République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur
et de la recherche scientifique
Université de Mouloud Mammeri de Tizi Ouzou

Faculté de Géni électrique et d'informatique

Mémoire de fin d'étude

En vue d'obtention de diplôme d'ingénieur d'état en automatique

Thème

Commande floue et neuro-floue du niveau de liquide d'un réservoir

étudié par

Saidoun Ouerdia

encadré par

Hasnaoui Mohammed

Promotion : 2007/2008

Remerciements

Je tiens à exprimer mes remerciements a l'encontre de mon promoteur Mr. Hasnaoui d'avoir proposé ce travail, pour son aide et pour ses conseils précieux.

Je remercie aussi, les membres de jury qui ont acceptés de juger ce travail et d'y apporter leur contribution.

Dédicace

Je dédie ce travail

- ✓ *A ma très chère maman ; mon père*
- ✓ *Toute ma famille*
- ✓ *Tous mes amis ; à Meriama, Soraya, Karima, Nassima , Mokrane , Ghiles, Mouhamed . . . . . .*
- ✓ *A ceux qui m'ont aidés Djamel, Abrous, Mouhamed, Hakim.*

Sommaire

Chapitre II : Introduction a la logique floue

Chapitre III : La commande neuro-floue

Chapitre IV : Application et résultats de simulation

Introduction générale

L'industrie, de nos jours fait face aux problèmes de contrôle des systèmes non linéaires de plus en plus complexes qui présente des modèles mathématiques inconnus ou difficiles à exploiter. Cependant des développements importants sur le plan pratique et théorique ont été effectuées dans le domaine de la commande des systèmes industriels .Plusieurs travaux furent concrétisés ; pour sa part la recherche dans la dynamique et le contrôle des procédés non linéaires a connu également un développement remarquable.

Les méthodes traditionnelles utilisées pour la commande de ces processus sont parfois difficiles à concevoir en raison de la nécessitée de posséder un modèle mathématique précis du procédé. Elles peuvent aussi être lourdes à mettre en œuvre.

Ces dernières années des techniques non conventionnelles comme les réseaux de neurones et la logique floue ont connus un grand essor dans le domaine de l'automatique.

Dans le cas des réseaux de neurones leur propriété d'approximation parcimonieux ainsi que leur nature multi variable en ont fait des outils très puissants pour la modélisation et la commande de procédés non linéaires complexes .De même, dans le cas des systèmes d'inférence floue (SIF), leur propriété d'approximation universelle et surtout la possibilité de s'appuyer sur des connaissances experts sur les processus à piloter pour concevoir un système de commande , ont permis de les utiliser avec succès dans de nombreuses applications .ces connaissances se traduisent sous forme de règles telles que :

« Si telle condition alors telle conclusion ».

Par fois la connaissance d'un expert ne peut pas être entièrement verbalisé , cette limitation inhérente au système d'inférences floue a conduit à la conception de nouvelles stratégies de commande hybrides « NEURO-FLOU » intégrant les avantages que présente chacune de ces stratégies pour aboutir à une représentation plus complète des connaissances et ainsi à une amélioration des performances de régulation .Le but de ce travail est la commande d'un système non linéaire qui est un réservoir du liquide.

Les aspects théoriques et pratiques de ce travail sont répartis sur quatre chapitres.

Le chapitre1 présente les bases générales des réseaux de neurones.

Le chapitre 2 présente les bases de la logique floue et le réglage par logique floue des processus industriels.

Le chapitre 3 est dédie à la commande neuro-floue, et à la description de l'architecture ANFIS utilisée dans application.

Le chapitre 4 illustre les résultats de simulations de la commande par PID flou et neuro-floue appliquée à un système non linéaire (contrôle de niveau du liquide dans un réservoir)

Le mémoire se termine par une conclusion générale sur le travail réalisé.

I.1. Introduction

Lors de L'émergence d'une nouvelle technique l'ingénieur se demande naturellement en quoi elle peut lui être utile, surtout si elle est dotée d'un nom plus métaphorique que scientifique ce qui est évidement le cas pour les réseaux de neurones. Un réseau de neurones est conçu de structure aléatoire artificielle, sa conception est très schématiquement inspirée du fonctionnement de la cellule nerveuse (le neurone) biologique.

Un réseau de neurones est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente.

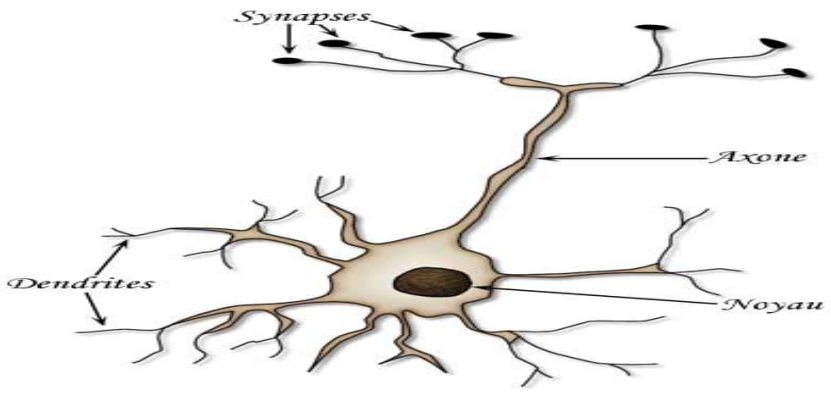
Un réseau de neurones est constitué d'un très grand nombre de petites unités de traitement identiques appelées neurones artificiels, elles étaient électroniques dans les premières implémentations (perceptrons de rosenblatt) aujourd'hui on les simule le plus souvent sur ordinateur pour des questions de cout et de commodité.

Dans ce chapitre nous abordons les principales architectures et règles d'apprentissage des réseaux de neurones. Il ne s'agit pas de les étudier toutes car elles sont trop nombreuses, mais plutôt de comprendre les mécanismes internes fondamentaux et de savoir comment et quand les utiliser.

I.2. Historique

L'étude sur les réseaux de neurones a débuter en 1890 par le célèbre psychologue américain W.James, qui a introduit le concept de mémoire associatives et propose un algorithme d'apprentissage .cependant, la première modélisation du neurone biologique a été faite en 1943 par J.Mc.Culloch et W.Pitts

La compréhension et la modélisation du cerveau ont permis d'isoler le composant cellulaire de base du cerveau, appelé neurone, qui est l'unité de traitement de l'information, dont les constituants élémentaires sont les dendrites, le noyau (ou corps cellulaire) et l'axone.



Chaque neurone reçoit un ensemble de potentiels excitateurs ou inhibiteurs par l'intermédiaire des synapses qui le relient aux neurones qui le précédent. Les dendrites calculent une somme pondérée de leurs entrées. Selon le niveau d'activation obtenu, le noyau génère ou non un potentiel d'action qui se propage le long de l'axone. Ainsi, ce modèle biologique simple sert de base au modèle mathématique du neurone formel.

I.3. Domaines d'utilisation

Les grands domaines d'application des réseaux de neurones découlent naturellement de leur propriété fondamentale :

• L'approximations des fonctions ou modélisation de données statiques : il existe une immense variété de phénomènes statiques qui peuvent être caractérisés par une relation déterministe entre des causes et des effets ; les réseaux de neurones sont de bons candidats pour modéliser de telles relations à partir d'observations expérimentales, sous réserve que celles-ci soient suffisamment nombreuses.

• la modélisation de processus dynamiques non linéaires : modéliser un processus, c'est trouver un ensemble d'équations mathématiques qui décrivent le comportement dynamique du processus, c'est-à-dire l'évolution de ses sorties en fonction de celle de ses entrées ; c'est donc typiquement un problème qui peut être avantageusement résolu par un réseau de neurones.

• la commande de processus : commander un processus, c'est imposer à celui-ci un comportement Défini à l'avance en fonction des signaux de commande ; l'ensemble commande et processus peut donc être considéré comme un système qui réalise une fonction (non-linéaire) qu'un réseau de neurones peut approcher.

• la classification : supposons que l'on désire classer des formes en deux catégories, A ou B, en fonction de certaines caractéristiques de ces formes ; on peut définir une fonction. Qui vaut +1 pour toutes les formes de la classe A et -1 pour toutes les formes de la classe B. Les réseaux de neurones sont de bons candidats pour réaliser une approximation de cette fonction

I.4.1. Model de base d'un neurone formel

Un neurone formel se modélise comme un système a n entrées, pondérées par des coefficients parfois appelés coefficients synaptiques et une sortie.

La fonction de transfert également appelée fonction d'activation du neurone est généralement une fonction non linéaire croissante et bornée, le potentiel d'activation de neurone qui s'exprime comme une somme pondérée de l'activité des neurones qui sont connectés a ce neurones.

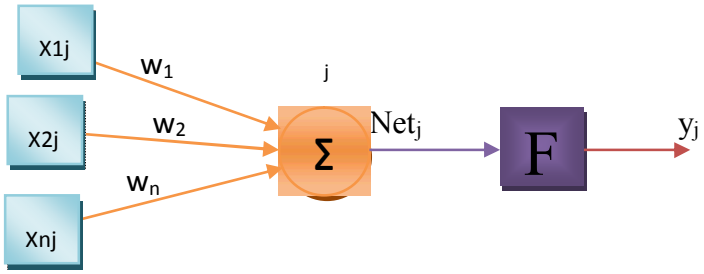


Fig.1 : schéma d'un neurone formel

- i=1……n
- X_i : entrée de neurone j
- W_i : poids associé a chaque entrée de neurone
- Net_j : potentiel de neurone
- Y_j : sortie de neurone
- F : fonction d'activation de neurone

I.4.2. Fonctionnement d'un neurone formel

On peut distinguer deux phases, la première est le calcul de potentiel du neurone qui est la somme pondérée des entrées.

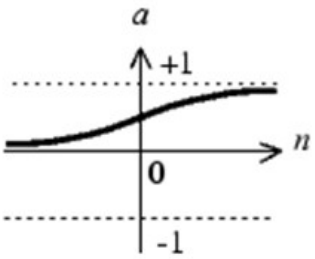
$$Net_j = \sum_{i=1}^{n} x_i w_{ij} + \theta_j$$

θ_j appelé biais c'est un poids associé a une entrée toujours égale a 1.

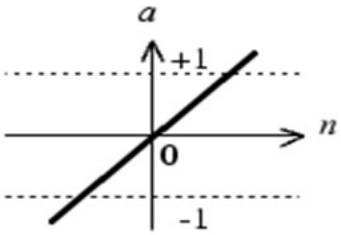
La deuxième c'est le calcul de la sortie de neurone $y_j = F(Net_j)$.

I.4.3. Différents types de fonctions d'activation.

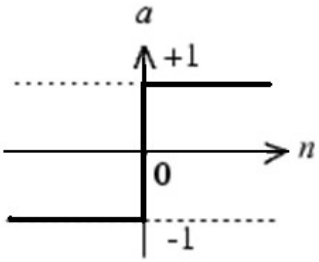
Il existe plusieurs fonctions d'activations, les plus utilisées sont



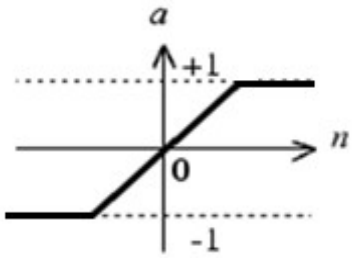
Sigmoïde



Linéaire



Fonction signe



Semi linéaire

I.5. Les réseaux de neurones

I.5. 1. Définition

Un réseau de neurones est simplement un ensemble de neurones liés entre eux de sorte que les signaux sortants des neurones deviennent des signaux entrants dans d'autres neurones, généralement organisés en couches ; ils sont des systèmes apprenants a réaliser des fonctions de mise en correspondance entre deux espèces qui sont l'entrée et la sortie du réseau.

I.5.2. Architectures de réseaux

On distingue deux types de réseau de neurones

I.5.2. a. Les réseaux de neurones non bouclés (feed-forward networks)

Un réseau de neurone non bouclé est représenté graphiquement par un ensemble de neurones connectés entre eux dont l'information circule des entrées vers les sorties sans retour arrière ; c'est-à-dire que si on se déplace dans le réseau partir d'un neurone quelconque en suivant les connexions on ne peut pas revenir au neurone de départ (le graphe des connexions est acyclique).

Les neurones qui effectuent le dernier calcul sont les neurones de sorties : ceux qui effectuent les calculs intermédiaires sont les neurones cachés.Ces réseaux sont généralement utilisés pour effectuer des taches d'approximations de fonctions non-linéaires, de classification ou de modélisation de processus statique non-linéaire.

I.5.2.b. Les réseaux de neurones bouclés (récurrents)

Les réseaux de neurones récurrents sont des réseaux ou la sortie d'un neurone du réseau peut être fonction d'elle-même c'est-à-dire lorsque on se déplace dans le réseau en suivant les connexions, il est possible de trouver au moins un chemin qui revient a son point de départ (un tel chemin est désigné sous le terme de (cycle)). Ainsi a chaque connexion d'un réseau de neurones bouclé est attaché autre un poids comme pour les réseaux non bouclés un retard car une grandeur a un instant donné ne peut pas être fonction de sa propre valeur au même instant (tout cycle du graphe du réseau doit avoir un retard non nul).

Les réseaux de neurones bouclés sont utilisés pour effectuer des taches de modélisation des systèmes dynamiques, de commande des processus et de filtrage.

I.5.3. Les différents modèles de réseaux de neurones

Il existe plusieurs modèles de réseaux de neurones

I.5.3. a. Le perceptron (réseau monocouche)

C'est un réseau simple, il se compose d''une couche d'entrée et d'une couche de sortie.

Il possède n informations en entrée, composé de p neurones, que l'on représente généralement alignés verticalement ; chacun peut en théorie avoir une fonction d'activation différente, en pratique ce n'est généralement pas le cas.

Chacun des p neurones est connecté au n information d'entrée.

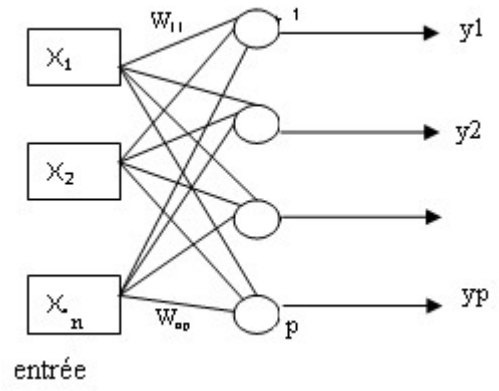


fig.2 : structure d'un perceptron monocouche

- x_i : (i=1…….n) information d'entrée
- w_{ij} : le poids reliant l'information x_i et le neurone j (j=1….p).
- y_j : sortie de j^{emme} neurone
- $Yj = F\left(\sum_{i=1}^{n} x_i w_{ij} + \theta_j\right)$
- θ_j le biais associé au neurone j

I.5.3. b. L'Adaline (adaptative lineaire élément)

Est un modèle de réseau monocouche proposé par windrow-hoff en 1960 dans un premier temps pour réaliser une fonction de classification .L'adaline est devenu un élément universel des réseaux de neurones il définit les concepts de mémorisation et d'adaptation.

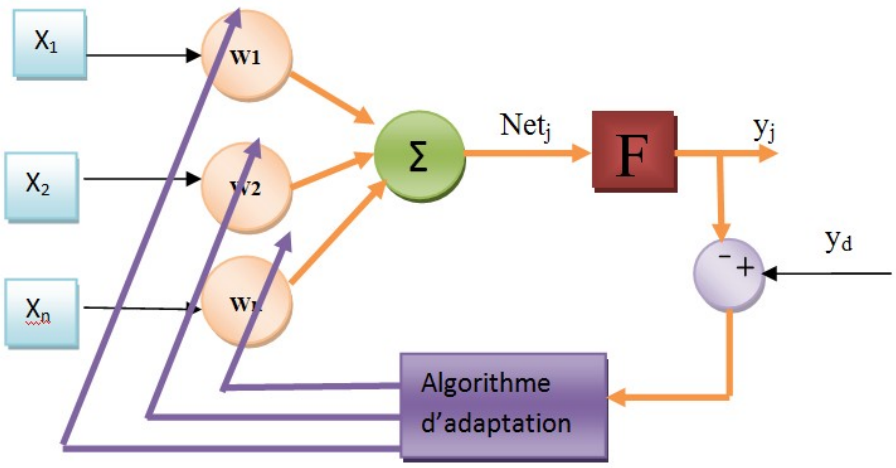


fig.2 : structure d'un Adaline

I.5.3. c. Le perceptron multicouche MLP (Multi layer perceptron)

C'est un réseau constitué d'une couche d'entrée, d'une couche de sortie et d'une ou plusieurs couche intermédiaires dites cachés

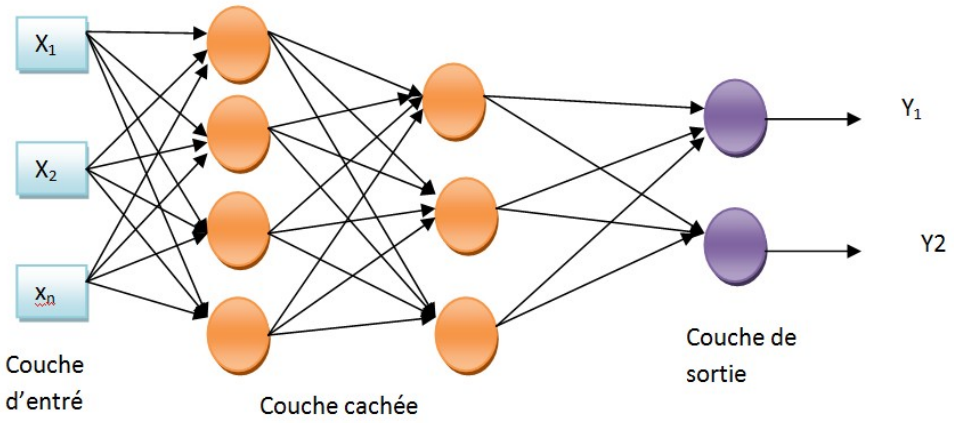


Fig.4 : architecture d'un MLP non récurent

I.5.3. d. Le réseau de Jordan

C'est l'un des premiers réseaux récurrents proposé par Jordan en 1986. La sortie du réseau est injectée dans la couche cachée en utilisant des cellules de retard, il est a signaler que les connexions entre les cellules de retard et de la sortie du réseau ont un poids égal a 1.

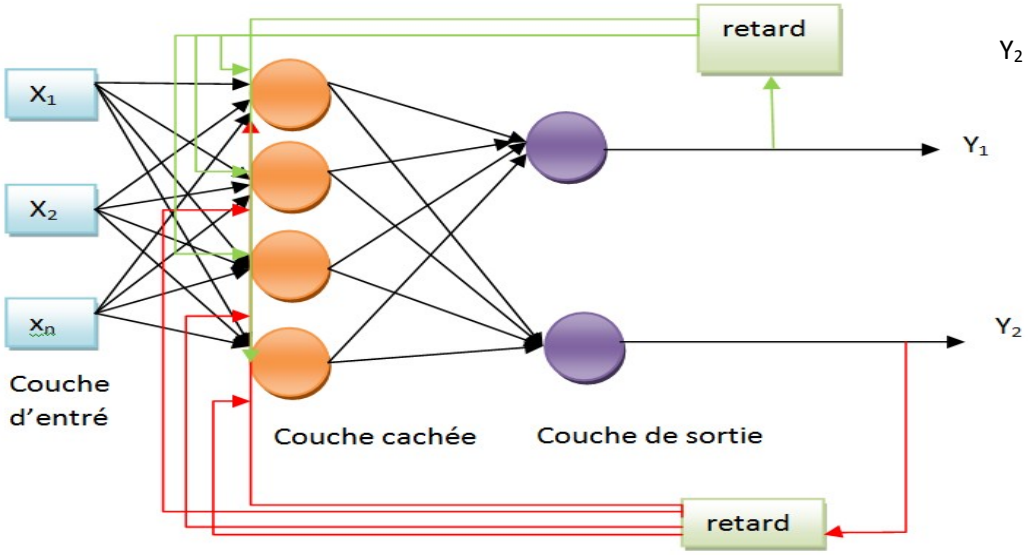


Fig.5 : réseau récurent de Jordan

I.5.3. e. Le réseau d'Elman

C'est un réseau multicouches récurrent proposé par Elman en 1990, il est similaire au réseau de Jordan sauf que c'est les sorties des neurones de la couche cachée qui sont injectées

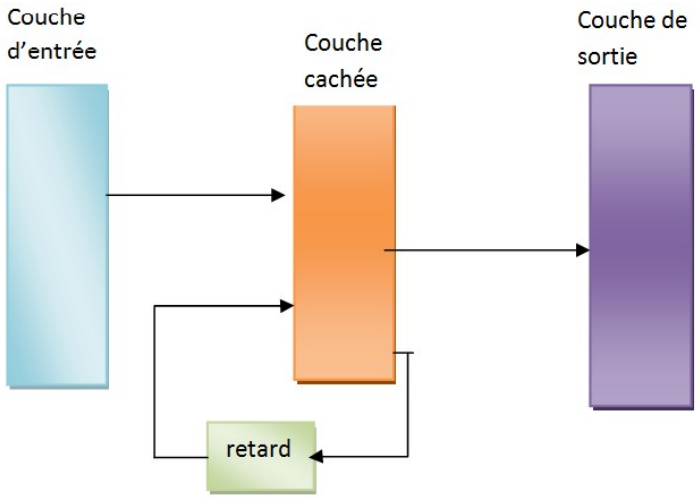


Fig.6 : réseau récurrent d'Elman

I.5.3. f. Le réseau de Hopfield

C'est un réseau totalement connecté, proposé par Hopfield en 1982 et il fonctionne d'une manière asynchrone.

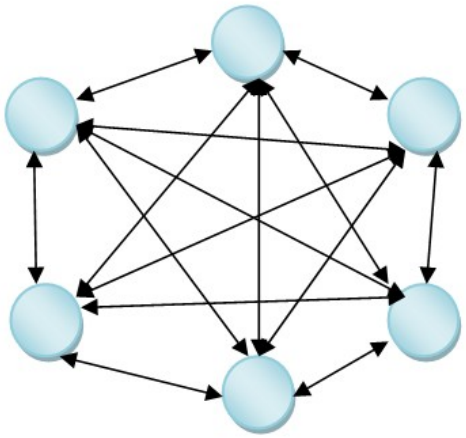


Fig.7 : réseau de Hopfield

I.5.3. g. Le réseau Madaline (Multi-Adaline)

Il est constitué d'un ensemble d'éléments Adaline connectés en parallèle ayant le même vecteur d'entrée et des sorties indépendantes qui constituent les composantes du vecteur de sorties du réseau Madaline.

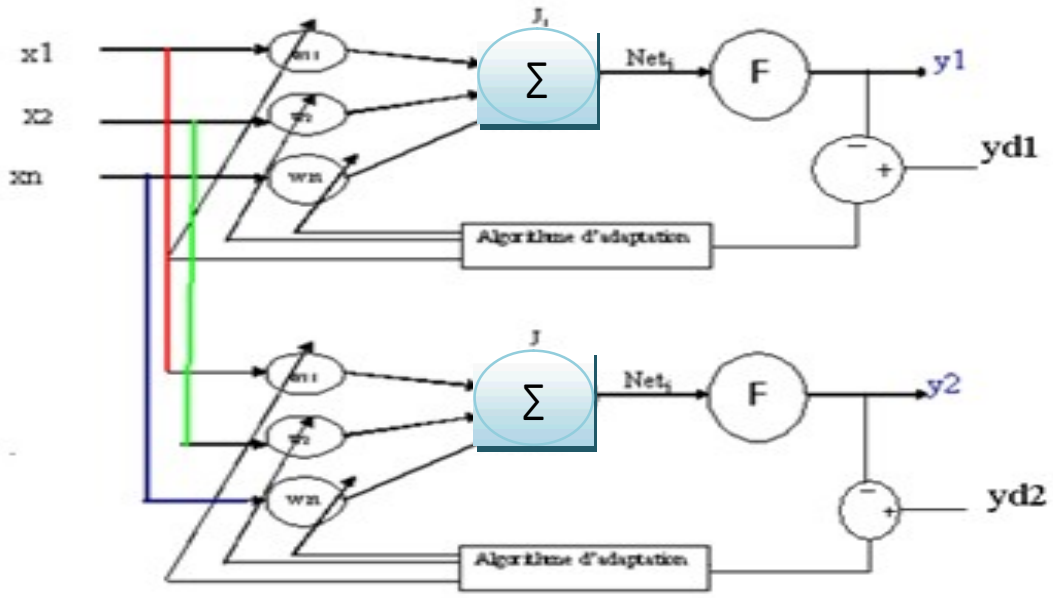


Fig.8 : réseau Madaline a 3 entrées et deux sorties

I.5.3. h. La carte auto-organisatrice de Kohonen

Kohonen s'y inspiré de la topologie de certaines zones du cortex qui présentent la même organisation que les capteurs sensoriels.Des zones proches correspondent a des neurones proches. Tous les neurones sont interconnectés mais seuls les neurones proches ont de l'influence, les neurones les plus proche sont excitateurs, ceux un peu plus éloignés inhibiteurs et les plus éloignés n'ont aucune influence .La fonction d'activation est de type sigmoïde et le réseau présente la particularité de n'avoir qu'un seul neurone actif à la fois il réalise une tache de classification. Des entrées aux caractéristiques proches donnent les mêmes sorties.

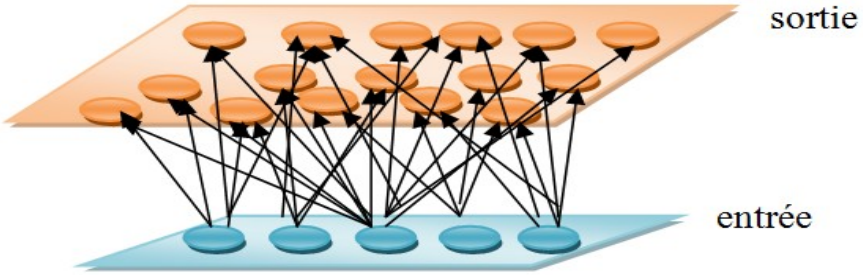


fig.8 : carte auto-organisatrice de Kohonen

I.6. Apprentissage

Dans les années **60**, on ne savait pas réaliser correctement un apprentissage sur un réseau à rebouclages ou sur un réseau à plusieurs couches .Cela mit en sommeil les recherches sur les réseaux neuronaux pendent plusieurs années .Ce n'est qu'au début des années **80** que de nouveaux travaux ont conduit à des méthodes d'apprentissages autorisant l'utilisation de réseaux neuronaux complexes

L'apprentissage est la propriété principale des réseaux de neurones ; c'est la procédure qui consiste à estimer les paramètres des neurones du réseau afin que celui-ci remplisse au mieux la tache qui lui est affectée. Au cours d'apprentissage, le réseau corrige sa structure en augmentant ou en diminuant ou en annulant les valeurs des connexions entres ses neurones. On distingue trois classes d'apprentissage.

I.6. 1. L'apprentissage supervisé

Ce mode dispose d'un comportement de référence, vers lequel il tente à converger la sortie du réseau .Durant une première phase, appelée phase d'apprentissage, on présente au réseau les valeurs d'entrée et on calcule sa sortie actuelle correspondante, ensuite les poids sont ajustés de façon à réduire l'écart entre la réponse désirée et celle du réseau en utilisant l'erreur de sortie qui est la différence entre la réponse du réseau et celle désirée. Pendant une seconde phase, appelée phase de test, on présente au réseau de nouveaux vecteurs qui n'ont pas servi à l'apprentissage et l'on observe ses réponses. Les performances du réseau sont alors évaluées par le pourcentage d'erreurs. Cette approche n'est donc possible que quand la conduite du système est connue a priori ou fixé par un expert.

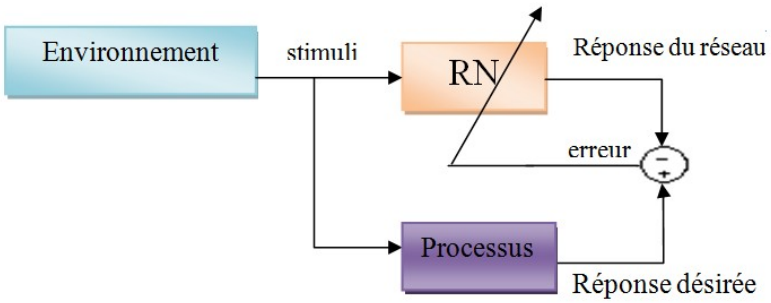


Fig.9 : apprentissage supervisé

Pour l'ajustement des paramètres plusieurs algorithmes sont proposés

I.6. 2. L'apprentissage non supervisé

Ce type d'apprentissage se fait sans la présence d'un expert c'est-a-dire pas d'informations sur la sortie désirée mais seulement des informations sur l'entrée du réseau. L'apprentissage repose sur un critère de conformité du comportement du réseau par rapport a des spécifications générales et non sur des observations externes.

Les neurones de ce réseau sont en compétition pour êtres actifs, alors que dans les autres types de réseau plusieurs sorties de neurones peuvent êtres actives simultanément ; le neurone actif est appelé neurone gagnant.

Les réseaux a apprentissage non supervisé les plus étudiés et utilisés sont "les cartes auto-organisatrices" ou "cartes de Kohonen.

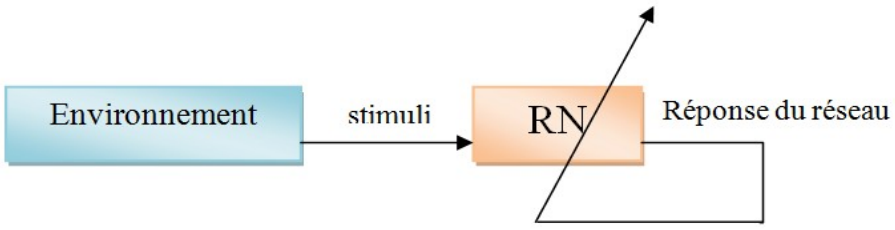


Fig.10 : apprentissage non supervisé

I.6. 3. Apprentissage par renforcement

L'apprentissage est dit guidé, effectué par renfoncement si on ne possède que des indications imprécises(on indique seulement au réseau si la sortie est correcte ou non) ; le réseau va donc tendre a maximiser un indice de performances qui lui est fourni appelé signale de renforcement .Le système étant capable ici de savoir si la réponse qu'il fournit est correcte ou non mais il ne connais pas la bonne réponse.

I.7. Les algorithmes d'apprentissages

La plupart des algorithmes d'apprentissage des réseaux de neurones formels sont des algorithmes d'optimisations, ils cherches a minimiser une fonction de cout qui constitue une mesure de l'écart entre les réponses réelles du réseau et ses réponses désirées , cette optimisation se fait d'une manière itérative, en modifiant les poids synaptiques (connexions entre neurones) et il existe plusieurs algorithmes pour faire l'apprentissage.

I.7. 1. Apprentissage par descente de gradient

C'est l'une des principales méthodes pour faire apprendre un réseau de neurones monocouche. Cette méthode consiste à comparer le résultat qui était attendu pour les exemples puis à minimiser l'erreur commise sur les exemples.

On va noter α un nombre réel auquel on donne le nom de taux d'apprentissage. C'est nous qui devons lui donner une valeur lors de la mise en pratique de l'apprentissage.

Comme nous ne considérons qu'un neurone a la fois on va noter W_i le poids reliant la i^{emme} information à notre neurone.

Les étapes de la méthode de descente de gradient

- Créer n variables w_i et leurs attribuer des valeurs aléatoires.
- Créer n variables dw_i égales a 0.
- Prendre un exemple e_k ; (k=1…N)
- Calculer la sortie obtenue avec les poids actuels, noté s_k. (1)
- Rajouter a dw_i , pour tout (i=1….n) le nombre $\alpha*(y_k - s_k)*x_i$ (2)
- Réitérer (1) et (2) sur chacun des exemples.
- Pour (i=1…n) remplacer w_i par w_i+dw_i .

Voici donc l'algorithme d'apprentissage par descente de gradient appliqué a un seul neurone, qu'il faudra donc répéter a chacun des neurones.

Entrée :

- n poids a valeurs aléatoires

-n exemples (x_k , y_k) avec x_k est un vecteur a n informations.

Sortie :

Les n poids modifiés

Pour i=1……n
  dwi=0
Fin
pour tout exemple e_k= (x_k ,y_k)
  Calculer la sortie s_k du neurone
  Pour (i=1….n)
    $dw_i=dw_i+ \alpha*(y_k - s_k)*x_i$
  Fin
Fin
Pour i=1……n
  wi= w_i+dw_i .
Fin

Afin d'obtenir des bons résultats, il faudra passer plusieurs fois les exemples a chaque neurone de sorte que les poids convergent vers des poids idéaux .le problème avec cette méthode est que l'on corrige sur la globalité des exemples ce qui fait que le réseau ne s'adapte aux exemples qu'après un certains moment. Il y a une autre méthode qui permet de corriger sur chacun des exemples et qui se nomme méthode d'apprentissage de Windrow-Hoff.

I.7. 2. Algorithme de Windrow-Hoff

L'algorithme de Windrow-Hoff ou la règle de dalta n'est en fait qu'une variante de l'algorithme précédent.

Comme nous l'avons vu dans l'algorithme précédent on engrange les erreurs commises sur chaque exemple puis pour terminer on corrige le poids et ce pour chaque poids. Cette méthode met beaucoup de temps avant de tendre vers le bon coefficient et elle ne corrige qu'un petit peu alors qu'elle pourrait corriger beaucoup mieux pour chaque exemple et c'est la que l'algorithme de Windrow-Hoff intervient, cette méthode consiste à modifier les poids après chaque exemple et non pas après que tous les exemples aient défilés, ceci va donc

minimiser l'erreur de manière précise et ce sur chaque exemple. Voici donc l'algorithme de Windrow-Hoff.

pour tout exemple $e_k = (x_k, y_k)$
  calculer la sortie s_k du neurone
  pour (i=1….n)
    $w_i = w_i + \alpha * (y_k - s_k) * x_i$
  Fin
Fin

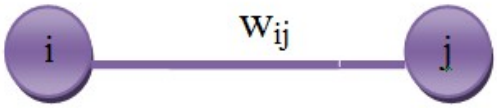
Appliquer plusieurs fois cet algorithme permettra la correction d'erreur et d'obtenir un réseau de neurones de plus en plus performant .Toute fois l'appliquer un trop grand nombre de fois mènera a ce que l'on appelle l'overfitting ou sur-apprentissage c'est-à-dire que le réseau devient très performant pour les exemples d'apprentissage mais parvient peu ou ne parvient pas a généraliser pour des informations quelconques.

Cet algorithme est très efficace pour apprendre les réseaux Adaline et Madaline.

I.7. 3. La règle de Hebb

Elle est basée sur le principe de Hebb (1949) qui consiste à renforcer la liaison entre deux neurones s'ils sont activés simultanément et de la maintenir dans les autres cas.

Soient deux neurones i et j, w_{ij} est la connexion entre eux et Net_i ,Net_j l'activation des deux neurones respectivement.



$w_{ij}(t+1) = w_{ij}(t) + dw_{ij}$

$dw_{ij} = \alpha * Net_i * Net_j$

I.7. 4. Algorithme d'apprentissage de Kohonen

La règle d'apprentissage de la carte topologique est compétitive. A la présentation d'une entrée, un neurone sur la carte est sélectionné, il correspond au neurone le plus proche possible de cette entrée (minimisation d'une distance). Le modèle de réseau neuronal proposé par Kohonen montre des propriétés d'auto-organisation topologiques de l'espace d'entrée.

L'algorithme est présenté comme suit

1. Initialiser les poids w_{ij} à des petites valeurs aléatoires (i=1….N ; j=1…..M)
   N : classe du vecteur d'entrée
   M : classe du vecteur de sortie
2. Présenter le nouveau vecteur d'entrée
3. Calculer la distance entre le vecteur d'entrée et le vecteur des poids pour tous les neurones de sortie.

$$d_j = \frac{1}{2}\sum_{i=1}^{N}(x_i(t)-w_{ij}(t))^2$$

4. Sélectionner le neurone de sortie le plus actif j^* appelé "gagnant" qui a la distance la plus petite d_{min} tel que :
   $D_j = d_{min}$ alors $j = j^*$ et $y_j^* = 1$
   Si non $y_j = 0$ ($j \neq j^*$)
5. Ajouter les poids du gagnant
   $w_{ij}(t+1) = w_{ij}(t) + \alpha\, y_j (x_i(t) - w_{ij}(t))$
6. Tant que les performances sont insuffisantes on retourne à l'étape 2 et on sélectionne l'exemple suivant dans la base d'apprentissage. Ce processus est utilisé dans la discrimination.

I.7. 5. Algorithme de rétro-propagation du gradient de l'erreur

La rétro propagation est l'algorithme d'apprentissage supervisé pour ajuster les poids d'un réseau MLP. Dans cette méthode, de même que l'on est capable de propager un signal provenant des cellules d'entrée vers la couche de sortie. On peut en suivant le chemin inverse rétro- propager l'erreur commise en sortie vers les couches cachées, d'ou le nom rétro propagation.

La rétro- propagation consiste a rétropropaguer l'erreur commise par un neurone a ses synapse et aux neurones qui y sont reliés .Pour les réseaux de neurones on utilise habituellement la rétro-propagation du gradient de l'erreur qui consiste a corriger les erreurs selon l'importance des éléments qui ont justement participé a la réalisation de ces erreurs.

Considérons un problème d'optimisation quadratique :

$$E = \frac{1}{2}\sum_{k=1}^{M}(d_k - y_k)^2$$

On dira que la réponse est correcte a un ε (réel a choisir) pré si

$$E = \frac{1}{2}\sum_{k=1}^{M}(d_k - y_k)^2 < \varepsilon$$

Nous allons décrire ici l'algorithme général puis nous passerons ensuite a la description détaillée de la méthode

Tant que

La condition n'est pas satisfaite

Faire

Choisir un exemple d'apprentissage

Calculer la sortie du réseau

Calculer l'erreur

Si E> ξ

Corriger les poids de la couche de sortie

Corriger les poids des couches cachées

Fin si

Fin tant que

Considérons le réseau à trois couches suivant

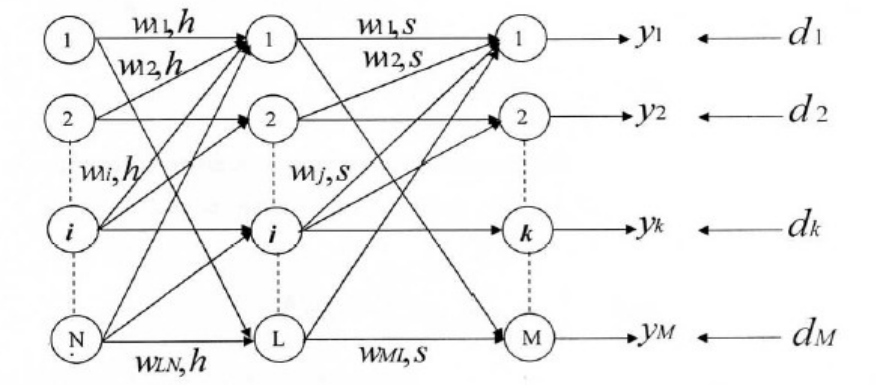


Fig.11. Réseau de neurone a trois couches

$w_{kj,s}$:poids synaptique entre la cellule j de la couche cachée h et la cellule k de la couche de sortie.

Net_j ; Net_k :c'est l'activation des neurones de la couche cachée et la couche de sortie respectivement.

$$\Delta w = -\alpha \frac{\partial E}{\partial W} \qquad \alpha > 0$$

Les étapes de l'algorithme de rétro propagation du gradient sont :

1. initialisation des poids synaptique $w_{ij,h}$,$w_{kj,s}$ a des valeurs aléatoires faibles.
2. application des vecteurs d'entrée et des vecteurs de sortie désirée.
3. calcul des activations des cellules cachées :

$$Net_j = \sum_{i=1}^{N} w_{ji,h} x_i^T$$

4. calcul des sorties des cellules cachées :

$$y_j = F_j(Net_j)$$

F_j : Fonction d'activation du neurone j de la couche cachée h

5. calcul des activations des cellules de sortie

$$Net_k = \sum_{j=1}^{k} w_{kj,s} y_j^T$$

6. calcul des sorties des cellules de sortie :

$$y_k = F_k(Net_k)$$

F_k : Fonction d'activation du neurone k de la couche de sortie.

7. calcul de l'erreur quadratique :

$$E = \frac{1}{2} \sum_{k=1}^{M} (d_k - y_k)^2$$

8. Adaptation des poids de la couche de sortie

$$\frac{\partial E}{\partial w_{kj,s}} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial w_{kj,s}} = -(d_k - y_k)\frac{\partial y_k}{\partial w_{kj,s}}$$

$$e = (d_k - y_k)$$

$$\frac{\partial E}{\partial w_{kj,s}} = -e\frac{\partial y_k}{\partial Net_k}\frac{Net_k}{\partial w_{kj,s}} = -eF'_k(Net_k)\frac{\partial Net_k}{\partial w_{kj,s}}$$

$$\frac{\partial E}{\partial w_{kj,s}} = -e\,F'_k(Net_k)\,y_j$$

$$\Delta w_{kj,s} = -\alpha\frac{\partial E}{\partial w_{kj,s}} \Rightarrow \Delta w_{kj,s} = \alpha[e\,F'_k(Net_k)]$$

L'adaptation des poids de la couche de sortie est donc donnée par

$$w_{kj,s}(t+1) = w_{kj,s}(t) + \alpha[e\,F'_k(Net_k)\,y_j]$$

9. adaptation des poids de la couche cachée :

$$\frac{\partial E}{\partial w_{ij,h}} = -\frac{1}{2}\sum_{k=1}^{M}\frac{\partial (d_k - y_k)^2}{\partial w_{ij,h}} = -\sum_{k=1}^{M}\frac{\partial y_k}{\partial w_{ij,h}}(d_k - y_k) = -\sum_{k=1}^{M} e\frac{\partial y_k}{\partial w_{ij,h}}$$

$$= -\sum_{k}^{M} e\left(\frac{\partial y_k}{\partial Net_k}\right)\left(\frac{\partial Net_k}{\partial y_j}\right)\left(\frac{\partial y_j}{\partial Net_j}\right)\left(\frac{\partial Net_j}{\partial w_{ij,h}}\right)$$

$$\frac{\partial E}{\partial w_{ij,h}} = -\sum_{k=1}^{M} e(F'_k(Net_k))\,w_{kj,s}(F'_j(Net_j))\,X_i$$

$$\frac{\partial E}{\partial w_{ij,h}} = -X_i F'_j(Net_j)\sum_{k=1}^{M} e(F'_k(Net_k))\,w_{kj,s}$$

$$\Delta w_{ji,h} = \alpha\left[X_i F'_j(Net_j)\sum_{k=1}^{M} e(F'_k(Net_k))\,w_{kj,s}\right]$$

L'adaptation des poids de la couche cachée est donc donnée par :

$$w_{kj,s}(t+1) = w_{kj,s}(t) + \alpha [Xi \; F'_j(Net_j) \sum_{K=1}^{M} e \; F'_k(Net_k) \; w_{kj,s}]$$

Remarque :

Le pas d'apprentissage vari entre 0 et 1, s'il est proche de 1 l'algorithme risque de diverger et s'il est proche de 0 la vitesse de convergence de l'algorithme est très faible.

I.8. L'identification neuronale

Certains réseau de neurones sont des approximateurs parcimonieux, Le plus souvent, le problème qui se pose à l'ingénieur est le suivant : il dispose d'un ensemble de variables mesurées (x), et d'un ensemble de mesures (z) d'une grandeur relative à un processus de nature quelconque (physique, chimique, économique ....) Il suppose qu'il existe une relation entre le vecteur des variables (x) et la grandeur (z), et il Cherche à déterminer une forme mathématique de cette relation, valable dans le domaine ou les mesures ont été effectuées. Sachant que toutes les variables qui déterminent (z) ne sont pas forcément mesurées .En d'autre termes , l'ingénieur cherche à établir un « model » du processus qu'il étudie , à partir des mesures dont il dispose , et d'elles seules : on dit qu'il effectue une modélisation « boite noire » .L'identification de processus constitue une des applications importantes des réseaux de neurones, elle offre des avantages très intéressant lorsqu'il s'agit d'approximation des fonctions a base d'exemples, comme on peut bénéficier de ces avantages dans la commande des processus. Le synoptique d'une identification neuronale est donnée sur la figure suivant

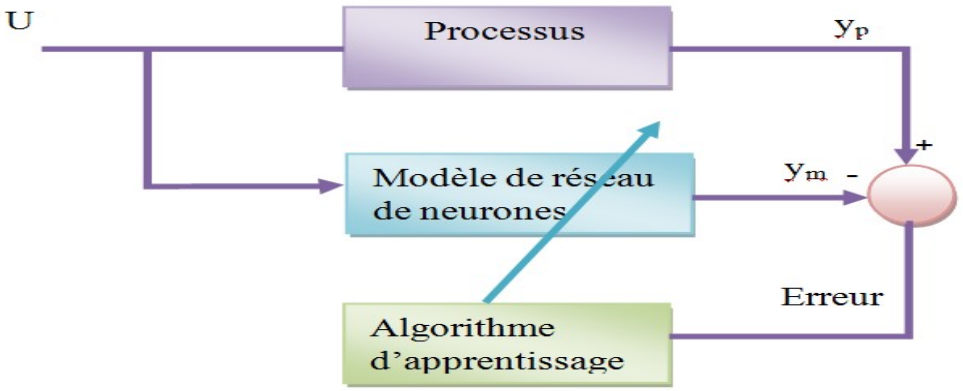


Fig.12. Synoptique d'une identification neuronale

L'identification neuronale consiste à faire apprendre au réseau modélisé le comportement du processus méconnu ou partiellement connu.

La connaissance des paramètres du processus ne peut contribuer qu'au bon choix de la structure et de la topologie du réseau à même de mimer le comportement du système modélisé.

I.8.1. Les étapes d'une modélisation neuronale : sont les suivantes

1. Déterminer les entrées pertinentes
2. Collecter les données nécessaires à l'apprentissage et l'évolution des performances du réseau de neurones.
3. Trouver le nombre de neurones cachés nécessaires pour obtenir une approximation satisfaisante.
4. Estimer les valeurs des paramètres correspondant a un minimum de la fonction de coût c'est-a-dire d'effectuer un apprentissage.
5. évaluer les performances du réseau de neurones a l'issue de l'apprentissage (validation du modèle ou généralisation).

I.9. Commande neuronale

Les réseaux de neurones artificiels sont des outils permettant de représenter les relations fonctionnelles complexes nécessitées par les systèmes de régulation modernes.

I.9. 1. Les différents types de la commande neuronale

On distingue plusieurs architectures de commande par réseaux de neurones et voici quatre principales

- Par imitation : consiste à imiter un système de commande déjà existant.
- Par apprentissage généralisé
- Par apprentissage spécialisé
- Commande prédictive

I.10. conclusion

Dans ce chapitre, nous avons exposé les éléments essentiels qui permettent de comprendre pourquoi, et dans quels cas, il est avantageux de mettre en œuvre des réseaux de neurones .et comme suit les points fondamentaux qu'il convient de toujours garder à l'esprit lorsque on cherche à mettre en œuvre des réseaux de neurones :

- Les réseaux de neurones sont utilisés comme outils statistiques, qui permettent d'ajuster des fonctions non linéaires très générales à des ensembles de points, l'utilisation de réseaux de neurones nécessite que l'on dispose de données suffisamment nombreuses
- Les réseaux de neurones à apprentissage supervisé sont des approximateurs parcimonieux, qui permettent de modéliser des phénomènes.
- Les réseaux de neurones à apprentissage non supervisé peuvent apporter des éléments précieux pour la détermination d'une bonne représentation des formes.

Il est toujours souhaitable, et souvent possible d'utiliser, pour la conception du réseau, les connaissances mathématiques dont on dispose sur le phénomène à modéliser ; les réseaux de neurones ne sont pas nécessairement des « boites noires ».

L'entraînement et la conception d'un réseau de neurones demeure aujourd'hui un art et non une science, à la fois objets d'études et outils applicatifs.les réseaux de neurones ont un rôle à

jouer dans un nombre rapidement croissant de domaines autant en recherche qu'en industrie .Ainsi l'association ou l'hybridation des techniques neuronales avec d'autres méthodes et la connaissances disponibles sur le comportement de système à traiter , pourrait améliorer les performances du réseau et permet aussi de réaliser efficacement les tâches désirées .Parmi ces méthodes on trouve la logique floue , de fait de son caractères approximatif et qualitatif inspirés de la pensé humaine .

II.1. Introduction

La logique floue à connue un intérêt important dans la communauté scientifique au cours des dernières années l'une des raisons principales est l'énorme succès des équipements domestiques produits par l'industrie japonaise utilisant des régulateurs flous.

Les systèmes flous appartiennent à la classe des systèmes à base de connaissances, leurs but principal consiste à implémenter un savoir faire humain sous la forme d'un programme informatique**.**

La logique floue fournit un formalisme mathématique pour réaliser ce but.
Les régulateurs flous modélisent l'expérience humaine sous la forme de règles linguistiques.
Si.....alors.....

Les algorithmes basés sur la logique floue sont considérés comme une solution très intéressante pour le réglage des systèmes non linéaires ou les systèmes pour lesquels il n'existe pas de modèles mathématiques.

II.2. Historique

Les racines de la logique floue se trouvent dans le principe d'incertitude de Heisenberg dans les années 20 ou il a introduit la troisième valeur 1/2 dans le système logique bivalent {0 , 1}.

Au début des années 30, le logiciens polonais Jan Lukasieurcz a développé le système logique avec 3 valeurs puis l'a étendu à tous les nombres Rationnels entre 0 et 1 ,et c'est le premier qui a définit la logique floue comme une logique qui associe à une affirmation un niveau de vérité qui peut prendre toutes les valeurs entre 0(faux) et 1(vrais) .

Dans les années 30 aussi, Max black applique la logique floue aux ensembles d'éléments ou de symboles, et dessine la première fonction d'appartenance (membership function) d'un ensemble floue.

En 1965 le professeur Lotfi Zadeh a publié l'article "fuzzy sets" dans le quel il a développé la théorie des ensembles floues et introduit le terme **"fuzzy "** dans la littérature technique, les premières investigations de zadeh étaies l'utilisation de la logique floue pour représenter le langage naturel, il a était le premier à formaliser les règles floue (fuzzy rules) et à suggérer une approche **"système- expert"** au réglage automatique.

Les premiers résultats en commande floue ont étaient publiés par Mandani et Assilian en 1975

ce qui a encouragé différentes activités en Angleterre, au Danemark et en France.

Après 1980,les recherches s'arrêtent en Europe mais les japonais les reprennent ,leurs industrie a lancé de nombreux produits basés sur la logique floue ; plus de 2000.

Actuellement la logique floue est considérée comme un outil de base au Japon et les principales directions de recherche sont la combinaison de la logique floue, des algorithmes génétiques et des réseaux de neurones.

II.3. Domaines d'application de la logique floue

- la commande
- traitement d'image et reconnaissance des formes
- la modélisation

II.4. Définitions

II.4.1. La logique floue

C'est une logique qui associe à une affirmation un niveau de vérité qui peut prendre toutes les valeurs entre 0(faux) et 1(vrais).

La conception d'un système basé sur la logique floue commence par le choix de variables linguistiques (entrées et sorties du contrôleur), le pas suivant consiste à créer l'univers du discours pour chaque variable linguistique, c'est -à -dire un ensemble de valeurs linguistiques que la variable peut prendre. Ces valeurs sont représentées avec des ensembles qui peuvent être discrets ou continus.

II.4.2. Variable floue

Une variable linguistique (floue) représente un état dans le système à régler ou une variable de réglage dans un contrôleur flou. Sa valeur est définie dans des termes linguistiques qui peuvent être des mots ou des phrases d'un langage naturel.

Chaque variable floue est caractérisée par un ensemble tel que {x, T(x), G, M, U}

- X : est le nom de la variable.
- T(x) : est l'ensemble des valeurs linguistiques que peut prendre x.
- U : c'est l'univers du discours associé avec la variable de base.
- G : est la règle syntaxique pour générer les valeurs linguistiques de x.
- M **:** est la règle sémantique pour associer un sens à chaque valeur linguistique.

Exemple

La variable floue x → température ambiante.

Elle peut être définie par un ensemble de termes linguistiques :

T(x) → {extrêmement froide, très froide, froide, chaude, très chaude, excrément chaude}.

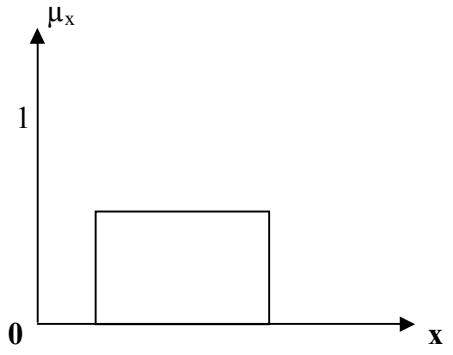
son univers de discours

U = [-20°C, 40°C]

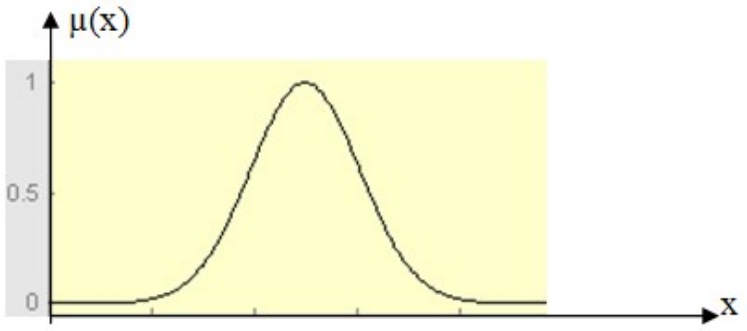
La variable de base est température, le terme froide représente une valeur linguistique qu'on peut interpréter, par exemple comme "les températures plus petites du 15°C"

II.4.3. Fonction d'appartenance

L'ensemble des valeurs linguistiques que peut prendre la variable floue sont représentées dans la logique floue par des fonctions dites d'appartenances (mumberships functions) et chaque élément de cet ensemble a une valeur d'appartenance qui est le degré de compatibilité de cet élément avec le concept qui est représenté par l'ensemble flou. Et cette valeur varie entre 0 et 1 contrairement au ensemble classique ou binaire qui ont deux valeurs seulement **0**(faux) ou **1**(vrais).



Ensemble classique



Ensemble flou

II.4.3. a. les formes de la fonction d'appartenance

Les fonctions d'appartenances peuvent avoir différentes formes ; elles peuvent êtres.

- monotones décroissantes

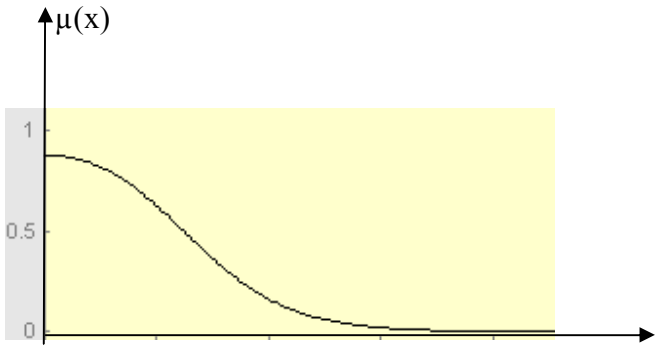


fig.1.1 : fonction d'appartenance décroissante

- monotones croissantes

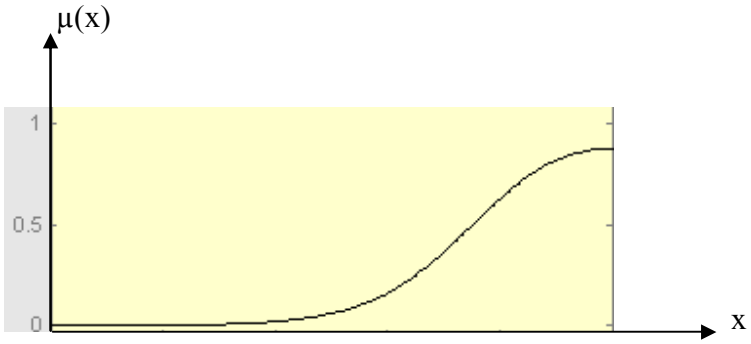


Fig.1.2 : fonction d'appartenance croissante

- triangulaires

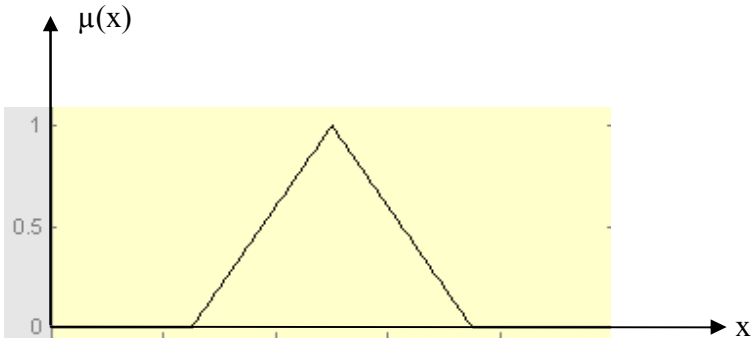


Fig.1.3 : fonction d'appartenance triangulaire

- trapézoïdales

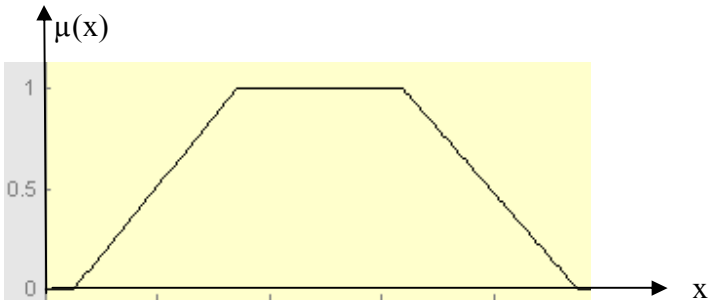


Fig.1.4 : fonction d'appartenance trapézoïdale

- en forme de cloches (gaussiennes)

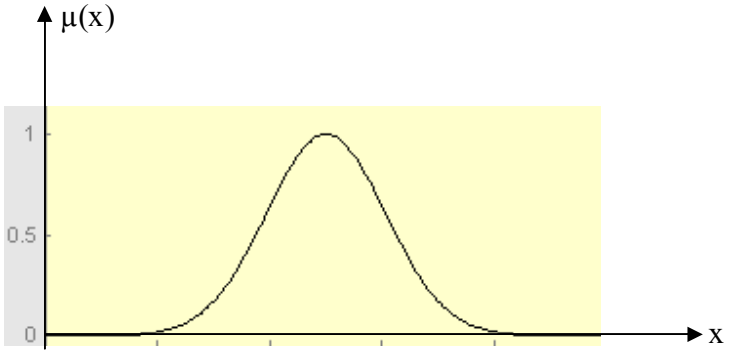


Fig.1.5 : fonction d'appartenance gaussienne

Il est à signaler que le choix de la forme de la fonction d'appartenance dépend de l'application et

des connaissances de l'opérateur.

II.4.3. b. Univers de discours

Un des premiers pas dans la conception d'une application floue est de définir l'ensemble de référence ou univers du discours pour chaque variable linguistique.

L'ensemble de termes **T** (température ambiante) de l'exemple précédent peut être caractérisé par un ensemble de sous-ensembles flous dont les fonctions d'appartenances sont montrées à la figue 2.

Chaque sous-ensemble flou dans l'univers de discours représente une valeur linguistique.

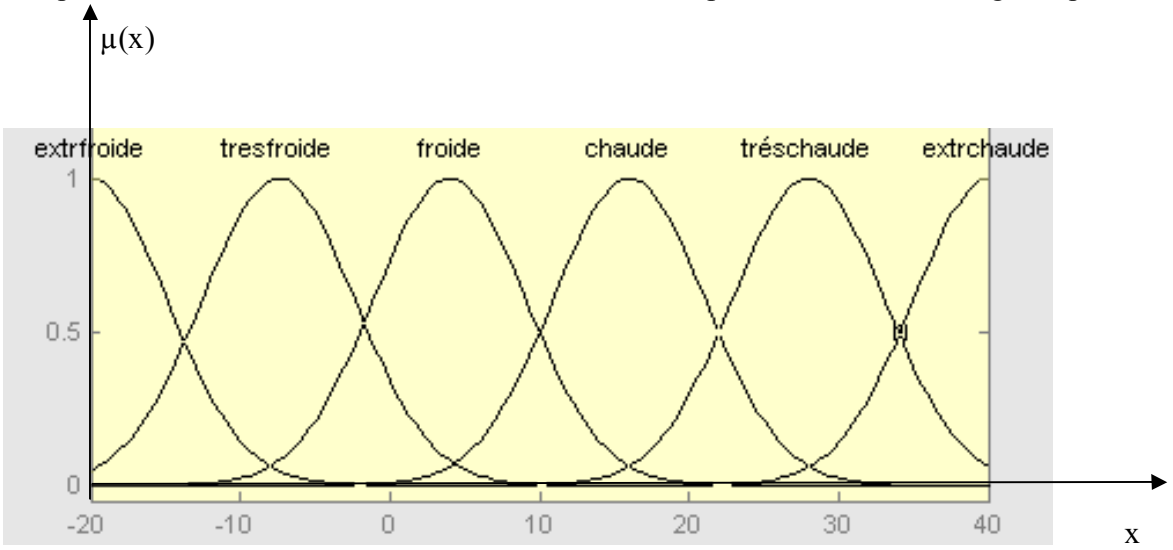


Fig.2 : les valeurs de la variable floue x

II.4.4 Les règles linguistiques

L'idée principale des systèmes basés sur la logique floue est d'exprimer la connaissance humaine sous la forme de règles linguistiques de forme Si**……**alors**……**

Chaque règle a deux parties

- partie antécédente (prémisse ou condition), exprimée par Si
- partie conséquente (conclusion) exprimée par alors

La partie antécédente est la description de l'état du système, la partie conséquente exprime l'action que l'opérateur du système doit exécuter.

Chaque règle floue est basée sur l'implication floue

Il y a plusieurs formes de règles **Si.....alors……**

La forme générale est

-**Si** (un ensemble de conditions est satisfaites) **alors** (un ensemble de conséquences peut être exécutées).

-Zadeh a été le premier à introduire la nation de règle floue sous la forme

-Si x est A alors Y est B

Exemple

Supposons qu'une règle linguistique de contrôle d'un système est

-Si la température est grande alors la pression doit être petite

Il est évident que la forme de cette règle est

Si x est A alors Y est B

Ou la température (x) et la pression (Y) sont les variables linguistique d'entée et de sortie respectivement

La partie antécédente : si la température est grande représente l'état du système à régler et la partie conséquente exprime l'action à exécuter par l'operateur (dans ce cas il va diminuer la pression) (A) et (B) sont les valeurs linguistique caractérisées par des fonctions d'appartenances dans les univers du discours de variables x et Y respectivement.

II.4.4.1 Les opérateurs flous

Les règles d'inférences font appel aux operateurs (et, ou , non) qui s'appliquent aux variables floues. Et voila les operateurs utilisés

Operateur	Opération sur le degré de vérité de variables
Et	Minimum
	Produit
ou	Maximum
	Valeur moyenne
non	Complément à 1

Les operateurs minimum et maximum présentent l'avantage de la simplicité lors du calcul par contre ils privilégient l'une des deux variables.

Les opérateurs, produit et valeur moyenne sont plus complexes à calculer, mais ils produisent un

résultat qui tient compte des valeurs des deux variables.

Exemple

Prenons deux propriétés floues : " jeune " et vieux" dans l'ensemble de référence de l'âge humain

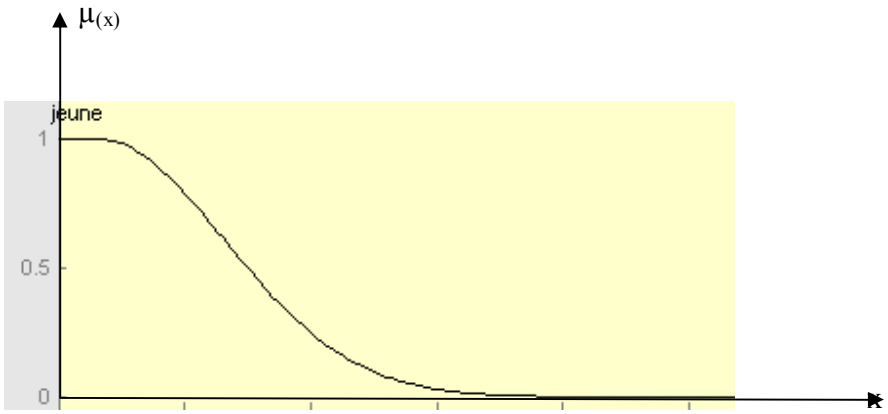


Fig.3.1 : fonction d'appartenance de la valeur jeune

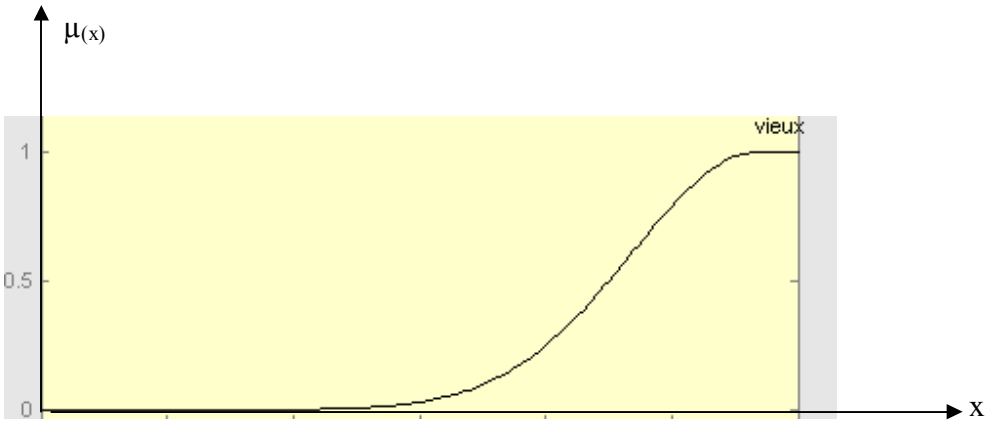


Fig.3.2 : fonction d'appartenance de la valeur vieux

II.4.4.1.a. L'opérateur NON

La propriété « cette personne n'est pas jeune » peut être caractérisée de façon évidente par la fonction d'appartenance $\mu_c = 1-\mu_g$.

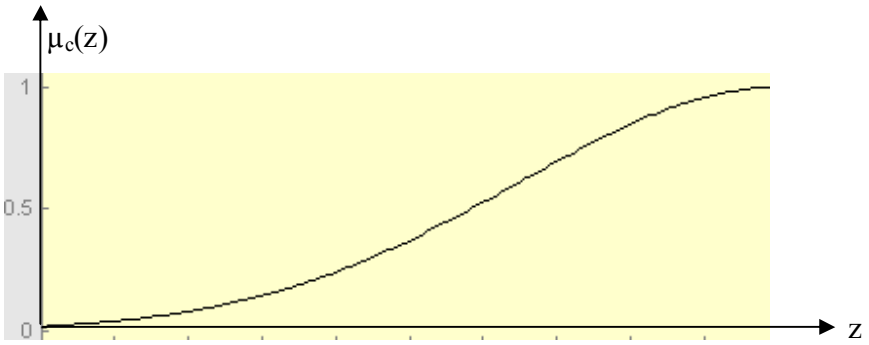


Fig.3.3 : opérateur NON

II.4.4.1.b. L'opérateur ET

La relation la plus simple et la plus utilisée pour caractériser la satisfaction simultané de deux propriété est de poser

$\mu_E(z) = \mu_{A \text{ et } B}(z) = \min \{\mu_A(x), \mu_B(y)\}$

On parle alors d'operateur minimum cet opération est représenter a la figure 3.4.

On remarque que la fonction d'appartenance résultante $\mu_E(z)$ n'atteigne pas la valeur 1, on peut facilement vérifier que l'opérateur min est commutatif c'est-à-dire qu'il est possible d'inverser $\mu_A(x)$ et $\mu_B(y)$ sans que le résultat change**.**

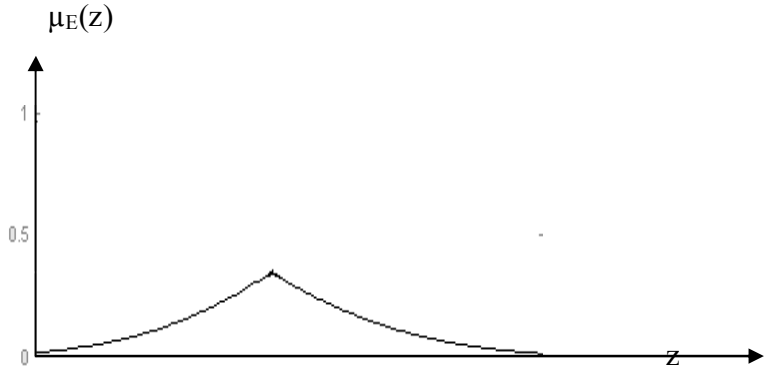


Fig.3.4 : opérateur ET ou (intersection)

II.4.4.1.c. L'opérateur OU

La réalisation de l'operateur OU se fait en général par la formation de maximum. En l'appliquant aux fonctions d'appartenance $\mu_A(x)$ et $\mu_B(y)$ des deux ensembles A et B on a donc l'operateur maximum

$\mu_O(z) = \mu_{A \text{ ou } B}(z) = \max \{\mu_A(x), \mu_B(y)\}$, la figure fig. 3.5 montre cette opération , il est a noter qu'il est possible que la fonction d'appartenance résultante $\mu_O(z)$ atteigne deux fois la valeur 1 .

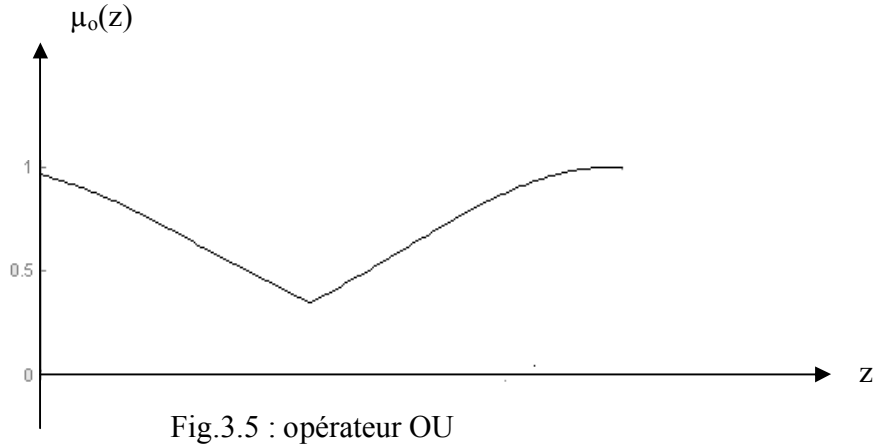


Fig.3.5 : opérateur OU

II.4.4.1.d. les opérateurs ET flou et OU flou de Werners

Les opérateurs ET flou et OU flou sont des opérateurs combinés entre l'opérateur min et max et la moyenne arithmétique. Ces opérateurs ont étés proposés par Werners .

-ET flou est définie de la manière suivante :

$\mu_{A\cap B}(z) = \alpha \min(\mu_A(x), \mu_B(y)) + (1-\alpha)/2\ [\mu_A(x) + \mu_B(y)]$

-OU flou comme suit :

$\mu_{AUB}(z) = \alpha \max(\mu_A(x), \mu_B(y)) + (1-\alpha)/2\ [\mu_A(x) + \mu_B(y)]$

il est évident que le facteur α pondère les influences des deux termes ; pour α=1 on aboutit a l'opérateur min ou max respectivement : par contre pour α=0 on obtient la moyenne arithmétique pour les deux opérateurs, dans ce cas le ET flou et le OU flou se confondent. Il est possible d'étendre ces deux opérateurs a trois ou plusieurs termes

II.5. Structure générale d'un système flou

Chaque système basé sur la logique floue est composé de quatre blocs principaux

1- Base de connaissances (règles et paramètres des fonctions d'appartenances)
2- Bloc de décision ou le moteur d'inférence des opérateurs sur les règles
3- Fuzzification (transformation des entrées précises en degrés d'appartenance)
4- Defuzzification (transformation des résultats flous en sorties précises

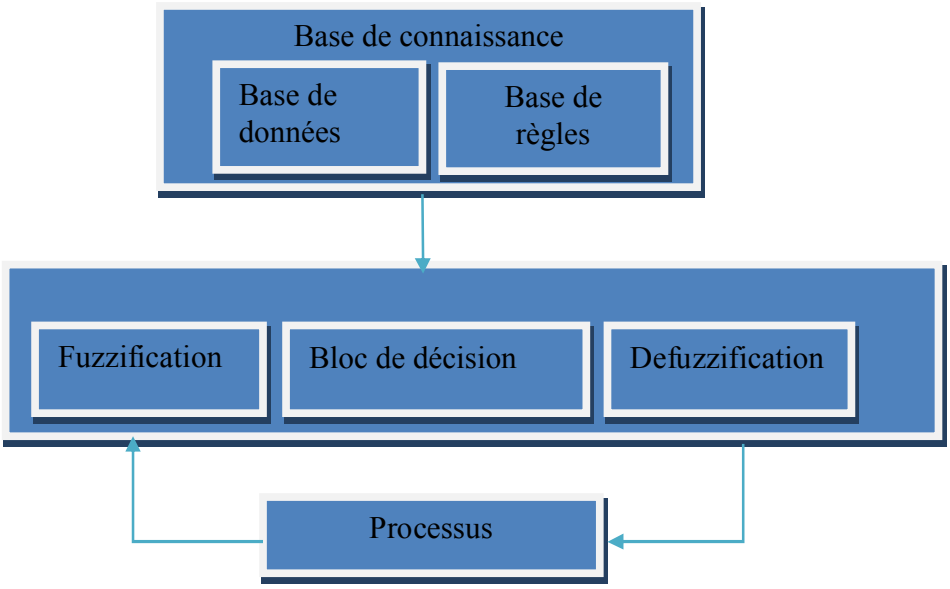


Fig.4.1 : structure générale d'un système flou

II.6.Procédure de raisonnement flou

II.6.1. Fuzzification

II.6.1.a. Définition :

la fuzzification est l'étape de passage de domaine numérique au domaine linguistique, cette étape est nécessaire dés lors que l'on veut manipuler à l'aide de la théorie des ensembles flous des grandeurs physiques mesurables.

II.6.1.b. Les étapes de la fuzzification

La Fuzzification comprend :

- La mesure des variables d'entrées.
- La conversion A/D ainsi que le traitement des grandeurs mesurées.
- L'attribution d'un ensemble de degrés d'appartenances a chaque valeur d'entrée.

Les entrées dans un régulateur flou sont en général mesurées à l'aide d'organes de mesures qui sont le plus souvent de type analogique. Étant donné que l'implémentation de régulateur flou se fait presque toujours d'une manière digitale, il faut d'abord prévoir une conversion analogique/digitale.

L'attribution de degrés d'appartenances a chaque valeur d'entrée est un passage des grandeurs physiques aux variables linguistiques qui sont définies par leurs valeurs linguistiques.

En général les fonctions d'appartenances qui représentent les valeurs linguistiques sont de forme triangulaires, trapézoïdales ou en forme de cloche.

On peut introduire pour une variable x trois ou cinq ou sept valeurs linguistiques, ce nombre dépend de la résolution du réglage désiré.

La désignation standard des ensembles flous est

NG: négative grand

NM : négative moyen

NP : négative petit

EZ : environ zéro

PP : positive petit

PM : positive moyen

PG : positive grand

Un nombre de valeurs linguistiques supérieur a sept n'apporte en général aucune amélioration de comportement dynamique du réglage flou par contre un tel choix compliquerait la formulation

des règles d'inférences.

- Les fonctions d'appartenances peuvent êtres symétriques, régulièrement distribuées ou avoir une distribution non uniforme.
- Il est important d'éviter les lacunes ou les chevauchements insuffisants de deux ensembles voisin car ceci provoque des zones de non-intervention du réglage (zone morte) ce qui conduit a une instabilité de réglage .De même il faut éviter un chevauchement trop important car il conduit a un aplatissement des caractéristiques du régulateur.
- Il ne faut pas oublier qu'il est indispensable de créer des fonctions d'appartenance pour les variables de sortie.

D'habitude on les crée de la même manière que les fonctions d'entrées, il est aussi possible de créer les sous-ensembles comme ceux de la figure 4.2, qui désigne des valeurs précises.
Ces sous ensembles sont connues dans la littérature comme des ensembles de Sugeno.

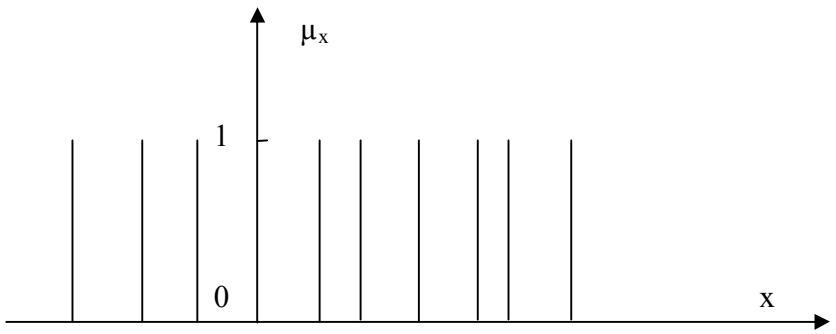


Fig.4.2 : ensemble flou de sugeno

II.6.2. Inférences

Les inférences lient les grandeurs mesurées (fuzzifiées) et les variables de sorties par des règles linguistiques, ces règles sont combinées en utilisant les connections ET et OU.

Supposons que le système flou a deux entrées convenablement transformées en variables linguistiques x et y et une sortie z et que l'on a défini n règles linguistiques comme suit.

Si $x=A_1$ et $y=B_1$ alors $z=C_1$

Si $x=A_2$ et $y=B_2$ alors $z=C_2$

Si $x=A_3$ et $y=B_3$ alors $z=C_3$

.

.

.Si $x=A_n$ et $y=B_n$ alors $z=C_n$

Ou x, y et z sont les variables linguistiques qui représentent les variables d'états du processus et les variables de contrôle.

A_i, B_i et C_i (i=1 ……..n) sont les sous-ensembles flous définis dans l'ensembles de référence pour x , y et z respectivement.

En toute généralité n'importe quelle combinaison des opérateurs ET, OU et NON peut apparaître dans la condition suivant les conditions imposées par le système a régler.

II.6.2. a. Activations de règles linguistiques

Dans le sens mathématique l'activation de règles est l'application d'une relation pour obtenir le poids d'activation de chaque règle. D'habitude on applique l'opérateur Min ou le produit Sur les valeurs d'appartenances. Le poids W_i obtenu de la i^{eme} règle est : $W_i = \mu_{Ai}(x)$ et $\mu_{Bi}(y)$

Ou $\mu_{Ai}(x)$ et $\mu_{Bi}(y)$ sont les valeurs d'appartenances de x et y respectivement aux sous-ensembles A_i et B_i. Cela veut dire que la partie conséquente de la i^{eme} règle ($z=c_i$) doit être activée avec un niveau de vérité W_i.

Une règle est activée si $W_i \neq 0$.

II.6.2. b. Types d'inférences flous

Afin de mettre en évidence le traitement numérique on va considérer un système de réglage basé sur la logique floue qui a deux entrées et une sortie.

Le chauvechement de deux fonctions d'appartenance conduit a l'activation d'une ou plusieurs règles en même temps. Supposons que les deux règles activées sont les règles suivantes :

R_1 : Si $x=A_1$ et $y=B_1$ alors $z=C_1$

R_2 : Si $x=A_2$ et $y=B_2$ alors $z=C_2$

Il y a plusieurs sortes d'inférences floues et il ne faut pas s'attacher trop a leurs noms car ils varient d'un auteur à un autre. Les principes de raisonnement pour quartes méthodes sont

considérés par la suite.

II.6.2. b.1.la **méthode Max-prod**

Cette méthode utilise les représentations standards pour les sous-ensembles d'entées et de sorties. Le poids d'activation d'une règle est utilisé pour multiplier la fonction d'appartenance de sous-ensemble de sortie imposé par cette règle. L'action globale ou la valeur de commande est l'union des actions produites par chaque sous-ensemble individuellement .Cette méthode est graphiquement expliquée a la figure 4.3.a

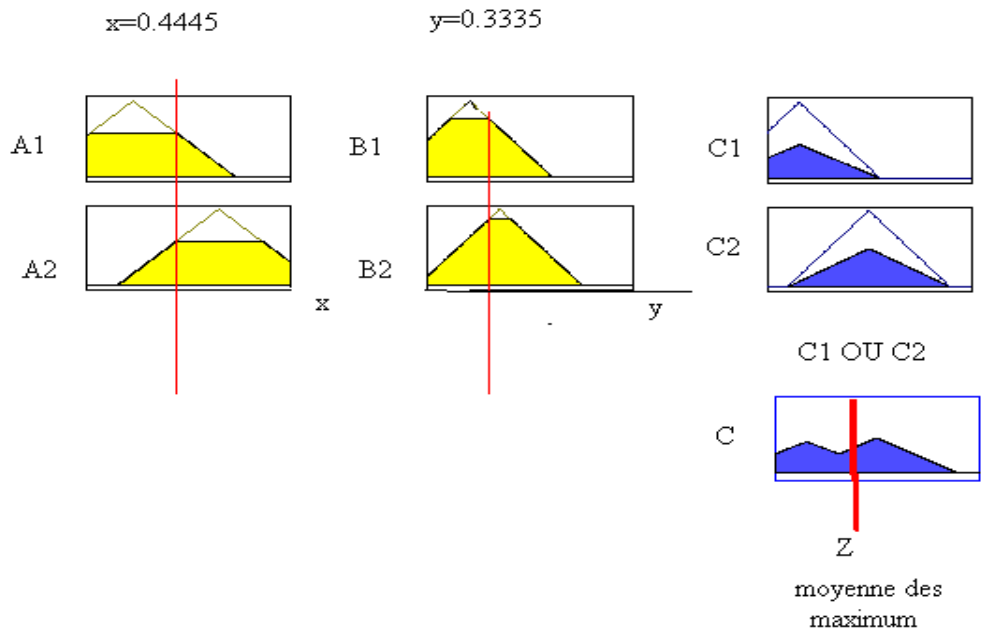


fig.4.3.a : méthode de Max-prod

II.6.2. b.2. Min –Max méthode

Cette méthode est la plus mentionnée dans la littérature sur les régulateurs flous. Elle utilise les même descriptions pour les sous- ensembles de sorties que pour les sous ensembles d'entrées à la condition de chaque règle R_i est attribuer un poids d'activation W_i qui dépend de la conditions elle-même et des valeurs d'entrées .Pour l'opération ET, on utilise l'opérateur min le poids d'activation est utilisé comme la constante d'écrêtage pour le sous-ensemble de sortie imposée par la partie conséquente de la règle R_i , la réunion des sous-ensembles écrêtés forme le sous-ensemble de sortie figure..4.3.b.

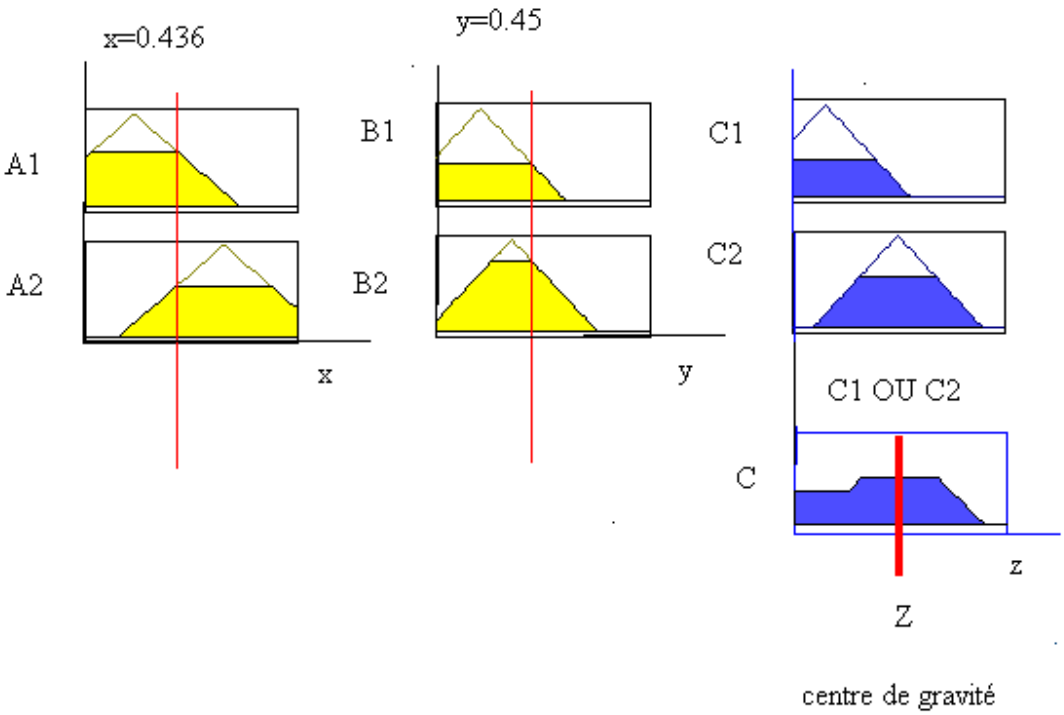


fig.4.3. b : méthode de Min-Max

La valeur précise de commande est donnée d'habitude par la méthode de centre de gravité qui sera expliqué dans la partie défuzzification.

II.6.2. b.3. Méthode de Tsukamonto

Les fonctions d'appartenance de sorties doivent êtres monotones est non décroissantes .la sortie de chaque règle est une valeur de la fonction du poids d'activation.la valeur de commande est la moyenne pondérée des poids d'activation et des sorties de fonction d'appartenances.

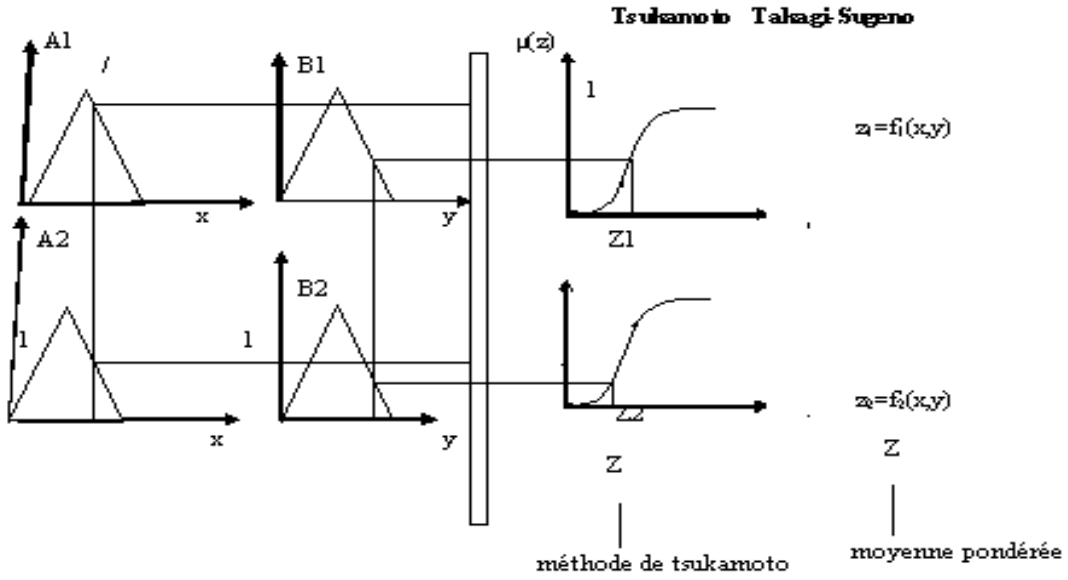


Fig.4.3.c : méthode de Tsukamoto et méthode de Takagi et Sugeno

II.6.2. b.4. Méthode de Takagi et Sugeno

Chaque fonction d'appartenance de la sortie est une combinaison linéaire des valeurs d'entrées. La sortie précise est la moyenne pondérée des poids d'activations et des sorties des fonctions d'appartenances.

La méthode de Sugeno ou les fonctions d'appartenances sont des valeurs précises constituent un cas particulier de cette méthode.

Les sous-ensembles flous qui forment les valeurs précises sont montrés sur la figure 4.2

II.6.3. Defuzzification

Le résultat d'une inférence floue est une fonction d'appartenance il est un sous-ensemble flou. Un organe de commande nécessite un signal de commande précis.

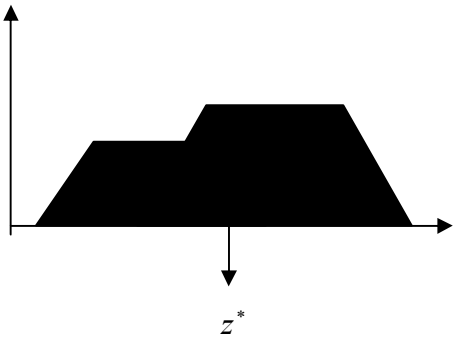
La transformation d'une information floue en une information déterminée est la defuzzification (quancritisation) .De plus on dois prévoir un traitement de signal et une conversion (digital/analogique).

Il y a plusieurs méthodes de defuzzification proposées dans la littérature il y a pas de stratégie systématique pour choisir parmi l'une de ces méthodes.

II.6.3. a. La méthode de centre de gravité

La méthode de centre de gravité est une des méthodes les plus mentionnées dans la littérature. L'abscisse du centre de gravité peut être déterminée en utilisant la forme générale

$$z^* = = \frac{\int_{z_0}^{z_1} \mu_z(z)\, z\, dz}{\int_{z_0}^{z_1} \mu_z(z)\, dz}$$



Lorsque la fonction μ(z) est discrétisée le centre de gravité est donné par

$$z^* = \frac{\sum_{i=1}^{n} \mu_i(z)\, z_i}{\sum_{i=1}^{n} \mu_i(z)}$$

Ou n est le nombre de niveau de quantisation, z_i est la valeur de sortie pour le niveau i et μ_i sa valeur d'appartenance.

II.6.3. b. Méthode de moyenne des maximums

Cette méthode génère une commande précise en calculant la moyenne des valeurs pour lesquelles l'appartenance est maximale

$$z^* = \sum_{i=1}^{l} \frac{r_i}{l}$$

Ou l est le nombre de valeurs quantifiées de r pour lesquelles l'appartenance est maximale.

II.6.3. c. Méthode de Tsukamoto

Si l'on utilise les fonctions d'appartenance monotones et non décroissantes pour la sortie, la valeur précise de la commande est calculée comme suite

$$z^* = \frac{\sum_{i=1}^{n} w_i f(x_i, y_i)}{\sum_{i=1}^{n} w_i}$$

Ou n est le nombre de règles activées avec le poids $w_i > 0$ et z_i la valeur de la fonction de sortie pour la règle i.

II.6.3. d. Méthode de moyenne pondérée

Cette méthode est utilisée lorsque les sorties sont définies comme fonction linéairement dépendantes d'entrées.

En général la partie conséquente de la règle est **z=f(x ,y)=ax+by+c**

Si W_i est le poids d'activation de la règle i, la valeur précise de la commande est

$$z^* = \frac{\sum_{i=1}^{n} w_i f(x_i, y_i)}{\sum_{i=1}^{n} w_i}$$

Ou n est le nombre de règles activées.

II.7.1 .réglage flou

Lors du réglage par logique flou, le régulateur subit plusieurs réglages.

Les différents paramètres du régulateur qui peuvent être réglés sont les fonctions d'appartenances des prémisses et des conclusions, choix de la méthode d'inférence et de défuzzification et le choix des facteurs d'échelle ; ces derniers jouent un rôle important dans la stabilité et l'obtention de bonnes performances.

Dans cette section, on étudiera le réglage en utilisant des facteurs d échelle d'un régulateur flou. On considère ici, le cas ou les variables d'entrée sont l'erreur et sa dérivée.

Fig.5.1. facteurs d'échelle.

Ge : gain de l'erreur, agit sur l'amplitude de la réponse.

Gde : gain de la dérivée de l'erreur, agit sur la rapidité du système.

Gu : gain de la commande, agit sur la stabilité du système.

Les trois facteurs d'échelle Ge, Gde et Gu sont fixés pour que l'étalement de chaque variable réelle corresponde à l'étendue normalisée des univers de discours. Généralement, ils sont déterminés en faisant des essais de simulation et varier ces facteurs jusqu'à ce qu'on trouve un phénomène de réglage convenable.

II.7.2.les régulateurs PD, PI, PID flous.

La figure suivante montre des correcteurs PD, PI, PID par logique floue ou on précise le type des entrées et des sorties :

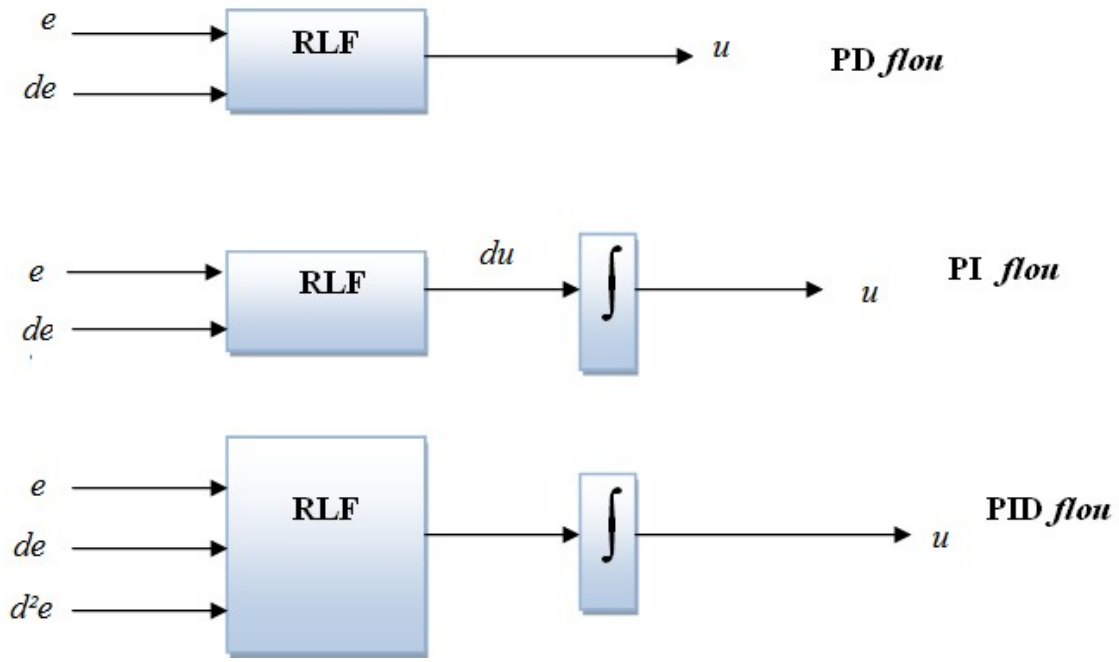


Fig.5.2. schéma simplifié de correcteur flou de type PD, PI et PID

II.7.2.a. Correcteur floue de type PD

La figure ci-dessous montre la structure du régulateur PD flou.

Les deux entrées sont l'erreur et sa dérivée qui sont normalisées au moyen des gains de normalisation **Ge** et **Gde** respectivement. Un gain de dé normalisation **Gu** est affecté a la sortie.

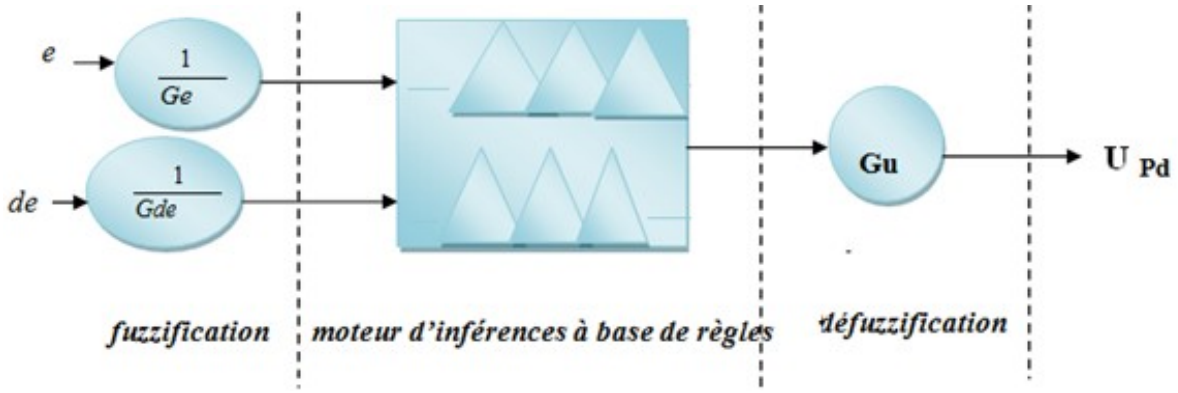


Fig.5.3. régulateur PD flou

La première étape consiste à définir la base de règles du correcteur, c'est à dire le moteur d'inférence flou.

II.7.2.b. Correcteur flou de type PI

Pour réaliser un correcteur de type PI, il suffit d'intégrer la sortie du moteur d'inférence flou indiqué en notant u_{pi} la sortie du contrôleur flou de type PI

$$U_{pi} = \int U_{pd}\, dt$$

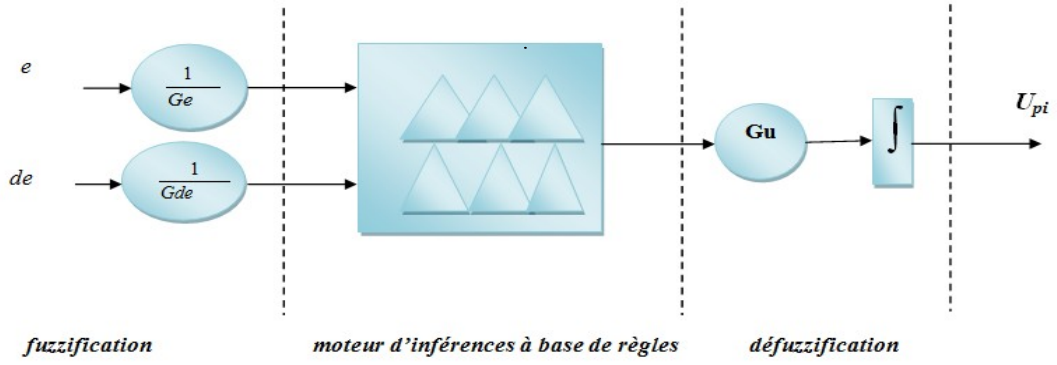


Fig.5.4 régulateur flou, type PI.

II.7.2.c. Correcteur flou de type PID

Pour réaliser un correcteur de type PID flou, une partie intégrale est ajoutée en parallèle au moteur d'inférence flou.

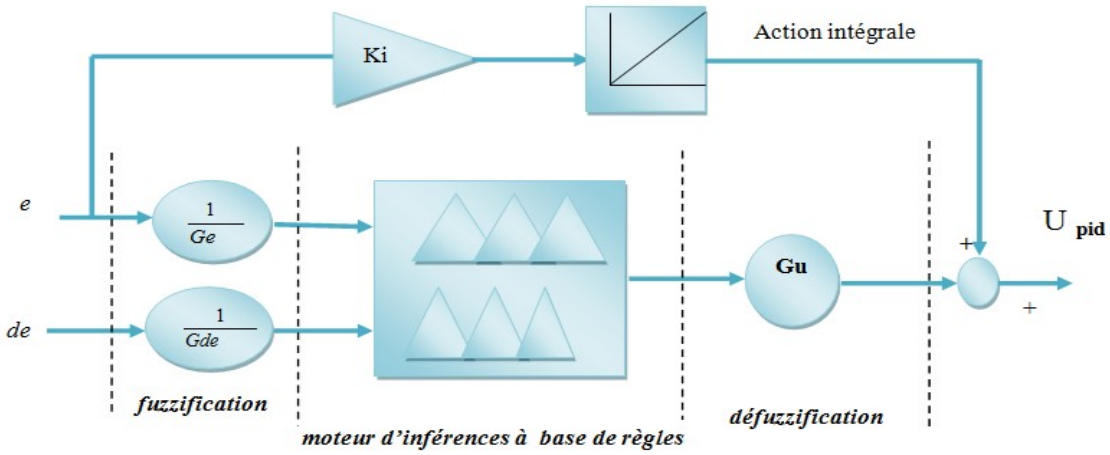


Fig.5.5. régulateur flou, type PID.

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

En notant u_{pid} la sortie du contrôleur flou de type *PID*, on aura :

$$U_{pid} = K_i \int e(t)dt + u_{pd}$$

II.8.1. Les avantages principaux des régulateurs flous sont les suivant

L'avantage d'un système flou est que seules les connaissances du comportement de procédé à commander sont suffisante pour la synthèse de la loi de commande , et ils soulèvent un large intérêt , tant théorique que pratique , dans la commande du processus complexes et N-L .Cela est dû essentiellement a trois traits principaux :

1- Les systèmes flous n'exigent pas l'existence d'un modèle analytique du processus à contrôler, et peu d'information est suffisante pour mettre en œuvre la boucle de commande.

3- Les systèmes flous sont des systèmes non linéaires et de ce fait plus adaptés à la commande des processus non linéaire.

4-La possibilité d'implémenter des connaissances linguistiques de l'opérateur de processus.

II.8.2. Les désavantages sont les suivants

- Manque de directive précise pour la conception d'un régulateur
- Approche artisanale et non systématique (il est souvent très difficile d'implanter les connaissances de l'opérateur)
- Impossibilité de démontrer la stabilité du circuit de réglage en toute généralité (en absence d'un modèle valable)
- Précision de réglage en générale peu élevée
- Cohérence des inférences non garanti a priori (possibilité d'apparition de règles d'inférences contradictoires)

II.9. Conclusion

Au cours de ce chapitre, nous avons présenté le principe de la commande par logique floue ainsi que la principale démarche pour la conception d'un régulateur flou et des régulateurs flous de types PD, PI, et PID ainsi que les bases nécessaires à la compréhension des méthodes à base de la logique floue. Nous avons appliqué la stratégie de commande par PID fou à un système non linéaire, les résultats de simulation sont donnés dans le dernier chapitre.

Les régulateurs standard traitent les relations mathématiques bien définies. Dans le cas de réglage par logique floue il n'est pas nécessaire d'établir un modèle du système à régler.

Si un modèle existe quand même on peut l'utilisé pour tester et modifier la stratégie du réglage a l'aide d'une simulation numérique.

La conception d'un système flou commence par le choix de variables linguistiques qui déterminent son état, puis des règles linguistiques qui établissent les relations d'inférences entre ces variables. En général les règles sont proposées par un expert, on partitionne ensuite le domaine de chaque variable linguistique en un ensemble de fonctions d'appartenances qui expriment les valeurs de façon approximatives ,petit ,moyen, grand, énorme par exemple.

Dans les systèmes flous il y a pas de méthodes standards pour la transformation de la connaissance ou de l'expérience humaine vers la base de règles linguistiques d'un système flou, il y a pas de procédure générale pour choisir un nombre optimal de règles, de plus lorsqu'un expert humain est disponible, sa connaissance est plutôt incomplète que systématique.

Dés lors il est indispensable de trouver une méthode automatique pour l'ajustement des fonctions d'appartenances, qui minimise l'erreur de la sortie du système.

En résumé ; il serait utile d'automatiser le processus de conceptions des systèmes flous. On pourrait diviser cette tache en deux parties

- Identification de la structure du système (recherche de nombre idéal de règles)
- Identification des paramètres (ajustement des fonctions d'appartenances)

Pour ces raisons, des recherches très poussées ont conduit au développement des méthodes systématiques pour la conception des contrôleurs flous .Parmi ces méthodes: Les réseaux de neurones qui sont très puissants dans le domaine de l'apprentissage, et cela en utilisant les fonctions qu'on veut optimiser comme poids à ajuster. De ce fait, le contrôleur flou peut être utilisé avec les techniques des réseaux de neurones pour la conception de contrôleur **« neuro - flou ».**

III.1.Introduction

Les systèmes hybrides qui combinent la logique floue, les réseaux de neurones, les algorithmes génétiques prouvent leur efficacité dans une variété de problèmes de monde réel et dans l'industrie.

Chaque technique intelligente a des propriétés particulière par exemple (capacité d'apprentissage, explication de décision) chaque technique convient a résoudre certains problèmes particuliers.

En effet, les réseaux de neuronaux sont par exemple utilisés pour la reconnaissance des modèles, la commande….

Cependant, ils sont incapables d'expliquer comment ils atteignent leurs décisions ;aussi pour les systèmes de la logique floue qui peuvent raisonnez avec l'information imprécise et expliquer leurs décisions mais ne peuvent cependant pas acquérir automatiquement les règles qu'ils utilisent pour prendre ces décisions.

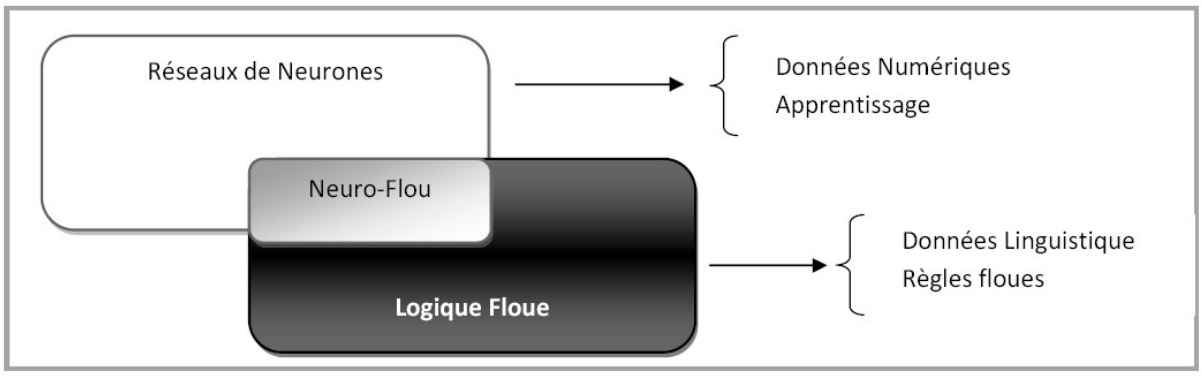
Ces limites ont étés une raison derrière la création des systèmes hybrides intelligents ou deux ou plus de techniques sont combinées afin de vaincre les limitations d'une seule technique.

L'utilisation conjointe des méthodes neuronales et floues dans un système hybride permet de tirer davantage des qualités de l'une et de l'autre, principalement les capacités d'apprentissage des premières et la lisibilité et la souplesse des éléments manipulés par les secondes.

De nombreux auteurs ont donc tout naturellement cherché à combiner ces deux techniques depuis le début des années 1990 et ceci de plusieurs manières.

III.2. définition de système neuro-flou

Les réseaux neuro-flous c'est la mise en commun des réseaux de neurones et de la logique floue



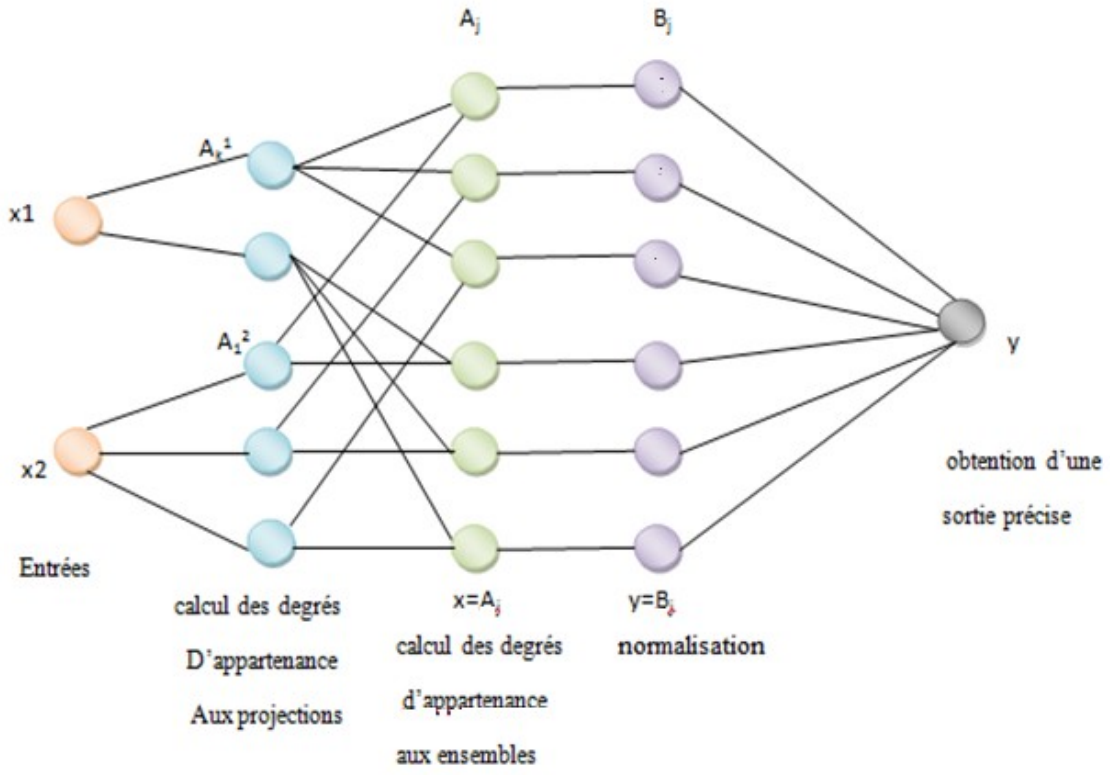
Un système neuro-flou consiste à utiliser un réseau de neurones multicouches pour lequel chaque couche correspond à la réalisation d'une étape d'un système d'inférence flou.

III.3. Architectures de réseaux neuro-flous

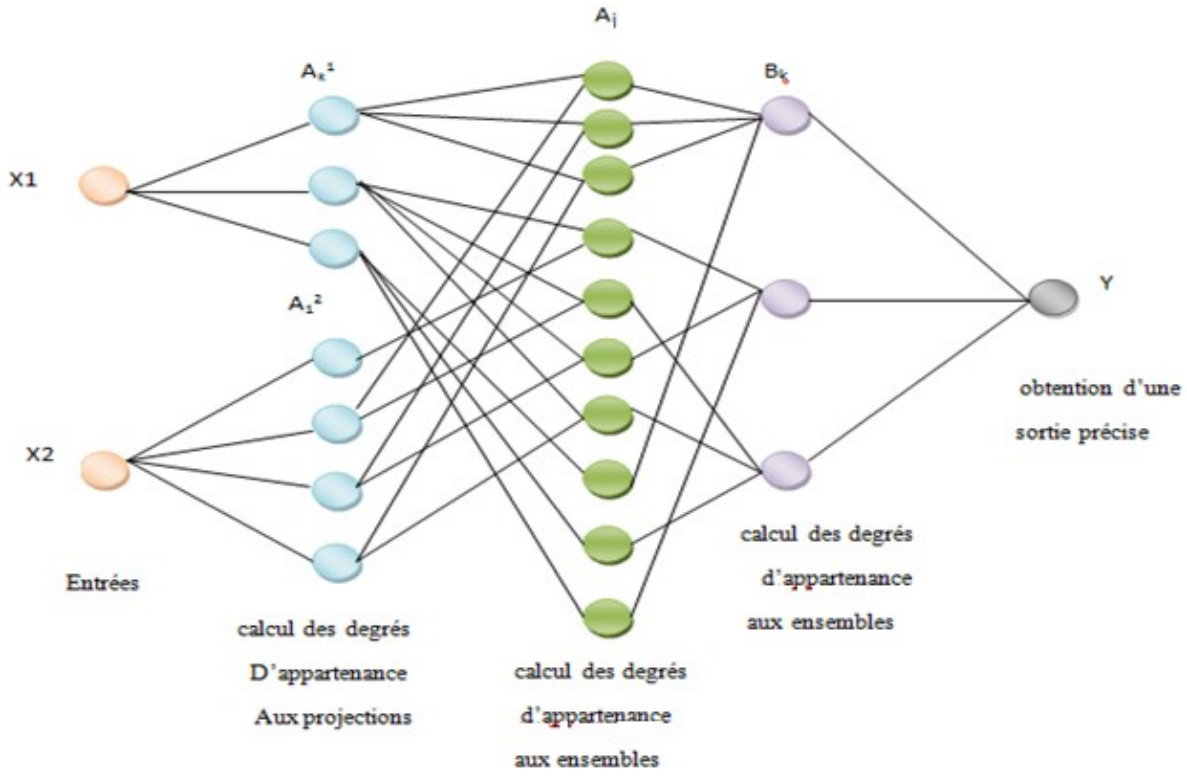
On trouve essentiellement deux types d'architectures connexionnistes des systèmes d'inférences floue ; l'une correspondant a un système avec J prémisses et J conclusions et l'autre avec J prémisses et k conclusions.

III.3.1.architecture J prémisses J conclusions

Cette architecture de réseau qui est la plus répondue prend pour point de départ le codage qui associe directement J prémisse à J conclusion, l'implémentation neuronale la plus synthétique de l'inférence des règles J prémisse J conclusion s'appuis sur cinq couche, la première c'est la couche d'entrée, la deuxième et la troisième codent les ensembles représentant les prémisses tandis que la quatrième code un lien entre un neurone « prémisse » et un neurone « conclusion » en effectuant une simple normalisation des sorties de la couche précédente enfin le neurone de sortie linéaire fournit la sortie défuzzifiée, parmi les systèmes mettant en œuvre une telle architecture, le plus connu est sans doute AFNIS (adaptative network-based fuzzy inférence systèm)développé par Jang



III.3.1.architecture J prémisses K conclusions



on s'intéressera dans ce chapitre a un réseau neuro-flou de type J prémisses J conclusions qui est le système d'inférence flou basé sur les réseaux de neurones **ANFIS** (Adaptative Network Fuzzy Inférence Systém).

III.4.1.architecture du réseau ANFIS

le réseau adaptatif ANFIS est un réseau multicouches à cinq couches dont les connexions ont toutes un poids égale a 1. Les nœuds sont de deux types différents selon leur fonctionnalité. Les nœuds carrés (adaptatifs) contiennent des paramètres, et les nœuds circulaires (fixes) n'ont pas de paramètres. Toutefois, chaque nœud (carré ou circulaire) applique une fonction sur ses signaux d'entrées.

Pour simplifier la compréhension et sans perte de généralité, nous considérons un système à deux entrées x1 et x2 et une sortie y. Considérons aussi un modèle flou composé des deux règles suivantes :

Si x1 est A1et x2est B1 alors Y1=f1(x1, x2)=a1x1+b1x2+c1

Si x1 est A2 et x2 est B2 alors Y2=f2(x1,x2)=a2x1+b2x2+c2

Jang a proposé de représenter cette base de règles par le réseau adaptatif de la figure suivante

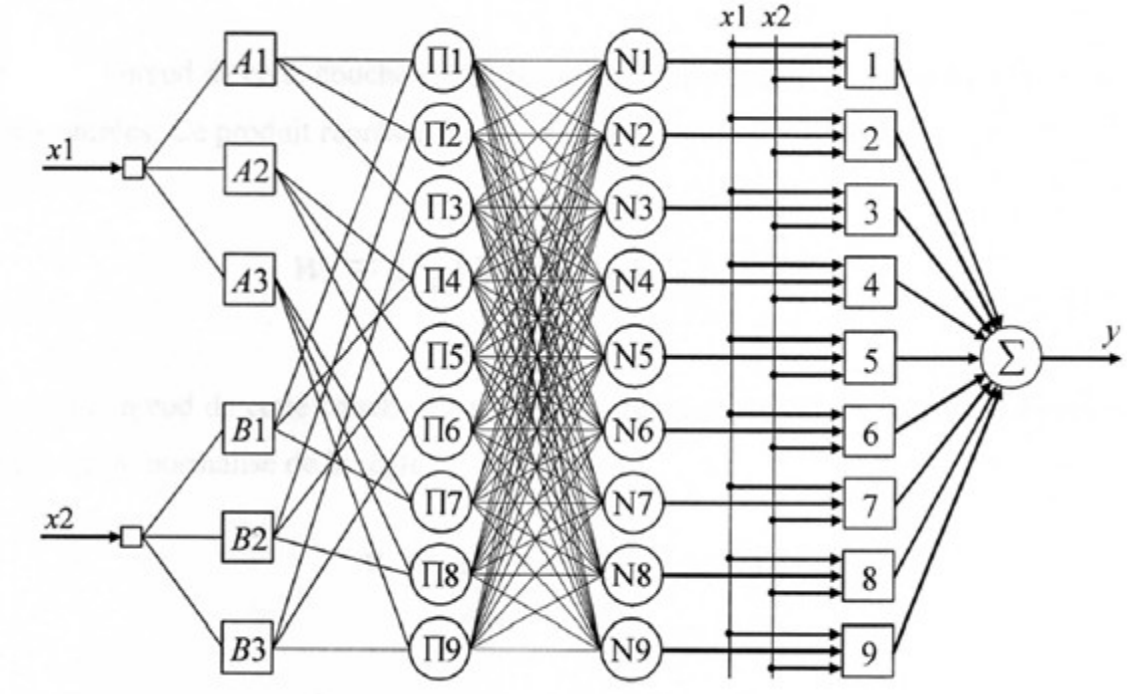


Fig.1. architecture ANFIS

La sortie o^k_i du nœud i de la couche k (appelée nœud (i, k)) dépend des signaux provenant de la couche K-1 et des paramètres du nœud (i, k), c'est-à-dire,

$$o_i^k = f(o_1^{k-1},.....,o_{n_{k-1}}^{k-1},a,b,c...)$$

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

Couche 1 :

Les neurones de cette couche réalisent les ensembles flous qui serviront dans les antécédents des règles. Dans le modèle de Jang, les fonctions d'appartenance sont des gaussiennes.

$$o_i^1 = \mu_{Ai}(x)$$

Ou x est l'entrée du nœud i, et A_i le terme linguistique associé à sa fonction .En d'autres termes, o_i^1 est le degré d'appartenance de x à A_i.les paramètres d'un nœud de cette couche sont ceux de la fonction d'appartenance correspondante.

Couche 2 :

Chaque neurone dans cette couche correspond à une règle floue. Il reçoit les sorties des neurones de fuzzification et calcule son activation. La conjonction des antécédents est réalisée avec l'opérateur produit.

$$w_i = \mu_{Ai}(x_1) * \mu_{Bi}(x_2)$$

Couche 3 :

Chaque neurone calcule le degré de vérité normalisé d'une règle floue donnée. La valeur obtenue représente la contribution de la règle floue au résultat final.

$$v_i = \frac{w_i}{\sum_i w_i}$$

Couche 4 :

Chaque neurone i de cette couche est relié à un neurone de normalisation correspondante et aux entrées initiales du réseau. Il calcule le conséquent pondéré de la règle.

$$o_i^4 = v_i f_i = v_i(a_i x_1 + b_i x_2 + c_i)$$

Ou v_i est la sortie de la couche 3, et 4{ai, bi, ci} est l'ensemble des paramètres de sorties de la règle i.

Couche5 :

Comprend un seul neurone qui fournit la sortie de ANFIS en calculant la somme des sorties de tous les neurones de défuzzification.

$$o_1^5 = y = \sum_i v_i f_i$$

III.4.2. apprentissage du réseau ANFIS

L'apprentissage à partir d'un ensemble de données concerne l'identification des paramètres des prémisses et des conséquences, la structure du réseau étant fixée. L'algorithme d'apprentissage commence par construire un réseau initial, ensuite on applique une méthode d'apprentissage par rétro-propagation de l'erreur. Jang a proposé d'utiliser une règle hybride d'apprentissage qui combine un algorithme de rétro propagation de l'erreur avec une estimation par moindres carrées.

III.4.2.a. détermination des paramètres des conséquentes

Les paramètres des conséquentes sont déterminés par une estimation par moindres carrées. Considérant p paires de données d'apprentissage ; on peut former p équations linéaires en fonction des paramètres des conséquentes.

$$\begin{cases} y_d(1) = \overline{\mu}_1(1) f_1(1) + \overline{\mu}_2(1) f_2(1) + \ldots + \overline{\mu}_n(1) f_n(1) \\ y_d(2) = \overline{\mu}_1(2) f_1(2) + \overline{\mu}_2(2) f_2(2) + \ldots + \overline{\mu}_n(2) f_n(2) \\ \vdots \\ y_d(p) = \overline{\mu}_1(p) f_1(p) + \overline{\mu}_2(p) f_2(p) + \ldots + \overline{\mu}_n(p) f_n(p) \\ \vdots \\ y_d(P) = \overline{\mu}_1(P) f_1(P) + \overline{\mu}_2(P) f_2(P) + \ldots + \overline{\mu}_n(P) f_n(P) \end{cases}$$

Où $\overline{\mu_i}$ est l a valeur moyenne de μ_i et $f_i()$ est la fonction de sortie dont on veut déterminer les paramètres. On peut écrire l'équation précédente sous forme $y_d = AK$,où y_d est un vecteur désiré de dimension Px1

$$y_d = \begin{bmatrix} y_d(1) \\ y_d(2) \\ \vdots \\ y_d(p) \\ \vdots \\ y_d(P) \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} \overline{\mu}_1(1) & \overline{\mu}_1(1)x_1(1) & \cdots & \overline{\mu}_1(1)x_m(1) & \cdots & \overline{\mu}_n(1) & \overline{\mu}_n(1)x_1(1) & \cdots & \overline{\mu}_n(1)x_m(1) \\ \overline{\mu}_1(2) & \overline{\mu}_1(1)x_1(2) & \cdots & \overline{\mu}_1(2)x_m(2) & \cdots & \overline{\mu}_n(2) & \overline{\mu}_n(2)x_1(2) & \cdots & \overline{\mu}_n(2)x_m(2) \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \overline{\mu}_1(p) & \overline{\mu}_1(p)x_1(p) & \cdots & \overline{\mu}_1(p)x_m(p) & \cdots & \overline{\mu}_n(p) & \overline{\mu}_n(p)x_1(p) & \cdots & \overline{\mu}_n(p)x_m(p) \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \mu_1(P) & \overline{\mu}_1(P)x_1(P) & & \overline{\mu}_1(P)x_m(P) & & \overline{\mu}_n(P) & \overline{\mu}_n(P)x_1(P) & & \overline{\mu}_n(P)x_m(P) \end{bmatrix}$$

et K est le vecteur des paramètres de conséquents inconnus de dimension n(1+m)x1

$$k = [k_{10}\, k_{11}\, k_{12} \ldots k_{1m}\, k_{20}\, k_{21} k_{22} \ldots k_{2m} \ldots k_{n0}\, k_{n1}\, k_{n2} \ldots k_{n\,m}]\, T$$

On a donc $K = A^{-1} y_d$

III.4.2.b. détermination des paramètres des prémisses

Une fois le vecteur K est déterminé, le vecteur de sortie du réseau y peut être calculé ainsi que le vecteur associé e

$$e = y_d - y$$

Pour la détermination des paramètres des prémisses on utilise l'algorithme de rétro propagation de gradient de l'erreur, l'apprentissage est supervisé, pour chaque entrée on associe une sortie désirée. L'algorithme de rétro propagation est un algorithme de gradient itératif conçu pour minimiser un critère quadratique d'erreur entre la sortie obtenue et la sortie désirée .cette minimisation est réalisée par une configuration de paramètres adéquate. Considérons le problème de minimisation quadratique suivant :

$$e = \frac{1}{2}\sum_{k=1}^{M}(y_d - y)^2$$

Les paramètres des prémisses sont les paramètres des fonctions d'appartenances.

On considère par suite les fonctions de type gaussiennes

$$\mu_A(x) = \exp(-\frac{(x-c)^2}{2*\sigma^2})$$

III.4.2.b.1. codage des sous –ensembles flous

La première couche de l'architecture ANFIS comporte autant de neurones qu'il y a de sous ensemble flous. Dans le système d'inférence représenté chaque unité calcule le degré de vérité d'un sous ensemble particulier par sa fonction de transfert, la seule restriction sur le choix de la fonction concerne sa dérivabilité.

Les paramètres modifiables c'est-à-dire centre et varience de la gaussienne sont codée par les poids de connexions provenant d'un neurone dont la sortie reste fixée à1.

La figure.3. représente le codage d'une fonction d'appartenance de type gaussienne.

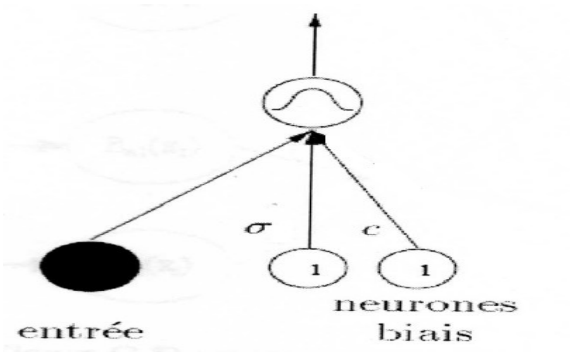


Fig.3. codage des paramètres associés à une fonction d'appartenance

La figure suivante représente la généralisation pour toutes fonctions d'appartenance

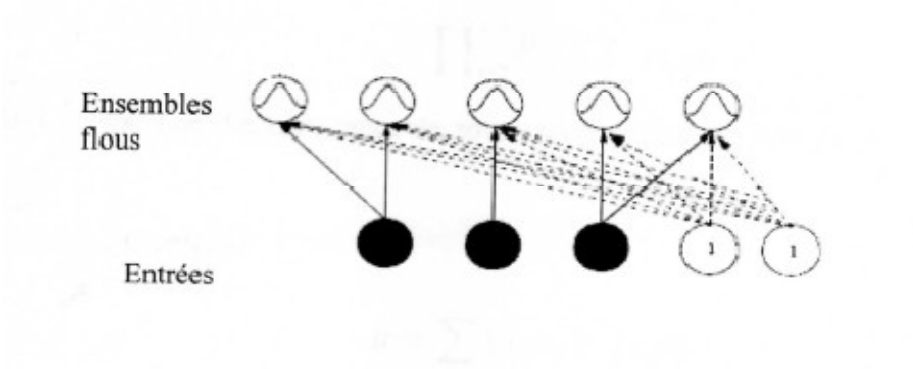


Fig.4.généralisation pour toutes les fonctions

III.4.2.b.2. Algorithme de rétro-propagation de gradient de l'erreur

On applique l'algorithme de rétro-propagation pour chercher la configuration des paramètres de prémisses qui minimise l'erreur quadratique donnée ci-dessus,

Considérons le réseau neuro-flou suivant :

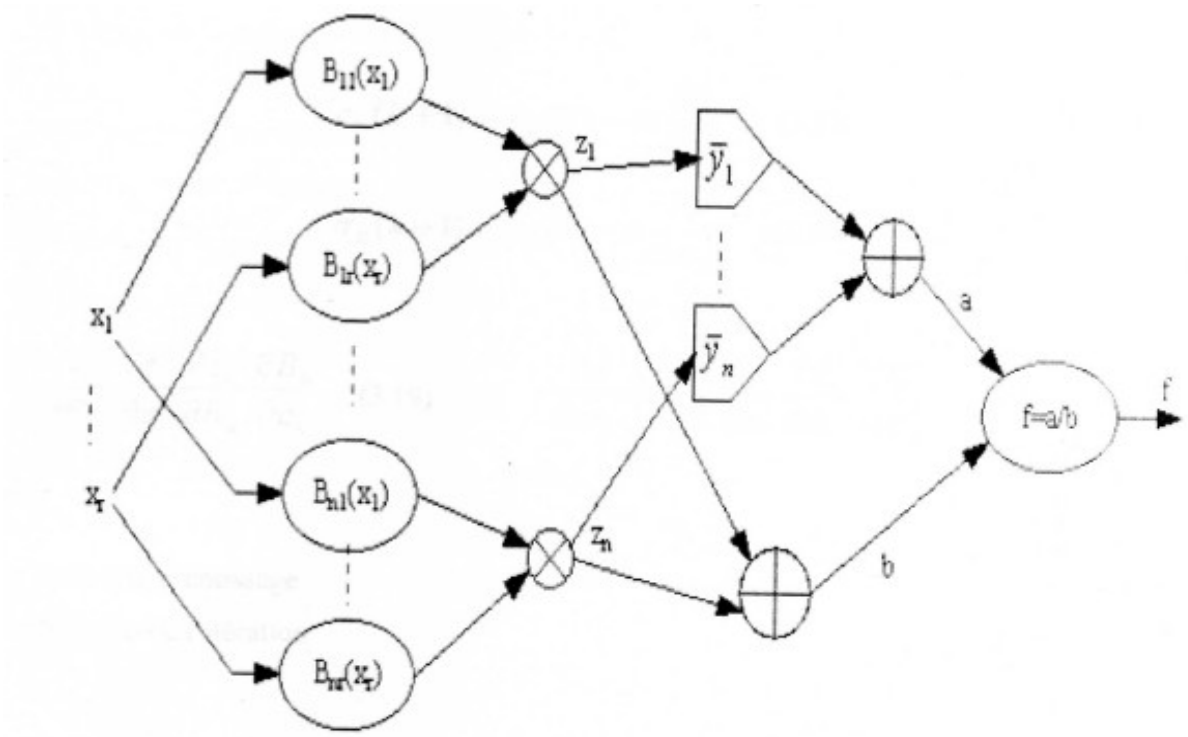


Fig.4. système neuro-flou

Dans les différentes étapes de l'algorithme nous utilisons la notation suivante :

$$z_l = \prod_{i=1}^{r} B_{li}(x_i)$$

avec $B_{li}(x_i)$ fonction d'appartenance du sous ensemble flou l de la variable x_i et (l=1,….n ; i=1,….r).

z_l : activation de la règle l

$$a = \sum_{l=1}^{n} (\overline{y_l} z_l) \qquad b = \sum_{l=1}^{n} z_l$$

Pour ajuster les paramètres c_{li} et σ_{li} nous utilisons les équations suivantes :

$$c_{li}(k+1) = c_{li}(k) - \alpha \frac{\partial e}{\partial c_{li}}$$

$$\sigma_{li}(k+1) = \sigma_{li}(k) - \alpha \frac{\partial e}{\partial \sigma_{li}}$$

$$\frac{\partial e}{\partial c_{li}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_l} \frac{\partial z_l}{\partial B_{li}} \frac{\partial B_{li}}{\partial c_{li}}$$

$$\frac{\partial e}{\partial \sigma_{li}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_l} \frac{\partial z_l}{\partial B_{li}} \frac{\partial B_{li}}{\partial \sigma_{li}}$$

α : Taux d'apprentissage.

K : numéro de l'itération.

III.4.3. Utilisation du réseau ANFIS pour l'identification et la commande de processus :

La commande des systèmes (processus) linéaires, ou pouvant être facilement approchés par des systèmes linéaires, est aujourd'hui un domaine bien maitrisé par les automaticiens.

Les outils de l'algèbre linéaire permettent d'obtenir pour ces systèmes, des contrôleurs possédant des propriétés de stabilité et d'optimalité. Dans le monde réel cependant un grand nombre de processus sont caractérisés par un comportement dynamique non linéaire complexe, et rendent impossible l'utilisation des outils classiques de l'automatique. Il en est de même pour les systèmes pour lesquels les modèles mathématiques connus sont incomplets ou de mauvaise qualité. Il n'existe pas aujourd'hui de théorie systématique et applicable de manière générale à la commande de tels processus.

Pour résoudre ce problème une des solutions proposées consiste à avoir recours à une phase d'apprentissage pour identifier le modèle du processus ou le contrôleur. Le terme « apprentissage » désigne ici le fait de modifier la structure et/ou les paramètres du modèle, de manière à améliorer les performances futures, en se basant sur des observations

expérimentales passées. Les réseaux AFNIS sont des procédures permettant d'approcher n'importe quelle fonction linéaire ou non. C'est cette propriété qui motive leur utilisation pour la réalisation de systèmes de commande non linéaires par apprentissage.Il existe plusieurs structures de commande, dont les plus importantes sont :

III.4.3.1. Utilisation du modèle inverse :

En considérant l'équation dynamique du système à contrôler x(k+1)=f(x(k),u(k)) (or x(k) est l'environnement perçu, et u(k) est l'action entreprise, on peut utiliser un réseau ANFIS pour faire correspondre les couples { x(k),x(k+1)} et u(k).

En phase ' apprentissage' on applique une action u(k) à partir de l'état x(k) ; l'état x (k+1) observé est utilisé pour entrainer le réseau. En phase d'exploitation, à partir d'une trajectoire {x(k), x (k+1)} souhaitée, le réseau permet de trouver l'action à entreprendre.

On peut représenter les deux phases par les figures suivantes :

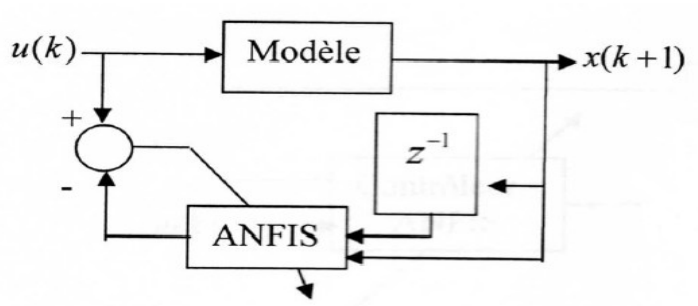


Fig.5. phase d'apprentissage

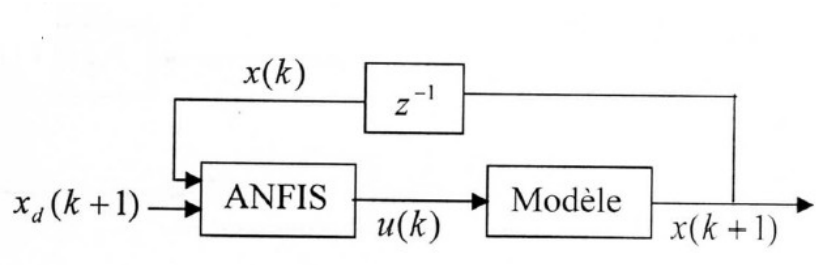


Fig.6. phase d'exploitation

En revanche l'apprentissage de la dynamique inverse pose un certain nombre de problèmes.

Lorsque l'équation dynamique f (..) n'est pas inversible, la méthode n'est pas applicable directement.

Plusieurs solutions ont été proposées pour résoudre ce problème. Dans certains cas on peut ramener à une fonction inversible en rajoutant des contraintes (minimiser les trajets, forcer un sens de parcours,…).

Ces inconvénients ont conduit plusieurs auteurs à modifier les principes de l'apprentissage de la dynamique inverse en appliquant à l'environnement la commande calculé par le réseau à partir de la trajectoire désirée. Le réseau est alors entrainé en fonction des trajectoires désirées (spécialized Learning).

III.4.3.2. Apprentissage spécialisé

L'apprentissage spécialisé permet de lever l'essentiel des inconvénients limitant l'apprentissage de la dynamique inverse. L'architecture utilisée autorise le contrôleur à explorer l'espace des commandes pour isoler celle conduisant à la meilleure concordance x (k+1)/x_d (k+1)

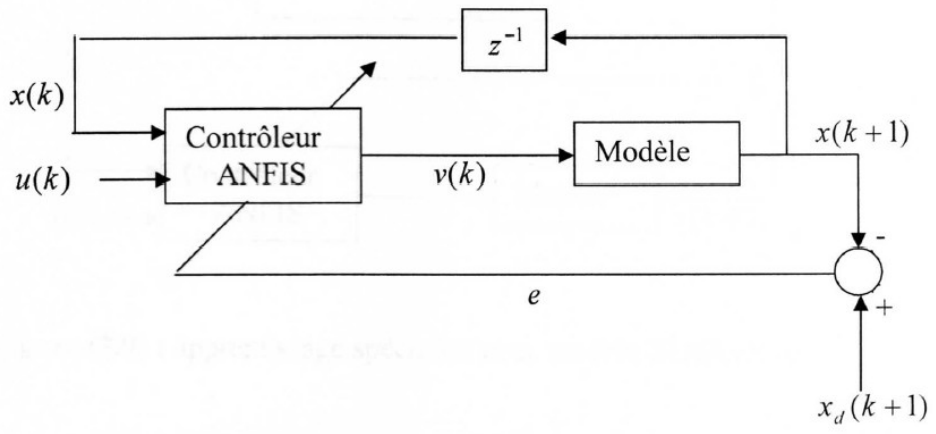


Fig.7.apprentissage spécialisé

La sortie du contrôleur est donnée par

$$v(k) = F(x(k), u(k), \theta)$$

$$x(k+1) = f(x(k), F(x(k), u(k), \theta))$$

Θ : paramètres du réseau

L'objectif de l'apprentissage est de minimiser l'écart entre la trajectoire désirée et la sortie du modèle.

On définit donc le critère quadratique qu'il faut minimiser :

$$J(\theta) = \sum_k \left\| f(x(k), F(x(k), u(k), \theta)) - x_d(k+1) \right\|^2$$

Après la définition du critère, on utilise l'algorithme de rétro propagation de l'erreur pour ajuster le vecteur ɵ jusqu'à une valeur de j(ɵ) acceptable.

Remarque : on trouve aussi l'apprentissage spécialisé avec modèle de référence, comme une autre alternative de l'apprentissage spécialisé

On peut schématiser ce type d'apprentissage par la figure suivante.

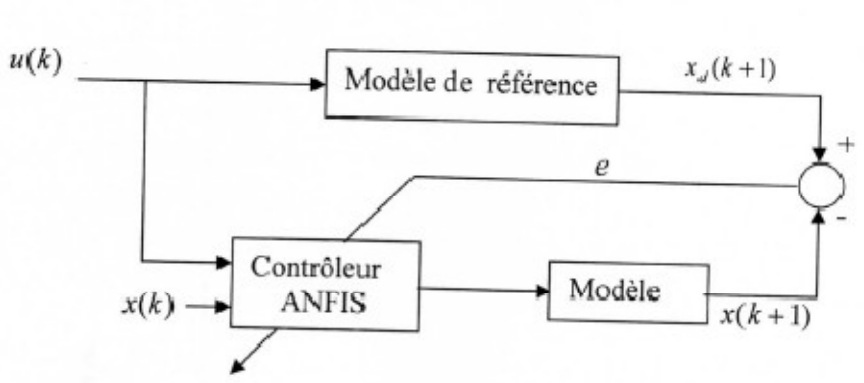


Fig.8.apprentissage spécialisé avec modèle de référence

Pour l'apprentissage du réseau, on procède de la même manière qu'avec l'apprentissage spécialisé avec une trajectoire désirée.

Conclusion :

Dans ce chapitre nous avons présenté la stratégie de commande neuro-floue utilisée lors de notre application

Cette stratégie repose sur l'intégration des avantages qu'offrent les régulateurs flous et neuronaux en un seul régulateur plus performant. n effet, elle repose sur l'implantation d'un régulateur flou par un réseau connexionniste équivalent

Nous avons appliqué cette stratégie sur système non linéaire, les résultats de simulation sont donnés dans le prochain chapitre.

IV.1. Introduction

Ce chapitre est consacré à l'application de deux régulateurs, PID flou et neuro-flou (ANFIS) pour commander le niveau de liquide dans un réservoir La première partie est une description du système à commander, la deuxième partie est consacrée a la commande du système par le contrôleur PID flou et la troisième partie c'est la commande du processus par le régulateur ANFIS et a la fin une conclusion sur les résultats simulation obtenus par les deux méthodes.

IV.2. description du système

La variable à commander est le niveau de liquide dans un réservoir ce dernier est montré sur la figure suivante.

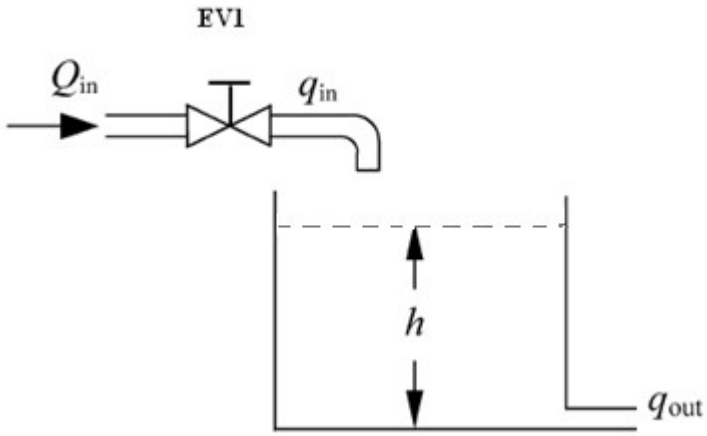


Fig.1. Réservoir

Q_{in} est le débit maximal de l'entrée exprimée en m^3/s.

q_{in} est le débit contrôlé de liquide entrant, donné par la relation suivante

$$q_{in} = Q_{in} \sin(\theta(t)).$$

Ou θ est l'angle de l'ouverture de l'électrovanne EV1 qui varie entre 0 et $\frac{\Pi}{2}$.

q_{out} est le débit de sortie $a_s\sqrt{2gh(t)}$.

Ou a_s est la surface de la sortie égale a $0.01m^2$

g=9.81m/s

$Q_{in}=0.12m^3/s$

h(t) est la variable de sortie du système donné par la relation suivante .

$$h(t) = h(o) + \frac{1}{A}\int_0^t (q_{in}(\tau) - q_{out}(\tau))\,d\tau$$

Ou A est la surface de la base de réservoir égale a $1m^2$.

Le réservoir est régit par le principe suivant : la variation de volume de liquide dans le réservoir est égale au débit entrant moins le débit sortant d'où la relation suivante

$$\frac{d[A(h(t) - h(o)]}{dt} = q_{in}(t) - q_{out}(t)$$

$$A\frac{d[(h(t) - h(o)]}{dt} = q_{in}(t) - q_{out}(t)$$

$$h(t) - h(o) = \int_0^t \frac{1}{A}(q_{in}(\tau) - q_{out}(\tau))d\tau$$

$$h(t) = h(0) + \int_0^t \frac{1}{A}(q_{in}(\tau) - q_{out}(\tau))d\tau$$

$$A\frac{d[(h(t) - h(o)]}{dt} = \mathrm{Q_{in}}\sin(\theta(\mathrm{t})) - a_s\sqrt{2gh(t)}$$

On remarque que le système est non linéaire.

Voici la représentation du système à l'aide de la boite à outil simulink

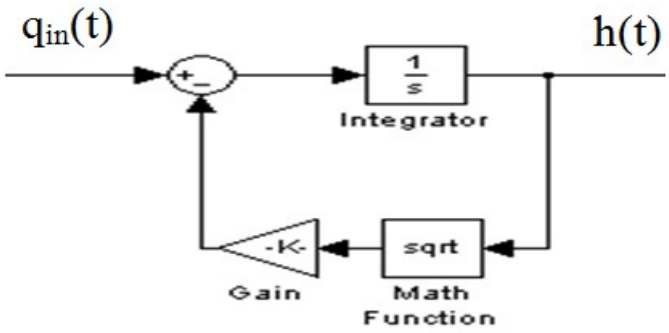


Fig.2. Représentation de système avec la boite a outil simulink

On remarque que ce système est non linéaire et pour le commander on propose deux commandes

La commande par un PID flou et la commande par un contrôleur neuro-flou (ANFIS).

IV.3. Conception de contrôleur PID flou.

Le PID flou est un contrôleur PD flou a qui on ajoute l'action intégrale qu'on applique sur l'erreur

Et voici le schéma de commande par régulateur PID flou

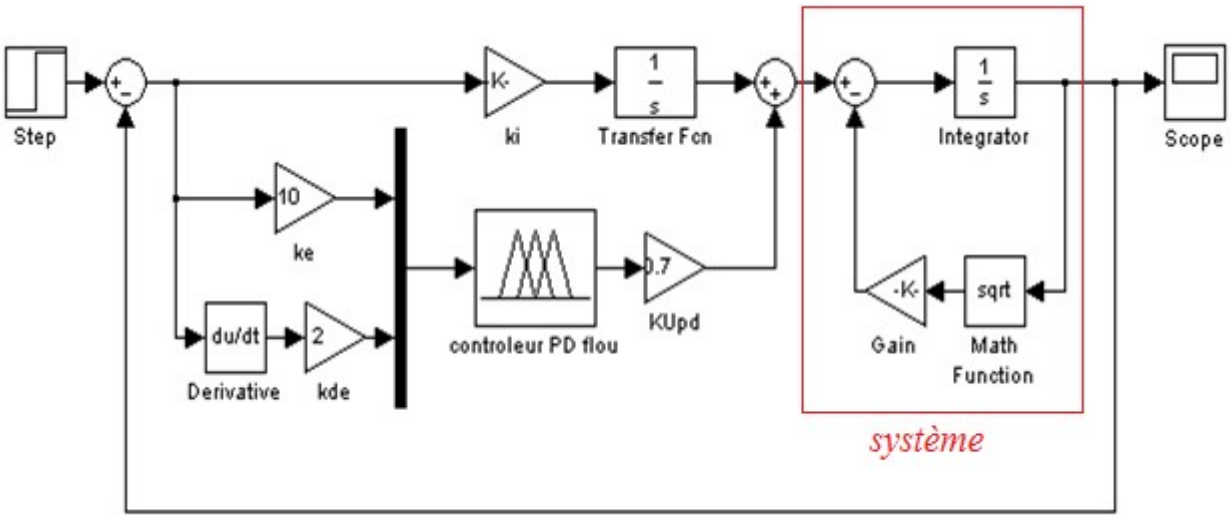


Fig.3. schéma de commande par régulateur PID flou

Pour la programmation de contrôleur PD flou on utilise La boite "fuzzy" du logiciel Matlab

Pour réaliser ce contrôleur on passe par les étapes suivantes

Définir les entrées et sorties de contrôleur

- Les entrées de système est l'erreur et la dérivé de l'erreur
- La sortie c'est le débit $q_{in}(t)$ correspondant au liquide désiré

Définir les univers de discours des deux variables d'entrées et de la sortie.

Générer les fonctions d'appartenances des deux variables

Définir la base de règles de contrôleur

Choisir les méthodes d'inférences et de deffuzzification

IV.3.1.Les fonctions d'appartenances des deux entrées et de la sortie de contrôleur PD flou

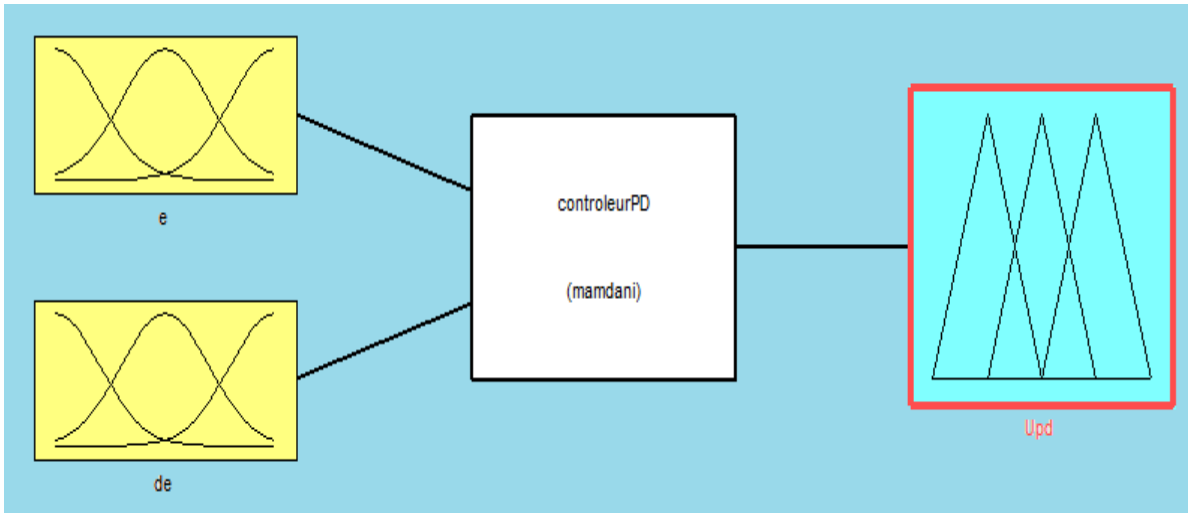


Fig.4. schéma du contrôleur PD flou

p : petit

m: zéro

g: grand

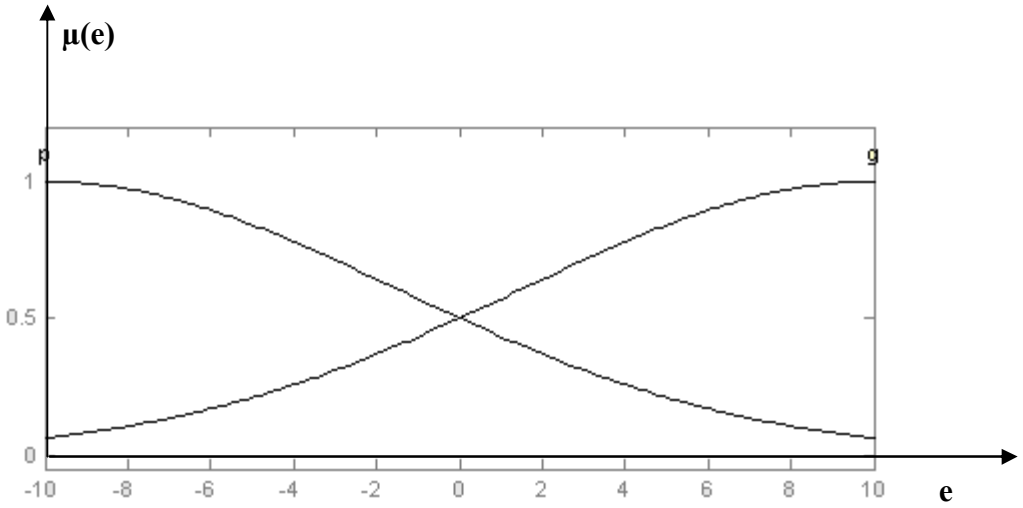


Fig.4.1. Fonctions d'appartenances de la variable e

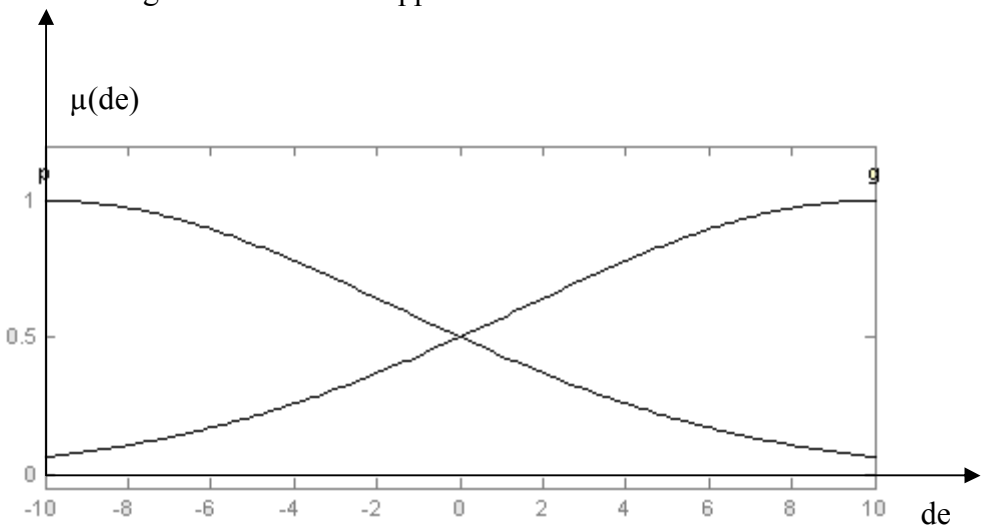


Fig.4.2. Fonctions d'appartenances de la variable de

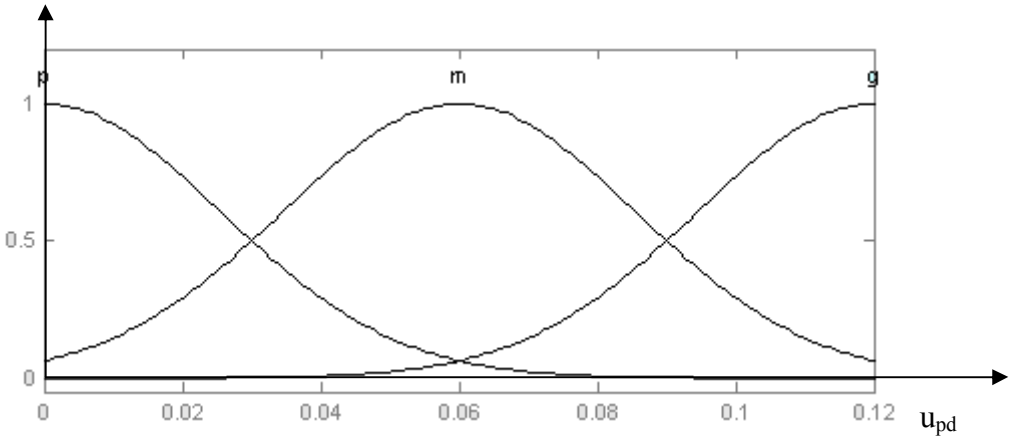


Fig.4.3. Fonctions d'appartenances de la variable de sortie u_{pd}

IV.3.2. la base de règles du contrôleur PD flou

Si "e" est petit et "de" est petit **Alors** u_{pd} est petit.

Si "e" est petit et "de" est grand **Alors** u_{pd} est moyen.

Si "e" est grand et "de" est petit **Alors** u_{pd} est moyen.

Si "e" est grand et "de" est grand **Alors** u_{pd} est grand

IV.3.3.Résultats de simulation

Pour les gains : Ke=10 , Kd_e=0.5 , Ku_{pd}=1.5 , Ki=0.001

On a obtenu les résultats suivants

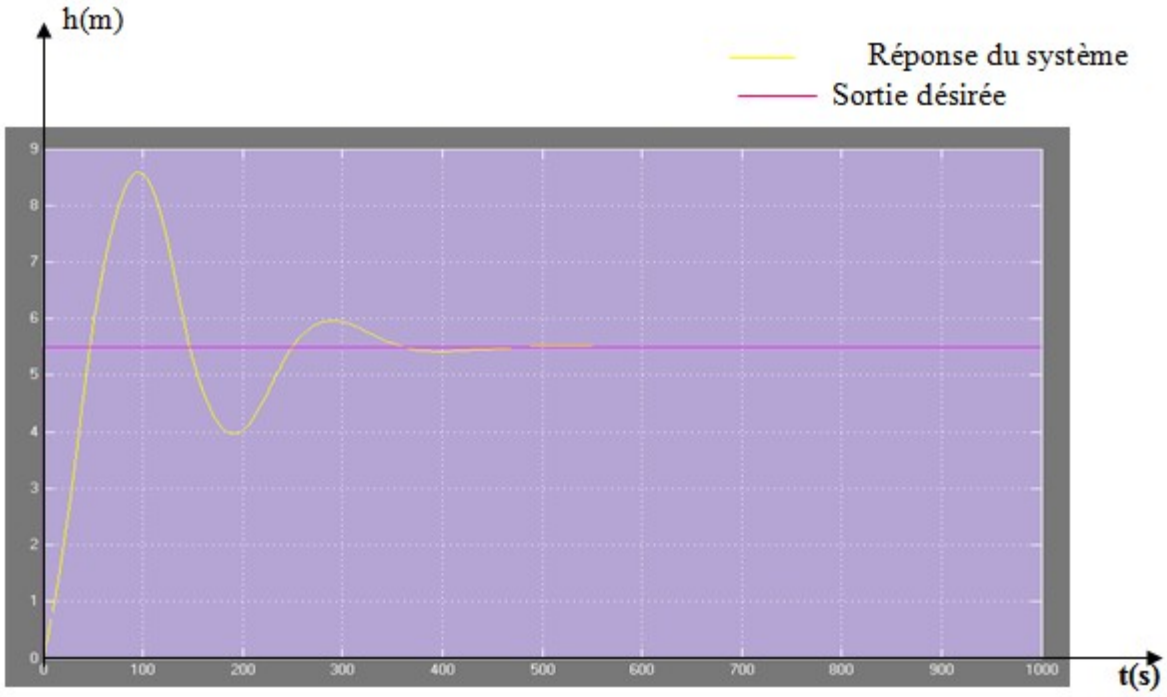


Fig.5.1. réponse du système pour un échelon de 5m

Fig.5.2. évolution de la commande

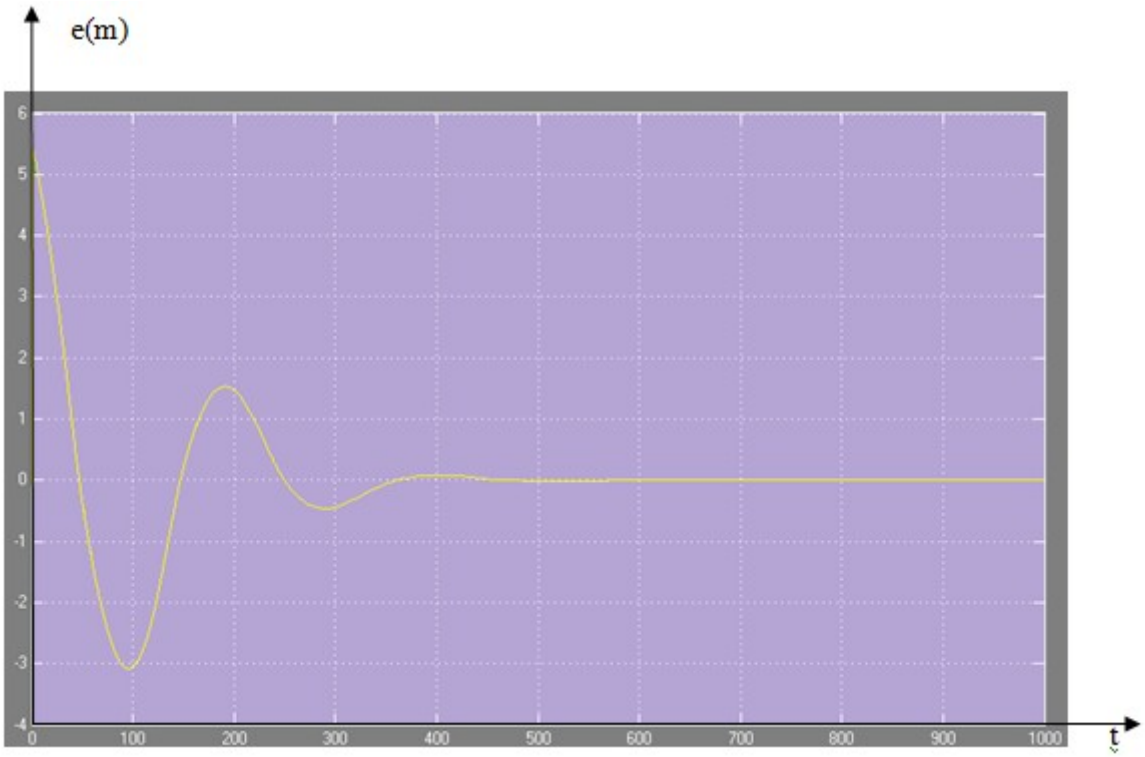


Fig.5.3.évolution de l'erreur

On aperçoit l'apparition des oscillations dans la réponse du système et pour les annulées on diminue la constante d'intégration.

Pour les gains suivants

Ki= 0.00006 ; Ku_{pd}=1.2 ; Kd_e=1.5 ; Ke=10

On a obtenus les résultats suivants

IV.3.3.1 Test en régulation

On veut réguler le niveau de liquide a 5.5m pour cela on soumit le système a un échelon de 5.5m.

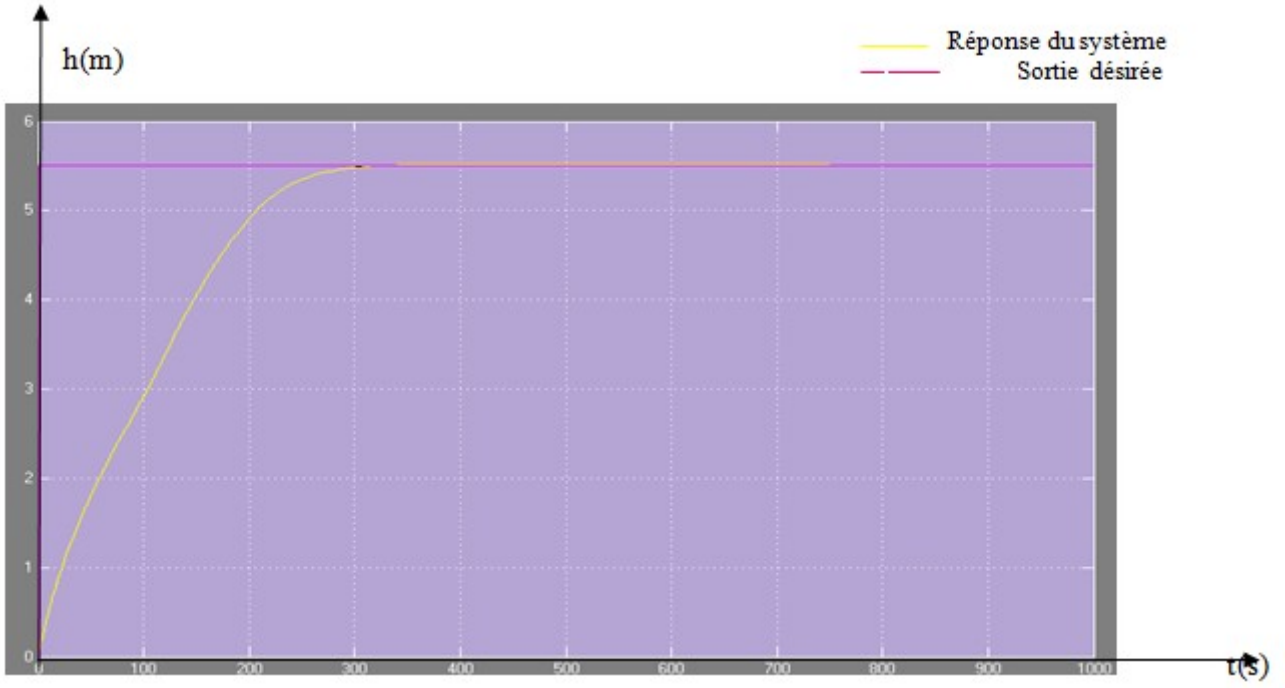


Fig.6.1.réponse du système pour un échelon de 5.5m

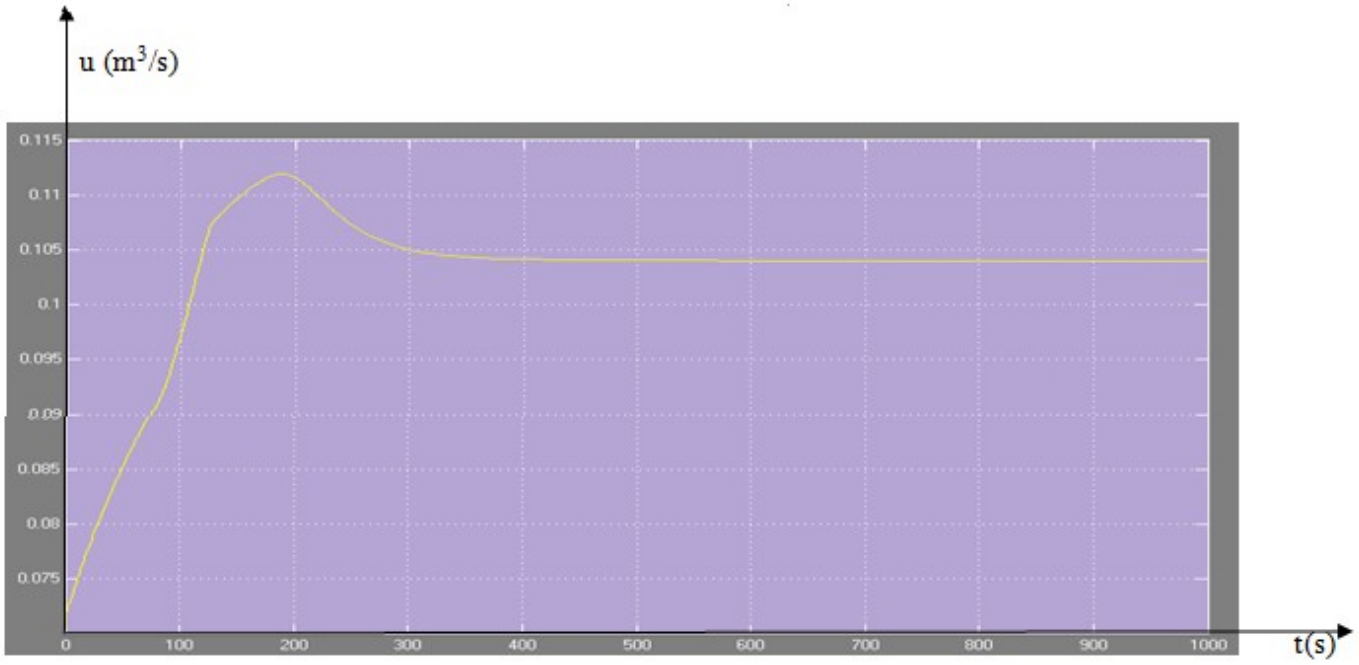


Fig.6.2.évolution de la commande

IV.3.3.2 Test en poursuite de trajectoire

On a testé le système pour deux trajectoires

- test de la première trajectoire

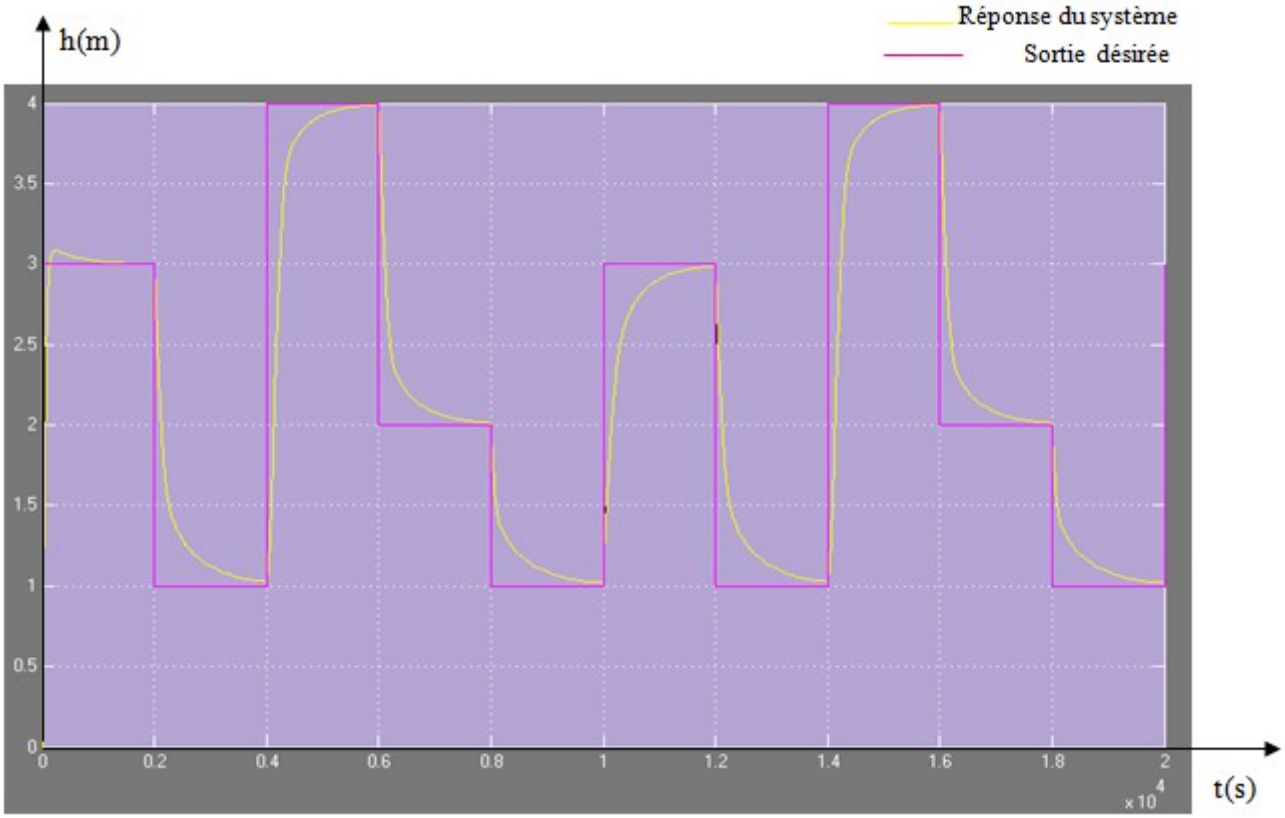


fig.7.1.réponse de système pour une trajectoire en forme d'escaliers

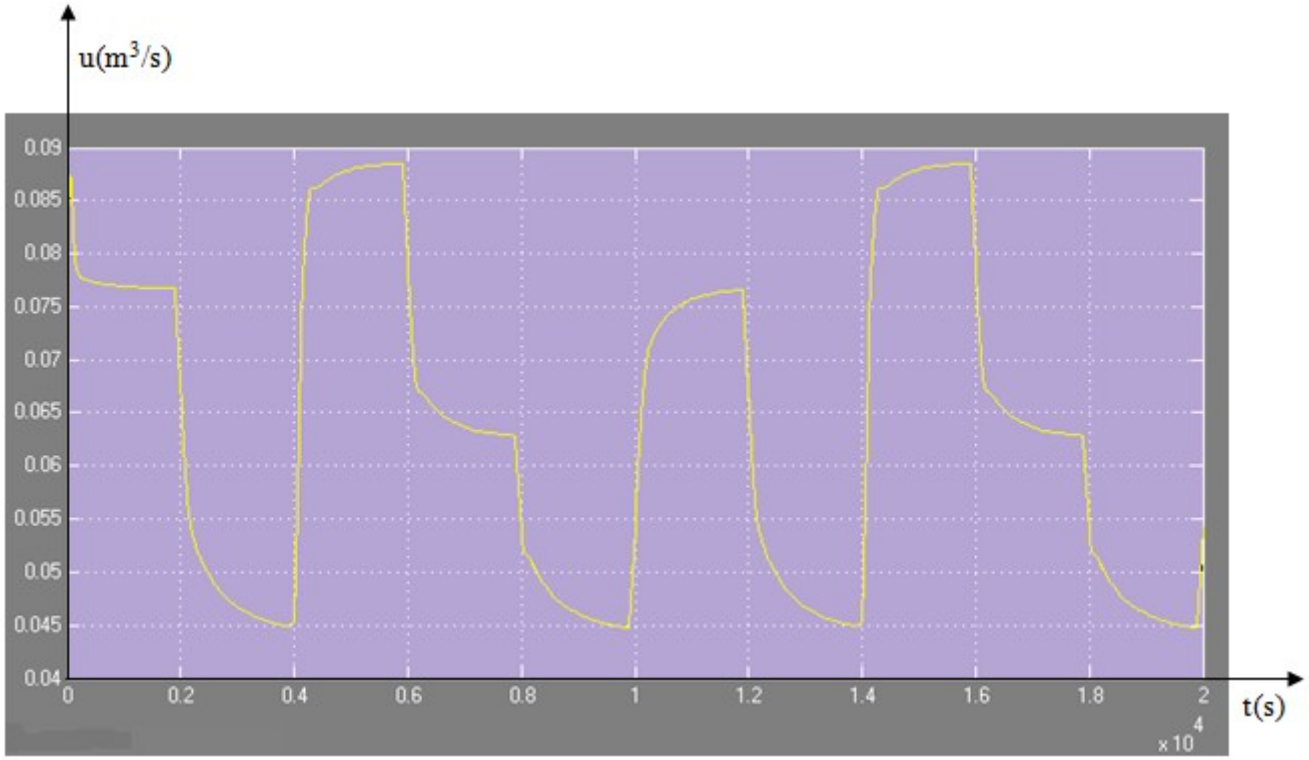


fig.7.2.évolution du signal de commande

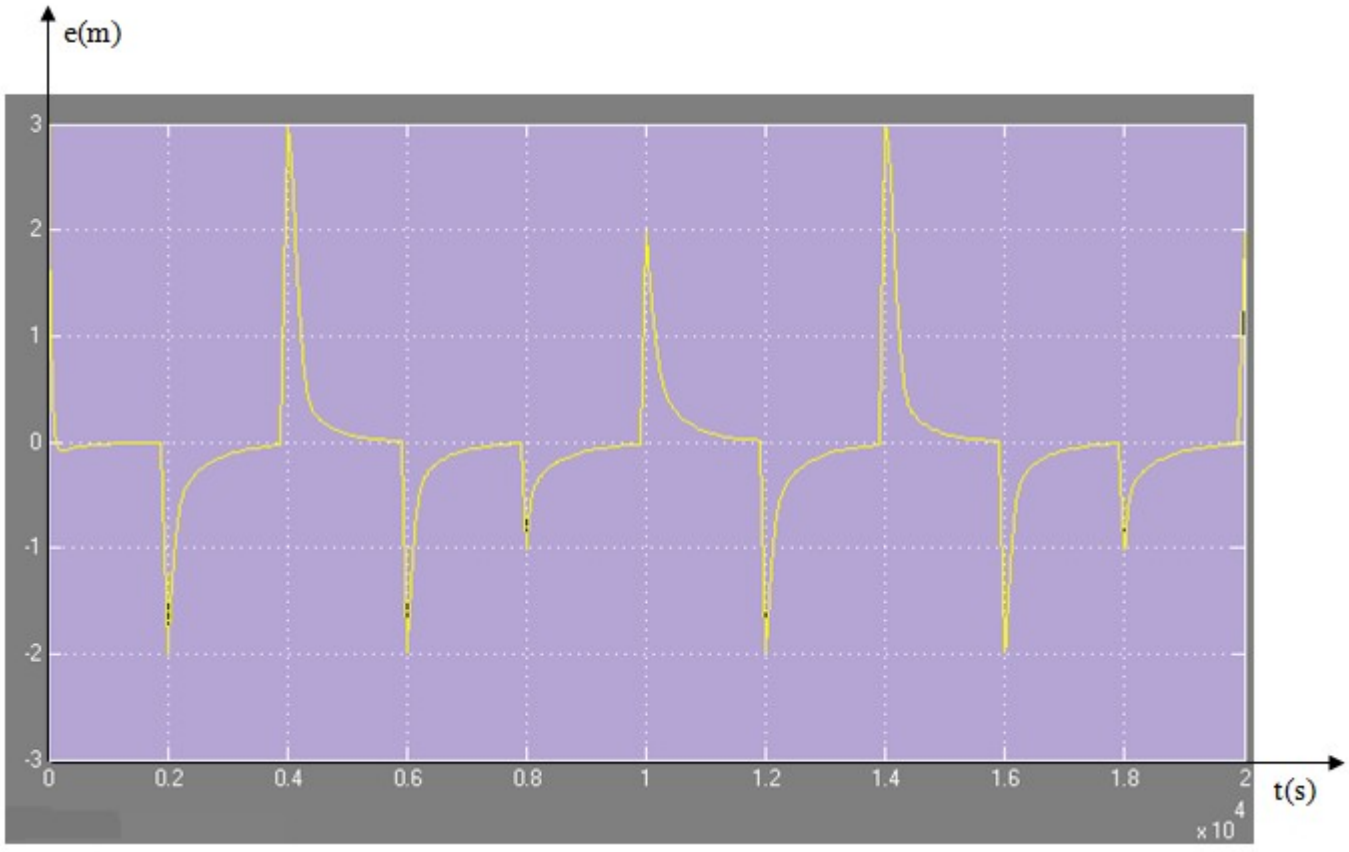


Fig.7.3. évolution de l'erreur

- test de la deuxième trajectoire

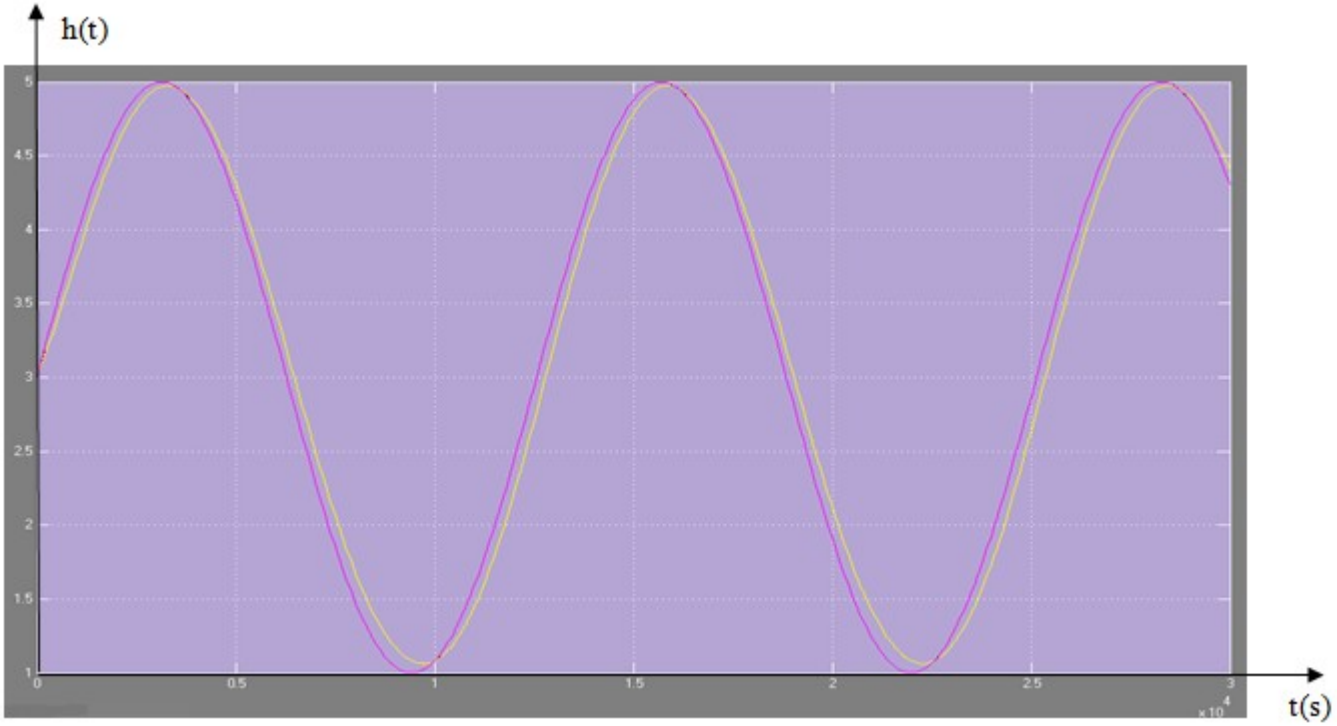


fig.8.1.réponse de système a un signal sinusoïdal

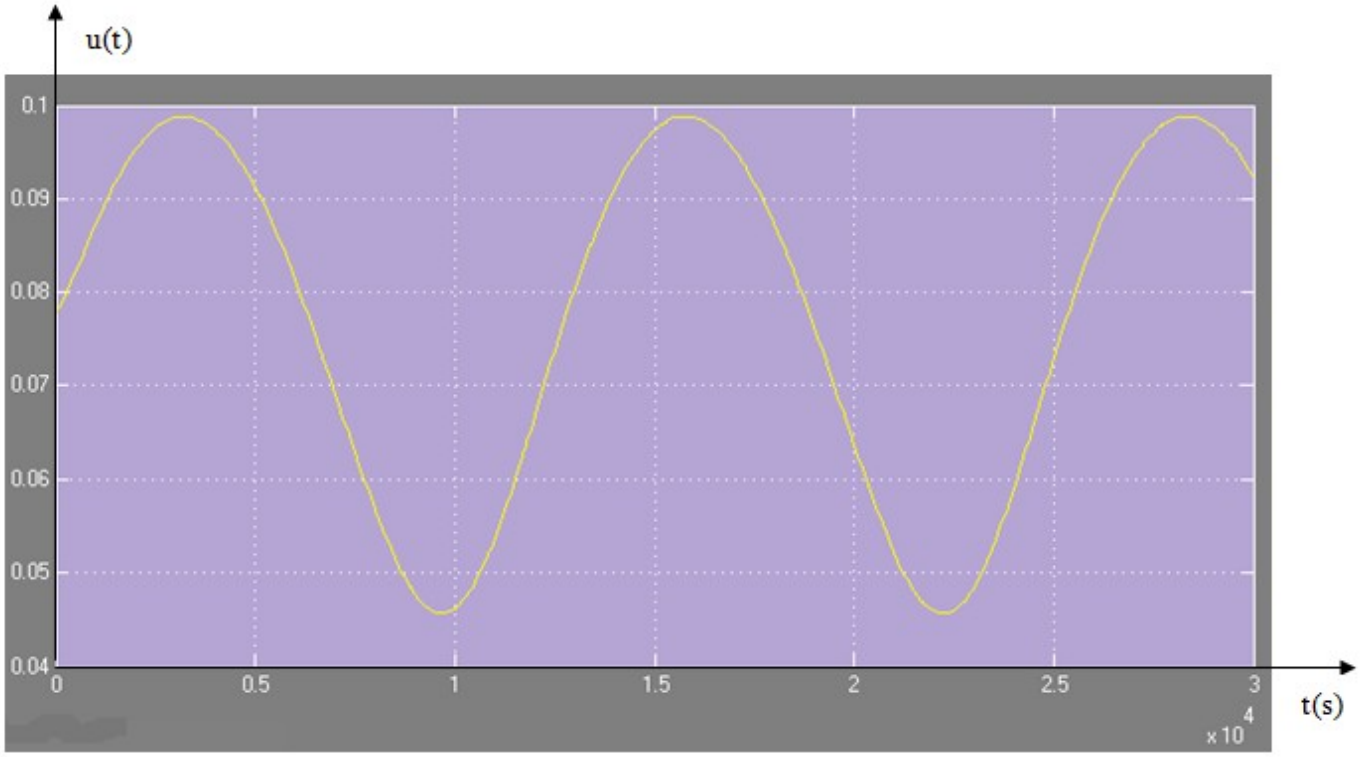


Fig.8.2. évolution de signal de commande

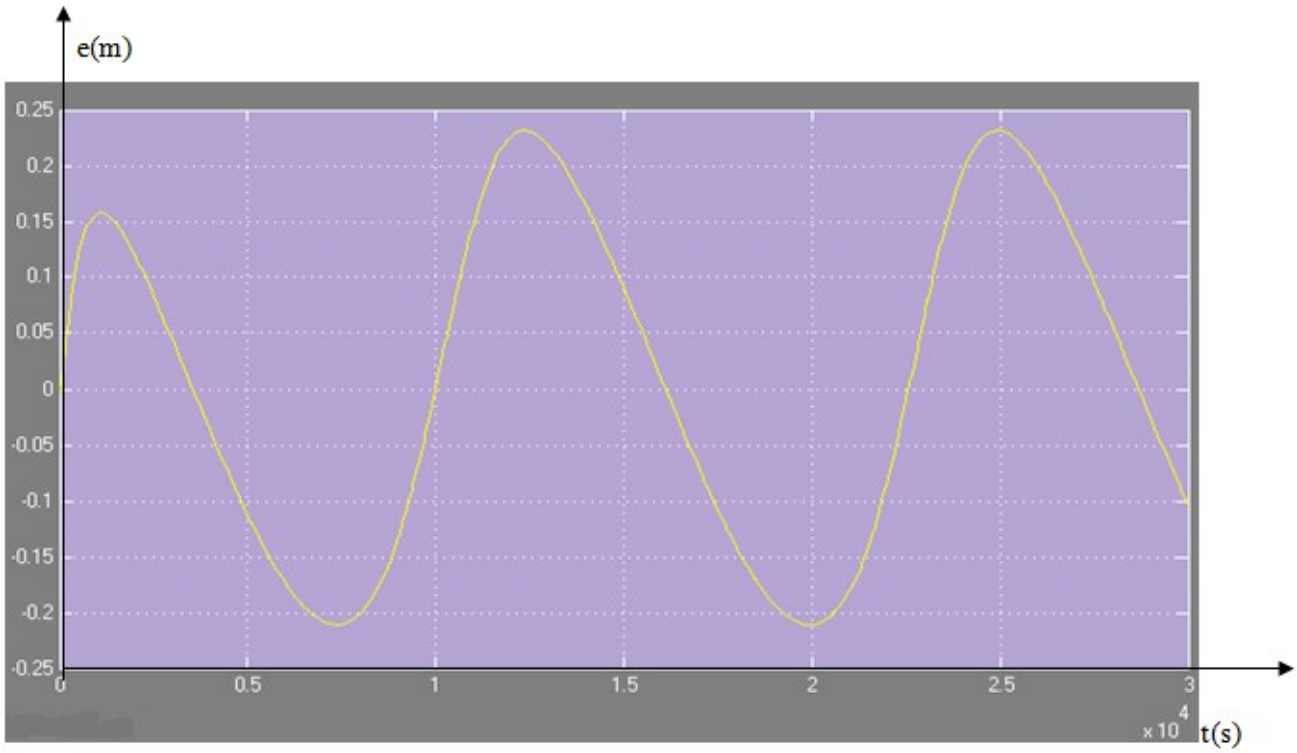


Fig.8.3. évolution de l'erreur

IV.3.4. conclusion

Les résultats obtenus avec la commande par PID flou sont très satisfaisants ; mais l'inconvénient de cette méthode c'est le non existence d'une méthode systématique pour le réglage des paramètres de PID.

IV.4. Conception de contrôleur ANFIS.

Pour la programmation de contrôleur ANFIS on utilise l'editor ″ANFIS″ du logiciel Matlab la version 7.0

Pour réaliser ce contrôleur on passe par les étapes suivantes.

- Choisir le type d'architecture de commande a utiliser ; dans cette application on utilise le modèle inverse.
- Définir les entrées et sorties de contrôleur
    - L'entrée est le niveau de liquide désiré dans le réservoir.
    - La sortie c'est le débit $q_{in}(t)$ correspondant au liquide désiré
- Définir les univers de discours des deux variables d'entrée et de sortie.
- Générer les fonctions d'appartenances des deux variables
- Définir la base de règles de contrôleur
- Générer la base de données d'apprentissage et la base de données de validation.
- Choisir les méthodes d'inférences et de deffuzzification
- Choisir le type de l'algorithme d'apprentissage a utiliser.
- définir le nombre d'époques d'apprentissage.
- Et en fin lancer l'apprentissage.
- Valider le modèle en utilisant la base de données de validation.

IV.4.1.Modèle inverse de processus

L'apprentissage utilisé est supervisé

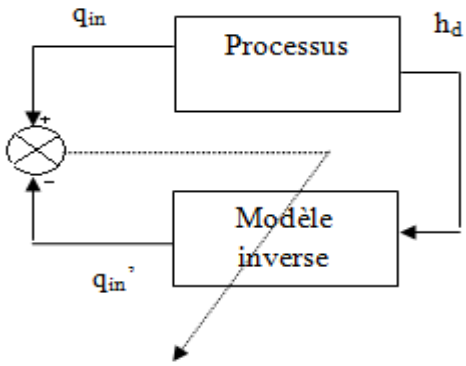


Fig.9. phase d'apprentissage

La figure suivante représente le structure de l'ANFIS utilisé dans l'application c'est un réseau a une seule entrée (niveau de liquide) et une sortie (débit d'entrée).

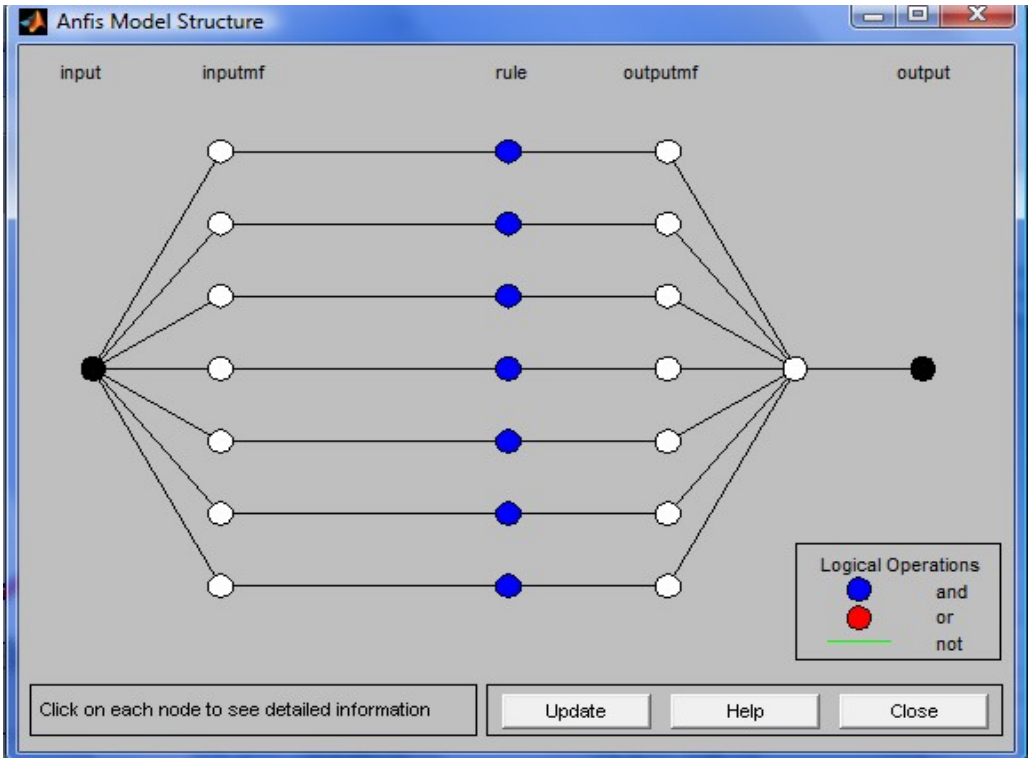
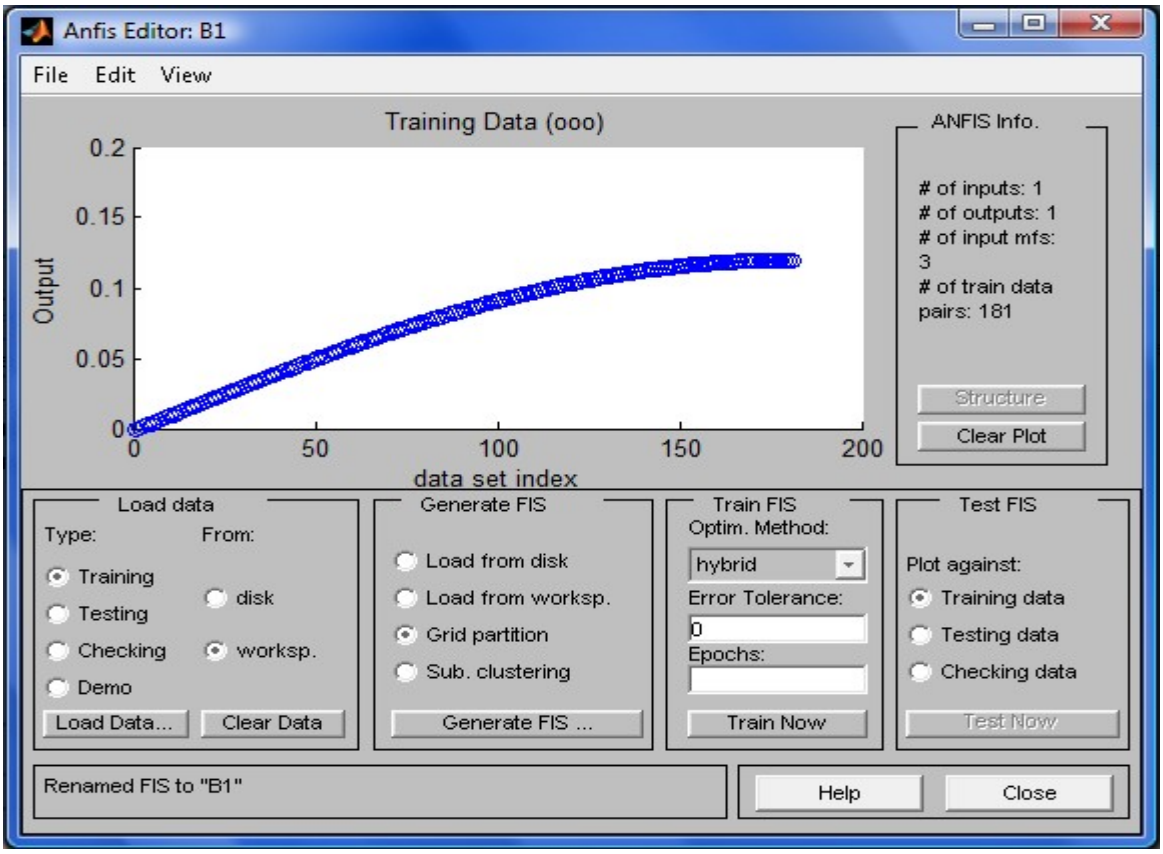
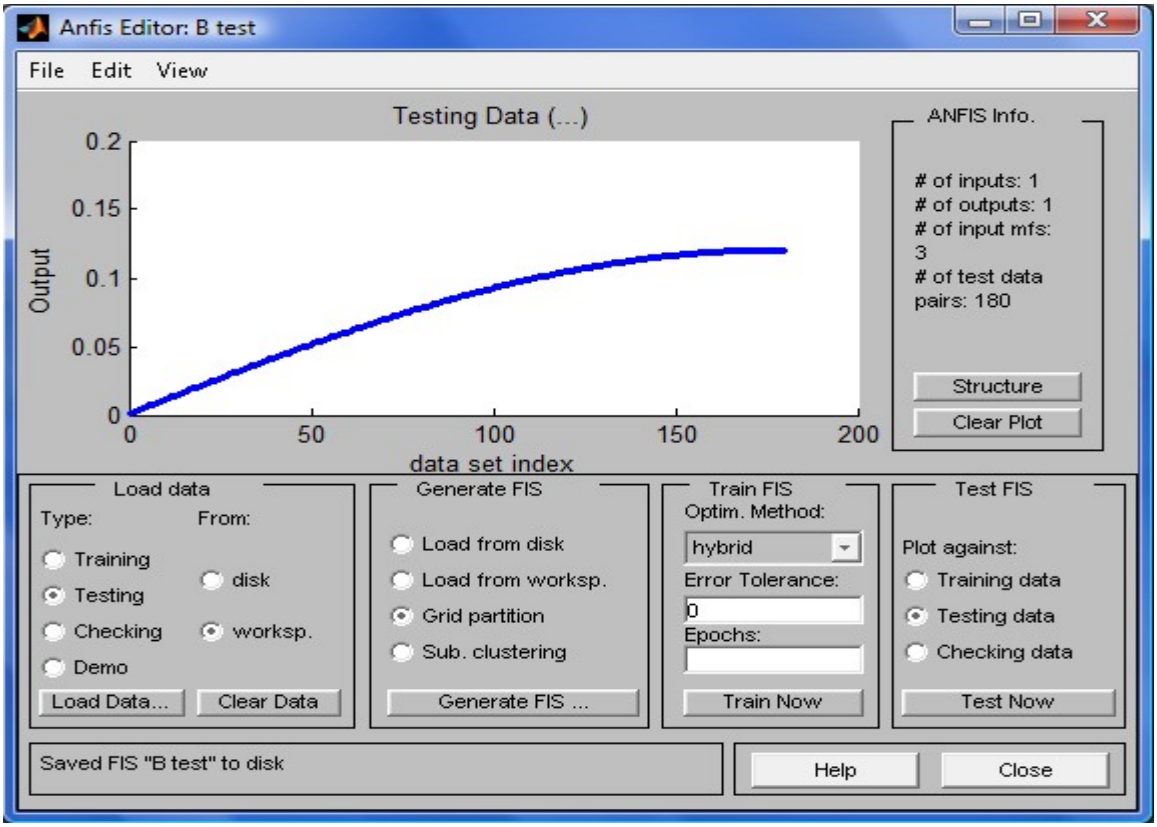


Fig.9.1.structure de réseau ANFIS a une entrée et une sortie

IV.4.2.la base d'apprentissage



IV.4.3. La base de test



IV.4.4. résultats après 2000 époques d'apprentissage

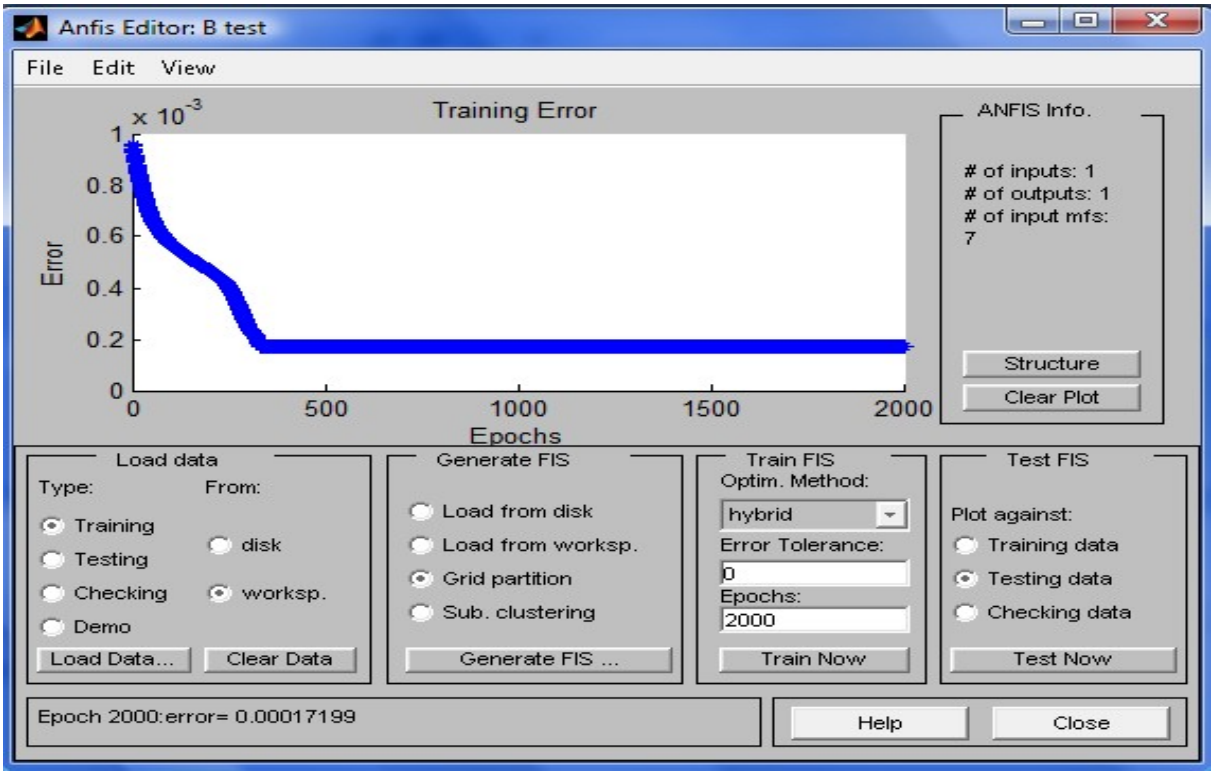


Fig.10.1. évolution de l'erreur en fonction de nombres d'époques d'apprentissage

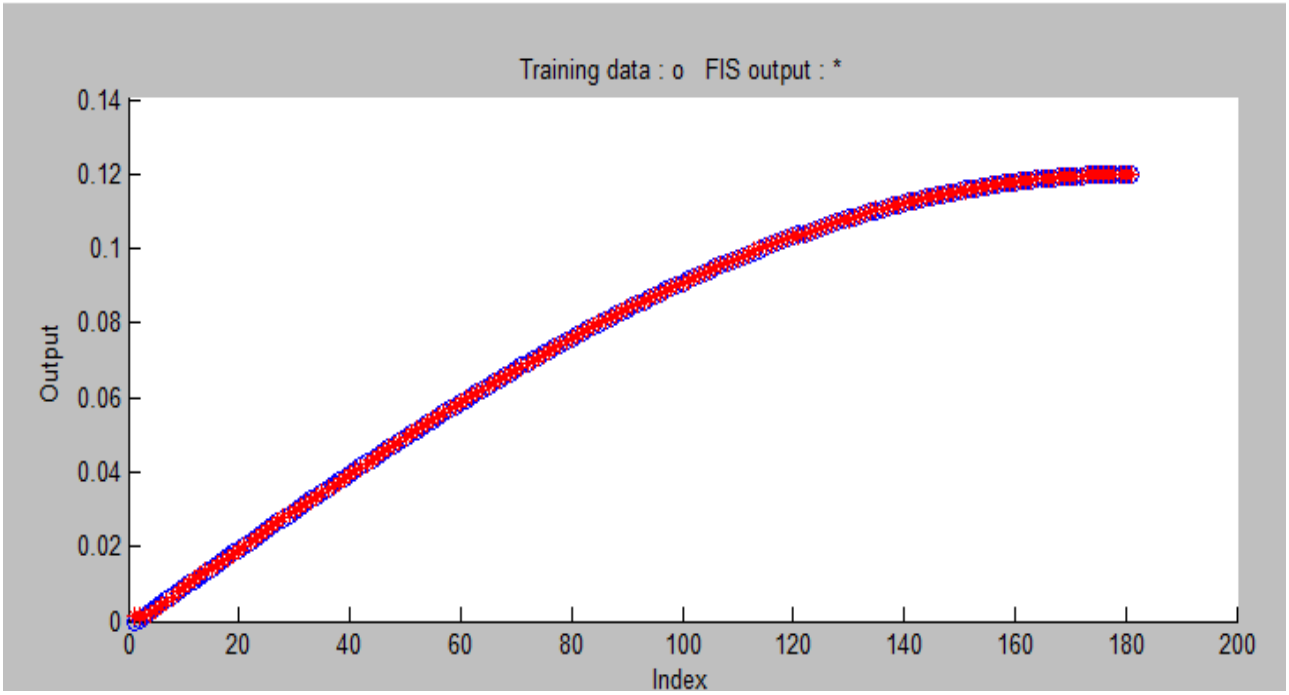


Fig.10.2. réponse du réseau après 2000 époques d'apprentissage

IV.4.5. Fonctions d'appartenances des deux variables

L'univers de discours de la variable d'entrée varie entre 0m et 7.4 m, la figure suivante montres les fonctions d'appartenances de h avant apprentissage

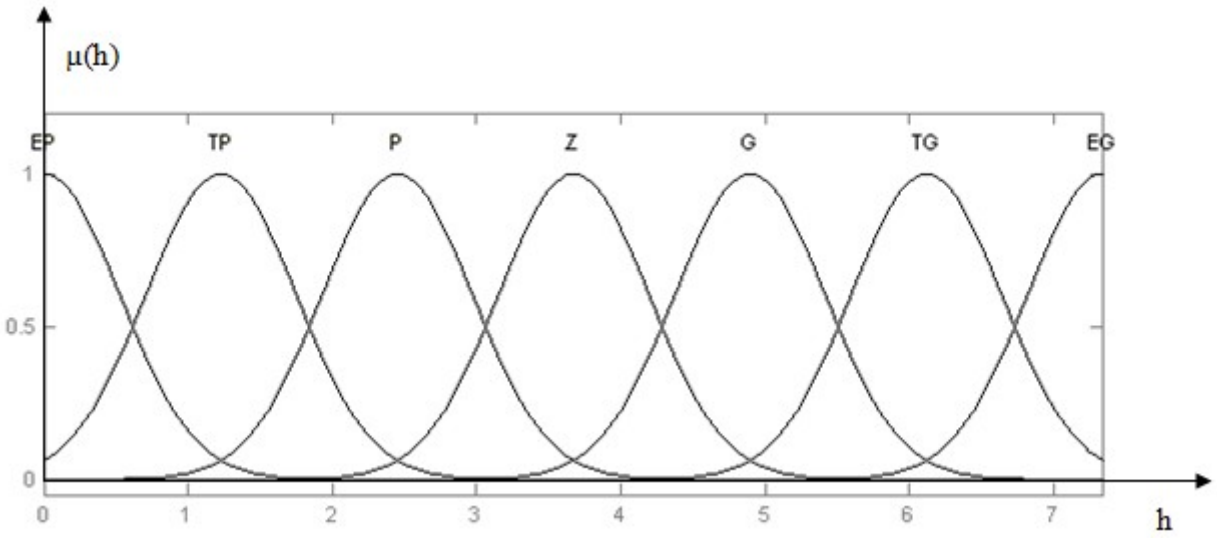


Fig.3. fonctions d'appartenance de la variable d'entrée avant apprentissage

L'univers de discours de la variable de sortie varie entre 0m et 0.12m^3/s, ses fonctions d'appartenances sont de type linéaire. Il est a signaler que les fonction de sortie de type Sugeno sont soit constantes, soit linéaires.

EP : extrêmement petit.
TP : très petit
P : petit
Z : zéro
G : grand
TG : très grand
EG : extrêmement grand

IV.4.6. les paramètres des prémisses et des conséquentes avant apprentissage

IV.4.6.a. Les paramètre des prémisses

La première valeur représente la pente de la gaussienne et la deuxième valeur représente le centre de la gaussienne.

EP	[0.5191 ; 0]
TP	[0.5191 ; 1.223]
P	[0.5191 ; 2.445]
Z	[0.5191 ; 3.668]
G	[0.5191 ; 4.889]
TG	[0.5191 ; 6.112]
EG	[0.5191 ; 7.334]

IV.4.6.b. paramètres des conséquentes

$$q_{in} = a\ h(t) + b$$

La première valeur c'est ″a″ la deuxième c'est ″b″

EP	[0 ; 0]
TP	[0 ; 0]
P	[0 ; 0]
Z	[0 ; 0]
G	[0 ; 0]
TG	[0 ; 0]
EG	[0 ; 0]

IV.4.7. la base de règles

Si h est EP Alors q_{in} est EP

Si h est TP Alors q_{in} est TP

Si h est P Alors q_{in} est P

Si h est Z Alors q_{in} est Z

Si h est G Alors q_{in} est G

Si h est TG Alors q_{in} est TG

Si h est EG Alors q_{in} est EG

IV.5. résultats d'apprentissage

Les figures suivantes montrent l'évolution de l'erreur entre la sortie de l'ANFIS et celle de la base d'apprentissage en fonction de nombre d'époques d'apprentissage.

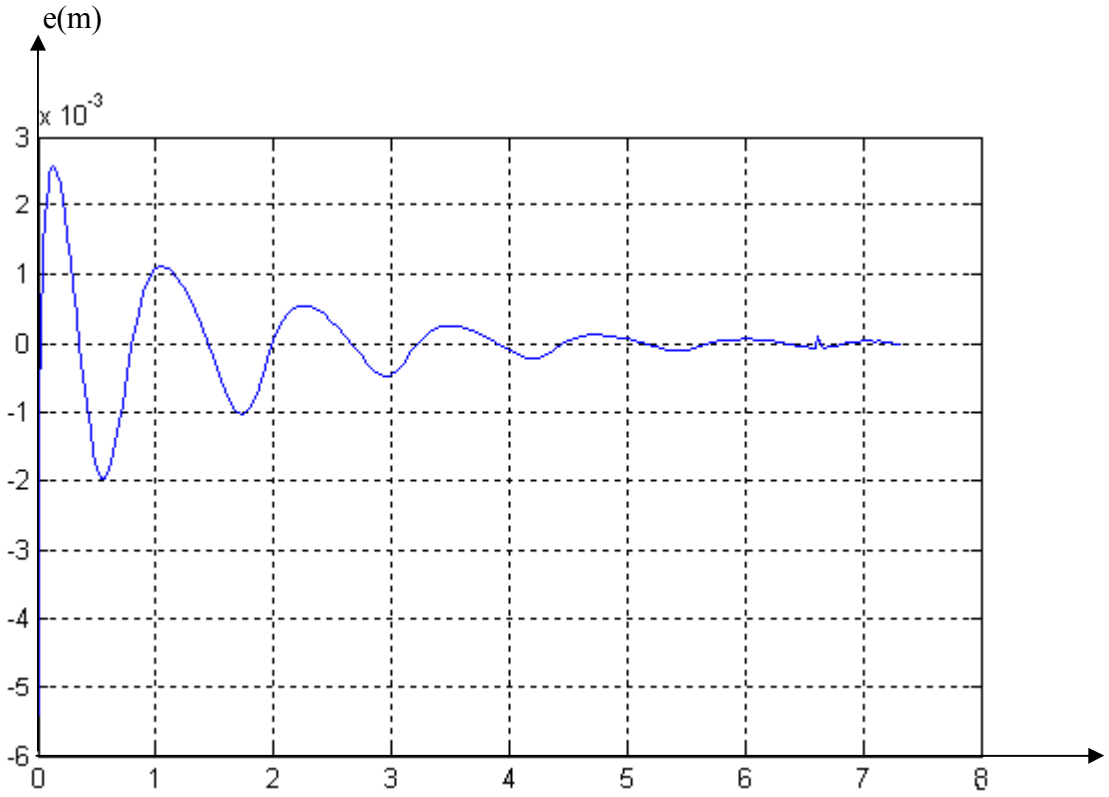


Fig.4.1. erreur après une époque d'apprentissage

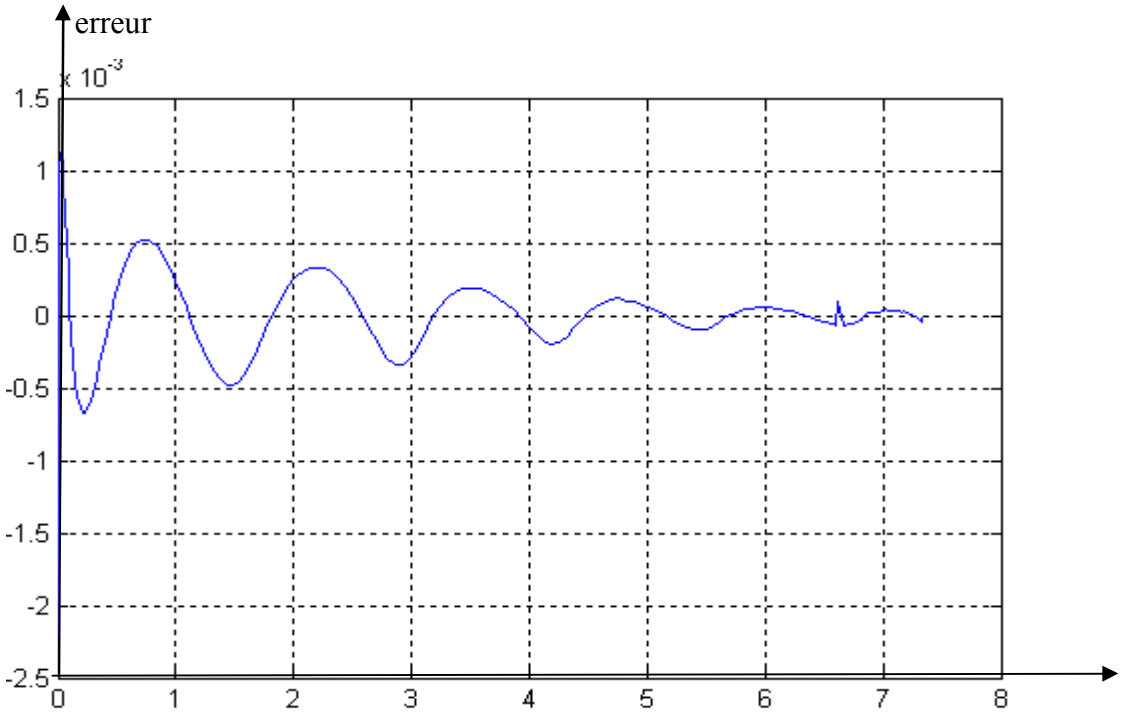


Fig.4.2. l'erreur après 50 époques d'apprentissage

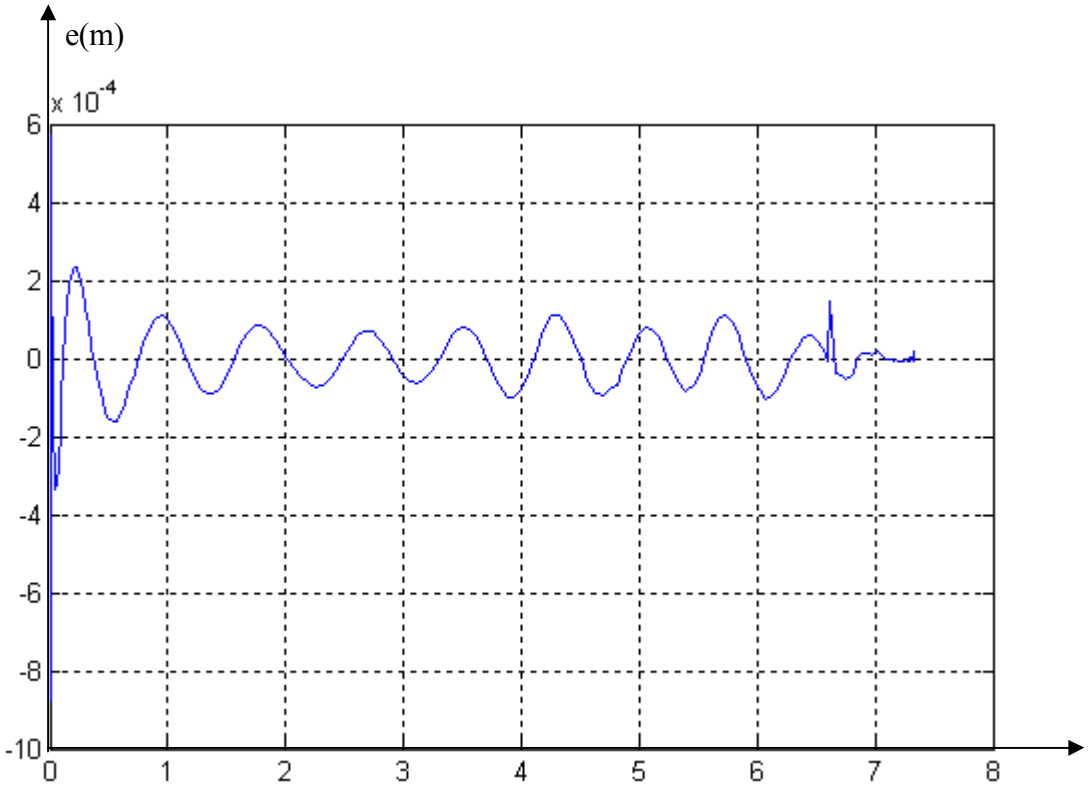


Fig.4.2. l'erreur après 500 époques d'apprentissage

Après 500 époque d'apprentissage (fin d'apprentissage) on a obtenu les résultats suivant, qui sont les modifications apportées sur les structures des fonctions d'appartenances de la variable d'entrée et de la variable de sortie.

IV.5.1.Les paramètres des prémisses après apprentissage.

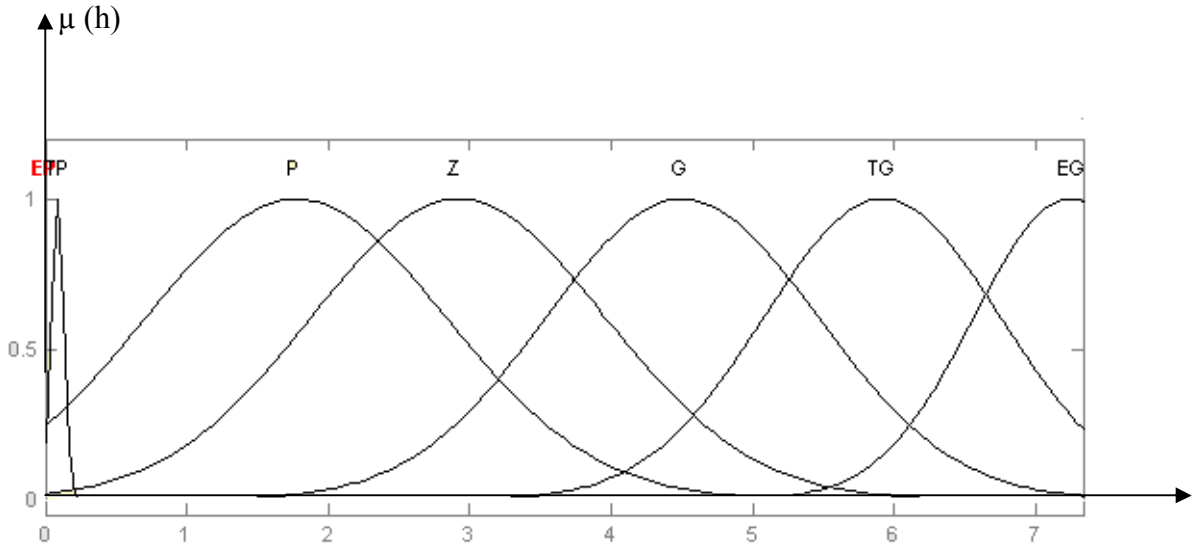


Fig.5.fonctions d'appartenance de la variable d'entrée aprés apprentissage

EP [0.4114; -1.212]

TP [0.0596; 0.0799]

P [1.054; 1.772]

Z [1.028; 2.828]

G [0.9715; 4.498]

TG [0.846; 5.91]

EG [0.6822; 7.231]

IV.5.2.Les paramètres des conséquentes après apprentissage.

EP [0.2397; -0.4451]

TP [0.05967; 0.009363]

P [0.04326; 0.02701]

Z [0.05079; -0.1109]

G [0.02876; -0.04339]

TG [0.01662; 0.00257]

EG [0.01013; 0.04401]

IV.6.phase d'exploitation du contrôleur

L'étape d'exploitation consiste à commander le système avec le contrôleur ANFIS développé

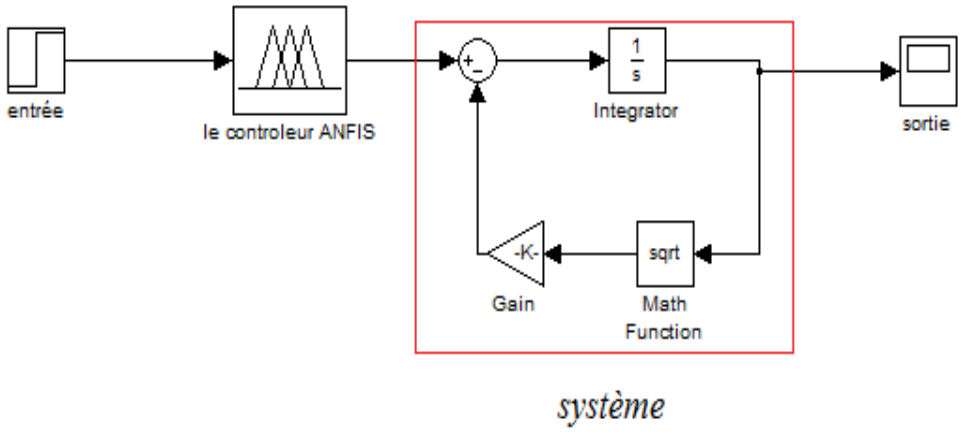


Fig.6. phase d'exploitation de contrôleur ANFIS

Le système est commandé en boucle ouverte.

La simulation est réalisée à l'aide de la boite à outil Simulink du logiciel Matlab version 7.0.

IV.6.1.Test en régulation

On veut réguler le niveau de liquide a 5m pour cela on soumit le système a un échelon de 5m.

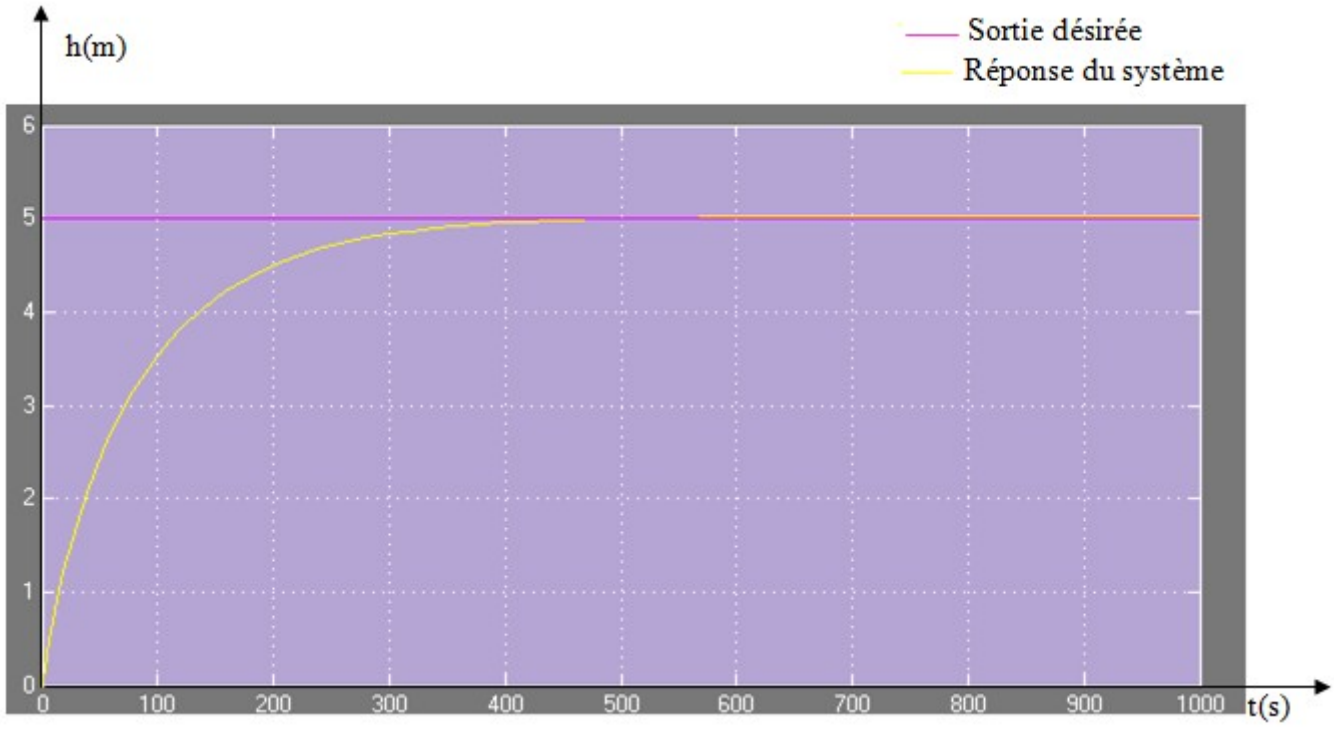


Fig.7.1.Réponse du système pour un échelon de 5m

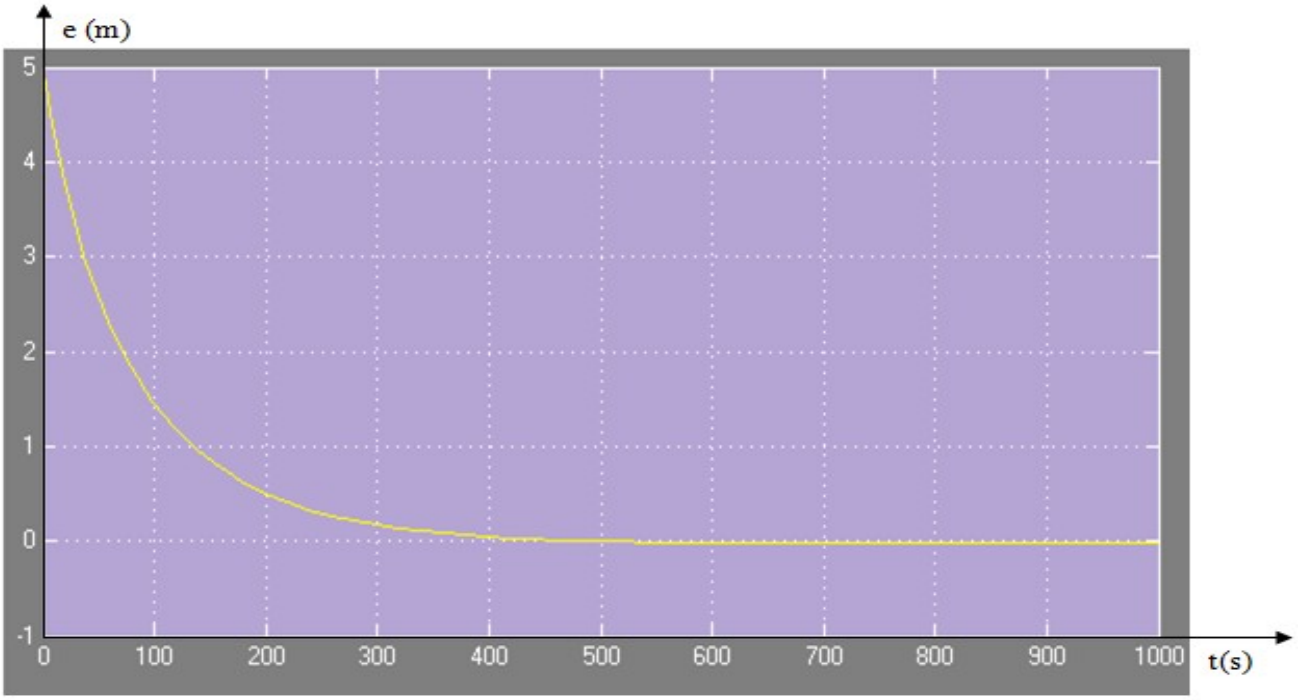


Fig.7.2. évolution de l'erreur

IV.6.2.Test en poursuite de trajectoires

- Test de la première trajectoire

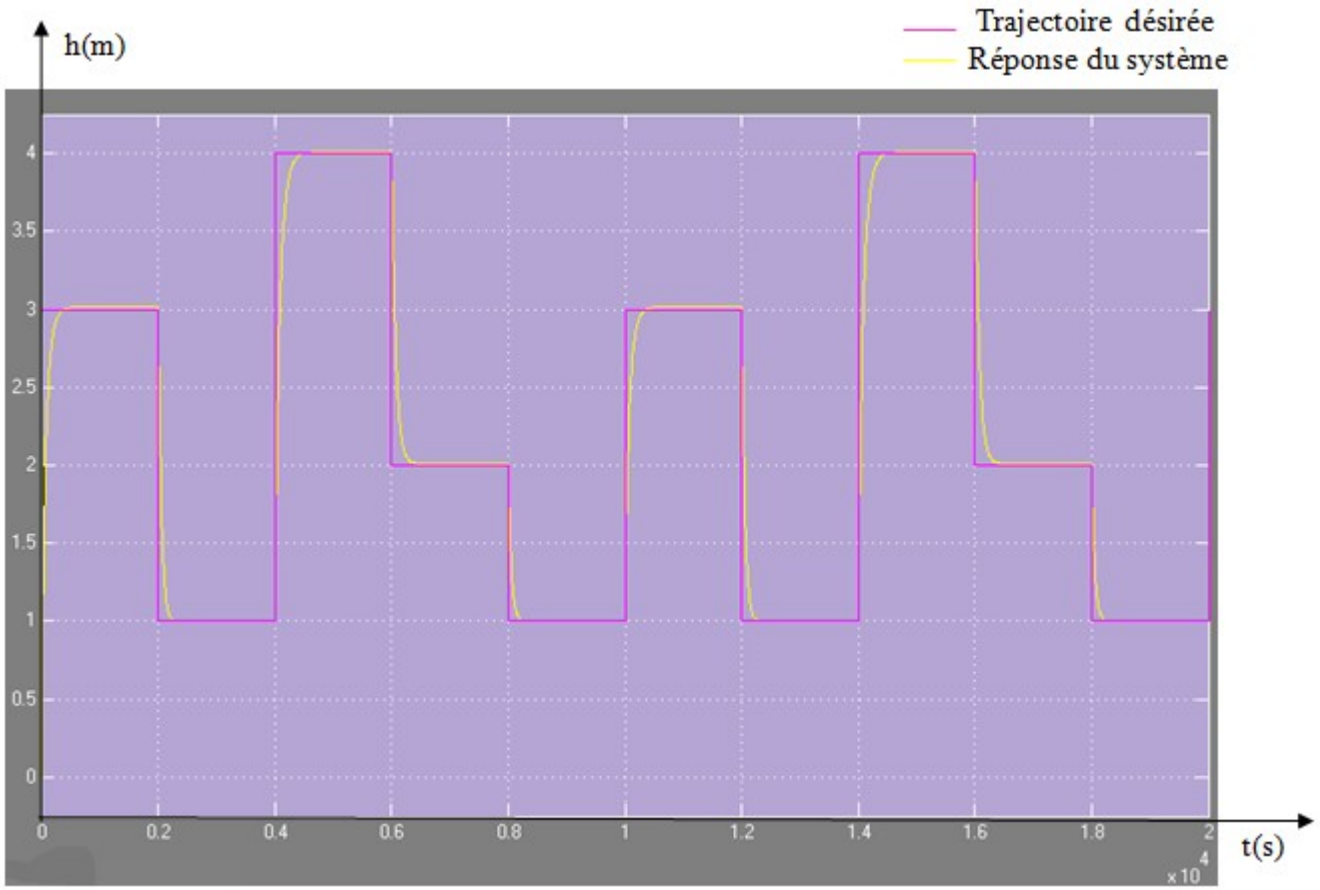


Fig.8.1. réponse du système à un signal en forme d'escalier

Fig.8.2.évolution du signal de commande

Fig.8.3.évolution de l'erreur

- Test de la deuxième trajectoire

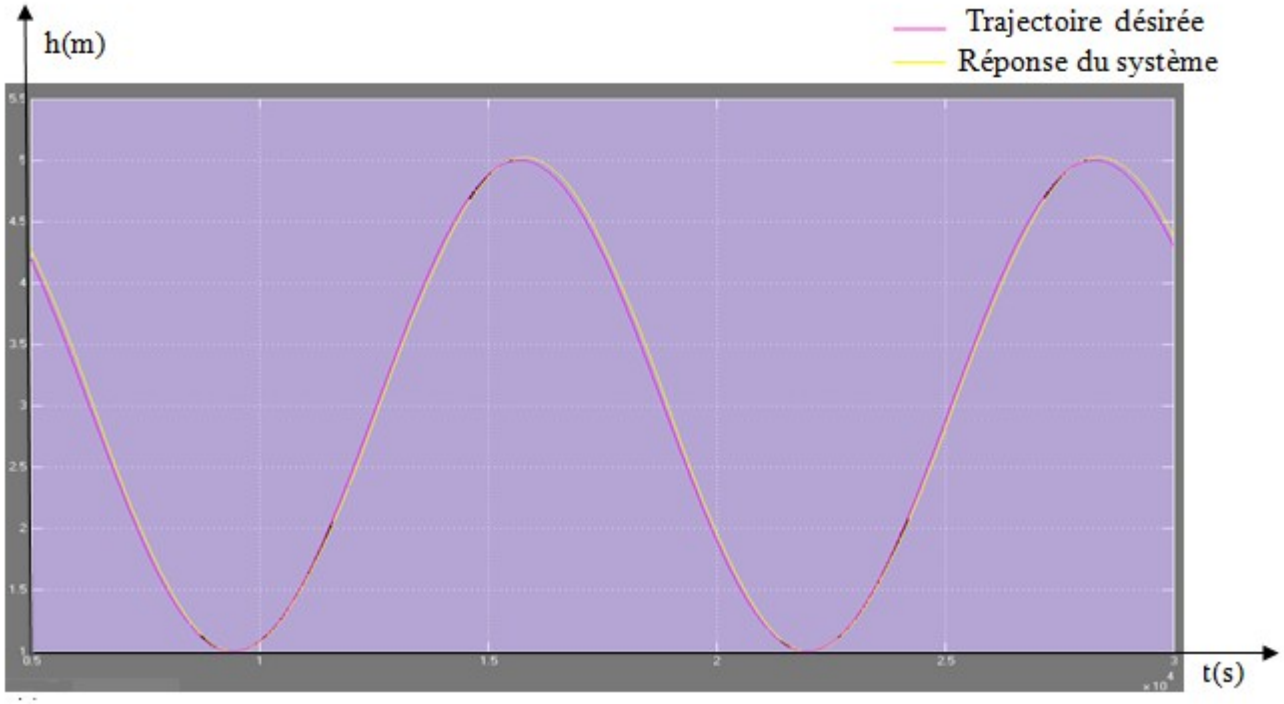
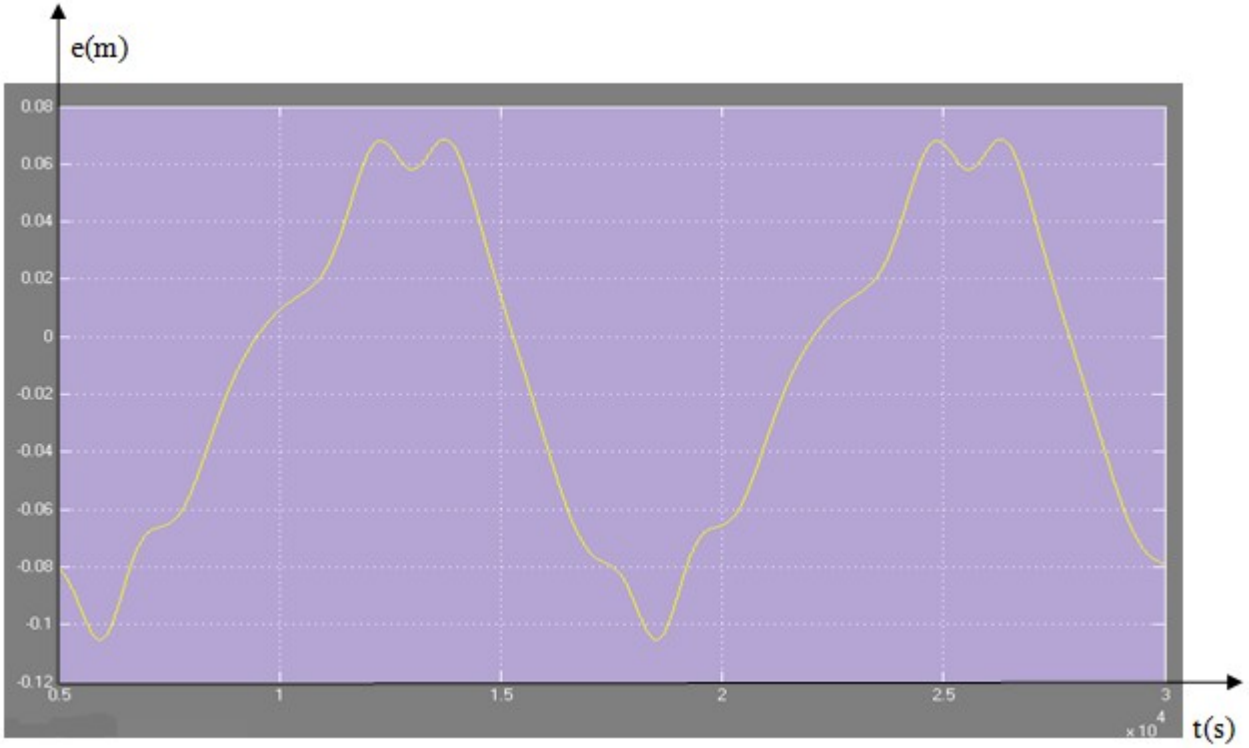


Fig.9.1. réponse du système pour un signal sinusoïdal

Fig.9.2.évolution du signal de commande

Fig.9.3.évolution de l'erreur



IV.7.conclusion

Dans ce chapitre nous avons vu l'application de deux stratégies de commande la première c'est la commande par PID flou et la deuxième par le contrôleur neuro-flou proposée dans le chapitre précédent.

Les résultats obtenus s'avèrent très satisfaisants pour les deux méthodes et montrent clairement la bonne poursuite du système aux trajectoires désirées.

Conclusion générale

la commande en logique floue permet de s'abstenir de l'utilisation des modèles mathématiques parfois difficiles a obtenir.sa capacité a traduire la connaissance d'un opérateur humain en règles d'expertise énoncées dans un langage simple .en fait une technique très prometteuse.

Les résultats obtenus en appliquant cette commande au système non linéaire mono variables s'avèrent très satisfaisants

Mais lorsque le nombre de variables entrant en jeu devient trop important la base de règles explose très vite, et des problèmes liés à sa réalisation pratique en découlent. Ce travail s'inscrit dans la mouvance des travaux actuels sur la commande floue et s'attache aux problèmes d'optimisation des paramètres des systèmes d'inférence flous.

Dans une première partie, les principes de base de la logique floue et de la commande floue sont rappelés. Dans une deuxième partie, des solutions visant à simplifier la synthèse d'un contrôleur flou sont présentées.une foi la structure de contrôleur flou défini. Le nombre de paramètres à régler pouvant atteindre un nombre impressionnant, il est intéressant d'utiliser des techniques d'apprentissage afin d'automatiser la mise au point de contrôleur flou.

Les paramètres de ce dernier (fonction d'appartenance, en entrée et en sortie) sont ici réglés a travers la méthode des systèmes hybrides neuro-floue (ANFIS).

La démarche proposée dans le troisième chapitre est appliquée avec succès à la commande du processus non linéaire.

Bibliographie

[1]: Fuller Robert, "Introduction to neuro-fuzzy systems ".

[2]: Aleksander Igor, "Introduction a la conception des systèmes intelligents ".

[3]: Dreyfus .G, "Réseaux de neurones : méthodologie et application ".

[4]: Foullay Laurent, "commande floue : de l'approximation a l'apprentissage ".

[5]: Karray Fakhredinne et Clarence De Silva, "Soft computing and intelligent systems design theory, tools and application ".

[6]: Perzonnaz Léon, "Réseaux de neurones formels pour la modélisation, la commande et la classification ".

[7] : Biran Adrian , " Matlab pour l'ingénieur versions 6 et 7 ".

[8] : Pierre Yves Glorennec, " Algorithmes d'apprentissage pour systèmes d'inférence floue ".

[9] : le mémoire de fin d'étude de Oukaci. M et Sadali.S, promotion 2005/2006 "commande neuro-floue d'un systéme non linéaire".

[10] : le mémoire de fin d'étude de Oukaci. M et Sadali.S, promotion 2005/2006 "commande neuro-floue d'un systéme non linéaire".

[11] : le mémoire de fin d'étude de Oukaci. M et Sadali.S, promotion 2005/2006 "commande neuro-floue d'un systéme non linéaire".

[12] : le mémoire de fin d'étude de Oukaci. M et Sadali.S, promotion 2005/2006 "commande neuro-floue d'un systéme non linéaire".

[13] : le mémoire de fin d'étude de Oukaci. M et Sadali.S, promotion 2005/2006 "commande neuro-floue d'un systéme non linéaire".