Cours: Commande Intelligente

Book · December 2019

CITATIONS
0

READS
325

1 author:

Radhwane Sadouni
Université de Ghardaia
15 PUBLICATIONS **42** CITATIONS

SEE PROFILE

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Ghardaia

Faculté des Sciences et de la Technologie

Département d'Automatique et d'Electromécanique

Cours

Commande Intelligente

Spécialité:
Master 2_Automatique et Systèmes

Par:
Dr. Radhwane SADOUNI

Avant-propos

Ce polycopié est destiné aux étudiants de la deuxième année Master Automatique, en respectant le programme officiel contenu au canevas du ministère de l'enseignement supérieur et de la recherche scientifique.

L'objectif étant de mettre le point sur les techniques de commande basées sur les outils de l'intelligence artificiels, afin de donner une base théorique indispensable à la compréhension de ces approches et à leurs utilisations dans le domaine de la commande des processus industriels.

Ce polycopié comporte deux parties essentielles à savoir:

- **Partie I**: Logique floue
- **Partie II:** Réseaux de neurones

De nombreuses applications (des exemples de simulations) ont été introduites dans ce polycopié afin de permettre aux étudiants de mieux comprendre les principes fondamentaux de la commande intelligente.

Partie I: Logique floue

1. Introduction:

De nos jours, la logique floue (en anglais fuzzy logic) est un axe de recherche important sur lequel se focalisent de nombreux chercheurs. Des applications technologiques sont maintenant disponibles, tant dans le domaine grand public (appareils photos, machines à laver, fours à micro-onde, … etc.), que dans le domaine industriel (réglage et commande de processus complexes liés à l'énergie, aux transports, à la transformation de la matière, à la robotique, aux machines-outils).

La logique floue est une logique qui substitue à la logique binaire une logique fondée sur des variables pouvant prendre, outre les valeurs (vrai 1) ou (faux 0), les valeurs intermédiaires «vrai» ou «faux» avec une certaine probabilité (un certain degré d'appartenance).

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

2. Bref historique:

Les origines de la logique floue se trouvent dans le principe de l'incertitude de Heisenberg dans les années 20, les physiciens ont introduit la troisième valeur ½ dans le système logique bivalent {0, 1}. Au début des années 30, le logicien polonais Jan Lukasiewicz a développé le système logique avec trois valeurs.

En ce qui suit, un bref historique sur la parution et les premières applications de la logique floue:

1965: Le Professeur: **L.A. Zadeh** de l'Université de Berkeley (Californie) pose les bases théoriques de la logique floue.

1973: L.A. Zadeh propose d'appliquer la logique floue aux problèmes de réglage.

1974: Première application du réglage par la logique floue appliquée à une turbine à vapeur. Suivie en 1980 par une application sur un four à ciment et en **1983** sur un épurateur d'eau.

1985: Premiers produits industriels (Japon) utilisant le principe de la logique floue appliqué à des problèmes de réglage et de commande.

3. Logique classique et logique floue:

Pour exprimer des informations dites incertaines ou imprécises, on se sert de certaines formes linguistiques telles que: grand, petit, très, peu, moins, chaud, froid… etc. La logique classique ne peut traiter que deux états: vraie ou faux, elle se trouve incapable de représenter convenablement ces formules linguistiques, c'est là où intervient la logique floue.

L'importance de la logique floue réside dans le fait qu'elle permet par sa méthode proche du raisonnement humain de représenter des informations inexactes ou imprécises provenant d'une connaissance globale du système.

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

3.1. Exemple:

Prenant l'exemple d'un récipient contenant de l'eau dont la température variée entre 0°C et 40°C, une personne qui va toucher le récipient sans qu'elle sait exactement sa température, peut dire s'il est chaud, froid, très chaud ou très froid, mais comment peut on représenter ces résultats ?

Si on tient compte de la logique classique, le problème peut être traité de la manière suivante:

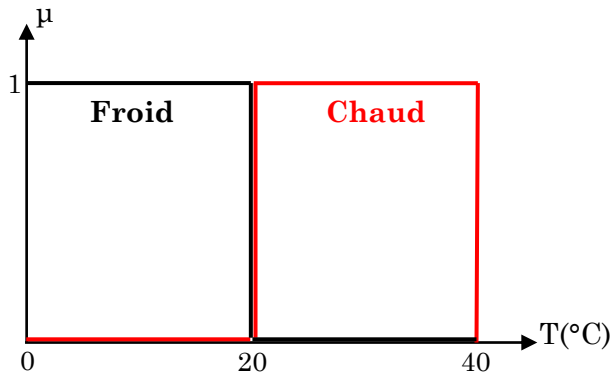


Fig. I.1: *Classification selon la logique classique*

D'après cette représentation, toute valeur de température inférieure à 20°C est considérée comme froide, et s'elle qui est supérieure à cette valeur est considérée chaude.

Cependant, une telle logique de classification n'est même pas une logique. D'abord pourquoi la valeur 19,9°C doit être considérée froide, tandis que la valeur

20,1°C est chaude? Alors que en réalité que ces deux valeurs sont presque les mêmes.

La logique floue envisage le problème d'une autre façon:

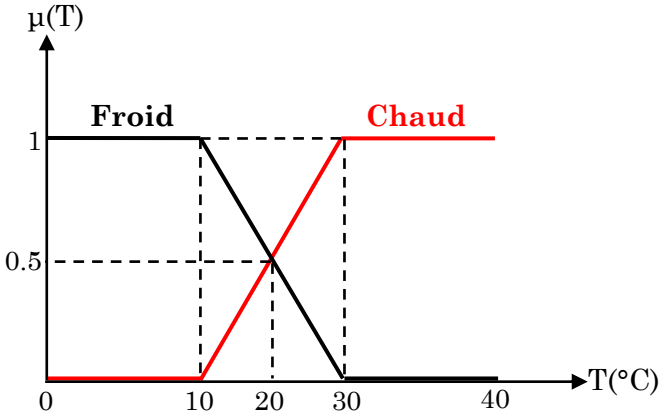


Fig. I.2: *Classification selon la logique floue*

Cette représentation indique que la valeur de 20°C est considérée «froid» avec un degré d'appartenance μ= 0,5 (50%) à la fonction d'appartenance «froid», et «chaud» avec le même degré d'appartenance (même taux de présence). Ainsi, la valeur 30°C considérée «chaud» à μ=1 et «froid» à μ=0.

Remarque: Le concept de sous ensemble flou a été introduit pour éviter le passage brusque d'une classe à une autre (de la classe «froid» à la classe «chaud» par exemple) et autoriser des éléments à n'appartenir complètement ni à l'une ni à l'autre (à être tiède, par exemple).

4. Principe de la commande floue:

Lorsqu'un opérateur humain commande manuellement un système, les actions qu'il réalise sont dictées par une connaissance subjective du fonctionnement de ce système. Par exemple, si l'eau est chaude dans une piscine, on la refroidit, si elle est très chaude on la refroidit plus. Cette commande du système peut être envisagée de façon différente selon la personne qui la réalise.

Ce principe est la base de la commande floue. La mesure «température» réalisée sur le système est prise en compte par l'intermédiaire d'une variable linguistique («froid», «tiède», «chaud»), qui est issue de l'analyse d'un expert humain. Ensuite, l'action à réaliser est déduite à la fois d'un ensemble de règle de commande («s'il fait froid, on chauffe plus»…) et de l'état du système, qualifiée par la variable linguistique.

5. Structure d'un contrôleur flou:

La commande floue est l'application la plus utilisée de la logique floue. La structure générale d'un contrôleur flou est montrée sur la figure suivante:

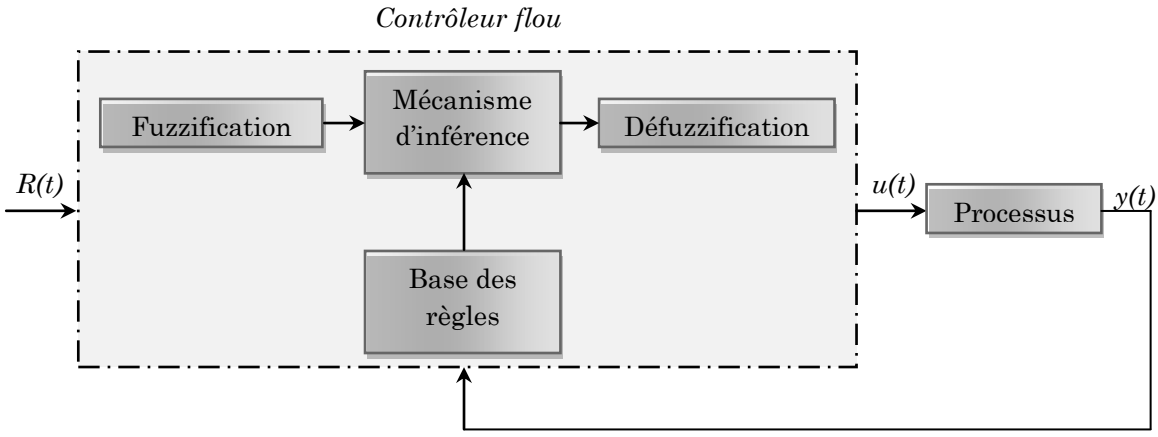


Fig. I.3: *Structure générale d'un contrôleur flou*

R(t): est le signal de référence.

u(t): est le signal de commande.

y(t): est la sortie du système à commander.

Le contrôleur flou comporte essentiellement quatre parties; un bloc de fuzzification, une base des règles, un mécanisme d'inférence et un bloc de défuzzification.

5.1. Bloc de fuzzification: il sert à transformer les variables numériques non floues provenant des entrées en variables linguistiques floues. La fuzzification consiste à définir les fonctions d'appartenance pour les différentes variables d'entrées et de sortie. Dans le cas de réglage par la logique floue, on utilise en général des formes trapézoïdales et triangulaires pour les fonctions d'appartenances.

Comment fuzzifier une variable numérique ?

Pour fuzzifier une variable numérique, il faut donner:

- L'univers de discours, c-à-d la plage de variation possibles des entrées et des sorties.
- Une répartition en classe floue de cet univers.
- Les fonctions d'appartenance de chacune de ces classes.

Exemple:

La figure ***(Fig. I.4)*** illustre un exemple de fuzzification d'une variable x en cinq (5) sous-ensembles flous.

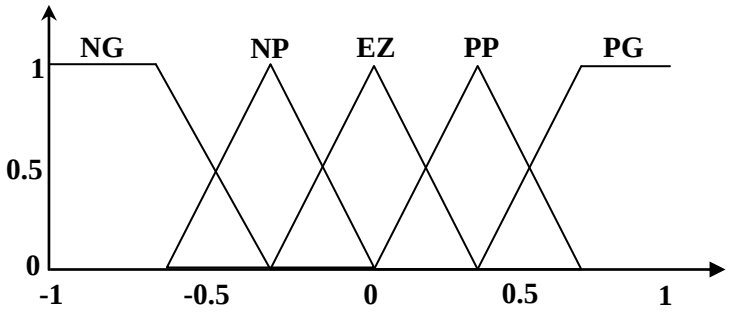


Fig. I.4: *Fuzzification en 5 sous-ensembles flous*

Les différents ensembles flous sont caractérisés par des désignations standards: la signification des symboles et par exemple:

NG: Négative Grande.

NP: Négative Petite.

EZ: Environ Zéro.

PP: Positive Petite.

PG: Positive Grande.

5.2. Base des règles et mécanisme d'inférence: à partir de la base des règles fournie par l'expert et les sous-ensembles flous des entrées et de sorties correspondants à la fuzzification, le mécanisme d'inférence effectue une relation entre les fonctions d'appartenances des entrées et les fonctions d'appartenances de sorties. En appliquant des règles de type: **«si condition alors conclusion»**. Le bloc d'inférence est le cœur d'un régulateur flou, il possède la capacité de simuler les décisions humaines et de déduire (inférer) les actions de commande floue. Une représentation graphique de l'ensemble des règles, appelée matrice d'inférence ou table des règles, permet de synthétiser le cœur de régulateur flou.

Le tableau ***(Tab. I.1)*** représente une table d'inférence avec cinq sous-ensembles flous pour deux variables d'entrée à savoir: l'erreur ***'e'*** et sa dérivée ***'de'***, et une variable de sortie: la commande ***'du'***.

L'inférence ou la prise de décision est le noyau du contrôleur flou. Elle a l'aptitude de simuler la prise de décision de l'être humain en se basant sur les concepts flous et l'expertise. Pour le réglage par logique floue, on utilise en général l'une des trois méthodes suivantes:

Tab. I.1: *Table d'inférence avec cinq sous-ensembles flous*

du		***E***				
		NG	*NP*	*EZ*	*PP*	*PG*
de	*NG*	*NG*	*NG*	*NP*	*NP*	*EZ*
	NP	*NG*	*NP*	*NP*	*EZ*	*PP*
	EZ	*NP*	*NP*	*EZ*	*PP*	*PP*
	PP	*NP*	*EZ*	*PP*	*PP*	*PG*
	PG	*EZ*	*PP*	*PP*	*PG*	*PG*

5.2.1. Méthode d'inférence max-min (méthode de MAMDANI): La méthode d'inférence min-max, utilise l'opérateur «ET» par la formulation du minimum. La conclusion dans chaque règle, introduite par «ALORS», qui est réalisée par la formation du minimum. Enfin l'opérateur «OU» lie les différentes règles, réalisé par la formation du maximum.

5.2.2. Méthode d'inférence max-produit (méthode de LARSEN): La méthode d'inférence max-produit, réalise l'opérateur «ET» par la formulation du produit. La conclusion dans chaque règle, introduite par «ALORS», est réalisée par la formation du produit. L'opérateur «OU» qui lie les différentes règles est réalisé par la formation du maximum.

5.2.3. Méthode de SUGENO: L'opérateur «ET» est réalisé par la formulation du minimum, la conclusion de chaque règle floue a une forme polynomiale.

<u>Exemple d'inférence floue</u>:

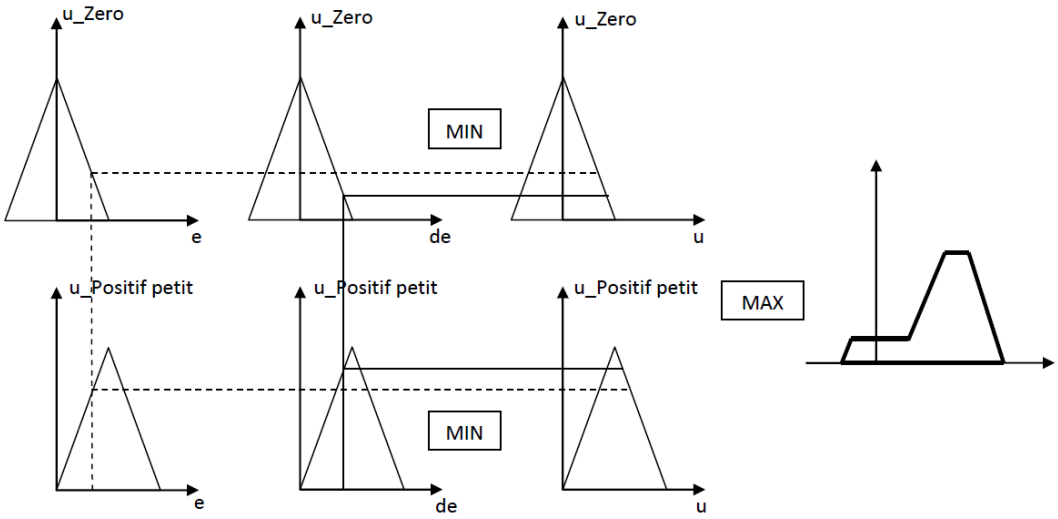


Fig. I.5: *La méthode d'inférence MAX/MIN*

5.3. Bloc de Défuzzification: Ce bloc est l'inverse de celui de fuzzification, il sert à transformer la décision floue à une valeur numérique afin de l'envoyer au système. Il existe plusieurs méthodes de défuzzification dont les principaux sont:

5.3.1. Méthode du centre de gravite:

C'est la méthode de défuzzification la plus courante. L'abscisse du centre de gravité de la fonction d'appartenance résultant de l'inférence correspond à la valeur de sortie du régulateur.

$$x_G = u = \frac{\int_{x0}^{x1} x\mu(x)dx}{\int_{x0}^{x1} \mu(x)dx} \qquad \text{(I.1)}$$

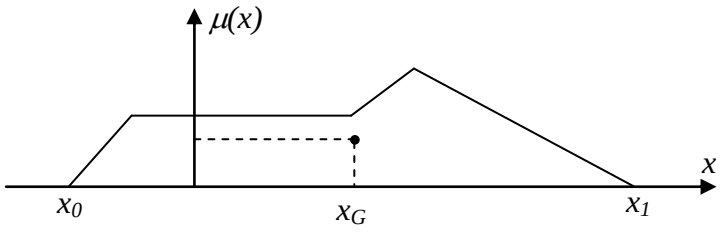


Fig. I.6: *Défuzzification par centre de gravité*

5.3.2. Méthode par valeur maximale:

Cette méthode est beaucoup plus simple, elle ne s'utilise que dans le cas discret. La valeur de sortie est choisie comme l'abscisse de la valeur maximale de la fonction d'appartenance. La figure ci-dessous illustre le principe de cette méthode.

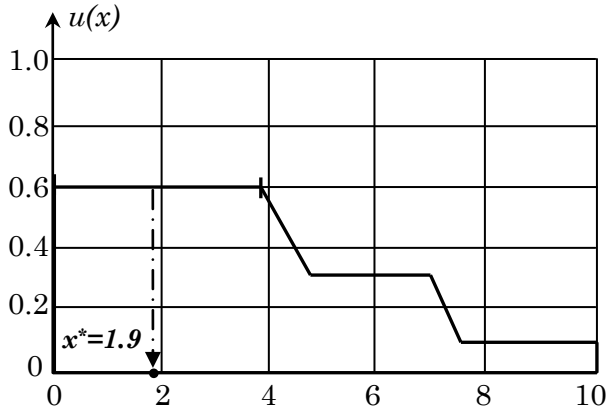


Fig. I.7: *Méthode de Défuzzification par valeur maximale*

5.3.3. Méthode de la moyenne des maximums:

Dans cette méthode, la valeur de sortie est estimée par l'abscisse du point correspondant au centre de l'intervalle pour lequel la fonction d'appartenance est maximale. Cette valeur est fournie par 1'expression:

$$y_{cm} = \frac{\inf M + \sup M}{2} \qquad \text{(I.2)}$$

Où: M est l'ensemble des points pour lesquels la fonction d'appartenance est maximale.

6. Avantages et inconvénients de la logique floue:

6.1. Avantages de la logique floue:

- La non-nécessité d'une modélisation du système à régler.
- La possibilité et la facilité d'implémenter des connaissances de l'opérateur de processus (intégration de l'expertise humaine sous forme de règles simples).
- Une solution efficace pour des problèmes complexes (fortement non-linéaire et difficile à modéliser).
- Robustesse vis-à-vis les incertitudes de modélisation.

6.2. Inconvénients de la logique floue:

- Manque de directives précises pour la conception du régulateur (choix des fonctions d'appartenance, nombre des sous-ensembles flous, l'univers de discours, …etc).
- Aucunes méthodes formelles pour l'ajustement.
- Les performances dépendent de l'expertise.
- Il n'existe pas de théorie générale qui caractérise rigoureusement la stabilité, la robustesse, …etc.

7. Exemples d'application de la logique floue dans la commande:

7.1. Réglage de la vitesse d'un moteur à courant continu:

Le schéma électrique d'un moteur à courant continu à excitation série est donné par la figure ci-dessous:

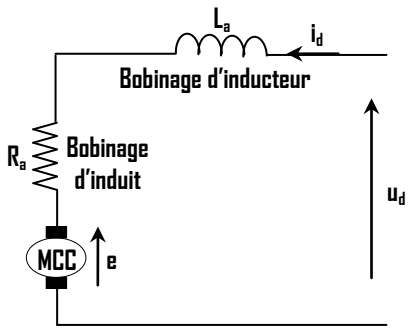


Fig. I.8: *Schéma électrique d'un MCC à excitation série*

Paramètres du moteur:

R_a=8 Ω: La résistance d'induit.

L_a=0.0597 H: L'inductance de l'inducteur.

J=0.003 kg.m²: L'inertie.

f=0.002 Nms/rd: Le coefficient de frottement.

Le schéma bloc du MCC sans réglage de la vitesse est donné par la figure suivante:

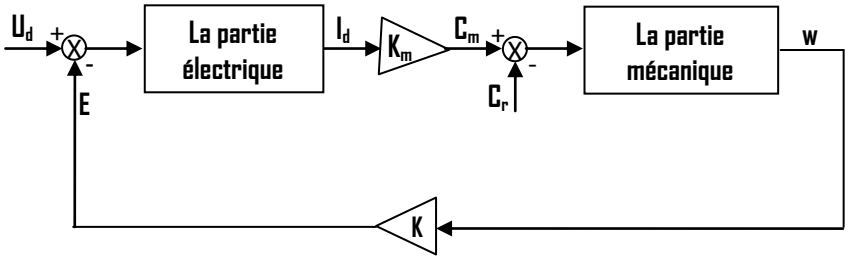


Fig. I.9: *Schéma bloc du MCC sans réglage de vitesse*

Notant que:

$C_m=K_m.I_d$

E=K.w

Modélisation de la partie électrique:

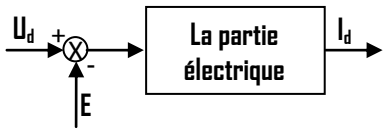
On a selon la loi des mailles:

$u_d(t) = e(t) + R_a.i_d(t) + L_a.\frac{di_d(t)}{dt}$ En appliquant la transformée de Laplace, on aura:

$$U_d(s) = E(s) + R_a.I_d(s) + L_a.s.I_d(s)$$

$$U_d(s) - E(s) = (R_a + L_a.s)I_d(s)$$

La fonction de transfert de la partie électrique:



$$G(s) = \frac{I_d(s)}{U_d(s) - E(s)} = \frac{I_d(s)}{(R_a + L_a.s)I_d(s)} = \frac{1}{(R_a + L_a.s)} = \frac{\frac{1}{R_a}}{(1 + T_e.s)}$$

$T_e = \frac{L_a}{R_a}$: Constante du temps électrique.

<u>Modélisation de la partie mécanique</u>:

On a selon la deuxième loi de la dynamique:

$$\sum C = j\frac{dw(t)}{dt}$$

$$C_m - C_r - C_f = j\frac{dw(t)}{dt}$$

Avec:

C_m: le couple moteur (couple mécanique).

C_r: le couple résistant (couple de charge).

C_f: le couple de frottement, $C_f = f.w$, dont f est le coefficient de frottement.

Donc:

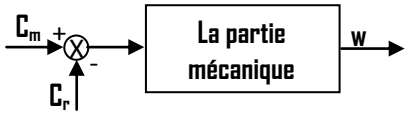
$$C_m - C_r - f.w = j\frac{dw(t)}{dt}$$

En appliquant la transformée de Laplace, on aura:

$$C_m(s) - C_r(s) = f.w(s) + j.s.w(s)$$

$$C_m(s) - C_r(s) = (f + j.s)w(s)$$

La fonction de transfert de la partie mécanique:



$$F(s) = \frac{w}{C_m(s) - C_r(s)} = \frac{1}{f + j.s}$$

Les gains: k (Constante du flux) et k_m (Coefficient électromécanique du moteur) sont calculés comme suit:

En calculant: U_d-R_a.I_d en régime établit, donc il faut connaitre préalablement R_a et mesurer I_d, et en mesurant également la vitesse w.

$$k = \frac{U_d - R_a.I_d}{w} = \frac{220 - 16.4}{209.3} = 0.9667$$

$$k_m = \frac{1}{k} = \frac{1}{0.9667} = 1.0343$$

Conception du contrôleur PID flou de la vitesse du moteur:

La structure générale d'un régulateur PID flou est donnée par:

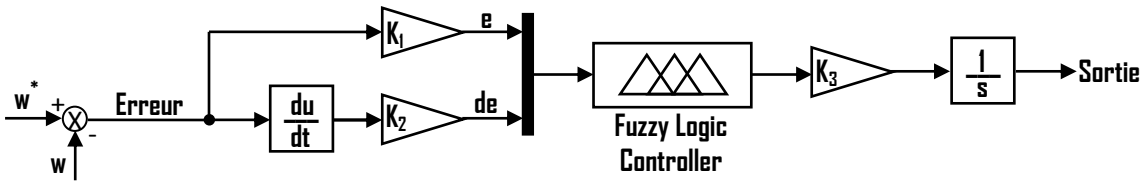


Fig. I.10: *Structure d'un régulateur PID flou*

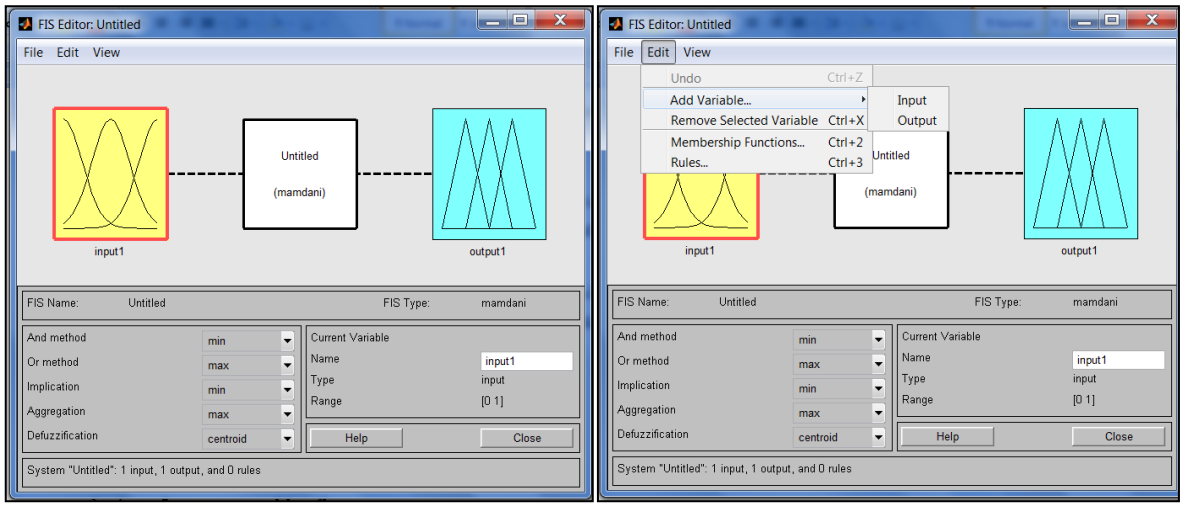
Ce régulateur possède 02 entrées: l'erreur (e) et la dérivée de l'erreur (de), et une seule sortie (du).

La synthèse du contrôleur flou nécessitant les étapes de: fuzzification, choix de la méthode d'inférence, chargement des règles,…etc. Cependant, les performances désirées peuvent être abouties via le réglage manuel (tâtonnement) des trois gains K_1, K_2 et K_3.

- **Les étapes à suivre:**

Pour commencer la synthèse du régulateur flou, il faut ouvrir tout d'abord l'éditeur FIS (Fuzzy Inference System) *(Fig.I.11, a)*, il suffit d'utiliser la commande **«fuzzy»** du l'environnement Matlab.

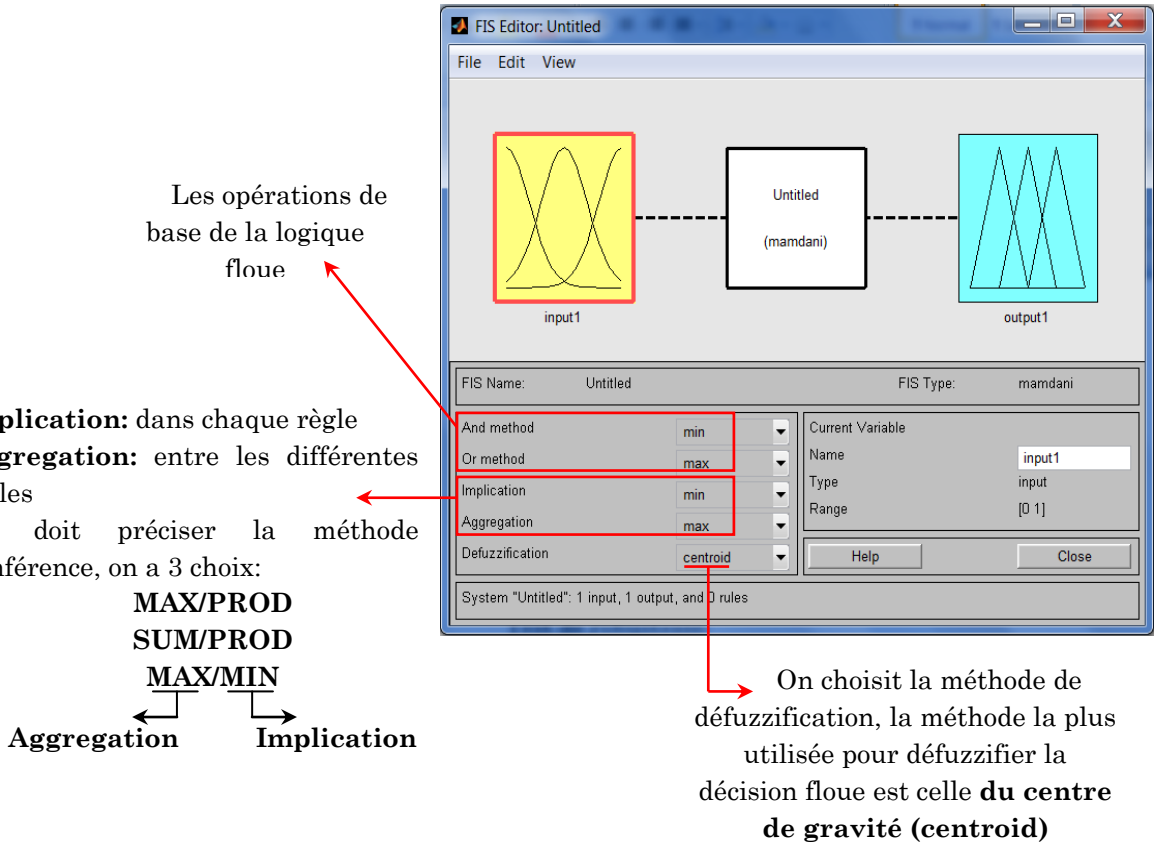
Par défaut l'éditeur nous donne une seule entrée et une seule sortie, pour ajouter d'autres entrées ou d'autres sorties, on appui sur le bouton **«Edit»** puis **«Add Variable»** puis **«Input»** et/ou **«Output»** *(Fig.I.11, b)*.



(a) *(b)*

Fig. I.11: *FIS editor*

- **<u>Remplissage des champs</u>:**



<u>NB</u>:

- Le mot **Aggregation** (en français Agrégation) veut dire: rassemblement, collecte.
- Le mot **Inference** (en français Inférence) vient du verbe: inférer qui veut dire: tirer une conséquence de quelque chose, conclure, déduire.

Après avoir choisir la méthode de fuzzification et de défuzzification, on doit maintenant nommer les entrées et les sorties (dans la case **«Name»**), et choisir la plage de variation de la variable (dans la case **«Range»**).

On commence à fuzzifier nos entrées et sorties:

1- Avec 3 sous-ensembles flous:

Plusieurs types des fonctions d'appartenance peuvent être choisis (triangulaire, trapézoïdales, cloches, …etc.). Le choix le plus célèbre est d'utiliser les deux fonctions: trapézoïdale et triangulaire, comme la montre la figure ci-dessous:

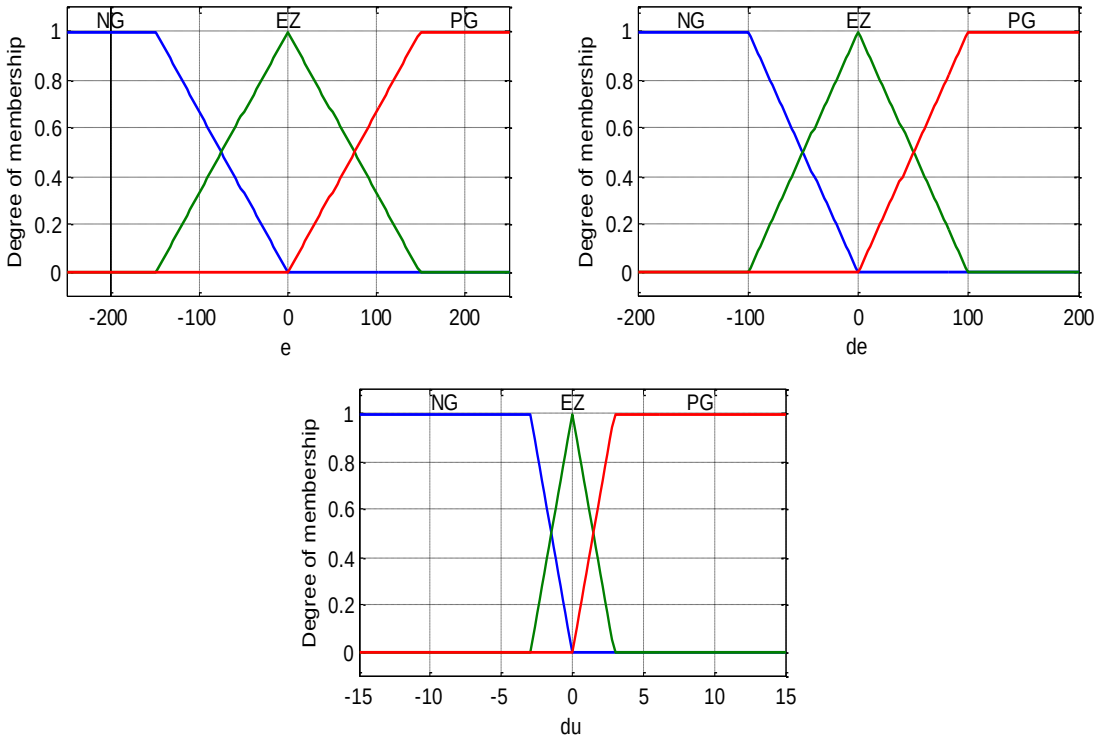


Fig. I.12: *Fuzzification des entrées et de la sortie en 3 sous ensembles flous*

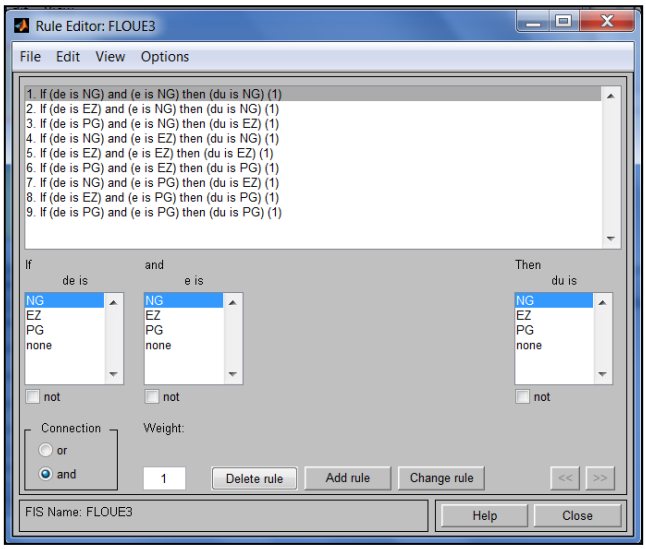
du		***e***		
		NG	*EZ*	*PG*
de	*NG*	*NG*	*NG*	*EZ*
	EZ	*NG*	*EZ*	*PG*
	PG	*EZ*	*PG*	*PG*

NG: Négatif Grand

EZ: Environ Zéro

PG: Positif Grand

On passe maintenant au chargement des règles à partir de la table d'inférence construite à base de raisonnement d'un expert humain ayant une parfaite maitrise sur le système à commander.



On a 2 entrées avec 3 sous ensembles flous pour chacune, donc on aura 9 règles.

Exemple d'une règle:

Dans la logique floue, les règles sont de la forme: **si** (condition sur l'entrée) **alors** (action sur la sortie).

If (de is NG) and (e is NG) **then** (du is NG)

Après le chargement des règles qui est l'étape finale du projet de conception d'un régulateur flou, on doit le nommer et le sauvegarder, pour pouvoir l'utiliser par la suite.

Remarques:

- Le projet a une extension: .fis (exemple: TP1.fis).
- Pour pouvoir l'utiliser dans le Simulink, on doit introduire un bloc qui s'appelle **«Fuzzy Logic Controller»** et qui se trouve dans: **Library_Fuzzy Logic_Toolbox**, ce bloc doit porter le même nom du projet (TP1).
- Le projet est inconnu par l'espace de travail (Workspace), et pour le rendre connu on doit l'exporter vers le Workspace en cliquant sur: **File_Export_To Workspace.**

2- Avec 5 sous-ensembles flous:

Pour bien quantifier nos variables flous et améliorer les performances, et comme dans le cas précédent, on va maintenant fuzzifier les entrées et la sorties en 5 sous ensemble flous *(Fig. I.13)*.

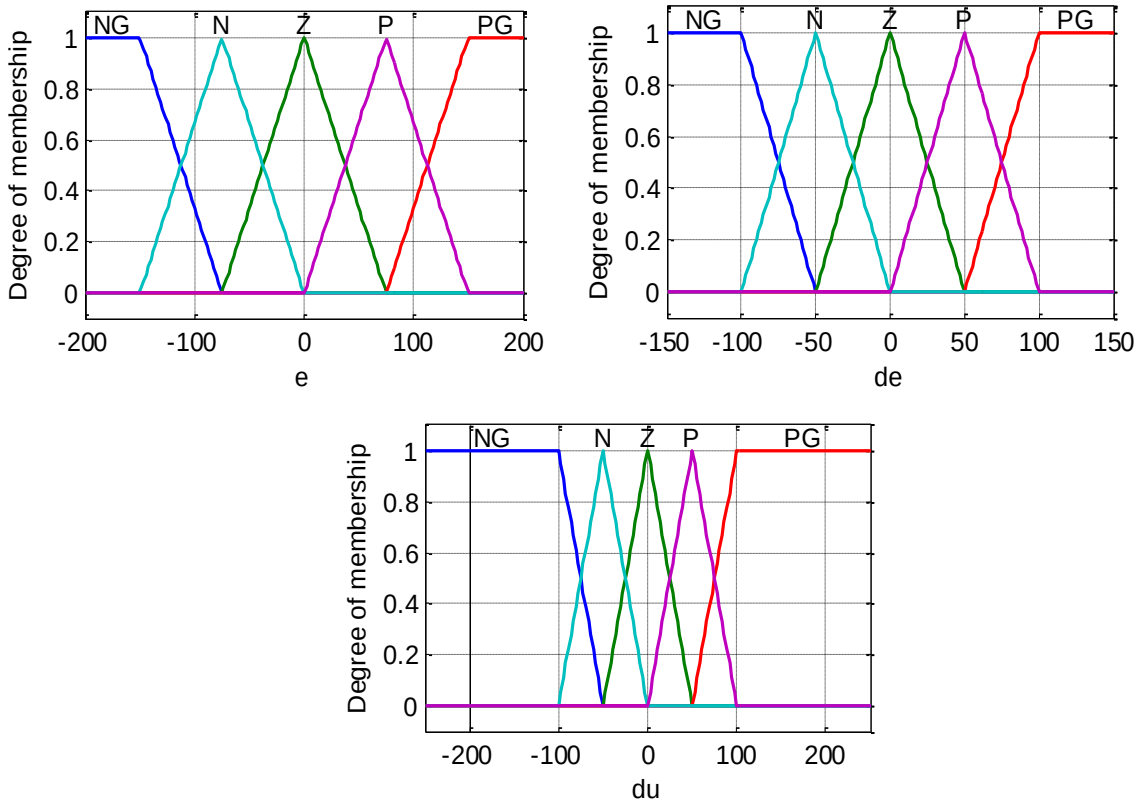


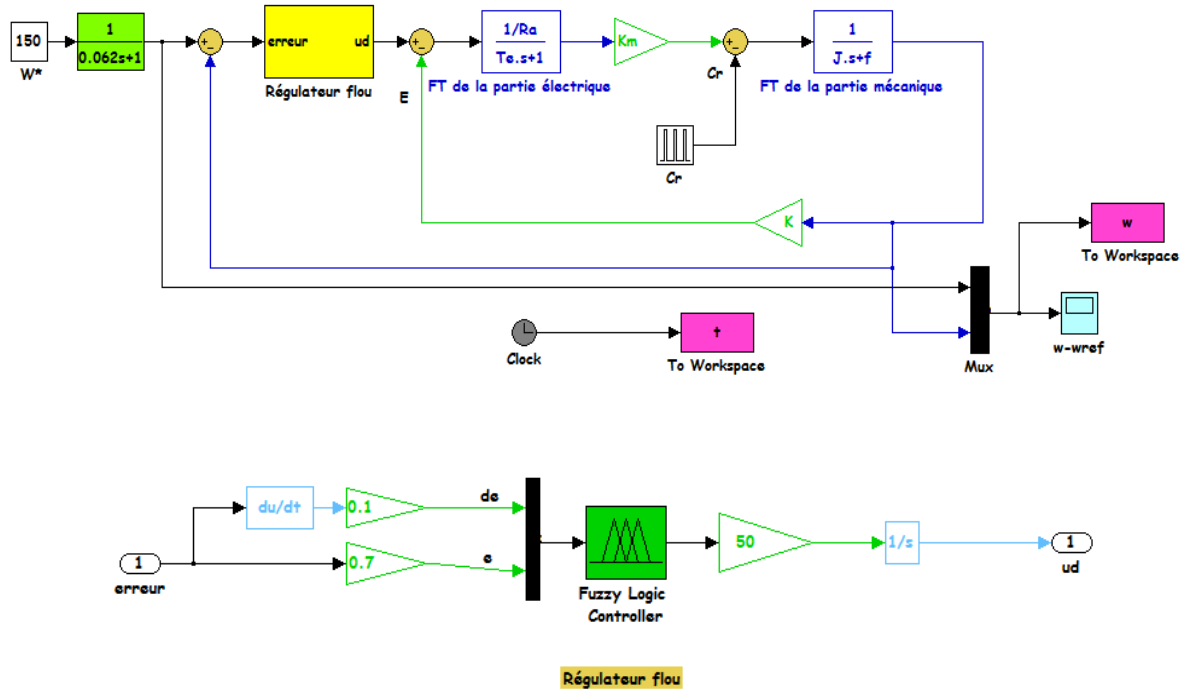
Fig. I.13: *Fuzzification des entrées et de la sortie en 5 sous ensembles flous*

La table d'inférence est donnée comme suit:

du		***e***				
		NG	*N*	*EZ*	*P*	*PG*
de	*NG*	*NG*	*NG*	***NG***	***N***	***EZ***
	N	*NG*	*NG*	*N*	*EZ*	*P*
	EZ	*NG*	*N*	*EZ*	*P*	*PG*
	P	*N*	*EZ*	***P***	*PG*	*PG*
	PG	*EZ*	*P*	*PG*	*PG*	*PG*

<u>Schéma bloc de simulation</u>:

Le schéma bloc de simulation est présenté dans la figure ci-après:



Résultats de simulation:

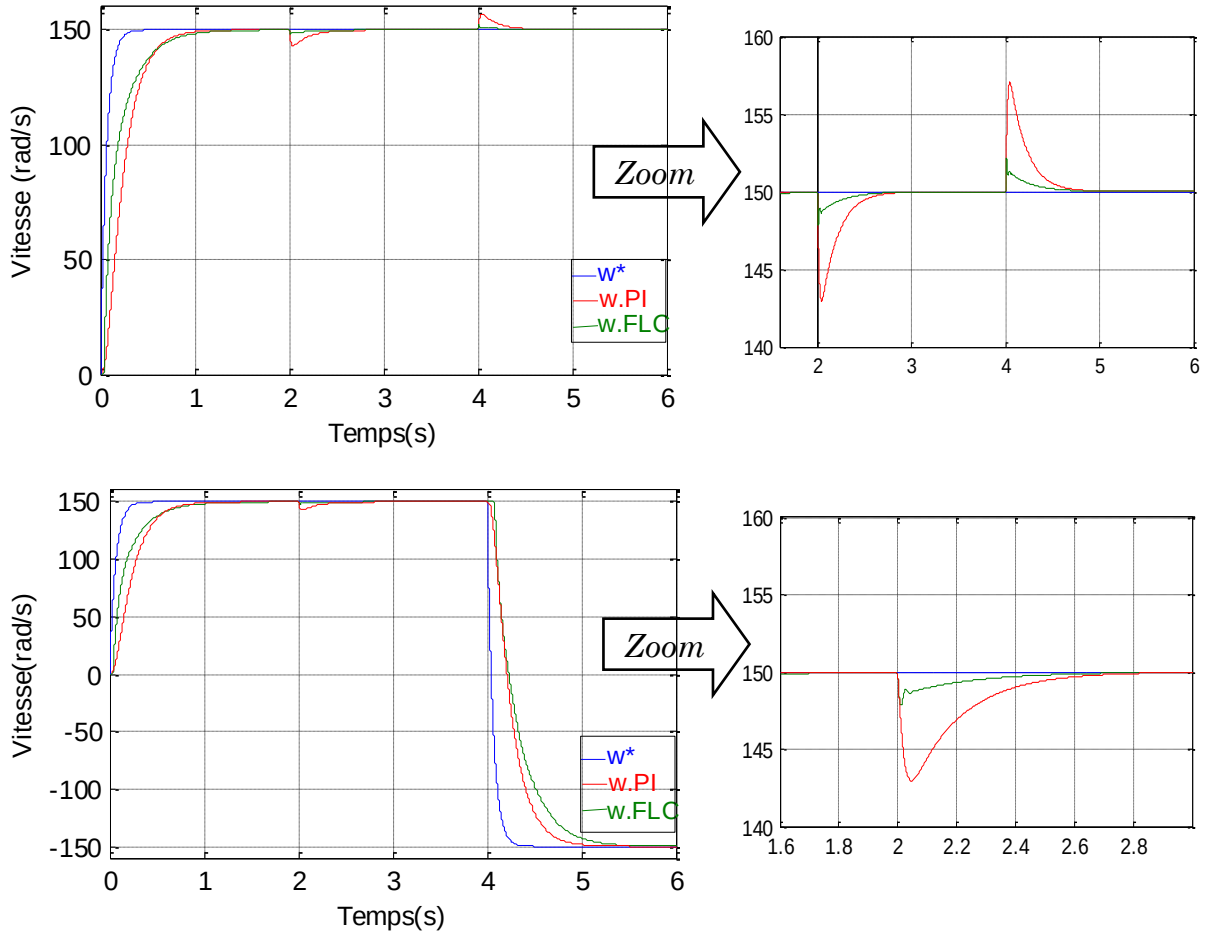
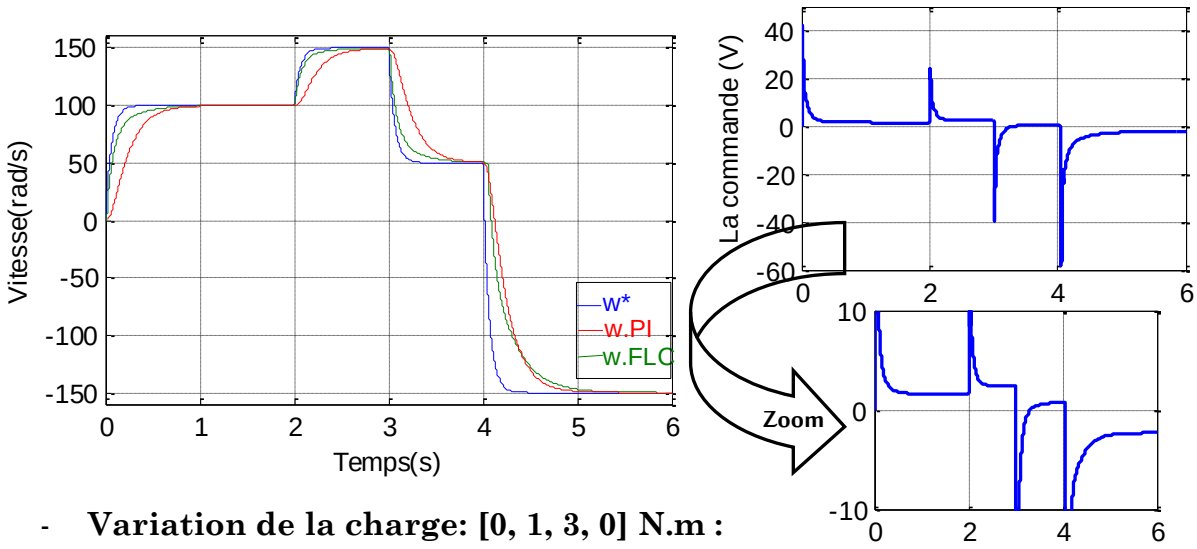
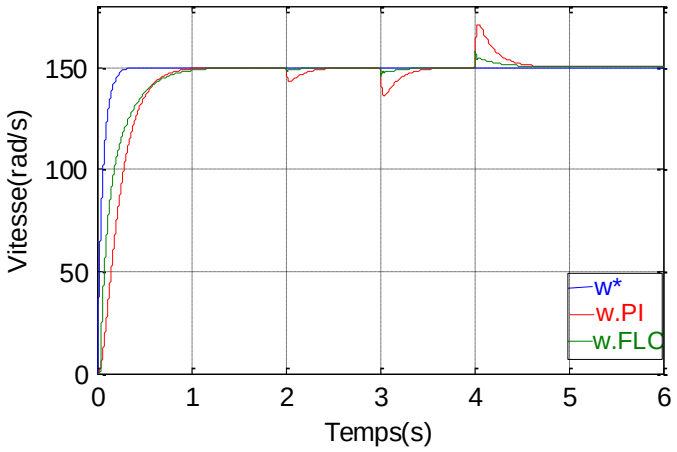


Fig. I.14: *Régulation de la vitesse d'un MCC avec application d'un C_r à t=2s*

- **Variation de la vitesse de référence: [100, 150, 50, -150] rad/s:**



- **Variation de la charge: [0, 1, 3, 0] N.m :**

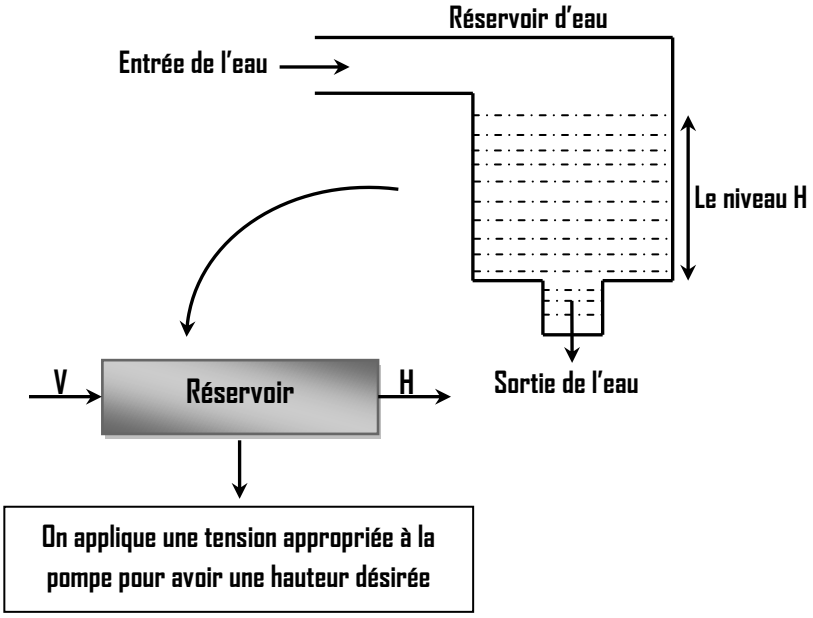


D'après les résultats on remarque que le réglage flou donne des meilleures performances par apport au régulateur classique PI, en termes de poursuite de la consigne, le temps de montée et le rejet de perturbation. En plus il est insensible aux variations paramétriques du système (dans ce cas (cas d'un MCC) l'influence de la variation paramétrique ne se voit pas clairement, par contre elle est bien claire dans la machine asynchrone par exemple), ceci revient au fait que la conception d'un régulateur flou ne tient pas en compte les paramètres du système à commander.

7.2. Réglage de niveau dans un réservoir d'eau:

<u>Schéma de principe</u>:

Le schéma de principe est montré par la figure ci-après:



Données:

La vitesse d'entrée est proportionnelle à la tension V appliquée à la pompe:

$$\mathbf{W_{entrée} = b.V}$$

La vitesse de sortie est proportionnelle à la racine carrée de la hauteur:

$$\mathbf{W_{sortie} = \boldsymbol{a}.\sqrt{\boldsymbol{H}}}$$

Modèle mathématique:

L'équation différentielle de la hauteur d'eau H dans le réservoir est donnée par:

$$\mathrm{Vol} = \mathrm{A.H} \Rightarrow \frac{\mathrm{dVol}}{\mathrm{dt}} = \mathrm{A}.\frac{\mathrm{dH}}{\mathrm{dt}} = \mathrm{b.V} - \mathrm{a}.\sqrt{\mathrm{H}}$$

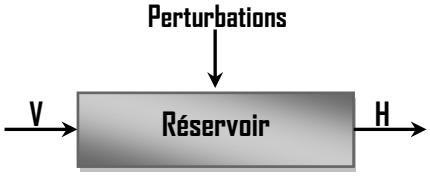
Est une fonction du temps à cause de la différence entre les débits entrants et les débits sortants.

A: Surface de la section transversale de réservoir.

b: Constante relative au débit à l'entrée du réservoir.

a: Constante relative au débit à la sortie du réservoir.

Simulink:



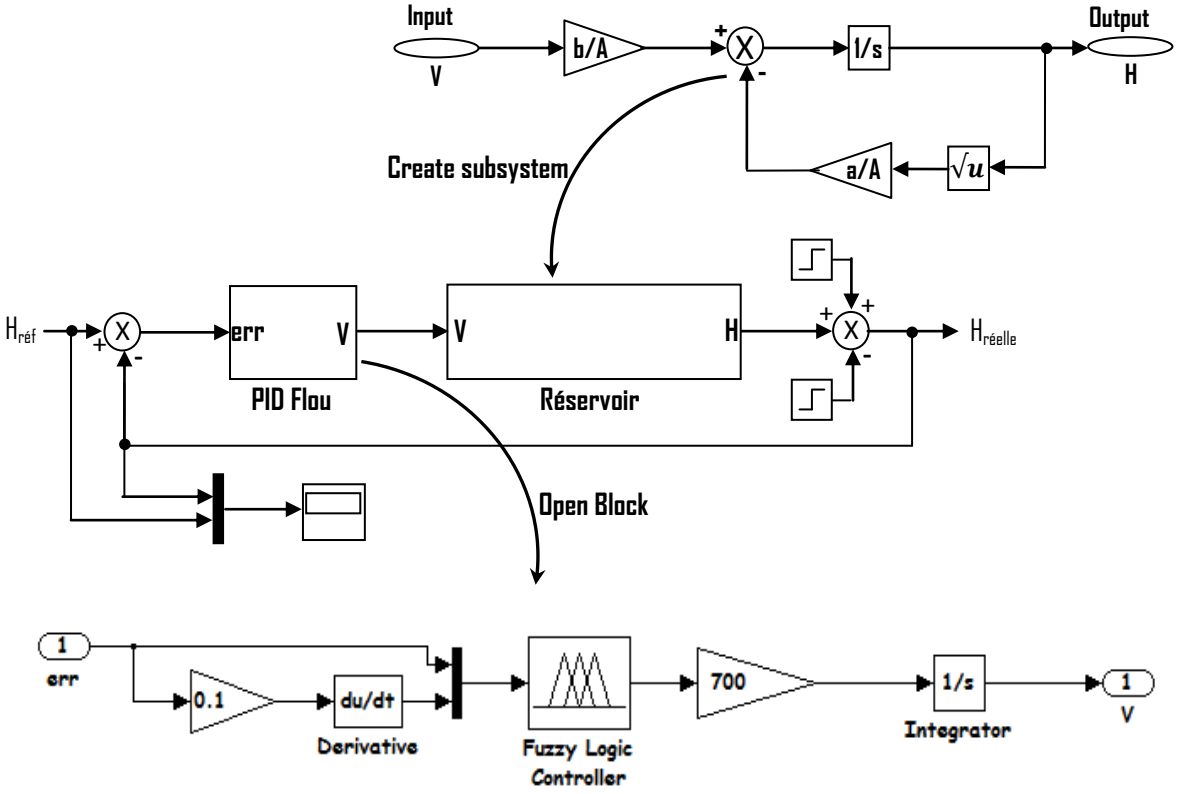
D'après L'équation différentielle de la hauteur d'eau H:

$$A.\frac{dH}{dt} = b.V - a.\sqrt{H}$$

$$\frac{dH}{dt}\frac{b}{A}.V - \frac{a}{A}.\sqrt{H}$$

$$\int \frac{dH}{dt} = \int \frac{b}{A}.V - \frac{a}{A}.\sqrt{H}$$

$$H = \int \frac{b}{A}.V - \frac{a}{A}.\sqrt{H}$$



Les entrées (l'erreur et sa dérivée) sont fuzzifiées en trois sous-ensembles flous, tandis que la sortie est fuzzifiée en cinq sous-ensembles flous. La table d'inférence et les fonctions d'appartenance sont données ci-après:

V		***e***		
		N	*Z*	*P*
de	*N*	*NG*	*N*	*Z*
	Z	*N*	*Z*	*P*
	P	*Z*	*P*	*PG*

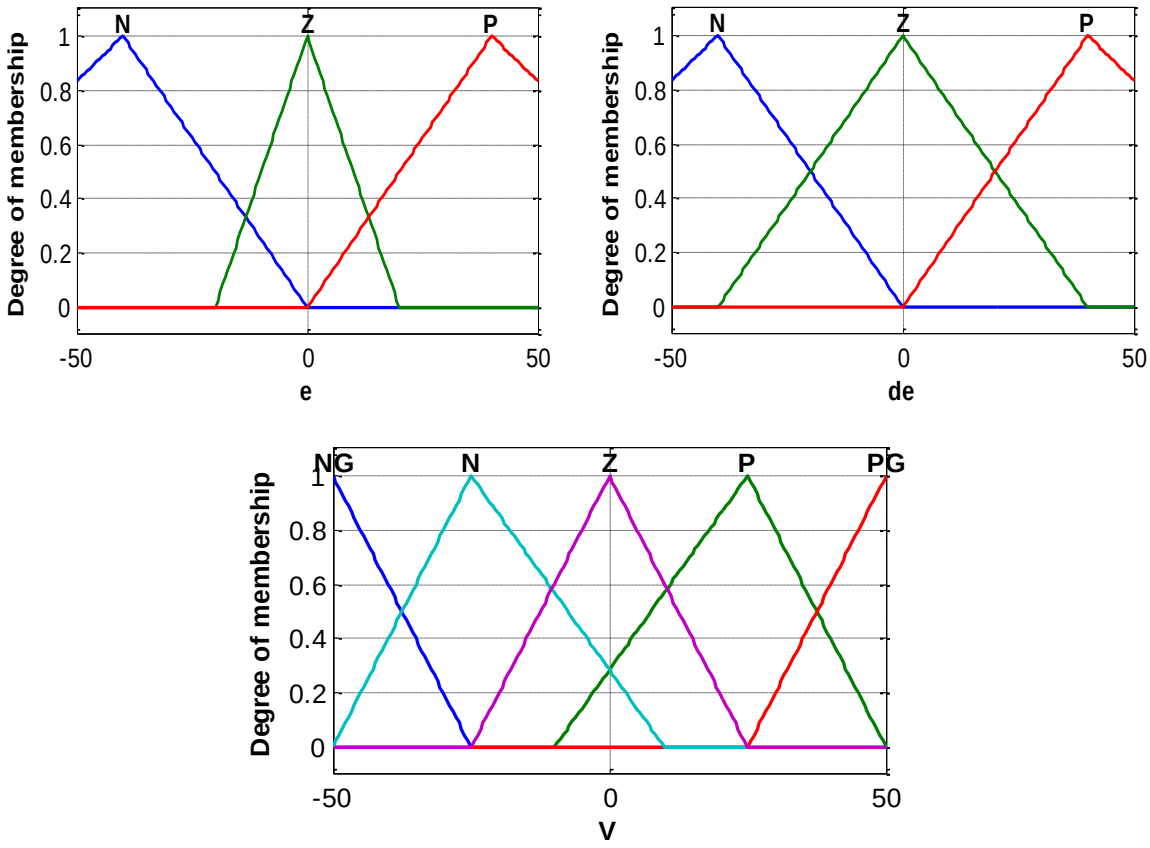


Fig. I.15: *Fonctions d'appartenance des entrées et de la sortie*

<u>Résultats de simulation</u>:

- Choisissant une hauteur de référence (le niveau d'eau désiré) égale à **70m.**
- Appliquant une perturbation égale à **-10** à **t=3s**, et une autre perturbation égale à **+10** à **t=8s.**

Avec: a=10; b=20; A=25;

Voici les résultats de simulation obtenus:

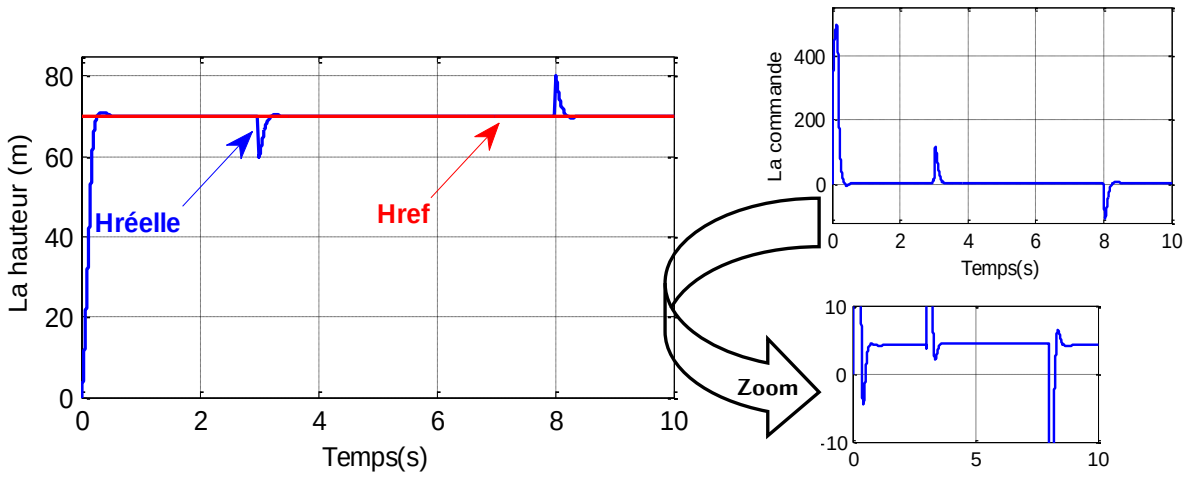


Fig. I.16: *Poursuite du niveau d'eau avec une consigne fixe*

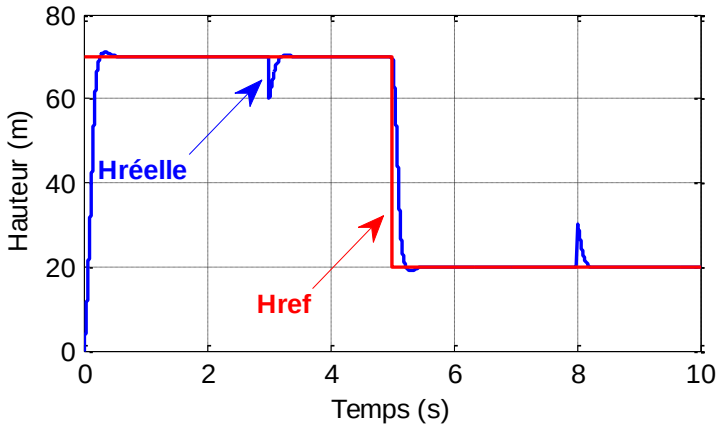


Fig. I.17: *Poursuite du niveau d'eau avec une consigne variable*

On remarque que la hauteur suit parfaitement sa consigne (sa référence), avec un rejet assez court lors de l'application d'une perturbation.

7.3. Régulation de la température:

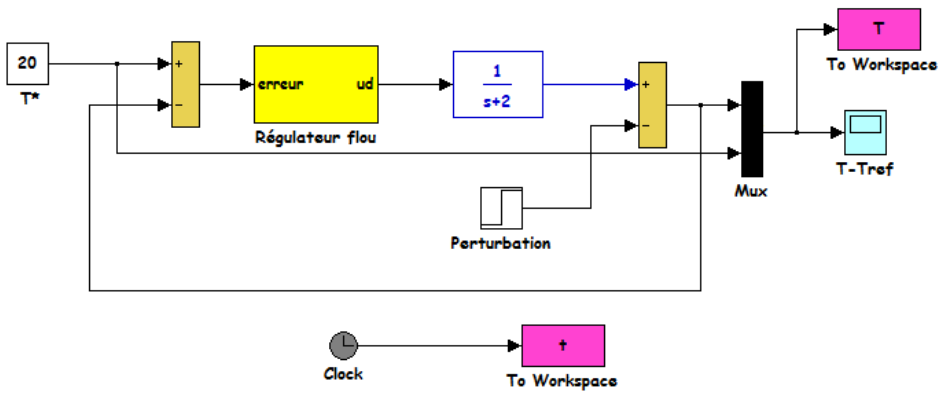
Cette partie à pour but de commander la température d'une serre par un contrôleur synthétiser par la logique floue. A cet effet, on utilise un capteur pour mesurer la température réelle, qui va se comparer par la suite avec la température désirée (température de référence) pour fournir l'erreur au régulateur.

Le comportement réel de ce système a été linéarisé autour d'une température de référence. Ceci permet de fournir un modèle linéaire de synthèse donné par la fonction de transfert ci-dessous:

$$\mathbf{F(s)} = \frac{\mathbf{1}}{\mathbf{s+2}}$$

Avec un moyen de chauffage et de ventilation, afin d'intervenir, selon le besoin, pour chauffer ou pour refroidir.

Le schéma bloc de simulation est donné par:



La température désirée est de 20°C, une perturbation (diminution de la température) a été appliquée à t=12 s. Trois types de régulateurs ont été utilisés à savoir: un régulateur classique PID, un régulateur flou avec 3 sous ensembles flous et un régulateur flou avec 5 sous ensembles flous. Les résultats obtenus sont présentés par la figure ci-dessous.

D'après les résultats de simulation, on remarque que les régulateurs flous donnent des meilleurs performances par apport au régulateur classique en termes de: dépassement, temps de réponse et de rejet de perturbation.

Concernant les régulateurs flous, celui de 5 sous ensembles flous porte des améliorations remarquables en comparaison avec celui de 3 sous ensembles flous.

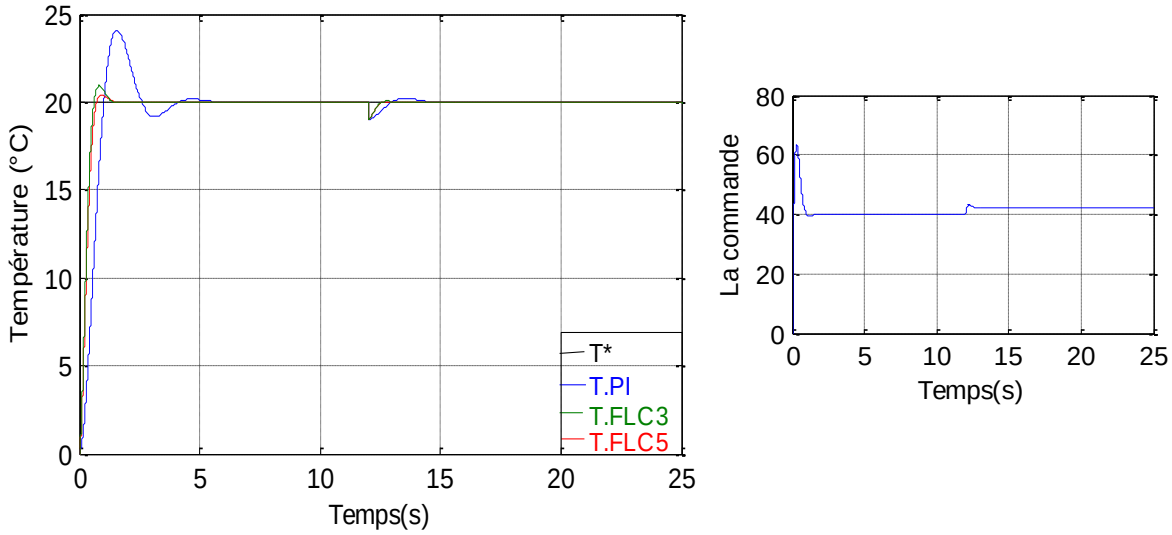


Fig. I.18: *Régulation de la température d'une serre*

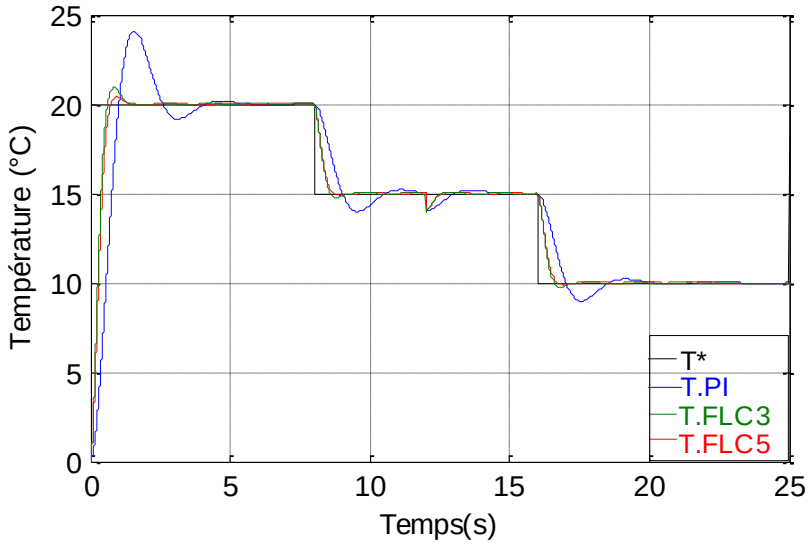


Fig. I.19: *Régulation de la température avec une consigne variable*

8. Conclusion:

La logique floue est l'une des techniques de l'intelligence artificielle qui nous permet de modéliser les connaissances d'un expert humain ayant une parfaite maitrise sur le système à commander.

Le régulateur flou possède de très bonnes performances dynamiques par apport au régulateur classique, en outre, il est robuste vis-à-vis la variation paramétrique du modèle. Ceci revient au fait que la synthèse du contrôleur flou ne se repose pas sur la connaissance a priori du modèle de synthèse décrivant la dynamique réelle du système à commander.

L'inconvénient majore de cette technique est le manque des directives précises pour la conception d'un régulateur floue.

Partie II: Les Réseaux de Neurones Artificiels

1. Introduction:

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

Les réseaux de neurones artificiels, ou réseaux neuromimétiques, sont des modèles inspirés du fonctionnement cérébral de l'être humain et de ses capacités d'apprentissage. Ce sont des structures organisées autour d'un ensemble de cellules (les neurones) interconnectées selon une certaine architecture par des liens pondérés et modifiables lors d'une procédure appelée apprentissage.

Ils sont employés dans toutes sortes d'applications et dans divers domaines. Ils sont d'excellent candidat pour résoudre les problèmes de modélisation, d'identification et de contrôle des processus non linéaires et complexes.

2. Le neurone biologique:

Le neurone biologique comprend essentiellement:

- **Les dendrites**: qui sont les récepteurs principaux du neurone, captant les signaux qui lui parviennent.
- **Le corps cellulaire**: qui fait la somme des influx qui lui parviennent; si cette somme dépasse un certain seuil, il envoie lui-même un influx par l'intermédiaire de l'axone.
- **L'axone**: qui permet de transmettre les signaux émis par le corps cellulaire aux autres neurones.
- **Les synapses**: qui permettent aux neurones de communiquer avec les autres via les axones et les dendrites.

La figure (Fig. II.1) présente les éléments principaux d'un neurone biologique:

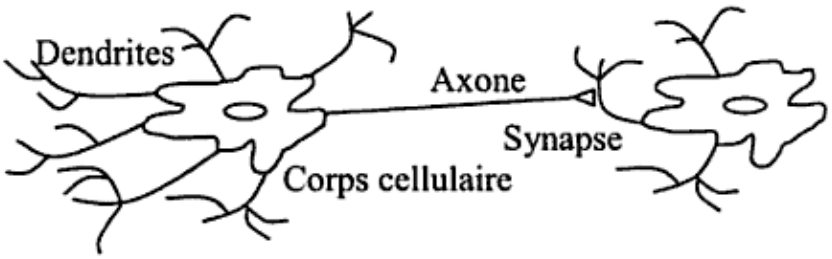


Fig. II.1: *Neurone biologique*

3. Le neurone formel (Artificiel):

Le neurone formel est un modèle théorique de traitement de l'information inspiré des observations relatives au fonctionnement d'un neurone biologique, pour but de reproduire le raisonnement intelligent d'une manière artificielle.

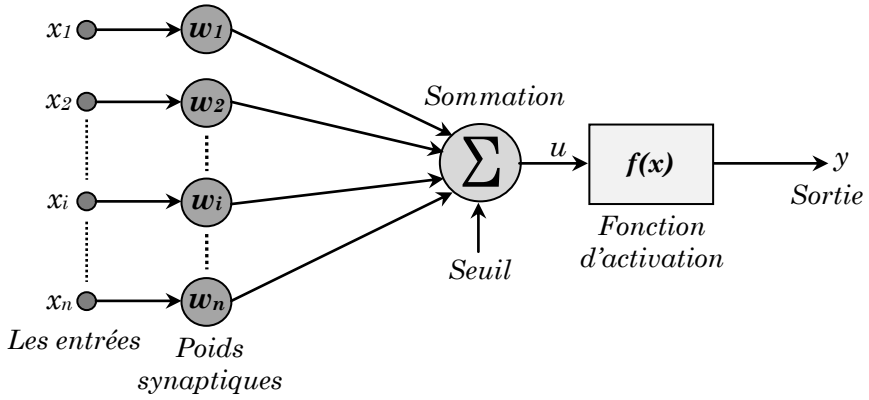


Fig. II.2: *Neurone artificiel*

Le tableau ci-dessous montre la mise en correspondance entre un neurone biologique et un neurone artificiel:

Tab. II.1: *Neurone biologique et neurone artificiel*

Neurone biologique	***Neurone artificiel***
Synapses	*Poids des connexions*
Axones	*Signal de sortie*
Dendrites	*Signal d'entrée*
Noyau ou Somma	*Fonction d'activation*

Le neurone formel peut être défini, d'une façon plus générale, par les éléments suivants:

1- **Les entrées du réseau neurone:** Elles peuvent être binaires (0, 1) ou réelles.

2- **Fonction d'activation:** Est une fonction mathématique qui permet de définir l'état interne du neurone en fonction de son entrée totale, la figure ci-dessous présente quelques fonctions les plus souvent utilisées:

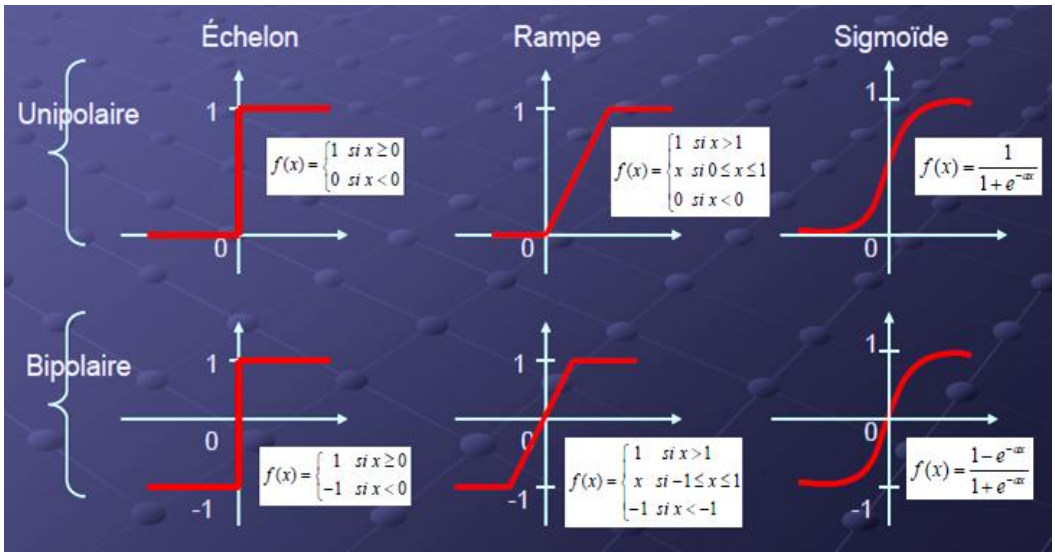


Fig. II.3: *Fonction d'activations*

3- **Fonction de sortie:** Elle calcule la sortie d'un neurone en fonction de son état d'activation. En général, cette fonction est considérée comme la fonction identité. Elle peut être: binaire *(0, 1)*, bipolaire *(−1, 1)* ou réelle.

4- **Poids des connexions:** Un poids w_{ij} est associé à chacune des connexions. Un poids d'un neurone artificiel représente l'efficacité d'une connexion synaptique. Un poids négatif vient inhiber une entrée, alors qu'un poids positif vient l'accentuer.

4. Différents modèles des réseaux de neurones:

Un réseau de neurone est une structure organisée autour d'un ensemble de neurones interconnectés, selon une certaine topologie, par des liaisons affectées de poids. Ces liaisons offrent à chaque neurone un canal pour émettre et recevoir des signaux venant d'autres neurones. On peut classifier les réseaux de neurones artificiels selon deux grands familles:

4.1. Réseaux de neurone monocouches:

Présenté par *Frank Rosenblatt*, en 1958, le **Perceptron** est le premier modèle et la forme le plus simple du réseau de neurones. Le perceptron monocouche se compose de deux couches: la couche d'entrée et la couche de sortie qui donne la réponse correspondant à la stimulation présente en entrée.

Un Perceptron simple ne peut traiter que les problèmes linéairement séparables où les états peuvent être séparés par une droite, comme les fonctions logiques **AND** et **OR** par exemple. Pour les problèmes qui ne sont pas

linéairement séparables comme la fonction logique **XOR** par exemple, on doit utiliser un Perceptron multicouche.

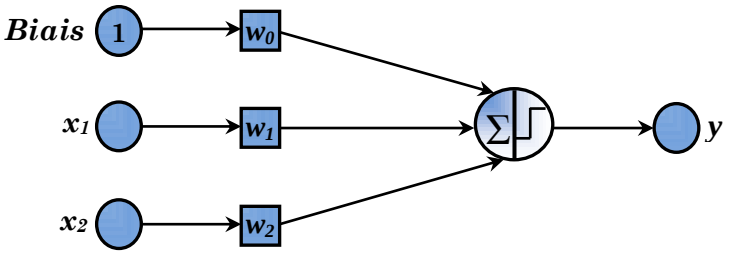


Fig. II.4: *Modèle d'un Perceptron simple (monocouche)*

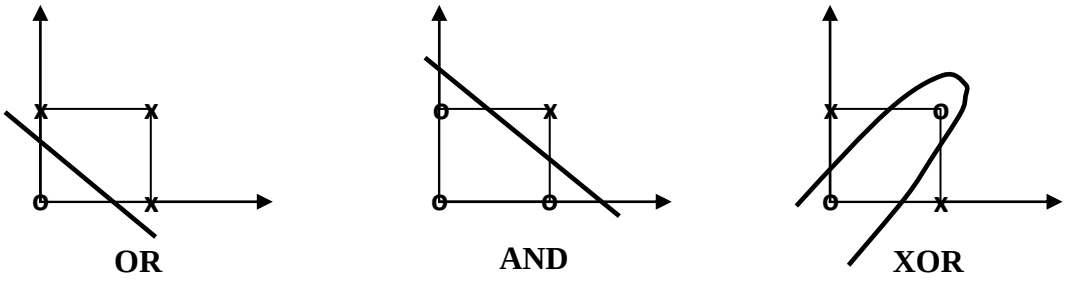


Fig. II.5: *Les problèmes linéairement et non linéairement séparables*

4.2. Réseaux de neurone multicouches:

C'est une extension du Perceptron monocouche, avec une ou plusieurs couches cachées entre l'entrée et la sortie. La figure (Fig. II.6) donne l'exemple d'un réseau contenant une couche d'entrée, deux couches cachées et une couche de sortie. La couche d'entrée représente toujours une couche virtuelle associée aux entrées du système. Elle ne contient aucun neurone. Les couches suivantes sont des couches de neurones. Dans l'exemple illustré, il y a trois entrées, quatre neurones sur la première couche cachée, trois neurones sur la deuxième couche cachée et quatre neurones sur la couche de sortie. Les sorties des neurones de la dernière couche correspondent toujours aux sorties du système.

Dans le cas général, un Perceptron multicouche peut posséder un nombre de couches quelconque et un nombre de neurones (ou d'entrées) par couche également quelconque.

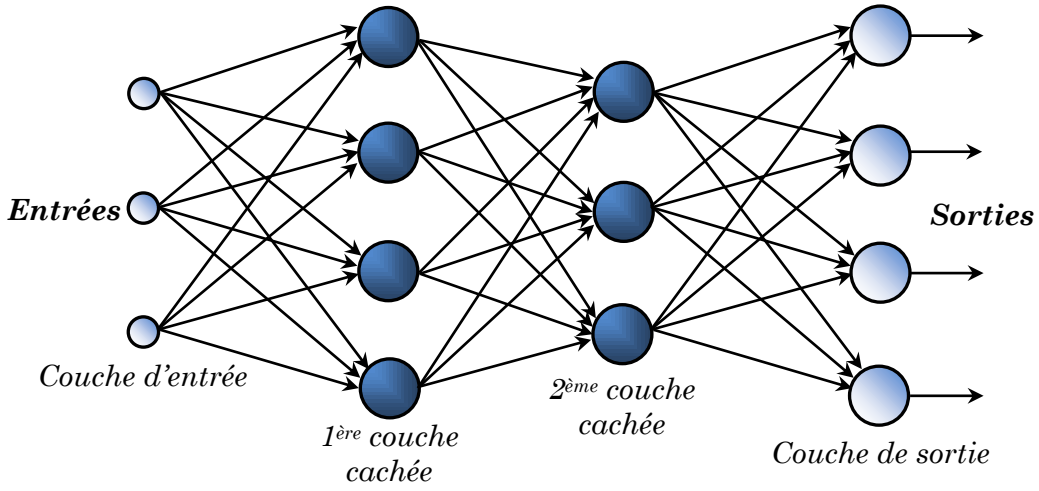
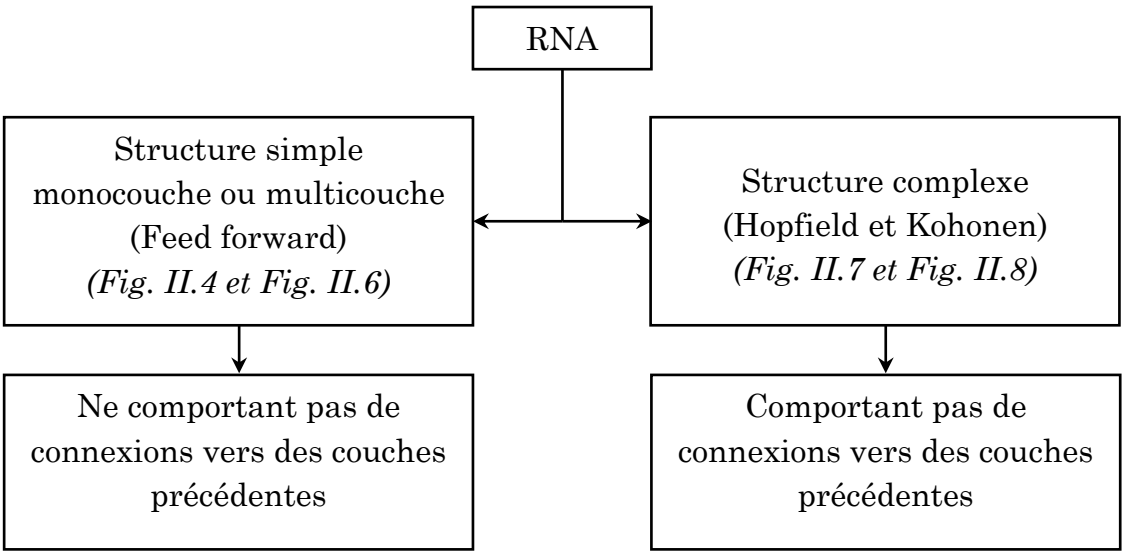


Fig. II.6: *Modèle d'un Perceptron multicouches*

5. Structure des réseaux de neurones:

L'organigramme suivant résume les différentes structures des réseaux de neurone:



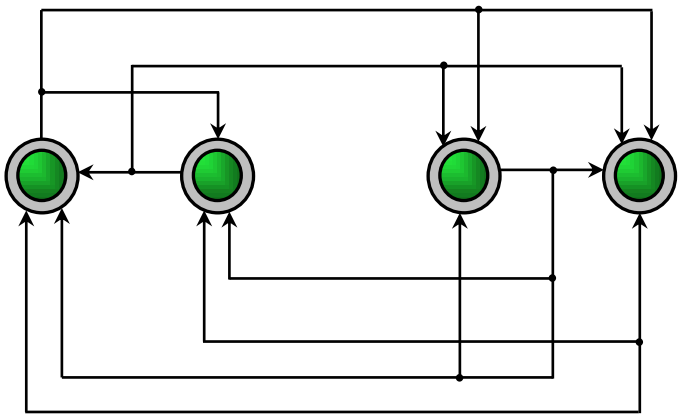


Fig. II.7: *Modèle de Hopfield*

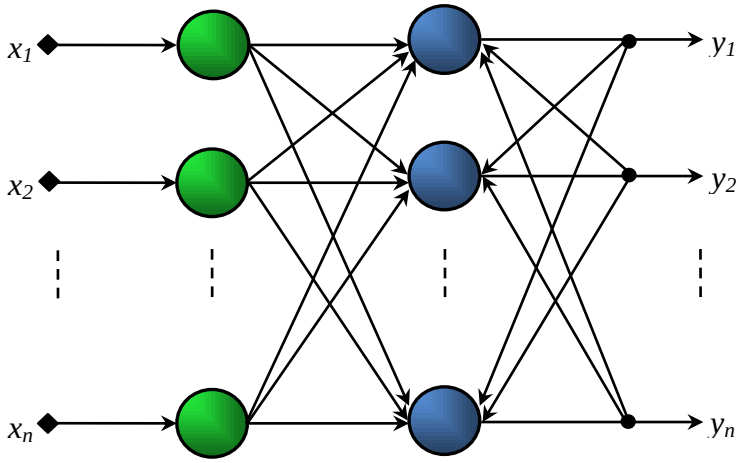


Fig. II.8: *Modèle de Kohonen*

6. L'apprentissage d'un réseau de neurones:

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. Elle ne concerne cependant pas tous les modèles, mais les plus utilisés. L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement.

Il existe deux grandes familles d'apprentissage, supervisé et non supervisé:

6.1. Apprentissage supervisé:

L'apprentissage est dit supervisé lorsque le réseau est forcé à converger vers un état final précis, ce qui nécessite la connaissance préalable (à priori) de la réponse désirée d(n). La méthode la plus utilisée est la rétro-propagation du gradient. Elle consiste à présenter des exemples au réseau, calculer sa sortie, ajuster les poids de façon à réduire l'écart entre cette sortie et la réponse désirée pour satisfaire un certain critère de performance.

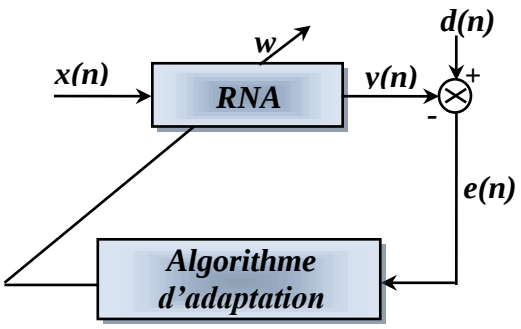


Fig. II.9: *Apprentissage supervisé*

6.2. Apprentissage non supervisé:

Dans l'apprentissage non supervisé, seules les valeurs d'entrée sont disponibles et le réseau est laissé libre de converger vers n'importe quel état final. La connaissance à priori de la sortie désirée n'est pas nécessaire, la procédure d'apprentissage est basée uniquement sur les valeurs d'entrées. Le réseau s'auto-organise de façon à optimiser une certaine fonction de coût.

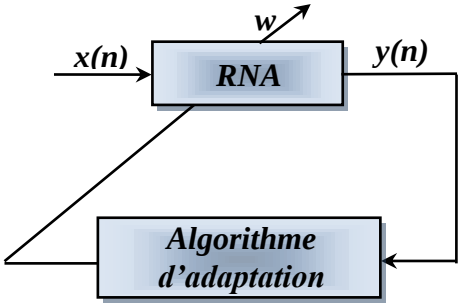
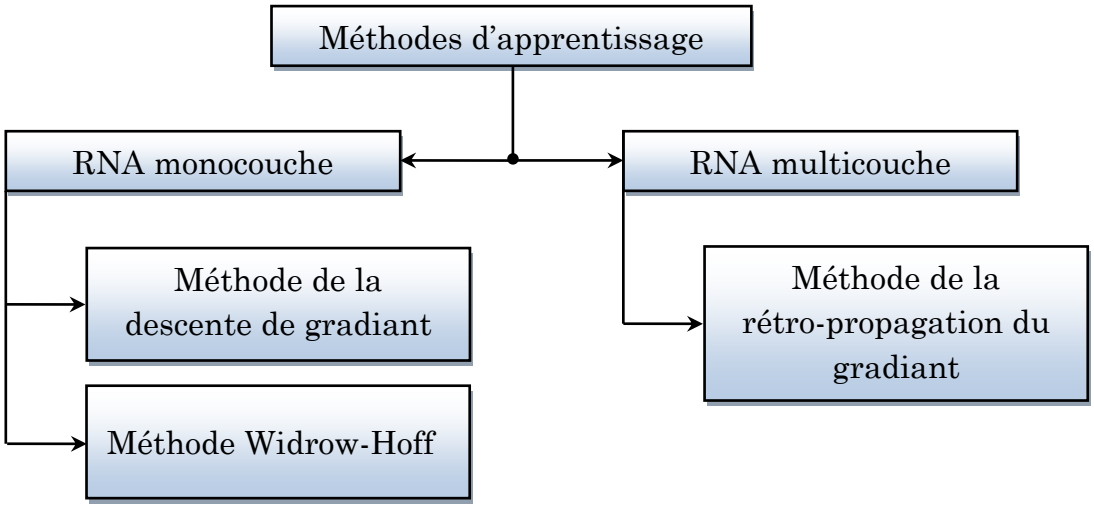


Fig. II.10: *Apprentissage non supervisé*

7. Méthodes d'apprentissage des réseaux de neurones:

Les algorithmes d'apprentissages donnent des meilleurs résultats lorsqu'on leur fournit des exemples multiples et variés. Il existe des différentes méthodes d'apprentissage selon le modèle du réseau (monocouche ou multicouche), l'organigramme suivant illustre les principaux méthodes d'apprentissages les plus couramment utilisées pour l'adaptation des poids.



7.1. Apprentissage des RNA monocouche:

Les deux méthodes mentionnées dans l'organigramme ci-dessus consistent à comparer le résultat qui était attendu (sortie actuelle) pour les exemples, avec la sortie désirée, puis à minimiser l'erreur commise sur les exemples.

La méthode de Widrow-Hoff n'est qu'une variante de la méthode de la descente de gradient, elle consiste à modifier les poids après chaque exemple (chaque step) et non pas après tous les exemples (ce qui est le cas dans la méthode de la descente de gradient), ce qui va minimiser l'erreur de manière précise, et ce sur chaque exemple. Donc le réseau de neurone va s'améliorer mieux et il va tendre plus rapidement à classifier parfaitement chacun des exemples. La formule utilisée pour adapter les poids est:

$$w(n+1) = w(n) + \alpha[d(n) - y(n)]x(n) \qquad \text{(II.1)}$$

Exemple:

Le tableau suivant montre une classification de deux catégories représentées par deux couleurs différentes rouge et bleu:

	R (Rouge)	V (Vert)	B (Bleu)
Classe 1	255	0	0
Rouge	248	80	68
Classe 2	0	0	255
Bleu	67	15	210

Choix de la fonction d'activation:

On a deux sorties, donc on cherche une fonction d'activation qui possède deux sorties, on peut choisir la fonction ***Signum*** par exemple.

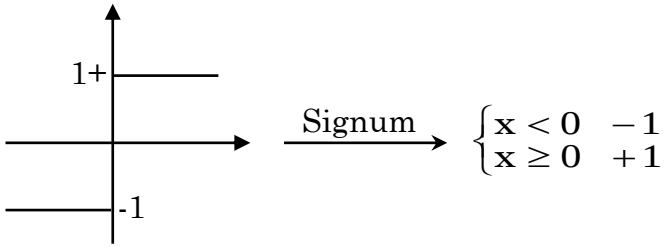
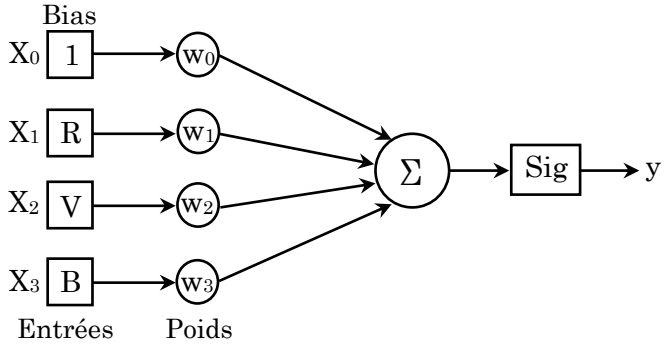


Schéma synoptique:



On peut résumer maintenant les paramètres de notre réseau:

- Les entrées $\mathbf{X_m}$ y compris le bias (qui égale toujours à 1) (X_1, X_2, X_3 et X_0).
- Les poids $\mathbf{W_m}$ y compris le poids du bias (W_1, W_2, W_3 et W_0).
- SOP: Sum Of Product, la somme des produits entre les entrées et les poids.
- La fonction d'activation: **Signum**.
- Les sorties $\mathbf{y_j}$, on a deux sorties: bleu et rouge.
- Le taux d'apprentissage $\boldsymbol{\alpha}$ dont: $0 \leq \alpha \leq 1$.
- Le pas (step) **n** dont: n=0, 1, 2, … (à chaque fois on applique une entrée, n incrémente de 1).
- La sortie désirée $\mathbf{d_j}$, soit égale à: +1 (si on est dans la classe bleu) ou bien égale à: -1 (si on est dans la classe rouge).

<u>La phase d'apprentissage</u>:

Voici les étapes à suivre pour faire l'apprentissage du neurone:

1- Initialisation des poids (on donne des poids initiaux).
2- Application des entrées.
3- Calcul de la somme des produits (SOP).
4- Calcul de la réponse de la fonction d'activation, ce qui va nous donner la sortie y_j.
5- Adaptation des poids (uniquement dans le cas où la sortie calculée est différente de la sortie désirée) on utilisant la formule (II.1).
6- Retour à l'étape numéro 2.

- **<u>Step n=0</u>** , On pose: α=0.001

  $X(n)=X(0)=[X_0, X_1, X_2, X_3]=[+1, 255, 0, 0]$: Vecteur d'entrée pour n=0

  $W(n)=W(0)=[W_0, W_1, W_2, W_3]=[-1, -2, 1, 6.2]$: Poids initiaux

  d(n)=d(0)= **-1**: La sortie désirée (cas de la classe 1)

  SOP=(+1)(-1)+(255)(-2)+(0)(1)+(0)(6.2)= **-511**

  y(n)= y(0)=Sig(SOP)= Sig(-511)= **-1** donc:

  Lorsque n=0, y(n)= d(n)= -1 c-à-d: la sortie mesurée=la sortie désirée, donc les poids sont correctes, donc pas d'adaptation des poids dans cette étape.

- **<u>Step n=1</u>**

  $X(n)=X(1)=[X_0, X_1, X_2, X_3]=[+1, 248, 80, 68]$

  $W(n)=W(1)=[W_0, W_1, W_2, W_3]=[-1, -2, 1, 6.2]$

d(n)=d(1)= **-1**

SOP=(+1)(-1)+(248)(-2)+(80)(1)+(68)(6.2)= **4.6**

y(n)= y(1)=Sig(SOP)= Sig(4.6)= **+1** donc:

Lorsque n=1, y(n)≠d(n), c-à-d: la sortie mesurée est différente de la sortie désirée, donc les poids sont incorrectes, donc une adaptation des poids est nécessaire.

Adaptation des poids:

W(n+1)= W(n)+α[d(n)-y(n)].X(n)

Quand n=1:

W(1+1)= W(1)+0.001[d(1)-y(1)].X(1)

W(2)= [-1, -2, 1, 6.2]+0.001[-1-(+1)] [+1, 248, 80, 68]

W(2)= [-1, -2, 1, 6.2]-0.002[+1, 248, 80, 68]

W(2)= [-1.002, -2.496, 0.84, 6.064]: les nouveaux poids (on va les utiliser dans les prochaines étapes).

- **Step n=2**

X(n)=X(2)= [+1, 0, 0, 255]

W(n)=W(2)= [-1.002, -2.496, 0.84, 6.064]

d(n)=d(2)= **+1**

SOP= **1545.32**

y(n)= y(2)=Sig(SOP)= Sig(1545.32)= **+1** donc:

Lorsque n=2, y(n)=y(2)=d(2)= +1 c-à-d: la sortie mesurée=la sortie désirée, donc les nouveaux poids sont correctes, donc pas d'adaptation des poids.

- **Step n=3**

X(n)=X(3)= [+1, 67, 15, 210]

W(n)=W(3)= [-1.002, -2.496, 0.84, 6.064]

d(n)=d(3)= **+1**

SOP= **1349.542**

y(n)= y(3)=Sig(SOP)= Sig(1349.542)= **+1** donc:

Lorsque n=3, y(n)=y(3)=d(3)= +1 c-à-d: la sortie mesurée=la sortie désirée, donc les poids sont correctes, donc pas d'adaptation des poids.

NB: On n'a pas encore terminé, il faut appliquer les nouveaux poids sur les deux premiers cas (n=0 et n=1) (il faut que les nouveaux poids soient valables avec tous les cas).

- **Step n=4**

  X(n)=X(4)= [+1, 255, 0, 0]

  W(n)=W(4)= [-1.002, -2.496, 0.84, 6.064]

  d(n)=d(4)= **-1**

  SOP= **-637.482**

  y(n)= y(4)=Sig(SOP)= Sig(-637.482)= **-1** donc:

  Lorsque n=4, y(n)=y(4)=d(4)= -1 c-à-d: la sortie mesurée=la sortie désirée, donc les poids sont correctes, donc pas d'adaptation des poids.

- **Step n=5**

  X(n)=X(5)= [+1, 248, 80, 68]

  W(n)=W(5)= [-1.002, -2.496, 0.84, 6.064]

  d(n)=d(5)= **-1**

  SOP= **-31.306**

  y(n)= y(5)=Sig(SOP)= Sig(-31.306)= **-1** donc:

  Lorsque n=5, y(n)=y(5)=d(5)= -1 c-à-d: la sortie mesurée=la sortie désirée, donc les poids sont correctes, donc pas d'adaptation des poids.

  Comme ça, les poids sont vérifiées avec tous les cas, et dans tous les cas on a trouvé que la sortie mesurée=la sortie désirée et donc:

  [-1.002, -2.496, 0.84, 6.064]: poids optimaux.

Test d'un échantillon inconnu:

Soit l'échantillon suivant:

(R, V, B)=(150, 100, 180)

W=[-1.002, -2.496, 0.84, 6.064]

SOP=800.118

y=Sig(800.118)=+1: ça veut dire que le réseau classe cet échantillon dans la classe bleu, donc il nous donne une bonne classification.

7.2. Apprentissage des RNA multicouche:

La méthode d'apprentissage utilisée dans ce cas est la méthode de rétro-propagation du gradient, qui consiste à une suite d'optimisations partielles, couche par couche.

Algorithme de la méthode:

La figure suivante présente l'algorithme de la méthode de la rétro-propagation du gradient:

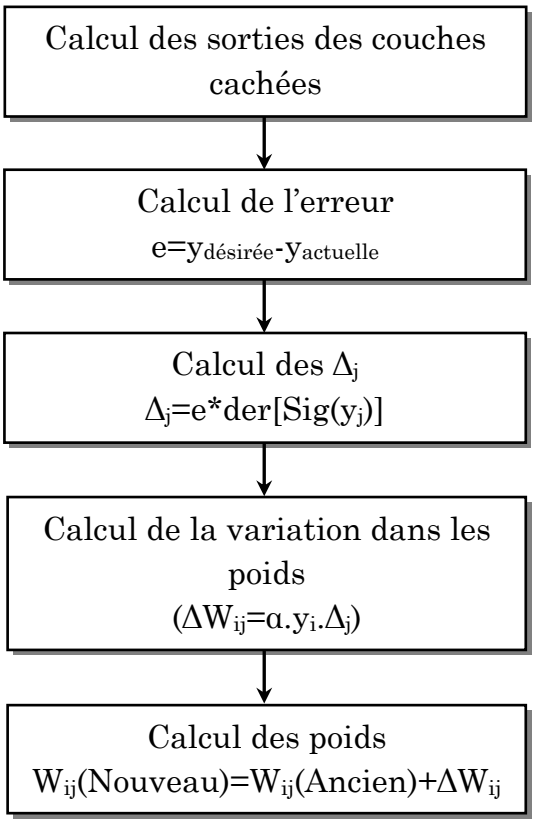


Fig. II.11: *Méthode de la rétro-propagation du gradient*

Exemple:

Soit le neurone multicouche présenté dans la figure ci-dessous, avec 1 couche d'entrée, 1 couche cachée et 1 couche de sortie.

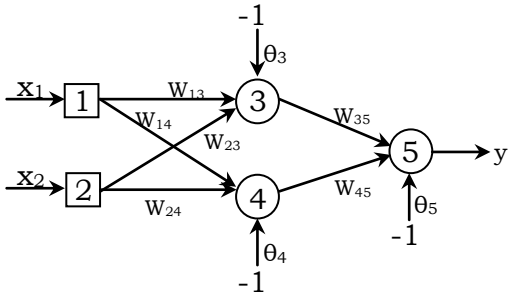


Fig. II.12: *Neurone multicouche*

Avec:

W_{13}=0.5 ; W_{14}=0.9 ; W_{23}=0.4 ; W_{24}=1 ; W_{35}=-1.2 ; W_{45}=1.1

θ_3=0.8 ; θ_4= -0.1 ; θ_5=0.3 ; α=0.1 (Taux d'apprentissage)

En utilisant l'algorithme de rétro-propagation du gradient, on va faire l'adaptation des poids pour obtenir le cas: x_1=1, x_2=1, y=0

La fonction d'activation est: $g = \frac{1}{1+e^{-x}}$

La condition d'arrêt: l'erreur obtenue est de l'ordre de 10^{-3}

- **Calcul des sorties:**

$y_3=Sig(x_1.W_{13}+x_2.W_{23}-1.\theta_3)=Sig(0.5+0.4-0.8)=1/(1+e^{-0.1})=\mathbf{0.525}$

$y_4=Sig(x_1.W_{14}+x_2.W_{24}-1.\theta_4)=Sig(0.9+0.1+1)=Sig(2)=\mathbf{0.8808}$

$y_5=Sig(y_3.W_{35}+y_4.W_{45}-1.\theta_5)=\mathbf{0.5097}$

- **Calcul de l'erreur:**

$e=y_{dés}-y_{cal}=0-0.5097=$ **-0.5097→ Adaptation des poids est nécessaire**

- **Calcul des Δ (Δ_5, Δ_4, Δ_3):**

$\Delta_5=e.derSig(y_5)=e.[y_5.(1-y_5)]=(-0.5097)(0.5097)(1-0.5097)=$ **-0.1274**

$\Delta_3=W_{35}.\Delta_5.derSig(y_3)=(-1.2)(-0.1274)(0.525)(1-0.525)=\mathbf{0.0381}$

$\Delta_4=W_{45}.\Delta_5.derSig(y_4)=(1.1)(-0.1274)(0.8808)(1-0.8808)=$ **-0.0147**

- **Calcul de la variation dans les poids:**

$\Delta W_{35}=\alpha.y_3.\Delta_5=\mathbf{-0.0067}$

$\Delta W_{45}=\alpha.y_4.\Delta_5=\mathbf{-0.0112}$

$\Delta\theta_5=\alpha.(-1).\Delta_5=0.1.(-1).(-0.1274)=\mathbf{0.0127}$

$\Delta W_{13}=\alpha.x_1.\Delta_3=(0.1)(1)(0.0381)=\mathbf{0.00381}$

$\Delta W_{23}=\alpha.x_2.\Delta_3=(0.1)(1)(0.0381)=\mathbf{0.00381}$

$\Delta W_{14}=\alpha.x_1.\Delta_4=(0.1)(1)(-0.0147)=$ **-0.00147**

$\Delta W_{24}=\alpha.x_2.\Delta_4=(0.1)(1)(-0.0147)=$ **-0.00147**

$\Delta\theta_3=\alpha.(-1).\Delta_3=(0.1)(-1)(0.0381)=$ **-0.00381**

$\Delta\theta_4=\alpha.(-1).\Delta_4=(0.1)(-1)(-0.0147)=\mathbf{0.00147}$

- **Calcul des nouveaux poids:**

$W_{ij}=W_{ij}+\Delta W_{ij}$

$W_{13}=W_{13}+\Delta W_{13}=0.5+0.00381=\mathbf{0.50381}$

$W_{14}=0.9-0.00147=\mathbf{0.89853}$

$W_{23}=0.4+0.00381=\mathbf{0.40381}$

$W_{24}=1-0.00147=\mathbf{0.99853}$

$W_{35}=-1.2-0.0067=$ **-1.2067**

$W_{45}=1.1-0.0112=\mathbf{1.0888}$

$\theta_3=0.8-0.00381=\mathbf{0.79619}$

$\theta_4=-0.1+0.00147=$ **-0.09853**

$\theta_5=0.3+0.0127=\mathbf{0.3127}$

- **Calcul de la nouvelle sortie:**

$y_3=Sig(x_1.W_{13}+x_2.W_{23}-1.\theta_3)=Sig(0.50381+0.40381-0.79619)=\mathbf{0.5278}$

$y_4=Sig(x_1.W_{14}+x_2.W_{24}-1.\theta_4)=Sig(0.89853+0.99853+0.09853)=$**0.8803**

$y_5=Sig(y_3.W_{35}+y_4.W_{45}-1.\theta_5)=Sig(0.5278*-1.2067+0.8803*1.0888-0.3127)=$**0.00887**

Donc:

$e=y_{dés}-y_{cal}=0-0.00887=$ **-0.00887→ de l'ordre de 10^{-3}**

8. Utilisation des instructions sous Matlab pour le calcul neuronal:

8.1. Les réseaux de neurone monocouche:

- **Définir les entrées et les sorties (Target):**

```
P = [0 0 1 1; 0 1 0 1];
T = [0 1 1 1];
```

- **Création du réseau:**

```
Net = newp(P,T)
```

NB: Par défaut la fonction d'activation est: **Step** (**hardlim**), elle peut être:

  - **hardlims**: Fonctions signe: Net=newp(P,T,'hardlims')
  - **logsig**: Fonction logarithme sigmoïde.
  - **tansig**: Fonction tangente sigmoïde.
  - **pureline**: Fonction linéaire.
  - **satlins**: Fonction linéaire à seuil.

- **Voir la sortie (avant l'apprentissage):**

```
Y = sim(Net,P)
```

- **Phase d'apprentissage:**

```
Net = train(Net,P,T);
```

- **Options:**

```
Net.trainParam.epochs = 20; % Maximum number of epochs to train.
Net.trainParam.goal = 1e-5; % Performance goal.
```

- **Voir la sortie (après l'apprentissage):**

```
Y = sim(Net,P)
```

<u>Remarques</u>:

- **Tracer et voir si les états sont linéairement séparables:**

```
P = [-0.5 -0.5  0.3 0; -0.5 0.5 -0.5 1];
T = [1 1 0 0];
plotpv(P,T);
```

- **Calcul des poids pour les entrés et les bais:**

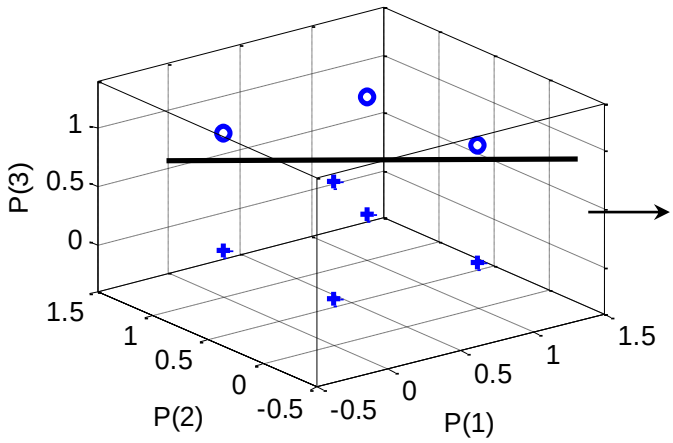
[W,b] = trainp(W,b,P,T)

Exemple d'application:

P = [1 1 1 1 0 0 0 0 ;1 1 0 0 1 1 0 0 ;1 0 1 0 1 0 1 0];

T = [0 1 0 1 0 1 1 1];

plotpv(P,T);



C'est un problème linéairement séparable, il peut être donc traité avec un réseau monocouche

Net = newp(P,T);

Y = sim(Net,P)

Y =

1 1 1 1 1 1 1 1 différente de la sortie désirée: **0 1 0 1 0 1 1 1**

[W,b] = initp(P,T) calcul des poids initiaux pour les entrés et le bais

W =

0.8268 0.2647 -0.8049

b =

-0.4430

Net = train(Net,P,T);

Y = sim(Net,P)

Y =

0 1 0 1 0 1 1 1 égale à la sortie désirée: **0 1 0 1 0 1 1 1**

[W,b] = trainp(W,b,P,T) poids finaux pour les entrés et le bais

W =

-1.1732 -1.7353 -4.8049

b =

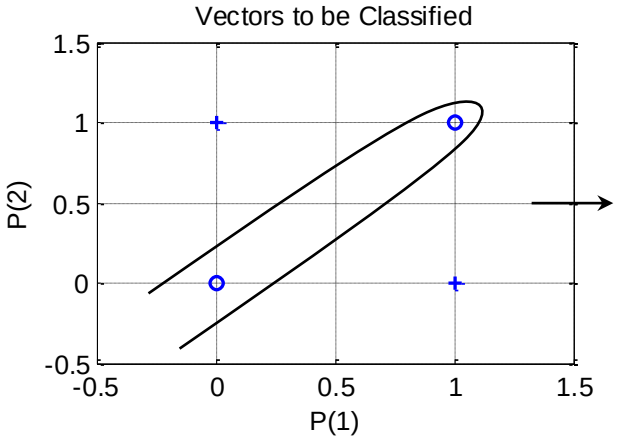
5.5570

8.2. Les réseaux de neurone multicouche:

De même pour le cas des réseaux monocouche, la seule instruction qui va changer c'est celle de la création du réseau. Donc au lieu d'utiliser l'instruction **«newp»**, on utilise l'instruction **«newff»**.

Exemple d'application:

```
P = [0 0 1 1 ; 0 1 0 1];
T = [0 1  1 0];
plotpv(P,T);
```



C'est un problème qui n'est pas linéairement séparable, donc il faut utiliser un réseau multicouche pour le résoudre.

```
Net=newff(P,T,4);  % 4 c'est le nombre des couches cachées
Net.trainFcn='trainlm';  % choix de la fonction d'activation
Net.trainParam.lr=0.1;  % choix du taux d'apprentissage
Net.trainParam.epochs = 50;  % choix du nombre d'itération
Net.trainParam.goal = 0.01;  % choix de l'erreur admissible
Y1=sim(Net,P)
Net=train(Net,P,T);
Y2=sim(Net,P)
Y1 =
  -0.2626    0.5913    1.3496    0.9515
Y2 =
   0.3382    0.9128    1.0042    0.0108
```

La sortie obtenue très proche à la sortie désirée: 0 1 1 0

9. Exemples d'application des RNA dans la commande:

Pour avoir une idée sur la différence; du point de vue performance; entre le régulateur neuronale et le régulateur flou étudié dans la première partie de ce document, on va prendre les mêmes exemples d'application.

9.1. Réglage de niveau dans un réservoir d'eau:

Le schéma bloc du contrôleur neuronal utilisé est présenté par la figure suivante:

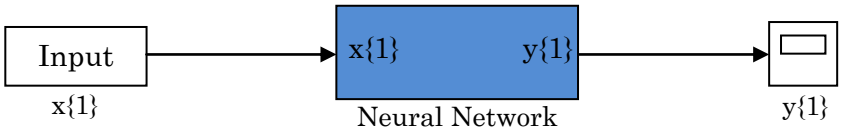


Fig. II.13: *Schéma bloc du contrôleur neuronal*

Les deux couches de ce réseau sont présentées par la figure ci-après:

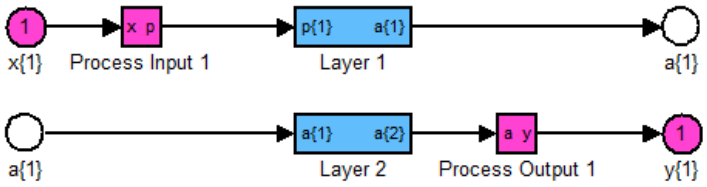


Fig. II.14: *Structure du contrôleur neuronal avec deux couches*

La première couche utilise la fonction «tansig» comme fonction d'activation, elle est présentée par la figure ci-dessous:

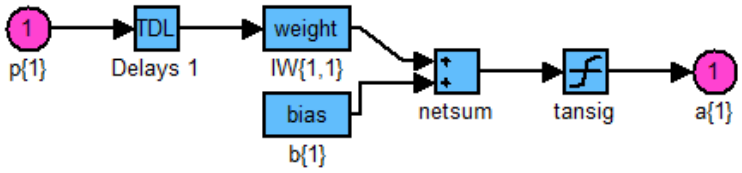


Fig. II.15: *La première couche du contrôleur neuronal*

Cette couche possède 15 neurones comme le montre la figure (Fig. II.16).

La deuxième couche utilise aussi la fonction «tansig» comme fonction d'activation, elle possède un seul neurone (Fig. II.17).

La figure (Fig. II.18) montre les poids de cette deuxième couche.

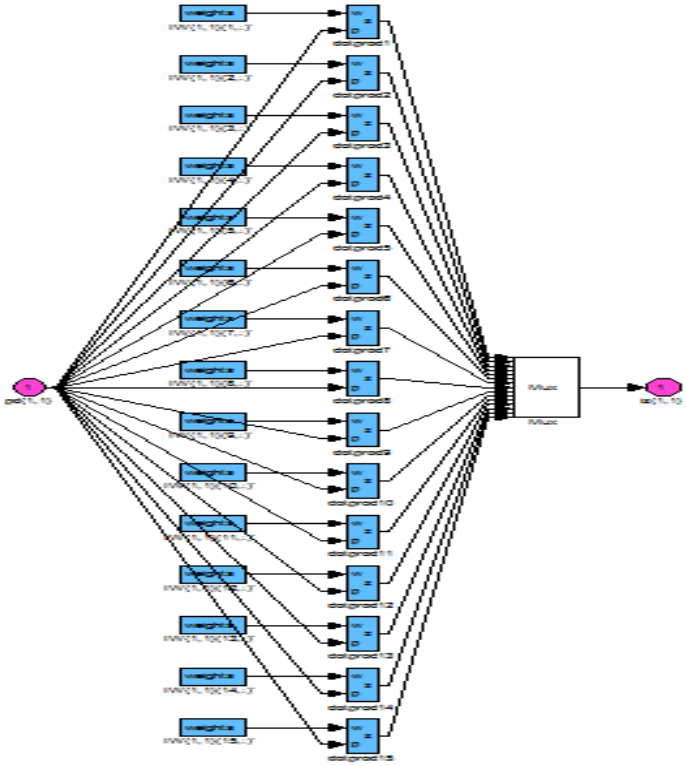


Fig. II.16: *Les poids et les biases de la couche cachée du contrôleur neuronal*

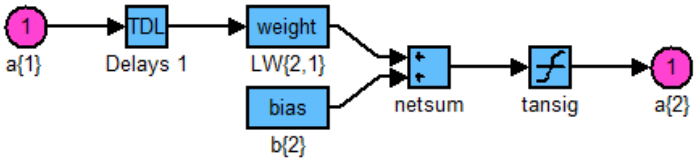


Fig. II.17: *La deuxième couche (la sortie) du contrôleur neuronal*

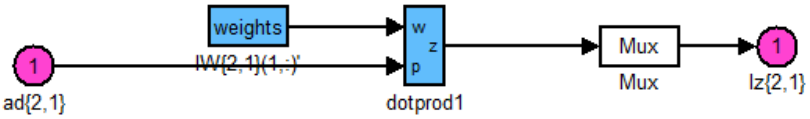


Fig. II.18: *Les poids et les biases de la deuxième couche*

Pour le schéma bloc de simulation, on a juste remplacé dans le diagramme global présenté dans la première partie (page 20), le régulateur PID flou par un régulateur neuronal. Les résultats obtenus sont montrés dans les figures ci-après:

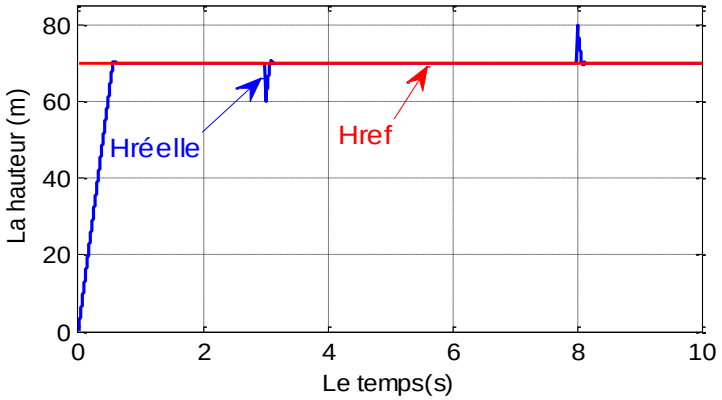


Fig. II.19: *Poursuite du niveau d'eau avec une consigne fixe*

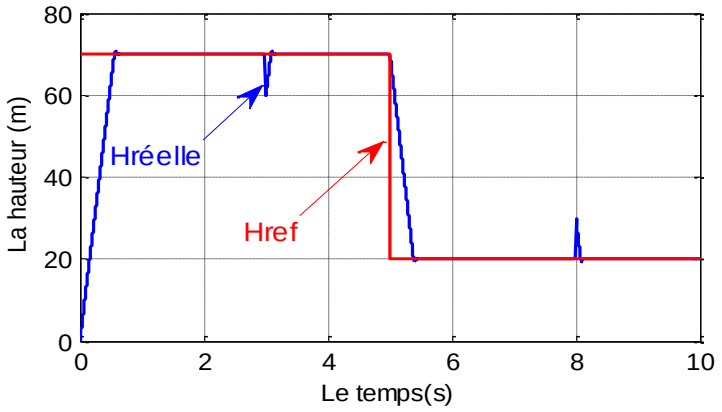


Fig. II.20: *Poursuite du niveau d'eau avec une consigne variable*

On remarque que la hauteur suit parfaitement sa consigne (sa référence), avec un rejet quasi total lors de l'application d'une perturbation.

9.2. Régulation de la température:

Le schéma bloc de simulation est identique à celui présenté dans la première partie (page 22), on change uniquement le régulateur flou par un régulateur neuronal. Les résultats obtenus sont montrés dans les figures ci-après:

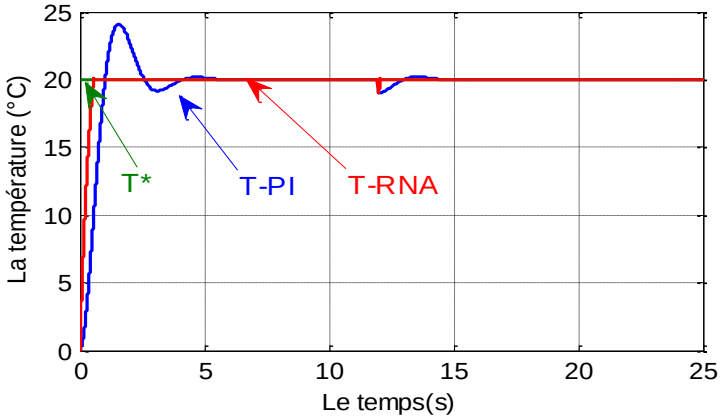


Fig. II.21: *Régulation de la température d'une serre*

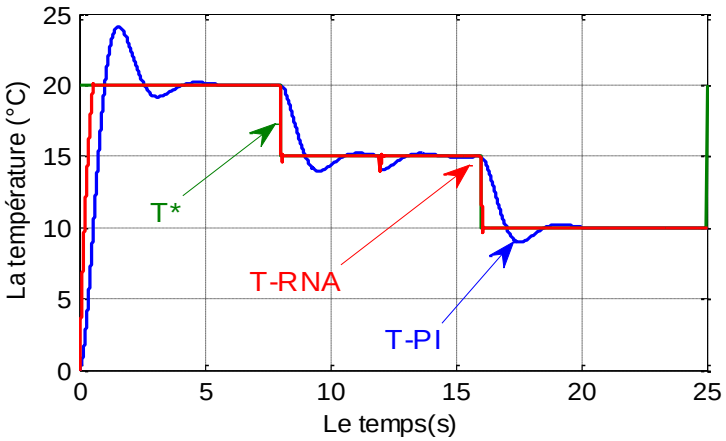


Fig. II.22: *Régulation de la température avec une consigne variable*

D'après les résultats de simulation, on remarque que le régulateur neuronal donne des meilleurs performances par apport aux régulateurs flou et PID en termes de: dépassement, temps de réponse et de rejet de perturbation.

Prenant l'exemple de régulation de la température (Fig. I.18 et Fig. II.21), le tableau ci-dessous présente une étude comparative entre les trois types des régulateurs (PI classique, régulateur flou et régulateur neuronal) en terme de rapidité, dépassement et rejet de perturbation.

Type de régulateur	Critères		
	Rapidité	Dépassement	Rejet de perturbation
PI	Lent	20%	2s
FLC	Rapide	5%	0.6s
ANNC	Très rapide	0.4%	0.09s

10. Conclusion:

Considérée comme l'un des outils de l'intelligence artificielle, la technique de réglage basée sur les réseaux de neurones artificiels présente des bonnes performances et de meilleur robustesse vis-à-vis les perturbations internes et externes.

Le problème du sur-apprentissage, et l'absence d'une méthode systématique permettant de définir la meilleure topologie du réseau et le nombre de neurones à placer dans la (ou les) couche(s) cachée(s), restent les inconvénients majors de cette technique de commande.

Symboles et Abréviations

μ	Degré d'appartenance
e	L'erreur
MCC	Moteur à Courant Continu
R_a	Résistance d'induit
L_a	Inductance de l'inducteur
j	L'inertie.
f	Coefficient de frottement
k_m	Coefficient électromécanique du moteur
I_d	Courant d'induit
E	Force contre électromotrice
k	Constante du flux
w	Vitesse du moteur
T_e	Constante du temps électrique
PID	Régulateur Proportionnel-Intégral-Dérivateur
C_m	Couple moteur
C_r	Couple résistant
C_f	Couple de frottement
FIS	Fuzzy Inference System
NG	Négatif Grand
EZ	Environ Zéro
PG	Positif Grand
FLC	Fuzzy Logic Controller
H	Hauteur d'eau
A	Surface de la section transversale de réservoir.
b	Constante relative au débit à l'entrée du réservoir.
a	Constante relative au débit à la sortie du réservoir
RNA	Réseaux de Neurones Artificiels
d(n)	la réponse désirée
X_m	Les entrées
W_m	Les poids
SOP	Sum Of Product
α	Taux d'apprentissage

Bibliographie

R. ABD JELIL, Modélisation de la relation entre les paramètres du procédé plasma et caractéristiques de la qualité du matériau textile par apprentissage de données physiques, Thèse de Doctorat, Université de Lille 1, France, Avril 2010.

M. ALHANJOURI, Speed Control of DC Motor Using Artificial Neural Network, International Journal of Science and Research, Vol.6, Issue 2, February 2017.

I. BENABADJI, Optimisation d'une base de règles floues: Application à la commande d'un drone. Mémoire de Magister, Université des sciences et de la technologie d'Oran, 2011.

I. BENDAAS, Contribution à la commande hybride par mode glissant floue appliquée à un moteur à induction. Apport des techniques de l'intelligence artificielle, Thèse de Doctorat, Université de Batna, Algérie, Avril 2016.

P. BORNE, M. BENREJEB, J. HAGGEGE, Les réseaux de neurones: Présentation et applications, Editions TECHNIP, France, 2007.

A. BORNI, Étude et optimisation d'un multi système hybride de conversion d'énergie électrique, Thèse de Doctorat, Université de Constantine 1, Algérie, Mars 2015.

M. BSISS, Sûreté de fonctionnement d'un système d'inférence floue avec une architecture redondante un parmi deux avec diagnostic (1oo2D) à base de FPGA. Thèse de Doctorat, Université Abdelmalek Essaadi de Tanger, 2013.

A. CHAIBA, Commande de la machine asynchrone à double alimentation par des techniques de l'intelligence artificielle, Thèse de Doctorat, Université de Batna, Algérie, Juillet 2010.

S. CHENNAI, Etude, modélisation et commande des filtres actifs : apport des techniques de l'intelligence artificielle, Thèse de Doctorat, Université de Biskra, Algérie, Septembre 2014.

R. KETATA, Méthodologies de régulation numérique incluant la logique floue. Thèse de Doctorat, L.A.A.S Toulouse, 1992.

D. MOKEDDEM, Contrôle Flou des Processus Biotechnologiques à Base d'Algorithmes Génétiques. Thèse de Doctorat, Université de Sétif, 2010.

K. NABTI, Contribution à la commande de la machine asynchrone par DTC et logique floue, Mémoire de Magister, Université de Constantine, Algérie, Juillet 2006.

B. REGUIG, Modélisation et commande floue d'une génératrice asynchrone à double alimentation: Application à l'énergie éolienne. Mémoire de Master, Université de M'sila, 2016.

C. TOUZET, Les réseaux de neurones artificiels introduction au connexionnisme, cours, exercices et travaux pratiques, Juillet 1992.

Z. ZERDOUMI, Application des réseaux de neurones artificiels à la poursuite des non linéarités fluctuantes des systèmes satellitaires, Mémoire de Magister, Université de M'sila, 2006.