le raisonnement flou appelé aussi raisonnement approximatif, est un mécanisme d'inférence utilisé pour déduire des conclusions à partir d'un ensemble flou, de règles «Si ,…,alors» et d'une ou plusieurs conditions. Avant d'introduire le raisonnement flou, il est nécessaire d'introduire les règles d'inférences qui jouent un rôle essentiel lors du raisonnement flou.

La composition des règles d'inférence est une généralisation des notions familières suivantes. Supposons que l'on dispose d'une courbe $y = f(x)$ qui régule la relation entre x et y. Lorsque $x = a$ alors à partir de $y = f(x)$ on peut inférer que $y = b = f(a)$. Voir **figure.I.5.a.** La généralisation du processus de façon à ce que **a** soit un intervalle et $f(x)$ une fonction d'intervalles comme illustré sur la **figure.I.5.b**. Pour trouver l'intervalle résultant $y = b$ correspondant à l'intervalle x =a, on doit d'abord construire une extension cylindrique de **a** (qui étend le domaine de a de X vers $X \times Y$), ensuite on doit trouver son intersection I avec l'intervalle de la courbe. La projection de I sur l'axe Y donne l'intervalle $y = b$.

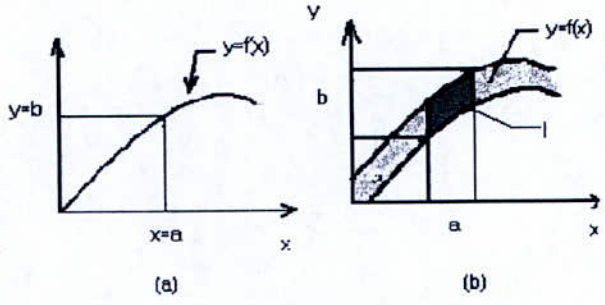


Figure I.5. Dérivation de y=b à partir de x=a et y=f(x)
(a) a et b sont des points, y=f(x) est une courbe.
(b) A et b sont des intervalles, y=f(x) est une fonction d'intervalles.

Pour pousser plus loin la généralisation, on suppose que A est un ensemble flou de X et F est une relation floue dans $X \times Y$, comme indiqués aux figures I.6.a et I.6.b. Pour retrouver l'ensemble flou B résultant, on construit à nouveau l'extension cylindrique C(A) ayant pour base A (qui est, une extension du domaine de A vers le domaine X xY pour retrouver C(A)). L'intersection de C(A) et F (figure I.6.c) forme une région analogue à l'intersection I de la figure I.6.b. Par projection de C(A) $\cap$ F sur l'axe des Y, on infère y comme un ensemble flou B sur l'axe des Y (voir fig. I.6.d).

Si μ_A, μ_B, $\mu_{C(A)}$, μ_F sont les fonctions d'appartenance de A, B, C(A) et F respectivement où la relation suivante est vérifiée : $\mu_{C(A)}(x,y) = \mu_A(x)$

Alors :

$$\mu_{C(A)\cap F}(x, y) = \mu_A(x) min[\mu_{C(A)}(x, y), \mu_F(x,y)] = min[\mu_A(x) \mu_F(x,y)] \quad (I.15)$$

En projetant C(A) ∩ F sur l'axe des y on obtient la relation suivante :

$$\mu_B(y) = max\ min[\mu_A(x, y), \mu_F(x,y)] = V_x[\mu_A(x) \wedge \mu_F(x,y)]. \quad (I.16)$$

B est alors représenté par

$B = A \circ F$

Où o représente l'opérateur de composition. Si l'on choisit le produit pour le ET flou et le max pour le OU flou alors on a le la composition du produit maximale et $\mu_B(y)$ est égale à $V_x[\mu_A(x) \wedge \mu_F(x,y)]$.

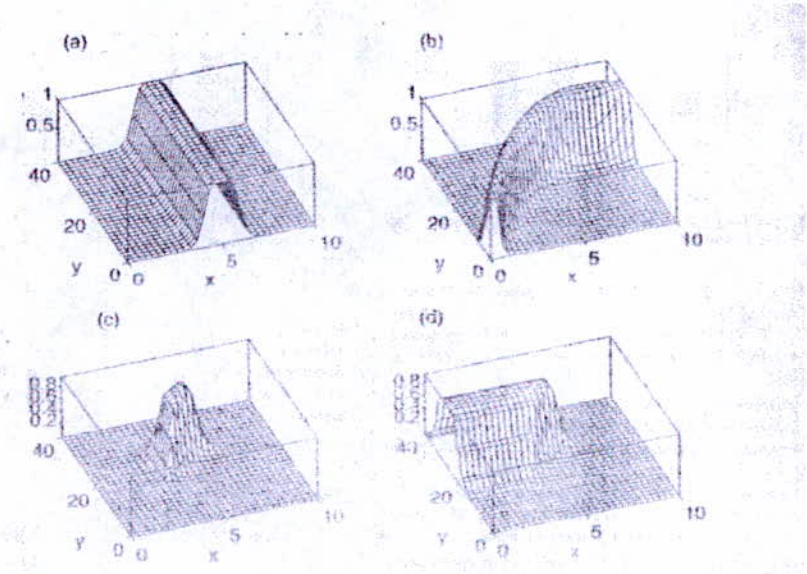


Figure I.6. Règles d'inférence

En utilisant la composition des règles d'inférences, il est possible de formaliser une procédure d'inférence appelée raisonnement flou, sur un ensemble de règles floues si-alors. Dans la logique classique booléenne nous pouvons affirmer la véracité d'une proposition B à partir de la véracité d'une proposition A et de l'implication A→B. exemple : si A = « la tomate est rouge » et B = « la tomate est mûre », alors si il est vrai que « la tomate est rouge », il est aussi vrai que « la tomate est mûre ». Ce concept est illustré par ce qui suit :

- Hypothèse 1 (cause) : *x est A*
- Hypothèse 2 (règle) : *si x est A alors y est B*

- Conséquence (conclusion) : *y est B*

Cependant, dans la majorité des raisonnements humains, l'implication est utilisée d'une manière approximative. Par exemple, si on avait la même règle d'implication précédente à savoir «Si la tomate est rouge alors elle est mûre » et on savait que « la tomate est plus ou moins rouge » alors on peut conclure que « la tomate est plus ou moins mûre ». Ceci peut être écrit comme suit :

- Hypothèse 1 (cause) : x est A'
- Hypothèse 2 (règle) : si x est A alors y est B

- Conséquence (conclusion) : y est B'

Où A' est proche de A et B' est proche de B. A' et B' sont des ensembles flous d'univers appropriés. La procédure d'inférence précédente est dite raisonnement flou ou raisonnement approximatif.

En utilisant la composition des règles d'inférence introduite précédemment, il est possible de formuler la procédure d'inférence du raisonnement flou comme suit :

Raisonnement flou basé sur la max-min composition

Soit A, A' et B des ensembles flous de X, X et Y respectivement. Supposons que l'implication floue A→B est exprimée comme étant une relation floue R dans X x Y. alors l'ensemble flou B' induit par « x est A' » et la règle floue « si x est A alors y est B » est définit par :

$$\mu_{B'}(y)=max\ min\ [\mu_{A'}(x),\ \mu_R(x,\ y)] = \vee_x[\mu_{A'}(x) \wedge \mu_R(x,\ y)] \qquad (I.17)$$

D'une manière équivalente on écrit :

$$B' = A' \circ R = A' \circ (A \rightarrow B) \qquad (I.18)$$

L'expression (I.17) est une expression générale pour le raisonnement flou, alors que l'expression (I.18) est un exemple d'un raisonnement flou, où min et max sont les opérateurs flous AND et OR respectivement.

Maintenant, il est possible d'utiliser la procédure d'inférence pour l'implication floue généralisée pour aboutir à des conclusions, sachant que l'implication floue A→B est définie comme une relation binaire floue appropriée.

a. mono-règle avec mono-antécédent

Pour une règle avec un antécédent, la formule est celle donnée dans (I.17).Une simplification de l'équation donne :

$$\mu_{B'}(y)=V_x[\mu_{A'}(x) \wedge \mu_A(x)] \wedge \mu_B(y) = \omega \wedge \mu_B(y) \qquad (I.19)$$

En d'autres termes, premièrement on détermine le degré de ressemblance ω qui représente le maximum de $[\mu_{A'}(x) \wedge \mu_A(x)]$, ensuite, la MF de l'ensemble flou résultant B' est obtenue en sectionnant l'ensemble B à la hauteur ω (comme indiqué à la figure I.7).

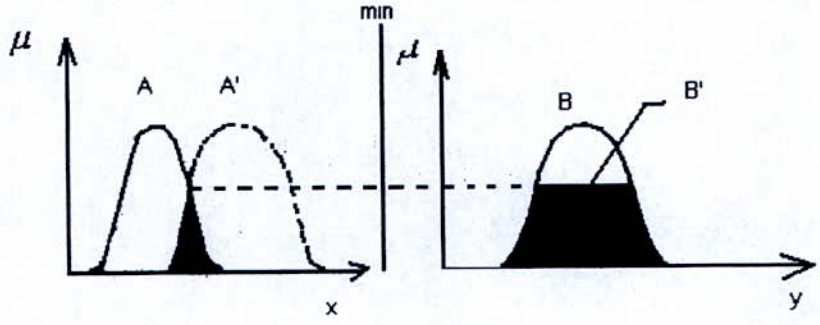


Figure I.7 : Raisonnement flou dans le cas d'une mono règle avec mono antécédent

Une règle floue «Si,...,alors» avec deux antécédents est souvent écrite sous la forme

« Si x est A et y est B alors z est C »

Le problème correspondant pour le raisonnement approximatif est exprimé par

- Hypothèse 1 (cause) : x est A' et y est B'
- Hypothèse 2 (règle) : si x est A_1 et y est B_1 alors z est C_1

- Conséquence (conclusion) : z est C'.

La règle floue dans l'hypothèse 2 peut être mise sous la simple forme suivante : « A x B → C ».

Intuitivement, cette règle floue peut être transformée en une relation ternaire R, qui est spécifiée par la fonction d'appartenance suivante :

$$\mu_R(x, y, z) = \mu_{(A \times B) \times C}(x, y, z) = \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z). \qquad (I.19)$$

Le résultat **C'** est alors exprimé par :

$$C' = (A' \times B') \circ (A \times B \rightarrow C)$$

Ainsi,

$$\begin{aligned}\mu_{B'}(z) &= V_{x,y}[\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge [\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)] \\ &= V_{x,y}[\mu_{A'}(x) \wedge \mu_{B'}(y) \wedge \mu_A(x) \wedge \mu_B(y)] \wedge \mu_C(z) \\ &= V_x[\mu_{A'}(x) \wedge \mu_A(x)] \wedge V_y[\mu_{B'}(y) \wedge \mu_B(y)] \wedge \mu_C(z) \\ &= \omega_1 \wedge \omega_2 \wedge \mu_C(z) \end{aligned} \qquad (I.20)$$

Où ω_1 est le degré de ressemblance entre A et A' ; et ω_2 est le degrés de ressemblance entre B et B', et $\omega_1 \wedge \omega_2$ est appelé le degré d'activation de la règle floue. Une interprétation graphique est donnée sur la **figure.I.8** où la MF de l'ensemble flou résultant C' est égale à la MF de C sectionnée par $\omega_1 \wedge \omega_2$. La généralisation pour plus de deux antécédents est directe.

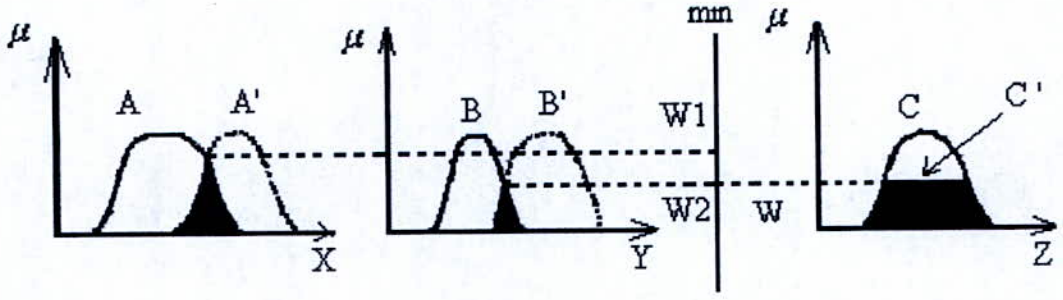


Figure I.8 : Raisonnement approximatif dans le cas multi antécédents

b. Multi règles, multi antécédents.

L'interprétation de multi règle est souvent donnée par l'union entre deux relations correspondantes à la règle floue. Soit par exemple les hypothèses suivantes :

- Hypothèse 1 (cause) : x est A' et y est B'
- Hypothèse 2 (règle 1) : Si x est A_1 et y est B_1 alors z est C_1
- Hypothèse 3 (règle 2) : Si x est A_2 et y est B_2 alors z est C2

- Conséquence (conclusion) : z est C'

On peut utiliser le raisonnement logique montré sur la **figure.I.9** comme procédure d'inférence afin d'en déduire les résultas de l'ensemble flou C'.

Afin de vérifier la procédure d'inférence, soit $R_1 = A_1 \times B_1 \rightarrow C_1$ et $R_2 = A_2 \times B_2 \rightarrow C_2$ Puisque l'opérateur de composition max-min, « o », est distributif par rapport à l'opérateur $\cup$ alors on obtient les résultats suivants :

$$C' = (A' \times B') \circ (R_1 \cup R_2) = [(A' \times B') \circ R_1] \cup [(A' \times B') \circ R_2] = C_1' \cup C_2' \qquad (I.21)$$

Où C_1' et C_2' représentent les ensembles flous inférés à partir des règles 1 et 2, respectivement. La **figure.I.9** illustre clairement le raisonnement flou pour les multi règles et multi antécédents. Lorsqu'une règle floue donnée prend la forme :

« Si x est A ou y est B alors z est C »,

Alors le degré d'activation est égal au maximum des degrés de ressemblance dans la partie antécédente pour une condition donnée.

Cette règle floue est équivalente à l'union entre les deux règles floues :

« Si x est A alors z est C »

Et

« Si y est B alors z est C ».

Si et seulement si la composition max- min est adoptée.

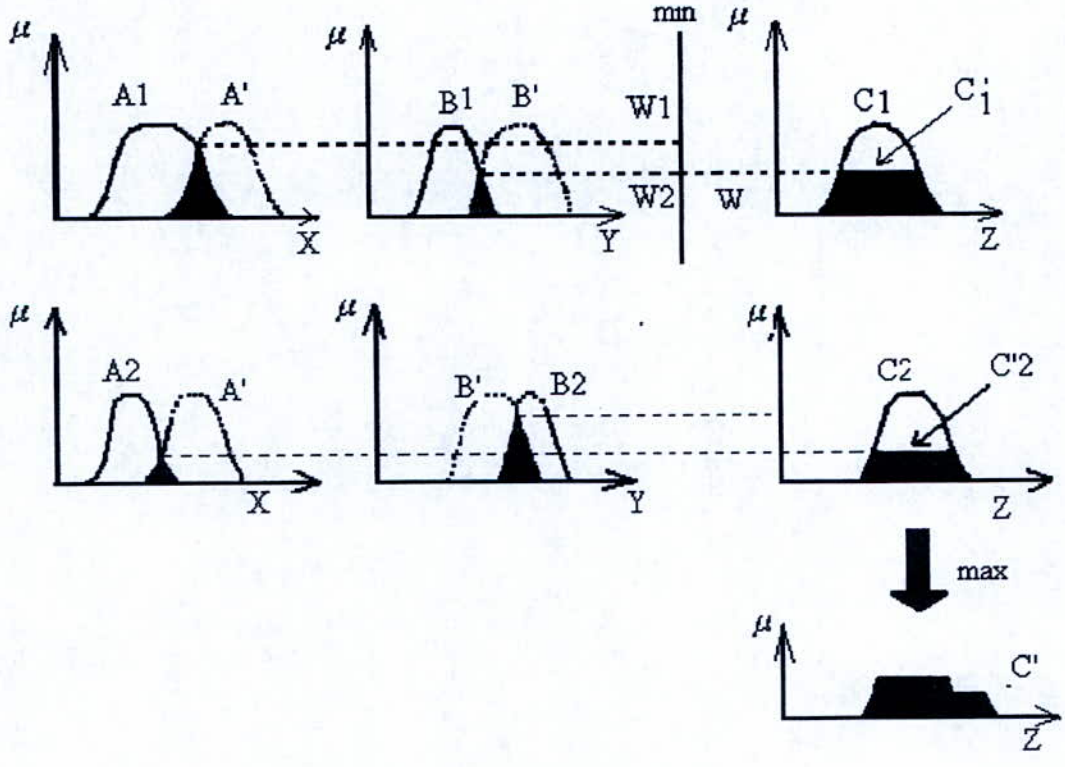


Figure I.9 : Raisonnement flou dans le cas multi règles, multi antécédents

I.1.5. LES SYSTEMES D' INFERENCES FLOUS [15]

Le système d'inférence floue FIS, est un système courant dans la théorie des ensembles flou, des règles floues « Si,...,alors » et du raisonnement flou. Il a été introduit avec succès dans les domaines du contrôle automatique, de la classification des données, l'analyse de la décision et des systèmes experts. A cause de ses diverses disciplines, le FIS est connu sous différents noms comme par exemple : « système à base de règle floue », « système expert », « modèle flou », « mémoire associative floue », « contrôleur logique flou » ou simplement « système flou ».

La structure de base d'un FIS consiste en trois composants conceptuels: une base de règles, qui contient une sélection des règles floues, une base de données, qui définit les fonctions d'appartenance utilisées dans les règles floues et un mécanisme de raisonnement qui exécute les procédures d'inférence (souvent, le raisonnement flou introduit précédemment) sur les règles et des conditions données pour en déduire une conclusion raisonnable.

Dans ce qui suit, nous allons introduire deux des principaux types de FIS.

I.1.5.a. LE MODELE FLOU DE MAMDANI

Le modèle flou de MAMDANI a été proposé, dans un premier lieu, pour des tentatives de control de la machine à vapeur et des chaudières par un ensemble de règles linguistiques de control obtenues à partir d'opérateurs humains. La **figure I.10** illustre comment deux règles de FIS du type MAMDANI permettent de déduire la sortie z à partir des deux entrées x et y.

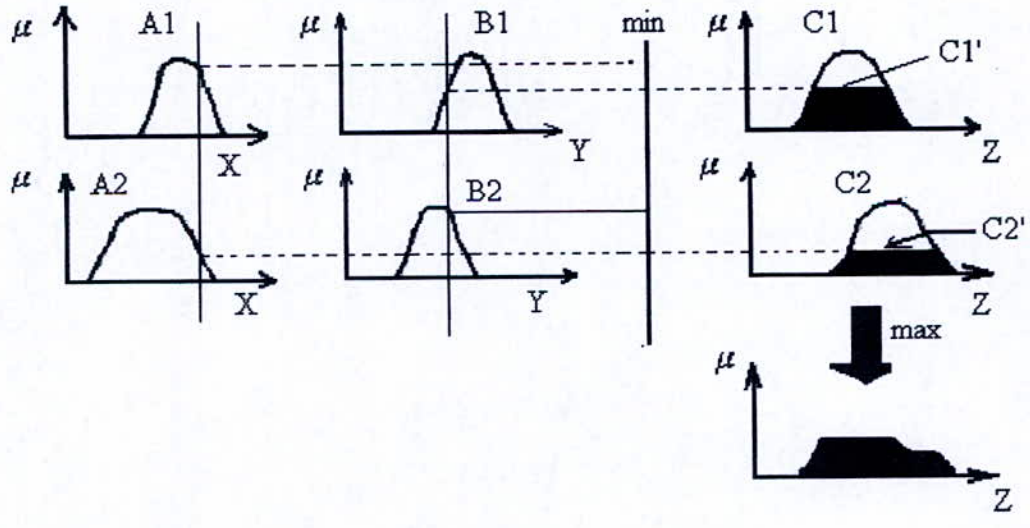


Figure I.10 : Modèle de MAMDANI utilisant min et max comme opérateurs ET et Ou respectivement

Si l'on adopte le **produit** et le **max** comme opérateurs flous ET et OU respectivement et utiliser la composition max-produit au lieu de la composition originale max-min, alors le raisonnement flou résultant est représenté sur la figure I.11 où la sortie inférée de chaque règle est un ensemble flou pondéré par son degrés d'activation à travers des produits algébriques. En dépit du fait que ce type de raisonnement flou n'a pas été utilisé dans le modèle original de MAMDANI, il a été introduit dans la littérature. D'autres variantes sont possibles si on choisissait d'autres opérateurs flous ET (T-norm), et le OU (T-conorm).

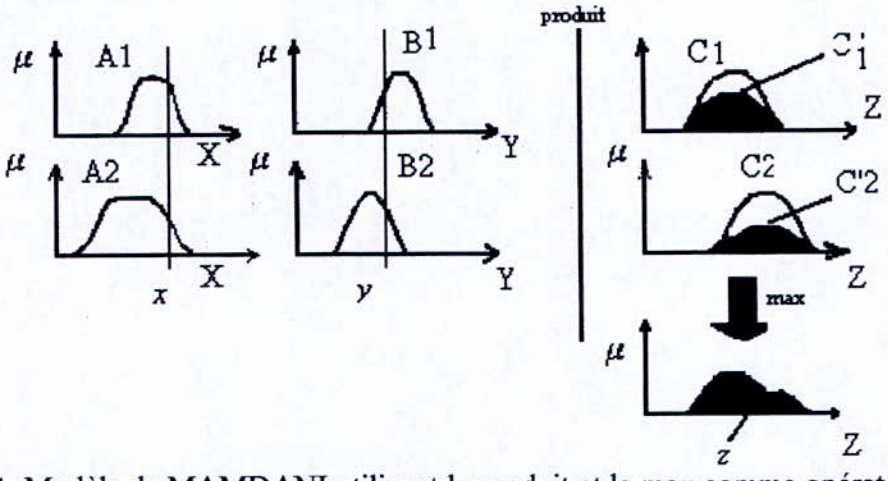


Figure I.11. Modèle de MAMDANI utilisant le produit et le max comme opérateurs ET et OU respectivement.

Dans les applications de MAMDANI, deux FIS ont été utilisées comme contrôleurs pour générer le fluide chaud dans la chaudière et accélérer l'ouverture du cylindre respectivement dans le but de réguler la pression de la vapeur dans la chaudière et la vitesse de la machine.

Etant donnée qu'à la sortie les valeurs sont floues, une Défuzzification est nécessaire pour convertir l'ensemble flou en en valeur numérique.

La défuzzification fait référence à la manière de convertir des valeurs floues extraites d'un ensemble flou en valeurs représentative.

La stratégie de défuzzification la plus utilisée est la stratégie du barycentre qui est définie par :

$$z_{CCA} = \frac{\int_Z \mu_{C'}(z) z dz}{\int_Z \mu_{C'}(z) dz} \qquad \text{(I.22)}$$

Où $\mu_{C'}(z)$ est la fonction d'appartenance de sortie correspondante. Cette formule rappelle le calcul d'espérance en probabilité. D'autres stratégies de défuzzification sont utilisées pour des applications spécifiques, on citera par exemple la méthode du plus grand ou du plus petit maximum.

La figure I.12 illustre ces différentes stratégies de défuzzification. En général, ces méthodes de défuzzification sont basées sur des calculs intensifs, sans aucune rigueur dans leur analyse. Sauf par des études expérimentales.

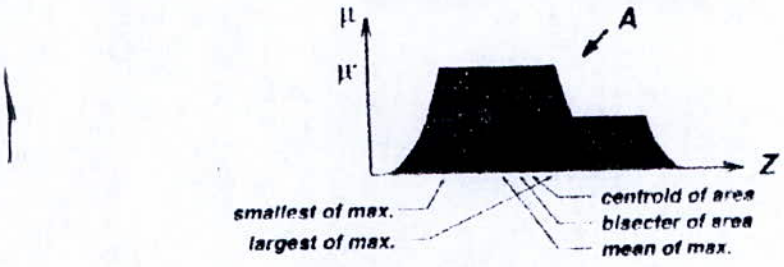


Figure I.12 : Stratégies de Défuzzification

Les figures I.10 et I.11 font référence au raisonnement flou introduit précédemment. En pratique, un FIS peut avoir certains mécanismes de raisonnement qui ne suivent pas d'une manière exacte la définition des règles d'inférence. Par exemple, l'on peut utiliser soit le min ou le produit pour le calcul du degré d'activation et/ou qualifier les règles de sortie. Une autre variation consiste à utiliser la sommation à la place du max dans les raisonnements flous standard malgré que la sommation ne soit pas réellement l'opérateur OR. Un avantage d'utiliser la composition somme/produit réside dans le fait que la sortie floue à travers la défuzzification par barycentre est égale aux moyennes pondérées de chaque règle de sorties floues, où le facteur pondéré de chaque règle est égal à son degré d'activation multiplié par la surface de la MF de sortie, et la sortie floue d'une règle est égale à la valeur du barycentre issue de la défuzzification de sa MF. Ceci permet la réduction des calculs lourds si l'on obtient la surface ainsi que le barycentre de chaque MF de sortie à l'avance.

I.1.5.b. LE MODELE FLOU DE SUGENO

Le modèle flou de SUGENO connu aussi sous le nom du modèle flou de TSK a été proposé par TAKAGI, SUGENO et KANG [16], dans le but de développer une approche systématique pour la génération de règles floues à partir d'un ensemble de données d'E/S.

Une règle floue typique dans le modèle de SUGENO a la forme suivante :

Si x est A et y est B alors $z = f(x,y)$

Où A et B sont des ensembles flous dans l'antécédent, alors que $z = f(x, y)$ est une fonction analytique, généralement un polynôme en x et y, mais peut être n'importe quelle fonction permettant de décrire d'une manière appropriée la sortie du système dans la région floue spécifiée par l'antécédent de la règle. Lorsque $f(x, y)$ est un polynôme du premier ordre, le FIS résultant est appelé modèle flou de SUGENO du premier ordre. Quand $f(x,y)$ est constante, on obtient un modèle flou de SUGENO d'ordre zéro, qui peut être considéré comme étant un cas particulier du système d'inférence flou de MAMDANI, dans lequel chaque règle conséquente est représentée par un singleton flou (une conséquence prédéfuzzifiée), ou bien un cas particulier du modèle flou de TSUKAMOTO, dans lequel la règle conséquente est représentée par une MF échelon dont les croisement s'effectuent à des points constants.

Il est à noter que la sortie d'un modèle de SUGENO d'ordre zéro est une fonction lisse des variables d'entrées tant que les MFs voisines sont suffisamment recouvertes.

La figure I.13 illustre la procédure du raisonnement flou pour un modèle flou de SUGENO du premier ordre. Etant donné que chaque règle a une sortie représentative, l'ensemble des sorties est obtenu à travers une moyenne pondérée et ainsi le temps mis par la procédure de défuzzification est évité. En pratique, des fois l'opérateur de la moyenne pondérée est remplacé par l'opérateur de la somme pondérée (qui est $z=\omega_1 z_1 + \omega_2 z_2$ dans la figure I.13) dans le but de réduire la lourdeur des calculs en particulier dans les FIS. Cependant, cette simplification peut engendrer la perte du sens linguistique des MF à moins que la somme des degrés d'activation soit proche de l'unité.

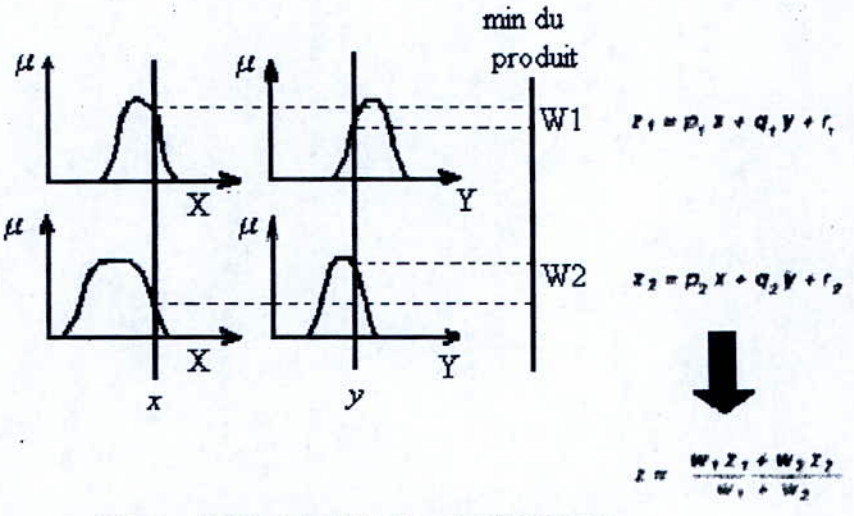
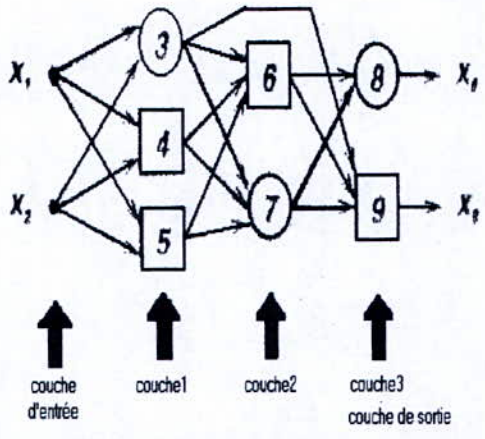


Figure I.13 : Modèle flou de SUGENO

PARTIE 2 : QUELQUES NOTIONS SUR LES RESEAUX ADAPTATIFS

a. Architecture :

Comme son nom l'indique, un réseau adaptatif (**figureI.14**) est un réseau dont le comportement général de ces paires d'**E/S** dépend des valeurs tirées d'une collection de paramètres modifiables. Plus précisément, la configuration de ce type de réseau est composée d'un ensemble de nœuds connectés par l'intermédiaire de liaisons directes, où chaque nœud est en fait un processus qui exécute une fonction de nœud statique sur le signal d'entrée dans le but de générer un signal de sortie, et selon la direction spécifiée par les connections, ce signal ira d'un nœud à un autre. Habituellement une fonction nœud est une fonction paramétrée comptant des paramètres modifiables.



FigureI.14. réseau adaptatif «feedforward» en représentation en couche

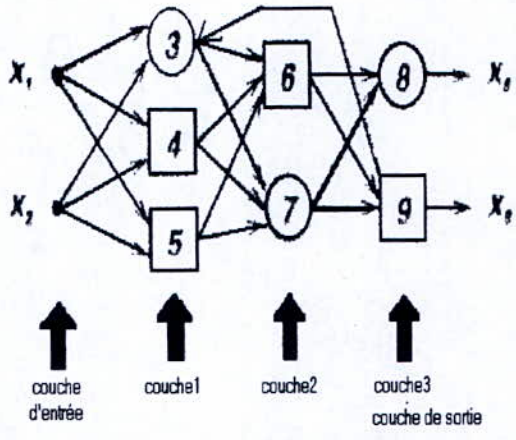


figure.I.15 réseau adaptatif récurant

Dans la plupart des cas, les réseaux adaptatifs sont hétérogènes. Chaque nœud peut avoir une fonction nœud différente. Il est à noter également que chaque liaison dans un réseau adaptatif n'est seulement utilisée que pour spécifier la direction de la propagation des signaux du nœud. Les liaisons ne sont ni paramètrées ni pondérées. La **figure.I.14** illustre un réseau adaptatif typique possédant deux entrées et deux sorties.

Chaque nœud possède un ensemble de paramètres qui est local. De ces ensembles locaux est tiré l'ensemble global des paramètres du réseau.

Si l'ensemble des paramètres d'un nœud n'est pas vide, sa fonction nœud dépend alors de la valeur de ces paramètres, dans ce cas on utilise un carré pour représenter ce type de nœud adaptatif. Dans le cas contraire, c'est à dire que si l'ensemble des paramètres nodaux est vide, la fonction est fixe, on utilise alors un cercle pour représenter ce type de nœud fixé.

Les réseaux adaptatifs sont en générale repartis en deux catégories sur la base des types de connexions utilisées. Nous avons : les réseaux récurant et feedforward. Quand un signal est injecté dans le nœud par le coté des entrées (gauche) alors nous sommes dans le cas d'un réseau feedforward comme illustre sur la **figure.I.14**, dans le cas contraire, s'il y a un retour

d'information sur un quelconque nœud alors nous sommes dans le cas d'un réseau récurant comme illustré sur la **figure.I.15.**

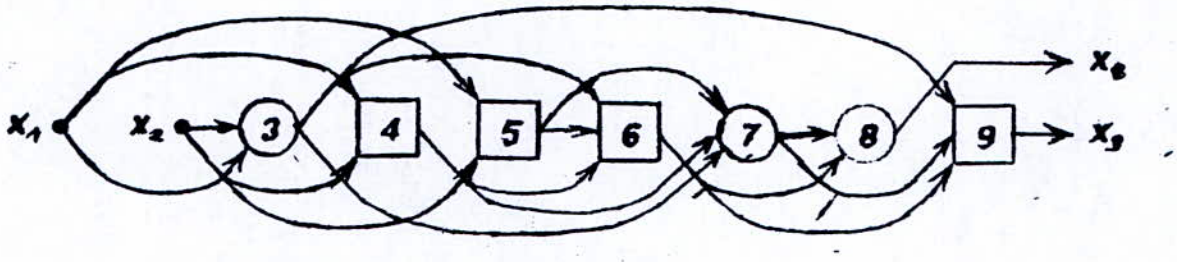


Figure.I.16. Représentation d'ordre topologique d'un reseau adaptatif feedforward

Dans la représentation en couche d'un réseau adaptatif feedforward, on note l'inexistence de liaisons entre des nœuds de la même couche, et les signaux de sorties de tous les nœuds sont toujours réinjectés dans les nœuds des couches suivantes. Cette représentation est appréciée pour sa modularité, car les nœuds de chaque couche ont la même fonctionnalité, ou génère les mêmes degrés d'abstraction pour les vecteurs d'entrées. Il existe une autre représentation des réseaux feedforward tel la représentation d'ordre topologique, où les labels des nœuds sont établis dans une séquence ordonnée : 1,2,3..., de telle manière à ne pas avoir de liaisons d'un nœud i à un nœud j si $i \geq j$. La **figure.I.16** est une représentation d'ordre topologique du réseau de la figure.I.14. Cette représentation perd en modularité si on la compare aux deux structures précédantes, mais facilite la formulation des règles d'apprentissage.

Conceptuellement, un réseau adaptatif feedforward est une relation statique entre l'espace des entrées et l'espace des sorties. Ce tracé peut être : soit un simple rapport linéaire, soit une relation fortement non linéaire ; cela dépendra de la structure du réseau et des fonctions de chaques nœuds. Le but de la construction de réseau de ce type est d'arriver à identifier un système non linéaire, et de lui réaliser une commande, à partir d'une base de données constituée de paires d'**E/S** de ce système. Cet ensemble de données est communément appelé ensembles de données d'apprentissage. La procédure, qu'on suit pendant l'ajustement des paramètres, pour améliorer les performances du réseau, est appelée **algorithme d'apprentissage**.

D'ordinaire la performance d'un réseau adaptatif est mesurée par la proportion que prend l'écart entre la sortie désirée et la sortie du réseau sous les mêmes conditions d'entrées. En générale cet écart est appelé **erreur de mesure**, elle peut prendre différentes formes pour différentes applications. Proprement dit, un apprentissage dérive d'une technique spécifique d'optimisation pour une mesure d'erreur donnée.

b. APPRENTISSAGE DES REGLES PAR RETRO-PROPAGATION

Le principal souci dans l'apprentissage d'un réseau adaptatif est comment obtenir, par récurrence, le vecteur gradient dont chaque élément représente la dérivée de l'erreur de mesure par rapport à un paramètre. Cela est fait en utilisant une chaîne de règles ; cette méthode est connue sous le nom de «règle d'apprentissage par rétro-propagation ». Cette

dénomination est due au fait que le vecteur gradient est calculé dans le sens inverse à celui de la sortie de chaque nœud.

Prenons l'exemple du réseau adaptatif en feedforward de la **figure.I.14** avec une représentation à **L** couches, avec $l = 0,\ldots,\mathbf{L}$; $l = 0$ représente la couches des entrées. Supposons que le réseau considéré possède $N(l)$ nœuds. Alors la sortie et la fonction du nœud **i** ($i = 0,\ldots,N(l)$) peut être représenté par $x_{l,i}$ et $f_{l,i}$,respectivement, tel présenté sur la **figure.I.17.a**. Comme illustré dans la **figure.I.17.b**, nous supposerons qu'il n y pas de liens entre des couches successives. Puisque la sortie des nœuds dépend des signaux d'entrées et de ses paramètres, nous avons l'expression générale de la fonction d'un nœud, $f_{l,i}$,

$$x_{L,i} = f_{L-1}(x_{l-1,1},\ldots,x_{l-1,N(l-1)}, \alpha, \beta, \gamma, \ldots) \qquad \text{(I.23)}$$

Ou α,β,γ,. . .sont les paramètres de ce nœud

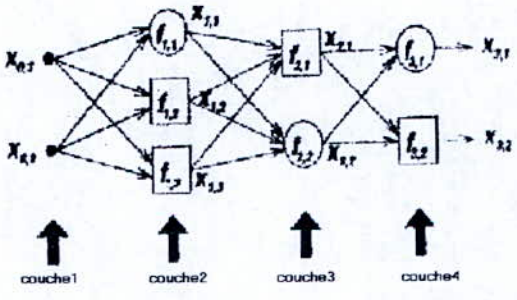


Figure.17.a. représentation en couches

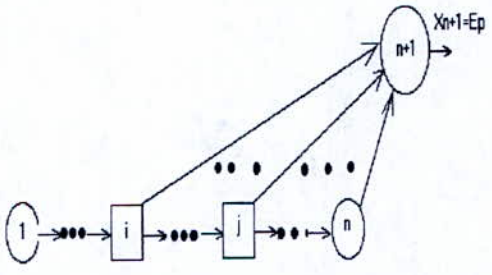


Figure.I.17.b. d'ordre topologique

Supposons que l'ensemble des données d'entraînement a P échantillons d'E/S, nous pouvons déterminer une erreur de mesure pour le p éme échantillon de données d'E/S comme étant la somme des carrées des erreurs

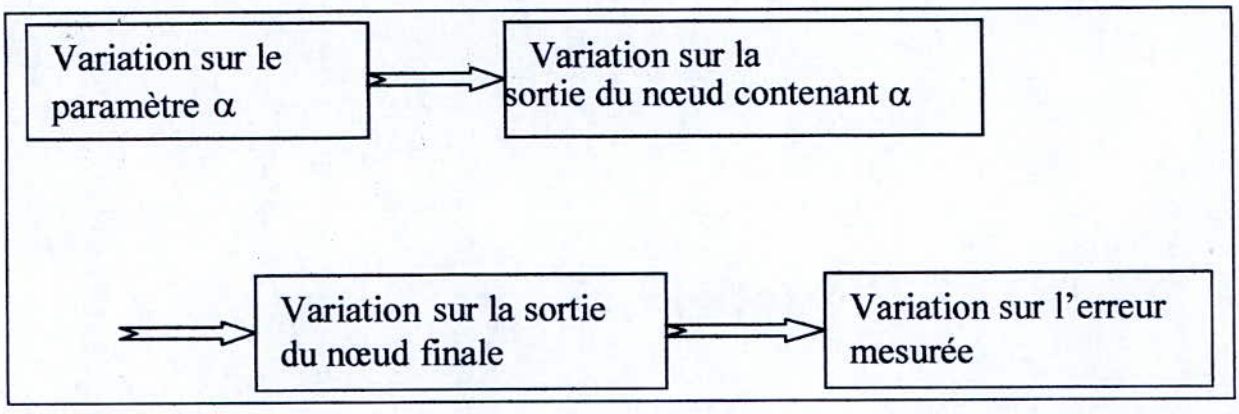
$$E_p = \sum_{k=1}^{N(L)} (d_k - x_{L,k})^2 \qquad \text{(I.24)}$$

Ou d_k, est la k ème composante du p éme vecteur de sortie désirée, et $x_{L,k}$ est la k éme composante du vecteur de sortie obtenue après l'introduction du p éme vecteur d'entrées dans le réseau. Evidemment quand E_p est égale a zéro, le réseau est capable de reproduire exactement la sortie désirée au p éme vecteur d'entrées. Ainsi il nous est imposé ici de minimiser l'erreur de mesure globale qui est définie par :

$$E = \sum_{p=1}^{P} E_p \qquad \text{(I.25)}$$

Il est à noter que la définition de E_p dans (I.24) n'est pas unique, d'autres définitions de E_p sont possibles pour des situations et applications spécifiques. Aussi, nous supposons que E_p dépend uniquement des sorties des nœuds.

Pour minimiser l'erreur de mesure en utilisant la méthode du gradient, premièrement nous devons obtenir le vecteur du gradient ; mais avant que ce calcul ne se fasse nous devons observer le fait que :



Où $\Rightarrow$ représente une relation de causalités. En d'autres mots, une petite variation sur α va affecter la sortie du nœud contenant α ; ce qui affectera la sortie de la dernier couche ainsi l'erreur mesurée. Donc le concept de base, dans le calcul du vecteur gradient des paramètres, est de transmettre une forme d'information sur la dérivée partant de la couche de sortie revenant en arrière couche après couche jusqu'à atteindre la couche des entrées (d'où d'ailleurs le nom de **rétro-propagation**).

Dans le but de nous faciliter la tache, le signal d'erreur $\varepsilon_{l,i}$ sera défini comme étant la dérivée de l'erreur de mesure E_p tout en prenant en considération que la sortie du nœud i de la couche l prend les deux chemins d'accès direct et indirect :

$$\varepsilon_{l,i} = \frac{\partial^+ E_p}{\partial x_{l,i}} \qquad \text{(I.26)}$$

Cette expression est appelée **dérivée ordonnée** de **Werbos.** La différence entre les dérivées ordonnées et les dérivées partielles ordinaire résident dans la manière de voir la fonction sur laquelle on applique une dérivation. Pour une sortie d'un nœud interne $x_{l,i}$ ou ($l \neq$ L) la dérivée partielle est égale à zéro si E_p ne dépend pas directement de $x_{l,i}$ .toutefois, il est évident que E_p dépend de $x_{l,i}$ indirectement, étant donnée qu'une variation sur $x_{l,i}$ se propage indirectement (vers la couche de sortie) ainsi est produite une variation sur la valeur de E_p. Donc, $\varepsilon_{l,i}$ peut être considéré comme étant le rapport de ces deux variations quand elles sont infimes.

Chapitre II

Les réseaux Neuro-Flous

II.1. INTRODUCTION [17]

La commande intelligente est une approche interdisciplinaire qui combine l'informatique, la théorie de la commande et des techniques d'intelligences artificielles. Il est connu que les systèmes intelligents, qui ont la capacité de fournir une expertise et un raisonnement incertain à caractère humain et pouvant s'adapter à un environnement bruité et variable dans le temps, occupent une place importante dans le traitement des problèmes pratique relatifs à la commande intelligente.

Les contrôleurs neuraux conventionnels(NC) utilisent des techniques d'apprentissage dans le but est de déterminer les grandeurs des variables de commande, tandis que les contrôleurs flous (FLC) utilisent des règles d'inférence floues et des variables linguistiques. Les algorithmes d'optimisation globale (GOA), tel que les algorithmes génétiques (GA), on été largement utilisés dans le but d'optimiser les architectures NC et FLC. La **figureII.1** décrit les différentes combinaisons entre ces différentes techniques conduisant à des contrôleurs intelligents.

La commande intelligente a connu un nouvel essor avec l'avènement d'une nouvelle approche dite commande neuro-flou (NFC), celle ci est capable de faire un apprentissage et un raisonnement sur les différents états d'un système.

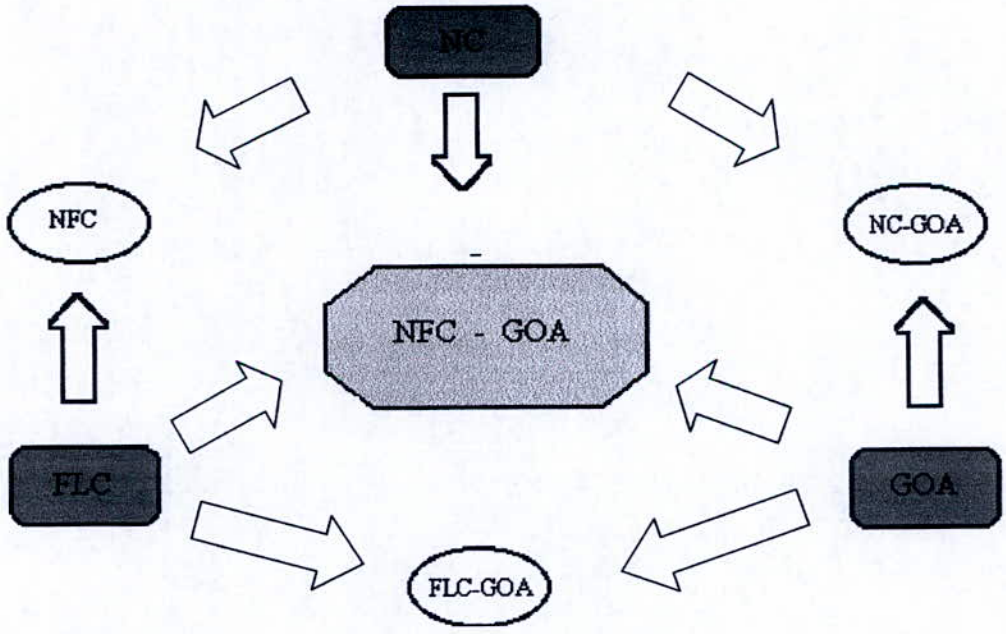


Figure II.1 : Structure générale des contrôleurs intelligents .

II.2. LES SYSTEMES NEURO-FLOUS

Un FLC exploite une expertise humaine en mémorisant ses composantes essentielles dans sa base de règles et de données, par la suite, il infère un raisonnement flou pour aboutir à une valeur, de la sortie, globale. Toutefois il n'existe pas de méthodes systématiques qui puisse nous donner une base de connaissance d'un FLC à partir d'une expertise humaine. Il existe, aussi, un besoin d'adaptation ou d'utilisation d'algorithmes d'apprentissage pour arriver à une erreur de modélisation dans des limites acceptables ; Par contre le mécanisme d'apprentissage d'un NC n'a pas besoin de l'appui d'une expertise humaine.

Du fait de l'homogénéité de sa structure, il est dur d'extraire des informations utilisables à partir des pondérations de la configuration du NC. Pour bon nombre de problèmes pratiques, des informations prédéterminées sur les systèmes, sont obtenu grâce au experts et il est plus approprié de les mettre sous forme d'ensembles flous et d'utiliser des règles floues du type (Si.... Alors), cependant il n'est pas aisé d'introduire ce type de données dans un NC. Dans le tableau ci-dessous nous allons comparer les NCs et les FLCs

NC	FLC
boite noire	Interprétables sous forme de règle «si...alors»
Algorithmes d'apprentissage compliqué	Simplicité d'interprétation et d'implémentation
Difficulté d'extraction de données	Avoir obligatoirement une expertise du système
Impossibilité d'utilisation de règles à base d'antécédents	Possibilité d'incorporation de règles à base d'antécédents
auto-construction	Utilise des connaissances linguistiques Qui ne supporte pas un apprentissage

Tableau.II.1. [20] Comparaison entre les caractéristiques des NCs et des FLCs

Donc, il va de sois, qu'il est plus approprié de combiner les avantages des deux approches afin d'obtenir des structures capables de faire un apprentissage et profiter en même temps des expertises existantes. Sachant qu'un algorithme d'apprentissage sur un ANN conventionnel ne peut être directement utilisé sur des FIS dont les fonctions utilisées lors du processus d'inférence sont non diffèrenciables (ex : MFs triangulaires), on pourra contourner ce problème en utilisant des fonctions diffèrenciables (MFs gaussiennes). Les structures ainsi obtenues sont appelées réseaux neuro-flous.

Dans la littérature on trouve divers modèles de structure neuro-flous, parmi elles : le **SONFIN**(Self Constructive Neural Fuzzy Inference Network) et l'**ANFIS**(Adaptative Neuro-Fuzzy Inference System). Nous allons présenter par la suite ces deux structures tout en mettant en évidence les complémentarités qui existent entre les ANNs et les FIS(Fuzzy Inference System) dans la perspective d'arriver à des systèmes intelligents plus performants.

II.3. LE SONFIN «SELF CONSTRUCTING NEURAL FUZZY INFERENCE NETWORK»

Le SONFIN est implémenté à partir d'une base de règles floues du type TAKGI-SUGENO-KANG qui est crée et adaptée par un procédé d'apprentissage ON-LINE identifiant, simultanément, la structure et les paramètres.

Dans la partie identification de la structure des antécédents, L'espace des entrées est partionné de manière flexible, grâce à un algorithme de clustering. Quant à la partie identification de la structure des conséquences ; une valeur est assignée, par la méthode du clustering, à chaque règle ainsi que des termes significatifs des variables d'entrées.

Pour l'identification paramétrique, les paramètres des conséquences sont optimisés par l'algorithme des moindres carrés ou de la méthode des moindres carrés récursifs ; les paramètres des antécédents sont obtenus par l'utilisation d'algorithmes par rétro-propagation.

Pour accroître l'habilité du SONFIN dans la représentation des connaissances, une transformation linéaire pour chaque variable d'entrée peut être incorporée dans le réseau, de façon à réduire le nombre de règle, et qu'un grand niveau de précision soit atteint. Les paramètres de cette transformation s'adaptent dynamiquement dans la phase d'identification du SONFIN. Lorsque des opérations ON-LINE sont requises, le SONFIN peut être utilisé à n'importe quel instant durant le processus d'apprentissage.

II.3.1. STRUCTURE D'UN SONFIN

Dans cette partie nous introduirons la structure générale du SONFIN (tel présenté sur la **figure.II.2.**) Ce réseau a six couches et est construit sur la base d'un modèle flou suivant la forme suivante :

$$i^{éme}\ Règle : (Si\ x_1\ est\ A_{i1}\ et \ldots et\ x_n\ est\ A_{in})\ Alors\ (y\ est\ m_{0i} + a_{ji}x_j + \ldots + a_{ni}x_n) \qquad \textbf{(II.1)}$$

Ou A_{ij} est un ensemble flou, m_{0i} le centre d'une fonction d'appartenance symétrique, et les a_{ji} sont les paramètres des conséquences. Il est à noter que dans l'équation linéaire de sortie, contrairement au modèle TSK ou toutes les variables d'entrées sont utilisées, seul les plus importantes d'entre elles sont utilisées dans le SONFIN.

Dans ce qui suit, nous allons définir les différentes fonctions caractérisant chacune des six couches de ce réseau, ensuite nous introduirons une structure améliorée du SONFIN

II.3.2. ARCHITECTURE DU SONFIN

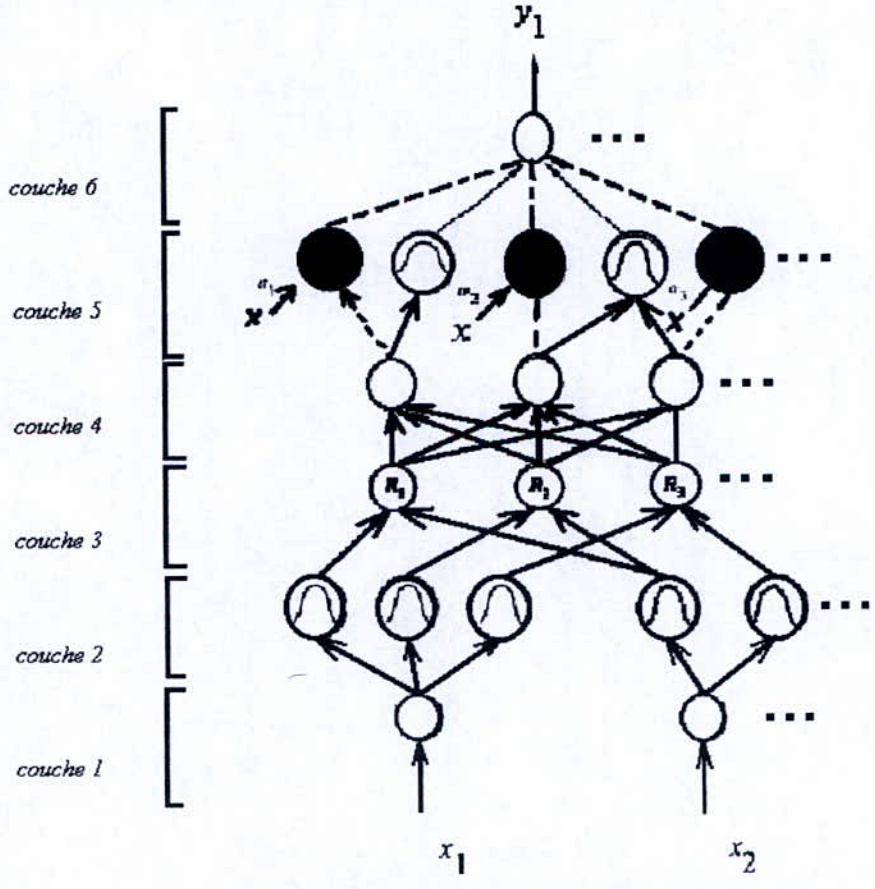
Le SONFIN est constitué d'un ensemble de nœuds, chaque nœud a un nombre fini d'entrées représentées par des valeurs pondérées des sorties de nœuds de la couche précédante, et possède aussi des sorties en direction des nœuds de la couche suivante. Chaque ensemble d'entrées d'un nœud est associé à une fonction qui sert à combiner l'information, l'activation ou l'évidence d'autres nœuds. Cette fonction nous donne le net-input de ce nœud.

$$\text{net-inputs} = f\,[u_1^{(k)}, u_2^{(k)}, \ldots, u_p^{(k)}; w_1^{(k)}, w_2^{(k)}, \ldots, w_p^{(k)}] \qquad \textbf{(II.2)}$$

Où : $u_1^{(k)}, u_2^{(k)}, \ldots, u_p^{(k)}$ sont les entrées de ce nœud, et, $w_1^{(k)}, w_2^{(k)}, \ldots, w_p^{(k)}$ sont les pondérations associés à chaque entrée. L'indice k dans les équations ci-dessus indique le numéro de la couche. Cette notation sera aussi utilisée dans les équations qui suivent. La deuxième action de chaque nœud est de fournir une activation de sortie (output activation) qui est une fonction de ces net-inputs.

$$\text{output} = O_i^{(k)} = a(f) \qquad \textbf{(II.3)}$$

Où a(...) dénote la fonction d'activation.



figureII.2 : structure d'un SONFIN

Nous décrirons ci-dessous les fonctions des nœuds de chaque couche du SONFIN

Couche 1 :

Dans cette couche aucun calcul n'est fait, chaque nœud correspond à une variable d'entrée et ne fait que transmettre sa valeur à la couche suivante, ce qui nous permet d'écrire :

$$f = u_i^{(1)} \qquad \textbf{(II.4)}$$

et

$$a^{(1)} = f \qquad \textbf{(II.5)}$$

De cette équation, l'on remarque que les pondérations des entrées de cette couche $[w_i^{(1)}]$ sont égale à l 'unité .

Couche 2 :

Chaque nœud de cette couche correspond à un terme linguistique (large, petit, grand, mince, etc.) d'une variable d'entrée de la couche 1. En d'autres mots, le degré d'appartenance de la valeur d'une variable d'entrée à un ensemble flou est calculé dans la couche 2. Il est plus indiqué d'utiliser des fonctions d'appartenances gaussiennes pour deux raisons : premièrement, il a été démontré qu'un système flou utilisant des fonctions d'appartenance gaussiennes est une approximation universelle pour n'importe quelle fonction non linéaire dans un ensemble compacte[6] ; Deuxièmement, une fonction d'appartenance gaussienne multidimensionnelle, générée pendant le processus d'apprentissage, peut être facilement décomposée en un produit de fonctions d'appartenance gaussiennes unidimensionnelle.

Avec le choix de fonction d'appartenance gaussienne, l'opération accomplie dans cette couche est :

$$f[u_{ij}(2)] = -[u_i^{(2)} - m_{ij}]^2 \Big/ \sigma_{ij}^2 \qquad \textbf{(II.6)}$$

$$a^{(2)}(f) = \exp(f) \qquad \textbf{(II.7)}$$

Où m_{ij} et σ_{ij} sont, respectivement, le centre (la moyenne) et la largeur (la variance) de la fonction d'appartenance gaussiène du $j^{ème}$ terme de la $i^{ème}$ variable d'entrée x_i . Donc, le poids des entrées dans cette couche peut être interprété comme étant les m_{ij}. Contrairement à quelques autres méthodes de partition, où chaque variable d'entrée a le même nombre d'ensembles flous, les variables d'entrées n'ont pas nécessairement le même nombre d'ensembles flous dans le SONFIN.

Couche 3 :

Un nœud dans cette couche représente une règle floue, il calcule le degré d'activation des antécédents de chaque règle. on utilise l'opérateur ET, suivant, pour chaque nœud de la couche 3 :

$$f[u_i^{(3)}] = \prod_i^n u_i^{(3)}$$

$$= exp(-[D_i(x-m_i)]^T[(D_i(x-m_i)]) \quad \textbf{(II.8)}$$

Et

$$a^{(3)}(f) = f \quad \textbf{(II.9)}$$

Où n est le nombre des nœuds de la couche 2 qui participent dans le «Si » de la règle,

$$D_i = diag(1/\sigma_{i1}, 1/\sigma_{i2}, ... 1/\sigma_{in}) \quad \textbf{(II.10)}$$

et

$$m_i = (m_{i1}, m_{i2}, ..., m_{in})^T . \quad \textbf{(II.11)}$$

Les pondérations des entrées dans la troisième couche est l'unité. La sortie f dans un nœud de la couche 3 représente le degré d'activation de chaque règle.

Couche 4 :

Le nombre de nœuds dans cette couche est égal au nombre de nœuds dans la couche 3 ; La fonction de chaque nœud est de normaliser les degrés d'activation des règles provenant des nœuds de la couche 3.

$$f[u_i^{(4)}] = \sum_i u_i^{(4)} \quad \textbf{(II.12)}$$

$$a^{(4)}(f) = u_i^{(4)} / f \quad \textbf{(II.13)}$$

Comme pour la couche 3, les pondérations des entrées de la couche 4 est l 'unité.

Couche 5 :

Cette couche est appelée couche des conséquences. Deux types de nœuds sont utilisés, ils sont représentés par des cercles blancs et noirs comme illustré dans la **figure.II.2.** Les nœuds représentés par des cercles blancs sont les principaux, ils représentent les ensembles flous (décris par des fonctions d'appartenance gaussienne) des variables de sortie. Seules les centres de ces fonctions d'appartenance sont délivrés à la couche suivante (défuzzification par la moyenne locale du maximum), et la largeur (variance) est utilisée seulement pour la partition.

Différents nœuds de la couche 4 peuvent être connectés au même nœud de la couche 5, ce qui signifie que plusieurs règles peuvent avoir la même conséquence. La fonction d'un nœud blanc est :

$$f = \sum_i u_i^{(5)} \quad \textbf{(II.14)}$$

Et

$$a^{(5)}(f) = f.a_{0i} \qquad \textbf{(II.15)}$$

Où $a_{0i} = m_{0i}$ est le centre de la fonction d'appartenance gaussienne représentant une variable de sortie. Les nœuds noirs ne sont générés que s'il est nécessaire. Chaque nœud de la couche 4 a sont unique nœud noir, lui correspondant, dans la couche 5. Une des entrées d'un nœud noir est la sortie de son nœud correspondant de la couche 4, et les autres entrées sont les variables d'entrées de la couche 1.

La fonction d'un nœud noir est :

$$f = \sum_{j} a_{ji} x_j \qquad \textbf{(II.16)}$$

et

$$a^{(5)}(f) = f u_i^{(5)} \qquad \textbf{(II.17)}$$

Où la sommation est appliquée seulement sur les termes connectés à ce nœud, et a_{ji} est le paramètre correspondant à chaque entrée. En combinant les deux types de nœuds de la couche 5 nous obtenons la fonction globale de sotie de cette couche.

$$a^{(5)}(f) = \Big(\sum_{j} a_{ji} x_j + a_{0i}\Big) u_i^5 \qquad \textbf{(II.18)}$$

Couche 6 :

Chaque nœud de cette couche correspond à une variable de sortie. Il somme toutes les entrées issues de la couche 5 et agit comme defuzzificateur avec :

$$f[u_i^{(6)}] = \sum_{i} u_i^{(5)} \qquad \textbf{(II.19)}$$

et

$$a^{(6)}(f) = f\ . \qquad \textbf{(II.20)}$$

II.3.3. STRUCTURE DU SONFIN AMELIOREE

Pour la structure du SONFIN citée précédemment, la région de l'espace, couverte par la fonction d'appartenance, est restreinte à une forme elliptique, avec les axes de l'ellipse qui sont parallèles aux axes des coordonnées des entrées correspondantes ; comme illustré sur la **figure.II.3.a**. Habituellement, une région de forme elliptique ne couvre pas d'une bonne manière la distribution des échantillons d'entrée. Par exemple, si la distribution des échantillons d'entrée est pareille à la région nuancée sur la **figure.II.3.b**, indiquant que les variables d'entrées sont fortement corrélées entre elles, alors le besoin en règle, pour lier cette région à celle lui correspondant dans l'espace des sorties, sera de toute évidence supérieure à une seule règle. Afin de réduire ce nombre de règles, une transformation linéaire est appliquée

sur les variables d'entrées du SONFIN. Cette transformation peut être considérée comme un changement des cordonnées d'entrées, tandis que les paramètres de chaque fonction d'appartenance sont gardés inchangés, en d'autres mots, la moyenne et la variance de chaque fonction d'appartenance, dans le nouveau repère, restent invariant. Cette transformation est mathématiquement donnée par :

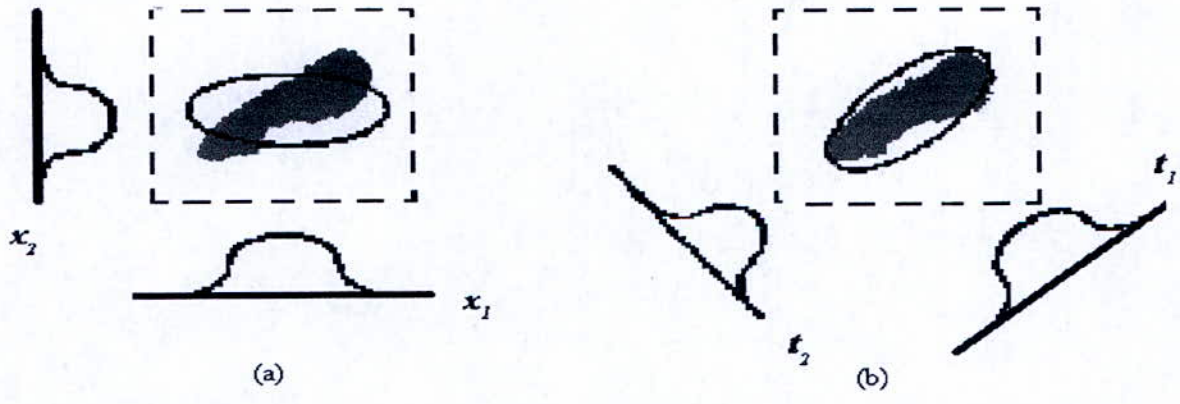
$$Règle\ i : \qquad t_i = R_i(x - m_i) + m_i \qquad \textbf{(II.21)}$$

Ou

$t_i \in \Re^n$ Sont les nouvelles variables dentrées générées

Et

$R_i \in \Re^{n \times n}$ est la transformation matricielle de la $i^{éme}$règle



figureII.3 (a) cluster avant la transformation ; (b) cluster après la transformation

Apres la transformation, la région que couvrent les fonctions d'appartenance de l'entrée est illustrée sur la **figureII.3.b**. Il est à noter que l'ellipse, après sa rotation, couvre bien la distribution des données d'entrée, et ainsi une seule règle floue peut associer cette région à sa région de sortie conséquente.

Du fait de la transformation que subissent les coordonnées d'entrées, le degré d'activation est calculé dans la couche 3 mais avec une fonction qui prendra une autre forme qui est :

$$f[u_i^{(3)}] = \prod_i^n u_i^{(3)}$$
$$= exp(-[D_i(t_i - m_i)]^T [(D_i(t_i - m_i)])$$
$$= exp(-[D_i R_i (x - m_i)]^T [(D_i R_i (x - m_i)] \qquad \textbf{(II.22)}$$

Et

$$a^{(3)}(f) = f \qquad \textbf{(II.23)}$$

Tous les paramètres de la transformation matricielle de **(II.21)** sont des paramètres indépendants. Toutefois, si l'on rajoute une nouvelle contrainte tel que $\boldsymbol{R_i}^T \boldsymbol{R_i} = \boldsymbol{I}$ alors d'un point de vue géométrique, l'opération R_i sera équivalente à une rotation de la région couverte par les fonctions d'appartenance originales. Dans cette situation, $\boldsymbol{m_i}$ aura comme tache la localisation de la fonction d'appartenance, D_i pour sa largeur, et finalement, $\boldsymbol{R_i}$ pour l'orientation des repères des entrées, la règle générale explicitée dans **(II.1)** deviens alors :

$$\begin{aligned} \textit{Règle i :} \quad & \textbf{\textit{Si}}\ t_{il} = \sum_{k=1}^{n} r_{lk}^{i}(x_k - m_{ik}) + m_{il}\ \textbf{\textit{est}}\ A_{il}\ \textbf{\textit{et}}\ ..... \\ & .... \quad t_{in} = \sum_{k=1}^{n} r_{nk}^{i}(x_k - m_{ik}) + m_{in}\ \textbf{\textit{est}}\ A_{in} \\ & \textbf{\textit{Alors}}\ \ y_i\ \ \textit{est}\ \ m_i. \end{aligned} \qquad \textbf{(II.24)}$$

Où r_{nk}^{i} est le (n, k)^{ème} élément de R_i. L'implication linguistique A_{in} de la variable originale x_n est maintenant impliquée par la nouvelle variable t_{ij}(voir **figureII.3** pour plus de clarté), qui est une combinaison linéaire des variables originales. Il faut noter que lorsque $\boldsymbol{R=I}$, alors, les règles obtenues après transformation sont les mêmes que les originales.

Généralement, la flexibilité offerte par ***Ri*** peut réduire le nombre de règles ou accroître la précision de la modélisation du SONFIN. Cette transformation est très utile dans les modèles à dimensions d'entrées réduites. Pour des entrées à grandes dimensions, cet avantage peut être compromis par un besoin additionnel en mémoires requis pour la sauvegarde de Ri.

II.4. ANFIS «ADAPTATIVE NEURO FUZZY INFERENCE SYSTEM »

l'ANFIS fut l'une des premières structures à avoir été élaboré par les chercheurs dans le domaine du neuro-flou adaptatif. Cette structure implémente des systèmes d'inférence flous du type TAKAGI-SUGENO, elle est caractérisée par une architecture à cinq couches comme on le remarque sur la **figure.II.4**

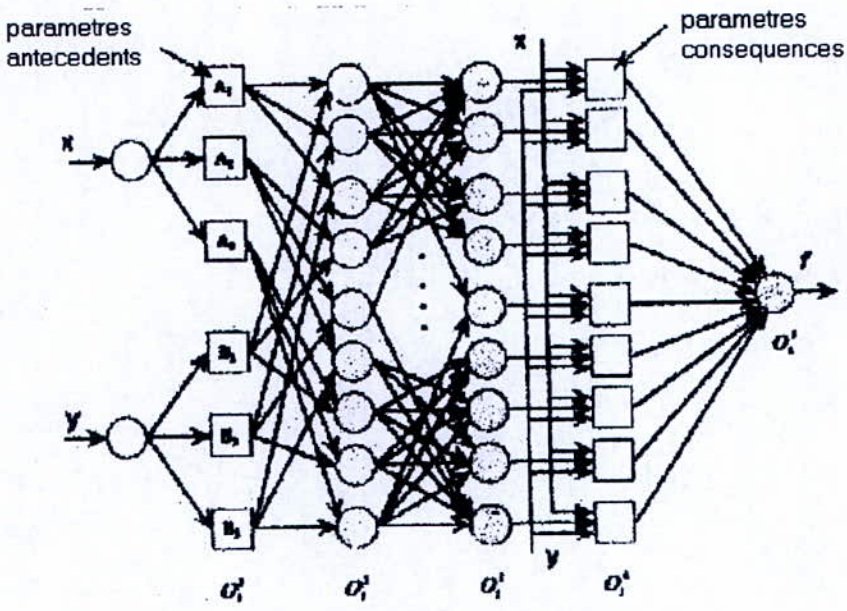
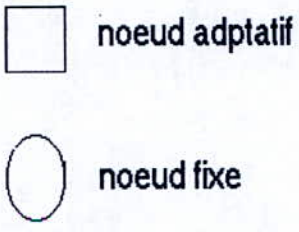


Figure II.4 architecture de l'ANFIS

Avec :



La première couche sert de fuzzificateur pour les différentes grandeurs d'entrées ; les opérateurs T-norm sont déployés dans la seconde dans le but de calculer les antécédents. Dans la troisième nous aboutirons à la normalisation des degrés d'activation qui par la suite passeront par la couche quatre qui nous donnera comme signal de sortie les paramètres de conséquences des règles ; Et finalement la cinquième couche nous fera aboutir au signal global de sortie.

l'ANFIS utilise des algorithmes d'apprentissage par rétro-propagation afin d'estimer les paramètres du système, et/ou la méthode des moindres carré pour l'identification des conséquences.

Pour ce qui est de la procédure d'apprentissage, elle se divise en deux parties : dans la première, les échantillons de données seront injectes dans la structure, et une estimation des paramètres des conséquences, de manière optimale, se fera grâce à la méthode des moindres carré récursif, par contre les estimées des paramètres des antécédents seront fixées pendant toute la durée du cycle courant. Dans la seconde partie les échantillons seront injectés une seconde fois, à cet événement sera associé une rétro-propagation qui permettra une réévaluation des estimées des paramètres des antécédents; les paramètres des conséquences quant à eux resteront fixes.

Cette procédure n'est en faite qu'une seule itération, et elle est amenée à être répéter autant de fois que la procédure l'exigera.

Dans ce qui suit nous verrons le détail des différentes couches, les nœuds qui les constituent et le détail des fonctions utilisées

Couche 1

Chaque nœud de cette couche représente une fonction de telle sorte a avoir,

$$O_i^1 = \mu_{Ai}(x) \quad pour\ i = 1,2$$

ou

$$O_i^1 = \mu_{Bi-2}(y) \quad pour\ i = 3,4 \qquad \textbf{(II.25)}$$

Ou O_i^1 représente le degré d'appartenance à l'ensemble flou A qui est composé de plusieurs sous-ensembles floues (A1, A2, B1, et B2) et nous spécifie aussi jusqu'à quel degré les entrées x, y satisfassent le quantificateur A.

En général on utilise des fonctions paramètres, comme fonction de nœud. Tel la fonction d'appartenance gaussienne qui est caractérisé par les paramètres c (moyenne) et σ (variance)

$$guaussien(x, c, \sigma) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})}$$

Couche 2

Les nœuds de cette couche sont représentés par l'opérateur «ET», «multiplication». En effet le signal de sorite n'est autre que le produit des signaux d'entrées et représente le degré d'activation de chaque règle. la fonction nœud de cette couche est tel que :

$$O_i^2 = w_i = \mu_{Ai}(x) \times \mu_{Bi}(y) \qquad \textbf{(II.26)}$$

Couche3

Dans cette couche on aura comme entrées les degrés d'activation de tout les nœuds de la couche précédente, et comme signal de sortie le quotient du degré d'activation de la **i** éme règle par la somme des degrés d'activations de toutes les autres règles on aura alors :

$$O_i^3 = \overline{w}_i = \frac{w_i}{w_1 + w_2} \quad , \quad i = 1,2\ldots\ldots \qquad \textbf{(II.27)}$$

Couche 4

chaque nœud de cette couche représente une fonction nœud

$$O_1^4 = \overline{w}_i f_i = \overline{w}_i (p_i + q_i + r_i) \qquad \textbf{(II.28)}$$

Cette équation représente la fonction qui nous permet d'aboutir au signal de sortie du i éme nœud il est à noter que {p, q, r }est l'ensemble des paramètres du système et que les paramètres recherchés dans cette couche représentent les conséquences.

Couche 5

Dans cette couche nous il n'existe qu'un unique nœud. Ce nœud nous fait aboutir à la sortie globale qui n'est autre que la somme des signaux de sortie des nœuds de la précédante couche.

CONCLUSION

Dans l'ANFIS le processus d'adaptation concerne uniquement les paramètres car il utilise une structure fixe. Pour un grand nombres de problèmes, il est très difficile de déterminer une structure antécédents-conséquences optimum ainsi que le nombre de règle. La structure de l'ANFIS assure que chaque terme linguistique n'est représenté que par un seul ensemble flou.

Chapitre III

Algorithme d'apprentissage du SONFIN

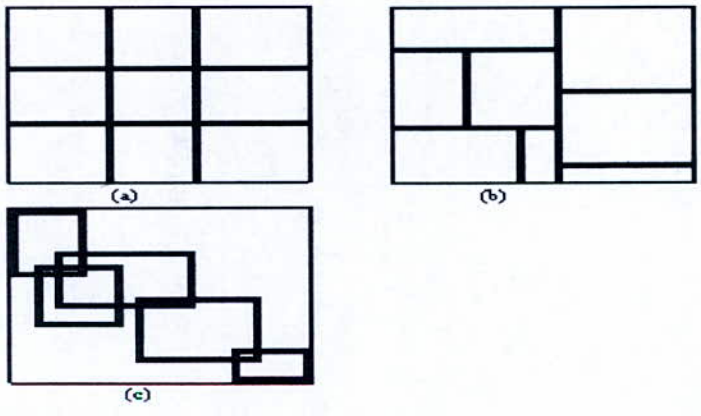
III.1.INTRODUCTION

Un système flou est constitué d'ensembles de règles du type«Si....Alors ». Conventionnellement, le choix des règles se base sur une quantité d 'observations heuristiques pour exprimer une stratégie de connaissance propre au système. Evidement, il est difficile pour un expert humain de balayer tout le champ des E/S d'un système complexe afin de trouver un nombre de règles représentatives de ce système. Pour contrecarrer ce problème, plusieurs approches pour la génération de règles à partir de données ont été proposées [8]-[9]. Généralement, ces approches consistent en deux phases d'apprentissage : l'apprentissage de la structure et l'apprentissage des paramètres. Traditionnellement, ces deux phases sont accomplies d'une manière séquentielle ; l'apprentissage de la structure est employé afin de bâtir une structure de règle, ensuite l'on passe à l'apprentissage des paramètres afin d 'affiner les coefficients de chaque règle. Un des inconvénients de cette approche réside dans le fait qu'elle ne peut s'appliquer que pour des opérations OFF-LINE d'où la nécessité d'avoir une quantité de données représentatives à l'avance, de plus elle prend beaucoup de temps. Pour parer à ce problème, il a était proposé un réseau neuro-flou auto-construit qui agit d'une manière ON-LINE (SONFIN : On-Line Self-Constructing Neural Fuzzy Inference Network), caractérisé par la simultanéité de l'apprentissage de la structure et des paramètres.

III.2.PARTITION DE L 'ESPACE DES ENTREES

Il est clair que le principe de fonctionnement des systèmes d'inférence floue applique la règle du «divises et tu conquerras ». En effet les antécédents d'une règle floue partitionnent l'espace des entrées en un nombre de régions floues, pendant que leurs conséquences décrivent le comportement dans une région donnée via des constituants variables qui peuvent être une fonction d'appartenance de sortie (modèles flous de Mamdani et Tsukamoto), une constante (modèle de Sugeno d 'ordre zéro), ou une équation linéaire (modèle de Sugeno du premier ordre).

Différents constituants des conséquences donnent de différents systèmes d'inférences floues, mais leurs antécédents restent inchangés. Pour cette raison, les notions, qui vont suivre , sur la partition de l'espace des entrées afin de former les antécédents d'une règle, sont applicables pour n'importe quel type de système d'inférence neuro-floue.



FigureIII.1 :
a. Partition en grille ; **b**. Partition avec la Méthode Hill-climbing **c.** Partition éparpillée

III.2.a. PARTITION EN GRILLE

La **figureIII.1.a** illustre une partition en grille typique dans un espace des entrées à deux dimensions. Cette méthode de partition est souvent utilisée dans la conception des contrôleurs flous, qui habituellement n'incluent que quelques variables d'état comme entrées du contrôleur. Cette stratégie de partition s'applique bien aux modèles avec un petit nombre de fonctions d'appartenance pour chaque entrée ; Mais elle rencontre des problèmes lorsque le nombre d'entrées est modérément large. Prenons l'exemple d'un système flou avec 10 entrées et deux fonctions d'appartenance. Chaque entrée va engendrer $2^{10} = 1024$ règles ; un nombre trop élevé. Ce problème peut être évité par l'utilisation d'autres méthodes de partition introduites ci-dessous

III.2.b. PARTITION DE L 'ESPACE DES ENTREES EN REGIONS EPARPILLEES (SCATTER PARTITION)

Comme illustré dans la **figure.III.1.c**, cette stratégie est basée sur le recouvrement d'un sous–espace de l'espace des entrées, qui caractérise la région dans laquelle tous les points sont susceptibles d'appartenir aux réalisations réelles du vecteur des entrées. Cette stratégie peut limiter le nombre de règles d'une manière raisonnable. Cette méthode sera en peu plus détaillée par la suite

III.2.c. PARTITION ARBORESCENTE PAR LA MÉTHODE HILL-CLIMBING (TREE PARTITION WHITH HILL-CLIMBING METHOD)

Une partition arborescente résulte d'une série de coupes ; chaque coupe est faite tout le long d'un sous-espace ; une région ainsi produite peut être sujette à une coupe indépendante. Au début de la $\mathbf{i}^{\text{éme}}$ itération, l'espace des entrées est partitionné en ***i*** régions ; suite à cela, une autre coupe est appliquée à l'une de ces régions dans le but d'obtenir ***i+1*** région.

Il existe plusieurs stratégies pour décider des dimensions à couper, de la position ou la coupe doit être faite et cela à chaque itération. Quelques-unes sont basées tout simplement sur la distribution des échantillons des entrées ; les autres prennent en considération les méthodes d'identification paramétriques. Sachant aussi que cette méthode est basée sur des fonctions objectives de partition :

Considérons n comme étant le nombre de règles que nous voulons obtenir, le choix de n est influencé par des considérations de performances. A chaque étape (itération), un ensemble flou est défini pour une région (cluster) avec la fonction d'appartenance suivante :

$$u_{ik}(\vec{x}) = \prod_{j=1}^{v} \frac{1}{1+\left[\left(\frac{x_{kj} - c_{ij}}{a_{ij}}\right)^2\right]^{b_{ij}}} \qquad \textbf{(III.1)}$$

Sachant que $u_{ik}(\vec{x})$ nous donne l'appartenance du $k^{ème}$ point $(\vec{x})$ dans la $i^{ème}$ région, v est le nombre de variables, x_{kj} est la $j^{ème}$ coordonnée de $\vec{x}_k$ et $\prod$ est l'opérateur flou «ET». Le calcul des paramètres a_{ij}, b_{ij}, c_{ij} est dépendant de l'hyper-réctangle définit par une région résultante des coupes. Soit h_i le centre physique du $i^{ème}$ hyper-réctangle, alors c_{ij} est définit comme étant la $j^{ème}$ coordonnée de h_i, a_{ij} est calculée comme étant la moitié de la longueur de la $j^{ème}$ dimension de cet hyper-réctangle, et b_{ij} est déterminé par le degré de recouvrement désiré entre deux régions floues.

A la fin de la phase d'identification de la structure, les a_{ij}, b_{ij}, c_{ij} sont introduit dans le réseau comme étants les paramètres initiaux.

Maintenant, sont définies deux fonctions objectives qui permettent de sélectionner la région à sectionner. Afin d'obtenir une structure ayant un sens pratique, le choix des mesures doit être approprié. Dans notre approche, nous allons utiliser les fonctions objectives suivantes :

La première est la mesure de la densité qui a été proposé par Ruspini [7] :

$$Jd = \sum_{j=1}^{P} \sum_{k=1}^{P} \left\{ \left[\sum_{i=1}^{C} (u_{ij} - u_{ik})^2 \right] - d_{jk}^2 \right\}^2 \qquad \textbf{(III.2)}$$

Où P est le nombre d'échantillons d'apprentissage, **C** est le nombre de règles (cluster), d_{ik} est la distance (la mesure de la dissimilarité) entre deux échantillons j et k, et u_{ij} est l'appartenance d'un point j au cluster i. Comme il a été démontré par Ruspini, Jd est une mesure de la qualité d'un cluster basée sur la densité locale, il sera petit quand les termes de l'équation III.2 seront singulièrement petits ; et ceci sera possible lorsque des paires de points proches auront approximativement le même degré d'appartenance dans les C clusters.

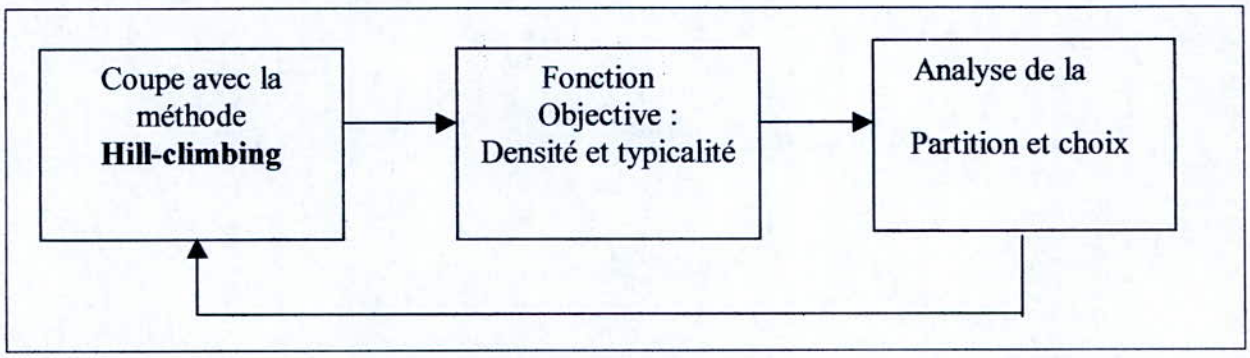
La deuxième est la mesure de la typicalité qui est une variation du fonctionnel du moindre carré proposé par Bezdek [21] :

$$Jt = \sum_{j=1}^{P} \sum_{k=1}^{C} (u_{ik}^2 \quad d_{ik}^2) \qquad \textbf{(III.3)}$$

Où d_{ik} est la distance entre un point k et le centre h_i du cluster (région) i. On appelle Jt la mesure de la typicalité parce qu'elle sera petite lorsque les points dans un cluster se resserrent bien autour de leur centre $\vec{h}_i$ (prototype).

La densité et la typicalité sont d'importantes mesures car elles sont en relation très proche avec deux importantes caractéristiques des termes linguistiques qui sont : **Le support** et **la partie centrale** respectivement. D'un coté, le support est le rang des appartenances non nulles ($u > 0$), d'un autre coté, la partie centrale est le rang des appartenances complètes ($u = 1$).

En général, l'objectif est d'aboutir à un terme linguistique à fort support (grande densité ou petit Jd) et une partie centrale représentative (bon prototype ou petit Jt). Donc il serait judicieux de choisir Jd + Jt comme fonction objective. En d'autres termes, pour chaque coupe possible, le Jd + Jt de la partition résultante est calculé, en suite est choisie la partition dont Jd + Jt est minimale, puis l'opération est répétée de nouveau. La **figureIII.2** résume l'algorithme de cette méthode.



FigureIII.2. Algorithmes de la méthode de HILL-CLIMBING

III.3. ALGORITHME D'APPRENTISSAGE DU SONFIN

Comme cité plus haut deux types d'apprentissage sont utilisés –l'apprentissage de la structure et l'apprentissage des paramètres- pour la construction d 'un SONFIN.

L 'apprentissage de la structure inclue l 'identification de la structure des antécédents et des conséquences des règles floues. L 'identification de la structure des antécédents correspond à la partition de l'espace des entrées et peut être formulée comme un problème d'optimisation combinatoire avec deux objectifs : Minimiser le nombre de règles générées, et minimiser le nombre d'ensembles flous attribués à chaque entrée. La plus importante tache à effectuer lors de L'identification des conséquences est de décider quand est il nécessaire de générer une nouvelle fonction d'appartenance (un nouvel ensemble) pour une sortie et quel sont les règles qui ont pour conséquence ce nouvel ensemble.

Le SONFIN peut être utilisé pour des opérations à n'importe quel instant pendant le processus d'apprentissage, sans échantillons d 'E/S d'entraînement, lorsque des opérations ON-LINE sont requises.

Il n'y a pas de règles initialement dans le SONFIN ; elles sont crées dynamiquement pendant que le processus d'apprentissage reçoit des données d'entraînement en suivant les étapes suivantes :

- Partition de l'espace des entrées.
- Construction des règles floues.
- Identification paramétrique.

III.3.a. PARTITION DE L'ESPACE DES ENTREES

La manière dont l'espace des entrées est partitionné détermine le nombre de règles extraites des données d'entraînement et le nombre d'ensembles flous dans un univers de discours pour chaque variable d'entrée. Pour chaque échantillon d'entrée $\boldsymbol{x}$ le degré d'activation d'une règle peut être interprété comme le degré d'appartenance de $\boldsymbol{x}$ au cluster (région) correspondant à cette règle :

Soit $F^i(\mathbf{x})$ ce degré et $\boldsymbol{x}$ le nouvel échantillon qui vient d'être réceptionné, alors :

$$F^i(\mathbf{x}) = \prod_i^n u_i^{(3)}$$

$$= \exp(-[D_i(x-m_i)]^T[(D_i(x-m_i)]) \quad . \quad \textbf{(III.4)}$$

Ou $F^i \in [0, 1]$ dans l'équation ci-dessus. Le terme $[D_i(x-m_i)]^T[(D_i(x-m_i)]$ représente en fait, la distance entre $\boldsymbol{x}$ et le centre du cluster *i*. En utilisant cette mesure, nous obtiendrons le critère suivant qui nous permet de générer de nouvelles règles :

$$J = \arg \max_{1 \le j \le c(t)} F^i(\mathbf{x}) \qquad \textbf{(III.5)}$$

C'est à dire nous cherchons l'argument de la composante max du vecteur $F^{j=1\ldots c(t)}$ où c(t) est le nombre de règle à l'instant t. Si $F^J \le \bar{F}(t)$ alors une nouvelle règle est générée, où $\bar{F}(t) \in [0, 1]$ est un seuil pré-spécifié qui peut décroître pendant le processus d'apprentissage. Une fois la nouvelle règle générée, l'étape suivante est de lui assigner les centres et les largeurs initiaux des fonctions d'appartenance lui correspondants.

Sachant que notre but est de minimiser une fonction objective, et que les centres et les largeurs sont ajustables pendant la phase de l'identification paramétrique ; de ce fait il ne serait pas judicieux d'imposer au calculateur le calcul de valeurs parfaites pour ces paramètres. Ainsi, on a tout simplement:

$$\mathbf{m}_{[c(t)+1]} = \mathbf{x}$$

Et

$$D_{[c(t)+1]} = \frac{-1}{\beta}.\mathrm{diag}\left[\frac{1}{\ln(F^J)} \quad \ldots \quad \frac{1}{\ln(F^J)}\right] \qquad \textbf{(III.6)}$$

En accord avec le principe heuristique du premier plus proche voisinage [10] où $\beta \geq 0$ décide du degré de recouvrement entre deux clusters. Après qu'une règle ait été générée, la seconde étape est de décomposer la fonction d'appartenance multidimensionnelle dans **(III.6)** en des fonctions d'appartenance unidimensionnelles, correspondantes à chaque variable d'entrée. Pour le cas des fonctions d'appartenances gaussiennes –ce choix n'est pas fortuit- la tache peut être réalisée facilement, aussi il vient:

$$\exp(-[D_i(x-m_i)]^T[(D_i(x-m_i)]) = \prod_j \exp(-\left[(x_j - m_{ij})^2 / \sigma_{ij}^2\right]) \qquad \textbf{(III.7)}$$

Où m_{ij} et σ_{ij} sont, respectivement, les projections du centre et de la largeur des fonctions d'appartenance dans chaque dimension.

Dans le but de réduire le nombre d'ensembles flous pour chaque variable d'entrée, et pour éviter l'existence de similarités entre certains d'entre eux, il faut comparer anciens et nouveaux ensembles flous dans chaque dimension et cela avant d'entamer le processus

La similarité entre deux ensembles flous –dans notre cas ensembles gaussien- est mesurée A l'aide des formules suivantes :

Soient A et B deux ensembles flous dont la mesure de la similitude est à déterminer et ayant pour les fonctions d'appartenance:

$$u_A(x) = \exp\{-(x-m_1)^2/\sigma_1^2)\} \quad \text{et} \quad u_B(x) = \exp\{-(x-m_2)^2/\sigma_2^2)\} \qquad \textbf{(III.8)}$$

On peut calculer la norme $|A \cap B|$ par :

$$|A \cap B| = \frac{1}{2}\frac{h^2\left[m_2 - m_1 + \sqrt{\pi}(\sigma_1 + \sigma_2)\right]}{\sqrt{\pi}(\sigma_1 + \sigma_2)} + \frac{1}{2}\frac{h^2\left[m_2 - m_1 + \sqrt{\pi}(\sigma_1 - \sigma_2)\right]}{\sqrt{\pi}(\sigma_2 - \sigma_1)} + \frac{1}{2}\frac{h^2\left[m_2 - m_1 - \sqrt{\pi}(\sigma_1 + \sigma_2)\right]}{\sqrt{\pi}(\sigma_1 - \sigma_2)} \qquad \textbf{(III.9)}$$

Où $h(x) = \max\{0, x\}$ Donc la mesure de la similitude est :

$$E(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{\sigma_1\sqrt{\pi} + \sigma_2\sqrt{\pi} - |A \cap B|}. \qquad \textbf{(III.10)}$$

Où nous avons utiliser le fait que : $|A| + |B| = |A \cap B| + |A \cup B|$. **(III.11)**

Posons u (m_i, σ_i) comme étant la fonction d'appartenance gaussienne dont le centre est m_i et la largeur est σ_i. L'algorithme qui génère les nouvelles règles floues ainsi que les ensembles flous pour chaque variable d'entrée peut être formulé comme suit :

Si x est le nouvel échantillon Alors faire :

Partie 1 {génération de la première règle avec :

le centre $\mathbf{m_1 = x}$

la largeur $D_1 = \text{diag}\left[\frac{1}{\sigma_{init}} \quad .... \quad \frac{1}{\sigma_{init}}\right]$

Où σ_{init} est une constante prés- spécifié.

Après la décomposition, nous aurons n fonctions d'appartenance unidimensionnelles avec :

$$m_{1i} = x_i \text{ et } \sigma_{1i} = \sigma_{init} \qquad i = 1 \ldots n$$

}

Sinon pour chaque nouvel échantillon **x**, faire :

Partie 2 {trouver $J = \arg \max_{1 \le j \le c(t)} F^i(\mathbf{x})$

Si $F^J \geq \bar{F}_{in}(t)$ alors ne rien faire

Sinon Génération d 'une nouvelle règle, d'où $c(t+1) = c(t) + 1$ avec :

$$\mathbf{m}_{[c(t)+1]} = \mathbf{x} \qquad \text{et} \qquad D_{[c(t)+1]} = \frac{-1}{\beta}.\text{diag}\left[\frac{1}{\ln(F^J)} \quad .... \quad \frac{1}{\ln(F^J)}\right]$$

Après la décomposition nous aurons :

$$m_{new\text{-}i} = x_i, \quad \sigma_{new-i} = -\beta.\ln(F^J) \qquad i = 1 \ldots n$$

Faire la mesure suivante pour chaque variable entrée i :

$$\{\text{degree}(i, t) = \max_{1 \le j \le k_i} E[u(m_{new\text{-}i}, \sigma_{new-i}), u(m_{ji}, \sigma_{ji})]$$

Où k_i est le nombre de partition de la $i^{ème}$ variable d'entrée.

Si degré$(i, t) \leq \rho(t)$ Alors : adopter cette nouvelle fonction d 'appartenance et poser $k_i = k_i + 1$

Sinon poser la projection de la nouvelle fonction d 'appartenance comme étant celle qui est la plus proche

}

FIN
}

Dans l'algorithme ci-dessus le seuil $\bar{F}_{in}$ détermine le nombre de règles générées. Pour une grande valeur de $\bar{F}_{in}$, un plus grand nombre de règles sera généré, ce qui augmentera la précision. $\rho(t)$ est un critère scalaire de similarité, à décroissance monotonique de tel sorte qu'une grande similitude entre deux ensemble flous est permise pendant la phase initiale de l'apprentissage.

Pour la partition de l'espace des sorties, la même mesure est utilisée. Puisque le critère de la génération d'un nouveau cluster de sortie est relié à la construction d'une règle floue, nous le décrirons avec plus de détails dans ce qui suit.

III.3.b. CONSTRUCTION DES REGLES FLOUES

Comme mentionné ci-dessus, la génération d'un nouveau cluster correspond à la génération d'une nouvelle règle floue, avec sa partie «antécédents » construite pendant la phase de partition de l'espace des entrées. Au même instant il est impératif de décider quelles seront les conséquences d'une règle.

Supposons qu'un nouveau cluster est formé après la présentation de la nouvelle paire d'échantillon E/S (x, d), alors la partie conséquence est construite à l'aide de l'algorithme suivant :

Si il n y a pas de cluster de sortie
Faire {Partie 1 de l 'algorithme précédent en remplaçant **x** par **d** }
Sinon

Faire { $J = \arg \max_{1 \le j \le cy(t)} F^j(\mathbf{d})$

Si $F^J \geq \bar{F}_{out}(t)$
Alors connecter le cluster d'entrée c(t+1) au cluster de sortie J existant

Sinon
Génération d'un nouveau cluster de sortie
connectée le cluster d 'entrée c(t+1)au nouveau cluster de sortie générée
}

Cet algorithme se base sur le fait que plusieurs antécédents peuvent avoir la même conséquence, donc plusieurs clusters d'entrées peuvent être connectés au même cluster de sortie.

III.4. IDENTIFICATION PARAMETRIQUE

Après avoir ajuster la structure du réseau en s'accordant à la paire courante des échantillons d'E/S, le réseau entre alors dans la phase d'identification paramétrique afin d'ajuster les paramètres de la structure tirée du même échantillon d'E/S.

Il est à noter que les paramètres d'apprentissages calculés plus loin, seront préformant dans l'ensemble du réseau (après l'incrémentation de l'algorithme), et cela quel que soit l'état des nœuds et des liaisons (nouveaux ou déjà existant).

L'idée d'un algorithme basé sur la rétro-propagation pour un apprentissage supervisé sera utilise dans ce qui suivra. Considérons le cas monovariable pour (plus de clarté). Notre but sera de minimiser la fonction d'erreur :

$$E = \frac{1}{2}[y(t) - y^d(t)]^2 \qquad \textbf{(III.12)}$$

Ou y^d est la sortie désirée et y(t) est la sortie réelle.

Pour chaque ensembles de données d'apprentissage, une phase de propagation directe est utilisée pour calculer les degrés d 'activation de tous les nœuds de la structure, ce qui nous permettra d'obtenir la sortie courante **y(t).** Ensuite va commencer à partir des nœuds de sortie une phase de rétro-propagation dans le but est de calculer **∂E/∂w** pour tout les nœuds des couches internes ; tout en supposant que **w** représente les paramètres ajustables ($\mathbf{a}_{ij}$,$\mathbf{m}_{ij}$ σij)dans chaque nœud (cela est valable pour le SONFIN). L'expression générale de la mise à jour des règles est écrite comme suit :

$$\Delta w \alpha - \partial E/\partial w \qquad \textbf{(III.13)}$$

$$W(t+1)=w(t)+\eta(-\partial E/\partial w) \qquad \textbf{(III.14)}$$

$$\partial E/\partial w = \frac{\partial E}{\partial(\text{fonction d'activation})} \times \frac{\partial(\text{fonction d'activation})}{\partial w}$$

$$=(\partial E/\partial a)\times(\partial a/\partial w). \qquad \textbf{(III.15)}$$

Où η est le gain d'apprentissage, et a est la fonction d'activation.

Nous allons établir le calcul du terme $\partial E/\partial w$ étape par étape (couche par couche); on commence le calcul à partir de la couche de sortie

Couche 6

Il n'y a pas de paramètres à ajuster dans cette étape ; seulement l'erreur de signal $\delta_i^{(6)}$ doit être calculer en dérivant $\partial E/\partial w$ ensuite propagé

$$\delta_i^{(6)} = -\frac{\partial E}{\partial a^{(6)}} = y^d(t) - y(t) \qquad \textbf{(III.16)}$$

Couche 5 :

En utilisant les équations (6),(7), et (24) la règle de mise à jour des a_{ji} est donnée par :

$$-\frac{\partial E}{\partial a_{ji}} = -\frac{\partial E}{\partial a^{(6)}} \cdot \frac{\partial a^{(6)}}{\partial a^{(5)}} \frac{\partial a^{(5)}}{\partial a_{ji}} \qquad \textbf{(III.17)}$$

Et :

$$\frac{\partial a^{(6)}}{\partial a^{(5)}} = 1 \qquad \textbf{(III.18)}$$

$$\frac{\partial a^{(5)}}{\partial a_{ji}} = x_j \sum_i u_i^{(5)} \qquad \textbf{(III.19)}$$

ou la sommation est sur toutes les sorties des nœuds de la couche 4 ayant un lien avec le $i^{ème}$ nœud de la couche 5.ainsi le paramètre a_{ji} est ajusté par :

$$a_{ji}(t+1) = a_{ji} + \eta[y^d(t) - y(t)]x_j \sum_i u_i^{(5)} \qquad \textbf{(III.20)}$$

Où x_0=1

Couche 4

Comme dans la couche 6, seule le signal d'erreur doit être calculé dans cette couche. A partir des équation (5), et (24), le signal d'erreur pourra être dérivé de

$$\delta_i^{(4)} = -\frac{\partial E}{\partial a^{(4)}} = -\frac{\partial E}{\partial a^{(5)}} \frac{\partial a^{(5)}}{\partial a^{(4)}} \qquad \textbf{(III.21)}$$

Où :

$$\frac{\partial a^{(5)}}{\partial a^{(4)}} = \sum_j a_{ji} x_{ji} \qquad \textbf{(III.22)}$$

Si nous étions dans le cas où il y aurait plusieurs sorties, alors le signal d'erreur deviendrait, $\delta_i^{(4)} = \sum_k \delta_k^{(4)}$ ou la sommation est appliquée sur toutes les conséquences d'une règle ; en d'autres mots, l'erreur d'un nœud règle n'est au fait que la somme des erreurs de ses conséquences.

Couche 3

Comme dans la couche 4, seule le signal d'erreur doit être calculé dans cette couche

$$\delta_i^{(3)} = -\frac{\partial E}{\partial a^{(3)}} = -\sum_j \frac{\partial E}{\partial a_j^{(4)}} \frac{\partial a_j^{(4)}}{\partial a_i^{(3)}} \qquad \textbf{(III.23)}$$

Où

$$\frac{\partial a_j^{(4)}}{\partial a_i^{(3)}} = \begin{cases} \dfrac{\sum_i a_i^{(3)} - a_j^{(3)}}{\left[\sum_{i=1}^{c} a_i^{(3)}\right]}, & si \quad \mathbf{j = i} \\ \dfrac{-a_j^{(3)}}{\left[\sum_{i=1}^{c} a_i^{(3)}\right]^2}, & \mathbf{si\ j \neq i} \end{cases} \qquad \textbf{(III.24)}$$

Ce qui nous fera aboutir à :

$$\delta_i^{(3)} = \sum_j \delta_j^{(4)} \frac{\partial a_j^{(4)}}{\partial a_i^{(3)}} \qquad \textbf{(III.25)}$$

Couche 2

En utilisant les expressions (3)et (37), la règle fraîchement générée de $\mathbf{m}_{ij}^{(2)}$ est dérivée de la manière suivante :

$$-\frac{\partial E}{\partial m_{ij}^{(2)}} = \frac{\partial E}{\partial a^{(3)}} \sum_k \frac{\partial a^{(3)}}{\partial a_k^{(2)}} \frac{\partial a_k^{(2)}}{\partial m_{ij}^{(2)}} \qquad \textbf{(III.26)}$$

Où

$$\frac{\partial a^{(3)}}{\partial a_k^{(2)}} = \frac{a^{(3)}}{a_k^{(2)}} \qquad \textbf{(III.27)}$$

$$\frac{\partial a_k^{(2)}}{\partial m_{ij}^{(2)}} = \begin{cases} a_k^{(2)} \dfrac{2(x_i - m_{ij})}{\sigma_{ij}^2} & , \quad \textit{si le terme du noeudj est comecté au noeud règlek} \\ 0 & , \quad \textit{ailleurs} \end{cases} \qquad \textbf{(III.28)}$$

Donc la règle mise à jour de $\mathbf{m}_{ij}^{(2)}$est donnée par :

$$m_{ij}^2(t+1) = m_{ij}^2(t) - \eta \frac{\partial E}{\partial m_{ij}^2} \qquad \textbf{(III.29)}$$

de la même manière (en utilisant les équations (3)et (37)), la mise à jour de la règle de $\boldsymbol{\sigma}_{ij}^{(2)}$est dérivée de :

$$-\frac{\partial E}{\partial \sigma_{ij}^{(2)}} = \frac{\partial E}{\partial a^{(3)}} \sum_k \frac{\partial a^{(3)}}{\partial a_k^{(2)}} \frac{\partial a_k^{(2)}}{\partial \sigma_{ij}^{(2)}} \qquad \textbf{(III.30)}$$

Ou

$$\frac{\partial a_k^{(2)}}{\partial \sigma_{ij}^{(2)}} = \begin{cases} a_k^{(2)} \dfrac{2(x_i - m_{ij})^2}{\sigma_{ij}^3} & \textit{si le terme du noeudj est comecté au noeud règlek} \\ 0 & \textit{ailleurs} \end{cases} \qquad \textbf{(III.31)}$$

Finalement la mise a jour de la règle de $\boldsymbol{\sigma}_{ij}^{(2)}$ est

$$\sigma_{ij}^2(t+1) = \sigma_{ij}^2(t) - \eta \frac{\partial E}{\partial \sigma_{ij}^2} \qquad \textbf{(III.32)}$$

Remarque

En plus de la méthode du gradient, et dans le but d'accroître l'efficacité de l'apprentissage, on peut utiliser l'algorithme des moindres carrés récursifs pour l'identification des paramètres linéaires des conséquences dans la couche 5. Cet algorithme est donné par les relations suivantes :

$$P(t+1)=\frac{1}{\lambda}\left[p(t)-\frac{p(t)u(t+1)u(t+1)^T p(t)}{\lambda+u(t+1)^T p(t)u(t+1)}\right]$$

$a(t+1)=a(t)+p(t+1)u(t+1)[y^d(t)-y(t)]$,

où :
p(t) est le Gain d'adaptation .
a(t) est le vecteur des paramètre à adapter formé par les coefficients linéaires (a_{ij} et m_j) de la couche 5 .
u(t) est le vecteur des observations formé par les entrée x_i du SONFIN et les sorties fn_i de la couche 4 .

λ ($0<\lambda\leq 1$) est le facteur d'oubli , $p(0)=\sigma I$ où σ est un nombre positif (valeur typique = 1000) .

Puisqu'on a :

$$y(t+1) = fn_1*(a_{11}\ x_1+ a_{12}\ x_2 +\ldots a_{1n}\ x_n) + \ldots.+fn_c\ *(a_{c1}\ x_1+ \ldots.a_{cn}\ x_n)+ fn_1m_1+\ldots.+fn_{cout}m_{cout}$$

où : ***c*** est le nombre de cluster dans l'espace des entrées et ***cout*** est le nombre de cluster dans l'espace des sorties .

Alors on déduit que :

$$u(t)=[\ fn_1*x\ \ fn_2*x\ \ldots.fn_c*x\ \ fn_1\ \ fn_2\ \ldots\ldots fn_{cout}]$$

$$a(t)=[a_1\ a_2\ \ldots\ldots a_c\ m_1\ m_2\ \ldots m_{cout}\ ]$$

ou f_i représente la i éme sortie de la quatriéme couche , et m_i le centre du i éme cluster de sortie.

Chapitre IV

Applications

IV.1 INTRODUCTION

En pratique, les systèmes sont généralement non linéaires, il est donc difficile de les commander en utilisant les commandes classiques. D'où la nécessité de trouver d'autres manières de modéliser et commander ce type de systèmes.

IV.2 COMMANDE D'UN PENDULE INVERSE PAR MODELE INVERSE

Dans ce chapitre, nous allons nous intéresser à la commande d'un système mécanique instable et non linéaire connu sous le nom du pendule inversé, et ceci en utilisant deux structures de réseaux neuro-flous à savoir : l'ANFIS (implémenté par MATLB) et le SONFIN (réalisé a l'aide d'un programme que nous avons écrit sous MATLAB) décrits aux chapitres précédents.

IV.2.1 MODELISATION DU PENDULE INVERSE

Comme illustré sur la **figure IV.1**, le problème de la commande du pendule inversé est de trouver un régulateur permettant d'asservir la position du chariot tout en stabilisant le pendule dans la position verticale (point d'équilibre instable). Sachant que les deux mouvements du chariot et du pendule se font dans le plan, le modèle du système est comme suit :

$$\dot{X} = A(X) + B(X)U$$

Où X est le vecteur d'état, et est donné par

$$X = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^T .$$

Sachant que :
x : représente la position du chariot
θ : représente l'angle que fait le pendule inversé avec la verticale.
U : représente la commande qui est la force horizontale appliquée sur le chariot.
A(X) et B(X) sont des distributions vectorielles définissant le système par rapport à l'état et à la commande respectivement. Elles sont données par les relations suivantes :

$$A(X) = \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \dfrac{ml\dot{\theta}\sin\theta - mg\cos\theta\sin\theta}{M + m\sin^2\theta} \\ \dfrac{-ml\dot{\theta}^2\sin\theta\cos\theta + (M+m)g\sin\theta}{l(M + m\sin^2\theta)} \end{bmatrix} \quad \text{et} \quad B(X) = \frac{1}{l(M + m\sin^2\theta)} \begin{bmatrix} 0 \\ 0 \\ l \\ -\cos\theta \end{bmatrix}$$

Où :
M : représente la masse du chariot, égale à 0.455 Kg.
m : représente La masse du pendule, égale à 0.21 Kg.
l : représente la longueur du pendule, égale à 0.61/2 m.
g : représente l'accélération, égale à 9.8 m/s^2 .

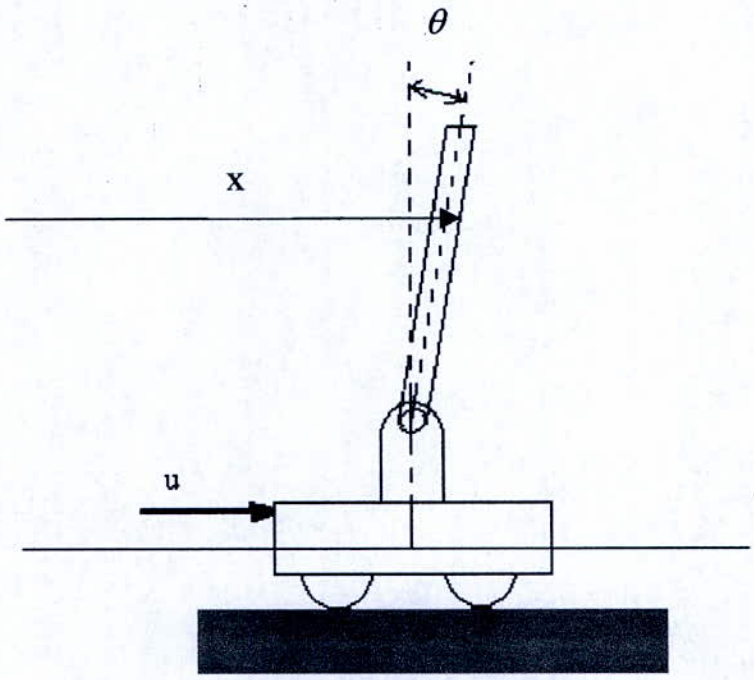


figure IV.1 : le pendule inversé .

IV.2.2 Stratégie de commande

La structure du régulateur utilisé pour la commande de notre pendule inversé s'inspire de la technique de commande par modèle inverse.

Dans la commande par modèle inverse, si l'on dispose d'un système dont l'entrée est U et la sortie y tel que $y = f(U, \dot{U}, ..., \dot{y}, \ddot{y}, ...)$ et si l'on veut réaliser la poursuite d'une référence, on peut placer un régulateur en amont du système et qui aura pour entrée le signale de référence U_r et sa sortie y_r sera connecté à l'entrée du système tel que la relation suivante soit vérifiée :

$$y_r = f^{-1}(U_r, \dot{U}_r, ..., \dot{y}_r, \ddot{y}_r, ...)$$

Le système global constitué par le régulateur et le système sera alors donné par :

$$y = fof^{-1}(U_r) = U_r$$

Le schéma suivant illustre ce principe :

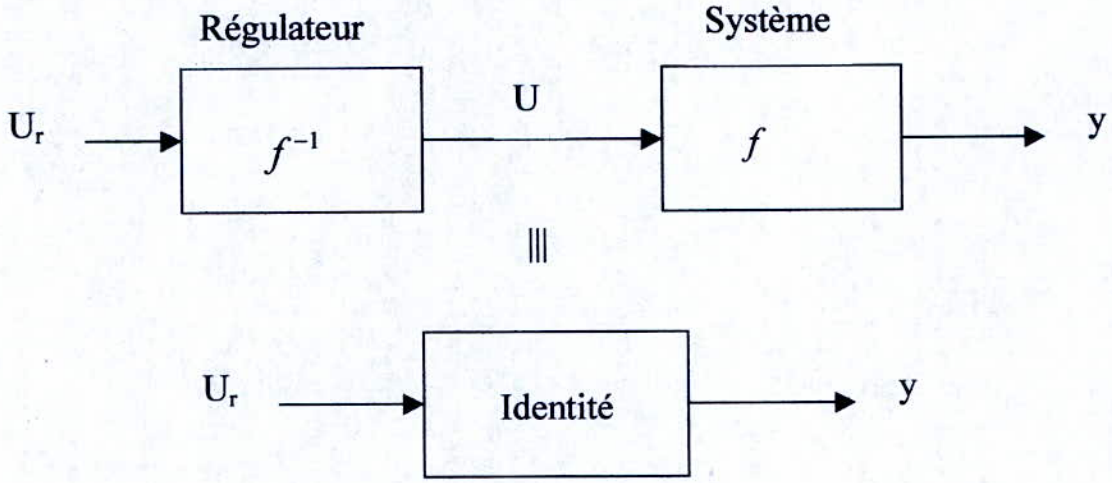


Figure IV.2 : Principe de la commande par modèle inverse

En ce qui concerne le pendule inversé dont le schéma bloc est représenté sur la **Figure IV.3**

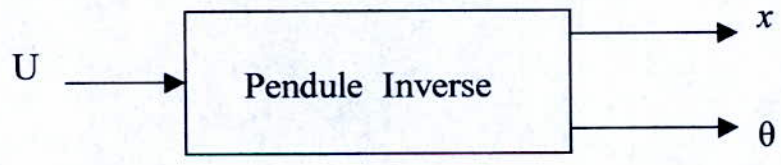


Figure IV.3 : Schéma bloc d'une pendule inverse

Si on essayait d'identifier le modèle inverse dont les entrées sont *(x, θ)*, on aura une erreur d'identification trop importante et on limitera la plage de variation de la variable *x*. Cela est dû au fait qu'il n'existe pas un transfert direct entre *x* et la force *U*.

En visualisant bien les équations d'état du pendule inversé, on constate que la force U agit directement sur les accélérations $\ddot{x}$ *et* $\ddot{\theta}$, ce qui peut être intuitivement ressentit du fait que physiquement la force et l'accélération sont dues au même phénomène. Ceci nous amène à proposer d'identifier le modèle inverse dont les entrées sont $(\ddot{x}, \ddot{\theta}, \theta)$ et la sortie est U. Le schéma d'identification par réseau Neuro-flou est illustré dans la **Figure IV.4**.

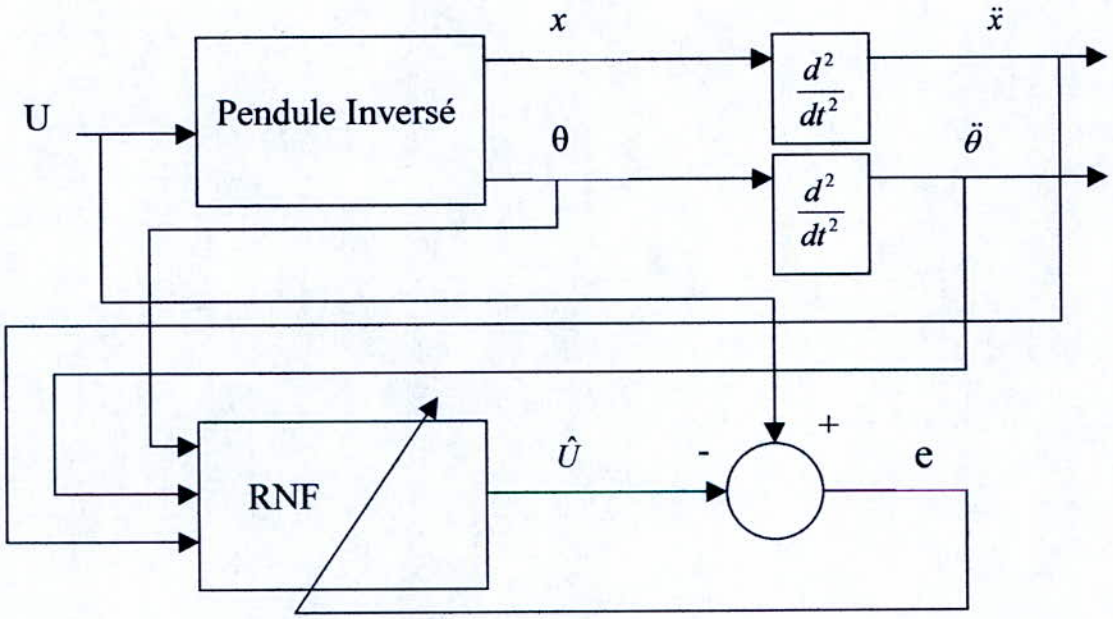


Figure IV.4 : Principe de l'identification par réseau neuro-flou

Une fois l'identification du modèle inverse terminée, on le place en série avec le pendule inversé tout en faisant un retour de la sortie θ comme illustré sur la **Figure IV.5.**

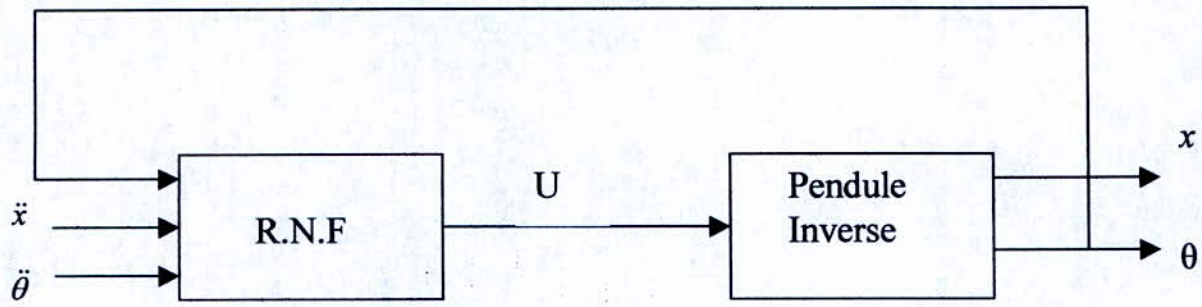


Figure IV.5 : Utilisation du RNF pour la commande .

Ainsi, le système global (pendule inversé et réseau neuro-flou) peut être considéré comme un ensemble de deux sous systèmes découplés dont la fonction de transfert de chacun d'eux est $\frac{1}{S^2}$. La **figure IV.6** illustre ceci :

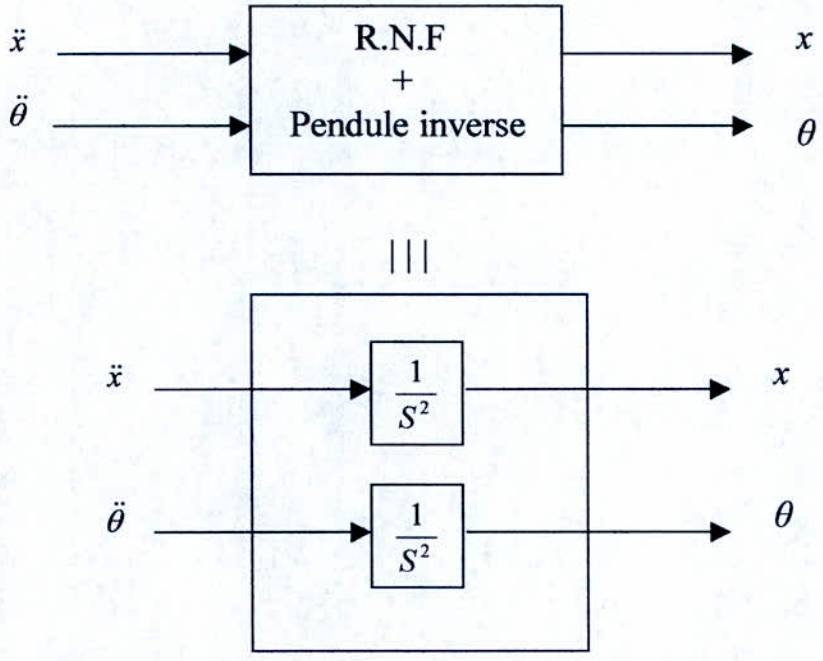


Figure IV.6 : Système R.N.F + pendule inverse

Il suffit maintenant d'utiliser un réglage classique pour chaque sous système à savoir : un placement de pôles par retour d'état pour la stabilisation et une action intégral « I » pour annuler l'erreur statique de chacune des variables x et θ. Le schéma bloc de la commande est illustré dans la **figure IV.7.**

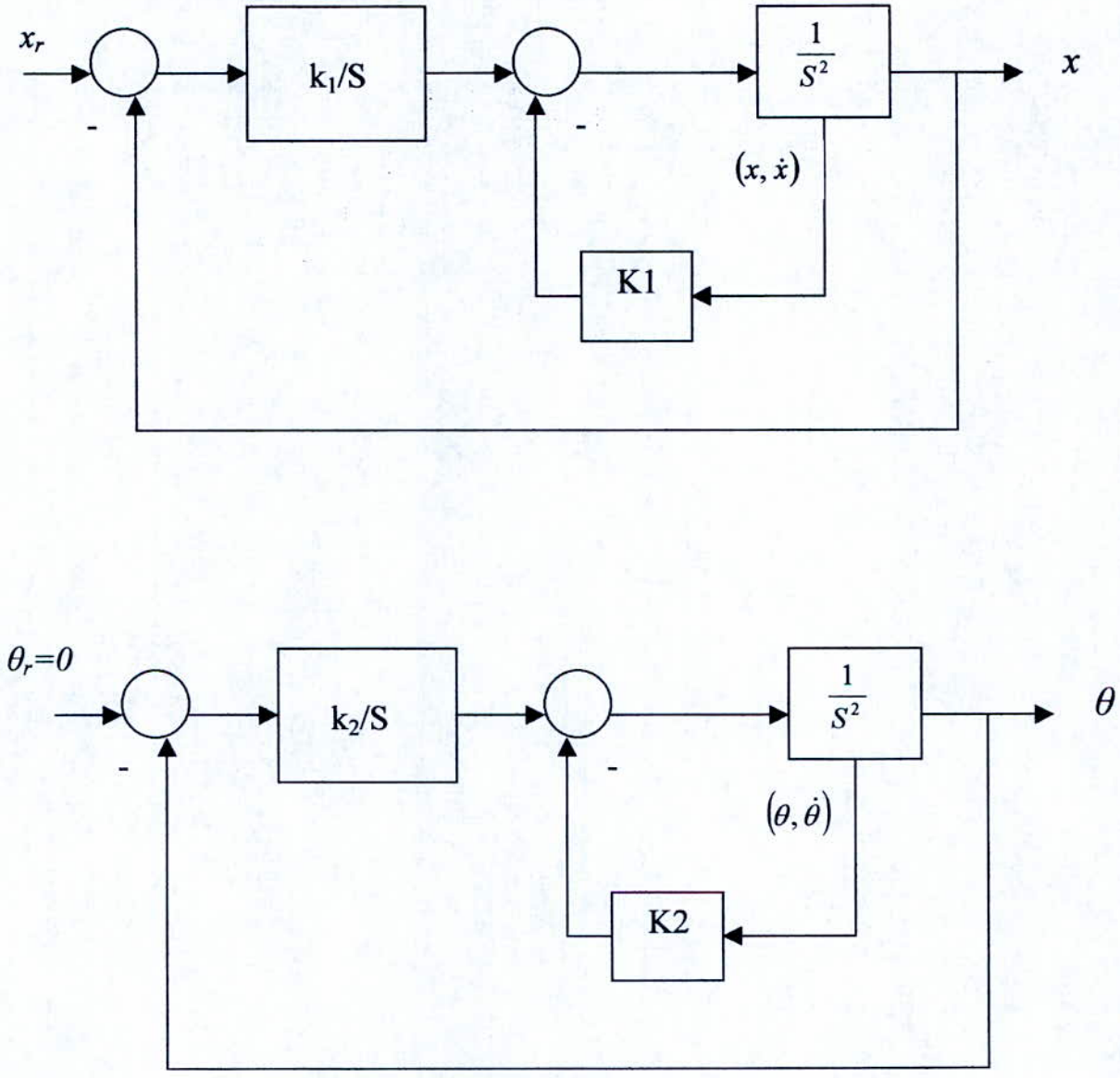


Figure IV.7 : Schéma bloc de la commande du système R.N.F + pendule inverse à l'aide d'un retour d'état + une action Intégrale

IV.2.3. SIMULATIONS ET INTERPRETATIONS

Dans ce qui suivra nous allons établir une comparaison entre les RNF ANFIS (implémenté sous MATLAB) et le SONFIN que nous avons implémenté sous forme de programme en utilisant le compilateur de MATLAB 5.3.

IV.2.3.1. ACQUISITION DE DONNEES

Afin d'effectuer l'apprentissage des deux RNF, que nous utiliserons pour la commande à savoir l'ANFIS et le SONFIN, nous aurons besoin de simuler la dynamique du pendule inversé dans le but de récupérer des échantillons d'E/S qui servirons par la suite a cet apprentissage. Les simulations ont été effectuées sous l'environnement MATLAB 5.3/ SIMULINK qui s'est avéré assez complet et conviviale.

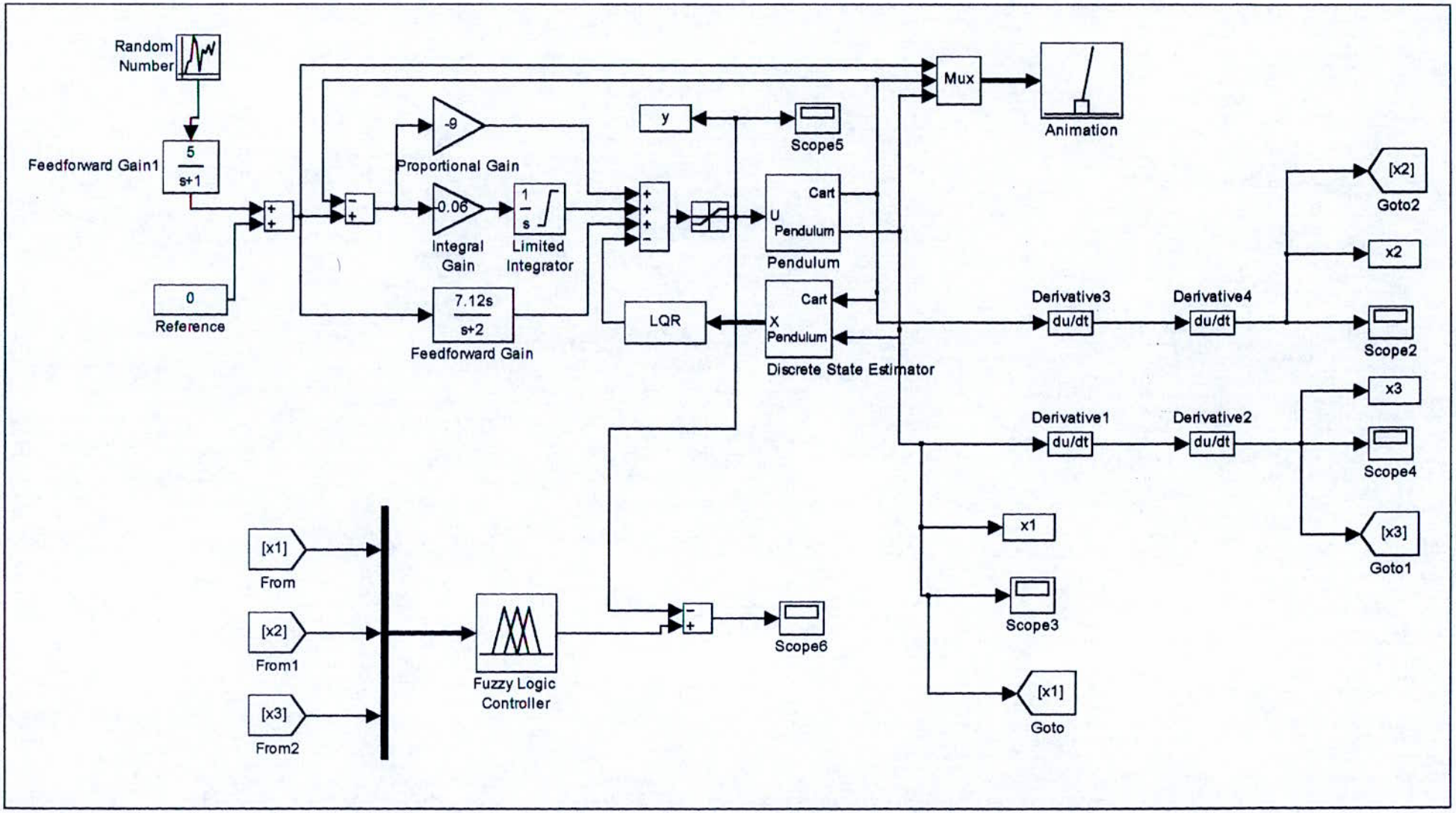
Vu que le pendule inversé est un système instable, nous avons besoin d'un régulateur qui le stabilisera. Comme exemple nous avons pris le régulateur implémenté sous MATLAB dans la section demo/simulink/complex systems; il est du type LQR associé à un PID. Le schéma bloc du pendule inversé et du stabilisateur est représenté dans la **figureIV.8**. Le RNF en bas de la figure nous servira à visualiser l'erreur une fois l'apprentissage accompli.

Nous lançons une simulation de 0 à 400secondes, et nous récupérons les valeurs des signaux de $\theta, \ddot{\theta}, \ddot{X}, et\ \mathbf{U}$ toutes les 0.2 secondes. A la fin de la simulation nous obtiendrons une matrice d'échantillons de 2000×4 qui, par la suite, sera introduite dans les deux RNF étudiés pour qu'ils puissent commencer leur apprentissage.

a. Utilisation de l'ANFIS de MATLAB 5.3 :

La commande «anfisedit » de MATLAB ouvre une boite de dialogue qui nous permettra d'initialiser la structure de l'ANFIS ; Pour cela on fait rentrer la matrice d'échantillons, puis on choisi une partition en grille, pour l'espace des entrées, avec 3 MFs du type gaussien pour chaque entrée et une défuzzification linéaire pour les sorties (on aura un SIF de type Sugeno); Donc, il sera générées $3^3 = 27$ règles. Après avoir initialiser la structure, on lance l'apprentissage.

Après avoir effectuer l'apprentissage, et pour vérifier l'efficacité de l'identification du modèle inverse par l'ANFIS, on injecte des échantillons de sorties du pendule $(\theta, \ddot{x}, \ddot{\theta})$ dans l'ANFIS, et nous comparons les sorties de ce dernier, $\hat{U}$, avec l'entrée U du pendule (la force appliquée sur le chariot); Le résultat (l'erreur entre le signal de commande du pendule inversé et la sortie du RNF) est illustré dans la **figureIV.10**. Nous remarquons que l'erreur de sortie $(\mathbf{u} - \hat{\mathbf{u}})$ est assez faible (de l'ordre de 5%) par rapport au signal de commande illustré dans la **figureIV.9**, Cependant nous constatons l'existence de petits pics.



figureIV.8 Schéma bloc utilisé pour récupérer les données E/S d'apprentissage

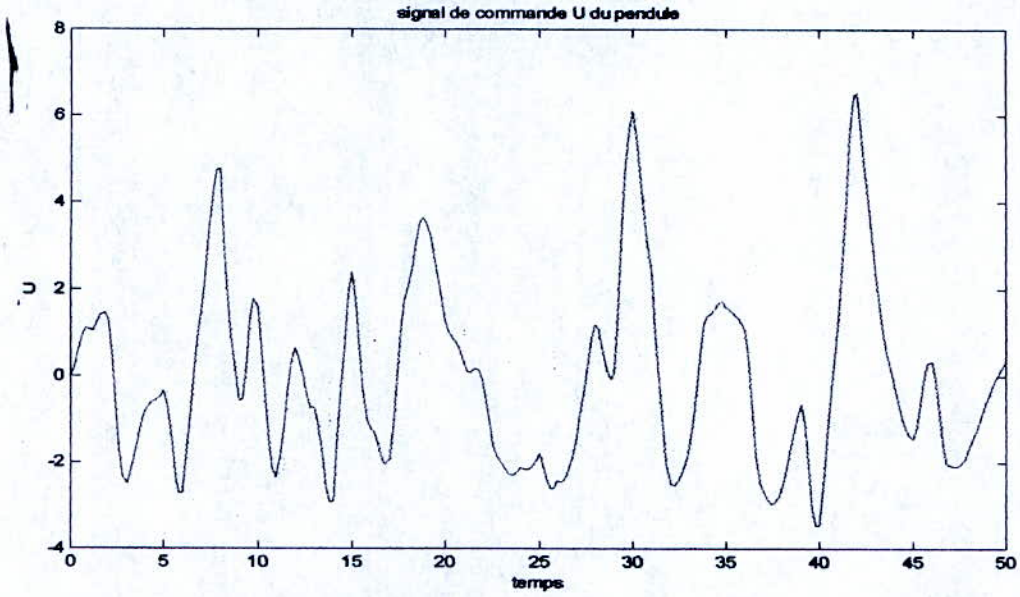
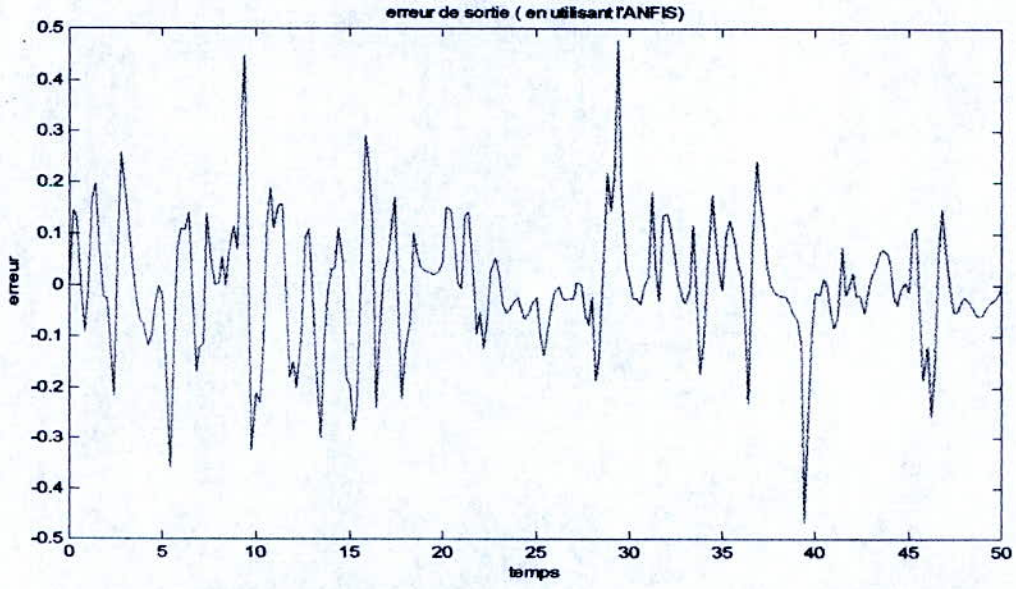


Figure IV.9 Signal de commande U du Pendule.



FigureIV.10 erreur entre la sortie Û de l'ANFIS et l'entrée U du pendule.

Les MFs de chaque entrée de l'ANFIS sont représenté dans les **figures.IV.11 ; IV.12 ; IV.13**

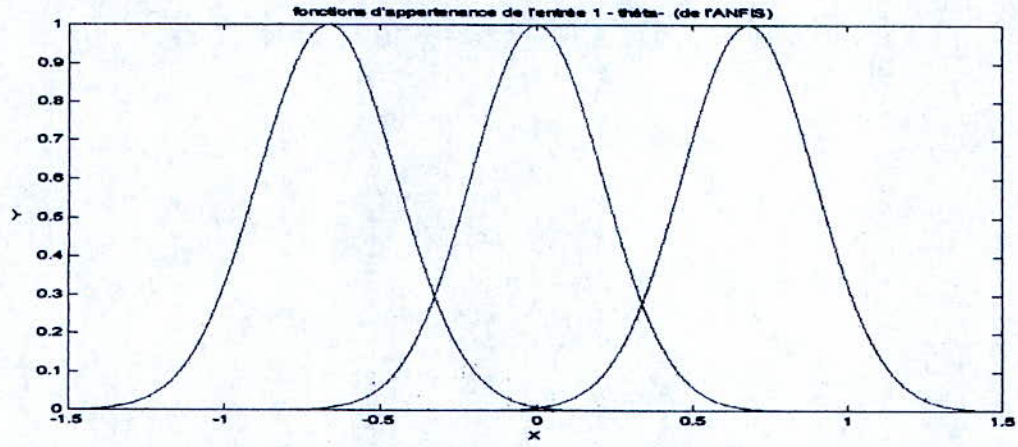


Figure.IV.11. MFs pour l'entrée θ (en radian)

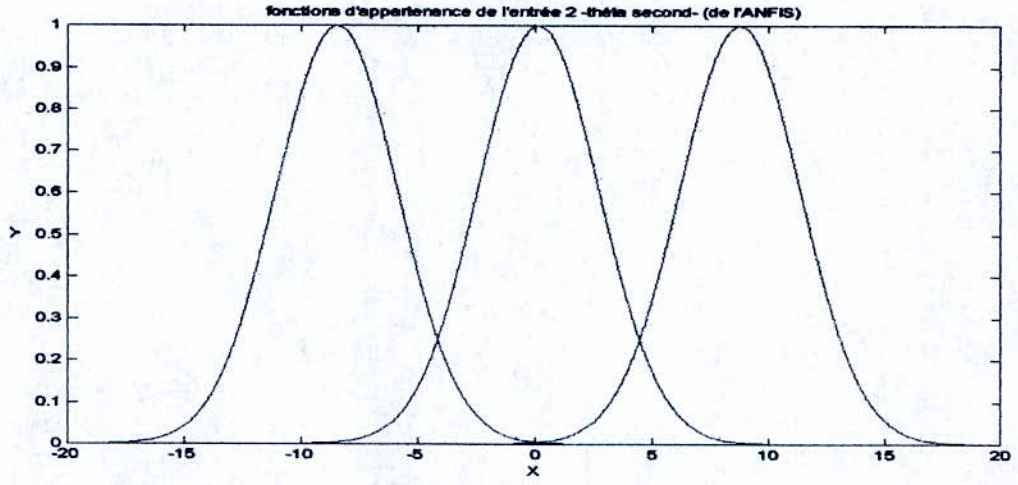


Figure.IV.12. MFs pour l'entrée θ seconde (en rad/s^2).

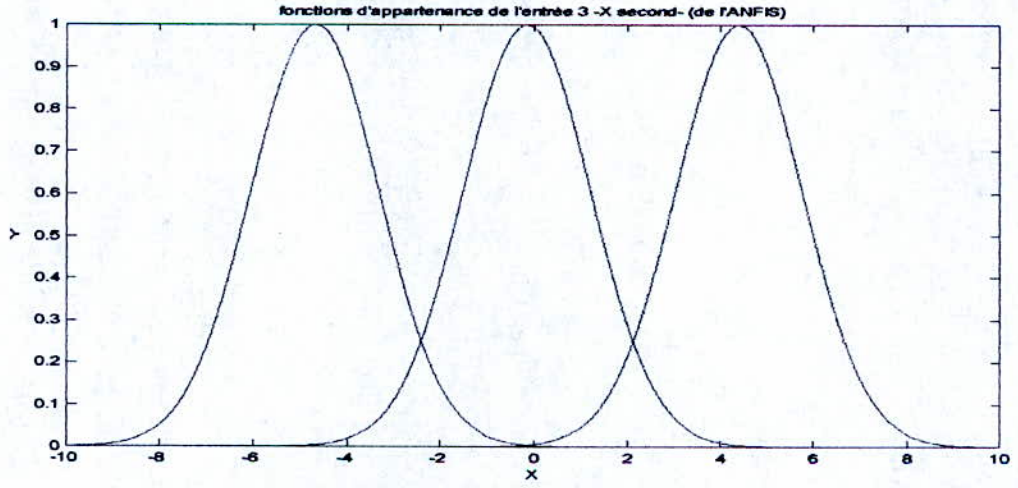


Figure.IV.13. MFs pour l'entrée x seconde (en m/ s^2).

b. L'utilisation du SONFIN

Comme vu dans le chapitre III, le SONFIN est un RNF auto-construit, nous n'avons pas donc besoin de créer une structure initiale sur laquelle sera appliquée une adaptation paramétrique (comme c'est le cas pour l'ANFIS). En effet, il suffit seulement de donner les valeurs initiales des paramètres ainsi que les seuils dont le programme a besoin (voir l'algorithme du SONFIN dans le chap.3). Pour cette application on a utilisé les valeurs suivantes :

$\sigma_{init} = [0.1,2,3]$ *pour les clusters d'entrée*

$\sigma_{init} = [1,5]$ *pour les clusters de sortie*

$\overline{F_{in}}(t) = 0.7$

$\varphi(t) = 0.85$

$\overline{F}_{out}(t) = 0.7$

Comme précédemment cité nous avons utilisé la méthode des moindres carrés récursifs pour l'identification des paramètres des conséquences et la méthode de la décente du gradient pour les paramètres des antécédents qui sont non linéaires.

La **figure.IV.14** illustre les résultats obtenus à la fin de l'apprentissage. On voit bien que l'erreur est assez faible (de l'ordre de 5% par rapport au signal U de la **Figure IV.9**) et qu'elle est assez similaire à celle obtenue en utilisant l'ANFIS.

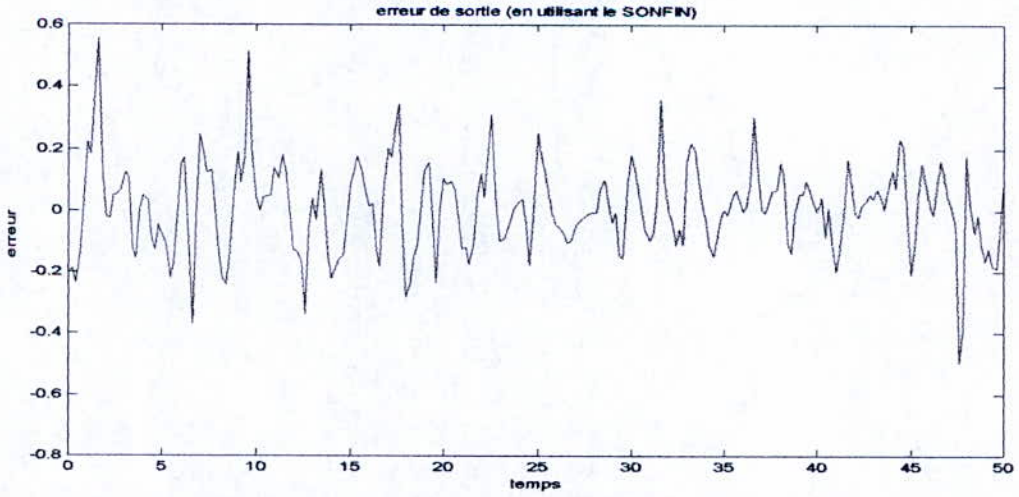


Figure.IV.14 erreur entre la sortie du SONFIN Û et l'entrée du pendule U.

Le nombre de règles générées est égal à 15, ce qui nous permet de dire qu'avec l'algorithme d'apprentissage utilisé par le SONFIN nous pouvons réduire d'une manière très satisfaisante le nombre de règles. C'est un excellent argument lorsqu'il s'agit d'appliquer cet algorithme pour la commande en temps réel.

Les MFs des Entrées et des Sorties sont représentées dans la **figureIV.15** et la **figureIV.16**. On remarque que l'utilisation du calcul de similitude entre deux ensembles flous (voir l'algorithme du SONFIN dans le chap.3) nous évite l'obtention de MFs similaires. En effet, le nombre de MFs par variables se trouve ainsi réduit. Par conséquent, le nombre de règles peut diminuer.

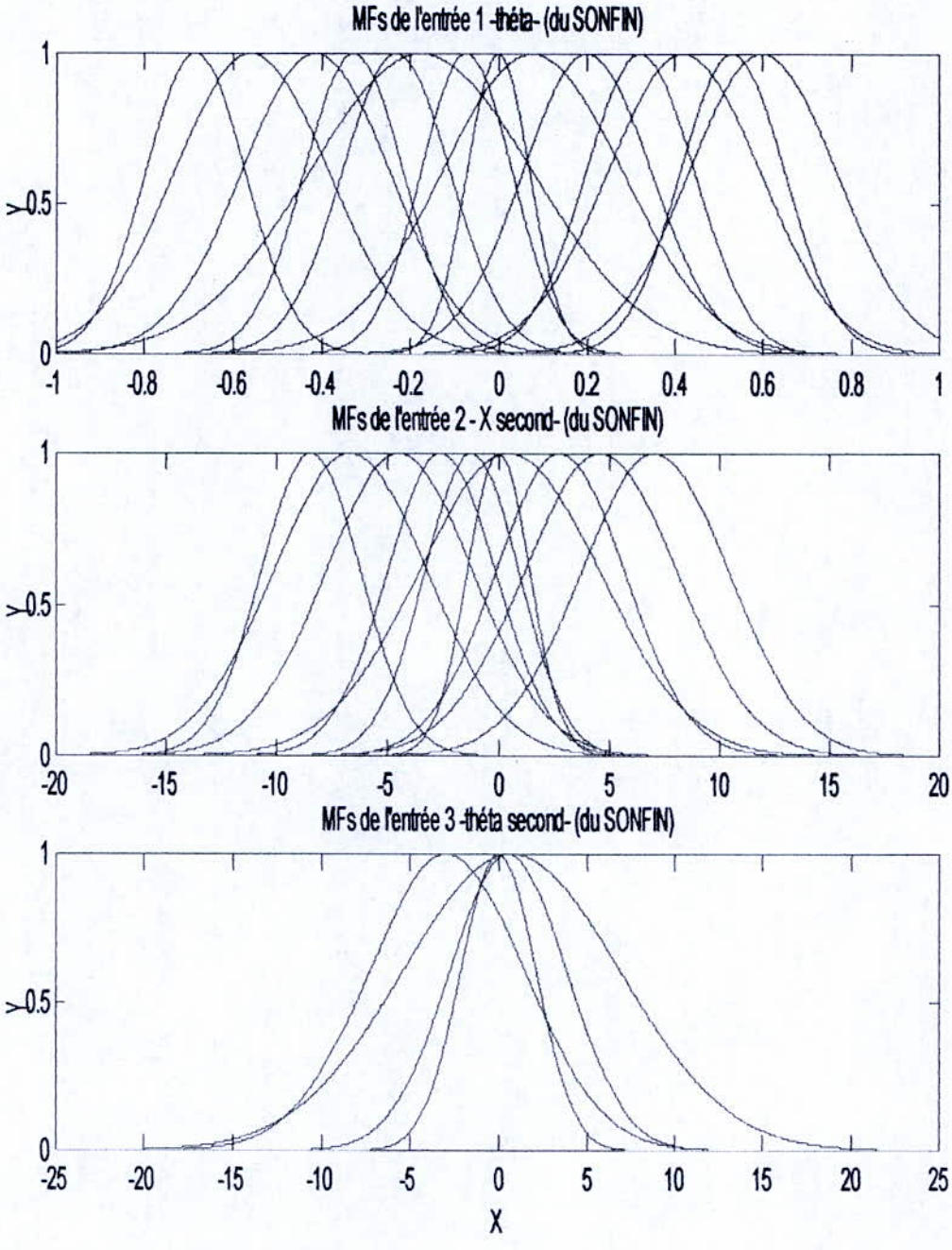


Figure.IV.15 MFs des entrées (du SONFIN)

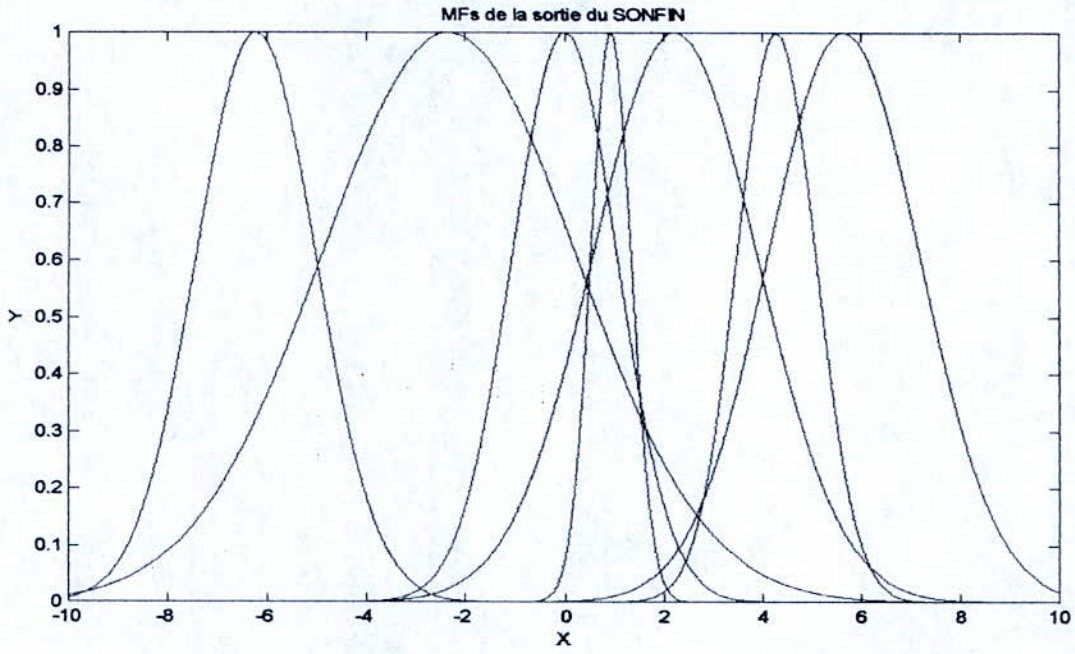


Figure.IV.16 MFs de la sortie du SONFIN

IV.2.3.2. Simulation de la commande

Une fois l'apprentissage des deux structures (ANFIS et SONFIN) achevé, nous les intégrerons l'une après l'autre dans la boucle de commande illustrée dans la **figure.IV.17** dans le but de tester l'efficacité de chacune d'elles pour la commande de la position du chariot, x, et du pendule, θ. On injectera un signal de référence carré (sur la variable x) de fréquence égale à 0.02Hz et d'amplitude égale à 5m. Ensuite on visualise les signaux de sortie x et θ, et le signal de commande. Les résultats obtenus en utilisant l'ANFIS sont illustrés dans les **figures IV.18; IV.19; IV.20** (pour x, θ, *et* u respectivement), et ceux obtenus par le SONFIN sont représentés dans les **figures.IV.21;IV.22;IV.23** (pour x, θ, *et* u respectivement).

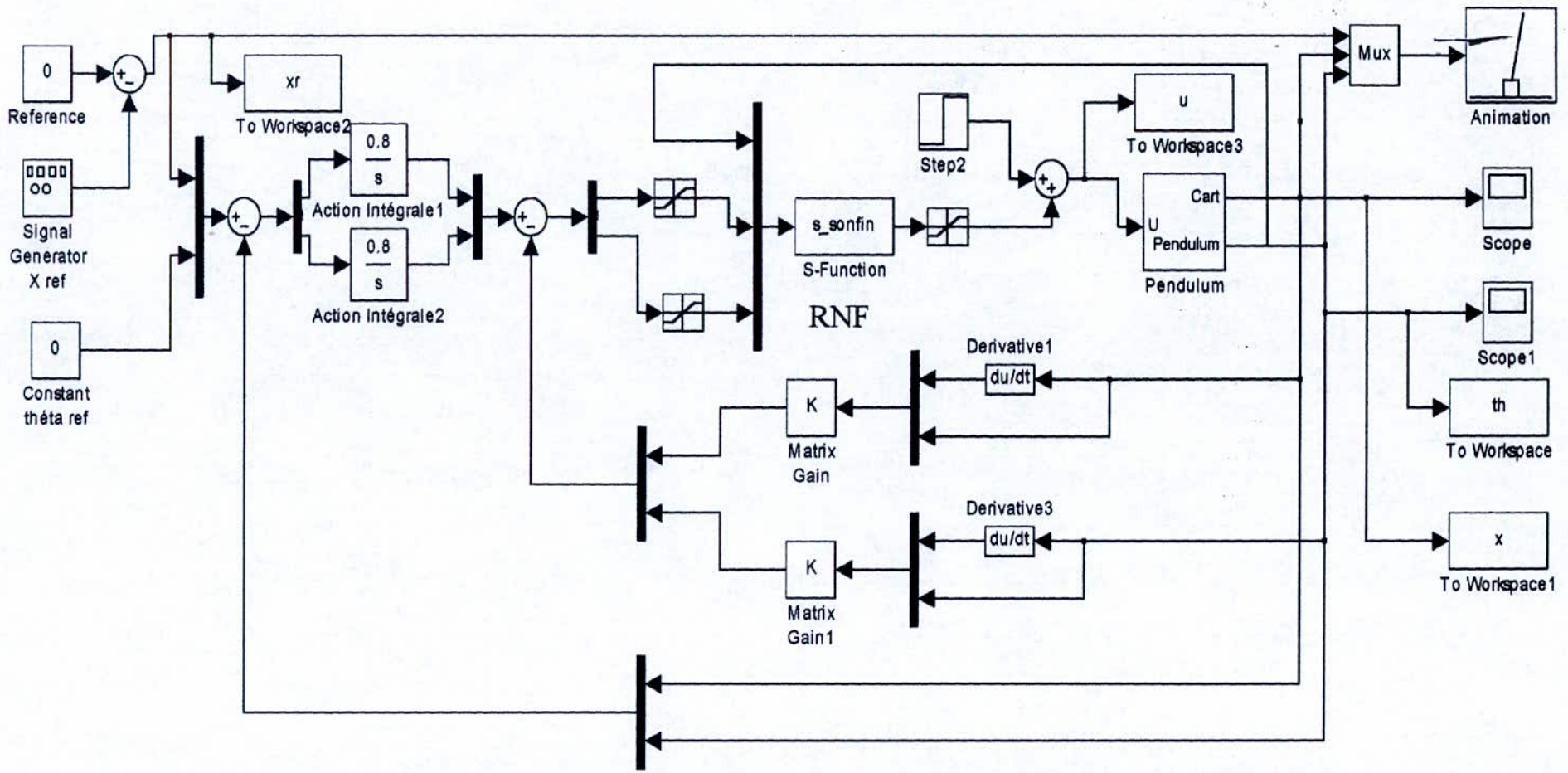


Figure.IV.17 : Schéma bloc du pendule inversé commandé par un réseau neuro-flou adaptatif + un retour d'état et + une action Intégrale.

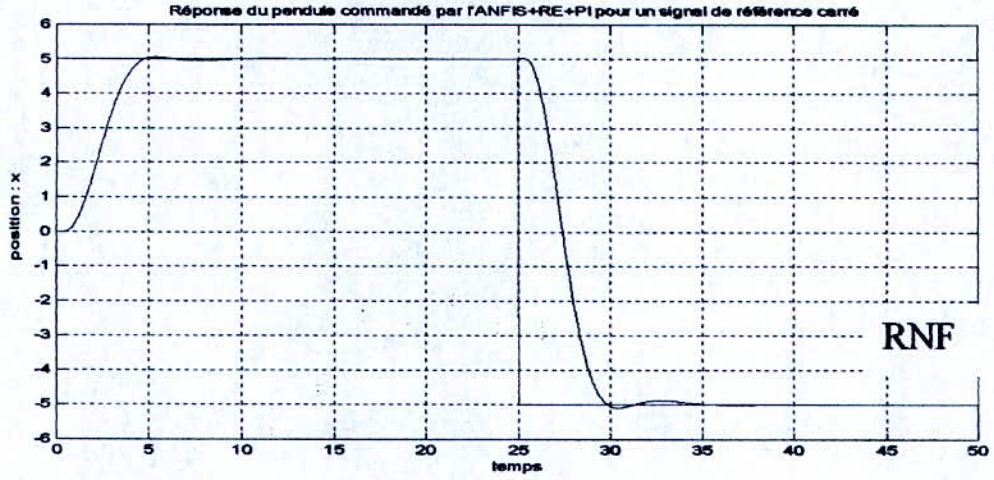


Figure.IV.18. signal de sortie x (en bleu), la référence (en rouge) –ANFIS-

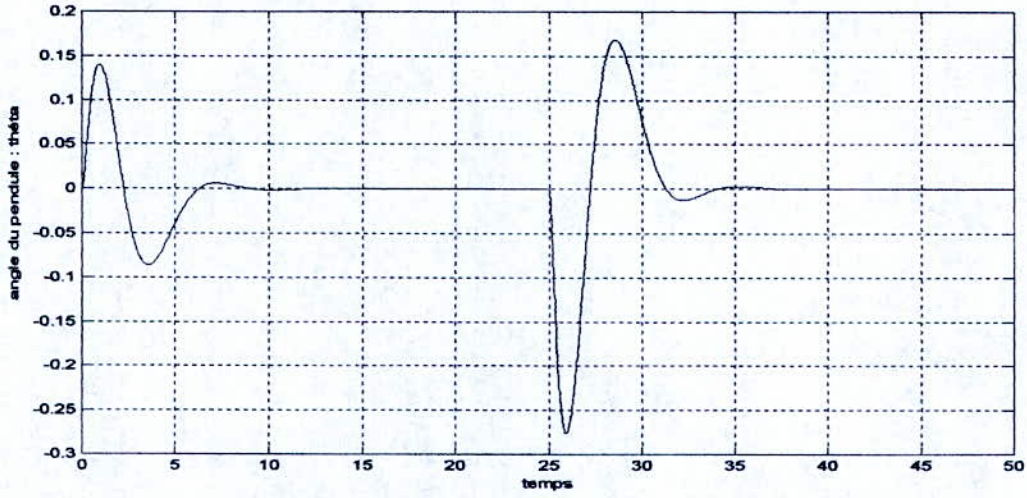


Figure.IV.19. signal de sortie θ (en rad) –ANFIS-

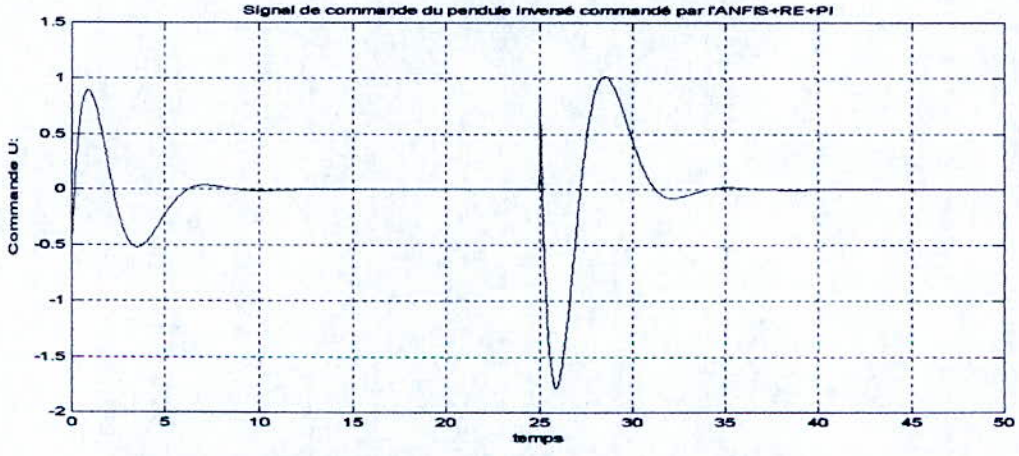


Figure.IV.20. signal de commande U (en Newton) –ANFIS-

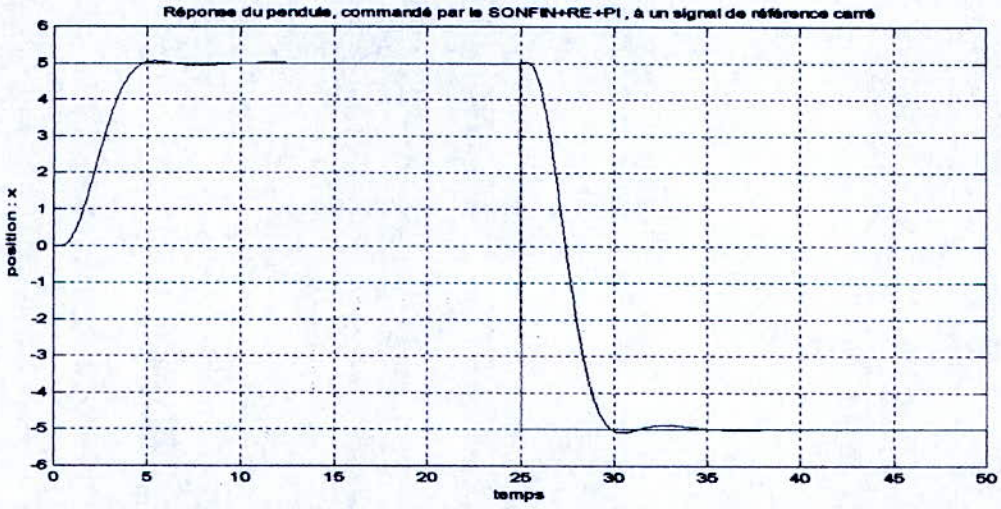


Figure.IV.21. signal de sortie x (en bleu) et signal de référence (en rouge) -SONFIN-

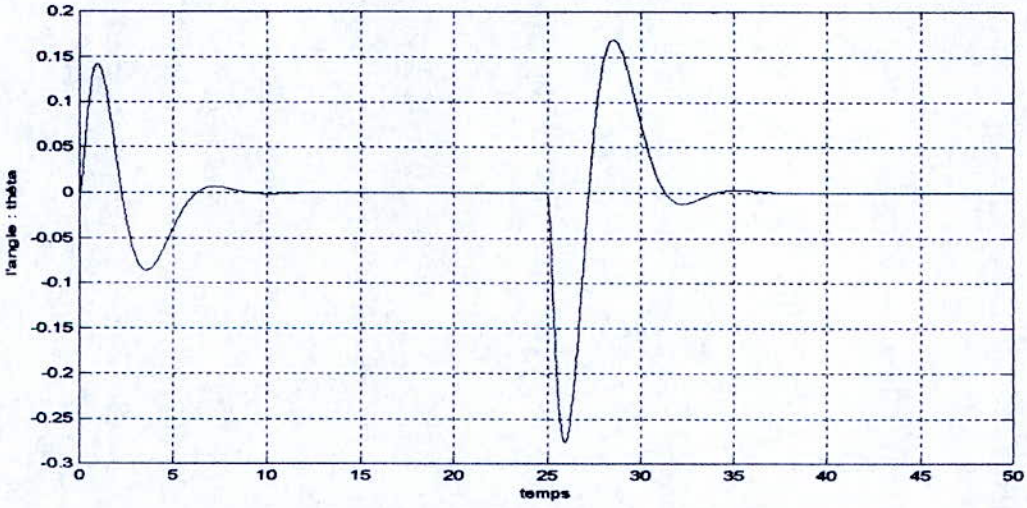


Figure.IV.22. signal de sortie θ (en rad) –SONFIN-

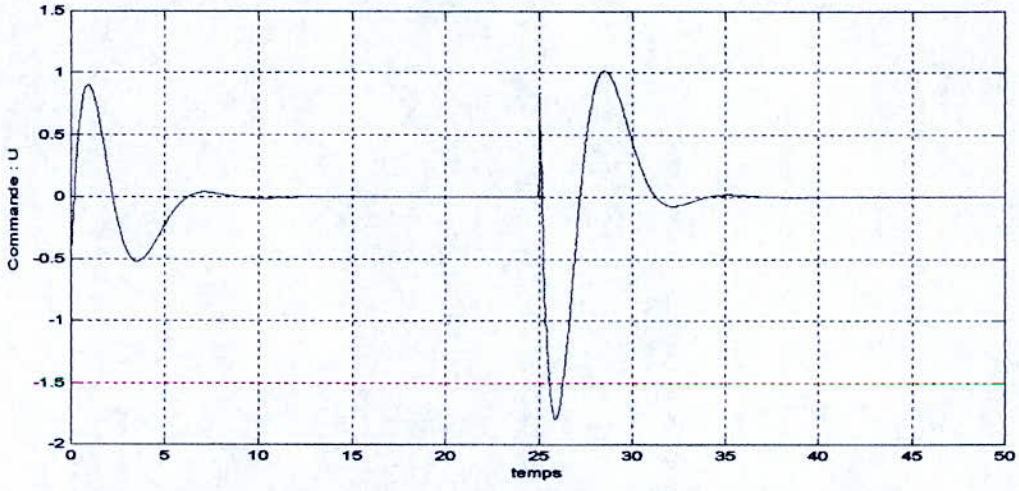


Figure.IV.23. signal de commande U (en Newton) –SONFIN-

On voit bien que la réponse est identique pour les deux cas de commande, par l'ANFIS et par le SONFIN, aussi, elle est très proche de celle d'un système du second ordre stable. Il est à noter que les erreurs statiques sur la position du chariot, x, et du pendule, θ, sont nulles; cela est du a l'action intégrale ajouté ; aussi, on voit que le temps de réponse est assez court, d'environs 4 seconde. (Remarque : l'échelle de temps est en seconde).

Ceci démontre l'efficacité de la stratégie de commande employée.

IV.2.3.3. SIMULATION EN PRESENCE DE PERTURBATIONS

Afin de vérifier le rejet d'une perturbation sur la force u appliqué au chariot, qui peut être causé par le milieu environnent (ex : frottement), nous allons injecter en entrée un échelon, d'amplitude égale à 7 newton à t = 20s, ce signal viendra s'additionné au signal de commande issue du régulateur. Ceci est illustré dans la **figure.IV.17** vu précédemment.

Les résultats de la simulation sont illustrés dans les **figures.IV.24, IV.25, IV.26** pour l'ANFIS et les **figures.IV.27, IV.28, IV.29** pour le SONFIN, où l'on constate qu'après un petit régime transitoire se produisant au moment de la perturbation, les erreurs statiques sur les positions du pendule et du chariot s'annulent. Ce qui démontre l'aptitude de la commande à rejeter ce genre de perturbation.

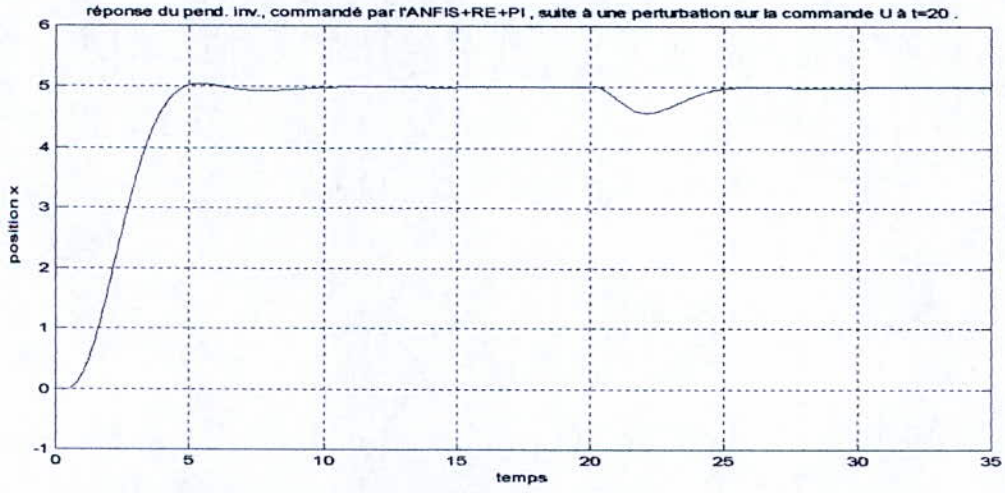


Figure.IV.24 Position du chariot x (en mètre) -ANFIS-

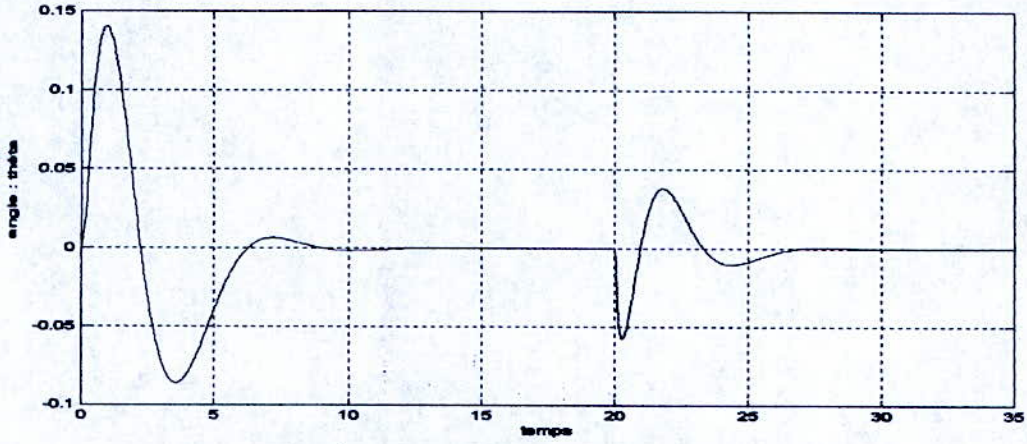


Figure.IV.25 Position du pendule θ (en rad) -ANFIS-

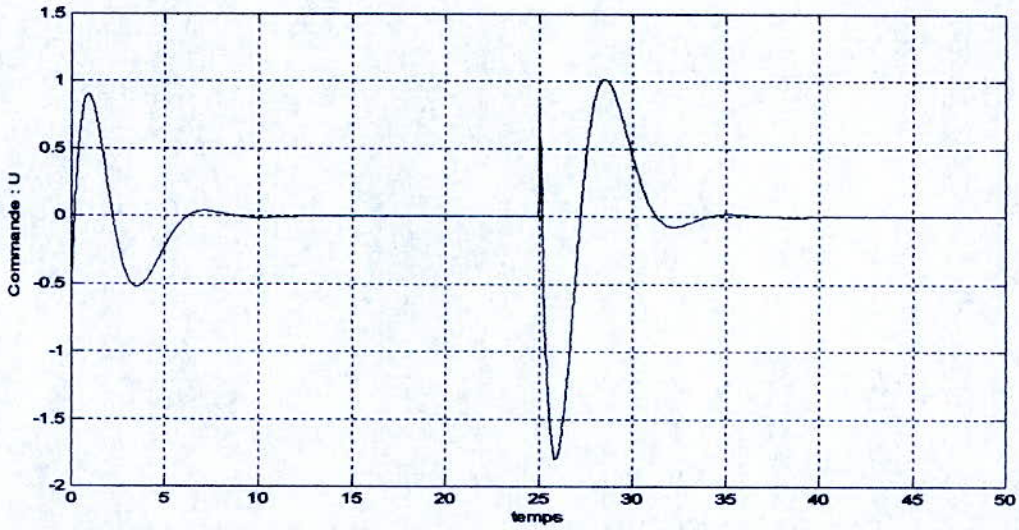


Figure.IV.26 Signal de commande U (en Newton) -ANFIS-

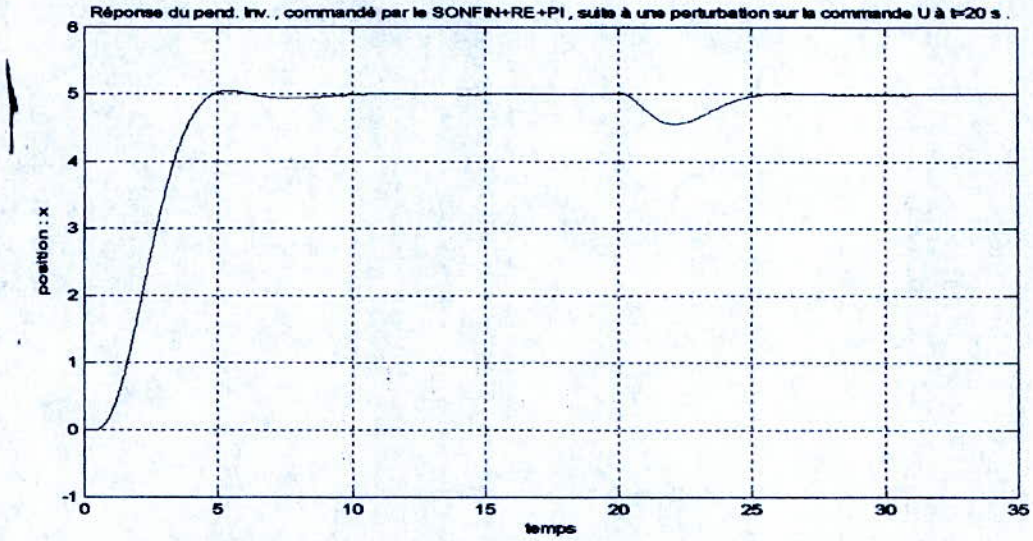


Figure.IV.27. Position du chariot x (en mètre) -SONFIN-

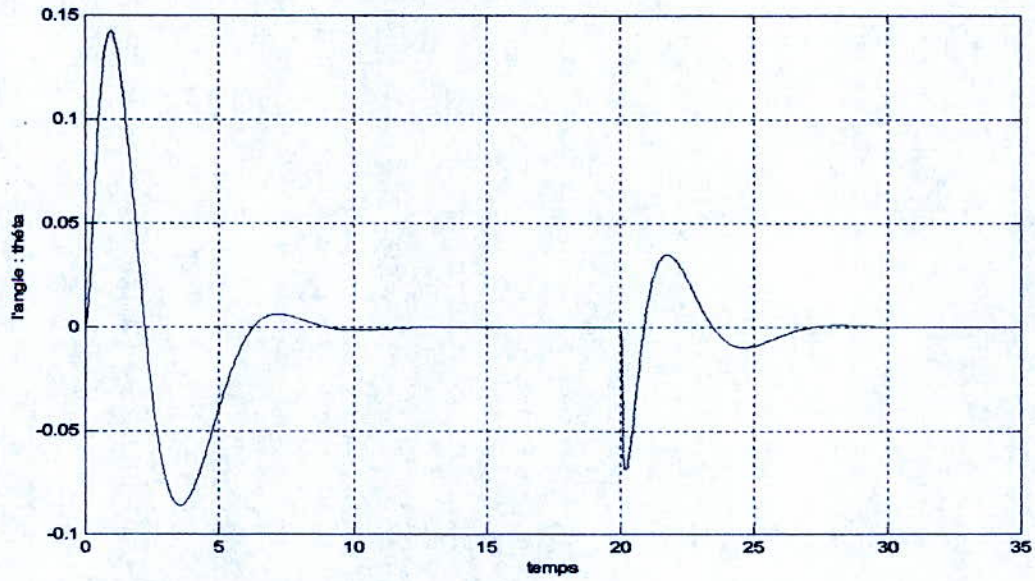


Figure.IV.28. Position du pendule θ (en rad) -SONFIN-

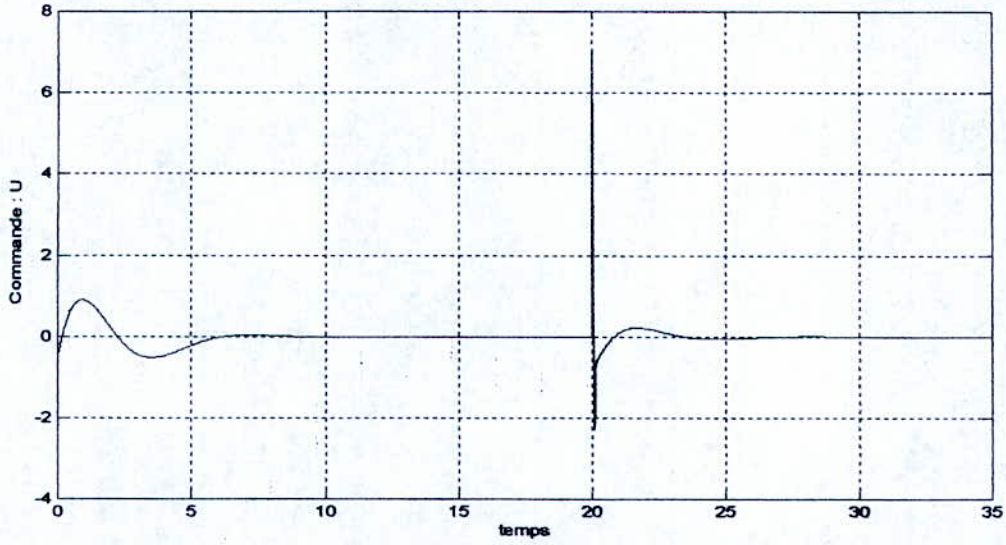


Figure.IV.29 Signal de commande U (en Newton) -SONFIN-

IV.2.3.4. Conclusion

L'observation des figures obtenues après simulation des deux commandes implémentées à l'aide de l'ANFIS et du SONFIN, nous permet de dire que les résultats sont satisfaisants.

Il est à noter aussi que les résultats auxquels nous sommes arrivés que ca soit avec l'ANFIS ou avec le SONFIN sont assez similaires.

IV.3 COMMANDE DE LA TEMPERATURE D'UN BAIN MARIE

L'objectif de cette application est le control de la température (en degrés °C) d'un bain marie en utilisant les deux structures de réseaux neuro-flou adaptatifs à savoir le SONFIN et l'ANFIS.

IV.3.1. STRATEGIE DE COMMANDE

La technique utilisée est la même que celle utilisée dans l'application précédente à savoir la commande par modèle inverse. La différence est que dans cet exemple le système est échantillonné.

Le schéma pratique du système + régulateur est illustré sur la figure suivante :

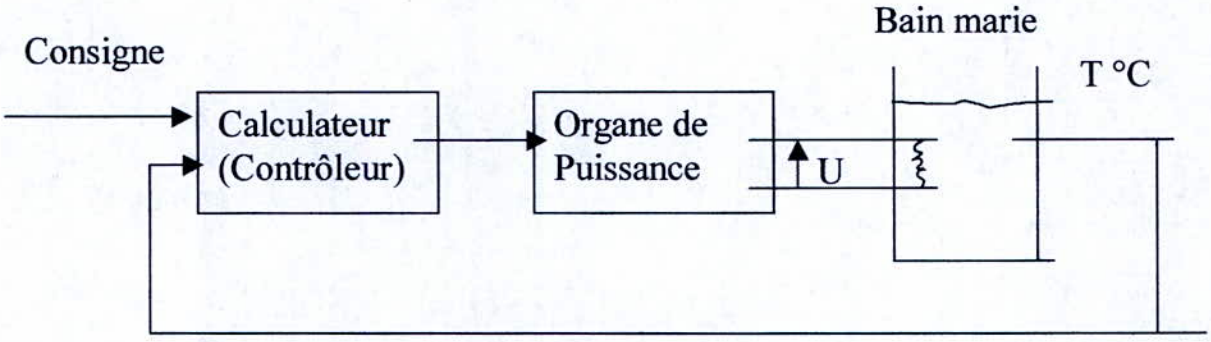


Figure.IV.31. Schéma du système « bain marie + régulateur »

La structure de la commande par modèle inverse, dans le cas discret, en utilisant un réseau d'inférence flou est présenté dans la figure suivante (**Figure.IV.32**):

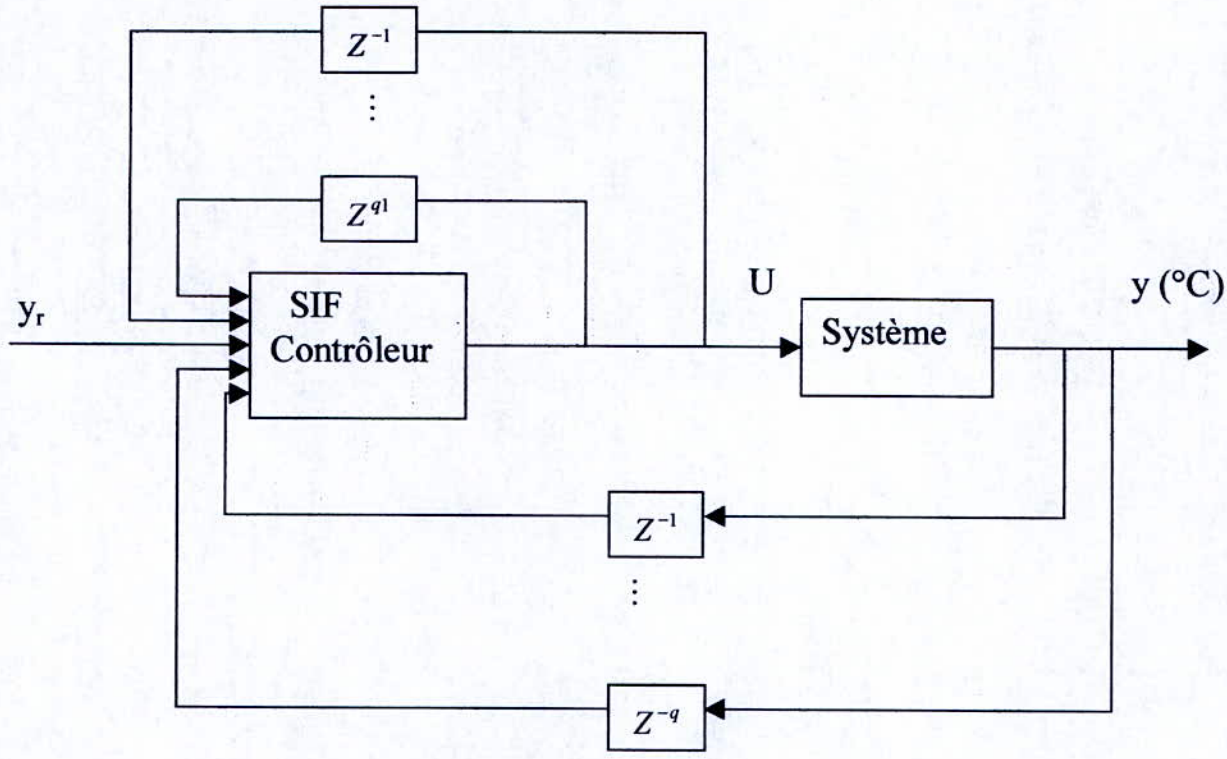


Figure.IV.32. Structure de la commande par modèle inverse, dans le cas discret, en utilisant un réseau d'inférences floue

Le bain marie utilisé dans cet exemple est régi par l'équation aux différences suivante :

$$y(k+1) = a(T_s)\, y(k) + \frac{b(T_s)}{1+\exp(0.5y(k)-r)} U(k) + [1-a(T_s)]\, y_0 \qquad \text{(IV.4)}$$

Où

$$a(T_s) = \exp(-\alpha\, T_s)$$

et

$$b(T_s) = \frac{b}{a}(1-\exp(-\alpha\, T_s))$$

Les paramètres du système utilisé sont [12] :

a = 1.0015 exp(-4)
b = 8.67973 exp(-3)
r = 40
Y_0 = 25 °C

IV.3.2. SIMULATIONS ET INTERPRETATIONS

IV.3.2.a. ACQUISITION DE DONNEES

L'entrée du système, *U*, est la tension appliquée à la résistance chauffante (en Volts). La sortie y représente la température de l'eau (en degrés °C). La période d'échantillonnage T_s est égale à 60 secondes.

Pour générer les échantillons d'apprentissage, le système opère en boucle ouverte. On lui injecte 2000 échantillons aléatoires de tension U et on récupère les échantillons (y(k+1), y(k)) (**Voir la figure.IV.34 réalisée sous MATLAB/SIMULINK**). Ces derniers seront utilisés afin d'entraîner le réseau neuro-flou adaptatif dans le but d'identifier le modèle inverse du système (bain marie) ; la **figure.IV.33**, illustre cette démarche.

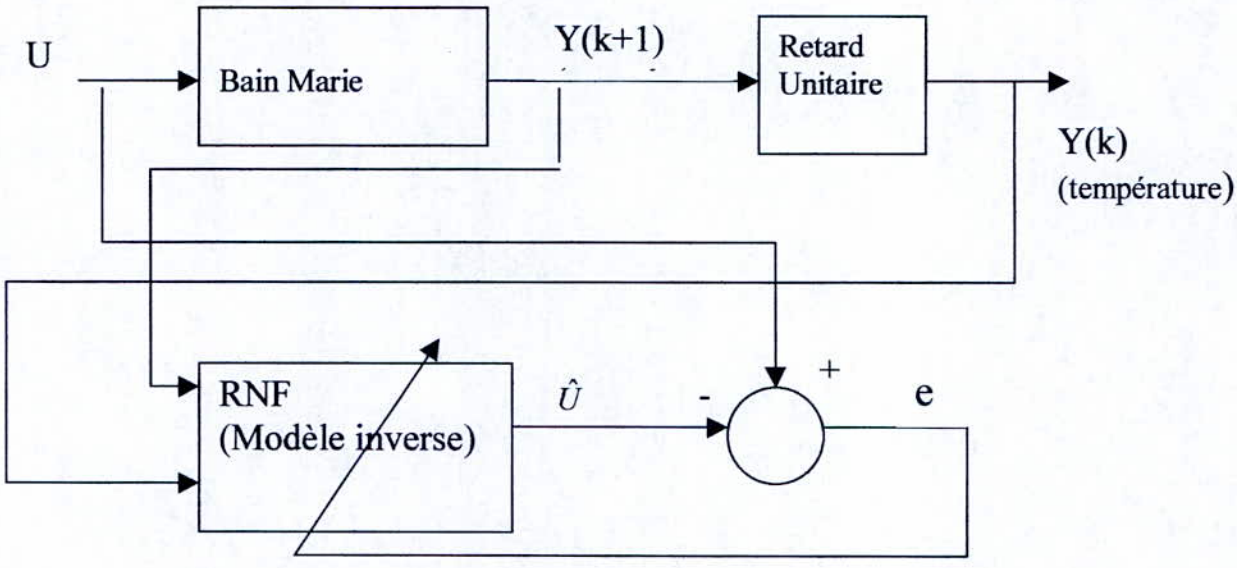
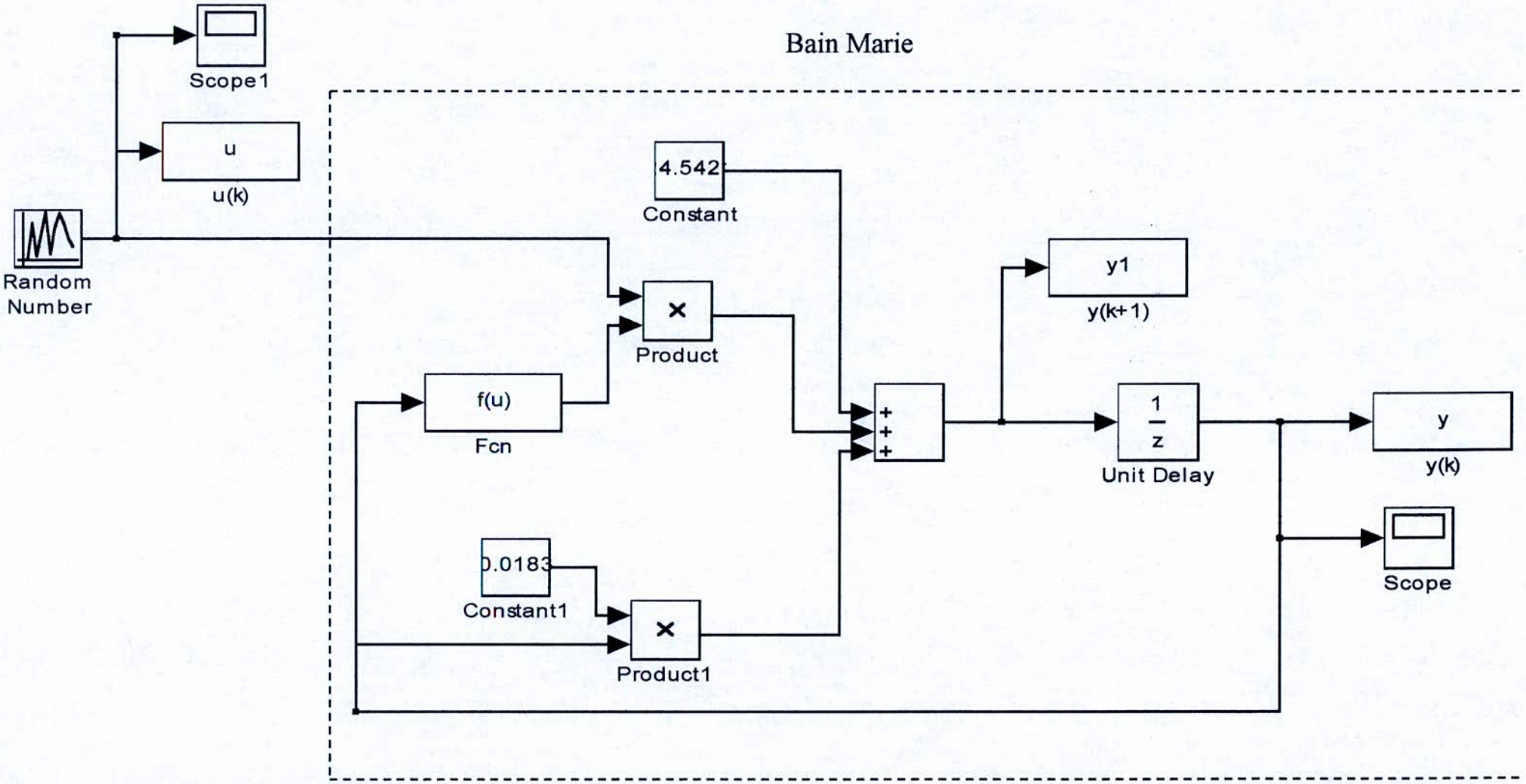


Figure IV.33. Principe de l'identification du modèle inverse du bain marie.

La **Figure VI.36** représente l'erreur obtenue après l'apprentissage en utilisant le SONFIN ; et la **Figure VI.37** représente celle obtenue en utilisant l'ANFIS. On remarque bien, que dans les deux cas, l'erreur est assez faible (de l'ordre de 1.5 %) par rapport au signal de commande, U, illustré dans la **Figure VI.35**, mais néanmoins on constate la présence de quelques pics. Ces derniers ne seront pas très préjudiciable sur la commande, cela peut être vérifié dans ce qui suit.

Figure.IV.34. Schéma bloc du bain marie en boucle ouverte utilisé pour la récupération des échantillons d'apprentissage



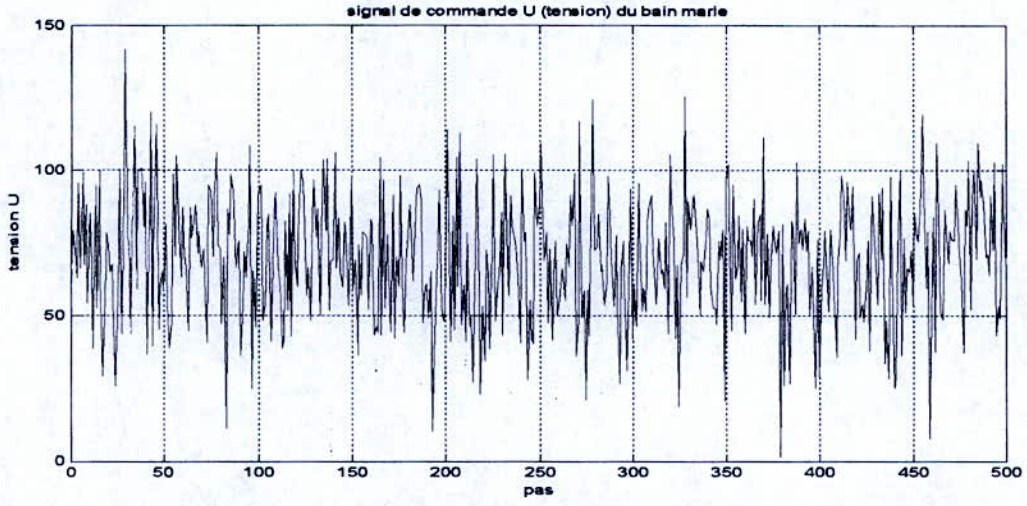


Figure.VI.35. signal de commande U (en Volts) du bain marie

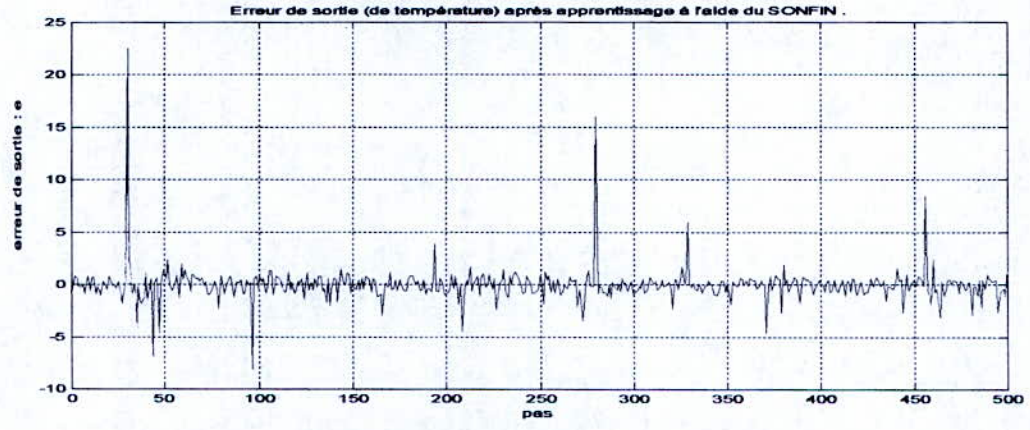


Figure VI.36 erreur de sortie e (en volts) -SONFIN-

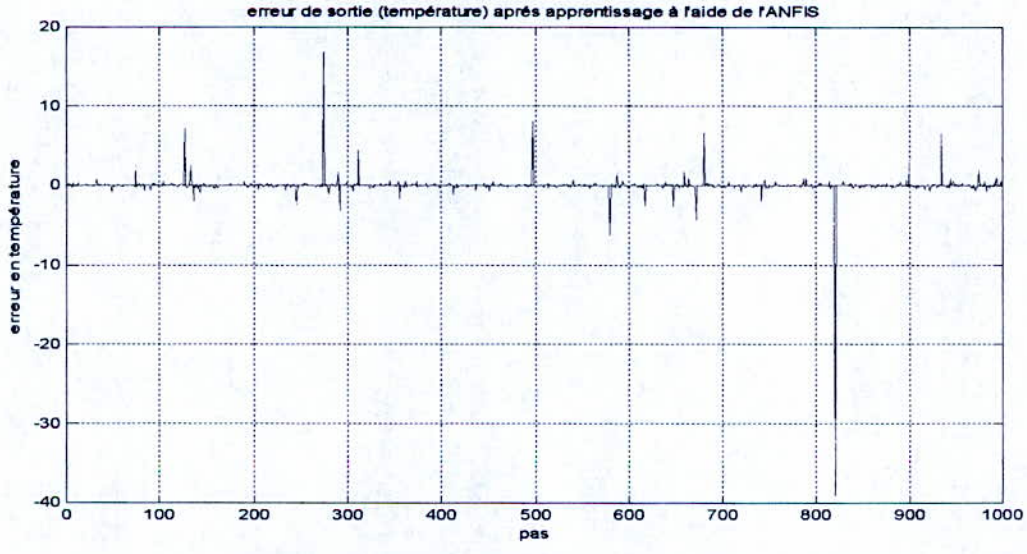


Figure.IV.37. erreur de sortie e (en volts) -ANFIS-

Dans les **figures IV.38, IV.39** sont représentées les MFs d'entrées de l'ANFIS ; elles sont au nombre de 5 pour chaque entrée. Il est à noter aussi que le nombre de règles générées est de 5^2=25 règles.

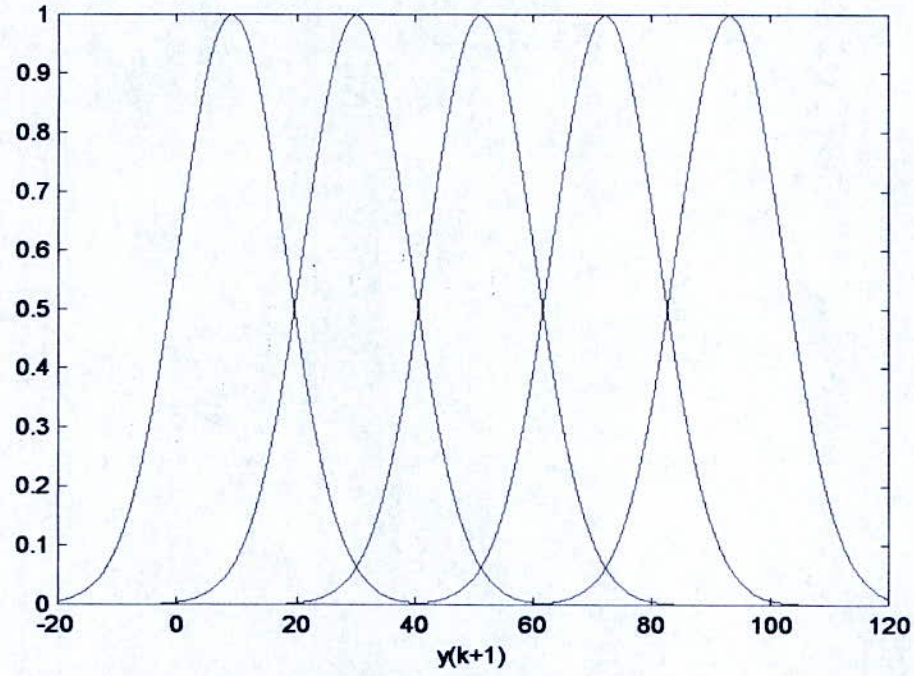


Figure.IV.38. MFs de l'entrée 1 y(k+1) (en °C) de l'ANFIS

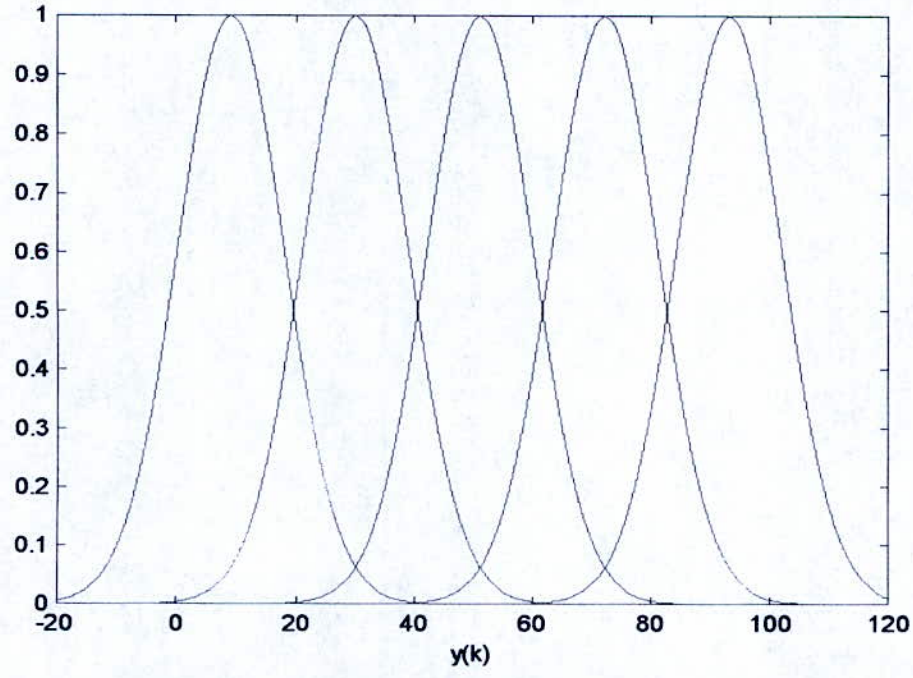
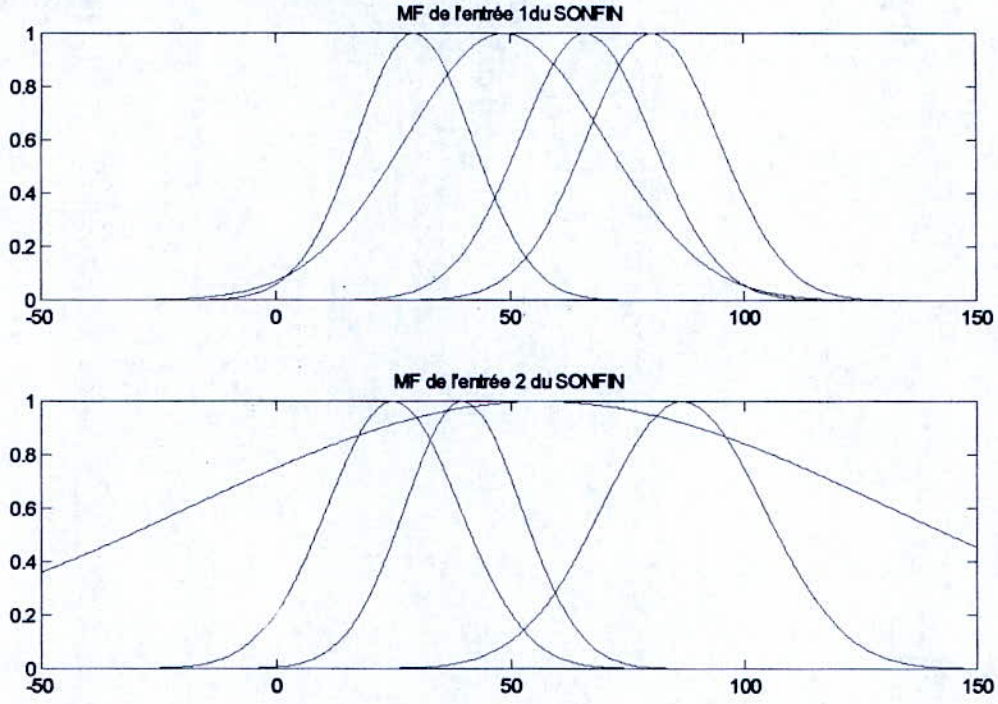


Figure.IV.39. MFs de l'entrée 2 y(k) (en °C) de l'ANFIS

La **figure.IV.40 ;** illustre les MFs des entrées 1 et 2 du SONFIN elles sont au nombre de 4 pour chaque entrée. La **figure.IV.41** quant à elle représente les 5cinqs MFs de sortie du SONFIN. Le nombre de règles générées est égal à 8.



FigureIV.40. MFs des entrées y(k+1) et y(k) (°C) du SONFIN

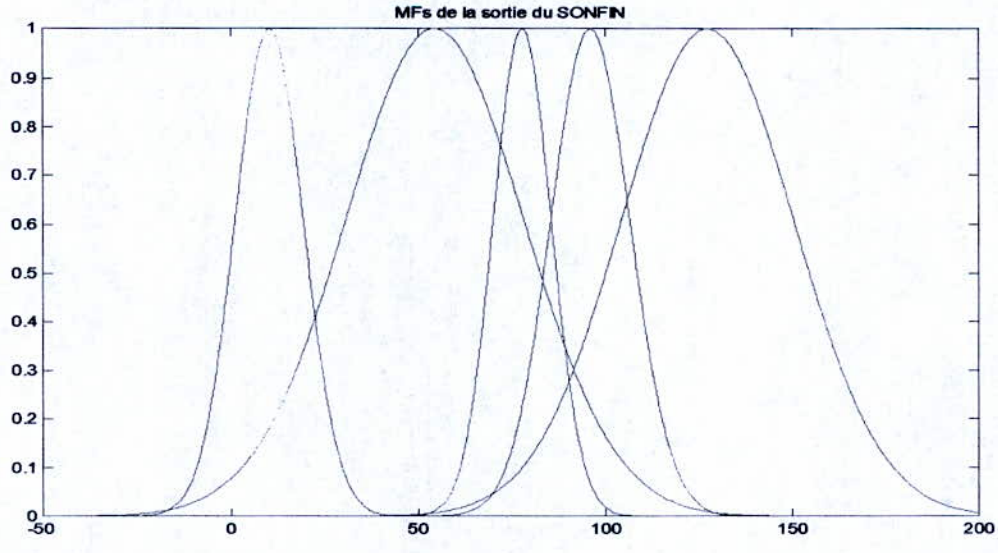
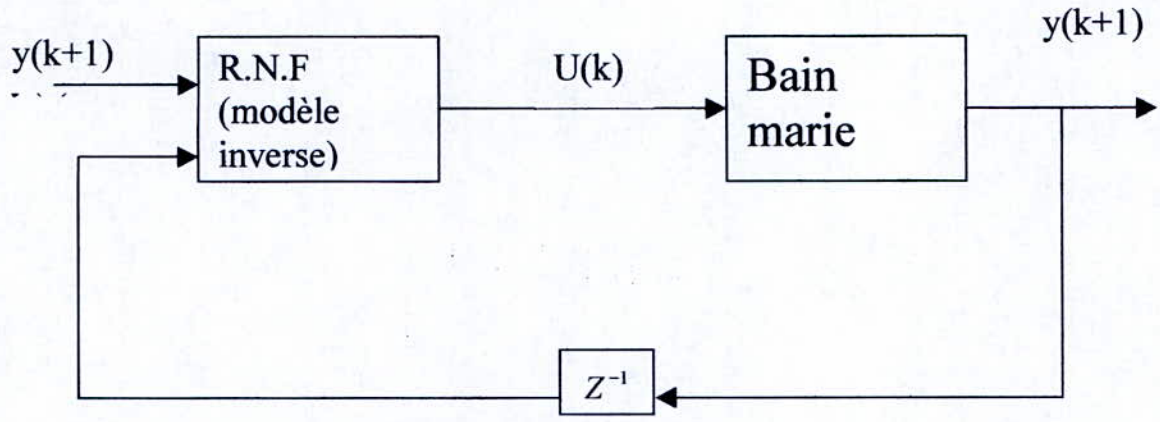


Figure.IV.41. MFs de la sortie u(k) (en volts) du SONFIN

IV.3.2.b. SIMULATION DE LA COMMANDE

Le schéma bloc du système commandé est comme montré sur la figure suivante :



FigureIV.42. schéma bloc du système commandé

Le schéma de simulation, de la commande du bain marie par un RNF, créé sous MATLAB/SIMULINK est montré dans la **figure.IV.43**.

Pour tester notre commande nous injectons un signal de référence en escalier (Yr = 40 à t=0 ; Yr = 55 à t = 5 et yr = 70 à t=10). On observe le signal de sortie y (température) dans la **figure.IV.44** (pour le SONFIN) et **figureIV.46** (pour l'ANFIS), on voit bien que la poursuite, dans les deux cas du SONFIN et de l'ANFIS, est assez performante et que le temps de réponse est de 2 à 3 périodes d'échantillonnage. Les signaux de commande sont représentés dans les figures **IV.45.** et **IV.47**

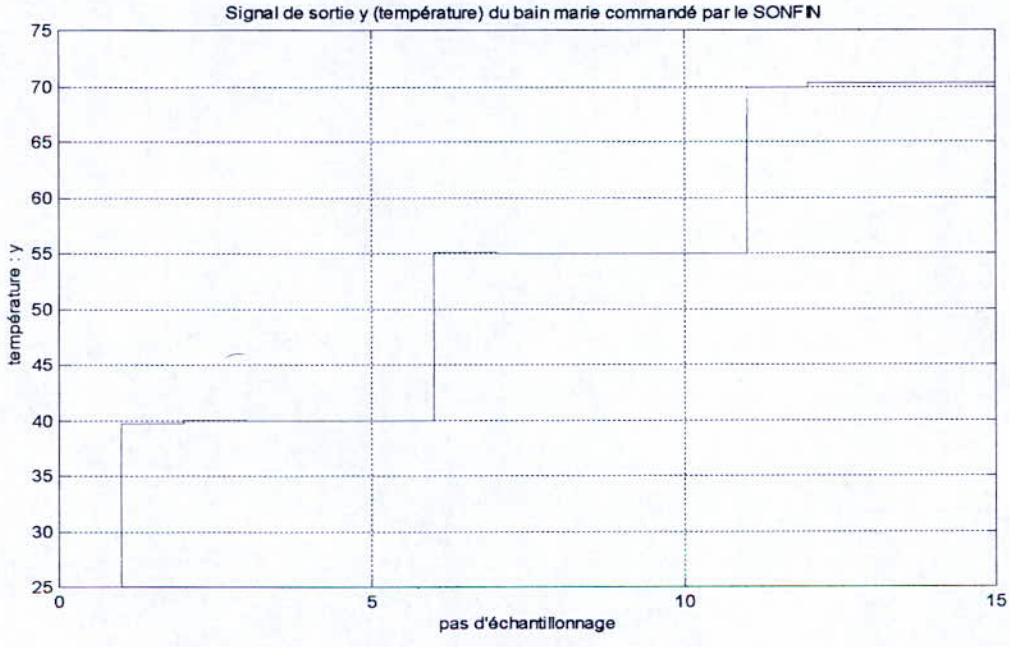


Figure IV.44. Signal de sortie y(k) (en °C) du bain marie –SONFIN-

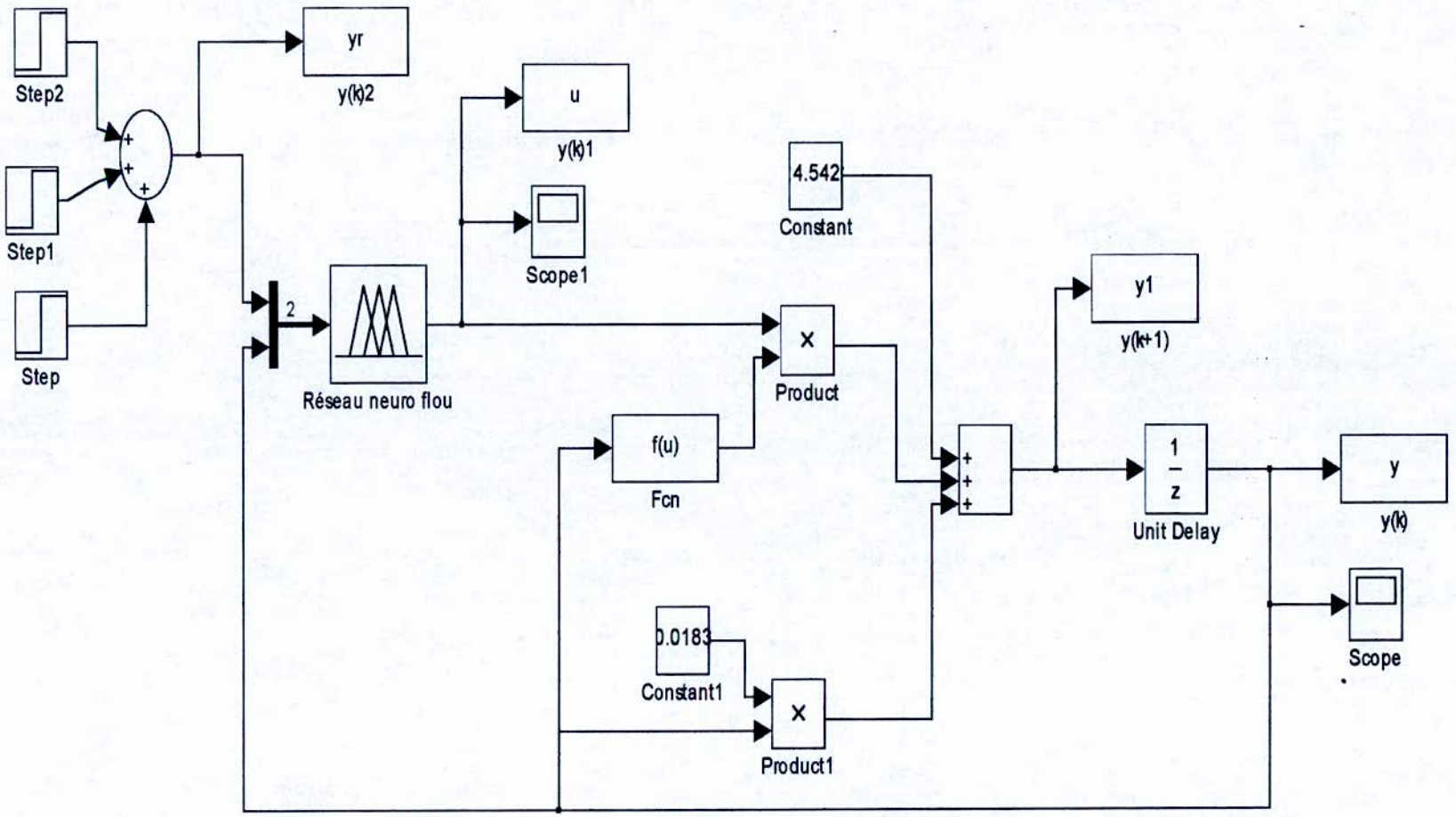


Figure.IV.43. Schéma bloc du bain marie commandé par un réseau neuro-flou adaptatif

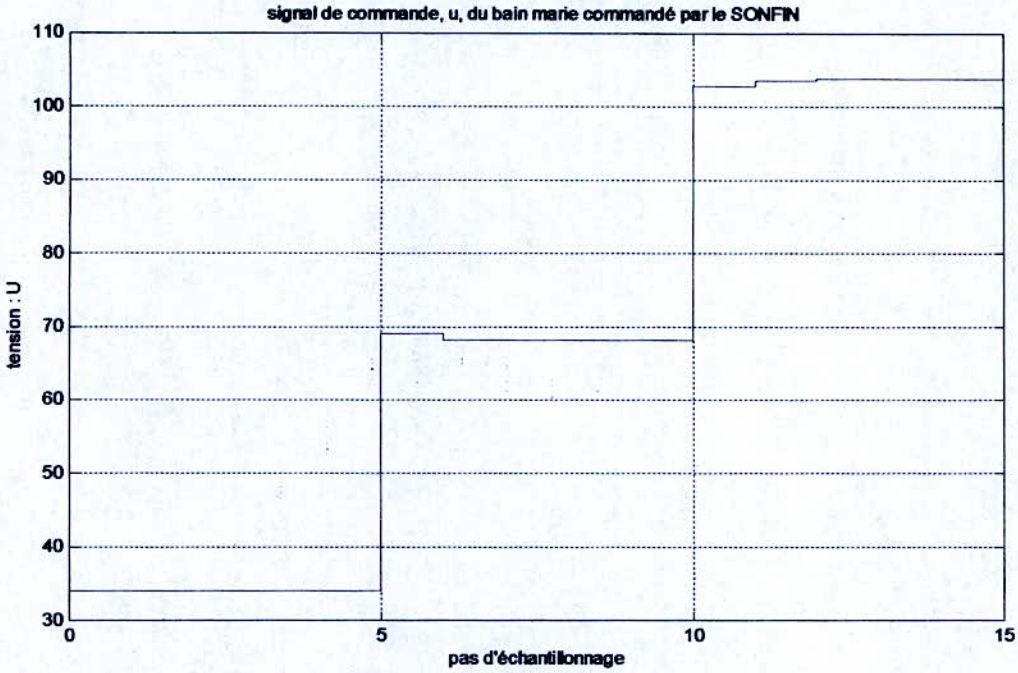
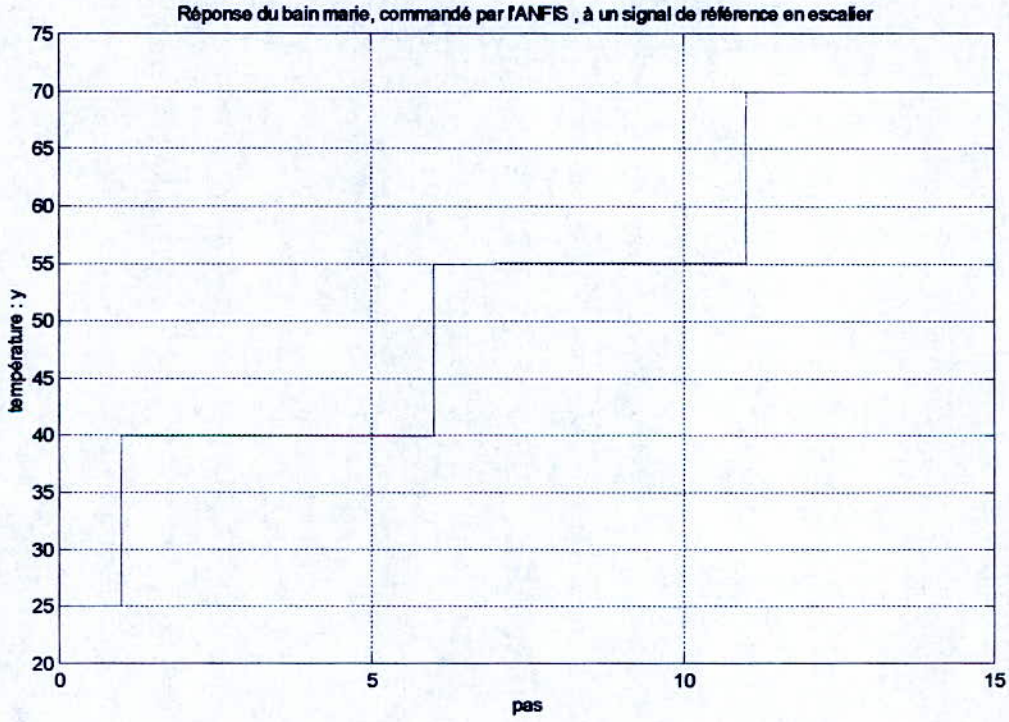


Figure.IV.45. signal de commande U(k) (en volts) -le SONFIN-



FigureIV.46. signal de réponse y(k) (en °C)-ANFIS-

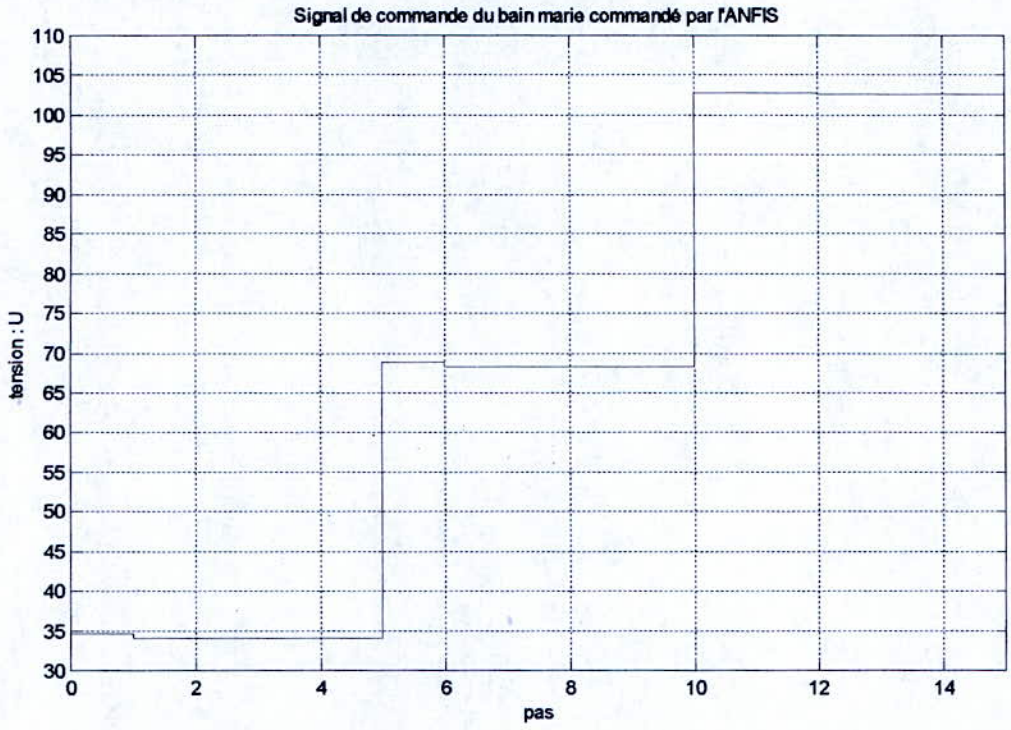


Figure.IV.47. signal de commande U(k) (en volts) (l'ANFIS)

IV.3.3. CONCLUSION

En conclusion, et après avoir simuler les deux commande se basant sur les deux structures neuro-flous à savoir l'ANFIS de MATLAB et le SONFIN que nous avons implémenté, on peut affirmer que ces deux commandes sont assez efficaces est même assez similaires vu les courbes obtenues ce qui démontre l'efficacité de la commande par model inverse combiné au réseaux neuro-flous dans le cas discret. Aussi, on peut constater l'avantage du SONFIN dû au petit nombre de règles générées comparé à l'ANFIS.

Conclusion Générale

CONCLUSION

L'étude effectuée dans ce projet de fin d'étude concerne une des méthodes récentes de modélisation des systèmes qui est basée sur l'utilisation des réseaux neuro-flous. Cette approche présente des performances considérables par rapport aux approches traditionnelles, car elle combine entre les avantages des méthodes de la logique floue et ceux des réseaux de neurone.

Après avoir introduit, dans les chapitres I et II, les concepts relatifs à la logique floue, ainsi que les algorithmes d'apprentissage des réseaux neuro-flous, nous avons présenté dans le chapitre III, en détail, l'algorithme utilisé par le réseau neuro-flou SONFIN. Nous avons alors implémenté ce dernier pour l'élaboration d'une commande en position pour le pendule inversé, ainsi que le réglage de la température d'un bain marie. La première application étant un cas continu, et la seconde un cas discret.

Nous avons donc constaté, à travers ces deux applications, que l'efficacité des RNF, combinée à une bonne stratégie de commande, permet d'obtenir des commandes intelligentes, très performantes et assez robustes par rapport aux perturbations, et ceci en évitant les problèmes rencontrés lors de l'utilisation d'une modélisation analytique, surtout si on a affaire à des systèmes très complexes et fortement non linéaires.

Comme perspectives, nous proposons d'améliorer la structure du SONFIN afin qu'il puisse achever un apprentissage d'un système instable sans avoir à recourir à un stabilisateur, comme c'était le cas pour l'exemple du pendule inversé. Et pour prendre vraiment conscience de l'intérêt de cette nouvelle approche, il est impératif d'essayer d'implémenter cette technique sur un système pratique, en utilisant un calculateur assez performant, ainsi on pourra adapter la structure du SONFIN et son algorithme d'apprentissage de telle façon qu'il prenne en considération les aspects pratiques qui n'apparaissent pas lors d'une étude théorique, telle que la non disponibilité de signaux de bonne qualité, surtout si on opère dans un milieu hostile et/ou très bruité.

Une autre perspective est l'application de ces techniques intelligentes dans d'autres domaines, particulièrement en robotique, et ceci lorsqu'il s'agit de l'apprentissage des trajectoires et de la reconnaissance de forme. Ainsi en améliorant ces techniques et leurs applications on arrivera à concevoir des machines assez autonomes pouvant remplacer l'humain dans des tâches pénibles et/ou délicates, et aussi, améliorer la sécurité de son environnement.

BIBLIOGRAPHIE

[1]- P.Y. Glorennec, «Algorithmes d 'apprentissage pour systèmes d 'inférence flous», Germes, sept. 1999.
[2]- S. Hrikawa, T. Fruhashi and Y. Uchicawa, «Fuzzy modeling using fuzzy neural networks with the back propagation algorithm », IEEE Trans, Neural network, vol.3 pp. 801-806, sep 1992.
[3]- M. Sugeno and T. Yasukawa, «A fuzzy logic based approch to qualitatif modeling », IEEE Trans. Fuzzy set vol.1, pp. 7-31, May 1995.
[4]- B. Kosko, «Neural Networks and Fuzzy Systems », Englewood Cliffs, NJ: Prentice-Hall, 1992.
[5]- C.T. Lin and C.S.G. Lee, « Neural fuzzy systems : A Neural fuzzy synergim to intelligent systems, Englewood Cliffs, NJ : Prentice Hall, 1996.
[6]- L.X. Wang and J.L. Mandel, « Fuzzy bases functions, universal approximation and orthogonal least squares learning », IEEE Trans, Neural networks, vol.3 pp. 807-814, sept. 1992.
[7]- E.H Ruspini, « Numerical methodes for fuzzy clustering », Inform Sci. Vol.2 pp. 319-350, 1970.
[8]- C.T Lin, « Neural fuzzy control with structure and parameter learning», IEEE Trans.
[9]- H. Ishibushi, K. Nosaki, N. Yamamoto, and H. Tanaka, « Selecting fuzzy ... », IEEE Trans. Fuzzy Syst. Vol.3 pp. 260-270, aug. 1995.
[10]- C.T. Lin and C.S.G. Lee, « neural network based fuzzy logic control and decision system » IEEE Trans. Comput, vol. 40, pp. 1320-1336, Dec 1991
[11]- D.B Parker, « Optimal Algorithms for adaptif network », IEEE int.conf on neural networks.
[12]- C.F. Juang and C.T. Lin, « An On-Line Self-Constructing Fuzzy Inference Network and its Applications », IEEE Trans. On fuzzy systems, vol.6 February 1998.
[13]- Brigitte d'Andréa-Novel, Michel Cohen de Lara, « Commande linéaire des systèmes dynamiques », Edition MASSON .
[14]- M. Mokhtari, A. Mesbah, «Apprendre et maîtriser MATLAB », Edition Springer, 1997.
[15]- J. Roger Jang and C.T. Sun, «Neuro-Fuzzy Modeling and Control », IEEE Trans, Vol.83, N° 3, March 1995.
[16]-M. Sugeno and G.T. Kang «structure identification of fuzzy model», Fuzzy Sets and Systems, vol.28 pp 15-33,1988.
[17]- A. Abraham and B. Nath, «Designing Optimale Neuro-Fuzzy Architectures for Inetlligent Control ».
[18]- R. Kruse and A.Nurnberger, « Learning Methods for Fuzzy Systems ».
[19]- C.T. Lin and C.S.G. Lee, « Reinforcement Structure/Parameter Learning

for Neural-Network-Based Fuzzy Logic Control Systems », IEEE Transactions on fuzzy systems, vol. 2, NO. 1, Feburuary 1994.

[20]- A.Ajith « Beyoud Integrated Neuro-Fuzzy Systems : reviews, perspectives and directions »

[21]- C.T Sun, «Rule-Base Structure Identification in an Adaptative-Network-Based Fuzzy Inference System», IEEE transactions of fuzzy systems, vol 2, NO.1, February

ملخص :

تخص مذكرة نهاية الدراسة هذه ، تمثيل المنظومات عن طريق الشبكات العصبونية الغامضة ذات التلاؤم الذاتي.
في هذا العمل تمت دراسة نوعين من الشبكات العصبونية الغامضة المتمثلين في ANFIS، SONFIS.
طبقت هذه الطريقة على نواس مقلوب بغرض التحكم في موضعه و كذا على حمام ماري من اجل ضبط درجة حرارته و من خلال النتائج المحصل عليها تم الإثبات على فعالية منظومات التحكم الناتجة، و كذا ضلاعتها اتجاه الاضطرابات الخارجية.

الكلمات المفتاحية : الشبكات العصبونية الغامضة، التعلم، التلاؤم، ANFIS, SONFIN.

Résumé

Ce projet de fin d'études concerne l'étude d'une méthode de modélisation sur les réseaux neuro-flous auto adaptatifs.
Dans ce travail deux structures de RNF ont été présentées : l'ANFIS et le SONFIN.
Une application de l'approche neuro-flous sur la commande en position d'un pendule inversé ainsi qu'a la commande de la température d'un bain marie démontre la bonne performance du point de vue poursuite de référence et stabilité des systèmes de commande résultant ainsi que leurs robustesse vis à vis des perturbations externes.

Mots clés : Réseaux neuro-flous, Apprentissage, adaptation, SONFIN, ANFIS.

Abstract

This final studies project consist on studying a modelling method based neural-fuzzy networks.
In this work two structures of neural-fuzzy inference networks are presented: the ANFIS and the SONFIN.
An application of neural fuzzy approach on position control of an inverted pendulum and temperature control of a water bath shows the efficiency of the control system resulting. Also, it demonstrates their robustness toward external disturbing.

Keywords : Neural fuzzy networks, Learning, adaptation, SONFIN, ANFIS.