الجمهـــوريـــــة الجزائـــريـــــة الديمقـــراطيـــة الشـعبيـــة

République Algérienne Démocratique et Populaire

وزارة التـــعلـــيـــــم العـــــالـــــي والبـــحـــــث العـلمـــي

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université d'Oran II Mohamed Ben Ahmed **جامعة وهران 2 محمد بن أحمد**

Institut de Maintenance et de sécurité industrielle **معهد الصيانة والأمن الصناعي**

Département de Maintenance en Instrumentation

MÉMOIRE

de fin d'études pour l'obtention du diplôme de Master

Filière : Génie industriel

Spécialité : Génie industriel

Thème :

Le rôle de l'apprentissage par ANFIS dans les choix de spécialités pour les étudiants

Présenté par : **LEILA MACHOU**

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
GUENDOUZ DJILALIA	PR	IMSI (Université d'Oran 2)	Président
LAZRAG MALIKA	MCA	IMSI (Université d'Oran 2)	Encadreur
BELKACEM BELKACEM	MCA	IMSI (Université d'Oran 2)	Examinateur

Année universitaire : 2021/2022

Remerciements

Je remercie ma mère en premier lieu qui je porte son prénom avec toute fierté… **Mama** la personne qui me donne toujours le courage de continuer à vivre encore... Que je fasse ou que je dise, je ne saurai point te remercier comme il se doit.

Mon **père** qui m'a soutenu à sa façon !

Vous m'avez supporté toutes ces années alors que j'étais vraiment têtu Je vous aime !

Ma petite sœur **Maroua** que j'aime trop que dieu la protège.

Mon petit frère **YOUNES** mon bébé mon amour <3..

Mes deux chères tante **FATIMA** et **SAIDA** qui ont été toujours là pour moi!!

Ma GRANDE SŒUR ABLA qui me manque tellement ...

Ma **grand-mère** la plus cher après ma mère...

Je **me** remercie... Enfin ..Je mérite ce remerciement parce que j'ai bien combattu contre mes problèmes psychologiques toute seule et je suis enfin arrivée à un certain bonheur mental grâce à dieu et grâce aux gens que j'aime.

Je remercie madame LAZRAG MALIA , Madame Guendouz Djilalia , Monsieur Belkacem Belkacem

Résumé :

De nos jours, la prédiction de la durée d'exécution d'un projet de conception est devenue d'une importance cruciale pour les gestionnaires.

Mon travail s'inscrit dans ce contexte en se basant sur la proposition d'une approche intégrée sur l'utilisation conjointe des réseaux de neurones artificiels (RNA), le système d'inférence floue (FIS) et le système adaptatif d'inférence neuro-floue (ANFIS).

Ce mémoire consiste trois chapitres, les deux premiers chapitres contiennent des notions théoriques et des explications, et le dernier chapitre est pour la simulation.

Mots clés : Apprentissage, ANFIS, floue, Matlab….

Table des matières :

Liste des figures

Liste des tableaux

Introduction générale

Récemment, de nombreuses applications d'appareils intérieurs, extérieurs et mobiles, comme la domotique et la sécurité, l'automatisation de l'espace des véhicules, les produits de consommation, les soins de santé, la surveillance de l'environnement et l'identification des emplacements intérieurs, etc., ont été développées pour améliorer la qualité de vie humaine.

L'ANFIS a été proposé il y a de nombreuses années et est largement utilisé dans les travaux de recherche. L'ANFIS révèle un réseau d'apprentissage efficace et ses applications peuvent être trouvées dans de nombreux ouvrages de la littérature. Le contrôleur ANFIS est utilisé pour contrôler les moteurs des ventilateurs pour le contrôle de la température intérieure.

Le contrôleur ANFIS est un algorithme convivial et sert de méthode pour induire de nombreuses règles floues avec des fonctions d'adhésion appropriées pour générer les paires d'entrées/sorties (E/S) de mémoire associée floue (FAM) et des règles floues raisonnables afin d'établir un système d'inférence flou (SIF). Un contrôleur fiable peut être conçu sur la base de ce FIS.

MATLAB™ Ltd. a fourni aux ingénieurs des outils très utiles et conviviaux pour concevoir ce contrôleur ANFIS.

Ce mémoire est constitué de trois chapitres :

Dans le premier chapitre, nous avons deux parties ; dans la première partie nous allons parler d'intelligence artificielle, la logique floue, des réseaux de neurones et en particulier de la méthode ANFIS. Et pour la deuxième partie ; un état de l'art sur les travaux avec ANFIS.

Dans le deuxième chapitre, une partie sur l'étude algorithmique de la méthode ANFIS, et l'autre est pour l'optimisation (méthode des moindres carrés et rétro propagation).

Le troisième et dernier chapitre est concernant la partie application ; simulation avec le logiciel MATLAB.

Chapitre 1 :

Généralités

Chapitre 1 : Généralités

Partie I :

1. Introduction :

Ce chapitre présente une méthode de commande moderne appelée logique floue. La logique floue diffère de la logique conventionnelle parce qu'elle permet de définir partiellement ou "flouement" les règles de contrôle.
La puissance de la logique floue vient de sa capacité à décrire un phénomène ou processus particulier de façon linguistique, puis de représenter ce phénomène par un faible nombre de règles. La connaissance d'un système flou est contenue dans des règles et des ensembles flous contenant des descriptions générales des propriétés du phénomène en question.

2. L’intelligence artificielle :

L'intelligence artificielle (IA) est un ensemble de technologies qui permettent aux machines d'effectuer des tâches et de résoudre des problèmes normalement réservés.

Les tâches de l'IA peuvent être très simples pour les humains, comme conduire une voiture. Une planification complexe peut être nécessaire, comme aux échecs ou au Go. Les tâches les plus complexes, comme traduire des textes ou animer un dialogue, demandent beaucoup de connaissances et de bon sens.

Malgré toutes ces avancées, nous avons encore un long chemin à parcourir pour créer des machines aussi intelligentes que les humains.

Bien sûr, nous avons des systèmes capables d'accomplir d'autres tâches difficiles de manière plus fiable et plus rapide que la plupart des humains, mais ce qui nous manque, ce n'est pas seulement de représenter le monde, mais de nous en souvenir, d'en discuter, de le prévoir, de planifier.

2.1.L’apprentissage machine :

L'apprentissage pilote les systèmes de toutes les grandes sociétés Internet.

Ils l'utilisent depuis longtemps pour filtrer le contenu indésirable, classer les réponses de recherche, faire des recommandations ou sélectionner des informations qui intéressent chaque utilisateur.

Dans sa forme la plus couramment utilisée, l'apprentissage automatique est supervisé. L'entrée de la machine est représentée par une image d'un objet et reçoit la sortie souhaitée pour cet objet.

Ainsi, la machine ajuste ses paramètres internes pour rapprocher les performances des performances souhaitées. Après avoir présenté aux machines des milliers d'exemples catégorisés, elle peut classer la plupart d'entre eux correctement.

3. La méthode ANFIS :

ANFIS (Adaptive Neuro Fuzzy Inference System) est un système d'inférence adaptatif neuro-flou qui consiste à utiliser un réseau de neurones à 5 couches (figure 1) pour lequel chaque couche correspond à la réalisation d'une étape d'un système d'inférence floue de type Takagi Sugeno.

Pour la simplicité, nous supposons que le système d'inférence flou à deux entrées x et y, et à comme une sortie f. Supposer que la base de règle contient deux règles floues de type Takagi-Sugeno :

Règle l :
SI x est Al et y est B1 ALORS f1 = p1 x+q1 y +r1
Règle 2 :
SI x est A2 et y est B2 ALORS f2= p2 x+q2 y +r2

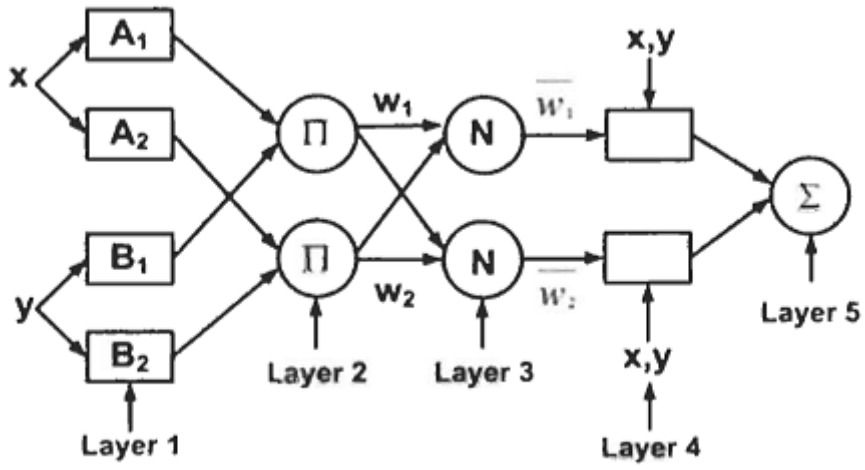


Figure 1 - L'Architecture de l'ANFIS

L'architecture classique d'un modèle ANFIS peut être décrite de la manière suivante :

1. La première couche comporte autant de neurones qu'il y' a de sous-ensembles flous dans le système d'inférence représenté. Chaque neurone calcule le degré de vérité d'un sous ensemble flou particulier par sa fonction de transfert.

2. La deuxième couche cachée sert à calculer le degré d'activation des prémisses. Les neurones reçoivent en entrée le degré de vérité des différents sous-ensembles flous composant cette prémisse et ont en charge le calcul de son propre degré de vérité.

3. La troisième couche cachée nom1alise de degré d'activation des règles. L'ensemble des sorties de cette couche seront appelées les poids normalisés.

4. La quatrième couche cachée sert à déterminer les paramètres de la partie conséquence des règles.

5. La couche de sortie contient un seul neurone dans cette couche et c'est un neurone qui calcule la sortie globale comme l'addition de tous les signaux entrants.

4. La logique floue :

La logique floue (fuzzy logic, en anglais) est une technique utilisée en intelligence artificielle. Elle a été formalisée par Lotfi Zadeh en 1965 et utilisée dans des domaines aussi variés que l'automatisme (freins ABS), la robotique (reconnaissance de formes), la gestion de la circulation routière (feux rouges), le contrôle aérien, l'environnement (météorologie, climatologie, sismologie, analyse du cycle de vie), la médecine (aide au diagnostic), l'assurance (sélection et prévention des risques) et bien d'autres.

Elle s'appuie sur la théorie mathématique des ensembles flous. Cette théorie, introduite par Zadeh, est une extension de la théorie des ensembles classiques pour la prise en compte d'ensembles définis de façon imprécise. C'est une théorie formelle et mathématique dans le sens où Zadeh, en partant du concept de fonction d'appartenance pour modéliser la définition d'un sous-ensemble d'un univers donné, a élaboré un modèle complet de propriétés et de définitions formelles. Il a aussi montré que cette théorie des sous-ensembles flous se réduit effectivement à la théorie des sous-ensembles classiques dans le cas où les fonctions d'appartenance considérées prennent des valeurs binaires ({0,1}).

Elle présente aussi l'intérêt d'être plus facile et meilleur marché à implémenter qu'une logique probabiliste, bien que cette dernière seule soit stricto sensu cohérente (voir Théorème de Cox-Jaynes). Par exemple la courbe Ev(p) peut être remplacée par trois segments de droite sans perte excessive de précision pour beaucoup d'applications.

5. Ensemble flou :

En logique floue, un ensemble flou contient plusieurs valeurs. L'ensemble floue est concerné par un degré d'appartenance (ou degré de vérité). On utilise un continuum de valeurs logiques entre 0 (complètement faux) et 1 (complètement vrai). Une fonction d'appartenance est utilisée pour mapper un item X dans le domaine des nombres réels à un intervalle de 0 à 1, ce qui permet un degré de vérité.

L'appartenance à un ensemble représente une valeur entre 0 et 1. Un ensemble flou peut être défini comme un ensemble ayant des frontières floues. Un ensemble flou est définit comme suit : soit S un ensemble et x un membre de cet ensemble.

Un sous-ensemble flou F de S est défini par une fonction d'appartenance µF(x) qui mesure le degré auquel x appartient à F.

Un exemple :

Soit S un ensemble des entiers positifs et F un sous-ensemble flou de petits entiers. Des entiers peuvent avoir une distribution de probabilité qui indiquent leur appartenance au sous-ensemble flou F : µF(1) = 1.0, µF(2) = 1.0, µF(3) = 0.9, ... µF(30) = 0.01.

La figure 2 montre cette fonction d'appartenance. Dans la théorie des ensembles flous, l'ensemble flou A de X (où X est l'univers d'étude).

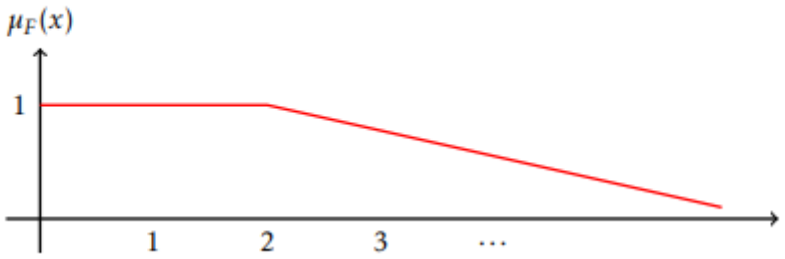


Figure 2 - Représentation du sous-ensemble flou F des petits entiers

Est définit comme une fonction :

$$\mu A(x) : X \rightarrow 0,1$$

Où $\mu A(x) = 1$ si x est totalement dans A, $\mu A(x) = 0$ si x n'est pas dans A et $0 < \mu A(x) < 1$ si x est partiellement dans A. La fonction d'appartenance est une mesure :

• Du degré auquel un élément est membre d'un ensemble.

• Du degré d'appartenance.

• De la valeur de l'appartenance.

• Du degré de confiance.

La logique floue permet de transformer plusieurs valeurs réelles en quelques variables floues avec différentes appartenances, ce qui permet de réduire le nombre de règles.

On utilise ces règles pour faire la commande d'un système.

Le tableau 1 compare des règles de commande classiques a la logique floue pour un système de chauffage, où v est la vitesse du ventilateur.

Tableau 1 - Comparaison entre la logique classique et la logique floue pour un système de chauffage

Logique classique	Logique floue
Si T= 21° ALORS υ = 30%	Si T = élevée ALORS υ = lent
Si T = 22° ALORS υ = 40%	Si T = assez élevée ALORS υ = rapide
Si T = 23° ALORS υ = 50%	Si T = très élevée ALORS υ = très rapide
Si T = 24° ALORS υ = 60%	
Si T = 25° ALORS υ = 75%	
Si T > 26° ALORS υ = 100%	

Pour représenter la logique floue sur ordinateur, il faut déterminer la fonction d'appartenance. On utilise le plus souvent une représentation graphique.

Les bornes de ces graphiques proviennent d'experts dans le domaine. La figure 3 montre deux exemples de représentation de la température, une en logique classique, et l'autre en logique floue. Selon la figure 3, en logique classique, une température de 22.5° est considérée comme élevée.

En logique floue, une température de 22.5° appartient au groupe « moyenne » avec un dégrée d'appartenance de 0.167, et appartient au groupe « élevée » avec un degré d'appartenance de 0.75.

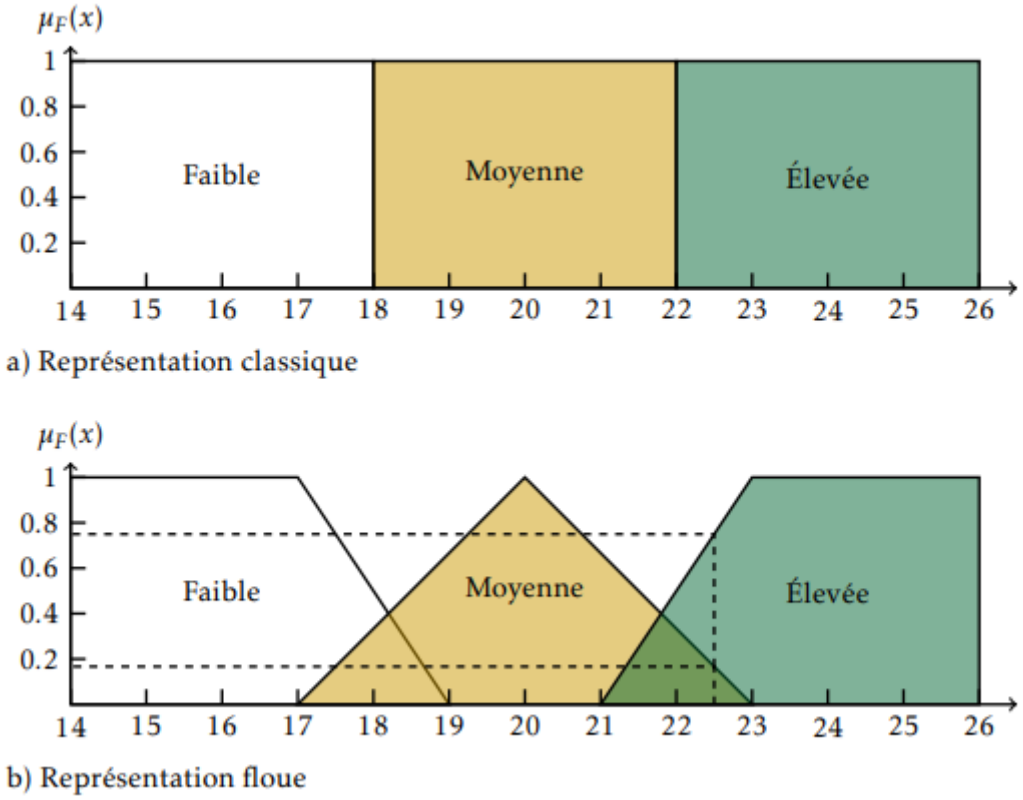


Figure 3 - Comparaison de l'appartenance de la température en logique classique vs la logique floue

Les variables floues faibles, moyennes et élevées sont représentées par des fonctions linéaires. D'autres fonction aurait pu être utilisées, comme des trapézoïdes, des paraboles, etc. Cependant, les fonctions linéaires sont beaucoup plus faciles à implémenter de façon pratique, et donnent de bons résultats.

On utilise souvent une notation vectorielle pour représenter les fonctions. Pour les fonctions d'appartenance de la figure 3, on peut utiliser la notation suivante :

• Température faible : (1/17, 0/19).

• Température moyenne : (0/17, 1/20, 0/23).

• Température élevée : (0/21, 1/23).

6. Sous-ensembles flous :

La logique floue repose sur la théorie des ensembles flous, qui est une généralisation de la théorie des ensembles classiques [Zadeh, 1965]. Par abus de langage, suivant les us de la littérature, nous utiliserons indifféremment les termes sous-ensembles flous et ensembles flous. Les ensembles classiques sont également appelés ensemble nets, par opposition à flou, et de même la logique classique est également appelée logique booléenne ou binaire.

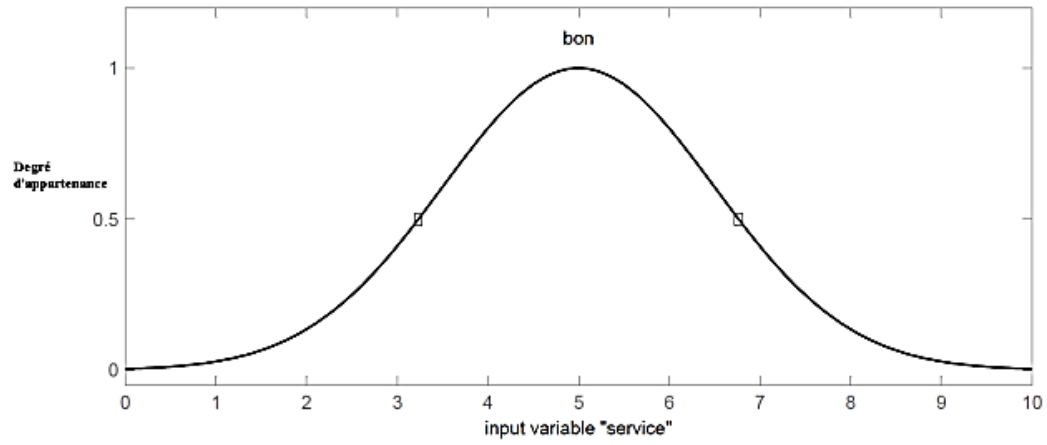


Figure 4 - Fonction d'appartenance caractérisant le sous-ensemble 'bon' de la qualité du service

La figure 4 montre la fonction d'appartenance choisie pour caractériser le sous-ensemble 'bon' de la qualité du service.

Définition 1 :

Soit X un ensemble. Un sous-ensemble flou A de X est caractérisé par **une fonction d'appartenance** $f^{a} : X \rightarrow [0, 1]$.

Note :

Cette fonction d'appartenance est l'équivalent de la fonction caractéristique d'un ensemble classique.

Dans notre exemple du pourboire, il nous faudra redéfinir des fonctions d'appartenance pour chaque sous-ensemble flou de chacune de nos trois variables :

– **Input 1 :** qualité du service. Sous-ensembles : mauvais, bon et excellent.

– **Input 2 :** qualité de la nourriture. Sous-ensembles : exécrable et délicieux.

– **Output :** montant du pourboire. Sous-ensembles : faible, moyen et élevé.

La forme de la fonction d'appartenance est choisie arbitrairement en suivant les conseils de l'expert ou en faisant des études statistiques : formes sigmoïde, tangente hyperbolique, exponentielle, gaussienne ou de toute autre nature sont utilisables.

La figure 5 montre graphiquement la différence entre un ensemble classique et l'ensemble flou correspondant à une nourriture délicieuse.

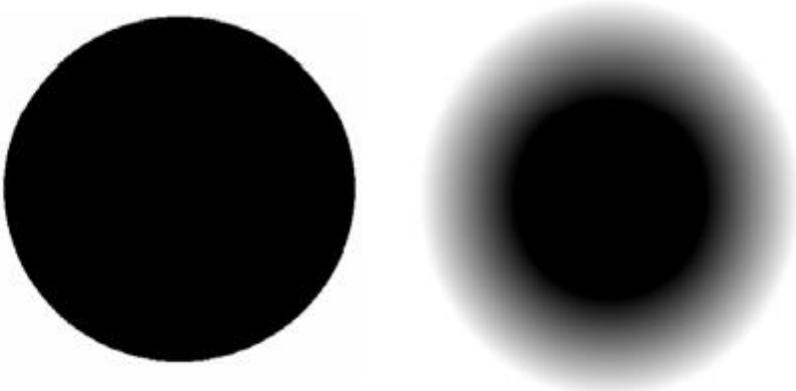


Figure 5 - Représentation graphique d'un ensemble classique et d'un ensemble flou

La figure 6 compare les deux fonctions d'appartenance correspondant aux ensembles précédents.

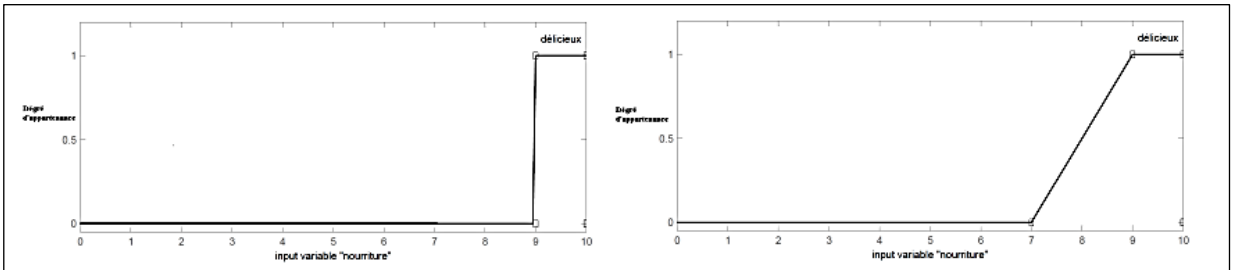


Figure 6 - Comparaison entre fonction caractéristique d'un ensemble classique et fonction d'appartenance d'un ensemble flou

Pour pouvoir définir les caractéristiques des ensembles flous, nous redéfinissons et étendons les caractéristiques usuelles des ensembles classiques.

Soit X un ensemble, A un sous-ensemble flou de X et μA la fonction d'appartenance le caractérisant.

Définition 2 :

La hauteur de A, notée h(A), correspond à la borne supérieure de l'ensemble d'arrivée de sa fonction d'appartenance : $h(A) = \sup \{\mu A(x) \mid x \in X\}$.

Définition 3 :

A est dit normalisé si et seulement si $h(A) = 1$. En pratique, il est extrêmement rare de travailler sur des ensembles flous non normalisés.

Définition 4 :

Le support de A est l'ensemble des éléments de X appartenant au moins un peu à A. Autrement dit, c'est l'ensemble $supp(A) = \{x \in X \mid \mu A(x) > 0\}$.

Définition 5 :

Le noyau de A est l'ensemble des éléments de X appartenant totalement à A. Autrement dit, c'est l'ensemble $noy(A) = \{x \in X \mid \mu A(x) = 1\}$. Par construction, $noy(A) \subseteq supp(A)$. **Définition 6 :**

Une α-coupe de A est le sous-ensemble classique des éléments ayant un degré d'appartenance supérieur ou égal à α : $\alpha\text{-coupe}(A) = \{x \in X \mid \mu A(x) > \alpha\}$.

Une autre fonction d'appartenance pour un pourboire moyen sur lequel nous avons fait figurer les propriétés précédentes est présentée sur la figure 7.

Nous remarquons que si A était un ensemble classique, nous aurions simplement $supp(A) = noy(A)$ et $h(A) = 1$ (ou $h(A) = 0$ si $A = \emptyset$).

Nos définitions permettent donc bien de retrouver les propriétés usuelles des ensembles classiques.

7. Les variables linguistiques :

Le concept de fonction d'appartenance vu précédemment nous permettra de définir des systèmes flous en langage naturel, la fonction d'appartenance faisant le lien entre logique floue et variable linguistique que nous allons d´définir à présent.

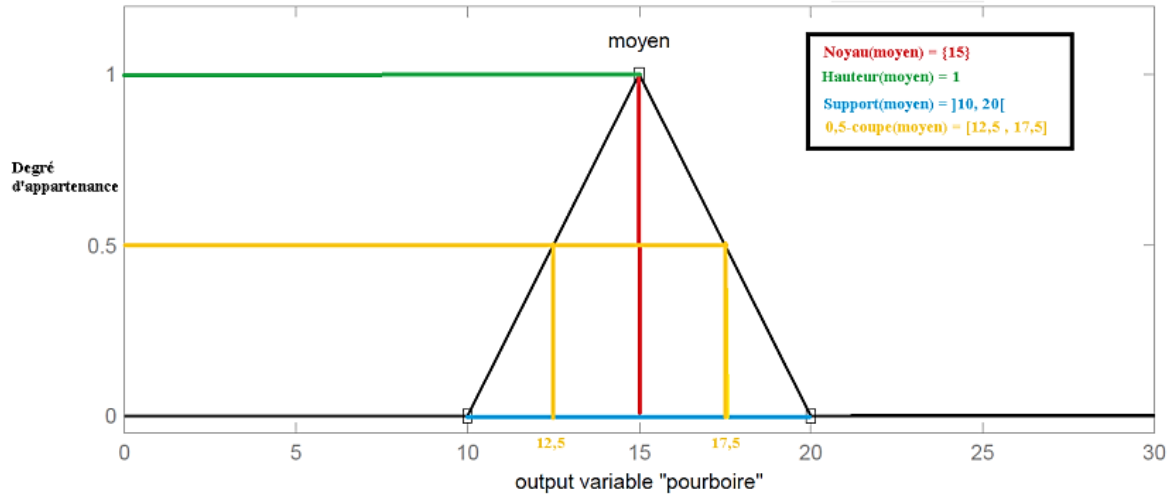


Figure 7 - Propriétés d'un ensemble flou

Définition 7 :

Soit V une variable (qualité du service, montant du pourboire, etc.), X la plage de valeurs de la variable et T_V un ensemble fini ou infini de sous-ensembles flous. Une variable linguistique correspond au triplet (V, X, T_V).

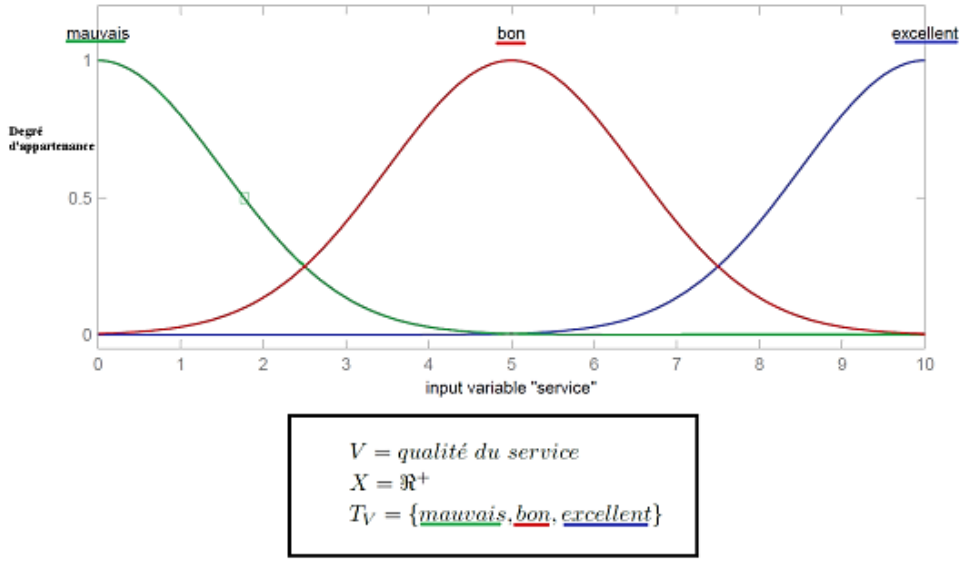
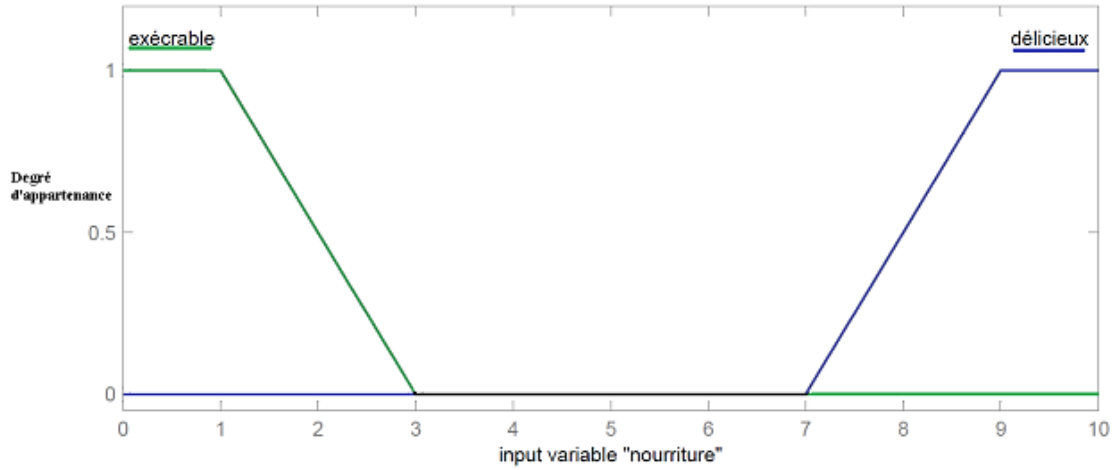
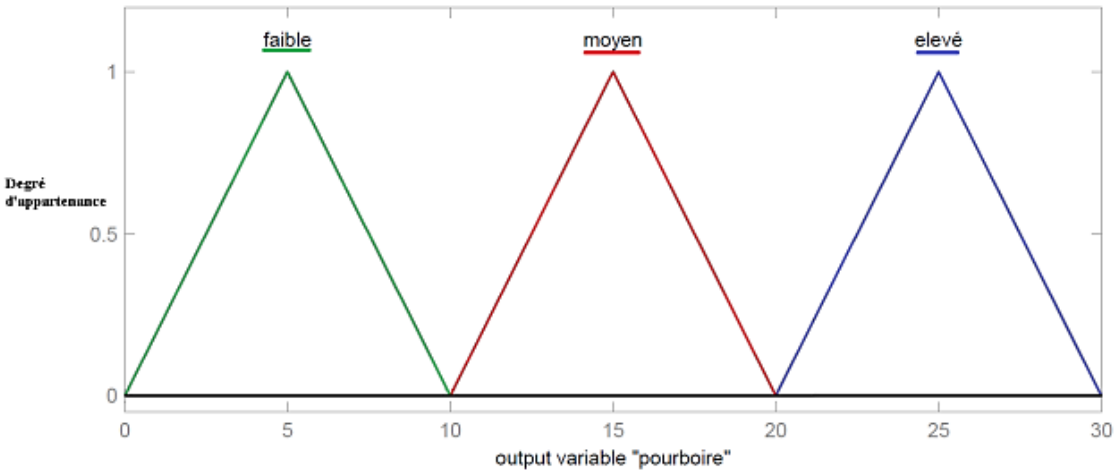


Figure 8 - Variable linguistique 'qualité du service'



$$V = qualité\ de\ la\ nourriture$$
$$X = \Re^{+}$$
$$T_V = \{exécrable, délicieux\}$$

Figure 9 - Variable linguistique 'qualité de la nourriture'



$$V = montant\ du\ pourboire$$
$$X = \Re^{+}$$
$$T_V = \{faible, moyen, élevé\}$$

Figure 10 - Variable linguistique 'montant du pourboire'

8. Opérateurs flous :

Afin de pouvoir manipuler aisément les ensembles flous, nous redéfinissons les opérateurs de la théorie des ensembles classiques afin de les adapter aux fonctions d'appartenance propres à la logique floue permettant des valeurs strictement entre 0 et 1.

Contrairement aux définitions des propriétés des ensembles flous qui sont toujours les mêmes, la définition des opérateurs sur les ensembles flous est choisie, à l'instar des fonctions d'appartenance.

Voici les deux ensembles d'opérateurs pour le complément (NON), l'intersection (ET) et l'union (OU) utilisés le plus couramment :

Dénomination	Intersection ET : $\mu_{A\cap B}(x)$	Réunion OU ! $\mu_{A\cap B}(x)$	Complément NON : $\mu_{\bar{A}(x)}$
Opérateurs de Zadeh MIN/MAX	min $(\mu_A(x), \mu_B(x))$	max $(\mu_A(x), \mu_B(x))$	$1 - \mu_A(x)$
Probabiliste PROD/PROBOR	$\mu_A(x) \times \mu_B(x)$	$\mu_A(x) + \mu_B(x)$ $- \mu_A(x) \times \mu_B(x)$	$1 - \mu_A(x)$

Avec les définitions usuelles des opérateurs flous, nous retrouvons toujours les propriétés de commutativité, distributivité et associativité des opérateurs classiques.

Cependant, relevons deux exceptions notables :

– En logique flou, le principe du tiers exclu est contredit : $A \cup \bar{A} \neq X$, autrement dit $\mu_{A\cap\bar{A}}(x) \neq 1$.

– En logique floue, un élément peut appartenir à A et non A en même temps : $A \cup \bar{A} \neq \emptyset$, autrement dit $\mu_{A\cap\bar{A}}(x) \neq 0$. Notons que ces éléments correspondent à l'ensemble supp(A) − noy(A).

9. Raisonnement en logique floue :

En logique classique, les raisonnements sont de la forme :

$$\begin{cases} Si\ p\ alors\ q \\ p\ vrai\ alors\ q\ vrai \end{cases}$$

En logique floue, le raisonnement flou, également appelé raisonnement approximatif, se base sur des règles floues qui sont exprimées en langage naturel en utilisant les variables linguistiques dont nous avons donné la définition précédemment. Une règle floue aura cette forme :

Si x ∈ A et y ∈ B alors z ∈ C, avec A, B et C des ensembles flous.

Par exemple :

« Si (la qualité de la nourriture est délicieuse), alors (le pourboire sera élevé) ».

La variable 'pourboire' appartient à l'ensemble flou 'élevé' `a un degré qui dépend du degré de validité de la prémisse, autrement dit du degré d'appartenance de la variable 'qualité de la nourriture' à l'ensemble flou 'délicieux'.

L'idée sous-jacente est que plus les propositions en prémisse sont vérifiées, plus l'action préconisée pour les sorties doit être respectée. Pour connaître le degré de vérité de la proposition floue 'le pourboire sera élevé', nous devons définir l'implication floue.

A l'instar des autres opérateurs flous, il n'existe pas de définition unique de l'application floue : le concepteur du système flou devra choisir parmi le large choix d'implications floues déjà définies, ou bien la définir à la main. Voici les deux définitions de l'implication floue les plus couramment utilisées :

Nom	Valeur de vérité
Mamdani	$\min(f_a(x), f_b(x))$
Larsen	$f_a(x) \times f_b(x)$

Fait notable, ces deux implications ne généralisent pas l'implication classique. Il existe d'autres d´définitions d'implication floue la généralisant, mais elles sont moins utilisées. Si nous choisissons l'implication de Mamdani, voici ce que nous obtenons pour la règle floue 'Si (la qualité de la nourriture est délicieuse), alors (le pourboire sera élevé)' lorsque la qualité de la nourriture est notée 8,31 sur 10 :

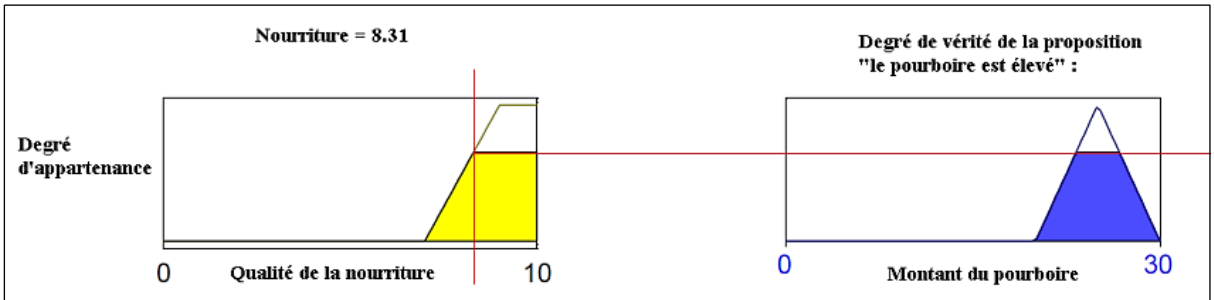


Figure 11 - Exemple d'implication floue

Le résultat de l'application d'une règle floue d´dépend donc de trois facteurs :

1. la définition d'implication floue choisie ;

2. la définition de la fonction d'appartenance de l'ensemble flou de la proposition située en conclusion de la règle floue ;

3. le degré de validité des propositions situées en prémisse.

Comme nous avons défini les opérateurs flous ET, OU et NON, la prémisse d'une règle floue peut très bien être formée d'une conjonction de propositions floues. L'ensemble des règles d'un système flou est appelé la matrice des décisions.

Voici celui de notre exemple du pourboire :

Si le service est mauvais ou la nourriture est exécrable	Alors le pourboire est faible
Si le service est bon	Alors le pourboire est moyen
Si le service est excellent ou la nourriture est délicieuse	Alors le pourboire est élevé

La figure 12 montre nous obtenons pour la règle floue 'Si (le service est excellent ou la nourriture est délicieuse), alors (le pourboire sera élevé)' lorsque la qualité du service est notée 7,83 sur 10 et la qualité de la nourriture 7,32 sur 10 si nous choisissons l'implication de Mamdani ainsi que la traduction du OU par MAX.

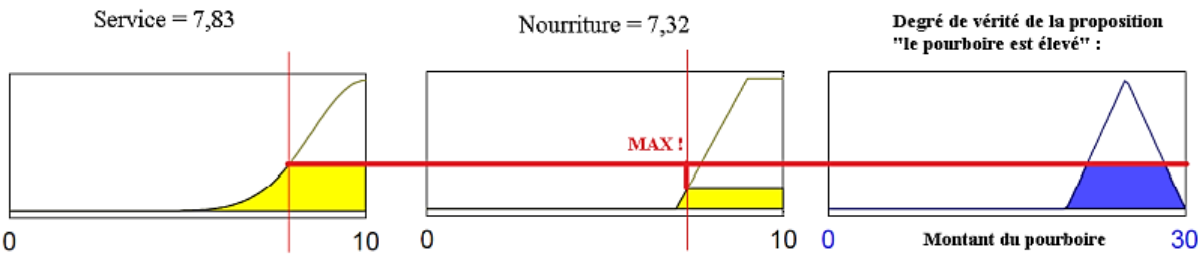


Figure 12 - Exemple d'implication floue avec conjonction OU traduite par un MAX

Nous allons maintenant appliquer l'ensemble des 3 règles de notre matrice des décisions. Cependant, nous allons obtenir 3 ensembles flous pour le pourboire : nous les agrégerons par l'opérateur MAX qui est presque toujours utilisé pour l'agrégation.

Comme nous le voyons, il ne nous reste plus qu'à prendre la décision finale, à savoir quel pourboire nous allons réellement donner sachant que la qualité du service est notée 7,83 sur 10 et la qualité de la nourriture 7,32 sur 10.

10.Règles floues :

Une règle floue est une déclaration de la forme suivante :

SI x est A ALORS y est B

Où x et y sont des variables linguistiques, et A et B sont des valeurs linguistiques, déterminées par les ensembles flous sur les ensembles X et Y. Une variable linguistique est une variable floue. Par exemple : La tension est haute. La variable linguistique tension prend la valeur linguistique élevée. La plage de valeurs linguistiques possibles d'une règle représente l'univers de cette variable. Un exemple de règle floue est : `

SI vitesse est lente ALORS arrêt est court

La variable vitesse peut avoir une plage de valeurs entre 0 et 220 km/h. On peut inclure des sous-ensembles flous (très lent, lent, moyenne, rapide, très rapide) pour modifier cette règle. Chaque sous-ensemble flou représente une valeur linguistique pour la variable.

La logique classique (SI – ALORS) utilise la logique binaire. La logique floue permet d'associer une plage de valeurs (un ensemble flou) à des variables linguistiques. On peut réduire le nombre de règles jusqu'à 90% en utilisant la logique floue.

11.Fuzzification :

La fuzzification est l'opération de rendre une entrée classique en valeur linguistique. Des valeurs d'entrée sont traduites en concepts linguistiques représentés comme des ensembles flous. Les fonctions d'appartenance sont appliquées aux mesures et des degrés de vérité sont établis pour chaque proposition. Les règles d'inférence permettent de calculer les valeurs d'appartenance de règles qui ont plusieurs antécédents. Si la conjonction qui unit deux antécédents est ET (AND), on prend le minimum des deux.

Ex : SI voiture a de l'essence ET voiture a un moteur ALORS voiture peut fonctionner Supposons que la voiture a le plein d'essence (appartenance 1.0), mais qu'elle n'a pas de moteur (appartenance 0).

La règle doit-elle être déclenchée ?

Bien sûr que non ; une voiture sans moteur ne fonctionne pas. Si la conjonction qui unit deux antécédents est OU (OR), on prend le maximum des deux.

Ex : SI le chien jappe OU il fait très froid dehors ALORS ouvrir la porte. Supposons que le chien ne jappe pas (appartenance 0.0), mais qu'il fait très froid dehors (appartenance 1.0).

Doit-on ouvrir la porte au chien ? Bien sûr que oui ; on ne va pas laisser le chien dehors en temps très froid. Les conclusions atteintes par les systèmes flous sont des faits flous ayant des degrés d'appartenance.

Ex : risque est faible avec une appartenance de 0.5. Cependant, le déroulement final doit être une décision concrète ; ex : louer de l'argent. Le processus de transformer un fait flou en un fait net est la défuzzification.

12.L'inférence floue :

L'inférence consiste à utiliser le moteur d'inférence, qui est un mécanisme permettant de condenser l'information d'un système à travers d'un ensemble de règles définies pour la représentation d'un problème quelconque. Chaque règle délivre une conclusion partielle qui est ensuite agrégée aux autres règles pour fournir une conclusion (agrégation). Les règles constituent le système d'inférence floue, dans la suite de ce chapitre nous donnons une description de règles floue dans un cadre plus formel.

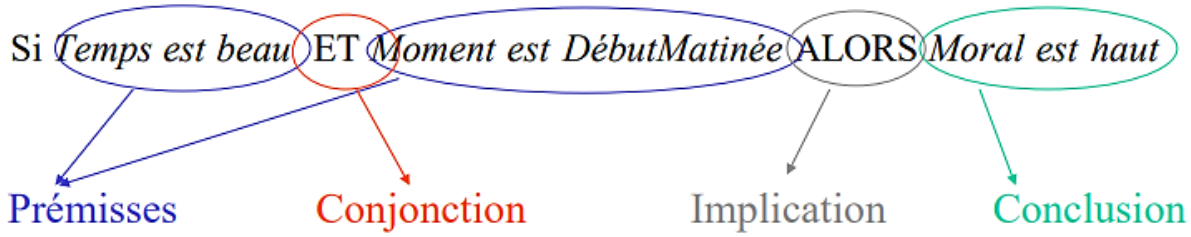


Figure 13 - Exemple d'inférence

13.Défuzzification :

La défuzzification consiste à caractériser les variables linguistiques utilisées dans le système. Il s'agit donc d'une transformation des entrées réelles en une partie floue définie sur un espace de représentation lié à l'entrée. Cet espace de représentation est normalement un sous-ensemble flou. Durant l'étape de la fuzzification, chaque variable d'entrée et de sortie est associée à des sous-ensembles flous.

Comme pour tous les opérateurs flous, le concepteur du système flou doit choisir parmi plusieurs définitions possibles de défuzzification. Une liste détaillée peut être consultée dans [Leekwijck and Kerre, 1999].

Nous allons présenter brièvement les deux principales m´méthodes de défuzzification : la méthode moyenne des maxima (MM) et la méthode du centre de gravité (COG).

La défuzzification MM définit la sortie (décision du montant du pourboire) comme :

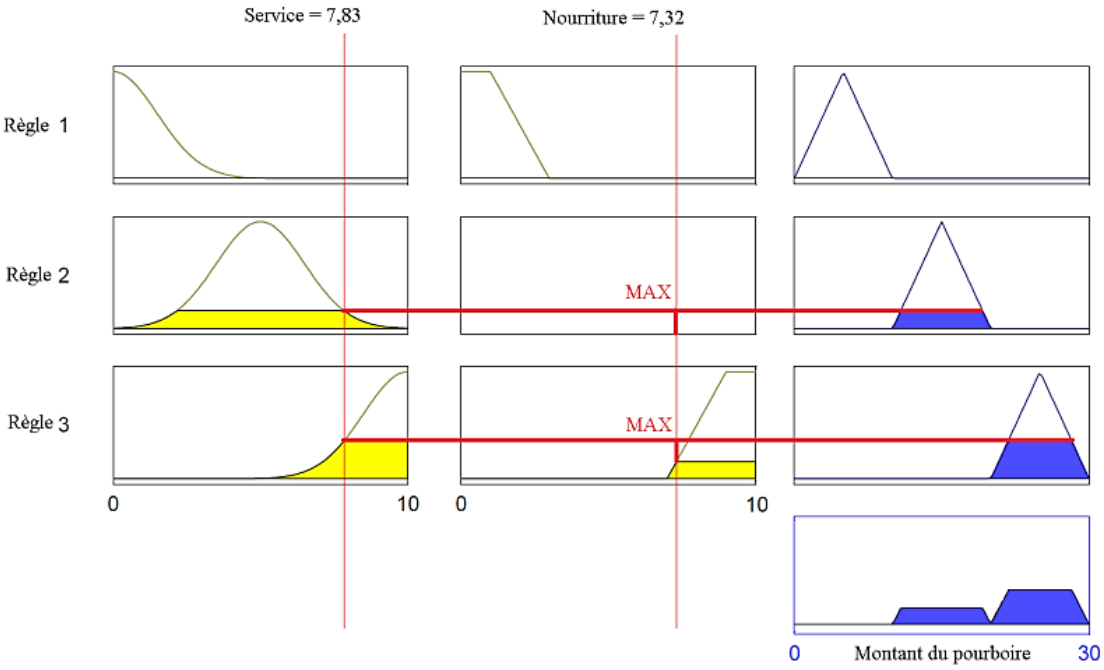


Figure 14 - Exemple d'implication floue en utilisant la matrice des décisions

Etant la moyenne des abscisses des maxima de l'ensemble flou issu de l'agrégation des conclusions.

Décision = $\frac{\int_S y.dy}{\int_S dy}$

Où $S = \{y_m \in R, \mu(y_m) = SUP_{y \in R}(\mu(y))\}$

Et R est l'ensemble flou issu de l'agrégation des conclusions.

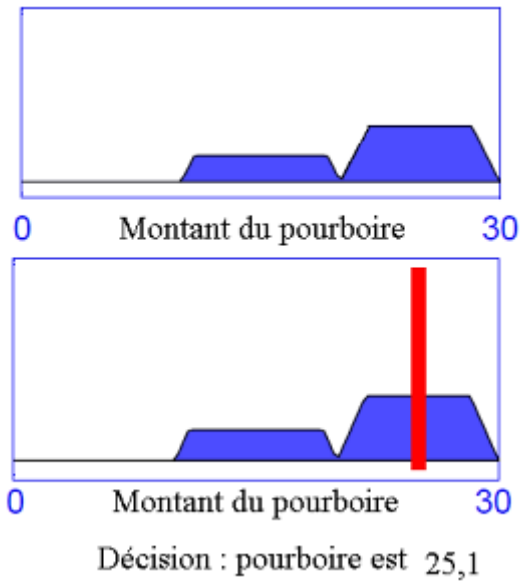


Figure 15 - Défuzzification avec la méthode moyenne des maxima (MM)

La défuzzification COG est plus couramment utilisée. Elle définit la sortie comme correspondant à l'abscisse du centre de gravité de la surface de la fonction d'appartenance caractérisant l'ensemble flou issu de l'agrégation des conclusions.

$$\text{Décision} = \frac{\int_S y.\mu(u).dy}{\int_S \mu(u).dy}$$

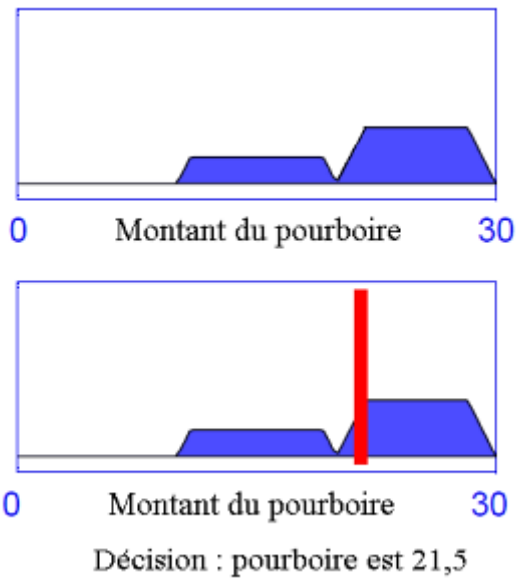


Figure 16 - Défuzzification avec la méthode centre de gravité (COG)

Cette définition permet d'éviter les discontinuités qui pouvaient apparaître dans la défuzzification MM, mais est plus complexe et demande des calculs plus importants.

Certains travaux tel [Madau D., 1996] cherchent à améliorer les performances en cherchant d'autres méthodes aussi efficaces mais avec une complexité algorithmique moindre. Comme nous le voyons sur les 2 figures montrant les méthodes de défuzzification MM et COG appliquées à notre exemple, le choix de cette méthode a un effet important sur la décision finale.

14.Réseau de neurones :

Cette méthode établit un système de connexions neurales/synaptiques qui permet d'apprendre en modulant le poids des différentes connexions. Elle repose sur les connexions que nous trouvons dans le cerveau humain. Nous avons un plan d'un neurone artificiel en bas. Il faut un certain nombre et applique à chacun un certain poids, puis donne une issue.

Au sein d'un réseau, les poids des différents neurones seront également modulés pour donner une grande structure. Cela permettra de combiner des neurones dans certaines formes.

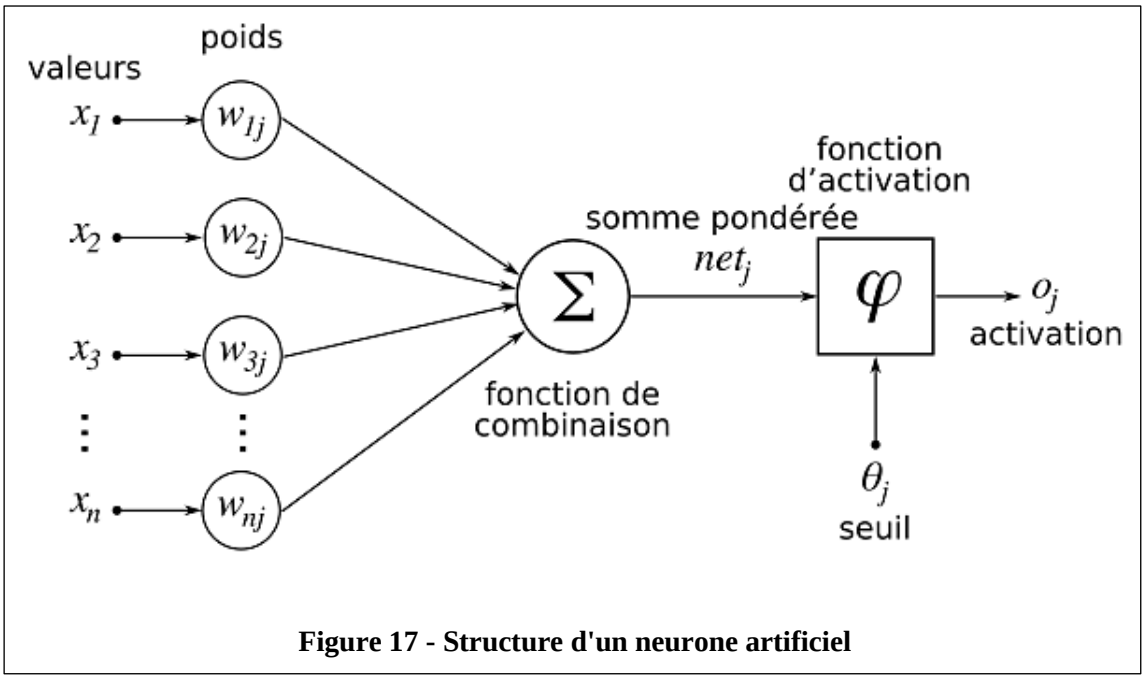


Figure 17 - Structure d'un neurone artificiel

La figure ci-dessous donne un exemple de forme de réseau de neurones. Celle-ci combine 4 couches de neurones. Dans ce cas précis une couche prend ses entrées de la couche précédente et donne ses sorties à la couche suivante.

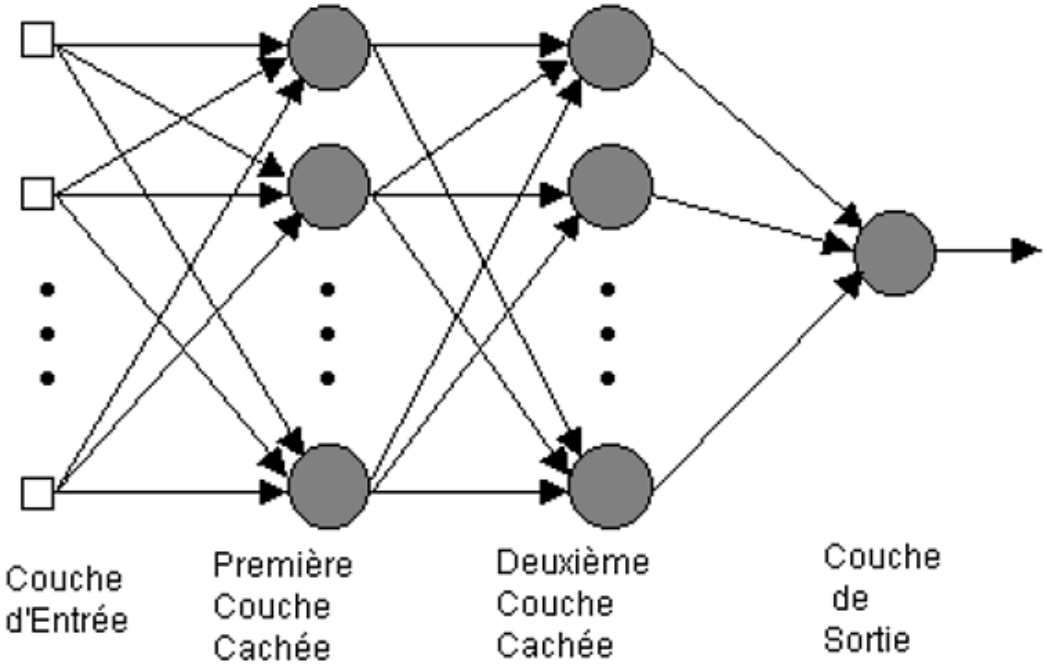


Figure 18 - Réseau de neurones multicouche (Zenou)

Le réseau tirera alors des leçons des exemples que nous fournirons. Il fonctionnera selon un algorithme à partir duquel il minimisera l'erreur qu'il fait en ce qui concerne la sortie que nous lui imposons. L'un des algorithmes les plus connus est la propagation inverse des dégradés. Il s'agit d'examiner la différence entre la valeur cible prévue par le réseau et la valeur donnée par le réseau après l'apprentissage. Alors le réseau va modifier le poids des neurones comme une fonction du gradient de cette différence afin de toujours diminuer l'erreur. Les neurones les plus impliqués dans l'erreur recevront un poids plus faible tandis que les autres recevront un poids plus élevé.

15.Avantages et inconvénients de la méthode :

- L’apprentissage par réseau de neurone est lent pour l’entrainement mais donne un taux de réponse en temps réel très rapide.
- Le taux d'erreur est généralement supérieur à la méthode SVM (Support vector machines). Malgré toutes les comparaisons ont été faites et ont déjà prouvé que parfois la méthode des réseaux de neurones avait un taux d'erreur inférieur au SVM. L'exemple qui suit compare les deux méthodes de distraction au volant (Tango F., Botta, Minin et Montanari, 2010).

- Le lien entre les entrées et les sorties est une boite noire dans laquelle il n'est pas possible d'avoir une connaissance sur notre système. C'est le plus gros inconvénient du réseau de neurones.

16.Conclusion :

Au cours des définitions, nous avons vu que le concepteur d'un système flou doit faire un nombre de choix important. Ces choix se basent essentiellement sur les conseils de l'expert ou sur l'analyse statistique des données passées, en particulier pour définir les fonctions d'appartenance et la matrice des décisions.

Partie II :

1. Introduction :

La technique de logique floue offre la possibilité de fournir un modèle et un contrôle simplifiés pour diverses applications non techniques (Javaherdashti, Nwaoha et Ebenso, 2012).

Le caractère fondé sur des règles des modèles flous permet pour une interprétation de modèle d'une manière similaire à celle que les humains utilisent pour décrire la réalité (Edwin et Kumanan, 2007).

Les méthodes conventionnelles de validation statistique fondées sur des données numériques peuvent être complétées par l'expertise humaine qui implique souvent une connaissance et une intuition heuristiques (Zalnezhad, Ahmed et Hamdi, 2013).

Le système d'inférence neuro-floue adaptative (ANFIS) est une technique avancée de systèmes à base floue pour modélisation et simulation des systèmes complexes (Siva, Murugan et Raghupathy, 2009).

Cette technique fournit une méthode de modélisation floue pour apprendre des informations sur un ensemble de données, afin de calculer les paramètres de la fonction d'adhésion qui permettent le mieux au système d'inférence floue associé de suivre les données d'entrée / sortie données.

2. Elaboration de modèles :

2.1.Fonctions d'adhésion pour les variables d'entrée et de sortie :

Les paramètres d'entrée utilisés pour le soudage TIG, y compris le « courant », la « tension », le « débit gazeux » et le « diamètre de l'électrode », ainsi que leurs termes linguistiques, leurs fonctions d'adhésion et leur plage de répartition sur les univers de discours respectifs, sont décrits aux figures 2 à 5. Une fonction d'adhésion triangulaire a été utilisée pour tous les termes linguistiques dans toutes les variables.

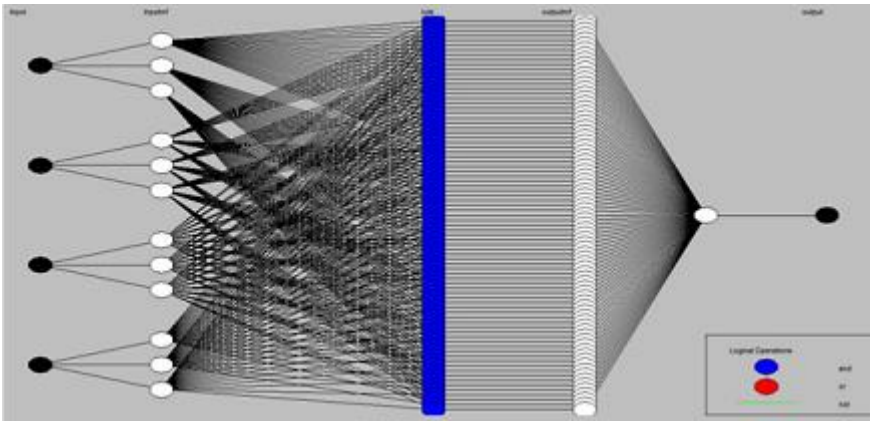


Figure 19 - ANFIS structure modèle TIG soudage

Dans la figure 2, la fonction d'adhésion indique que le débit de gaz est moyen.

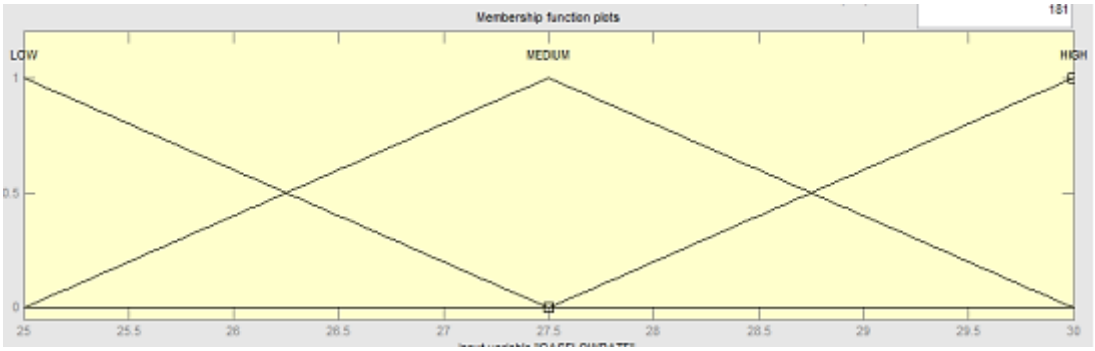


Figure 20 - Fonction d'adhésion pour la variable d'entrée du débit de gaz

Dans la figure 3, la fonction d'adhésion est courante comme moyenne.

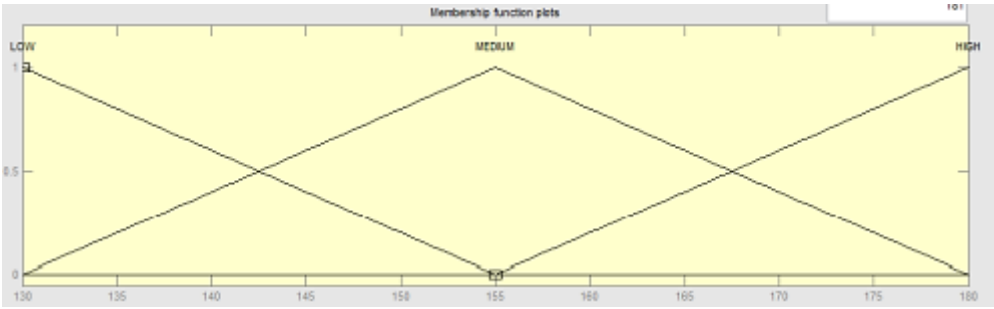


Figure 21 - Fonction d'adhésion pour la variable d'entrée actuelle

Dans la figure 4, la fonction d'adhésion indique que la tension est moyenne.

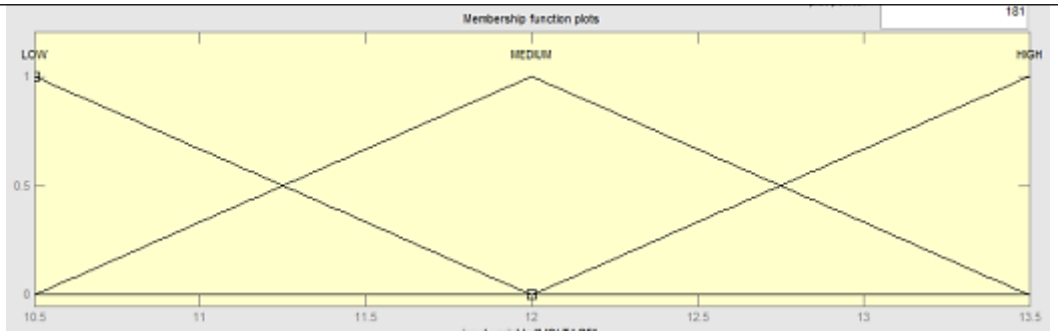


Figure 22 - Fonction d'adhésion pour variable d'entrée de tension

Dans la figure 5, la fonction d'adhésion indique que le diamètre de l'électrode est moyen.

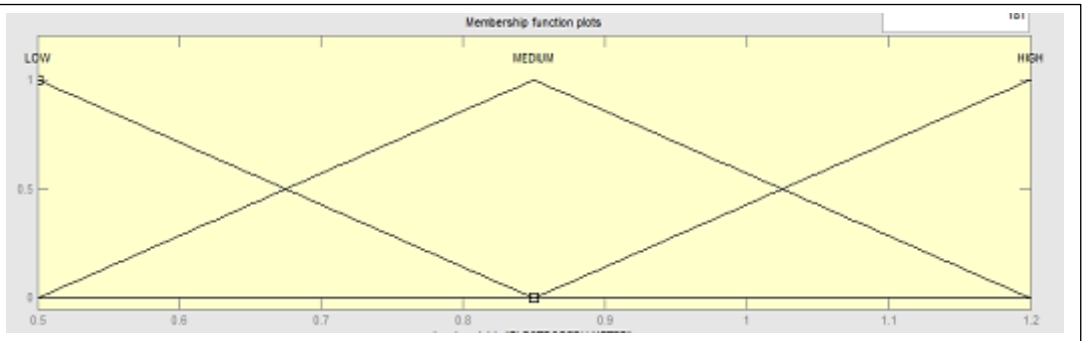


Figure 23 - Fonction d'adhésion pour la variable d'entrée de diamètre d'électrode

3. Règles du système de logique floue et surfaces de contrôle :

La modélisation floue du soudage TIG de tuyaux en acier doux utilise quatre paramètres d'entrée et deux paramètres de sortie modélisés les uns après les autres, dont chacun correspond à certaines variables linguistiques.

Les règles étaient les mêmes pour la modélisation de la résistance à la traction et de la limite d'élasticité ; lorsque la modélisation de la limite d'élasticité « Résistance à la traction » a été remplacée par « Limite d'élasticité ».

4. Résultats expérimentaux :

4.1.Processus TIG (avec MATLAB) :

Dans cette étude, le modèle ANFIS a été développé sur la base de 30 expériences de paramètres de processus TIG.

Un modèle MATLAB a été développé pour prédire la qualité de la soudure TIG en termes de paramètres structurels, y compris la résistance à la traction et la limite d'élasticité.

Les valeurs mesurées et prévues des caractéristiques de la soudure TIG sont présentées dans les tableaux 1 et 2.

Tableau 2 - Résistance à la traction prévue et expérimentale

S/N	Résistance à la traction expérimentale MPa	Résistance à la traction prévue MPa
1	462.05	462
2	438.94	439
3	343.78	344
4	404.29	433
5	462.05	462
6	462.05	462
7	346.55	347
8	508.25	508
9	462.05	462
10	462.05	467
11	508.25	508
12	485.15	485
13	485.15	485
14	462.05	462
15	462.05	462
16	462.05	462
17	485.15	485
18	485.15	485
19	415.84	439
20	415.84	416
21	462.05	462
22	462.05	462
23	462.05	462
24	462.05	462
25	462.05	439
26	462.05	439
27	462.05	433

28	438.94	439
29	438.94	439
30	415.84	439

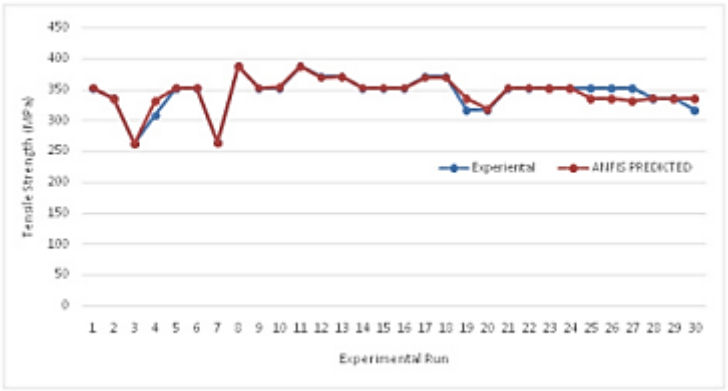


Figure 24 - Résistance à la traction prévue et expérimentale pour le modèle ANFIS

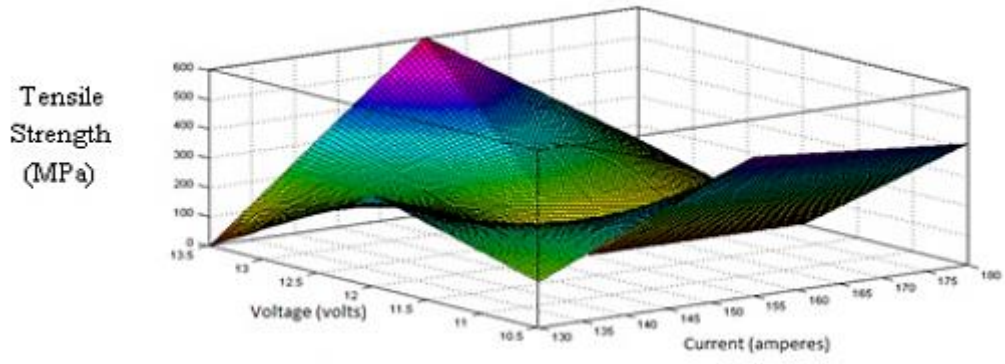


Figure 25 - Tracé 3D de la résistance à la traction montrant l'interaction de la tension et du courant

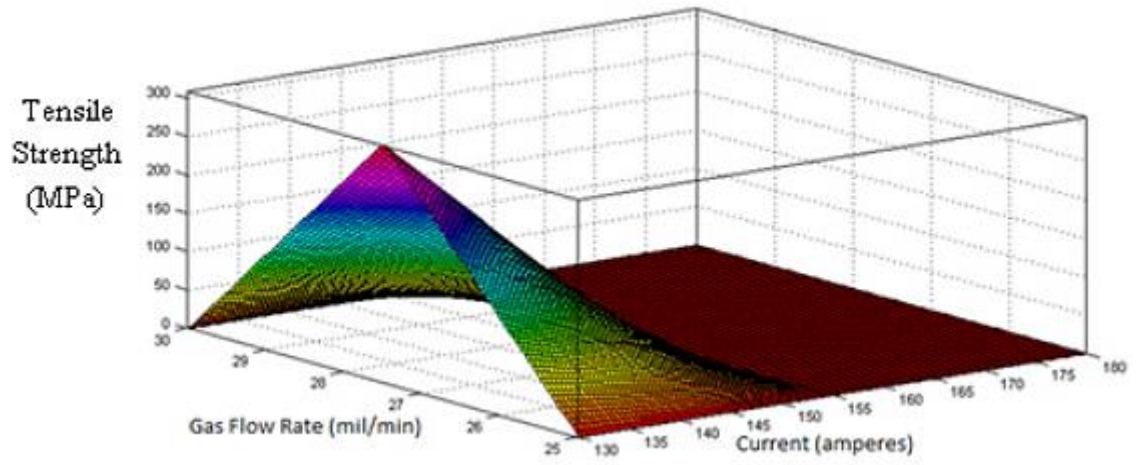


Figure 26 - Tracé de surface 3D de la résistance à la traction montrant l'interaction du courant et du débit de gaz

Tableau 3 - Limite d'élasticité prévue et expérimentale

S/N	Limite d'élasticité expérimentale MPa	Limite d'élasticité prédite MPa
1	352.71	353
2	335.07	335
3	262.43	262
4	308.62	331
5	352.71	353
6	352.71	353
7	264.53	263
8	387.98	388
9	352.71	353
10	352.71	354
11	387.98	388
12	370.34	370
13	370.34	371
14	352.71	353
15	352.71	353
16	352.71	353
17	370.34	370

18	370.34	370
19	317.44	335
20	317.44	318
21	352.71	353
22	352.71	353
23	352.71	352
24	352.71	352
25	352.71	335
26	352.71	335
27	352.71	331
28	335.07	336
29	335.07	335
30	317.44	335

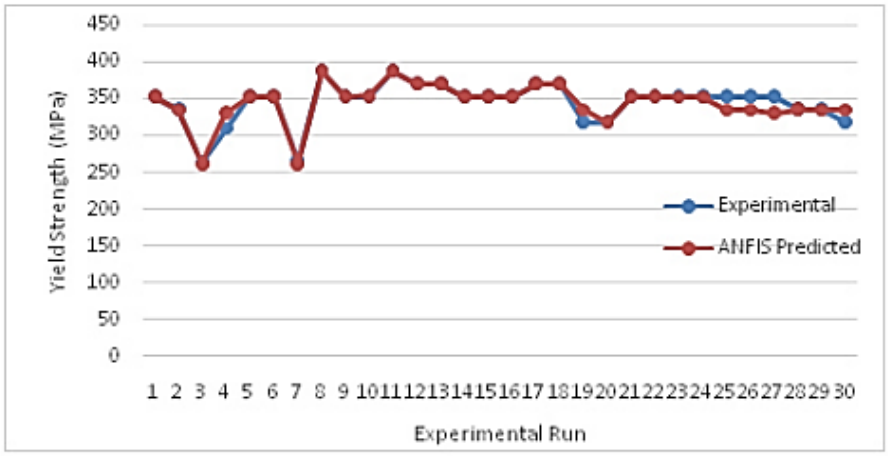


Figure 27 - Limite d'élasticité prévue vs la limite expérimentale

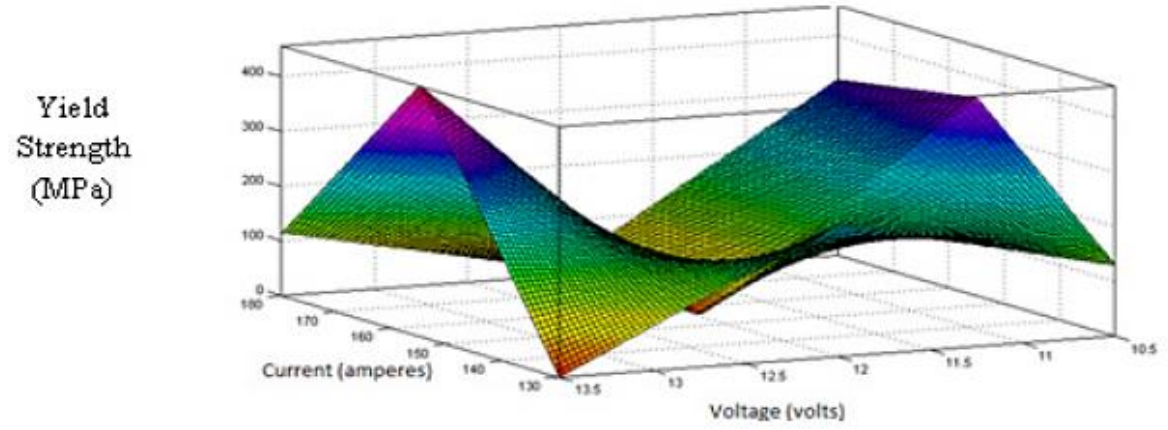


Figure 28 - Tracé de surface 3D de la limite d'élasticité montrant l'interaction de la tension et du courant

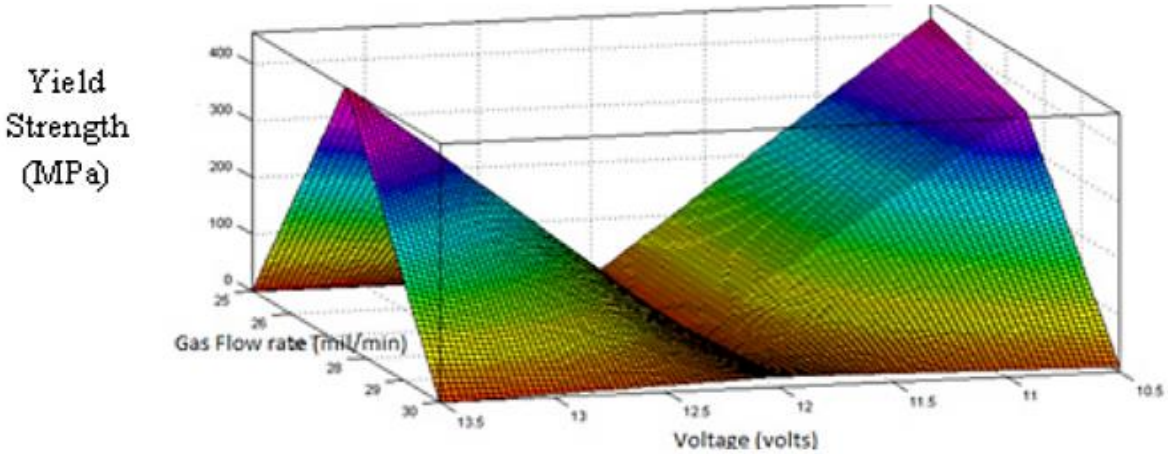


Figure 29 - Tracé de surface 3D de la limite d'élasticité montrant l'interaction de la tension et du débit de gaz

4.2.Programme WSN :

Ces résultats sont de programme du WSN :

Le programme du WSN est développé sur un logiciel libre appelé Code : Blocks.

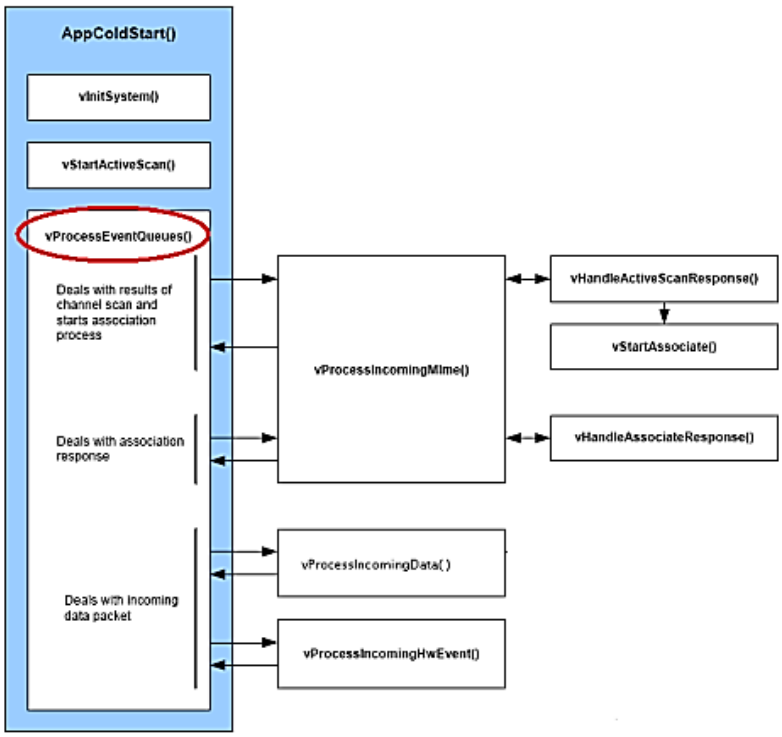


Figure 30 - Schéma de l'architecture logicielle du coordinateur de WSN

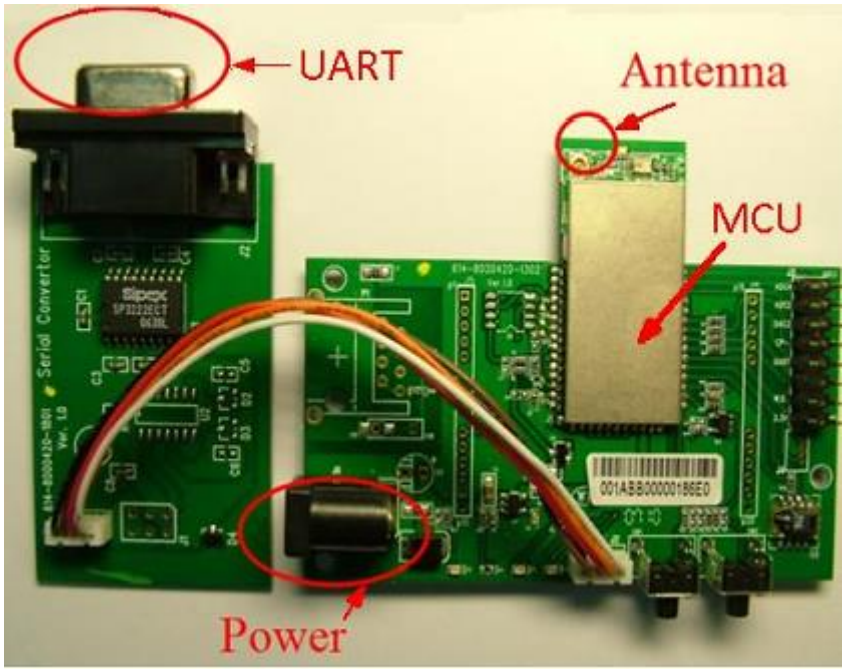


Figure 31 - Conseils de développement de WSN

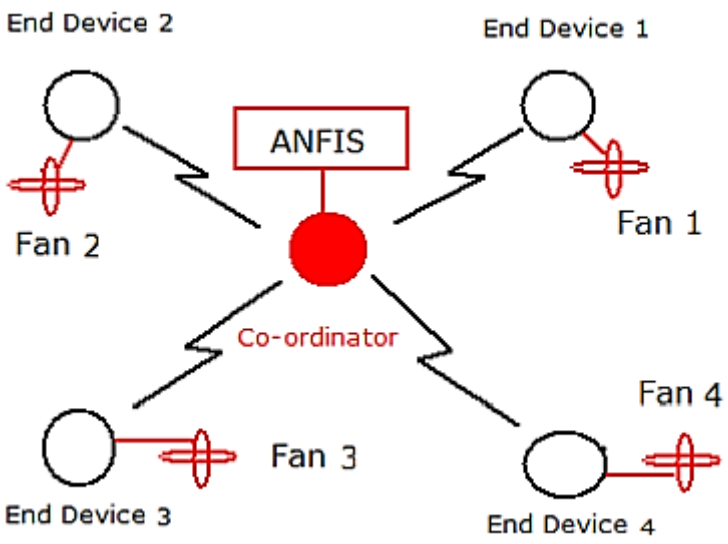


Figure 32 - Schéma conceptuel du WSN basé sur l'ANFIS pour le contrôle du climatiseur

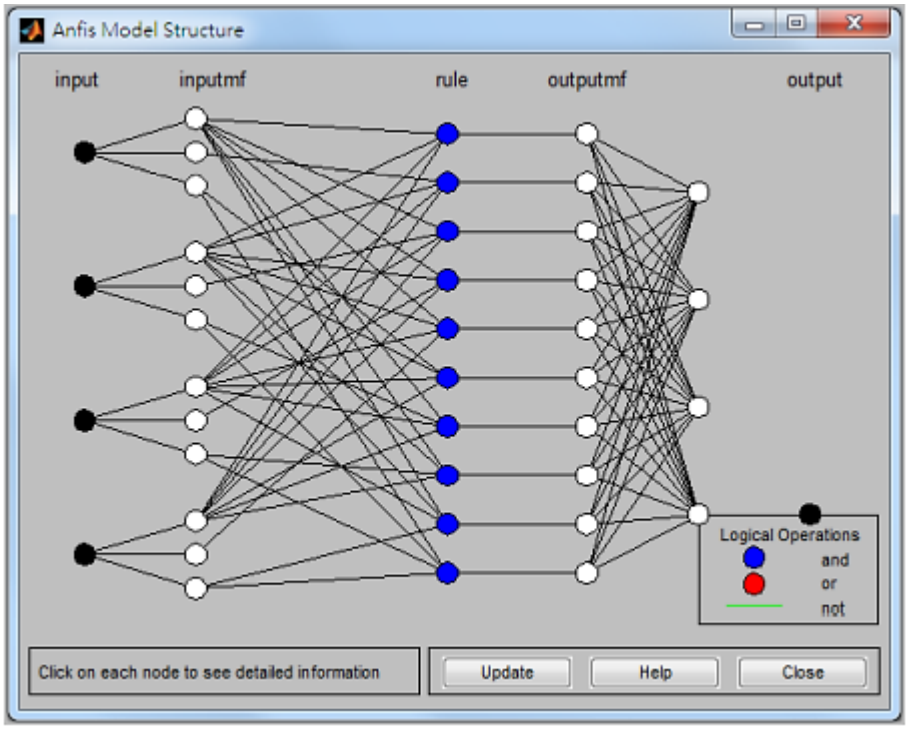


Figure 33 - ANFIS structure diagram

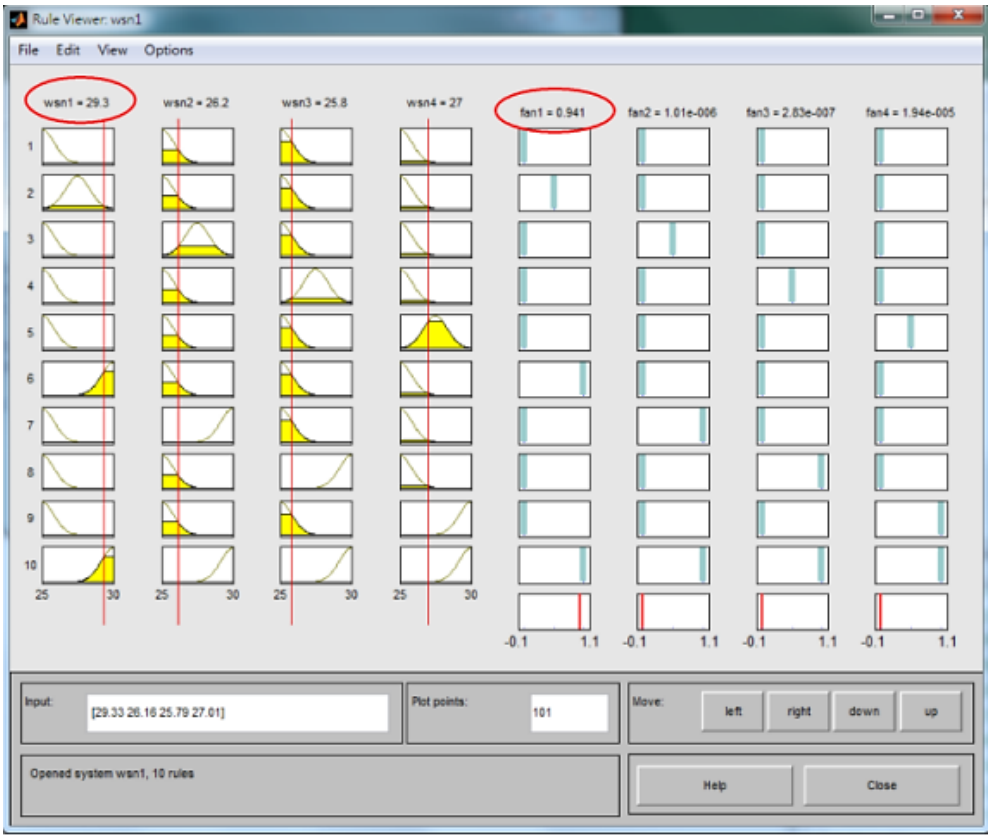


Figure 34 - Diagramme des résultats d'inférence ANFIS

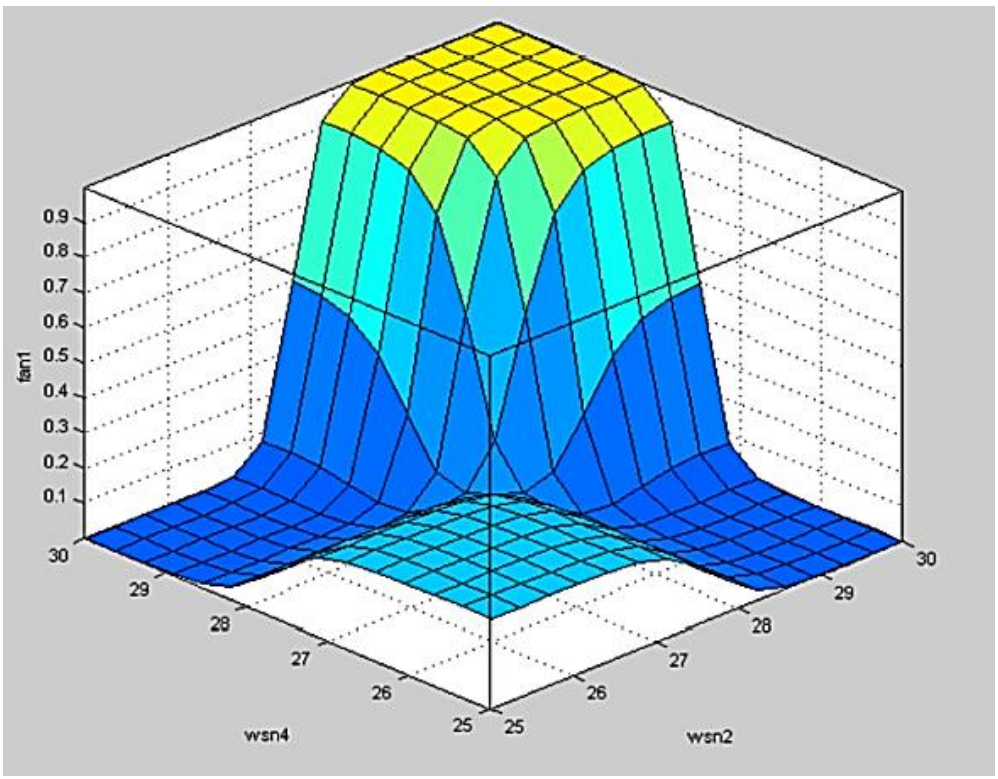


Figure 35 – Exemple de diagrammes de surface d'inférence de l'ANFIS

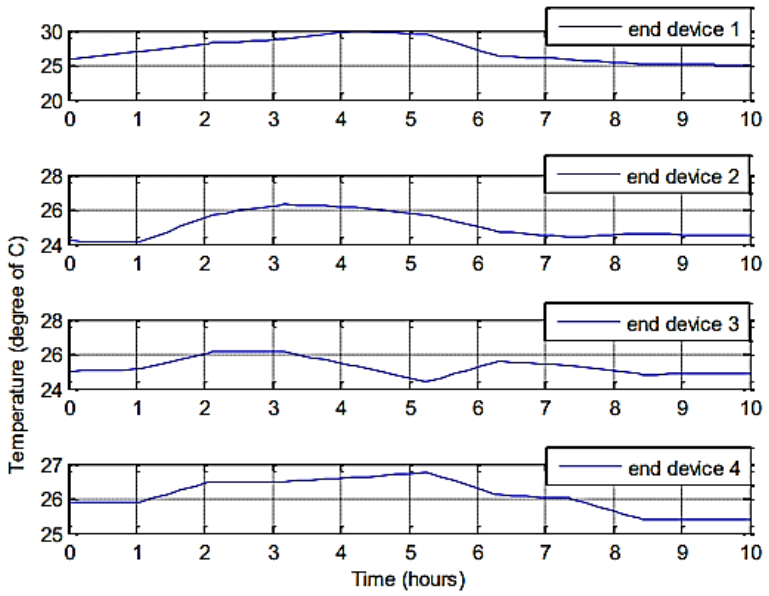


Figure 36 - Diagramme des températures mesurées des quatre coins intérieurs

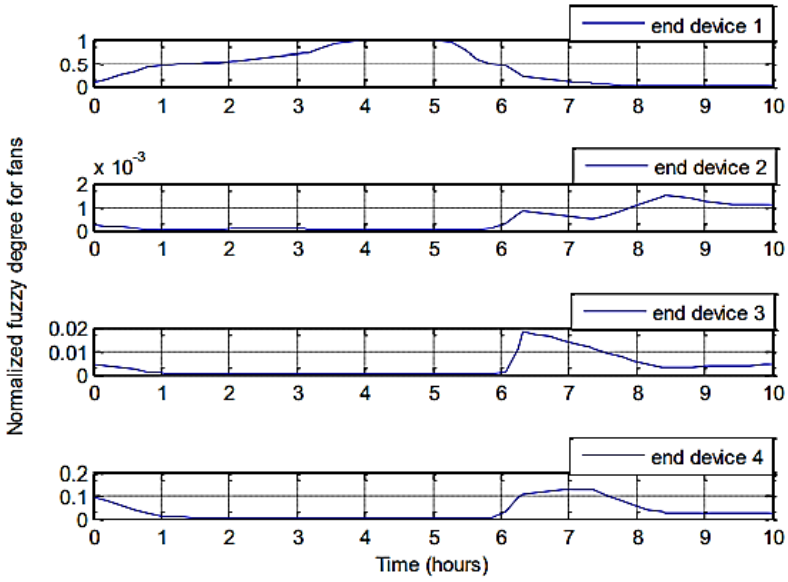


Figure 37 - Diagramme des résultats de simulation des quatre coins intérieurs

La méthode de conception pour le contrôle d'un climatiseur intérieur en utilisant le WSN basé sur l'ANFIS est proposée. Les vérifications physiques et les simulations sont également démontrées avec succès.

Chapitre 2 :

L'étude algorithmique et l'optimisation de la méthode ANFIS

Chapitre 2 : L'étude algorithmique et l'optimisation de la méthode ANFIS

Partie I :

1. Introduction :

Le système Neuro-Flous combine les avantages de deux technologies complémentaires. Les systèmes flous fournissent une bonne représentation des connaissances.

L'intégration de réseaux de neurones dans ces systèmes améliore les performances grâce à la capacité des réseaux de neurones à apprendre. A l'inverse, l'injection de règles floues du réseau de neurones clarifier la sémantique des paramètres réseau et faciliter leur initialisation. Cela permet d'économiser beaucoup de temps de calcul pour l'identification.

Bien que de nombreux types de systèmes neuro fuzzy aient été définis et développés ces dernières années, ils sont parfois non standardisés, vagues et déroutants.

Pour plus de clarté dans la définition, ce chapitre propose une brève description de plusieurs types de systèmes neuro fuzzy et une description plus détaillée de l'ANFIS.

2. Quelques types de combinaison Neuro-Floues :

Il existe de nombreuses façons de combiner les réseaux de neurones et les systèmes flous. Ces types peuvent être classés fonctionnellement et structurellement selon l'architecture et la structure.

Recherche sur les systèmes d'inférence floue et construction de réseaux de neurones.

2.1. Systèmes neuro-flou coopératifs et concourants :

Un système Neuro-flou coopératif peut être considéré comme préprocesseur où le mécanisme d'apprentissage de réseaux de neurones artificiels (RNA) détermine les fonctions d'appartenance de Système d'inférence flou (SIF) ou les règles floues à partir données d'apprentissage. Une fois que les paramètres de SIF sont déterminés, RNA va au fond. La règle basée est habituellement déterminée par un algorithme clustering flou. Les fonctions d'appartenance sont habituellement approximées à partir RNA par les données d'apprentissage. Dans un système Neuro-flou concourant, RNA aide le SIF continûment pour déterminer les paramètres exiger particulièrement si les variables d'entrée du contrôleur ne peuvent pas être

mesurées directement. Dans certains cas les sorties de SIF ne pourraient pas être directement applicables au processus. Les figures ci-dessous représentent les modèles Neuro-Flou coopératifs et concourants.

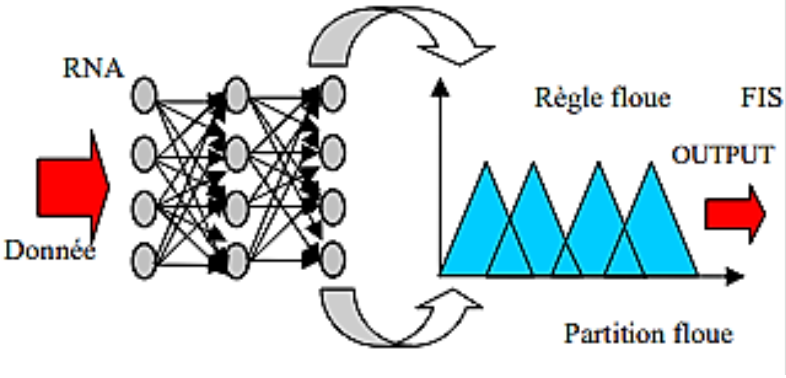


Figure 38 - Système neuro-flou Coopératif

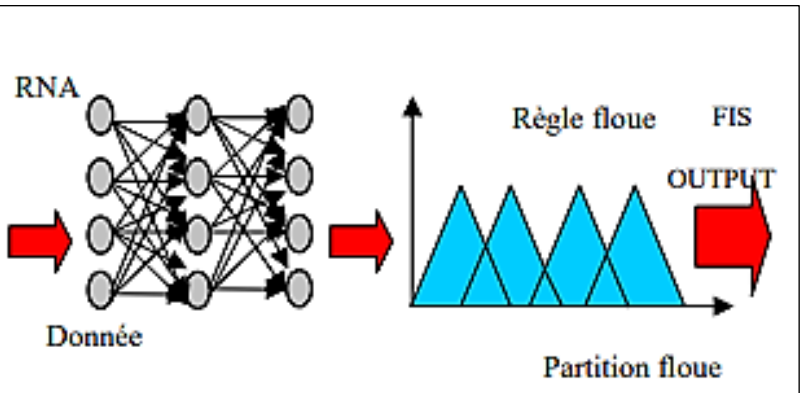


Figure 39 - Système neuro-flou concurrent

2.2. Les systèmes neuro- flou fondus :

Dans une architecture Neuro-Flou fondue, les RNA sont utilisées pour déterminer les paramètres de SIF. Les systèmes Neuro-Flou fondus partagent les structures de données et la représentation de connaissance. Une manière habituelle d'appliquer un algorithme d'apprentissage à un système flou est à représentez-la dans une architecture spéciale.

De quelque manière que l'algorithme d'apprentissage de l'RNA ne peut pas être appliqués directement à un système d'inférence comme une fonction parce que les fonctions utilisées dans le système d'inférence sont habituellement non différentiables. Ce problème peut être abordé en employant des fonctions différentiables dans le système d'inférence ou près ne pas utiliser l'algorithme d'apprentissage de RNA standard. Certains types de systèmes Neuro-Flou

principaux GARIC, FAUCON, ANFIS, NEFCON AMUSEMENT, SONFIN, et beaucoup d'autres.

2.3. Falcon (Fuzzy Adaptive Learning Control Network):

FALCON à une architecture cinq couches, comme il est représenté dans la figure suivante. Il y a deux neurones pour chaque variable de sortie. Une pour les données d'apprentissage (sortie désirée) et l'autre est pour la sortie de FALCON.

La première couche cachée sert à fuzzifier les variables d'entrées. Chaque neurone dans cette couche représentant une fonction d'appartenance à un ensemble floue.

La deuxième couche cachée définit les parties antécédentes des règles floues suivie par les parties conséquences des règles dans la troisième couche cachée.

FALCON emploie un algorithme d'apprentissage hybride comportant l'apprentissage non supervisée pour localiser des fonctions d'appartenance et base des règles initiale et l'apprentissage supervisé pour optimiser l'ajustement des paramètres du MF pour générer les sorties désirées.

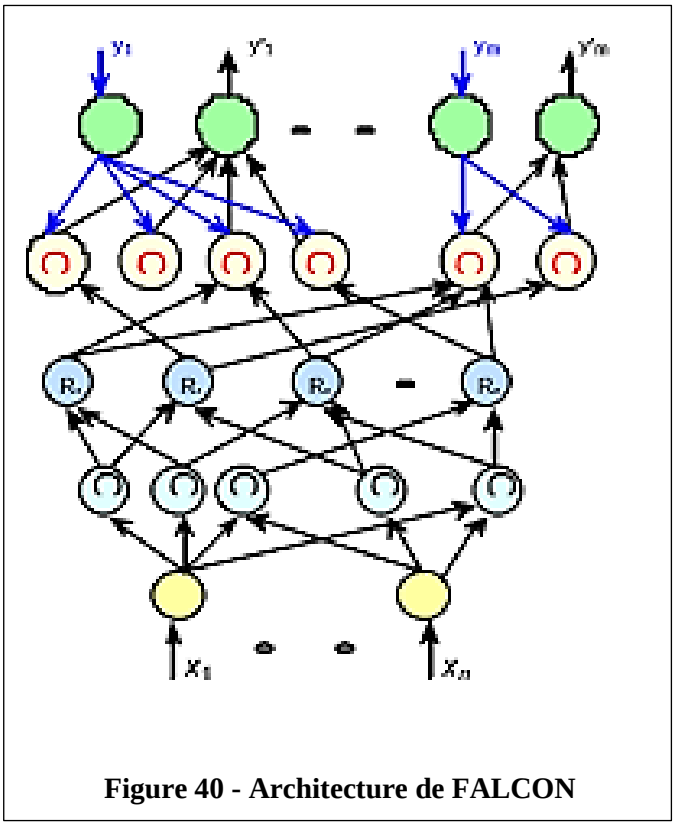


Figure 40 - Architecture de FALCON

2.4. Le NEFCON (NEuro-Fuzzy CONtrol) :

NEFCON est conçu pour mettre en application le système d'inférence flou type Mandani.

Il est consisté de 2 couches dont les poids sont les ensembles flous et les règles floues. Avec la même utilisation antécédente supposée ont partagé les poids, qui sont représentés par des ellipses dessinées autour des raccordements. Elles assurent l'intégrité de la base de règle. La couche d'entrée assure la tâche de l'interface de fuzzification, la logique d'inférence est représentée par les fonctions de propagation, et la couche de sortie est l'interface de défuzzification.

L'apprentissage du modèle de NEFCON est basé sur un mélange de l'apprentissage non supervisé et supervisée (rétropropagation). NEFCON peut être employé pour apprendre des règles initiales, si aucune connaissance du système n'est disponible ou même pour optimiser une base manuellement définie de règle.

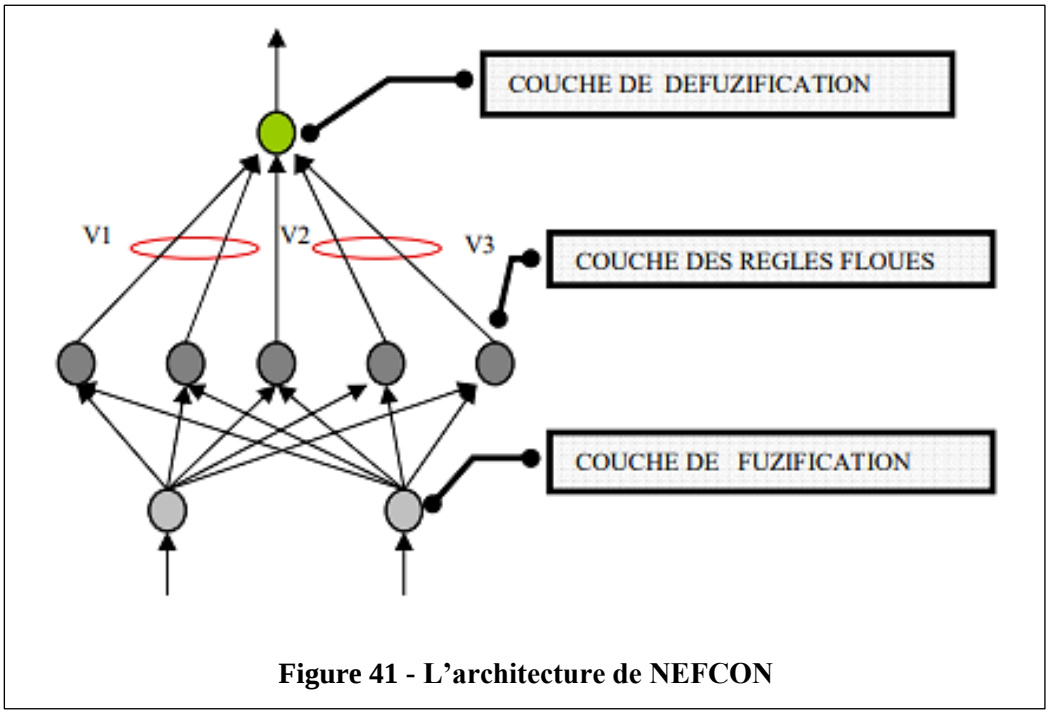


Figure 41 - L'architecture de NEFCON

3. Le modèle ANFIS :

3.1. Architecture de l'ANFIS :

ANFIS (Adaptive Network Based Fuzzy Inference System) est un système d'inférence neuro-floue adaptatif qui consiste à utiliser un réseau de neurones de type MLP à cinq couches, chaque couche correspondant à une réalisation d'une étape du système.

Raisonnement flou de type Takagi-Sugano. Pour simplifier, supposons que le système d'inférence floue a deux entrées, x et y, et f comme sortie. Supposons que votre base de règles contienne deux règles floues de type Takagi-Sugano.

Règle l :
SI x est Al et y est B1 ALORS f1 = p1 x+q1 y +r1
Règle 2 :
SI x est A2 et y est B2 ALORS f2= p2 x+q2 y +r2

L'ANFIS à une architecture posée par cinq couches comme représenté sur la figure :

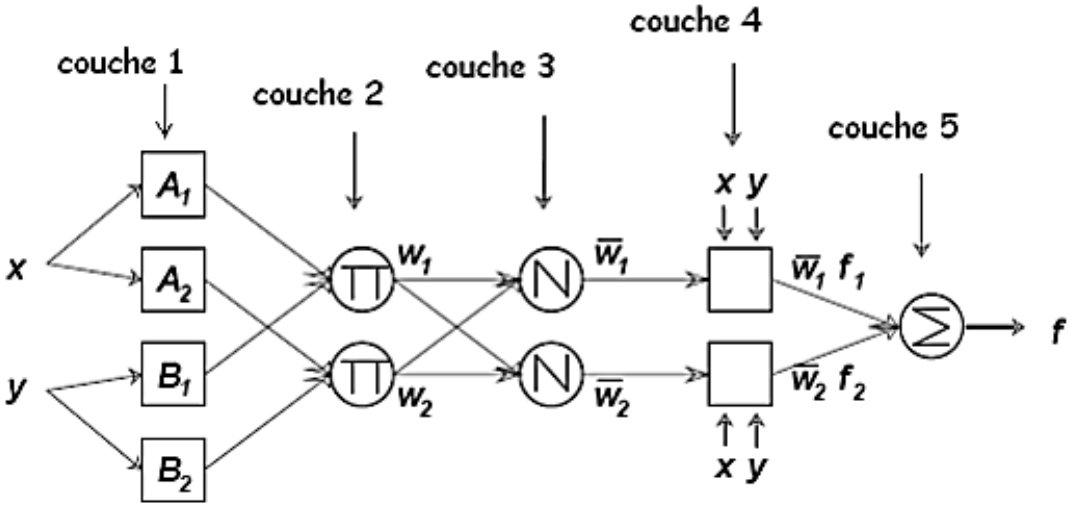


Figure 42 - L'Architecture de l'ANFIS

Une architecture classique peut être décrite de la manière suivante :

1. **La première couche :**

La première couche d'une architecture de type ANFIS comporte autant de neurones qu'il y'a de sous-ensembles flous dans le système d'inférence représenté.

Chaque neurone calcule le degré de vérité d'un sous ensemble flou particulier par sa fonction de transfert. La seule restriction sur le choix de cette fonction concerne sa dérivabilité.

En retrouve dans la littérature, l'utilisation, de fonctions gaussiennes et les paramètres modifiables sont le centre et la pente de la gaussienne (variance).

La fonction d'activation des neurones i de la première couche :

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

Tel que x est l'entrée au neurone i, et Ai est un sous ensemble floue correspondant au variable x.

En d'autres termes, f_i^1 est la fonction d'appartenance du Ai et il indique le degré auquel donné x satisfait le quantifier Ai.

Nous choisissons $\mu_{Ai}(x)$ pour être en forme de (Gaussien, triangle, trapézoïdal) avec le maximum égal à 1 et le minimum égal à 0, tel que les fonctions généralisées de ces formes sont :

Triangle:

$$\boldsymbol{\mu(x) = max\left[min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right]} \quad (2)$$

Trapézoïdale:

$$\mu(x) = max\left[min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right] \quad (3)$$

Gaussienne:

$$\boldsymbol{\mu(x) = exp\left(-\frac{(x-c)^2}{\sigma^2}\right)} \quad (4)$$

{a, b, c, σ} est l'ensemble des paramètres. Pendant que les valeurs de ces paramètres changent, les fonctions en forme précédente changent en conséquence, de ce fait présenter de diverses formes de fonction d'appartenance sur la variable linguistique Ai.

Les paramètres dans cette couche désigner sous le nom des paramètres de fonction d'appartenance.

2. La deuxième couche :

La deuxième couche cachée sert à calculer le degré d'activation des prémisses. Les neurones de là ces couches représentent chacun la prémisse d'une règle.

Ils reçoivent en entrée le degré de vérité des différents sous-ensembles flous composant cette prémisse et ont en charge le calcul de son propre degré de vérité.

Les fonctions d'activation utilisées pour ces neurones dépendant des opérateurs présents dans les règles (ET ou OU).

La fonction d'activation des neurones i de la première couche :

$$W_k = \mu_{Ai}(x) \times \mu_{Bj}(y) \quad (5)$$

Ou k : représente le nombre de règle,

i : représente le nombre de partition de x,

j : le nombre de partition de y.

3. La troisième couche :

La troisième couche cachée normalise de degré d'activation des règles. Chaque neurone dans cette couche est un neurone de cercle noté N.

Le i^{emme} neurone calcule le rapport entre i^{emme} poids de règles et la somme de tous les poids des règles. Cette opération est appelée la normalisation des poids.

$$\overline{W_k} = \frac{W_k}{\sum W_i} \quad (\mathbf{6})$$

L'ensemble des sorties de cette couche seront appelées les poids normalisés.

4. La quatrième couche :

La quatrième couche cachée sert à déterminer les paramètres la partie conséquence des règles (p, q, r).

La fonction de chaque neurone dans cette couche est la suivante :

$$f_i^4 = \overline{W_k} \times (p_k x + q_k y + r_k) \quad (7)$$

Où $\overline{W_k}$ est la sortie de la troisième couche, et {p_i, q_i et r_i} est l'ensemble des paramètres.

Ces paramètres désigner sous le nom les paramètres conséquents.

5. La couche de sortie :

La couche de sortie contient un seul neurone dans cette couche, est un neurone de cercle noté S qui calcule la sortie globale comme addition de tous les signaux entrants, c'est-à-dire :

$$f^5 = \sum_k \overline{W_k} \times f_k^4 \quad (8)$$

La figure qui suit représente un système ANFIS, à 2 entrées chaque entrée repartie en trois sous ensemble floue et 9 règles.

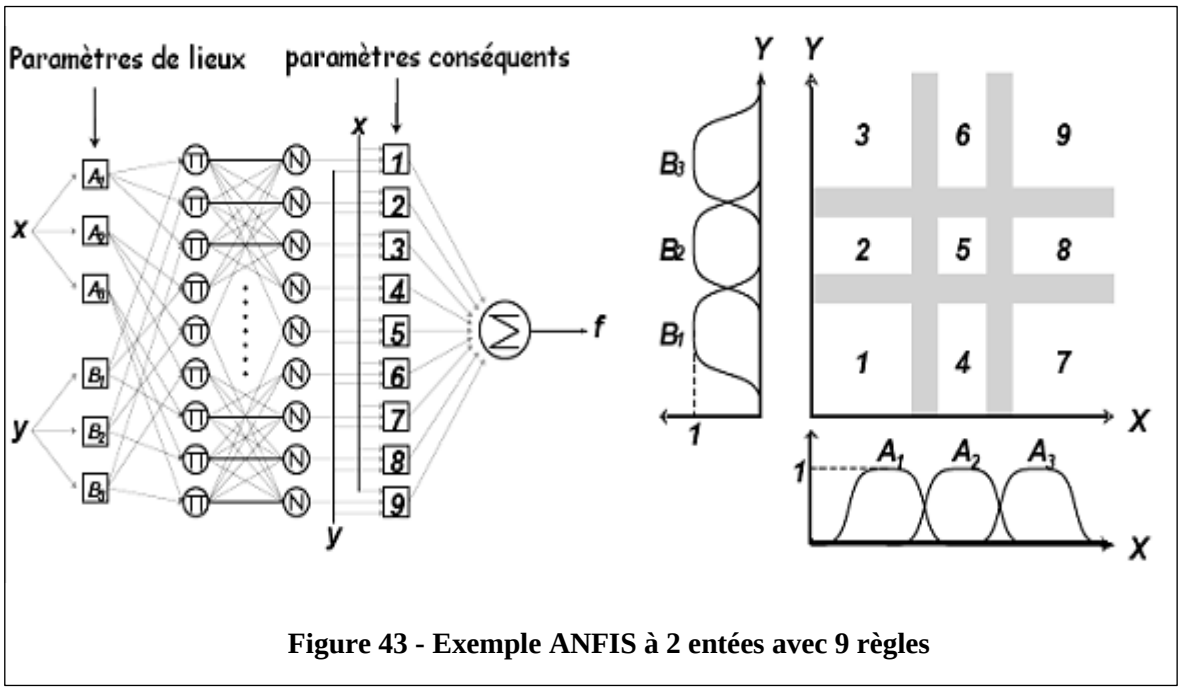


Figure 43 - Exemple ANFIS à 2 entées avec 9 règles

Tableau 1 - Les différentes couches d'un système ANFIS

Les différentes couches	Types des couches	Le nombre de neurone dans la couche
Couche 0	Les entrées	n
Couche 1	Les valeurs	(p.n)
Couche 2	Les règles	p^n
Couche 3	La normalisation	p^n
Couche 4	Linéarisation des fonctions	p^n
Couche 5	Somme	1

Tel que :

n : le nombre des entrées.

p : le nombre des sous-ensembles flous d'entrée (partition flou).

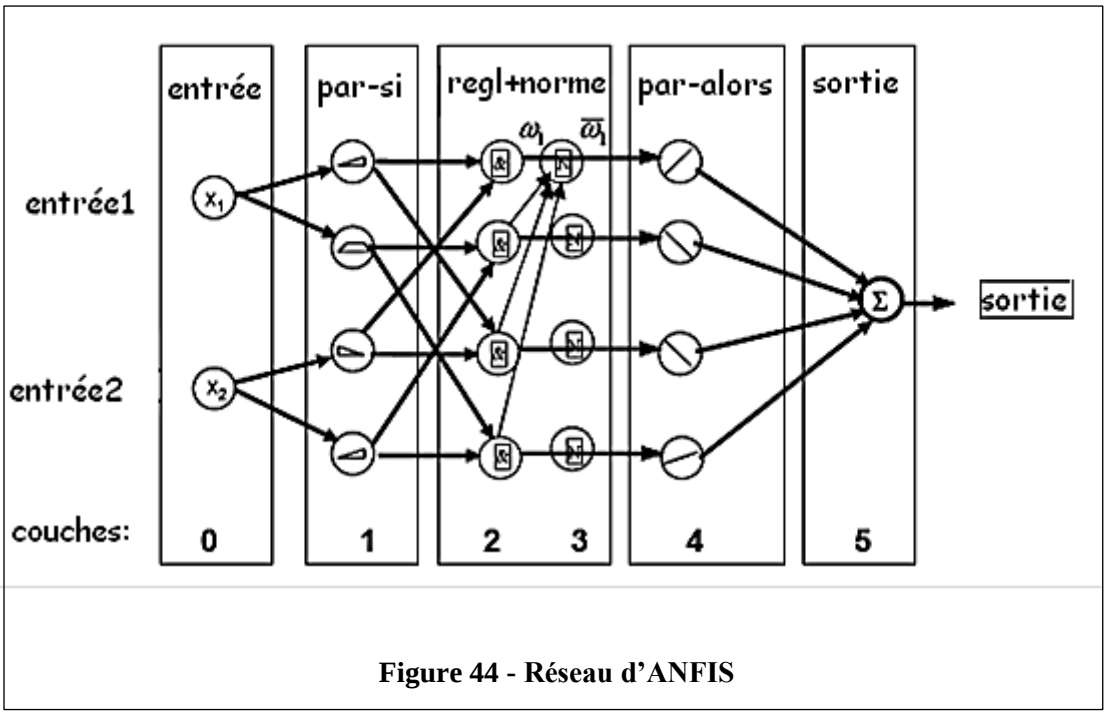


Figure 44 - Réseau d'ANFIS

Noter que les neurones dans ANFIS ont différentes structures :

- Valeurs [fonction d'appartenance définie par différentes formes].
- Règles [habituellement produit].
- Normalisation [division de somme et d'arithmétique].
- Fonctions [régressions linéaires et multiplication avec w, tel que west la normalisation du poids w].
- La sortie [Somme Algébrique].

3.2. Algorithme d'apprentissage de l'ANFIS :

Le système ANFIS applique le mécanisme d'apprentissage des réseaux neurone sur des techniques d'inférence floues.

Dans l'architecture ANFIS proposée dans la figure, la sortie globale peut être exprimé en tant que des combinaisons linéaires des paramètres conséquents.

Avec plus précision, la conclusion (la sortie) sur la figure peut être récrit comme :

$$f = \frac{w1}{w1+w2} f1 + \frac{w2}{w1+w2} f2 \qquad (9)$$

$$= (\overline{w_1}x)p_1 + (\overline{w_1}y)q_1 + (\overline{w_1})r_1 + (\overline{w_2}x)p_2 + (\overline{w_2}y)q_2 + (\overline{w_2})r_2 \qquad (10)$$

La sortie est une fonction linéaire des paramètres conséquences (p, q, r).

ANFIS est représentation paramétrique deux ensembles de paramètres : S1 et S2 tel que :

- S1 représente les paramètres des ensembles flous utilisés pour la fuzzification dans la première couche de système ANFIS.

$$S1 = \left\{\{a_{11}, b_{11}, c_{11}\}, \{a_{12}, b_{12}, c_{12}\}, \ldots., \{a_{1p}, b_{1p}, c_{1p}\}, \ldots, \{a_{np}, b_{np}, c_{np}\}\right\} \quad (11)$$

Où p est le nombre de partition floue de chacun des variables d'entrées et n est le nombre de variables d'entrées.

- S2 représente les coefficients des fonctions linéaires (les paramètres conséquentes).

$$S2 = p_1, p_2, p_3, \ldots, q_1, q_2, q_3, \ldots, r_1, r_2, r_3 \ldots \quad (12)$$

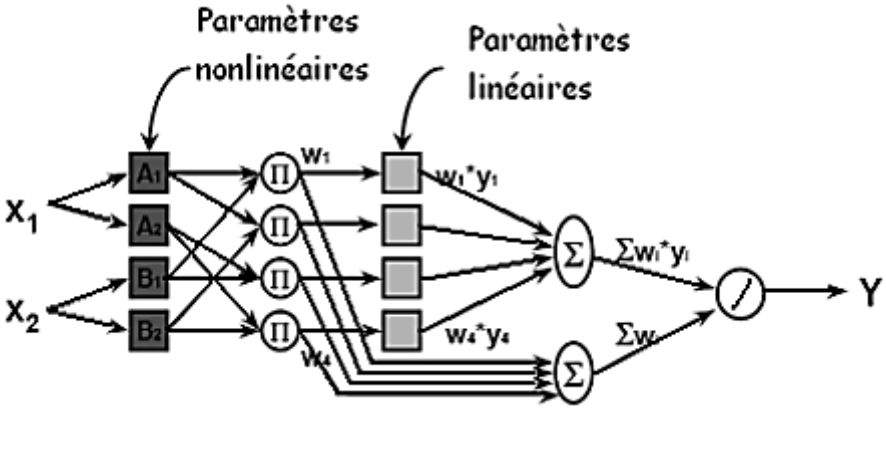


Figure 45 - Méthode d'apprentissage Hybride

ANFIS utilise un cycle d'apprentissage de deux passages :

- **Le passage en avant :** S1 est fixe et S2 est calculé en utilisant l'algorithme de moindre carré de l'erreur (LSE). (Le LSE est appliqué seulement une fois lorsque commencer à obtenir les valeurs initiales des paramètres conséquents)
- **Le passage en arrière :** S2 est fixe et S1 est calculé en utilisant l'algorithme de Rétro –Propagation.

Partie II :

1. Introduction :

L'ajustement est une opération d'optimisation portant sur la recherche du profil théorique qui colle le mieux possible aux données expérimentales.

Le processus d'ajustement doit permettre d'interpréter et de prévoir les variations de la variable dépendante en fonction de la variable expliquée, supposé connue.

De nombreuses méthodes existent pour ajuster les paramètres du modèle théorique afin de choisir le meilleur ajustement.

L'ANFIS est un système d'inférence floue (SIF) dont les paramètres des fonctions d'appartenances sont ajustés en utilisant l'algorithme d'apprentissage rétro propagation, ou en combinaison avec un autre type d'algorithmes comme le moindre carré.

2. Méthode des moindres carrés :

Une situation courante en sciences biologiques est d'avoir à sa disposition deux ensembles de données de taille n, {y1, y2,...,yn} et {x1, x2,...,xn}, obtenus expérimentalement ou mesurés sur une population.

Le problème de la régression consiste à rechercher une relation pouvant éventuellement exister entre les x et les y, par exemple de la forme y = f(x).

Lorsque la relation recherchée est affine, c'est-à-dire de la forme y = a*x + b, on parle de régression linéaire.

Mais même si une telle relation est effectivement présente, les données mesurées ne vérifient pas en général cette relation exactement.

Pour tenir compte dans le modèle mathématique des erreurs observées, on considère les données {y1, y2,...,yn} comme autant de réalisations d'une variable aléatoire Y et parfois aussi les données {x1, x2,...,xn} comme autant de réalisations d'une variable aléatoire X.

On dit que la variable Y est la variable dépendante ou variable expliquée et que la variable X est la variable explicative.

A. Méthode des moindres carrés ordinaires :

On considère une variable d'intérêt y appelée variable dépendante et un ensemble de K variables dites explicatives auquel on adjoint une constante. On dispose de N observations. On note y = (y1,...,yN) l'empilement des N observations de la variable dépendante.

On définit de même les vecteurs $x_1,...,x_K$ et x la matrice des variables explicatives à laquelle on adjoint le vecteur constant e = (1,..., 1)0 : x = ¡ e, $x_1,...,x_K$ ¢ est donc une matrice de dimension N × (K + 1).

Définition L'estimateur des moindres carrés ordinaires est défini comme le vecteur b de dimension K + 1, b = $(b_0,...,b_K)$ 0 , des coefficients de la combinaison linéaire de e, $x_1,...,x_K$ réalisant le minimum de la distance de y à l'espace vectoriel de RN engendré par e, x1,...,x_K, pour la norme euclidienne : $\hat{b}_{mco} = \arg\min \|y - xb\|^2$.

B. Méthode des moindres carrés contraints :

On peut souhaiter estimer un modèle économétrique linéaire en incorporant une information a priori sur les paramètres prenant la forme de contraintes linéaires. On peut aussi vouloir tester si certaines relations entre les paramètres sont bien acceptées par les données.

Les résultats obtenus au chapitre précédent ont montré comment tester des hypothèses très simples, s'écrivant sous la forme $H_0 : b_k = b^0_k$, où b^0_k est une valeur donnée.

On va examiner ici un cas un peu plus général dans lequel les hypothèses que l'on veut tester, ou bien les contraintes que l'on veut imposer font intervenir une ou plusieurs combinaisons linéaires des paramètres.

On va montrer obtenir un estimateur différent de celui des moindres carrés ordinaires, appelé estimateur des moindres carrés contraints (mcc) et on va montrer ses deux propriétés principales : l'estimateur des mcc est toujours plus précis que l'estimateur des mco ; l'estimateur des mcc est non biaisé seulement si la vraie valeur du paramètre satisfait les contraintes imposées.

Il y a donc un arbitrage entre robustesse et précision des estimateurs. Un tel arbitrage est très fréquent en économétrie. On va aussi introduire un test très utilisé permettant de tester des contraintes linéaires.

Ce test est connu sous le nom de test de Fisher, et on va voir comment le mettre en œuvre simplement à partir de deux régressions, l'une par les mcc et 'autre par les mco.

C. La méthode des Moindres Carrés Généralisés :

On introduit un autre estimateur appelé estimateur des moindres carrés généralisé. Il correspond à la minimisation de la distance entre les observations et l'espace engendré par les variables explicatives, non plus dans la métrique canonique de R^N, mais dans celle correspondant à Ω^{-1}.

2.1. La droite des moindres carrés :

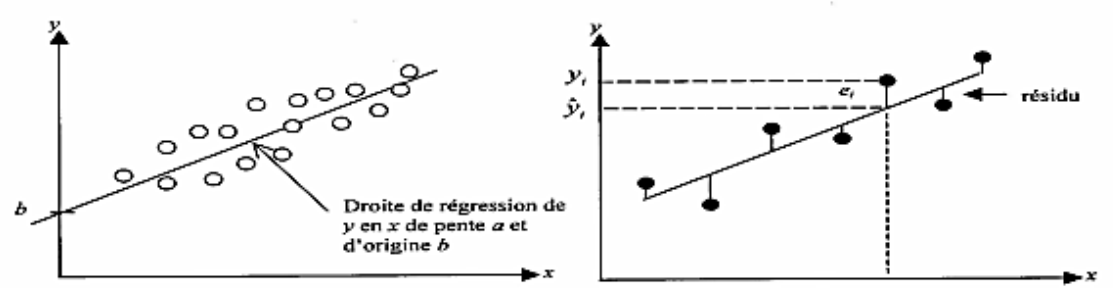
Les données $\{(x_i, y_i), i = 1,...,n\}$ peuvent être représentées par un nuage de n points dans le plan (x, y), le diagramme de dispersion.

Le centre de gravité de ce nuage peut se calculer facilement : il s'agit du point de coordonnées $(\bar{x}, \bar{y}) = (\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i)$.

Rechercher une relation affine entre les variables X et Y revient à rechercher une droite qui s'ajuste le mieux possible à ce nuage de points. Parmi toutes les droites possibles, on retient celle qui jouit d'une propriété remarquable : c'est celle qui rend minimale la somme des carrés des écarts des valeurs observées yi à la droite $y_i = ax_i + b$.

Si εi représente cet écart, appelé aussi résidu, le principe des moindres carrés ordinaire (MCO) consiste à choisir les valeurs de a et de b qui minimisent.

$$E = \sum_{i=0}^{n} {\varepsilon_i}^2 = \sum_{i=0}^{n} \big(y_i - (ax_i + b)\big)^2 \qquad (13)$$



Un calcul montre que ces valeurs, notées â et $\hat{b}$, sont égales à :

$$\hat{a} = \frac{\sum_{i=0}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n}(x_i - \bar{x})^2} \qquad (14)$$

$$\hat{b} = \bar{y} - \hat{a}\bar{x} \qquad (15)$$

On exprime souvent â au moyen de la variance de X, s_x^2, et de la covariance des variables.

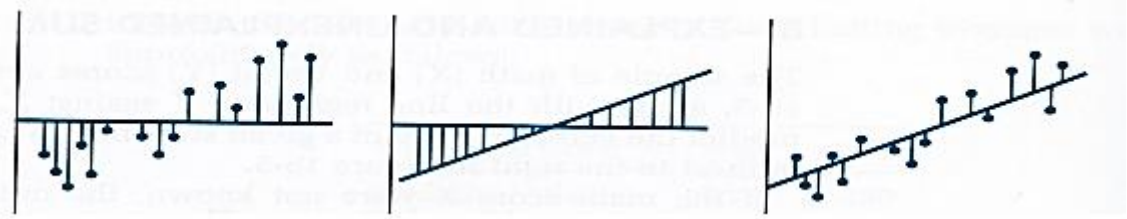


Figure 46 - Illustration de la formule DT=DA+DR

La droite horizontale passe par le centre de gravité du nuage ; la première figure représente la dispersion totale DT, la seconde la dispersion due à la régression DR (nulle si la pente de la droite des moindres carrés est nulle et importante si cette pente est forte) et la troisième la dispersion autour de la droite, ou dispersion résiduelle.

Aléatoires X et Y, cov_{xy} :

$$\hat{a} = \frac{cov_{xy}}{s_x^2} \tag{16}$$

Avec :

$$s_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x}) \tag{17}$$

Et :

$$cov_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \tag{18}$$

2.2. Evaluation de la qualité de la régression :

Pour mesurer la qualité de l'approximation d'un nuage $(x_i, y_i)_{i=1..n}$ par sa droite des moindres carrés (après tout on peut toujours faire passer une droite par n'importe quel nuage !), on calcule son coefficient de corrélation linéaire défini par :

$$r_{xy} = \frac{cov_{xy}}{s_x s_y} \tag{19}$$

C'est un nombre compris entre −1 et +1, qui vaut +1 (resp. −1) si les points du nuage sont exactement alignés sur une droite de pente a positive (resp. Négative). Ce coefficient est une mesure de la dispersion du nuage.

On considère que l'approximation d'un nuage par sa droite des moindres carrés est de bonne qualité lorsque $|r_{xy}|$ est proche de 1 (donc r_{xy} proche de +1 ou de −1) et de médiocre qualité lorsque $|r_{xy}|$ est proche de 0.

En pratique on estime souvent la régression acceptable lorsque $|r_{xy}| \geq \sqrt{3}\ 2 = \sqrt{0,\ 75} = 0.866$ .... Parfois on préfère calculer non plus r_{xy} mais son carré noté $R^2 = r_{xy}r_{xy}$ car on a la relation suivante :

$$\sum(y_i - \bar{y})^2 = \sum(y_i - \widehat{y_i})^2 + \sum(\widehat{y_i} - \bar{y})^2 \qquad (20)$$

Qui exprime que la dispersion totale de Y (DT) est égale à la dispersion autour de la régression (DA) plus la dispersion due à la régression (DR). Or on peut vérifier que l'on a R^2 = DR/DT, c'est-à-dire que le R^2 représente la part de la dispersion totale de Y que l'on peut expliquer par la régression.

Ainsi si l'on obtient une valeur de R^2= 0,85 (et donc r = ±0, 92 ...), cela signifie que la modélisation par la droite des moindres carrés explique 85% de la variation totale, ce qui est un très bon résultat. Cependant, même avec un R^2 excellent (proche de 1), notre modèle linéaire peut encore être rejeté.

En effet, pour être assuré que les formules données â et $\hat{b}$ fournissent de bonnes estimations de la pente et de l'ordonnée à l'origine de la droite de régression, il est nécessaire que les résidus εi soient indépendant et distribués aléatoirement autour de 0.

Ces hypothèses ne sont pas forcément faciles à vérifier. Un tracé des résidus et un examen de leur histogramme permet de détecter une anomalie grossière mais il faut faire appel `a des techniques statistiques plus élaborées pour tester réellement ces hypothèses.

2.3. Prévisions :

Si y = âx + $\hat{b}$ est la droite des moindres carrés d'un nuage de points $(x_i, y_i)_{i=1..n}$, on appelle valeurs prédites de y par le modèle les valeurs $\widehat{y_i} = \hat{a}x_i + \hat{b}$.

Notons cependant que s'il peut sembler naturel d'utiliser une valeur prédite pour compléter les données initiales dans l'intervalle des valeurs de X, on se gardera de prédire sans de multiples précautions supplémentaires des valeurs de X en dehors de cet intervalle.

En effet il se peut que la relation entre X et Y ne soit pas du tout linéaire mais qu'elle nous soit apparue comme telle `a tort parce que les xi sont proches les uns des autres.

3. Rétropropagation :

La technique de rétropropagation du gradient (Backpropagation en anglais) est une méthode qui permet de calculer le gradient de l'erreur pour chaque neurone du réseau, de la dernière couche vers la première. Les algorithmes traditionnels de correction d'erreurs basés sur le calcul des gradients par rétropropagation sont souvent appelés techniques de rétropropagation de gradient, et c'est exactement la méthode présentée ici. En fait, la correction d'erreur peut se faire par d'autres moyens, notamment en calculant la dérivée seconde.

La technique consiste à corriger les erreurs en fonction de l'importance des facteurs qui ont précisément contribué à la réalisation de ces erreurs. Pour les réseaux de neurones, les poids synaptiques qui contribuent à la production de grandes erreurs sont plus modifiés que les poids qui produisent de petites erreurs.

3.1. Algorithme de rétropropagation :

Cet algorithme permet de réaliser un apprentissage du réseau de neurones. On cherche à obtenir du réseau une réponse préétablie comme étant correcte. On dispose d'une base de connaissance de type entrée (p) - sortie attendue (d).

On compare ensuite la sortie (a) à la sortie attendue. On introduit une fonction e = d−a qui est la fonction d'erreur que l'on va chercher à minimiser en modifiant les poids du réseau. Une fois les poids définis par cet algorithme à partir d'exemples connus de l'utilisateur, on va chercher à extrapoler le réseau, en lui fournissant des entrées inconnues.

Voici un rappel des différentes notations utilisées pour décrire un réseau de neurones :

- L est le nombre de couches du neurone.
- σ est la fonction d'activation Sigmoïde.
- w_{ij}^{l}est le poids qui relie le i^{ième} neurone de la l^{ième} couche au j^{ième} neurone de la (l - 1)^{ième} couche.

- w^l est une matrice de taille i x j (i lignes et j colonnes) qui contient tous les poids de la l^ième couche.
- b_i^l est le biais associé au i^ième neurone de la l^ième couche.
- b^l est le vecteur qui contient tous les biais de la l^ième couche.

Nous allons également introduire deux éléments de notation supplémentaires :

z_i^l est la valeur d'agrégation du i^ième neurone de la l^ième couche, c'est à dire la valeur qu'un neurone calcule avant de la passer à la fonction d'activation.

$$z_i^l = \sum_{j=1}^{n} w_{ij}^l a_j^{l-1} + b_i^l \tag{21}$$

a_i^l est la valeur d'activation du i^ième neurone de la l^ième couche, c'est à dire la valeur définitive crachée par le neurone. On a donc $a_i^l = \sigma(z_i^l)$.

A. L'algorithme de la rétropropagation du gradient :

Nous pouvons maintenant entrer dans le vif du sujet et aborder le fonctionnement de l'algorithme d'apprentissage dans un réseau de neurones**.**

À quelques subtilités près, entraîner un réseau de neurones fonctionne exactement sur le même principe que pour les modèles plus simples : répéter jusqu'à ce que C(w, b) converge, pour tous les poids w_{ij}^l et b_i^l :

$$w_{ij}^l \leftarrow w_{ij}^l - \alpha * \frac{\partial C}{\partial w_{ij}^l}$$

$$b_i^l \leftarrow b_i^l - \alpha * \frac{\partial C}{\partial b_i^l}$$

Où α est le taux d'apprentissage.

Si l'algorithme ne change pas, la seule difficulté, ici, réside dans le calcul des différentes dérivées partielles.

D'habitude, si la fonction est simple, il suffit d'appliquer les règles de dérivation en s'aidant de quelques dérivées usuelles pour obtenir une dérivée en bonne et due forme.

Or, un réseau de neurones - modèle constitué de potentiellement millions de fonctions composées entres elles- n'est *pas* une fonction simple. Il faut trouver une autre solution.

Il existe heureusement une solution qui porte le doux nom d'algorithme de rétropropagation de gradient (backpropagation en angliche).

Avant d'étudier l'algorithme en détail, commençons par essayer de comprendre son fonctionnement de manière intuitive.

B. Rétropropagation du gradient : intuition :

Imaginons une donnée d'entraînement (X, T) avec X le vecteur qui contient les entrées, et T (pour target) la sortie attendue.

Donnons X à notre réseau de neurones. Les calculs se propagent de couche en couche jusqu'à la sortie qu'on notera Y.

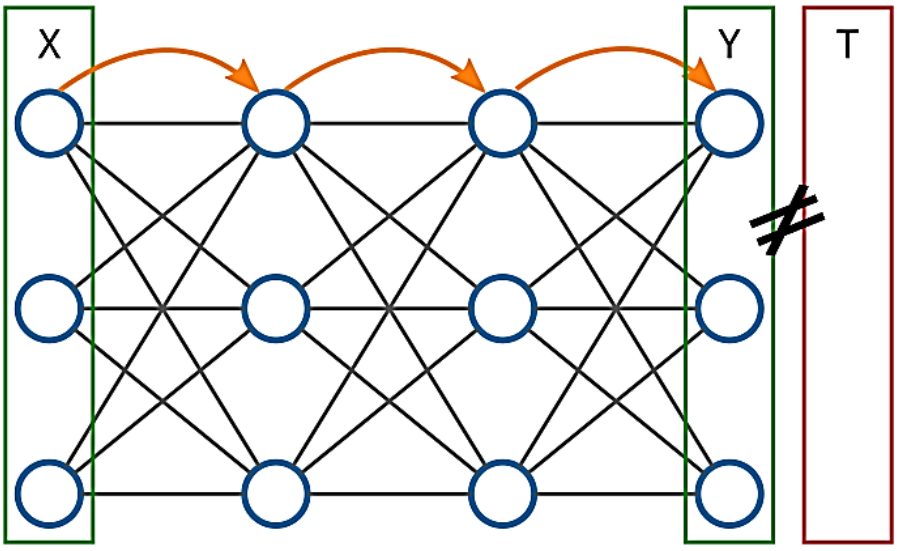


Figure 47 - Propagation des calculs dans le réseau de neurones.

Si notre réseau n'est pas entraîné, il y a des chances pour que Y et T soient différents. Nous pouvons calculer très simplement l'erreur du modèle par l'équation suivante : E = T - Y (on note parfois E : δ).

Puisque nous connaissons la valeur attendue pour les neurones de la dernière couche, il est assez facile de savoir dans quelles mesures les poids associés à chaque neurone ont contribué à cette erreur.

C'est à dire qu'il existe une formule (que nous verrons en détails plus loin) pour obtenir toutes les valeurs $\partial C / \partial w_{ij}^{l}$ à partir de δ^{L}.

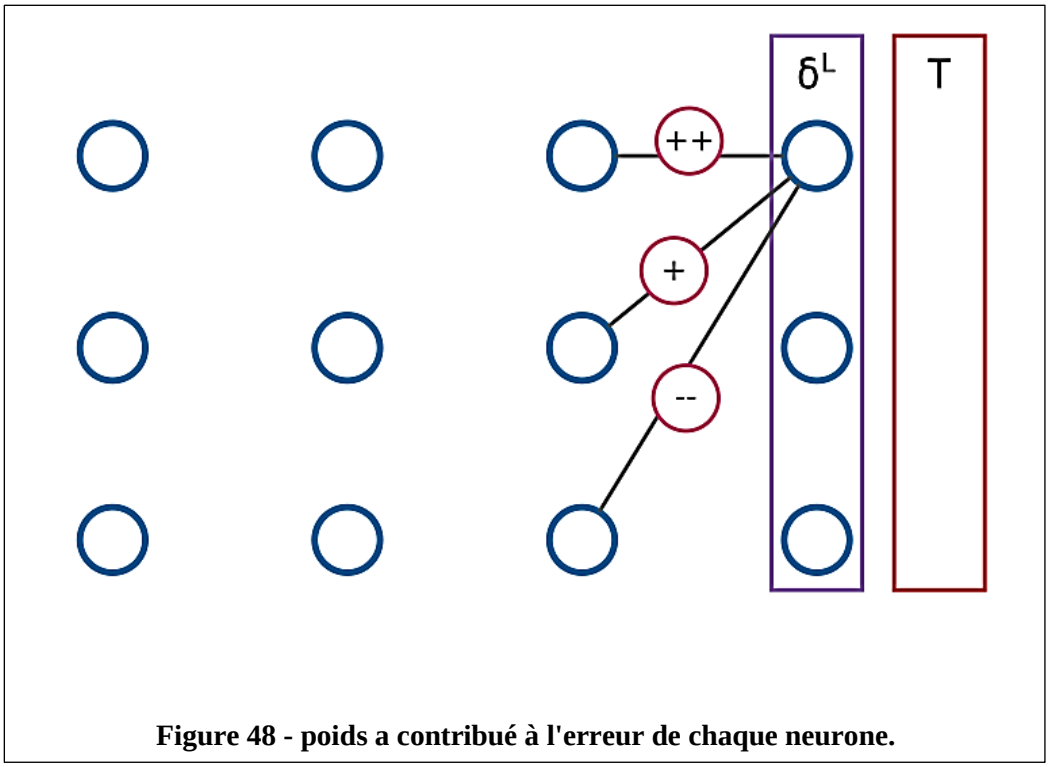


Figure 48 - poids a contribué à l'erreur de chaque neurone.

Nous connaissons la valeur attendue de la dernière couche, et nous savons quels calculs ont permis de passer de l'avant-dernière à la dernière couche. Par conséquent, il nous est possible, grâce à d'astucieuses opérations que nous verrons plus en détails ensuite, de calculer dans quelle mesure les neurones de l'avant-dernière couche ont contribué à l'erreur. En gros, nous pouvons calculer l'erreur de l'avant-dernière couche.

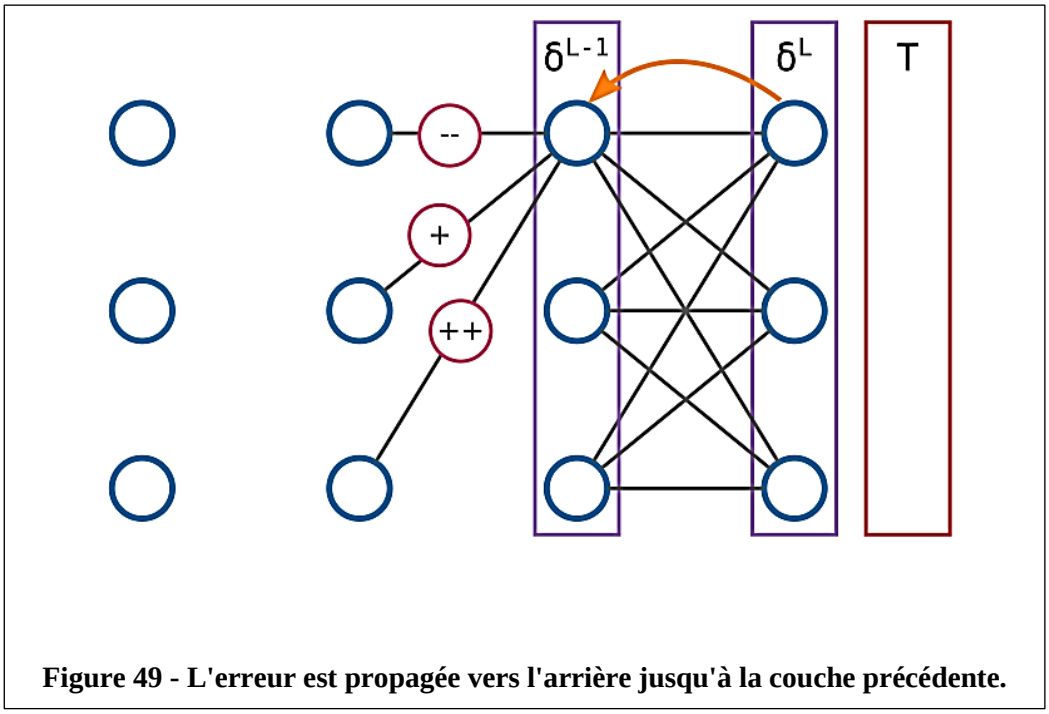


Figure 49 - L'erreur est propagée vers l'arrière jusqu'à la couche précédente.

Et si nous connaissons l'erreur de l'avant-dernière couche, nous pouvons calculer l'erreur de la couche précédente, et ainsi de suite, jusqu'à la première.

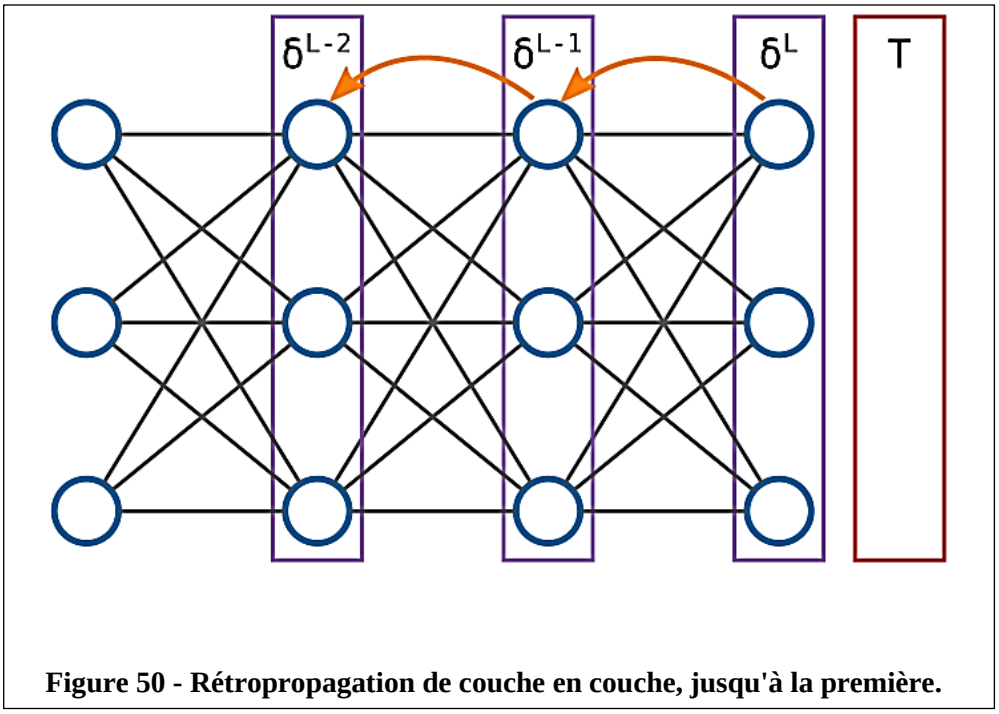


Figure 50 - Rétropropagation de couche en couche, jusqu'à la première.

Une fois que nous avons obtenu l'erreur de toutes les couches, il nous est facile d'appliquer la même bête formule pour obtenir les dérivées partielles de tous les poids du réseau d'un seul coup.

Ainsi, en une seule passe vers l'avant suivie d'une seule passe vers l'arrière, nous avons potentiellement calculé des millions de dérivées partielles d'une manière algorithmiquement efficace. C'est cette propagation de l'erreur du réseau vers l'arrière qui a donné son nom à l'algo de la rétropropagation du gradient.

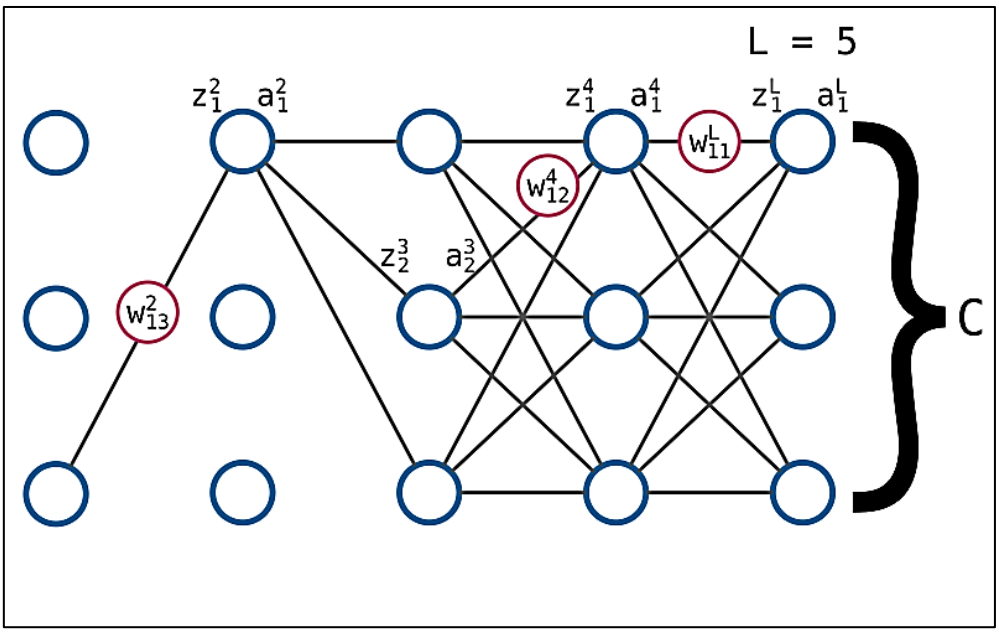
Voici le pseudo-code de l'algorithme d'apprentissage d'un réseau de neurones (algorithme du gradient + rétropropropagation du gradient) :

```
Initialiser le modèle avec les poids aléatoires
Tant que l'entraînement n'est pas terminé :
    Pour chaque exemple de la liste des données d'entraînement :
        Donner l'entrée au modèle pour obtenir la sortie
        Calculer l'erreur en comparant la sortie avec le résultat attendu
        Propager l'erreur de couche en couche vers l'arrière
        Mettre à jour tous les poids du réseau
```

Il ne s'agissait ici que de décrire sommairement le fonctionnement de l'algorithme.

C. Calcul d'erreur pour la dernière couche :

Nous avons le pseudo-code de l'algorithme, donc il nous faut étudier comment la modification d'un seul poids peut impacter le résultat de la fonction de coût.



Imaginons que dans le réseau de neurones illustré ci-dessus, pris d'un coup de folie, je décide de modifier un tout petit peu le poids w_{11}^L en lui ajoutant une petite valeur que je noterai Δw_{11}^1 .

En modifiant ce poids, je modifie la valeur de sortie du réseau, en donc la valeur calculée par la fonction de coût. Cette modification de la valeur C, je la noterai ΔC.

Par définition, on a l'équation suivante :

$$\frac{\partial C}{\partial w_{11}^L}\Delta w_{11}^L = \Delta C \tag{22}$$

On peut mieux comprendre comment une variation de w_{11}^L impacte C en détaillant les impacts sur les valeurs intermédiaires.

D'abord, modifier w_{11}^L va modifier le calcul de z_1^L. En suivant l'exemple ci-dessus, on peut modéliser cet impact de la façon suivante :

$$\frac{\partial z_1^L}{\partial w_{11}^L}\Delta w_{11}^L = \Delta z_1^L \tag{23}$$

Puisque z_1^L est utilisé pour calculer a_1^L, il est évident qu'une variation du premier aura un impact sur le second.

$$\frac{\partial a_1^L}{\partial z_1^L}\Delta z_1^L = \Delta a_1^L \tag{24}$$

Enfin, puisque la fonction de coût dépend de la sortie du réseau, une variation de a_1^L, fait varier C.

$$\frac{\partial C}{\partial a_1^L}\Delta a_1^L = \Delta C \tag{25}$$

En considérant les trois équations précédentes, il est facile d'effectuer quelques remplacements et simplifications pour obtenir l'équation suivante :

$$\frac{\partial C}{\partial w_{11}^L} = \frac{\partial z_1^L}{\partial w_{11}^L}\frac{\partial a_1^L}{\partial z_1^L}\frac{\partial C}{\partial a_1^L} \tag{26}$$

En clair : pour calculer comment w fait varier C, il suffit de calculer comment w fait varier z, comment z fait varier a, et comment a fait varier C. En fait, il ne s'agit ni plus ni moins que d'une illustration du théorème de dérivation des fonctions composées.

$\frac{\partial C}{\partial a_1^L}$ Est la variation de la fonction de coût en fonction de la sortie du réseau. Ce n'est rien d'autre que la dérivée de la fonction de coût**.**

Cette dérivée dépend évidemment de la fonction de coût utilisée, mais son calcul ne présente pas de problème particulier.

$\frac{\partial a_1^L}{\partial z_1^L}$ Désigne la variation de la fonction d'activation en fonction de l'agrégation. Pour obtenir cette valeur, il nous suffit de calculer la dérivée de la fonction d'activation.

$$\frac{\partial a_1^L}{\partial z_1^L} = \sigma'(z_1^L) \tag{27}$$

Enfin, $\frac{\partial z_1^L}{\partial w_{11}^L}$ est la variation de la fonction d'agrégation en fonction de ce seul poids. En utilisant les règles usuelles de dérivation, il est possible de montrer que cette valeur est exactement égale à a_1^{L-1}.

$$\frac{\partial z_1^L}{\partial w_{11}^L} = a_1^{L-1} \tag{28}$$

Si on récapitule, nous avons maintenant tout ce qu'il nous faut pour établir l'équation suivante :

$$\frac{\partial C}{\partial w_{11}^L} = a_1^{L-1} * \sigma'(z_1^L) * \cos t'(a_1^L) \tag{29}$$

Qu'on généralisera ainsi :

$$\frac{\partial C}{\partial w_{ij}^L} = a_j^{L-1} * \sigma'\left(z_i^L\right) * \cos t'\left(a_i^L\right) \tag{30}$$

Dans un but d'optimisation, on calculera d'abord la valeur intermédiaire

$$\delta_i^L = \sigma'\left(z_i^L\right) * \text{cost}'\left(a_i^L\right) \tag{31}$$

S'il vous reste un peu de matière grise en état de marche, vous n'aurez pas manqué de remarquer l'égalité suivante :

$$\delta_i^L = \sigma'\left(z_i^L\right) * \text{cost}'\left(a_i^L\right) = \frac{\partial a_i^L}{\partial z_i^L}\frac{\partial C}{\partial a_i^L} = \frac{\partial C}{\partial z_i^L} \tag{31}$$

On obtient enfin :

$$\frac{\partial C}{\partial w_{ij}^L} = a_j^{L-1} * \delta_i^L \tag{32}$$

D. Généralisation :

Même si nous avons utilisé des poids spécifiques pour notre étude, nous avons assez d'éléments pour généraliser et écrire les équations nécessaires à l'implémentation de l'algorithme :

$$\delta_i^L = \sigma'\left(z_i^L\right) * \text{cost}'\left(a_i^L\right) \tag{33}$$

$$\delta_i^l = \sigma'(z_i^L) * \sum_j w_{ji}^{l+1}\delta_j^{l+1} \tag{34}$$

$$\frac{\partial C}{\partial w_{ij}^L} = a_j^{l-1}\delta_i^l \tag{35}$$

$$\frac{\partial C}{\partial b_i^l} = \delta_i^l \tag{36}$$

Notez que la dernière équation est balancée en lucide sans aucune justification. C'est parce que je suis en train de faire une indigestion de maths.

4. Conclusion :

Les méthodes d'optimisation et de fiabilité des systèmes présentés permettent de satisfaire le niveau de fiabilité structurale requis et la détermination de la meilleure conception possible en termes de coût et de qualité.

Chapitre 3 :

Simulation avec ANFIS sur Matlab

Chapitre 3 : Simulation avec ANFIS sur Matlab

1. Mise en œuvre avec ANFIS sur Matlab :

Nous avons opté pour l'approche neuro-floue. Nous présentons ici l'environnement Matlab qui a permis de mettre en œuvre cette méthode.

1.1.Description de l'interface ANFIS sur Matlab :

Les méthodes de DM permettent en général de tirer une/des règles afin de répondre au problème donné, à savoir un certain nombre d'inputs donnant un certain nombre d'outputs. Avec le bruit, les données fausses, il se peut que la règle ne soit pas forcément juste. C'est pour cela qu'il convient de fonctionner par étapes pour ne pas obtenir de règles qui soient juste dans notre cas précis mais non généralisable.

Les trois étapes à suivre sont énoncés ci-dessous :

- **Etape d'entrainement :** Lors de cette étape, nous allons fournir à la machine un package de données qui lui permettra d'en sortir une règle générale. Le système boucle sur lui-même jusqu'à obtenir une erreur minime (que nous décidons au départ).
- **Etape de vérification :** Cette deuxième étape est cruciale. Elle fonctionne avec un package de données indépendantes de la première étape. Elle se fait en parallèle et elle décide de l'arrêt de la boucle. A partir du moment où l'erreur sur le package de données augmente de nouveau cela signifie que nous sommes en train de trouver des règles non généralisables. Il faut donc arrêter la boucle au minimum de l'erreur de ces données.
- **Etape de test :** nous prenons encore un set de data extérieur aux autres pours, après tout apprentissage, analyser la réponse du système enfin créé.

La répartition des sujets entre les différentes étapes est importante. Dans des systèmes assez complexes, l'erreur va atteindre un minima pour un certain nombre de cas étudiés à l'entrainement. Dépassant ce nombre de cas on raugmente donc l'erreur faite sur le modèle. L'étape de vérification nous permettra de nous en rendre compte au fur et à mesure. Cela va justifier que les règles que nous trouvons sont généralisables car elles peuvent s'appliquer à des personnes dont les mesures n'ont pas été exploitées pour notre système. Des papiers sur les méthodes de DM nous indiquent des répartitions suivantes (Tango & Botta, Evalutation of Disctraction in a Driver-Vehicle-Environment Framework: An Application of Différent Data-

Mining Techniques, 2009) (Mukkamala, Janovski, & Sung, 2002) : 60% de données pour l'entrainement et 40% pour la vérification. C'est ce que nous allons suivre dans la suite de notre apprentissage.

Les figures ci-dessous nous donnent l'interface que nous avons sous Matlab pour la méthode neurofuzzy. L'écran central nous permet d'accéder aux autres fenêtres.

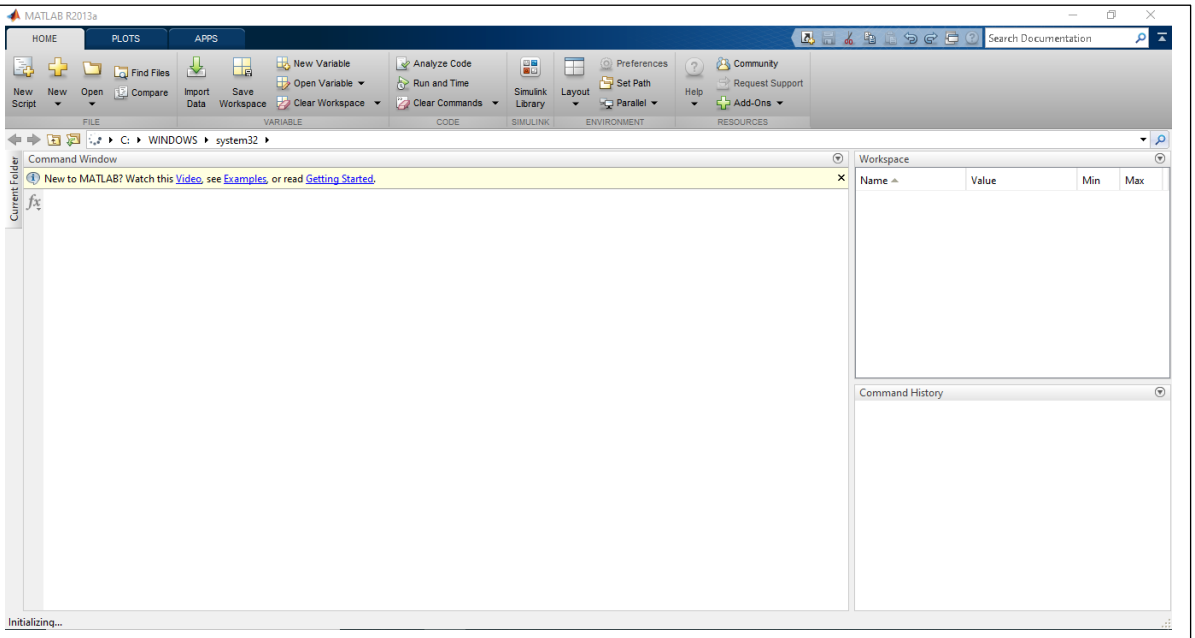


Figure 51 - Ecran central

Nous fonctionnons en interface Matlab et ANFIS fait partie de la ToolBox Fuzzy. Elle est assez simple à utiliser. Dans un premier temps nous devons préparer notre ensemble de données de la façon suivante : ce sera des matrices avec nos entrées sur toutes les premières colonnes et la sortie que nous désirons sur la dernière colonne. Nous pouvons charger la fenêtre centrale grâce à la commande anfisedit sur Matlab. A partir de cette fenêtre nous allons charger les données d'entrainement et de checking que nous avons préparées auparavant.

Ensuite il faut choisir un FIS (fenêtre en haut à gauche), c'est le modèle duquel on veut que notre système se rapproche. On peut ainsi choisir le nombre m de fonctions d'appartenance (Membership Fonctions) que nous souhaitons pour chaque entrée, et le type de fonction que nous voulons. Nous devons donc savoir en combien d'espaces différents nous voulons séparer nos données. Ces choix vont nous donner le nombre de règles r auxquelles vont répondre le système selon la formule ci-dessous. On considère sur cette dernière qu'il y a au moins deux entrées. Le cas avec une seule entrée est trivial.

$$r = m1 * m_i \qquad avec\ i \neq 1$$

m_i le nombre de séparation de l'entrée i.

Nous avons une représentation des liens entres les MFs, les inputs et les outputs (écran en bas à gauche). De même les règles floues obtenues sont visibles dans un autre écran (en haut à droite). A partir de ce dernier nous pouvons choisir de tester des entrées diverses pour étudier le comportement du système entrainé.

1.2.Apprentissage sur ANFIS :

L'apprentissage avec ANFIS sous Matlab va permettre d'ajuster tous les paramètres que nous avons présentés précédemment. Des données de références nous permettrons de les choisir pour obtenir une sortie qui soit le plus en phase avec celle que nous recherchons. La méthode la plus utilisée s'appelle la rétropropagation. Nous allons la détailler dans la suite de ce paragraphe.

Le système ANFIS va décrire une boucle. Celle-ci va permettre de déterminer dans un premier temps qu'elles sont les connections responsable de l'erreur et de combien elle participe à cette dernière. En fonction de ces informations ont va pouvoir modifier les poids qui ne vont pas.

L'erreur que nous utilisons est du type quadratique (le carré de la différence entre la sortie voulue et de la sortie obtenue). Le principe a modifié le poids des entrées du réseau en fonction de l'erreur que nous obtenons. Ainsi le système baissera le poids des neurones ayant le plus participé à l'erreur et en augmentera le poids des autres neurones.

2. Résultats :

2.1.Réglage de l'outil pour un bon apprentissage :

Dans le cadre de la tunnelisation attentionnelle nous ne disposons d'aucune connaissance experte. Nous avons un problème d'indicateur objectif qui nous dirait quand la personne est tunnélisée. Nous avons choisi de prendre l'alarme de batterie comme indice. Elle sera notre indicateur objectif. Nous pourrons considérer comme tunnélisée une personne qui entre en conflit avec le robot et ne détecte pas la panne à partir de la phase 4 et jusqu'à la fin. Et comme non tunnélisée une personne qui aura vu la panne grâce à la contre-mesure et du même coup pris conscience du problème. Grâce à la phase 4 de la mission nous avons donc un ensemble de données qui vont nous servir à entrainer le système ANFIS. Nous avons enlevé tous les cas que nous trouvions litigieux (car nous ne pouvions pas les considérer comme complètement tunnélisées où non tunnélisées).

Nous voulons un ensemble de données qui soit le plus propre possible. Pour cela nous avons considéré les personnes ayant reçu une contre-mesure comme non-tunnélisée (sortie égale à 0). Il y a un temps normal de détunnélisation. Le contre mesure prend déjà 6 secondes à se mettre en place. De plus des études chez Airbus ont permis de se rendre compte que le temps moyen de réaction d'un pilote face à une alarme « Pull Up » est de 3 à 5 secondes. Nous avons donc choisi de retirer les 10 premières secondes considérant que la personne n'est pas encore complètement détunnélisée. Pour les personnes n'ayant pas reçu de contre-mesure, nous n'avons choisi que les données pendant lesquelles le volontaire entrait en conflit avec le robot (sortie égale à 1).

Ces choix sont résumés dans la figure ci-dessous.

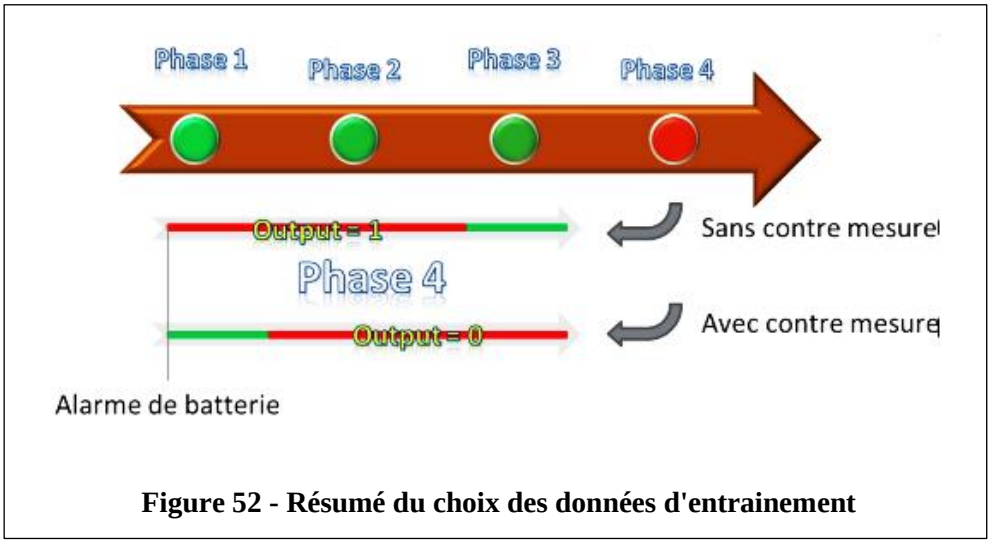


Figure 52 - Résumé du choix des données d'entrainement

L'entrainement peut maintenant commencer à partir de ces données. Il y a un certain nombre de paramètre à choisir dans ANFIS avant de commencer l'apprentissage.

A partir de ce package de données nous pouvons les diviser en deux parties. Une première partie servira à l'entrainement et la deuxième servira au checking. Nous allons mettre 60% des données dans le premier set et les 40% restant dans le deuxième (Mukkamala, Janovski, & Sung, 2002).

L'entrainement s'est fait sur un certain nombre de boucles pendant lesquelles la méthode va minimiser le taux d'erreur faite sur la partie entrainement. Pour obtenir un optimal il faut atteindre que l'erreur sur le set de checking remonte (voir graphe ci-dessous).

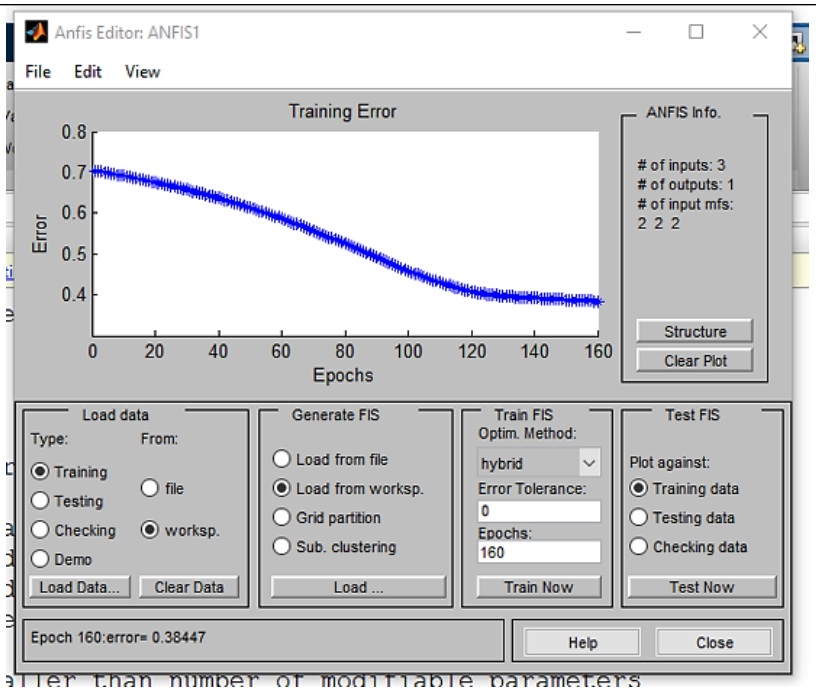


Figure 53 - Exemple d'entrainement d'un système ANFIS

La partie ANFIS système utilisé.

Nous considérons comme entrées X1, X2, et X3

X1 la note de Maths, X2 la note de Physique, et X3 la note de Chimie.

Les 3 notes seront considérées entre 8 et 16.

B =16 (bonne), A=8 (acceptable).

X1, X2, et X3 se divisent en deux règles floues A et B

Cette couche adaptative (couche de fuzzifcation) contient6 neurones qui transforment les données numériques des entrées mesurées par les capteurs en interprétations linguistiques. Chaque neurone calcule ses activations qui sont égales aux degrés d'appartenance des entrées Xi avec i = 1.2 dans les sous-ensembles flous représentes par les fonctions gaussiennes décrites par:

$$\mu_n = exp\frac{-0.5(x-c)}{\sigma} \qquad (37)$$

avec

c : Le centre de la règle floue représentée par une gaussienne

σ : L'écart type

- **La deuxième couche comporte aussi 8 neurones**

1-B B B

2-B B A

3-B A B

4-B A A

5-A B B

6-A A B

7-A B A

8-A A A

Cette couche appelée couche des règles, ou chaque neurone correspond à une règle floue, selon la méthode de Takagi- Sugeno. Les différents neurones calculent respectivement leur activation par un produit (operateur T-norme) pour donner la valeur de degré de vérité, exprimée par la relation suivante :

$\mu_i = \mu_{1j}(x_1).\mu_{2j}(x_2).\mu_{3j}(x_3).$

j= B, A

i= 1….k avec k=8 le nombre de règles selon la méthode de Sugeno-Takagi.

- **La troisième couche comporte 8 neurones**

C'est la couche de normalisation. Chaque neurone de cette couche reçoit la valeur de l'activation du neurone précédent et la somme de celles de la couche précédente et calcule le poids effectif de chaque règle en exprimant sa probabilité par :

$$\bar{\mu}_i = \frac{\mu_i}{\sum_{i=1}^{k} \mu_i} \qquad (38)$$

k=1….8

- **La quatrième couche comporte 8 neurones**

Couche adaptative ou couche de deffuzification. Le rôle de cette étape est important car il doit évaluer les valeurs des poids cohérents des règles données pour obtenir des résultats optimisés pour chaque règle. Chaque nœud de cette couche reçoit de la couche précédant la valeur normalisée correspondante et les entrées initiales. La deffuzzification est donnée par :

$$w_i = \bar{\mu}_i.f_i = \bar{\mu}_i[q_{i1}(x_1) + q_{i2}(x_2) + q_{i3}(x_3) + q_{i4}]$$

Avec tous, q_{in} $n = 1 \ldots 4$ les paramètres conséquents qui contribuent à donner la réponse désirée.

- **La cinquième couche**

Cette couche est représentée par un seul neurone noté par un nœud, et qui reçoit les différentes défuzzifications de la couche précédente pour déterminer la variable correspondant à l'angle de braquage du robot.

Cette variable est calculée par:

$$y = \sum_{i=1}^{k} \bar{\mu}_i . f_i = \frac{\sum_{i=1}^{k} \mu_i . f_i}{\sum_{i=1}^{k} \mu_i} \tag{39}$$

- **La Mise à jour :** l'apprentissage combine la méthode **des moindres carrés** et la méthode **de descente de gradient**. Les paramètres résultants q_{ij} sont calculés comme suit :

$$Y_d = A.q_{ij} \tag{40}$$

Tel que

$$Y_d = \sum_{j=1}^{k} y_{dj} \tag{41}$$

Avec Y_d le vecteur des valeurs désirées sachant que j=1….k.

k est le nombre de solutions, et A la matrice composée des paramètres du contrôleur ANFIS.

La méthode des moindres carrés calcule les paramètres conséquents q_{ij} Par :

$$q_{ij} = (A^T.A)^{-1}.Y_d.A^T \tag{42}$$

Une fois les paramètres conséquents q_{ij} obtenus et les solutions calculées nous les comparons avec les valeurs désirées en évaluant l'erreur exprimée par :

$$E_p = \sum_{j=1}^{l} \left(y_{dj} - y_j\right)^2 \tag{43}$$

La procédure de l'apprentissage par rétropropagation se développe par la série de dérivation se propageant de la couche représentant la sortie jusqu'aux couches entrées par l'expression suivante :

$$\frac{\partial E_p}{\partial y_j} = -2\left(y_{dj} - y_j\right) \tag{44}$$

En développant couche par couche nous obtenons :

$$\frac{\partial E_p}{\partial \alpha} = \frac{\partial E_p}{\partial y_j} * \frac{\partial y_{y_j}}{\partial w_i} * \dots\dots * \frac{\partial \mu_n}{\partial \alpha} \tag{45}$$

Avec, $\alpha = c \;\; et \;\; \alpha = \sigma$ pour ces deux paramètres de la mise à jour s'expriment comme suit :

$$\Delta\alpha = -\eta \frac{\partial E_p}{\partial \alpha} \tag{46}$$

η étant le rapport de l'apprentissage

Les règles floues selon Takagi sugeno :

R1= Si X1 est B et X2 est B et X3 est B alors $f_1 = [q_{11}(x_1) + q_{12}(x_2) + q_{13}(x_3) + q_{14}]$

R2= Si X1 est B et X2 est B et X3 est A alors $f_2 = [q_{21}(x_1) + q_{22}(x_2) + q_{23}(x_3) + q_{24}]$

⋮

R8= Si X1 est A et X2 est A et X3 est A alors $f_8 = [q_{81}(x_1) + q_{82}(x_2) + q_3(x_3) + q_{84}]$

2.2.Les étapes de la simulation :

Maintenant, on passe à l'application.

Les figures suivantes représentent les étapes du travail :

Dans un premier temps, on les données comme montrent les deux figures suivantes :

```
>> note2=[16 16;12 14;09 08;16 15;13 12;15 13;11 13;16 10;09 13;16 13;12 09;13 11]
note2 =
     16     16
     12     14
      9      8
     16     15
     13     12
     15     13
     11     13
     16     10
      9     13
     16     13
     12      9
     13     11

>> note4=[08 12;16 12;14 15;16 14;11 13;10 12;10 08;16 13;16 15;14 16;13 10;10 09]
note4 =
      8     12
     16     12
     14     15
     16     14
     11     13
     10     12
     10      8
     16     13
     16     15
     14     16
     13     10
     10      9
>>
>> note3=[12 15;13 16;11 15;16 16;13 11;16 15;11 11;16 14;15 15;10 11;10 09;09 08]
note3 =
     12     15
     13     16
     11     15
     16     16
     13     11
     16     15
     11     11
     16     14
     15     15
     10     11
     10      9
      9      8
```

Figure 54 - Les données

Puis on écrit la commande anfisedit, une interface apparaît, on clique sur File et sur new fis et choisis sugeno. On clique sur Edit et sur add variable choisis dans un premier temps input et refais la même chose pour obtenir 3 inputs.

Cliquer sur l'input les fonctions d'appartenances triangulaires apparait, puis recliquer sur edit et taper sur remove all variables, retape sur sur add variable, il apparait une petite interface, on choisit gaussmf et le nombre 2, ensuite on clique sur chaque courbe mf, on nomme la

première M (mauvaise note) la deuxième B (bonne note) et sur l'interface on tu replis range [8 16].

mf1=M et mf2=B on a fait ça pour toutes les trois inputs. Pour l'output on fait la même chose sauf que dans la petite interface on choisit constant et le nombre 8 qui représente le nombre de règles. On clique sur chaque mf et on la remplace par S1 jusqu'à S8.

Une fois avoir déclaré toutes les données, on clique sur edit et rules et on écrit les règles et on les exporte vers Workspace.

On revient vers l'interface anfisedit et on appuis sur load data pour charger nos données dans le Anfis comme on peut le voir dans la figure suivante :

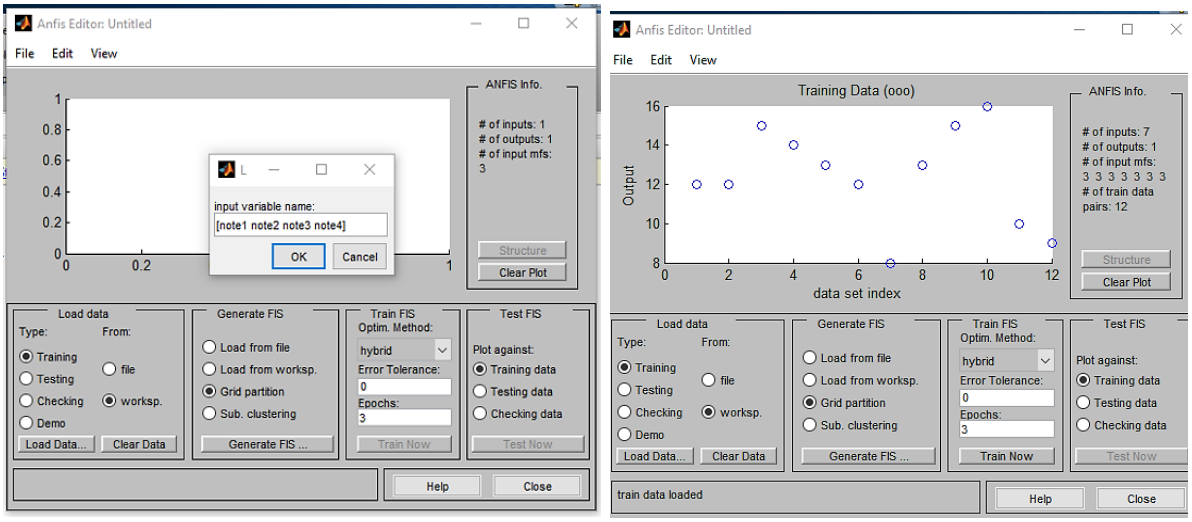


Figure 55 - Interface anfisedit

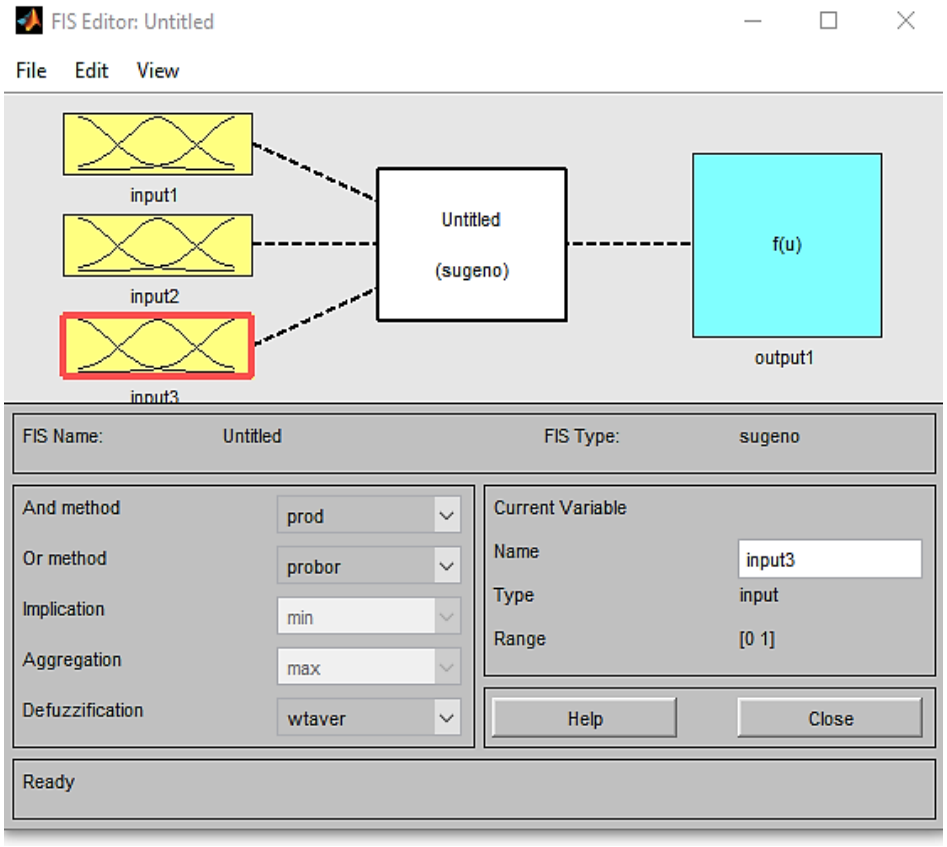


Figure 56 - FIS éditeur avec 3 inputs

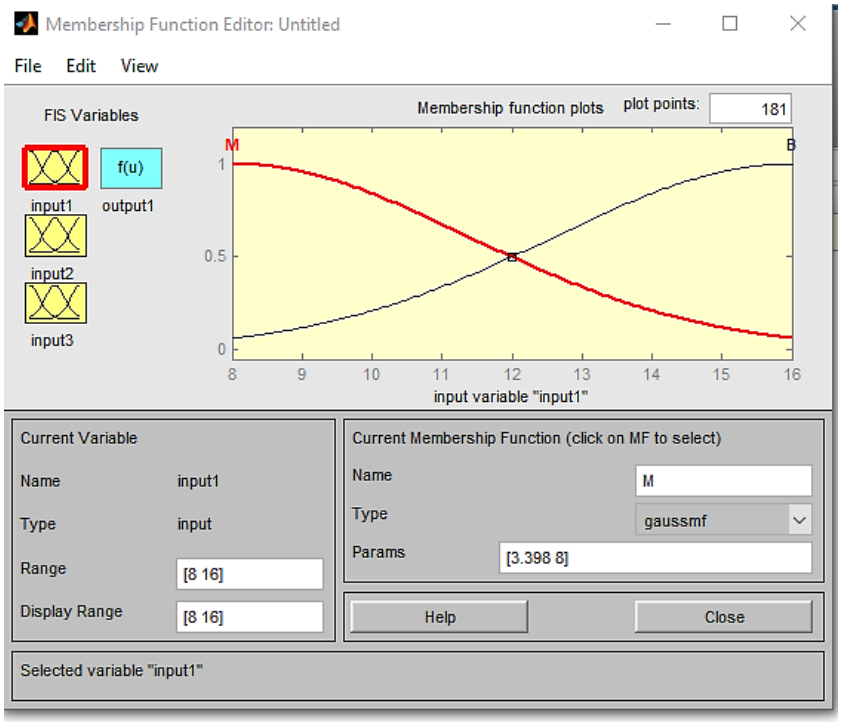


Figure 57 - FIS éditeur avec 3 inputs

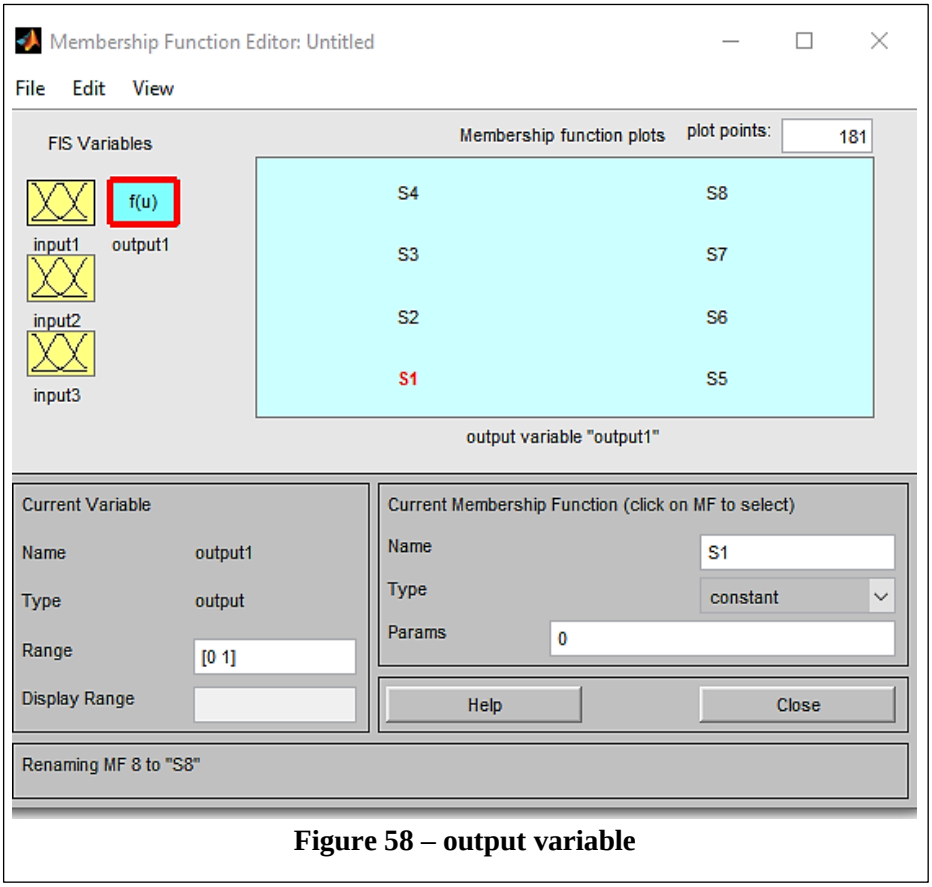


Figure 58 – output variable

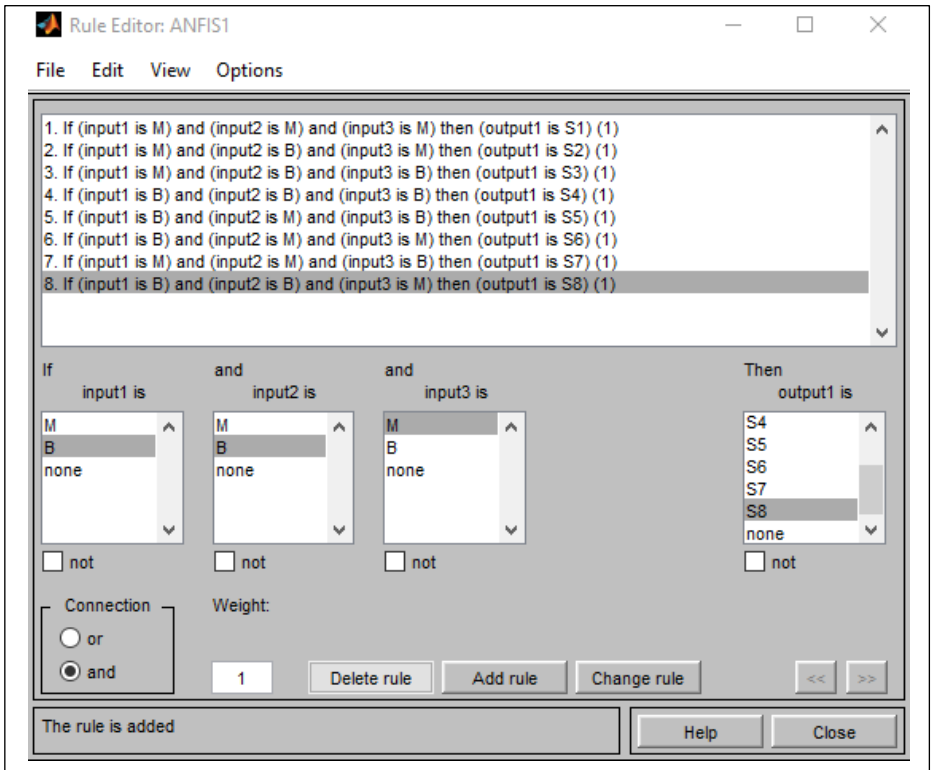


Figure 59 - Règle éditeur

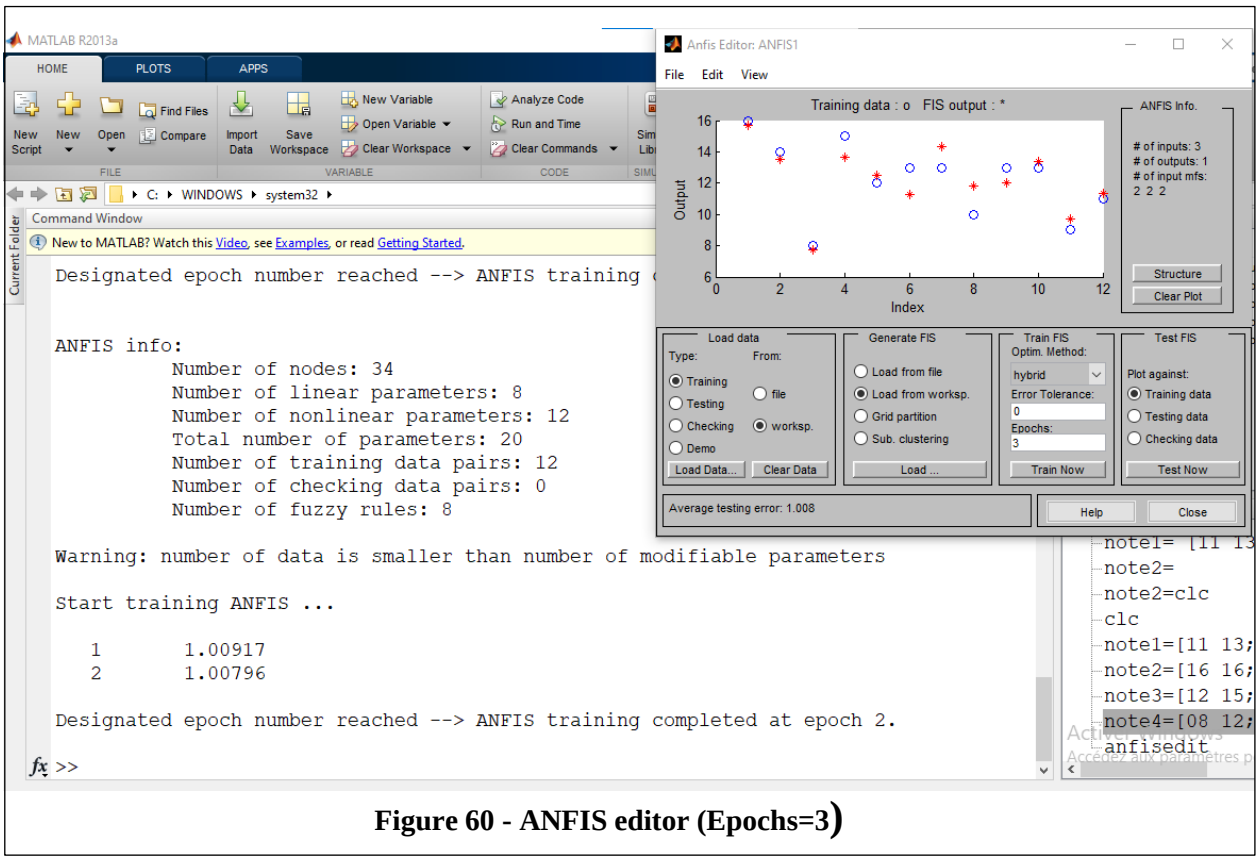


Figure 60 - ANFIS editor (Epochs=3)

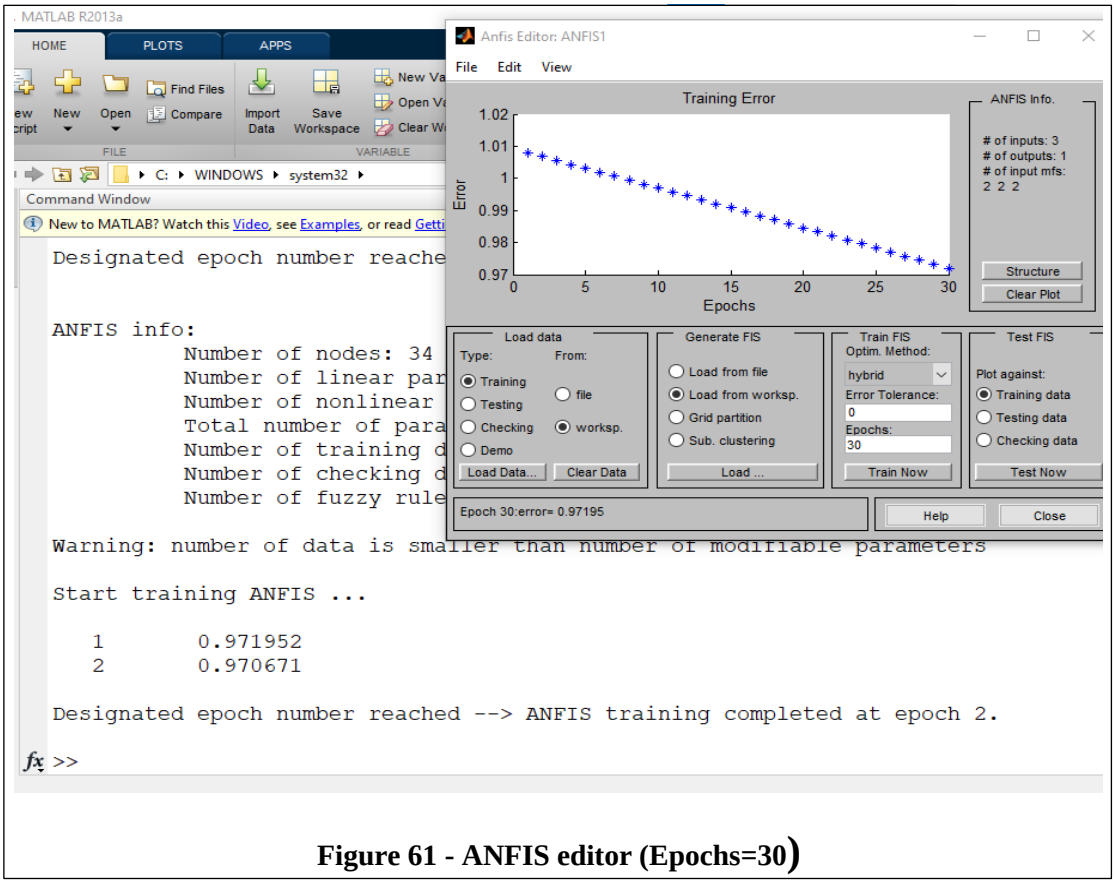


Figure 61 - ANFIS editor (Epochs=30)

Comme on peut le voir sur la figure au-dessus l'erreur est grande pour démunir cette erreur on augmente à chaque fois les Epochs jusqu'à avoir un résultat plus précis.

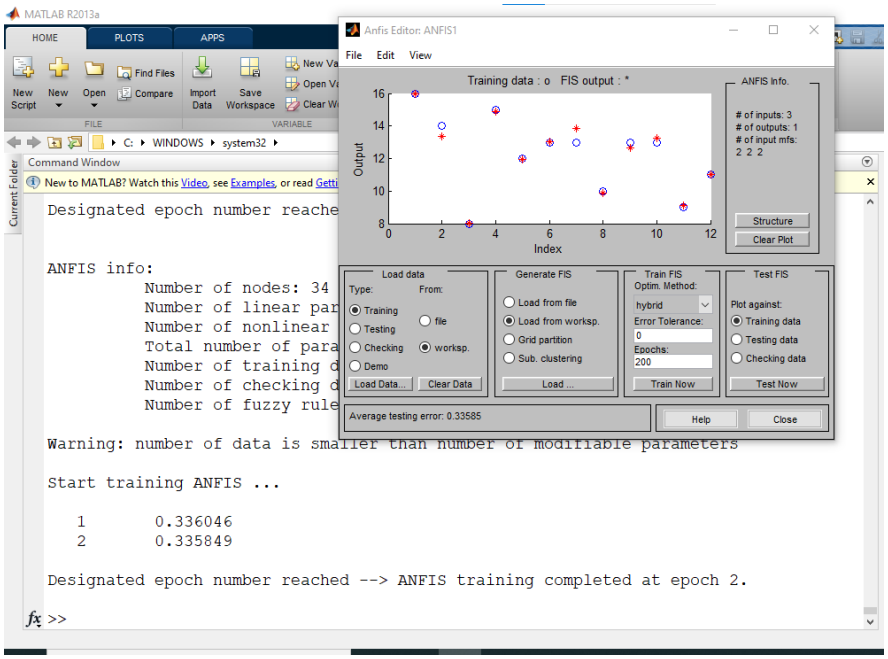


Figure 62 - ANFIS editor (Epochs=200)

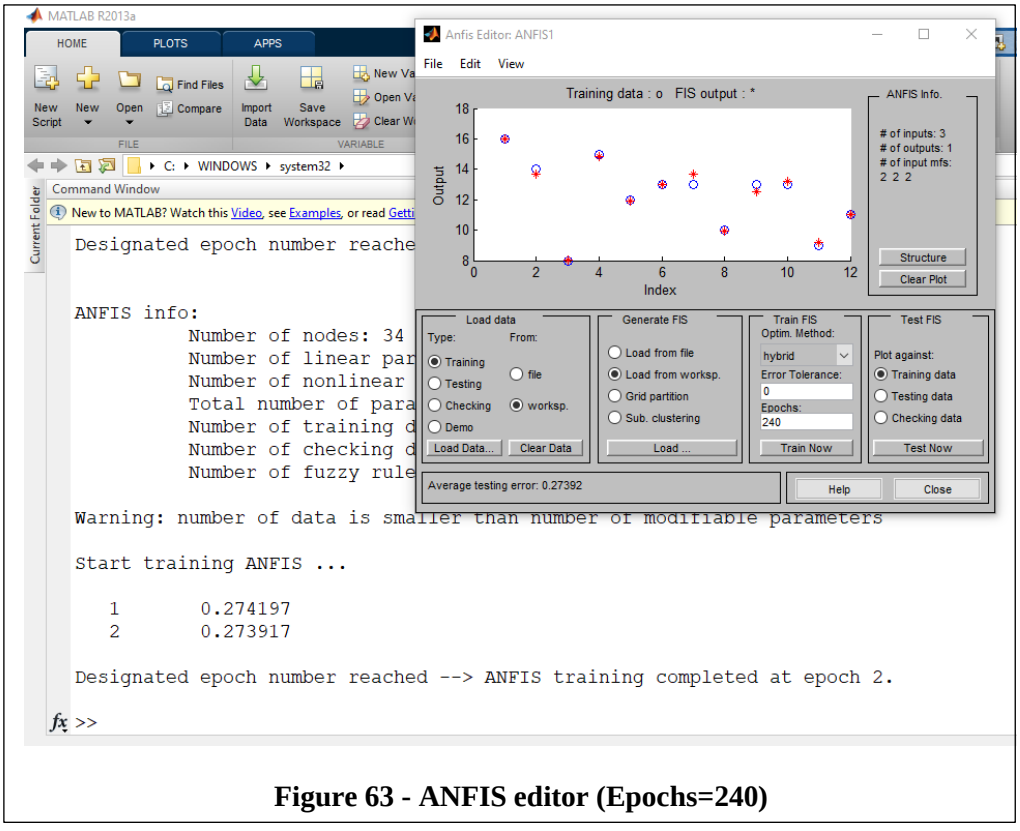


Figure 63 - ANFIS editor (Epochs=240)

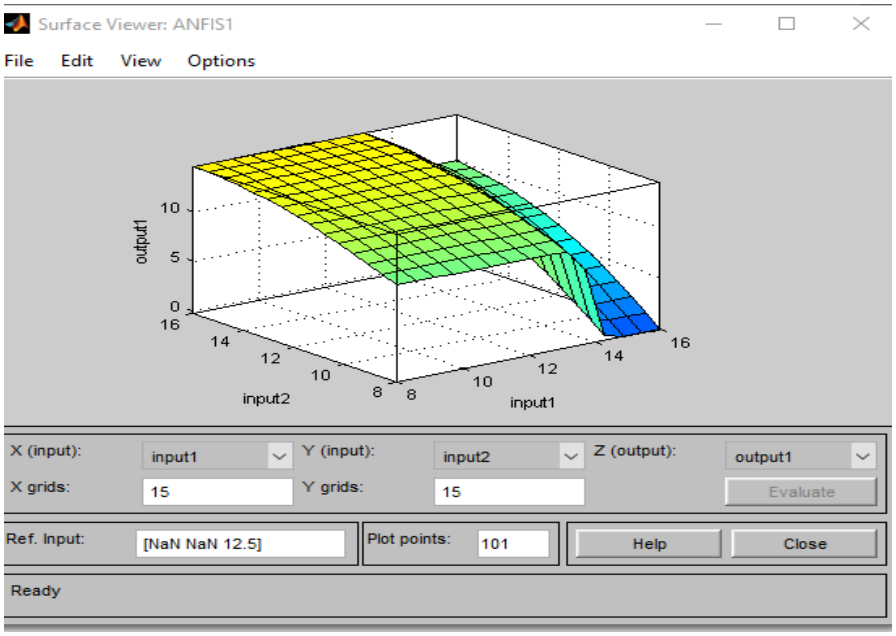


Figure 64 - La courbe en 3D de la sortie ;
S1 en fonction des deux entrées
B et M

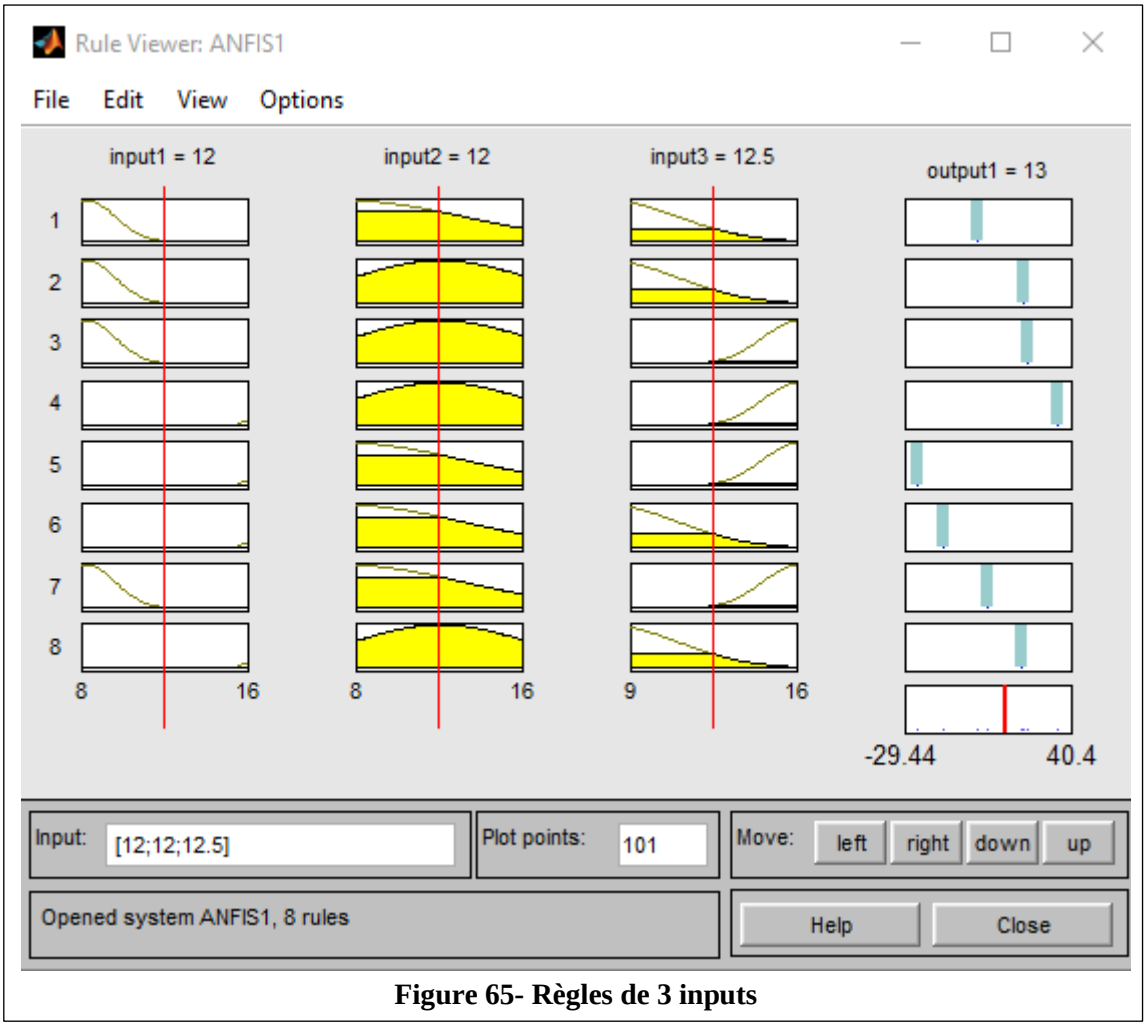


Figure 65- Règles de 3 inputs

Résultats et interprétation :

Après avoir simuler le anfis avec logiciel Matlab et étudier son rôle dans apprentissage on conclut que quand on augmente le nombre des itération (epochs) plus que le résultat est meilleur plus précis donc pour optimiser l'apprentissage il nous faut plusieurs itérations pour démunier l'erreur.

Conclusion générale :

L'objectif de ce mémoire est de permettre à des étudiants de choisir selon leurs notes la spécialité permise. Pour cela nous avons pensé à l'application d'une technique de l'intelligence artificielle à avoir l'algorithme ANFIS. Cette méthode utilise à la fois un raisonnement expert de la logique floue et la capacité de l'apprentissage de réseaux de neurones artificiels. L a collaboration de ces deux méthodes répond de manière efficace à l'optimisation dans l'étude des processus.

Le raisonnement expert doit définir les règles floues d'une manière intelligente et efficace de sorte que le moteur d'inférence arrive à trouver la meilleure mixture qui répondra favorablement aux attentes du processus. Suivra ensuite l'apprentissage qui est définit par une hybridation des méthodes des moindres carrés et de la rétro-propagation et qui permet au système d'acquérir la façon d'évoluer suivant une direction souhaitée, par la correction des caractéristiques des fonctions d'appartenance des règles floues définies par l'expert.

Ces deux méthodes appliquées à notre travail, nous font remarquer qu'après la simulation et dans la phase de l'apprentissage notre système s'améliore, et se rapproche du modèle désiré en fonction du nombre d'itérations utilisées.

Bibliographie :

[1] GELE5313, La logique floue , chapitre 11,Gabriel Cornier.

[2] Mémoire DÉVELOPPEMENT D'UN MODÈLE HYBRIDE DE PRÉDICTION D'ATTAQUES POUR LES RÉSEAUX VÉHICULAIRES AD HOC (VANETs), Université de Québec, CAROL Y GABRIELA PEREIRA DIAZ, Juillet 2018.

[3] Introduction à la logique floue, Franck Dernoncourt, Paris, Avril 2011.

[4] Les systèmes d'inférences flous, Université Tlemcen.

[5] Cours d'I.A. "Introduction à la logique floue", 3° année Antoine Cornuéjols.

[6] Open Archive TOULOUSE Archive Ouverte, Charles THOORIS, 2011.

[7] THE USE OF ADAPTIVE NEURO FUZZY INFERENCE SYSTEM (ANFIS) IN MODELING THE WELD OUTPUT OF A TIG WELDED PIPE JOINT, I. U. ABHULIMEN & J. I. ACHEBO, September 2014.

[8] ANFIS-based Wireless Sensor Network (WSN) Applications for Air Conditioner Control, Yi-Jen Mon , Chih-Min Lin , Imre J. Rudas, Acta Polytechnica Hungarica, 2013.

[9] SYSTEME HYBRIDE –ANFIS, Université Biskra, AE DJOKHRAB.

[10] SAMIR KENOUCHE - DEPARTEMENT DES SCIENCES DE LA MATIERE - UMKB METHODES NUMERIQUES ET PROGRAMMATION, 2019-2020.

[11] Algorithme de rétropropagation, emoutot.perso.math.cnrs.fr.

[12] Méthode des moindres carrés

[13] https://www.techno-science.net/definition/337.html

[14] https://www.miximum.fr/blog/introduction-au-deep-learning-2/