RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

UNIVERSITE CHEIKH CHAHID LARBI TEBESSI -TÉBESSA

FACULTÉ DES SCIENCES ET DE LA TECHNOLOGIE

DOMAINE DE FORMATION : **Génie Electrique**

Techniques de l'Intelligence Artificielle

Filière : *Electrotechnique*
Spécialité : *Électrotechnique Industrielle*
Niveau : *2eme Année Master*

Présenté par : **Dr. YOUSFI Laatra**

Année universitaire : **2023/2024**

Avant propos

Ce présent polycopié est un support pédagogique, destiné aux étudiants de la deuxième année Master, spécialité Electrotechnique Industrielle dans le cadre des programmes officiels. Ce document concerne le module Techniques de l'Intelligence Artificielle. Ce support de cours peut également être utilisé par les étudiants de spécialités similaires.

Ce cours est articulé autour des notions de bases des techniques de l'intelligence artificielle et son utilisation dans la commande, l'optimisation, le diagnostic et l'aide à la décision. Ce cours reprend les différentes topologies des réseaux de neurones et leurs algorithmes d'apprentissage, les différentes concepts de base de la logique floue et ses applications et, enfin, le principe des méthodes heuristiques et leur programmation.

Ce manuscrit est divisé en huit chapitres.

Le premier chapitre contient des généralités sur le "soft computing" et ses composantes essentielles. L'objectif du second chapitre est la théorie de la logique floue, ses applications et la constitution d'un système d'inférence floue avec un exemple d'application.

Le troisième chapitre donne des généralités sur les réseaux de neurones (définition, structure d'un neurone formel, les différents types des réseaux de neurones), principe d'apprentissage supervisé et non supervisé et le principe de l'algorithme de Back-propagation. Le quatrième chapitre concerne les réseaux de neurones adaptatifs et les réseaux neuro-flous. Le cinquième chapitre est dédié aux algorithmes génétiques. Le sixième chapitre donne des généralités sur les algorithmes évolutionnaires: recuit simulé, recherche tabou,… en détaillant la technique d'optimisation par essaims de particules. Le septième chapitre donne des généralités les probabilités et le raisonnement probabiliste. Le huitième chapitre donne des généralités sur les systèmes experts et ses applications (système expert floue et la prise de décision)

Ce document contient plusieurs exemples d'application avec leurs simulation sous Matlab/Simulink pour mieux comprendre les notions théoriques et savoir programmer les différentes techniques introduites.

Dr. Yousfi Laatra

SOMMAIRE

CHAPITRE I: INTRODUCTION A L'INTELLIGENCE ARTIFICIELLE

I. Intelligence naturelle et artificielle ***5***

II. Formes d'intelligence artificielle ***5***

III. Principaux domaines de l'IA ***6***

IV. Techniques de l'IA utilisées en Génie Electrique ***7***

CHAPITRE II: GENERALITES SUR "SOFT COMPUTING"

I. Introduction ***9***

II. Définition ***9***

III. Intérêt ***10***

IV. Composantes essentiels ***10***

V. Domaines d'applications ***10***

Un système intelligent flou (SIF) ***11***

Logique floue ***11***

Réseaux de neurones artificiels (RNA) ***11***

Les algorithmes évolutionnaires ***12***

Les algorithmes génétiques ***13***

VI. Comparaison entre Soft Computing et Hard Computing ***14***

CHAPITRE III: LA LOGIQUE FLOUE ET SES APPLICATIONS

I. Introduction ***16***

II. Historique ***16***

III. Avantages et inconvénients ***17***

IV. Différence entre la logique classique et logique floue ***17***

V. Ensembles floues et opérateurs flous ***19***

Sous-ensemble flou ***19***

Les fonctions d'appartenances ***20***

Caractéristiques des sous-ensembles flous ***21***

Opérateurs flous ***21***

VI. Systèmes flous ***22***

Exemple N°1 ***24***

Simulation de l'exemple par simulation sous Matlab ***29***

Exemple N°2 *35*

Exemple sur les mécanismes d'inférences *38*

CHAPITRE IV: LES RESEAUX DE NEURONES ARTIFICIELS

I. Introduction *41*

II. Historique des réseaux de neurones *41*

III. Domaines d'applications *42*

IV. Du neurone biologique au neurone formel *42*

V. Structure d'un neurone formel *45*

Fonction d'activation *45*

VI. Réseaux de neurones *46*

VII. Apprentissage (entrainement) *47*

a. Apprentissage supervisé *47*

b. Apprentissage non supervisé *48*

c. principe de l'algorithme de Back-propagation (B-P) *48*

VIII. EXEMPLES D'APPLICATION *51*

Exemple N°1 *51*

IX. LES RESEAUX A FONCTION DE BASE RADIALE (RBF) *53*

1. Définition et architecture du réseau RBF *53*

2. Algorithme d'apprentissage d'un réseau RBF *54*

X. LES RESEAUX DE NEURONES RECURRENT (RNN) *55*

CHAPITRE V: RESEAUX ADAPTATIFS ET RESEAUX NEURO-FLOU

I. Définition d'un système évolutif hybride *56*

II. Système Neuro-flou *56*

III. Modèles d'hybridation Neuro-flou *57*

1. Systèmes neuro-flou concurrents *57*

2. Réseaux Neuronaux flous *57*

3. Systèmes neuro-flou en coopération *57*

4. Systèmes neuro-flous hybrides *58*

5. Les modèles neuro-flous hybrides *59*

6. Systèmes d'inférence adaptatif neuro-flou (ANFIS) *60*

7. Entrainement d'un réseau ANFIS *61*

CHAPITRE VI: ALGORITHMES DE CALCUL EVOLUTIONNAIRE ET

D'INTELLIGENCE COLLECTIVE

I. ***Définitions*** ***62***

1. Les algorithmes évolutionnaires (AE) ***62***

2. L'intelligence collective ***62***

II. ***Les algorithmes génétiques*** ***62***

1. Définition ***62***

2. Principe des algorithmes génétiques ***63***

3. Comparaison entre les méthodes d'optimisation classiques et les algorithmes génétiques ***64***

4. Mécanisme de fonctionnement d'un algorithme génétique ***64***

a. La création de la population (Initialisation) ***65***

b. Le codage ***65***

c. Fonction d'évaluation ***66***

d. La sélection ***67***

e. La recombinaison génétique ***68***

Exemple ***70***

5. Applications ***71***

III. Algorithmes d'optimisation par essaims de particules ***78***

1. Définition ***79***

2. Les éléments de la PSO ***80***

3. Principe fondamental ***81***

IV. Algorithmes des colonies de fourmis ***83***

1. Introduction ***83***

2. Principe ***83***

3. Les fourmis artificielles ***84***

4. Algorithme ACO ***84***

CHAPITRE VII: PROBABILITE ET RAISONNEMENT PROBABILISTE

I. Raisonnement probabiliste ***88***

II. Réseaux Bayésiens ***89***

1. Définition ***89***

2. Construction d'un graphe ***89***

3. Domaines d'application ***90***

4. Exemple *90*

CHAPITRE VIII: SYSTEMES EXPERTS ET LEURS APPLICATIONS

I. Définition *93*

II. Structure et fonctionnement d'un système expert *93*

III. Domaines d'applications *94*

IV. Systèmes experts flous *95*

V. Systèmes experts en ingénierie électrique *96*

VI. Application au diagnostic *99*

CONCLUSION GENERALE *100*

BIBLIOGRAPHIE *101*

CHAPITRE I: *INTRODUCTION A L'INTELLIGENCE ARTIFICIELLE*

I. ***INTELLIGENCE NATURELLE ET ARTIFICIELLE:***

L'intelligence naturelle, humaine et animale non humaine, est une fonction composée de facultés à la fois mentale, psychique et psychophysiologique permettant de connaître et de comprendre. De plus, l'intelligence naturelle reste indissociable de la structure, à savoir les organes du cerveau et du corps.

En logique informatique, l'intelligence « artificielle » est désignée comme une « recherche de moyens susceptibles de doter les systèmes informatiques de capacités intellectuelles comparables à celles des êtres humains ». Dans le cadre des systèmes informatiques à la robotique, l'intelligence artificielle prétend être une fonction composée des mêmes facultés que celles englobées dans l'intelligence humaine. L'intelligence artificielle doit donc aussi rester indissociable de la structure, à savoir les organes d'un cerveau artificiel et d'un corps artificiel. En conséquence de quoi, le robot artificiel est la structure idéale pour voir l'émergence d'une intelligence artificielle forte. Notez qu'une intelligence artificielle faible se développe sur des systèmes informatiques par un programme de simulation de l'intelligence

II. ***FORMES D'INTELLIGENCE ARTIFICIELLE:***

Il existe trois types d'IA: l'intelligence artificielle étroite (ANI), l'intelligence artificielle générale (AGI) et la super intelligence artificielle (ASI).

- **L'intelligence artificielle étroite :** l'ANI est classée dans la catégorie d'intelligence artificielle faible car elle n'est spécialisée que dans une gamme étroite de paramètres ou de situations, comme la reconnaissance vocale ou les voitures sans conducteur, par exemple.
- **L'intelligence artificielle générale :** l'AGI est considérée comme une intelligence artificielle forte, car elle travaille à un niveau supérieur, qui correspond à l'intelligence humaine.

- **La super intelligence artificielle :** bien que ce type d'IA ne soit actuellement pas développé, l'ASI signifie qu'une machine dispose d'une super intelligence ou qu'elle est plus intelligente qu'un humain.

L'ANI est le type d'IA le plus simple à identifier, déjà utilisé par la plupart des individus. Voici quelques exemples d'ANI que vous reconnaîtrez certainement :

- **Les voitures sans conducteur :** les voitures sans conducteur n'ont ni volant ni pédales. Elles fonctionnent avec des cartes en 3D et sont contrôlées par un Chauffeur Google.
- **Les appareils à commande vocale:** de Siri à Alexa, la plupart des consommateurs possèdent et utilisent quotidiennement une interface ANI via des appareils intelligents.
- **Les filtres de messagerie:** toutes les boîtes de réception, sinon la plupart, offrent aux utilisateurs la possibilité de trier des spams et de signaler des messages spéciaux.

III. PRINCIPAUX DOMAINES DE L'IA:

L'IA présente un fort intérêt dans les domaines de la santé (imagerie médicale, bio-informatique, 17,5%), les transports (véhicules autonomes, avionique, 15,9%), la sécurité (14,9%) et les télécommunications (11,4%). Ces quatre domaines totalisent plus de 60% des applications IA.

Par ailleurs, les principaux domaines fonctionnels de l'IA témoignent de son immense potentiel, des innombrables réalisations auxquelles elle est susceptible de donner lieu. Il s'agit d'approfondir des domaines fonctionnels déjà connus, à l'instar des systèmes experts, de la planification/optimisation ou de la robotique, mais aussi de développer de nouveaux domaines tels que le machine learning, le natural language processing, la vision (capacité pour une machine d'appréhender son environnement) ou encore le speech (texte vers parole ou parole vers texte).

Le schéma ci-dessous est équivalant à des explications complètes, il nous montre les principaux domaines fonctionnels couverts par l'IA:



Figure (1.1): Domaines de l'intelligence artificielle

L'intelligence artificielle recouvre donc des domaines fonctionnels déjà connus tels que les systèmes experts, la planification/optimisation ou la robotique. Mais l'IA couvre également de nouveaux domaines tels que l'apprentissage automatique, le traitement du langage naturel, la vision (la capacité d'une machine à comprendre son environnement) ou encore le Speech (texte vers parole ou parole vers texte). Cette technologie consiste à mettre en œuvre un certain nombre de techniques visant à permettre aux machines d'imiter une forme d'intelligence humaine. L'IA se retrouve implémentée dans un nombre grandissant de domaines d'application.

IV. TECHNIQUES DE L'IA UTILISEES EN GENIE ELECTRIQUE:

L'intelligence artificielle (IA), comme la robotique, est une discipline aux contours peu précis. Si on laisse de côté l'intelligence artificielle forte, encore lointaine, qui vise à simuler un comportement humain dans l'ensemble des champs de la connaissance et des interactions, l'intelligence artificielle dite faible est une famille de méthodes utilisées pour résoudre des problèmes. Le terme intelligence vient de la ressemblance avec le raisonnement humain. Avant les années 2010, les systèmes dits intelligents calculaient rapidement des solutions aux problèmes à

partir de lois comportementales codées par des ingénieurs spécialistes du problème en question. Cela allait du jeu d'échec électronique pour débutant au système de délestage du réseau électrique. Depuis l'apparition des premiers neurones artificiels inspirés de nos neurones biologiques, la ressemblance avec l'intelligence humaine est descendue au niveau de l'apprentissage. Avec des algorithmes complexes pour optimiser l'apprentissage de centaines de couches composées de milliers de nœuds, les réseaux de neurones modernes se sont éloignés de leur modèle biologique et offrent aujourd'hui des perspectives de progrès dont on peine à envisager les limites. Chaque avancée de l'IA, ses victoires aux échecs puis au jeu de go, ses créations musicales ou picturales, ses traductions instantanées, ses diagnostics médicaux interrogent sur les métiers qui lui seraient inaccessibles et sur la nature de l'intelligence humaine. En permettant de traiter des systèmes complexes difficilement descriptibles par des équations ou des arbres (images, voix humaines, …), les réseaux de neurones sont un outil très prometteur également pour les sciences de l'ingénieur, que ce soit en les utilisant dans leurs champs de prédilection (le traitement d'images, de la voix, du langage) ou directement pour le contrôle de machines ou la modélisation de matériaux. Cela explique leur apparition dans les formations d'ingénieur, notamment dans le programme de sciences de l'ingénieur de CPGE.

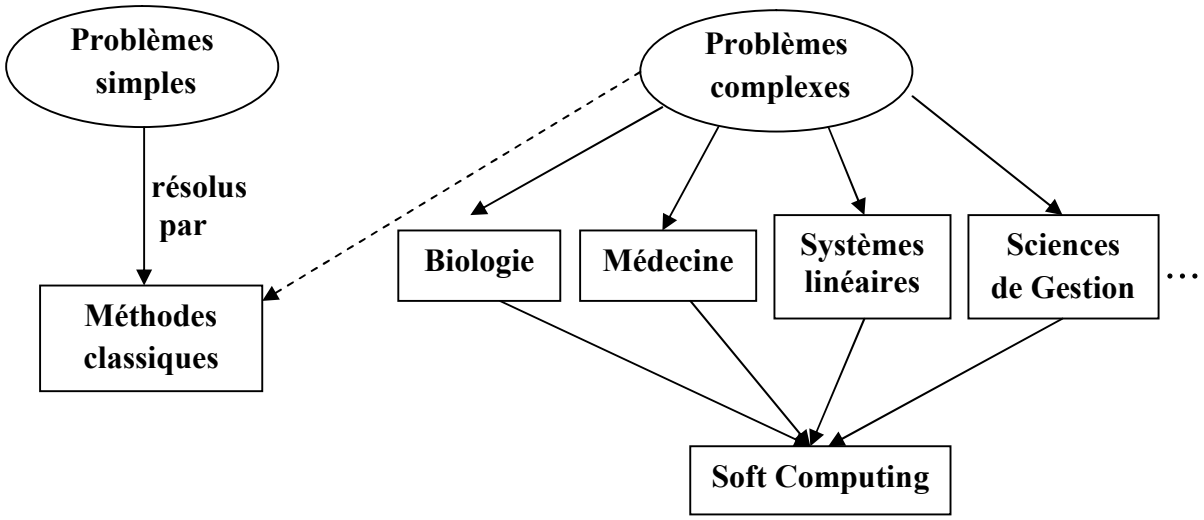
Les techniques de l'IA utilisées pratiquement en génie électrique sont nombreuses:

- Systèmes experts
- Réseaux de neurones
- Data-mining
- Algorithmes génétiques
- Programmation par contraintes
- Logique floue
- Systèmes multi-agents

CHAPITRE II: GENERALITEES SUR "SOFT COMPUTING"

I. INTRODUCTION:

Le "Soft computing" est devenu un domaine de recherche de l'informatique dans les années 1990. A l'époque, les approches du calcul informatique ne pouvaient aborder que de problèmes de simple complexité et les systèmes les plus complexes découlant de la biologie, la médecine, les sciences humaines, sciences de gestion et des domaines similaires sont souvent restés irrésolus par les méthodes classiques de mathématiques et de l'analyse.



Le soft computing a été introduit par Pr. Lotfi Zadeh en 1994 comme un noyau pour construire des systèmes intelligents.

II. DEFENITION:

Le soft computing (l'informatique douce) selon le Pr Zadeh, est une "approche émergente de l'informatique, qui est parallèle à la capacité remarquable de l'esprit humain à raisonner et à apprendre dans un environnement d'incertitude et d'imprécision".

En général, il s'agit d'une collection d'outils et de techniques informatiques, partagée par des disciplines étroitement liées qui incluent la logique floue, les réseaux de neurones (R.N) et les algorithmes génétiques (A.Gs).

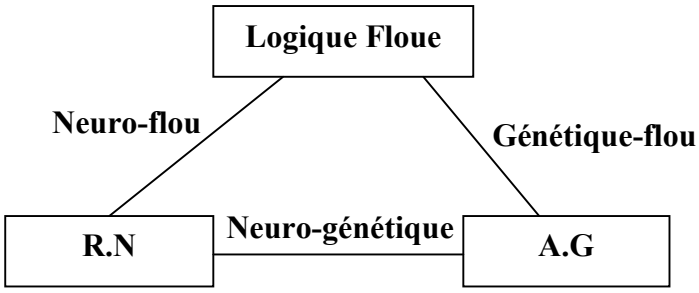


Figure (2.1): Fusion des méthodes

III. INTERET:

Leur point de départ est la gestion d'imprécision et d'incertitude inhérentes à la plupart des problèmes. Le principal intérêt du Soft Computing réside dans l'utilisation conjointe de plusieurs de ces composantes dans des systèmes hybrides tirant parti d'une synergie entre les différentes méthodes afin d'exploiter les avantages de chacune tout en compensant ses inconvénients par l'utilisation d'une autre dont les propriétés sont complémentaires.

Le Soft Computing ne se limite pas aux combinaisons mettant en jeu la logique floue et les combinaisons de réseaux de neurones et d'algorithmes génétiques ou, plus couramment de réseaux de neurones et de méthodes probabilistes, trouvent de nombreuses utilisations.

IV. COMPOSANTES ESSENTIELS:

- Systèmes intelligents flous (SIF)
- Réseau de neurones artificiels (RNA)
- Algorithmes évolutionnaires (AE) tels que: A.G.

V. DOMAINES D'APPLICATIONS:

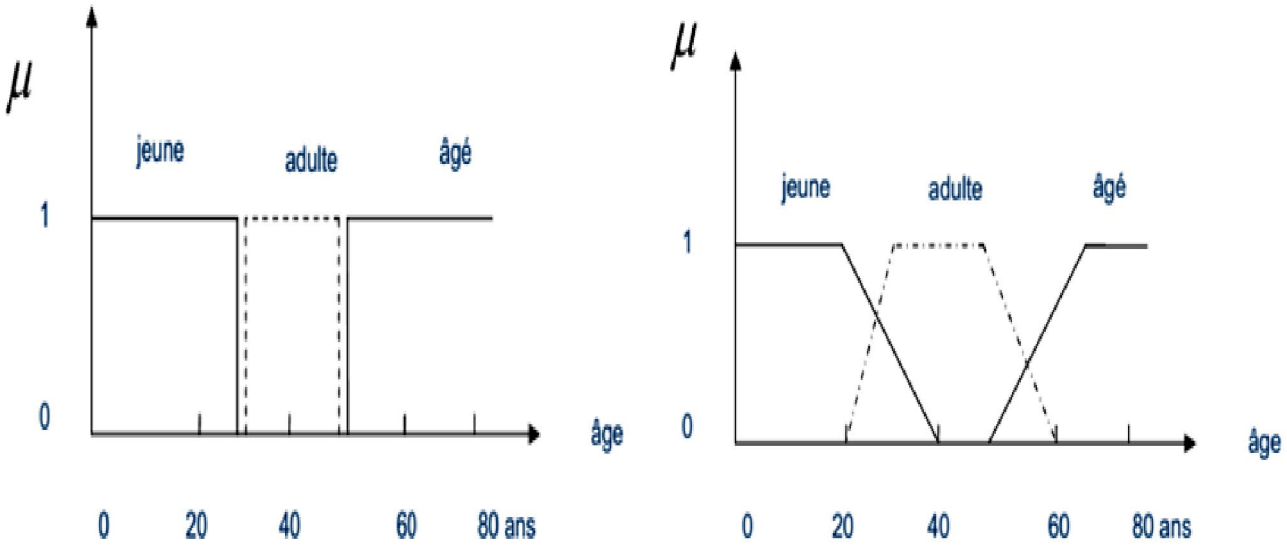
- Médecine: aide au diagnostic... .
- Télécoms: compression, routage, sécurité... .
- Industrie: supervision, contrôle de qualité... .
- Finance: prédiction... .
- Commerce: Exploitation des données... .
- Bioinformatique: authentification par analyse d'iris, d'empreintes digitales, de visages... .
- Reconnaissances de formes: parole, écriture, images,... .
- Internet: classification et recherche d'informations... .

Un système Intelligent flou (SIF): est un système qui intègre de l'expertise humaine et qui vise à automatiser le raisonnement d'experts humains face à des systèmes complexes. Un système IF se base sur la théorie logique qui est la ***logique floue***.

Logique Floue: La théorie de la logique floue est l'une des rares théories qui s'est fixée le but de copier la performance humaine. Les bases de la logique floue ont été établies en 1965 par le Pr. Lotfi Zadeh. Contrairement à la logique classique de Boole, la logique floue est un outil qui permet de manipuler des variables dites flous pouvant appartenir simultanément à des ensembles qui sont, selon la logique classique, disjoints. Cette appartenance est caractérisée par un facteur d'appartenance: un nombre en **0** et **1** donnant le degré de confiance à cette appartenance.

Exemple:

- La variable linguistique "Température" ⟶ caractérisé par deux ensembles flous: froid et chaud.
- La variable linguistique "Age" ⟶ caractérisé par trois ensembles flous: Jeune, adulte et âgé.



(a) Logique classique (de Boole) ***(b) Logique floue***

Figure (2.2): Différence entre logique classique et logique floue

22 ans → appartient à l'ensemble flou "**Jeune**" avec un facteur ou un degré d'appartenance "**μ = 0.8**" et appartient à "**Adulte**" avec un facteur d'appartenance "**μ = 0.2**".

Réseaux de neurones artificiels (RNA): est un ensemble d'algorithmes dont la conception est à l'origine très schématiquement inspirée du fonctionnement de neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processus élémentaires fonctionnant en parallèle. Chaque processus élémentaire (neurone artificiel) calcule une sortie unique sur la base des informations qu'il reçoit.

(Inspiration naturelle: analogie avec le cerveau).

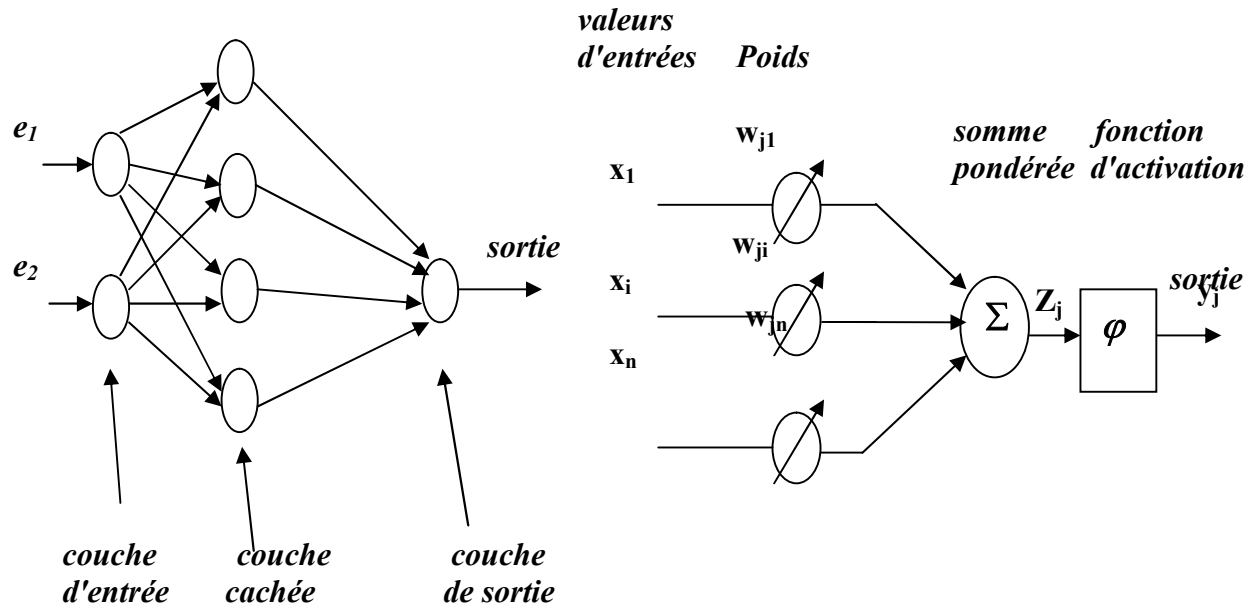


Figure (2.3): Réseau de neurones artificiel et Neurone artificiel

- Réseau **MLP** (réseau multicouches) (Multilayer perceptron)
- Réseau **RBF** (fonction de base radiale) (Radial basis function)

Les algorithmes évolutionnaires: sont une famille d'algorithmes dont le principe s'inspire de la théorie de l'évolution pour résoudre des problèmes divers. L'idée est de faire évoluer un ensemble de solutions à un problème donné, pour trouver les meilleurs résultats. Ces sont des algorithmes dits stochastiques, car ils utilisent itérativement des processus aléatoires.

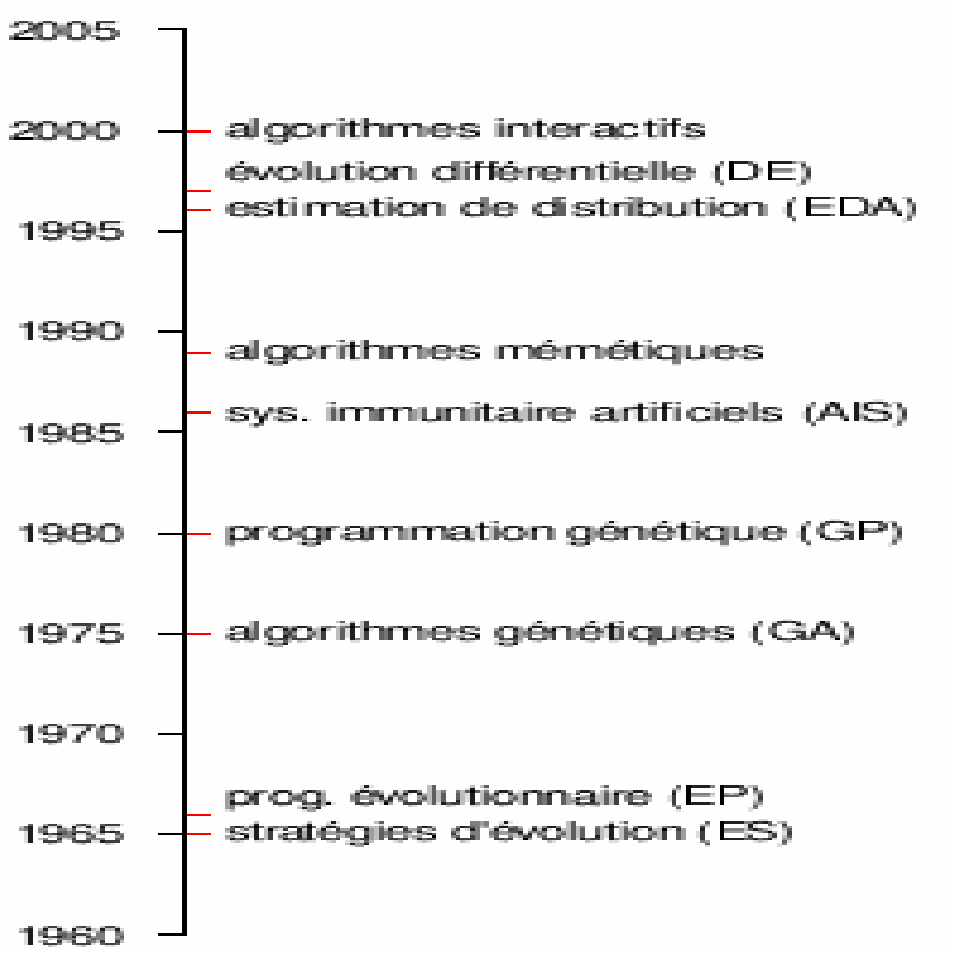


Figure (2.4): Chronologie des principales

Les algorithmes génétiques: L'algorithme génétique (AG) est un algorithme de recherche basé sur les mécanismes de la sélection naturelle et de la génétique. Il combine une stratégie "survie des plus forts" avec un échange d'information aléatoire mais structuré. L'AG fut développé par J.Holland.

L'AG est un algorithme d'optimisation stochastique basé sur la stratégie d'évolution naturelle de Darwin.

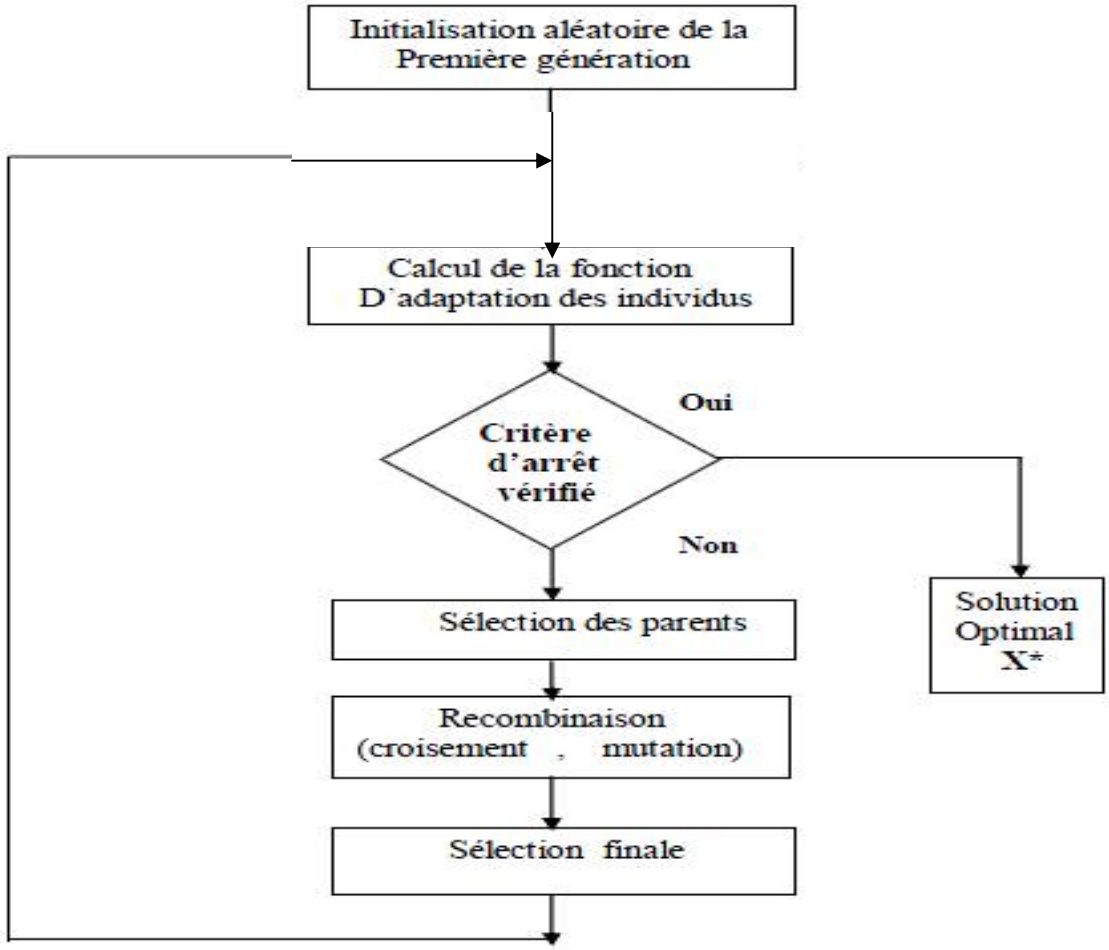


Figure (2.5): Organigramme d'un A.G

VI. COMPARAISON ENTRE SOFT COMPUTING ET HARD COMPUTING:

L'informatique douce et l'informatique dure sont des méthodes informatiques, l'informatique dure étant la méthodologie conventionnelle qui repose sur les principes de précision, de certitude et de rigidité. À l'inverse, l'informatique douce est une approche moderne fondée sur l'idée de l'approximation, de l'incertitude et de la flexibilité.

1. Informatique dure:

Si le résultat est précis et l'action de contrôle est sans ambiguïté et doit être formellement définie (avec un modèle mathématique ou un algorithme), alors ce type d'informatique est appelé ***Informatique dure***.

Caractéristiques du calcul dur:

- Un résultat précis est garanti
- L'action de contrôle est sans ambiguïté

- L'action de contrôle est formellement définie (c'est-à-dire un modèle mathématique ou un algorithme)

2. Informatique douce

Le Soft Computing est un ensemble de méthodologies qui visent à exploiter la tolérance à l'incertitude pour atteindre la traçabilité, la robustesse et un faible coût de solution.

Caractéristiques du soft computing

- Solution d'imprécision
- Nature adaptative/dynamique
- Incertitude
- Faible coût de solution
- Ne nécessite pas de modélisation mathématique
- Il utilise certaines méthodologies d'inspiration biologique telles que des algorithmes génétiques et des algorithmes évolutifs.

Le tableau suivant donne une comparaison entre le "soft computing" et le "hard computing":

Tableau (2.1): Comparaison entre "Soft Computing" et "Hard Computing"

Base de Comparaison	Soft Computing	Hard Computing
Base	Tolérant à l'imprécision, à la vérité partielle et à l'approximation	Utilise un modèle analytique énoncé avec précision
Basé sur	La logique floue et raisonnement probabiliste	Logique binaire et système net
Caractéristiques	Approximation	Précision
Nature	Stochastique	Déterministe
Travaille sur	Données ambiguës	Données Exactes
Calcul	Parallèle	Séquentiel
Résultat	Approximatif	Précis

CHAPITRE III : LA LOGIQUE FLOUE ET SES APPLICATIONS

I. INTRODUCTION :

La logique floue (en anglais : *Fuzzy Logic*) est une extension de la logique booléenne créée par le professeur Lotfi Zadeh en 1965 en se basant sur sa théorie mathématique des ensembles flous, qui est une généralisation de la théorie des ensembles classiques. En introduisant la notion de degré dans la vérification d'une condition, permettant ainsi à une condition d'être dans un autre état que vrai ou faux, la logique floue confère une flexibilité très appréciable aux raisonnements qui l'utilisent, ce qui rend possible la prise en compte des imprécisions et des incertitudes.

Un des intérêts de la logique floue pour formaliser le raisonnement humain est que les règles sont énoncées en langage naturel.

II. HISTORIQUE :

- En 1965, Pr. L. Zadeh a introduit le concept d'ensemble flou.
- En 1970, les premières applications de la logique floue : systèmes experts, aide à la décision en médecine, commerce,… .
- En 1974, M. Mamdani expérimentait la théorie énoncée par Zadeh sur une chaudière à vapeur, matériel dont on connait la complexité, introduisant ainsi la commande floue dans la régulation d'un processus industriel. Plusieurs applications ont alors vu le jour en Europe, pour des systèmes parfois très complexes, telle la régulation de fours de cimenterie réalisée par la société F. L. Smidt-Fuller.
- Grace au chercheur japonais M. Sugeno, la logique floue était introduite au Japon dés 1985. Les sociétés japonaises comprirent l'avantage à la fois technique et commercial de la logique floue.
- En 1990, généralisation de l'utilisation de cette technique. l'industrie japonaise a développé des produits grand public à base de logique floue : appareils électroménagers (lave-linge, aspirateurs, autocuiseurs,...), systèmes audio-visuels (appareils photos autofocus, caméscope à stabilisateur d'images, ...), systèmes automobiles embarqués, systèmes autonomes mobiles, systèmes de transport (train, métro, ascenseur,...), systèmes de conditionnement d'ambiance, systèmes de décision, diagnostic, reconnaissance.

III. AVANTAGES ET INCONVENIENTS :

Avantages:

- facilité d'implantation ;
- solution de problèmes multivariables complexes ;
- robustesse vis à vis des incertitudes ;
- possibilité d'intégration du savoir de l'expert.
- Possibilités de commande auto-adaptative aux variations du procédé.

Inconvénients:

- Réglage empirique.
- Performances en boucle fermée dépendent de l'expertise.
- Il n'existe pas de théorie générale qui caractérise rigoureusement la stabilité, la robustesse.

IV. Différence entre la logique classique et logique floue :

La logique booléenne classique ne permet que deux états : VRAI ou FAUX. La logique floue permet d'exprimer différents niveaux, plutôt que seulement 1 ou 0. Par exemple : le moteur est chaud, le moteur est très chaud. Quelle est la différence entre « chaud » et « très chaud » ? Ou encore, un homme est haut s'il mesure 170cm. Un homme est très haut s'il mesure 190cm. Où est la ligne de démarcation ? Un homme de 180cm est-il haut ou très haut ? 180.5cm ? 179.5cm ?

La logique floue est une branche des mathématiques qui permet à un ordinateur de modéliser le monde réel de la même façon que les personnes. Elle est préoccupée par la quantification et le raisonnement en utilisant un langage qui permet des définitions ambiguë, comme ***beaucoup***, ***peu***, ***petit***, ***haut***, ***dangereux***. Elle s'occupe de situations où la question qui est posée et la réponse obtenue contiennent des concepts vagues.

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.

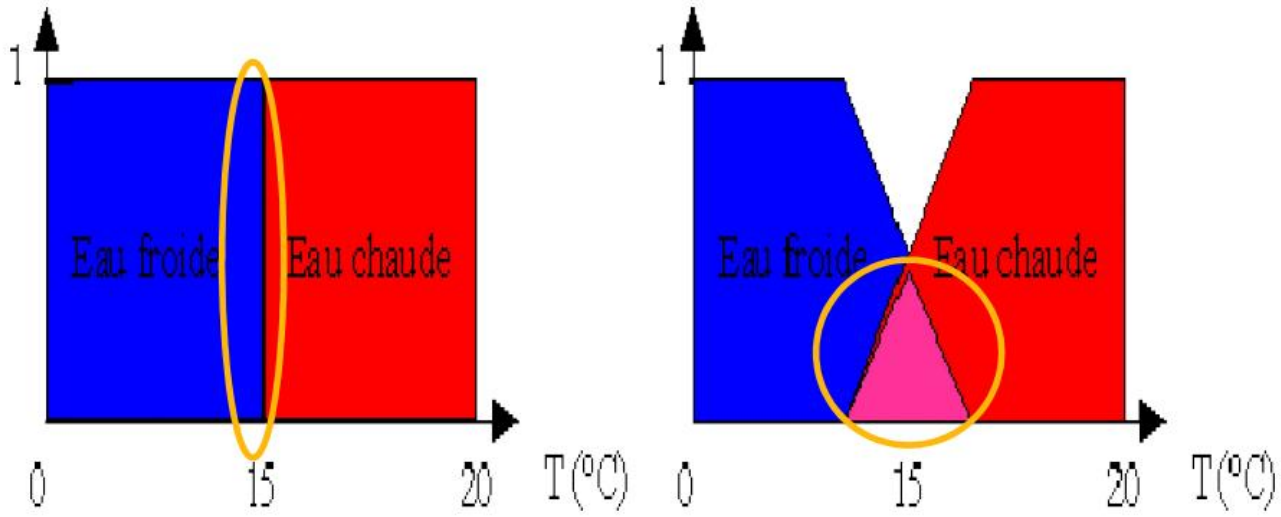


Figure (3.1) : Logique classique et logique floue

L'exemple de la frontière entre eau chaude et eau froide nous montre l'aspect artificiel de la logique classique puisque là où elle impose une frontière brutale toute théorique entre les 2 états de l'eau, on voit apparaître avec la logique floue un passage progressif d'un état à l'autre en introduisant de ce fait une zone tiède.

Exemple :

Selon cette figure, 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 degré d'appartenance de 0.167, et appartient au groupe "élevée" avec un degré d'appartenance de 0.75, (et au groupe "faible" avec un degré d'appartenance de 0).

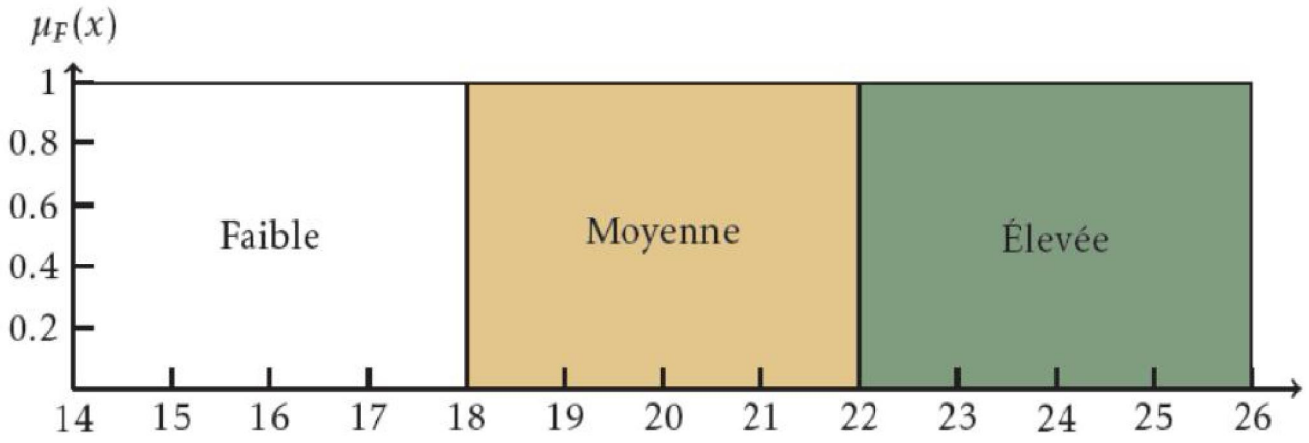


Figure (3.2): Représentation Classique

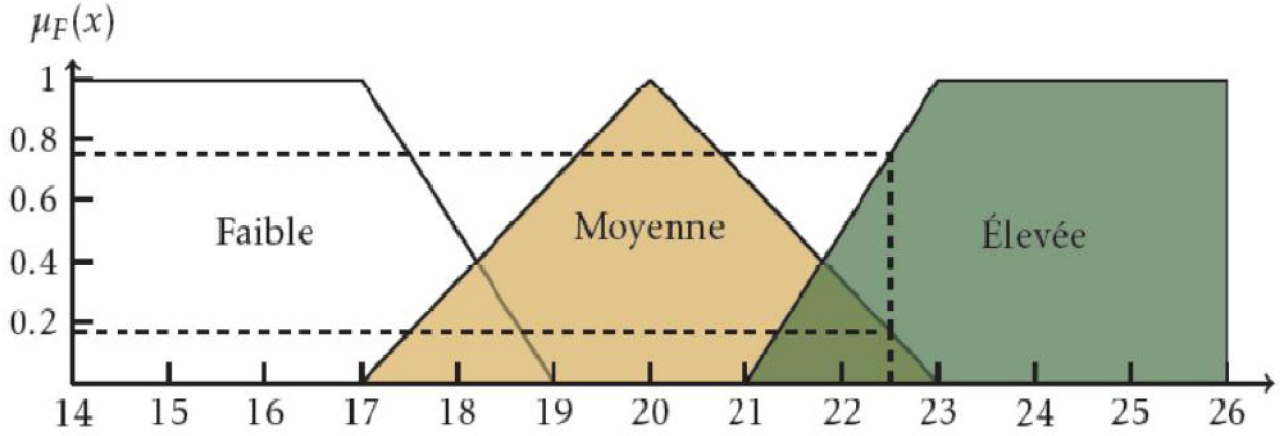


Figure (3.3): Représentation Floue

Avec : μ la fonction d'appartenance.

Les variables floues : ***faible, moyenne*** et **élevée** sont représentées par des fonctions linéaires. D'autres fonction auraient pu être utilisées, comme des trapézoïdes, des paraboles, etc. Cependant, les fonction linéaires sont beaucoup plus faciles à implémenter de façon pratique, et donnent de bons résultats.

V. ENSEMBLES FLOUES ET OPERATEURS FLOUS:

Sous-ensemble flou :

Dans la théorie des ensembles classiques, il n'y a que deux situations acceptables pour un élément, appartenir ou ne pas appartenir à un sous-ensemble. Le mérite de Zadeh a été de tenter de sortir de cette logique booléenne en introduisant la notion d'appartenance pondérée : permettre des graduations dans l'appartenance d'un élément à un sous-ensemble, c'est-à-dire d'autoriser un élément à appartenir plus moins fortement à ce sous-ensemble.

Soit X un ensemble de référence et soit x un élément quelconque de X. Un sous-ensemble flou A de X est défini comme l'ensemble des couples :

$$A = \{(x, \mu_A(x)), x \in X\}, avec: \mu_A: X \rightarrow [0, 1]$$

Ainsi, un sous-ensemble flou A de X est caractérisé par une fonction d'appartenance $\mu_A(x)$ qui associe, à chaque point x de X un réel dans l'intervalle *[0, 1]* ; $\mu_A(x)$ représente le degré d'appartenance de x à A. On observe les trois cas possibles suivants :

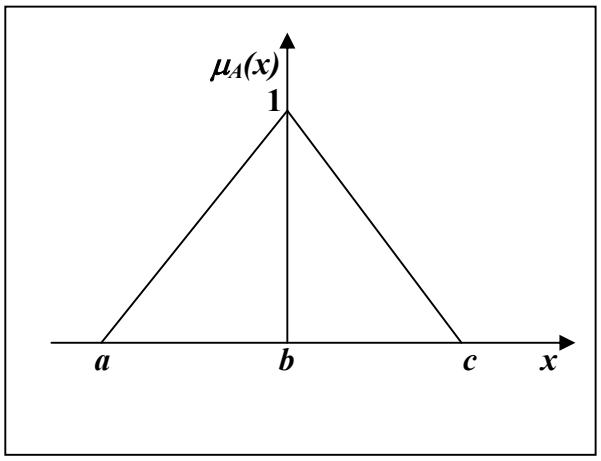
$$\begin{cases} \mu_A(x) = 0 & (x\ n'appartient pas\ \text{à}\ A) \\ 0 < \mu_A(x) < 1 & (x\ appartient\ partiellement\ \text{à}\ A) \\ \mu_A(x) = 1 & (x\ appartient\ entièrement\ \text{à}\ A) \end{cases}$$

Les Fonctions d'appartenances:

Il existe plusieurs formes de fonctions d'appartenances dont les plus utilisées sont: les formes trapézoïdales et triangulaires.

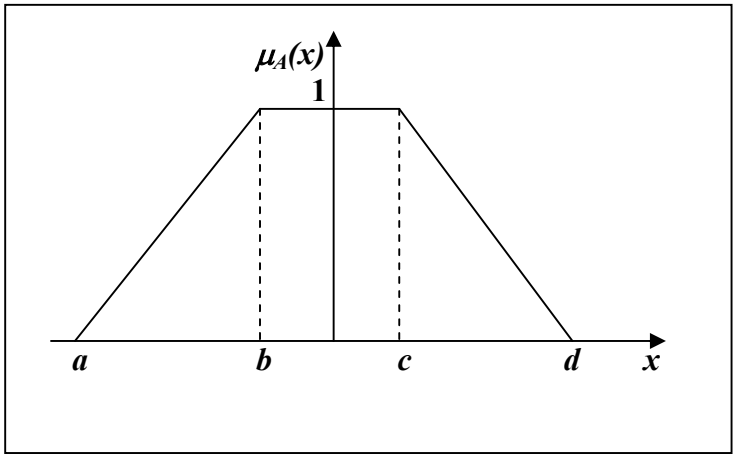
- ***Triangulaire:***

$$\mu_A(x) = \begin{cases} \dfrac{x-a}{b-a} & si \quad x \in [a,b] \\ \dfrac{c-x}{c-b} & si \quad x \in [b,c] \end{cases}$$



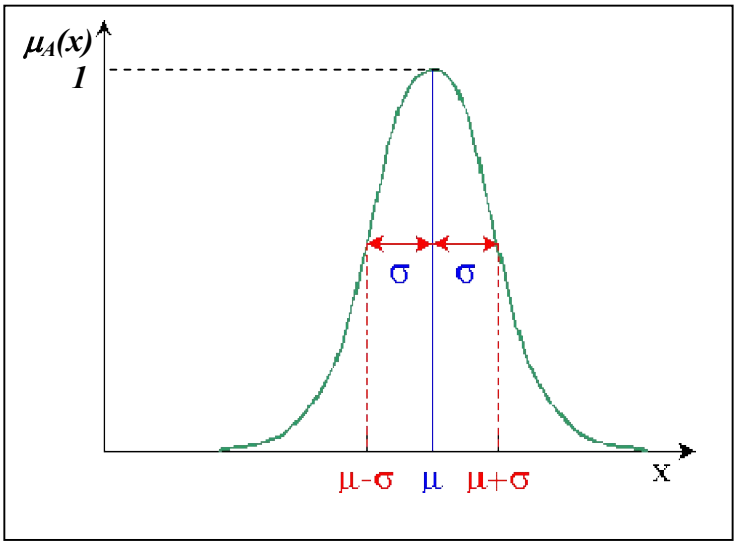
- ***Trapèze :***

$$\mu_A(x) = \begin{cases} \dfrac{x-a}{b-a} & si \quad x \in [a,b] \\ 1 \quad si \quad x \in [b,c] \\ \dfrac{d-x}{d-c} & si \quad x \in [c,d] \end{cases}$$



- ***Gaussienne :*** c'est une fonction en cloche, caractérisée par les deux paramètres: ***m*** et **σ**.

$$\mu_A(x) = exp\left(-\frac{(x-m)^2}{2\sigma^2}\right)$$



Caractéristiques des sous-ensembles flous:

Les fonctions d'appartenances décrivant des sous-ensembles flous sont caractérisées par les quatre propriétés suivantes:

- ***Hauteur (H(A)):*** C'est le plus fort degré avec lequel un élément de ***X*** appartient à ***A*** où :

$$\boldsymbol{H(A) = sup_{x \in X}(\mu_A(x)) = \{x \in A | \mu_A(x) \neq 0\}} \quad \textbf{(3.1)}$$

- ***Noyau (N(A)):*** C'est l'ensemble des points qui appartiennent intégralement à A :

$$\boldsymbol{N(A) = \{x | \mu_A(x) = 1\}} \quad \textbf{(3.2)}$$

- ***Support (S(A)):*** Cet ensemble décrit l'ensemble des éléments qui sont partiellement dans A:

$$\boldsymbol{S(A) = \{x | \mu_A(x) \neq 1\}} \quad \textbf{(3.3)}$$

Operateurs flous :

Les opérateurs flous décrivent comment des ensembles flous interagissent ensembles. On regardera certaines opérations communes, comme le complément, l'intersection et l'union. Ces opérateurs sont définis comme suit:

- ***Le complément*** permet de vérifier de combien un élément n'appartient pas à un ensemble. Comme exemple, si on a l'ensemble des températures élevées, le complément est l'ensemble des températures qui ne sont pas élevées. Si ***A*** est l'ensemble flou, son complément ¬ ***A*** est :

$$\boldsymbol{\mu_{\neg A}(x) = 1 - \mu_A(x)} \quad \textbf{(3.4)}$$

- ***L'intersection*** de deux ensembles, en logique flou, est un peu différente des méthodes classiques. On cherche à savoir de combien un élément est-il dans les deux ensembles. On utilise alors la valeur minimale d'appartenance pour calculer l'intersection.

$$\boldsymbol{\mu_{A \cap B}(x) = min\,[\mu_A(x), \mu_B(x)]} \quad \textbf{(3.5)}$$

- En logique floue, ***l'union*** est le contraire de l'intersection. On cherche à savoir de combien un est-il dans l'un des deux ensembles. On utilise la valeur maximale d'appartenance.

$$\boldsymbol{\mu_{A \cup B}(x) = max\,[\mu_A(x), \mu_B(x)]} \quad \textbf{(3.6)}$$

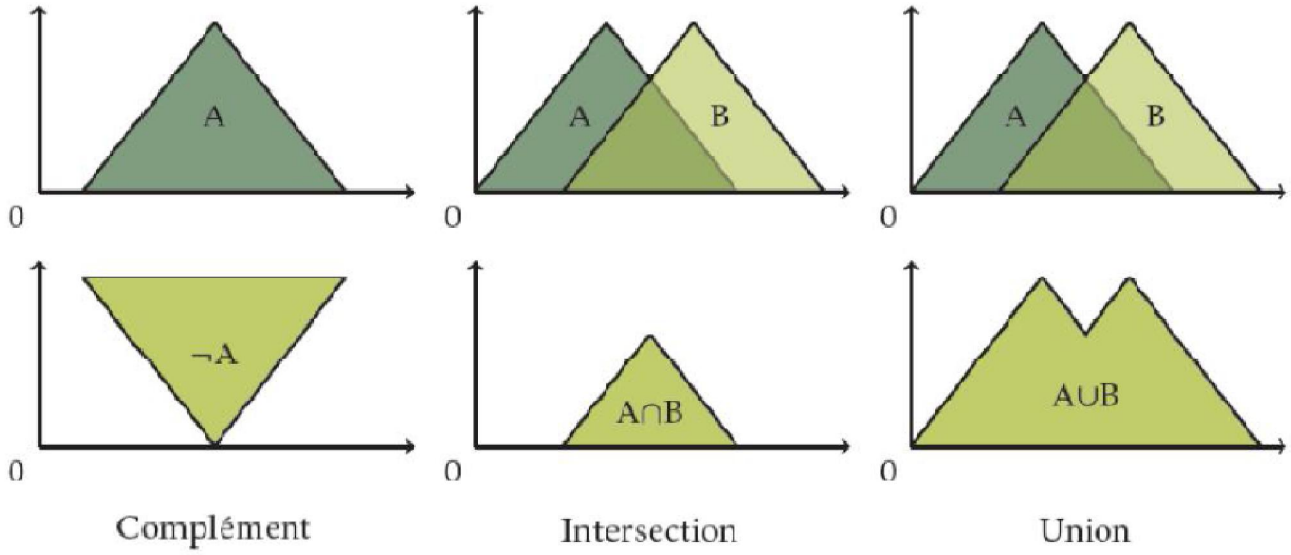


Figure (3.4): Exemple d'opérations sur des ensembles flous

On définit l'intersection et l'union à l'aide de deux relations floues, la t-norme et la t-conorme. Le tableau ci-dessous donne la définition des différents opérateurs par les deux approches: Zadeh Min/Max et probabiliste Prod/Probor.

Dénomination	Intersection ET (t-norm)
Opérateurs de Zadeh MIN/MAX	$\mu_{A\cap B}(x) = min(\mu_A(x),\mu$
Probabiliste PROD/PROBOR	$\mu_{A\cap B}(x) = \mu_A(x) \times \mu_$

VI. SYSTEMES FLOUS:

Un système flou est formé de trois étapes:

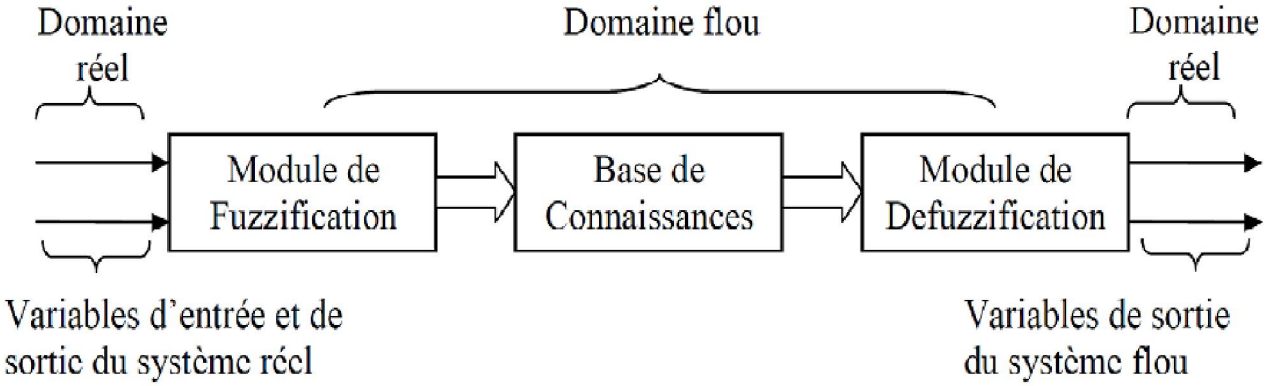


Figure (3.5): structure d'un système flou

- ***Fuzzification:*** est l'opération de rendre une entrée classique (valeur numérique) 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.

- ***Le module d'inférence:*** qui est constitué de deux blocs, le moteur d'inférence et la base des règles.

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 é*levée*. La plage de valeurs linguistiques possibles d'une règle représente l'univers de cette variable. Un exemple de régle 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''a 90% en utilisant la logique floue.

- ***Défuzzification:*** permet d'inférer une valeur nette (précise), utilisable en commande par exemple, à partir du résultat de l'agrégation des règles. Donc, La défuzzification est le processus de convertir une valeur floue en valeur nette.

Quelques méthodes existent, comme l'appartenance maximale, la méthode du centroïde, et la méthode des moyennes pondérées.

1) Inférence Mamdani:

L'inférence Mamdani procède selon quatre étapes :

1. Fuzzification des variables d'entrée

2. Evaluation des règles

3. Agrégation des sorties des règles

4. Défuzzification

Exemple N°1:

Un système de contrôle d'un ventilateur de maison, ayant 2 entrées (température et humidité) et une sortie (vitesse du ventilateur). Des règles possibles sont :

SI x **est** A_1 **OU** y **est** B_1 **ALORS** z **est** C_1

SI x **est** A_2 **ET** y **est** B_2 **ALORS** z **est** C_2

SI x **est** A_3 **ALORS** z **est** C_3

Où x est la température, y est l'humidité et z est la vitesse du ventilateur. Les ensembles flous sont :

- A_1 = **faible,** A_2 = **moyenne,** A_3 = **élevée**
- B_1 = **sec,** B_2 = **humide**
- C_1 = **lent,** C_2 = **moyenne,** C_3 = **rapide**

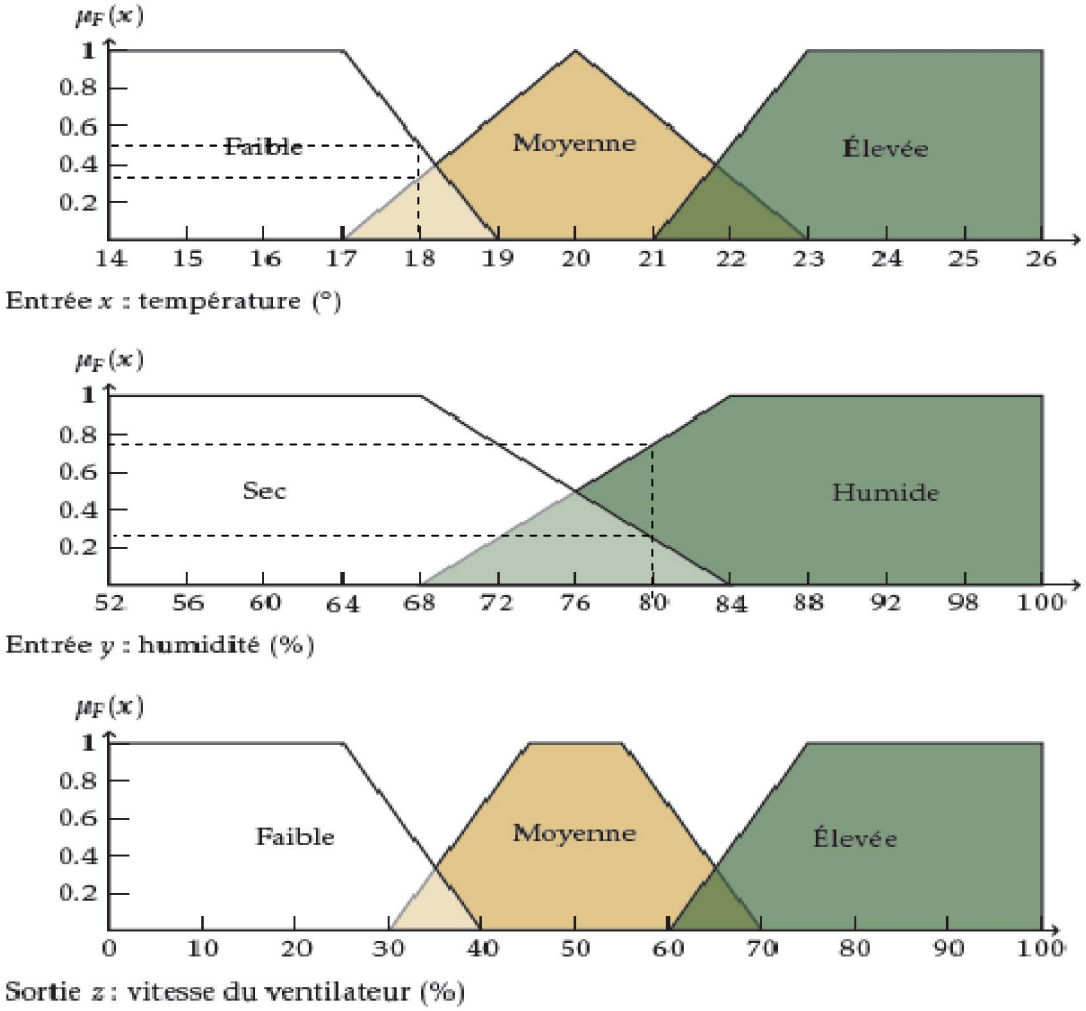


Figure (3.6): Ensembles flous pour exemple d'inférence de Mamdani

- Une ***température*** de ***18°C*** correspond à une appartenance de ***0.5*** à l'ensemble ***faible*** ($\mu_{A_1} = 0.5$) et une appartenance de **0.33** à l'ensemble ***moyenne*** ($\mu_{A_2} = 0.33$).
- Une ***humidité*** de ***80%*** correspond à une appartenance de ***0.25*** à l'ensemble ***sec*** ($\mu_{B_1} = 0.25$) et une appartenance de ***0.75*** à l'ensemble ***humide*** ($\mu_{B_2} = 0.75$).

Evaluation des règles :

Il faut maintenant évaluer les règles en fonction des entrées obtenues. On applique les opérateurs flous correspondants pour combiner les règles.

Si on reprend l'exemple précédent, on applique les valeurs d'appartenance à la règle 1 :

SI x est A_1 (0.5) OU y est B_1 (0.25) ALORS z est C_1 (?)

On peut utiliser deux méthodes pour combiner ces antécédents :

1. Méthode 1: maximum $\mu_{C_1} = max[\mu_{A_1}; \mu_{B_1}] = 0.5$

2. Méthode 2: probor $\mu_{C_1} = probor[\mu_{A_1}; \mu_{B_1}] = (\mu_{A_1} + \mu_{B_1}) - (\mu_{A_1} . \mu_{B_1})$

$= (0.5 + 0.25) - (0.5 * 0.25) = 0.625$

Pour la règle 2 :

SI x est A_2 (0.33) ET y est B_2 (0.75) ALORS z est C_2 (?)

On peut utiliser deux méthodes pour combiner ces antécédents :

1. Méthode 1: minimum $\mu_{C_1} = min[\mu_{A_2}; \mu_{B_2}] = 0.33$

2. Méthode 2: prod $\mu_{C_1} = prod[\mu_{A_2}; \mu_{B_2}] = \mu_{A_2} . \mu_{B_2} = (0.33)(0.75) = 0.25$

Au total, on obtient :

SI x est A_1 (0.5) OU y est B_1 (0.25) ALORS z est C_1 (0.5)

SI x est A_2 (0.33) ET y est B_2 (0.75) ALORS z est C_2 (0.33)

SI x est A_3 (0.0) ALORS z est C_3 (0.0)

Inférence:

Si on utilise la méthode de coupure pour combiner les règles, il faut créer un nouveau polygone à partir des trois fonctions d'appartenance de la conséquence ***C***. La hauteur du polygone est déterminée à partir de la valeur d'appartenance calculée plus haut.

L'agrégation des sorties des règles pour cet exemple est montrée sur la figure suivante :

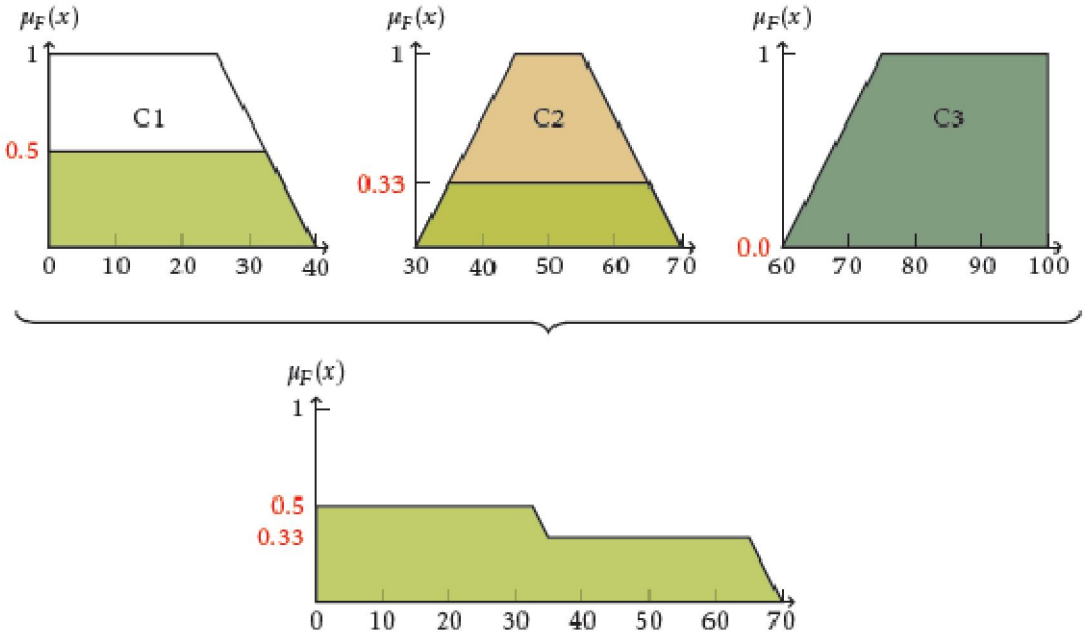


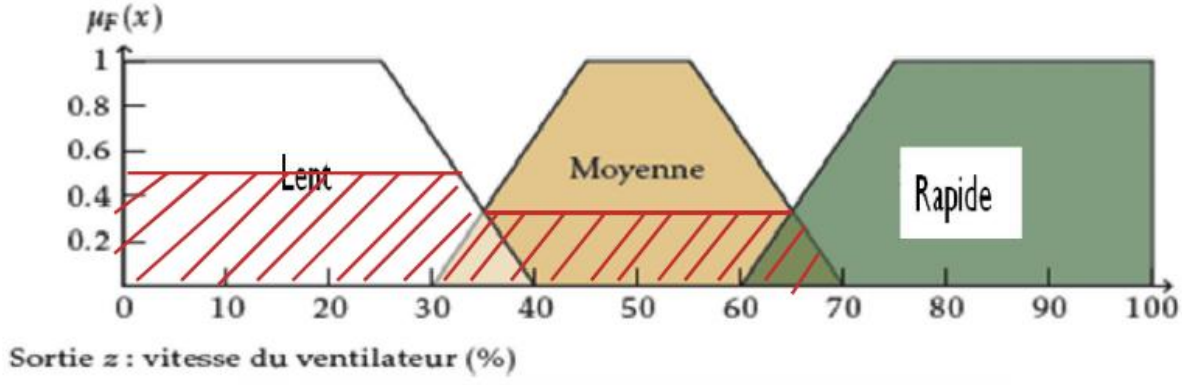
Figure (3.7): Agrégation des règles

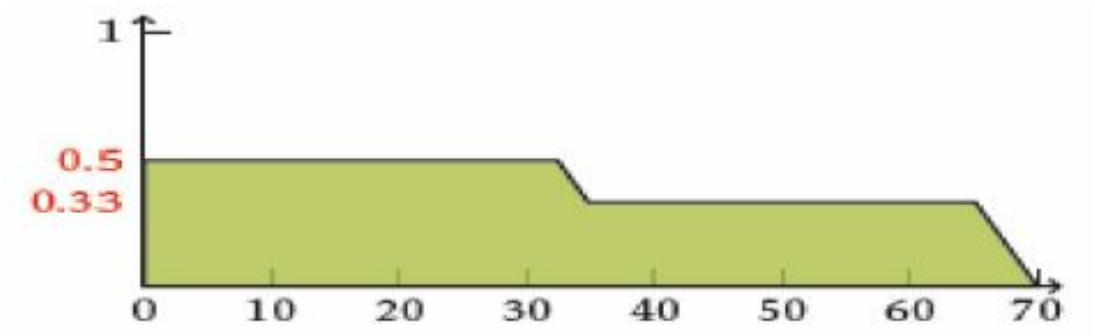
Défuzzification :

Après avoir combiné les règles, il faut maintenant produire un chiffre net comme sortie (la valeur réelle). Dans ce cas-ci, la sortie doit être la vitesse du ventilateur. La technique la plus populaire est la méthode du Centroïde: on cherche le centre de gravité du polygone obtenu. On peut exprimer le centre de gravité par la formule suivante;

$$C.G = \frac{\sum_{x=a}^{b} \mu_A(x).x}{\sum_{x=a}^{b} \mu_A(x)} \qquad (3.7)$$

Le polygone obtenu pour l'exemple proposé est détaillé par la figure ci-dessous





Le centre de gravité n'a pas besoin d'être calculé d'une façon très précise. On peut approximer, en calculant à tous les 10, par exemple. Pour l'exemple précédent :

$$C.G = \frac{(0+10+20+30).0.5+(40+50+60).0.33}{0.5+0.5+0.5+0.5+0.33+0.33+0.33} = 26.59$$

Donc, on peut conclure que le ventilateur doit être à ***26.67%*** de sa vitesse maximale.

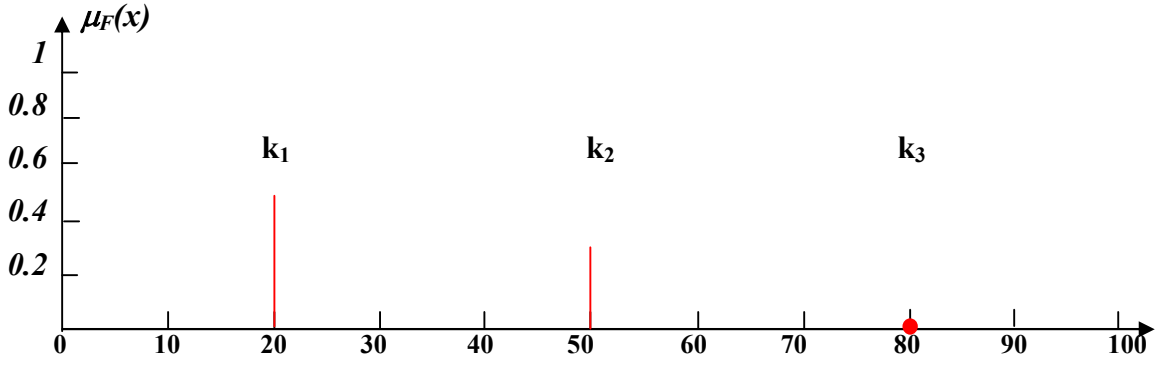
2) <u>Inférence Sugeno:</u>

La méthode d'inférence de Sugeno diffère de celle de Mamdani sur la partie défuzzfication. Sugeno propose de remplacer la défuzzification des variables de sortie par une combinaison linéaire des variables d'entrée (le plus souvent, cette combinaison linéaire se réduit à une constante « $Z_1 = k$ »)

Donc, la méthode de Sugeno permet de simplifier le calcul de l'agrégation, afin d'obtenir plus rapidement une solution nette. Cette méthode est souvent utilisée pour des applications à temps réel, où le temps de calcul est important.

La méthode de Sugeno utilise un seul pic comme fonction d'appartenance, plutôt qu'un polygone.

Les conséquences sont évaluées selon le résultat de l'évaluation des règles. Dans l'exemple précédent, on a obtenu une appartenance de 0.5 pour la règle 1, de 0.33 pour la règle 2, et 0 pour la règle 3. Ces trois valeurs deviennent alors l'amplitude des pics:



Pour faire l'agrégation des règles, on utilise une moyenne pondérée :

$$\boldsymbol{CG = \frac{\sum_i \mu_A(k_i).k_i}{\sum_i \mu_A(k_i)}} \qquad (\mathbf{3.8})$$

Selon l'exemple précédent, on obtient :

$$\boldsymbol{CG_{Sugeno} = \frac{(0.5).20 + (0.33).50 + 0}{0.5 + 0.33} = 32\%}$$

On obtient une vitesse de ventilateur de **32%**, ce qui est quand même prés de la valeur obtenue par l'autre méthode.

Plusieurs approches sont proposées pour le traitement numérique des règles d'inférences à savoir:

- **Méthode d'inférence Max-Min**
- **Méthode d'inférence Max-produit.**
- **Méthode d'inférence somme-produit**

Le choix de la méthode dépend de l'utilisateur et du cas à traiter. Dans notre exemple, on a adopté pour la méthode d'inférence max-min (méthode de **Mamdani**).

Méthode d'inférence somme-produit: On réalise au niveau de la condition, l'opérateur OU par la formation de la somme (valeur moyenne) des degrés d'appartenance intervenant dans l'inférence, et l'opérateur ET par la formation du produit. Pour la conclusion, l'opérateur ALORS est réalisé par un produit.

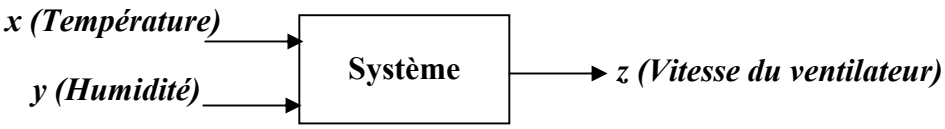
Méthode d'inférence max-produit: L'opérateur ET est réalisé par la fonction "Min", l'opérateur OU est réalisé par la formation du maximum, et ALORS (l'implication) est réalisée par la formation du produit.

La différence par rapport à la méthode précédente réside dans la manière de réaliser la conclusion "ALORS". Dans ce cas, on utilise le produit. Les fonctions d'appartenances partielles ont la même forme que la fonction d'appartenance dont elles sont issues multipliées par un facteur d'échelle vertical qui correspond au degré d'appartenance obtenu à travers l'opérateur "ET". On l'appelle également "implication de Larsen".

Méthode d'inférence max-min: Cette méthode réalise l'opérateur "ET" par la fonction "Min", la conclusion "ALORS" de chaque règle par la fonction "Min" et la liaison entre toutes les règles (opérateur "OU") par la fonction Max. La dénomination de cette méthode, dite Max-Min ou "implication de Mamdani", est due à la façon de réaliser les opérateurs ALORS et OU de l'inférence.

Solution de l'exemple par simulation sous Matlab:

Pour l'exemple précédent, on a un système à deux entrées (***x***: température, **y**: humidité) et une seule sortie (*z*: la vitesse du ventilateur).



On utilise l'instruction "Fuzzy" pour construire le système flou correspondant à l'exemple traité précédemment.

Pour cela, on tape "***fuzzy"*** à la ligne de commande.

>> Fuzzy

On obtient alors le ***FIS Editor*** de ***Matlab***, montré sur la figure suivante:

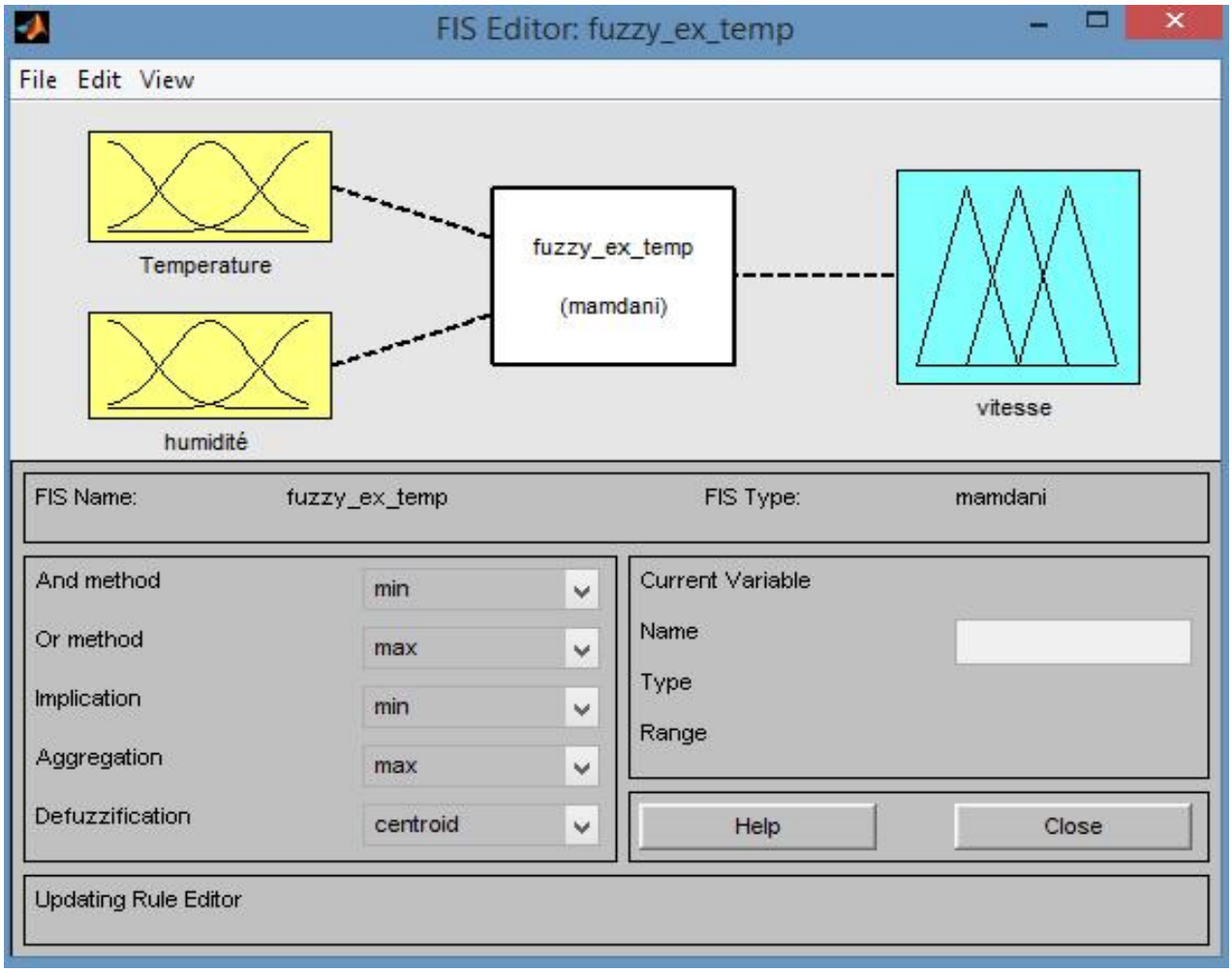


Figure (3.8): FIS Editor

Fonctions d'appartenances de la première entrée:

Les fonctions d'appartenances de la première entrée "Température" sont choisies comme le montre la figure ci-dessous, avec un univers de discours [14, 26].

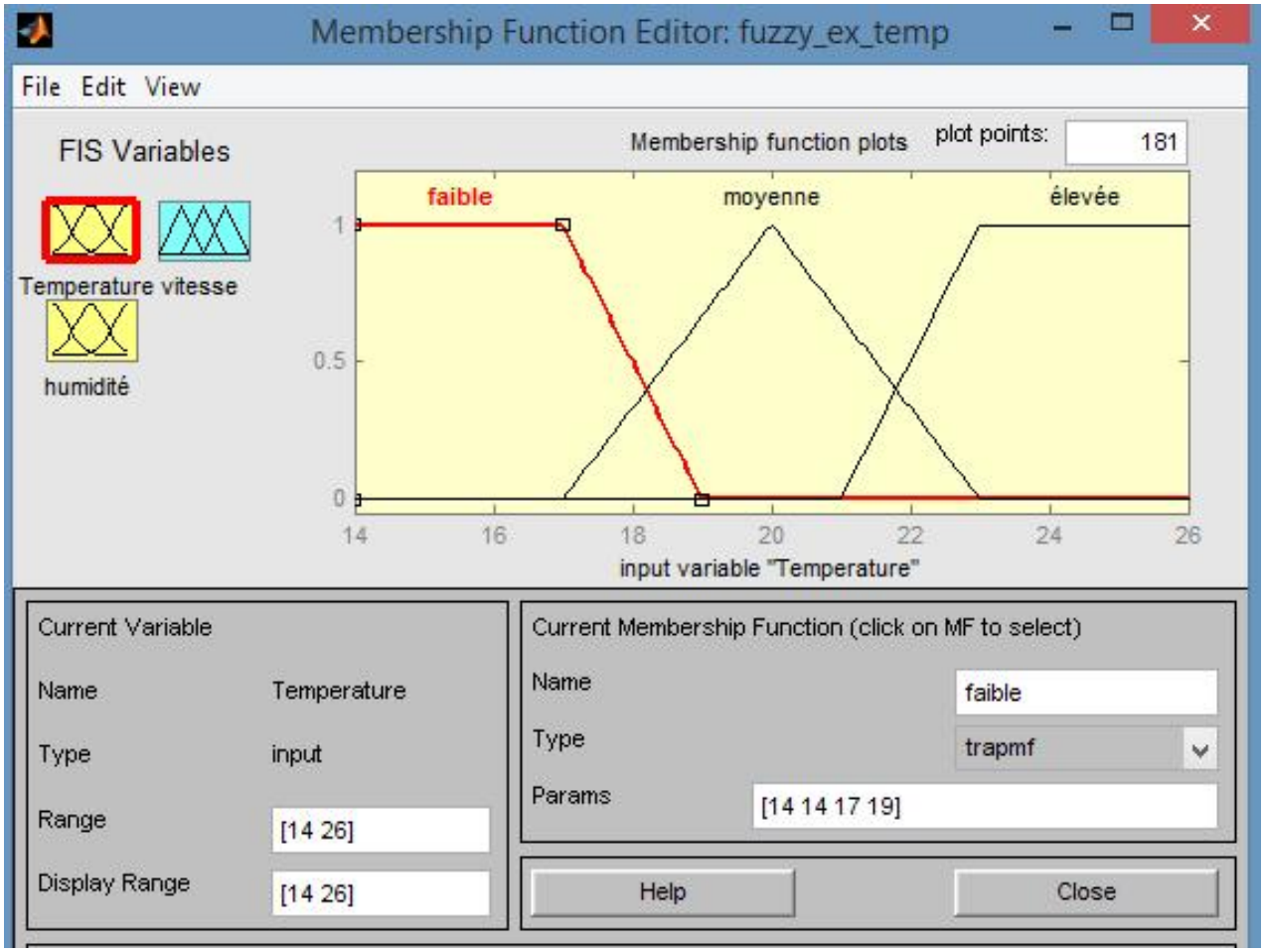


Figure (3.9): Fonctions d'appartenances de la 1^{ere} entrée générées par FIS

Fonctions d'appartenances de la deuxième entrée:

Les fonctions d'appartenances de la deuxième entrée "humidité" sont choisies comme le montre la figure ci-dessous, avec un univers de discours [52, 100].

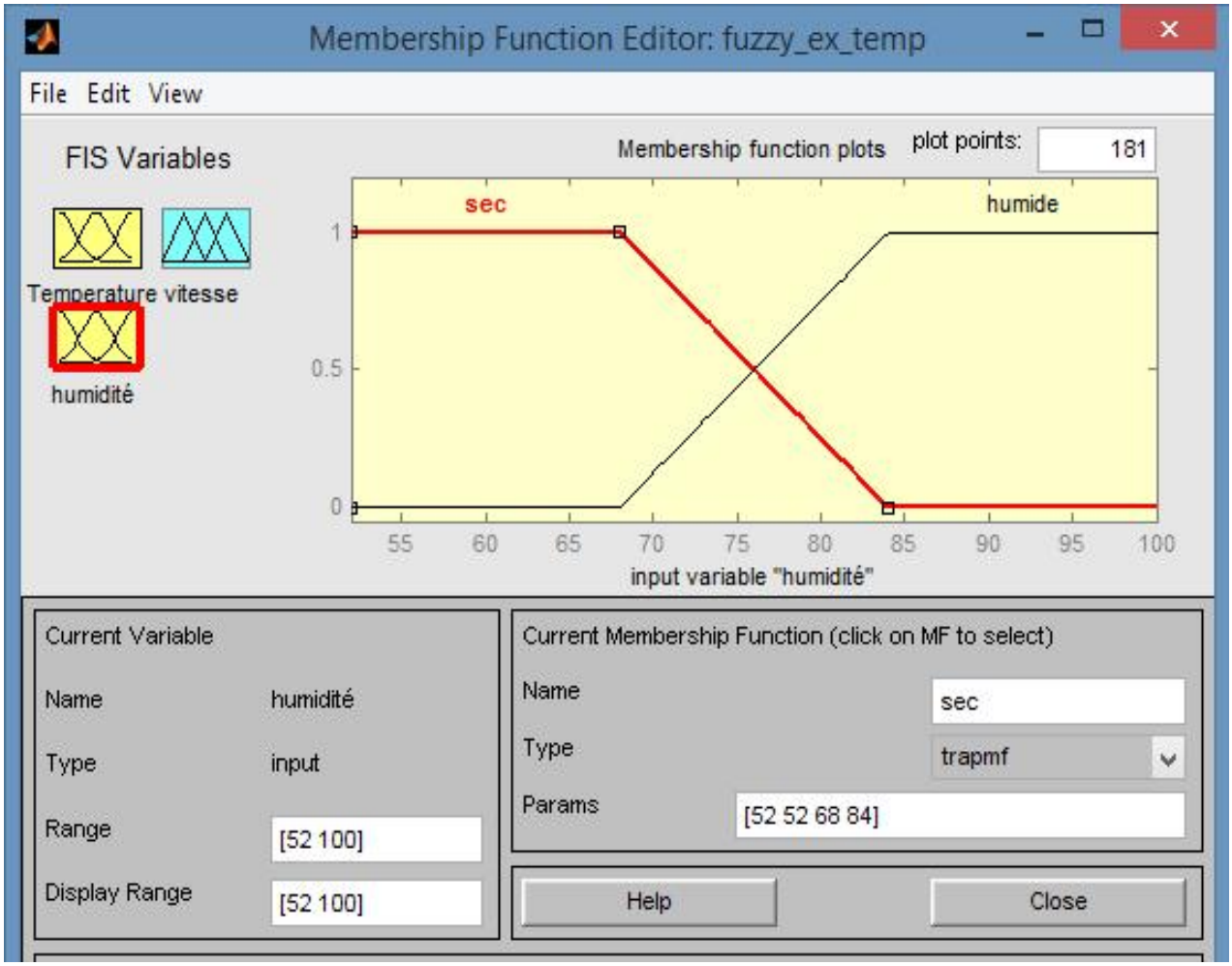


Figure (3.10): Fonctions d'appartenances de la 2ème entrée générées par FIS

Fonctions d'appartenances de la sortie:

Les fonctions d'appartenances de la sortie "vitesse" sont choisies de forme trapèze comme le montre la figure ci-dessous, avec un univers de discours normalisé entre [0, 100].

La vitesse réelle obtenue après défuzzification est un pourcentage par rapport à la vitesse nominale du ventilateur.

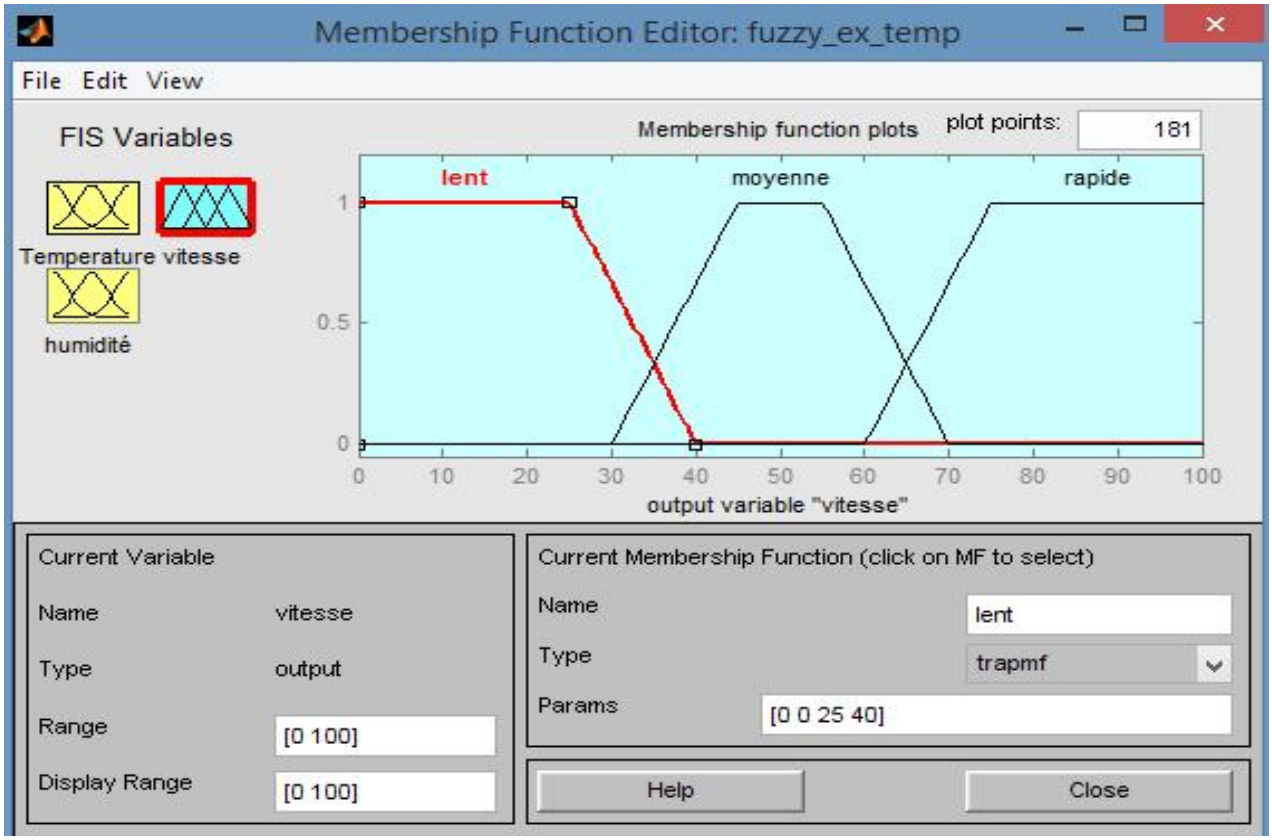
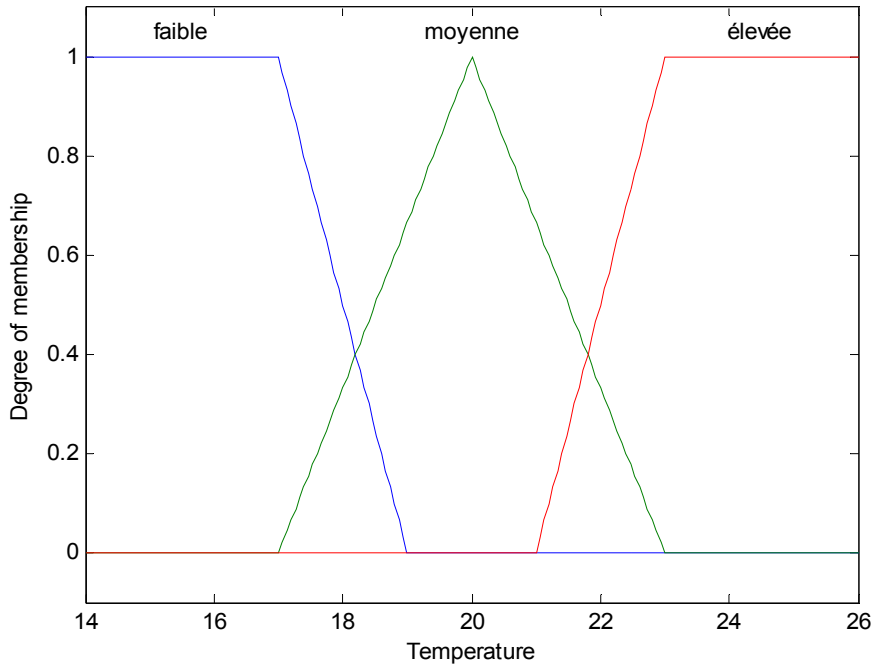


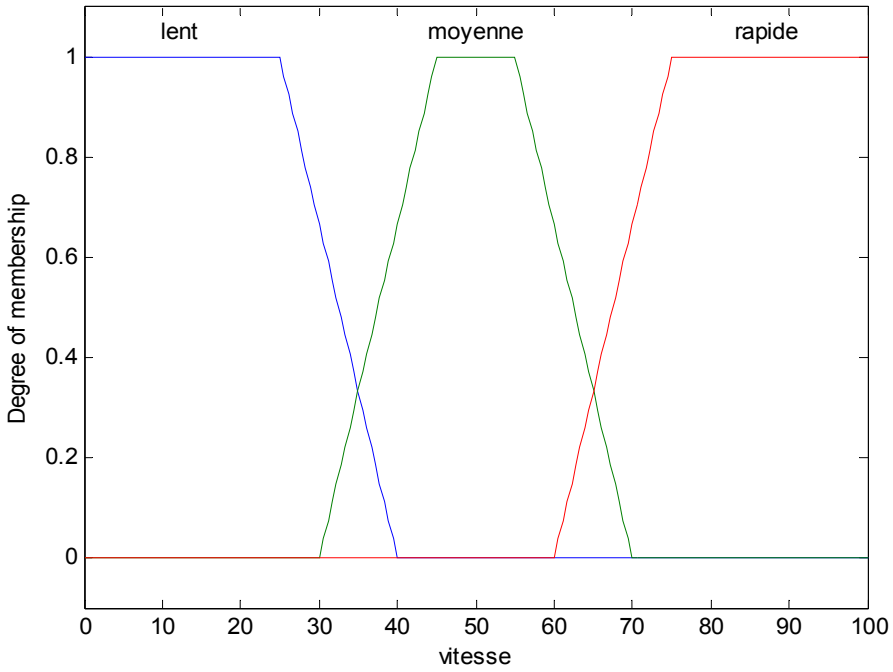
Figure (3.10): Fonctions d'appartenances de la sortie générées par FIS

On peut tracer les fonctions d'appartenances de la première entrée en utilisant l'instruction matlab:

plotmf(fuzzy_ex_temp, 'input', 1)



Pour la sortie: plotmf(fuzzy_ex_temp, ‘output', 1)



On peut visualiser la base des règles utilisée par l'instruction matlab:

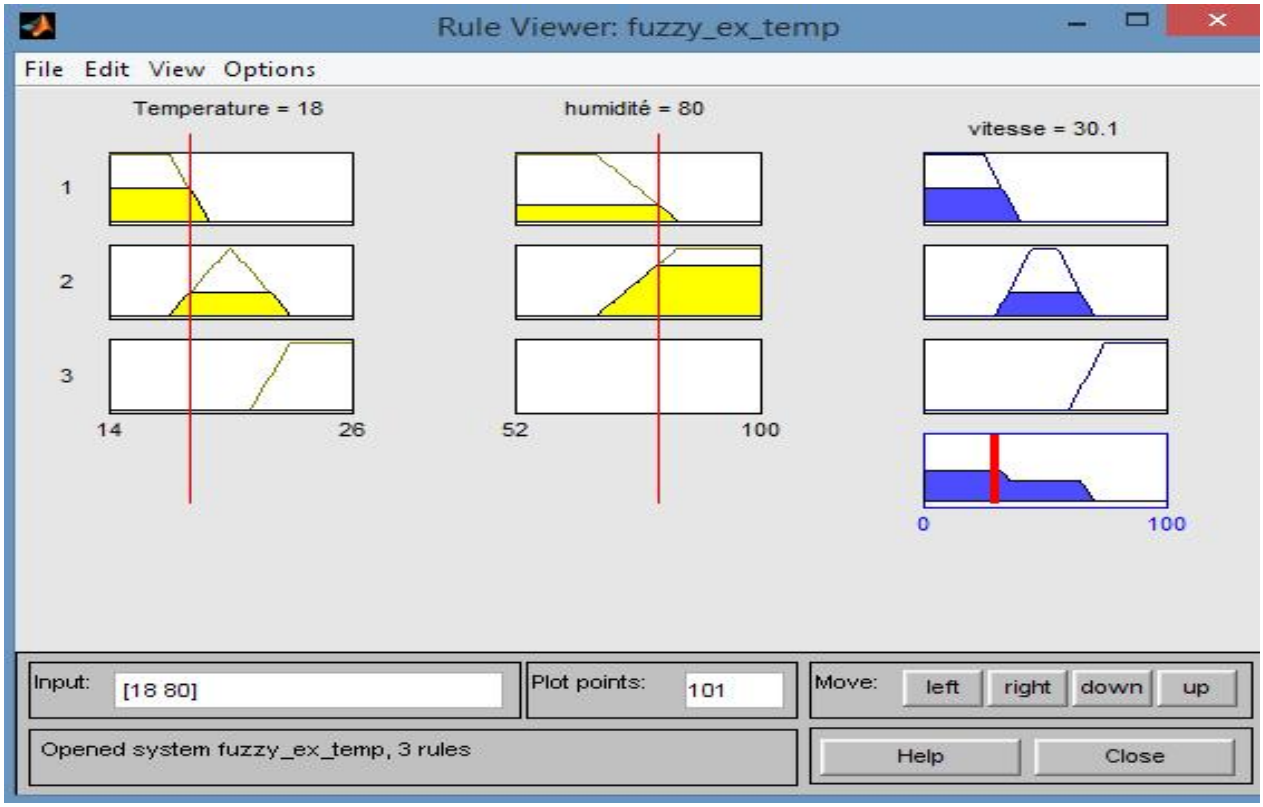
showrule(fuzzy_ex_temp)

On obtient la réponse:

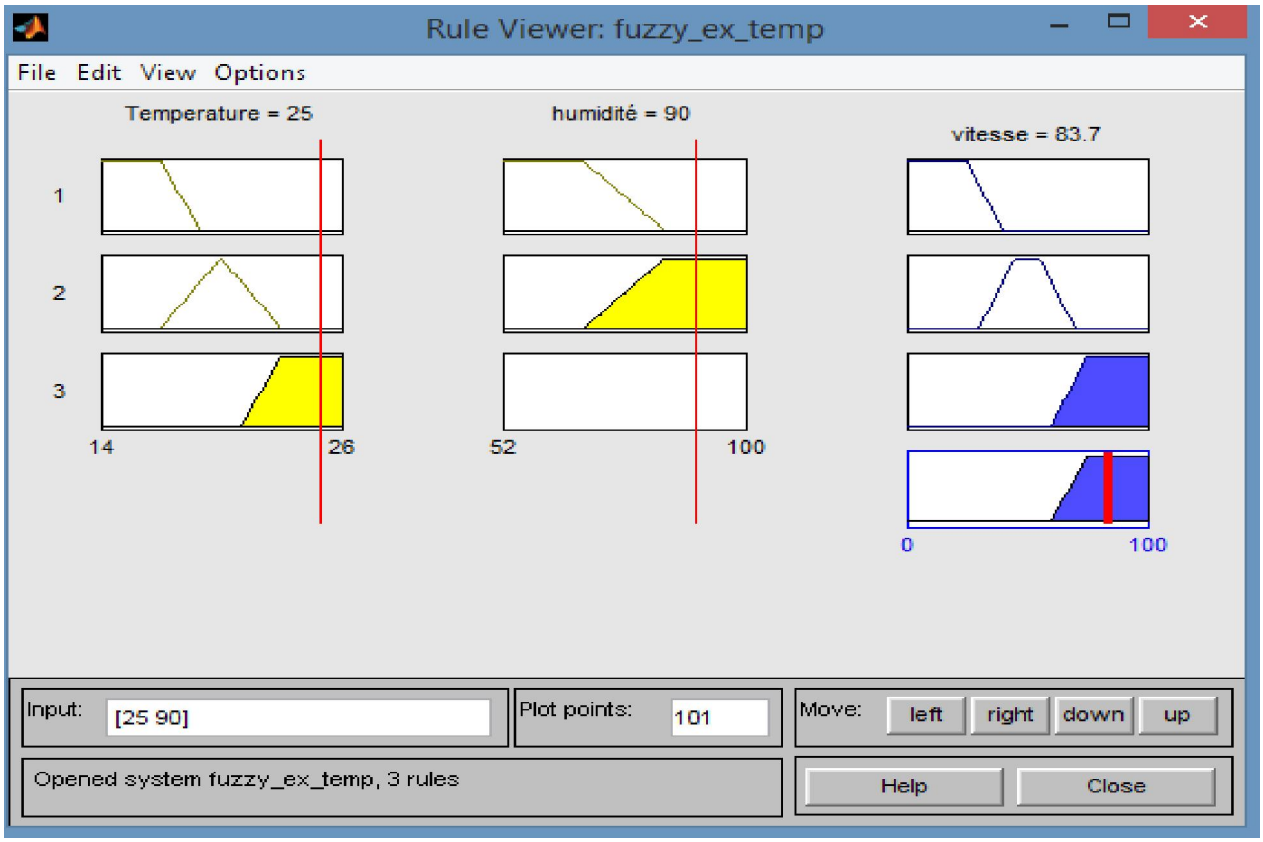
ans =

1. If (Temperature is faible) or (humidité is sec) then (vitesse is lent) (1)
2. If (Temperature is moyenne) and (humidité is humide) then (vitesse is moyenne) (1)
3. If (Temperature is élevée) then (vitesse is rapide) (1)

1er cas: On va choisir comme entrées: ***x = 18*** et ***y = 80***. Donc, on obtient une vitesse de **30.1%.** Cette vitesse est presque la même valeur obtenue théoriquement.



2ème cas: Dans ce deuxième cas, on choisit $x = 25$ et $y = 90$.



Dans ce cas, on obtient une vitesse de **83.7%** (vitesse rapide). Ce résultat est logique, puisque pour une maison (ou une chambre) de température élevée et humide, la vitesse de ventilateur doit être rapide.

***Exemple N°2:* Contrôleur d'un système décrit par F.T**

Le système à étudier est le suivant :

$$G(s) = \frac{3}{(s+1)(s+2)(s+3)}$$

En utilisant les méthodes classiques, on trouve que le contrôleur ***PID*** idéal a les valeurs suivantes:

K_P	12
K_I	12
K_D	3

On utilisera la réponse à une entrée échelon du système avec ***PID*** pour comparer les performances.

Le système avec contrôleur flou et contrôleur ***PID*** devrait être le suivant:

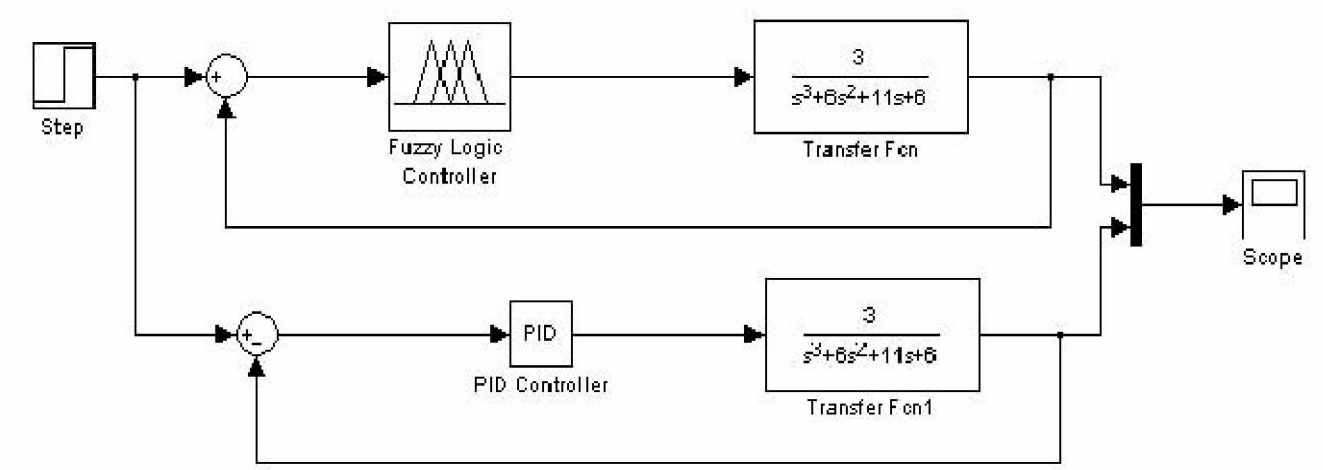
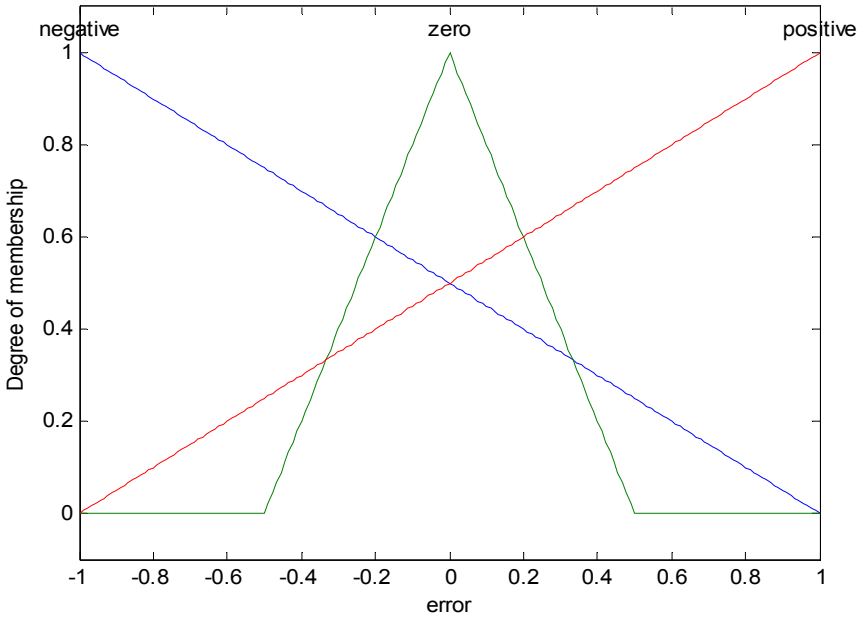


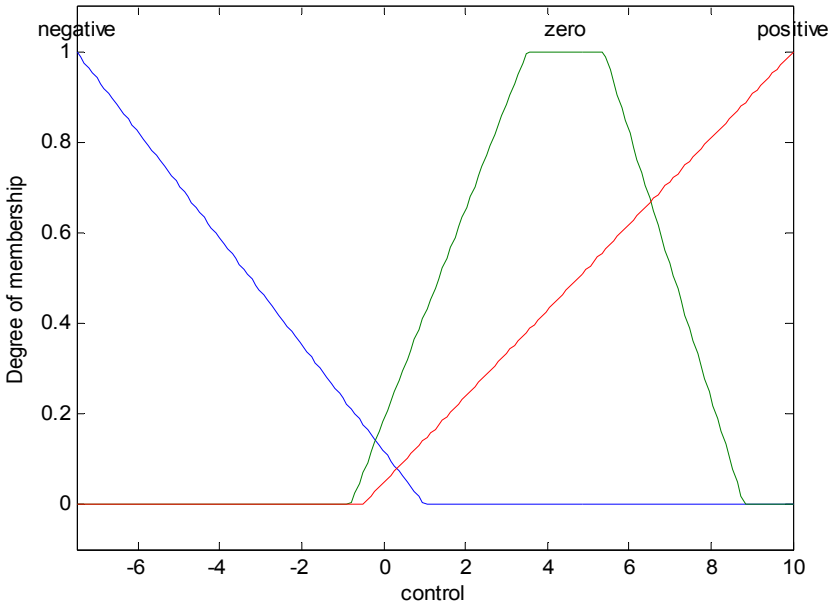
Figure (3.6): Schéma bloc du système étudié avec contrôleur PID et contrôleur flou

Pour le contrôleur flou utilisé, on choisit une entrée qui représente l'erreur ("Error") entre la consigne et la sortie du système, et une sortie qui représente le signal de commande ("Control").

Les fonctions d'appartenance de l'entrée ("Error") sont représentées sur la figure suivante:



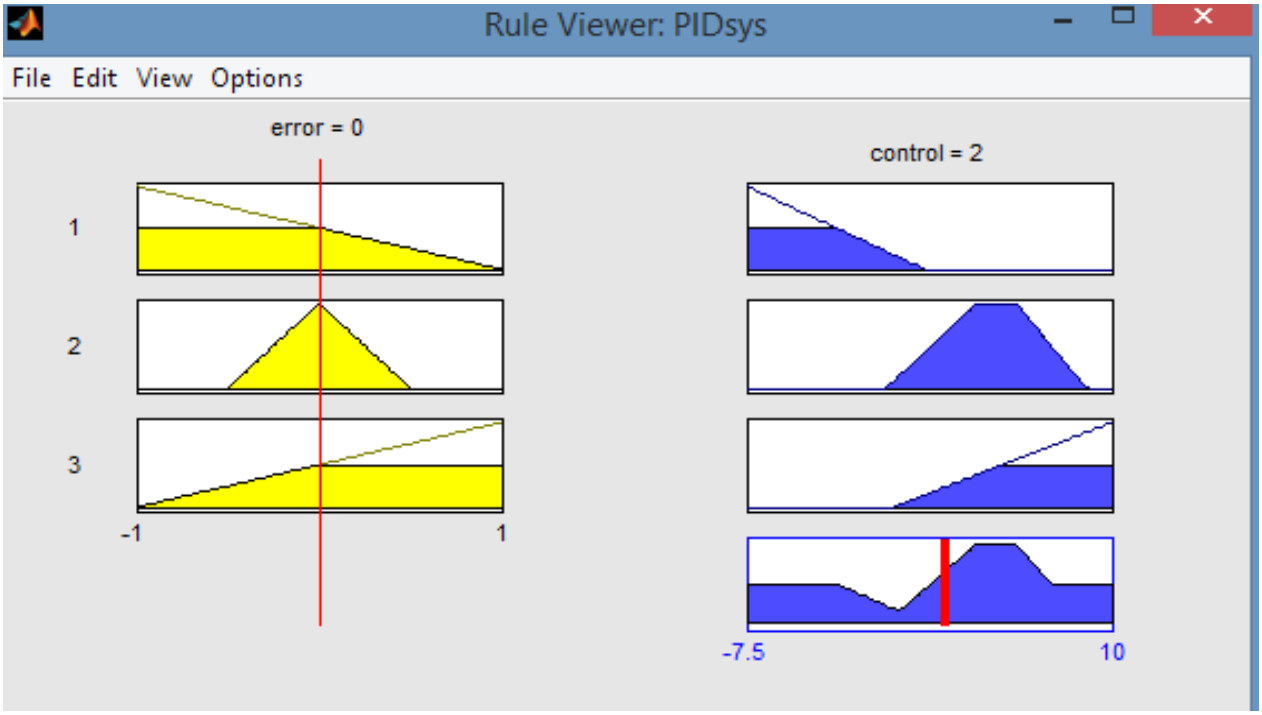
Les fonctions d'appartenance choisies pour la sortie ("Control") sont les suivants:



Les règles utilisées sont:

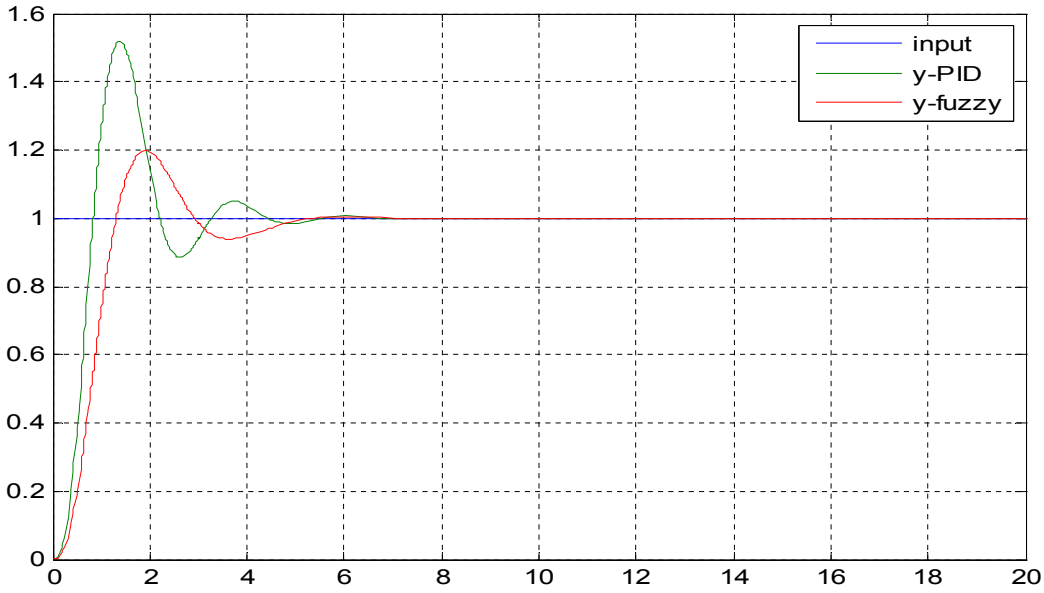
1. ***IF error is Negative THEN control is Negative***
2. ***IF error is Zero THEN control is Zero***
3. ***IF error is Positive THEN control is Positive***

La visualisation des règles est représentée sur la figure ci-dessous. On remarque que pour une erreur nulle (c'est-à-dire que la sortie atteint la valeur de la consigne), la commande doit prendre une valeur égale à 2. Cette valeur peut être prouvée théoriquement en utilisant le théorème de la valeur finale de la transformée de Laplace.



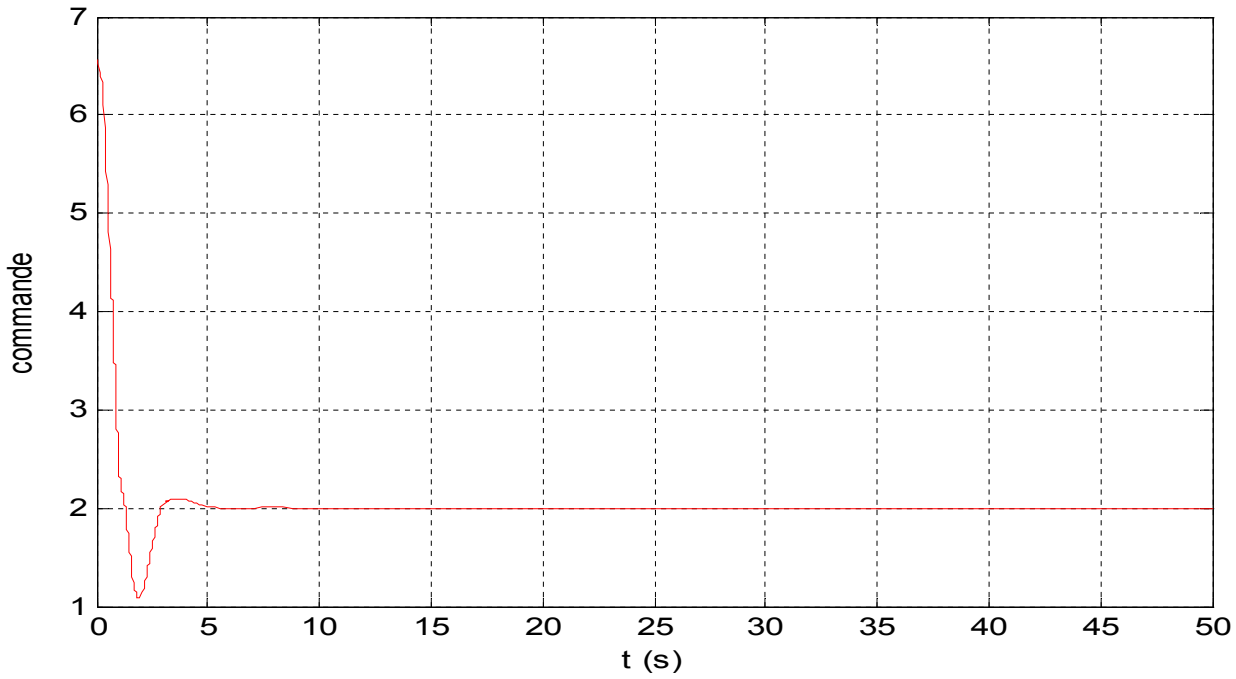
Après la simulation du schéma bloc du système en utilisant les deux contrôleurs PID et flou (Fuzzy logic controller FLC), la sortie du système est représenté par la figure ci-dessous.

Cette figure peut montrer la comparaison entre les performances du régulateur classique (PID) et le régulateur flou (FLC) pour une entrée qui représente un échelon unitaire.



-

La figure ci-dessous montre l'évolution du signal de commande au cours du temps.



Une comparaison entre les performances des deux régulateurs (PID et FLC) est effectuée. Les performances (temps de montée, temps de réponse et dépassement) sont mentionnés dans le tableau suivant:

Caractéristiques PID	***Caractéristiques FLC***
RiseTime (temps de montée): 0.4946	***RiseTime: 0.8012***
SettlingTime (temps d'établissement): 4.2012	***SettlingTime: 4.6906***
Overshoot: 51.8689	***Overshoot: 19.8373***

La comparaison des caractéristiques permet de prouver l'efficacité des deux régulateurs puisque leurs performances du point de vue temps de montée et temps de réponse sont comparables. Mais on peut dire que le régulateur FLC est plus efficace puisque il permet de réduire considérablement le dépassement.

Exemple sur les mécanismes d'inférences:

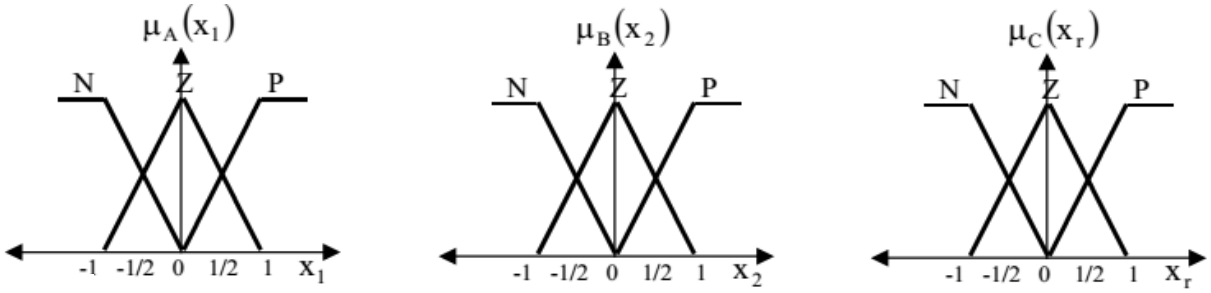
Nous avons deux entrées x_1, x_2 et une seul sortie x_r pour un régulateur flou. Considérons un ensemble des règles définies par :

Règle1: SI x_1 est N ET x_2 est Z ALORS x_r est Z. OU

Règle2 : SI x_1 est Z ET x_2 est P ALORS x_r est P.

Le cas: ***$x_1 = 1/4$; $x_2 = 1/2$.***

Les fonctions d'appartenances sont présentées comme suit:



Questions:

1- Déterminez la fonction résultante pour mécanisme **Max-Min**.

2- Déterminez la fonction résultante pour mécanisme **Max-Produit**.

3- Déterminez la fonction résultante pour mécanisme **Somme-Produit**.

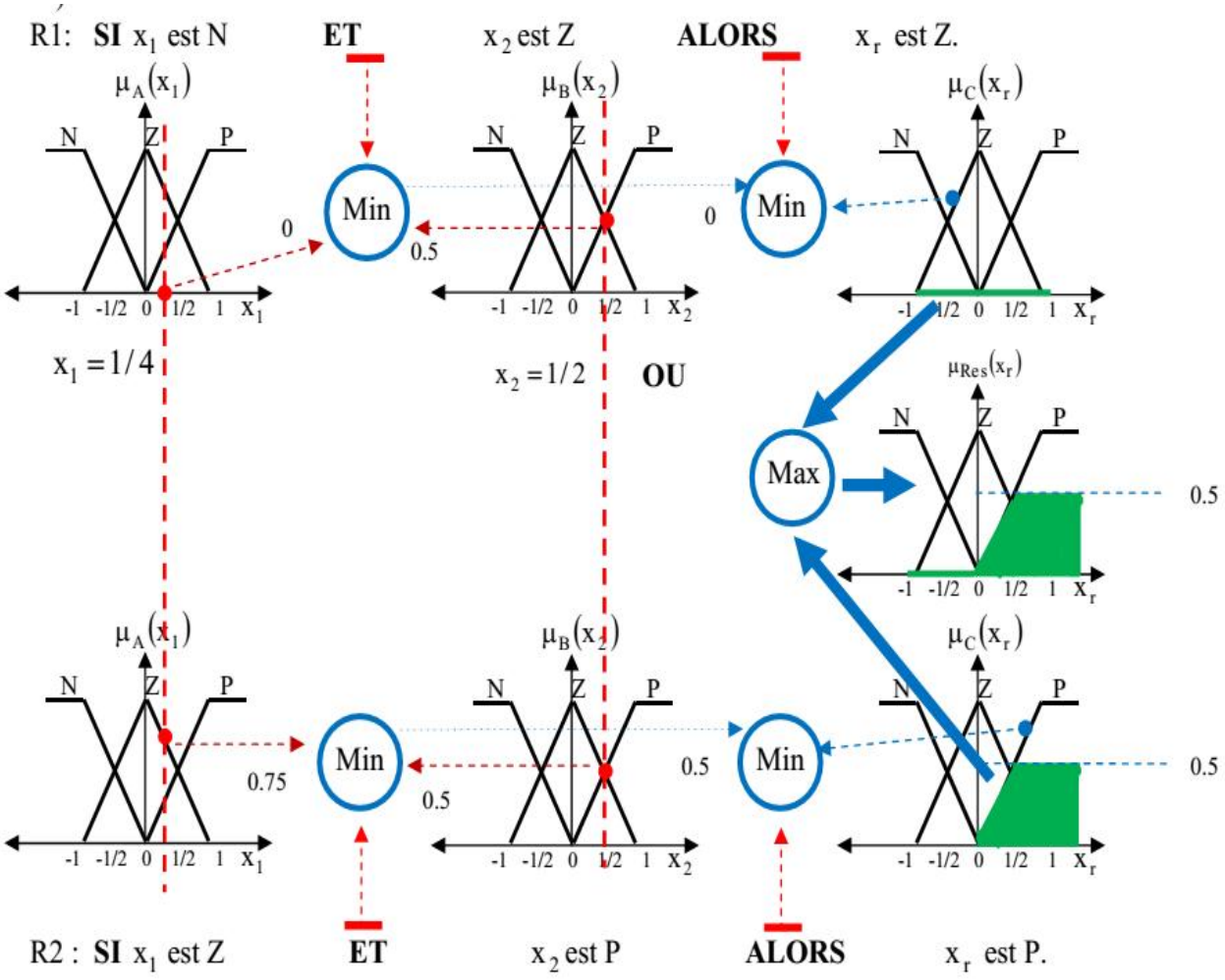
Réponse:

Nous allons expliquer le premier cas qui utilise la méthode d'inférence **Max-Min**

$$x_1 = \frac{1}{4} \Rightarrow \begin{cases} \mu_P(x_1) = \frac{1/4 - 0}{1-0} = \frac{1}{4} = 0.25 \\ \mu_Z(x_1) = \frac{1 - 1/4}{1-0} = 0.75 \\ \mu_N(x_1) = 0 \end{cases}$$

$$x_{21} = \frac{1}{2} \Rightarrow \begin{cases} \mu_P(x_1) = \frac{1/2 - 0}{1-0} = \frac{1}{2} = 0.5 \\ \mu_Z(x_1) = \frac{1 - 1/2}{1-0} = 0.5 \\ \mu_N(x_1) = 0 \end{cases}$$

Evaluation des règles et la fonction résultante:


R1: SI x1 est N
ET
x2 est Z
ALORS
xr est Z.
μA(x1)
μB(x2)
μC(xr)
N
Z
P
Min
Min
0
0.5
0
-1 -1/2 0 1/2 1 x1
-1 -1/2 0 1/2 1 x2
-1 -1/2 0 1/2 1 xr
x1 = 1/4
x2 = 1/2
OU
μRes(xr)
Max
0.5
0.75
0.5
0.5
0.5
R2 : SI x1 est Z
ET
x2 est P
ALORS
xr est P.

CHAPITRE IV: LES RESEAUX DE NEURONES ARTIFICIELS

I. INTRODUCTION:

Les réseaux de neurones formels sont des structures (la plupart de temps simulées par des algorithmes exécutés sur des ordinateurs d'usage général) qui prennent leur inspiration dans le fonctionnement élémentaire des systèmes nerveux. Ils sont essentiellement à résoudre des problèmes de classification, de reconnaissance de formes, d'association, d'extraction de caractéristiques, d'identification. Ils deviennent des compléments aux méthodes classiques, et sont mêmes susceptibles de se substituer à celles-ci avec un taux de succès supérieur.

- L'inspiration pour les réseaux de neurones provient de la volonté de créer des systèmes artificiels sophistiqués, voir intelligents, capables d'effectuer des opérations semblables à celle que le **cerveau humain** effectue de manière routinière.
- Les réseaux de neurones intéressent actuellement différents domaines tels que : la médecine, l'électronique, l'informatique, l'automatique, la robotique, la classification, le contrôle, le traitement des signaux, le traitement des images, …etc.
- Les cellules nerveuses, appelées neurones sont des éléments de base du système nerveux central qui se compose d'environ $\mathbf{10^{12}}$ neurones.

II. HISTORIQUE DES RESEAUX DE NEURONES:

- **1890**: W. James, célèbre psychologue américain introduit le concept de mémoire associative, et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones connue plus tard sous le nom de loi de Hebb.
- **1943**: J. Mc Culloch et W. Pitts laissent leurs noms à une modélisation du neurone biologique (un neurone au comportement binaire). Ceux sont les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes.
- **1949**: D. Hebb, physiologiste américain propose un mécanisme d'apprentissage.
- **1957**: F. Rosenblatt développe le modèle du Perceptron (Le perceptron est constitué d'un unique neurone), c'est le premier réseau de neurones artificiels.

- **1969**: M. Minsky et S. Papert publient un ouvrage qui met en exergue les limitations théoriques du perceptron (l'impossibilité de traiter par ce modèle des problèmes non linéaires).
- Les années **1980**: l'invention de l'algorithme de Rétro propagation des erreurs qui est appliqué aux réseaux de neurones multicouches (Multi-Layer Perceptron). Cet algorithme est le fruit des travaux de différents chercheurs : David Rumelhart, James Mc Clelland, Paul Werbos et Yann Le Cun.

III. DOMAINES D'APPLICATIONS:

Les différents domaines d'applications des réseaux de neurones sont résumés par l'organigramme suivant:

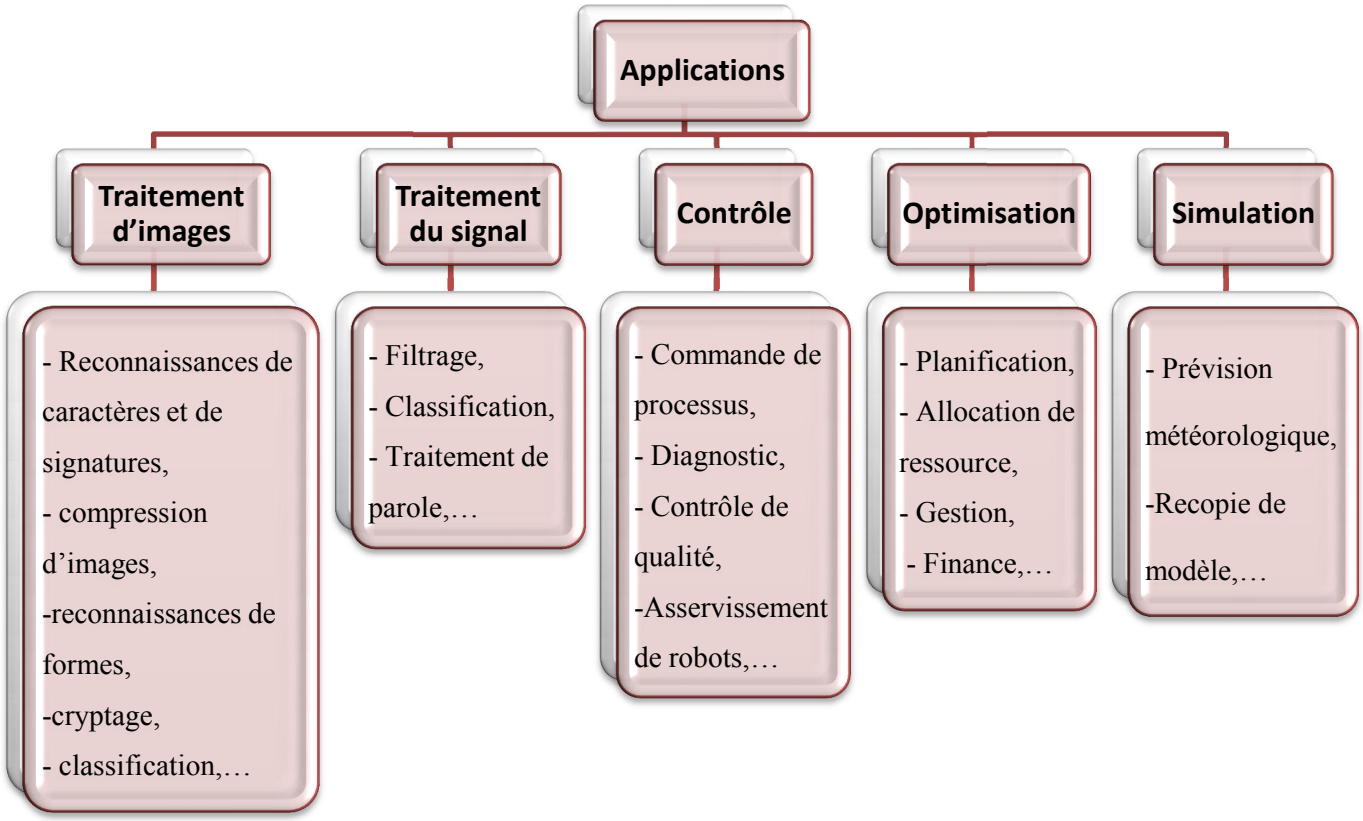
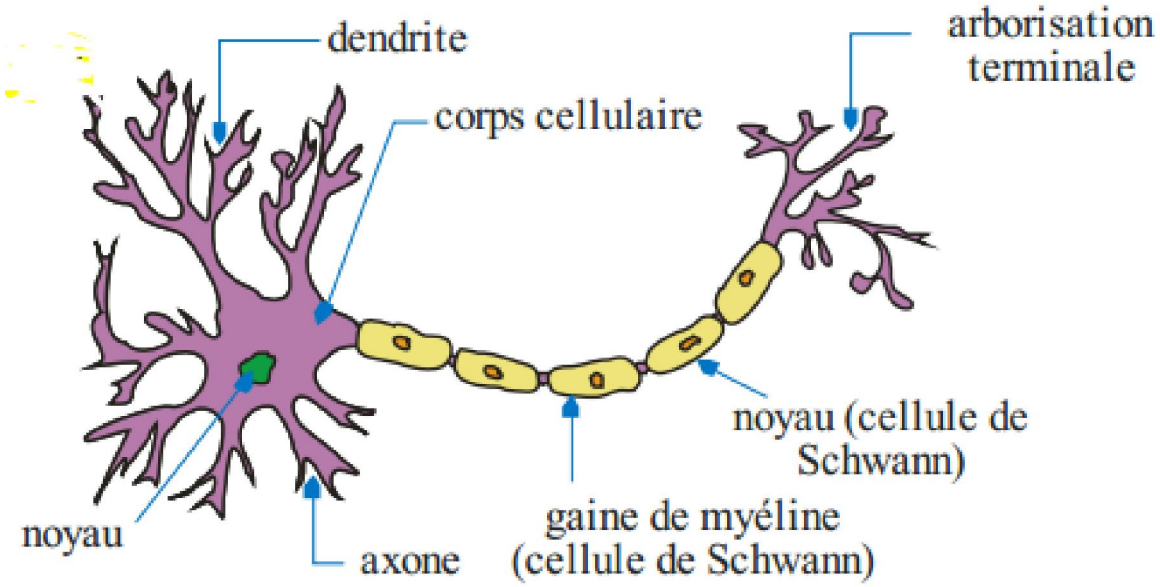


Figure (4.1): Domaines d'applications des réseaux de neurones

IV. DU NEURONE BIOLOGIQUE AU NEURONE FORMEL:

Les figures ci-dessous montrent la structure d'un neurone biologique et d'un neurone formel.



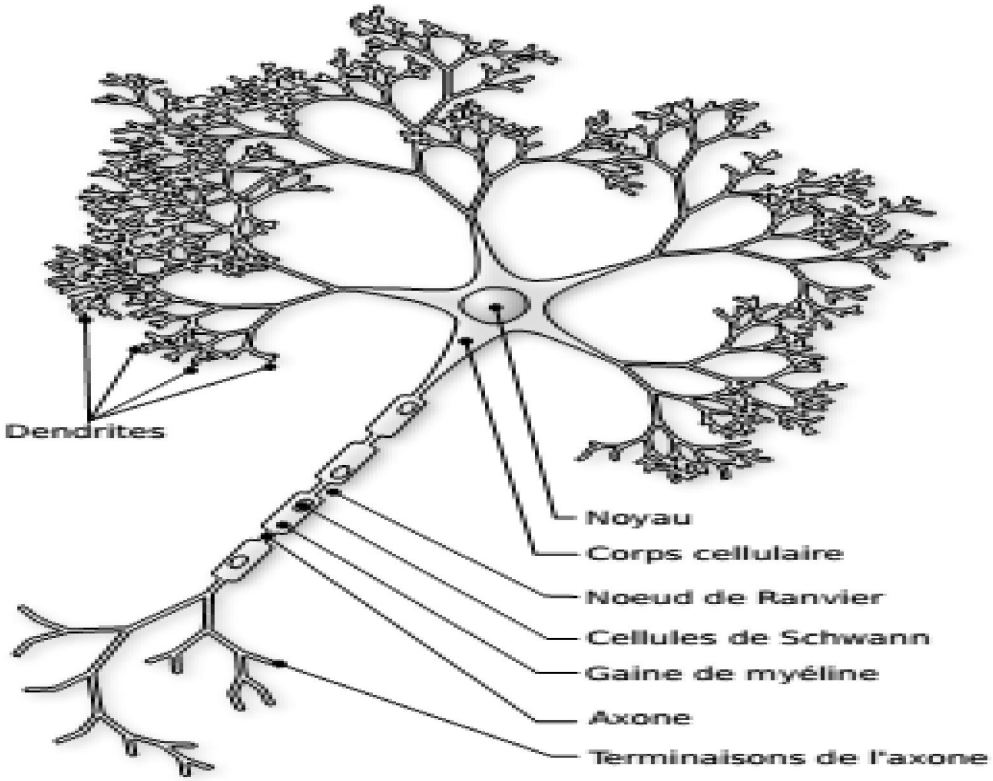


Figure (4.2): Neurone Biologique

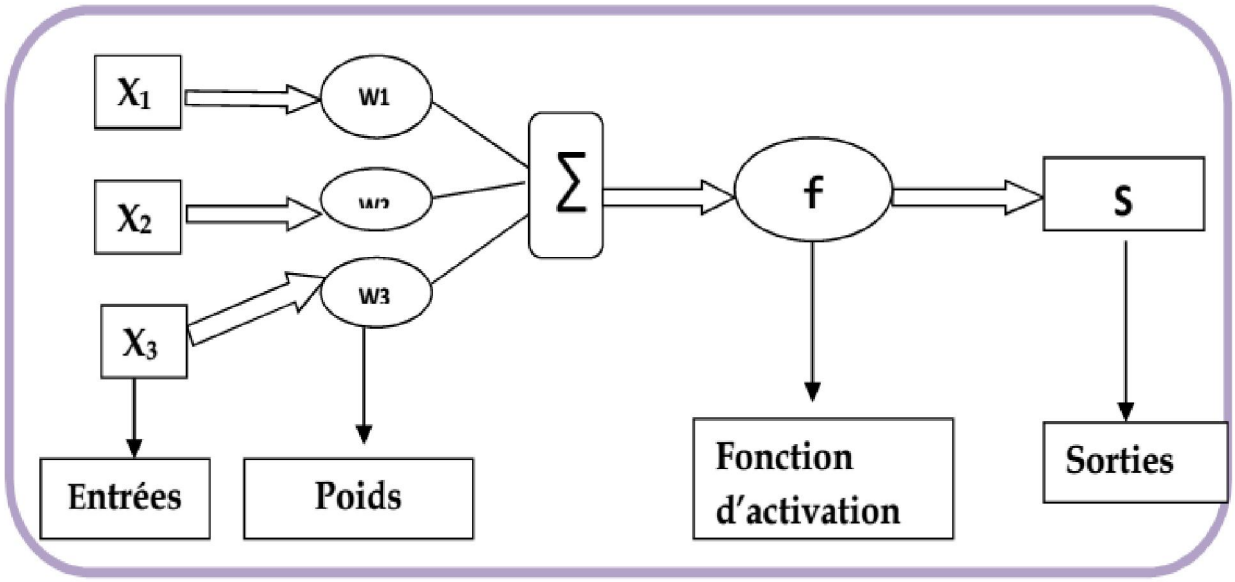
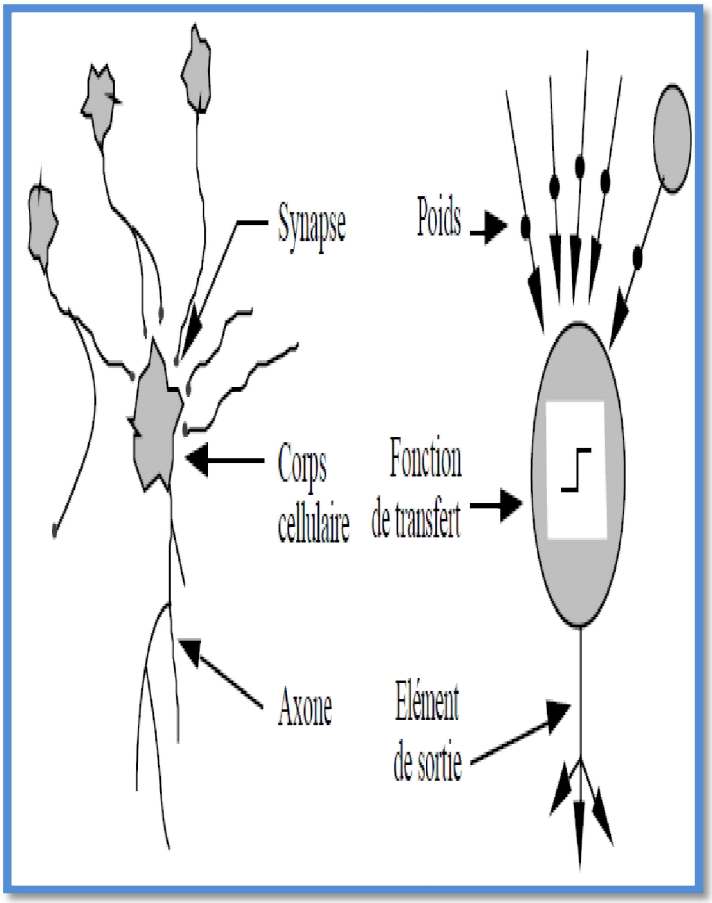


Figure (4.3): Neurone Formel

Comparaison entre un neurone biologique et un neurone formel



Neurone biologique	Neurone artificiel
Synapse	Poids de connexion (poids multiplicatifs)
Axone	Signal d'entrée (la communication avec d'autre neurone)
Dendrite	Signal de sortie
Le corps cellulaire (Somma)	Fonction d'activation

Figure (4.4): Comparaison entre neurone biologique et neurone formel

V. STRUCTURE D'UN NEURONE FORMEL:

L'élément de calcul individuel qui est la base des neurones artificiels et rarement appelé neurone artificiel mais plus souvent nœud, unité ou « Processing Element (P.E) ».

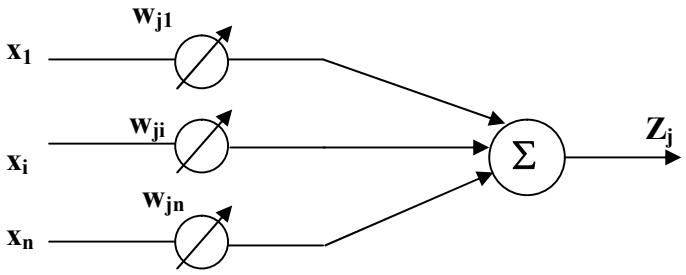


Figure (4.5): Processing Element (P.E)

Avec :

$Z_j = \sum_{i=1}^{n} w_{ji} x_i$ **Σ :** est la somme pondérée par les poids w_{ji} **.**

FONCTION D'ACTIVATION:

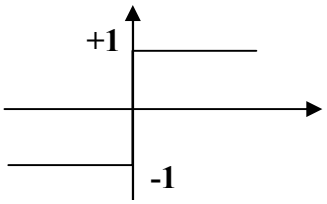
La sortie du P.E calculée, est convertie à une valeur d'activation. Donc, on peut l'écrire comme suit :

$y_j = F(Z_j)$ (**F :** fonction d'activation)

Les principales fonctions d'activation sont:

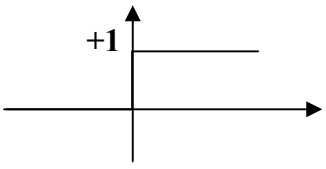
- ***La fonction signe:***

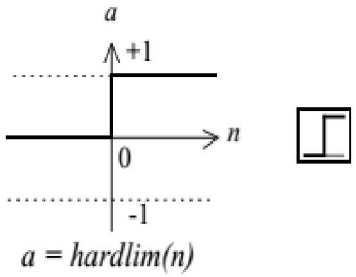
$$F(Z_j) = \begin{cases} +1 & si\ Z_j > 0 \\ -1 & si\ Z_j < 0 \end{cases}$$



- ***La fonction Heaviside:***

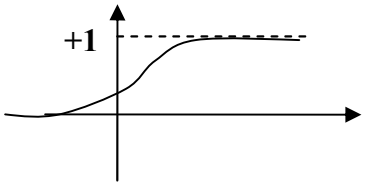
$$F(Z_j) = \begin{cases} +1 & si\ Z_j > 0 \\ 0 & si\ Z_j < 0 \end{cases}$$

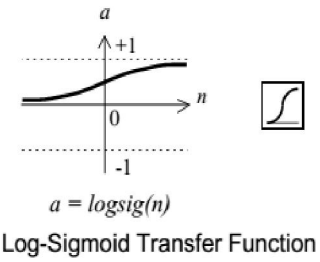




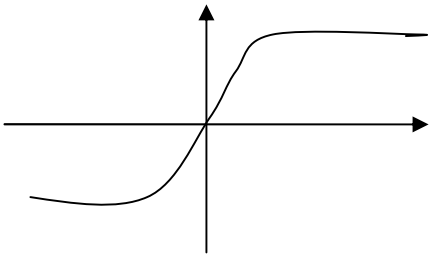
- ***Les fonctions Sigmoïdes:***

$$F(Z_j) = \frac{1}{[1 + exp(-Z_j)]} = \frac{1}{1 + e^{-Z_j}}$$



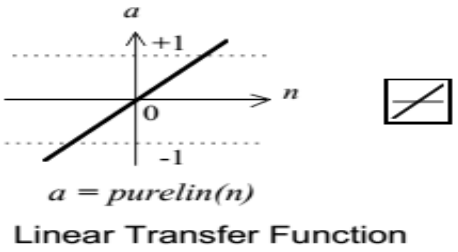


$$ou \quad F(Z_j) = \frac{[1 - exp(-Z_j)]}{[1 + exp(-Z_j)]} = \frac{1 - e^{-Z_j}}{1 + e^{-Z_j}}$$



- ***La fonction Linéaire:***

  $F(Z_j) = Z_j$



VI. RESEAUX DE NEURONES:

Un réseau de neurone est un ensemble de neurones connectés suivant une topologie spécifique et suit une loi d'apprentissage pour adapter les poids de connexions. Ces neurones constituent des éléments de traitement de l'information qui travaillent simultanément. Chaque neurone, son rôle est d'élaborer l'information reçue et à transmettre les résultats à d'autres neurones.

- ***Réseau de neurone multicouche:***

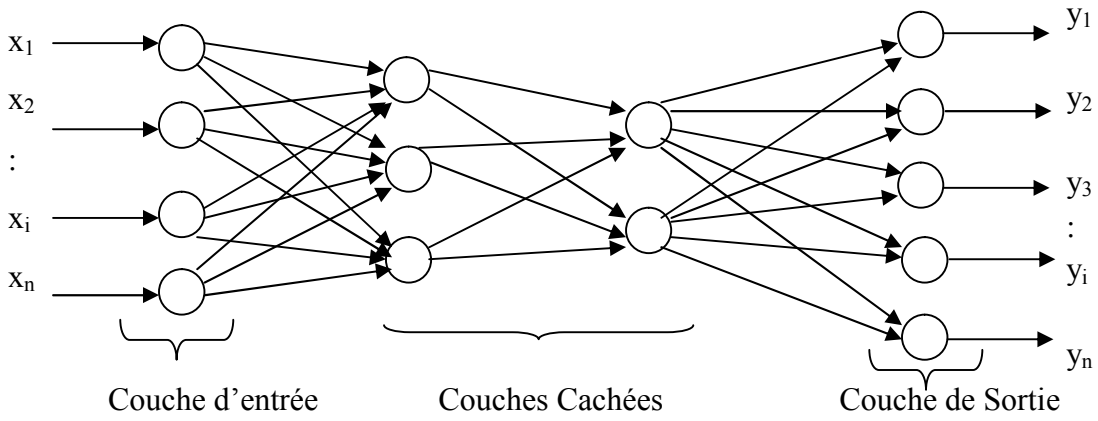


Figure (4.6): Réseau de neurone multicouche

Un réseau multicouche est un réseau statique. Il est également appelé « Feed-Forward » car le flux d'information est dirigé des entrées vers les sorties.

- ***Réseau multicouche récurrent:*** ***(Multilayer Feedback Network)***

Une extension du réseau précédent est le réseau multicouche récurrent qui est dynamique : ses poids dépendent non seulement des entrées apprises mais également des sorties précédentes.

Ce type de réseau est utilisé surtout pour estimer des processus et des systèmes dynamiques non linéaires.

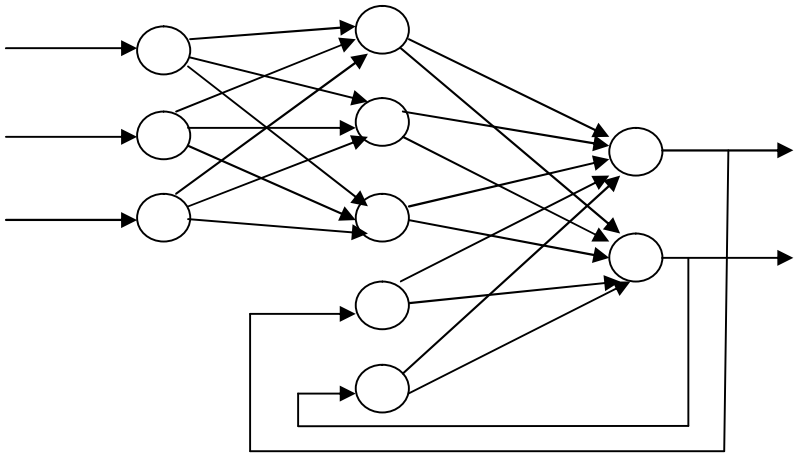


Figure (4.7): Réseau de neurone multicouche récurrent

VII. APPRENTISSAGE (ENTRAINEMENT):

L'entrainement du réseau consiste à modifier les poids jusqu'à ce qu'il se stabilise. C'est-à-dire, jusqu'à ce que les poids ne se modifient plus que d'une façon infime.

On distingue deux types d'apprentissage :

a) *Apprentissage supervisé:*

Dans ce cas, la connaissance à priori de la sortie désirée est nécessaire. Les poids du réseau sont ajustés de façon à réduire l'écart entre la sortie calculée (du réseau) et la sortie désirée.

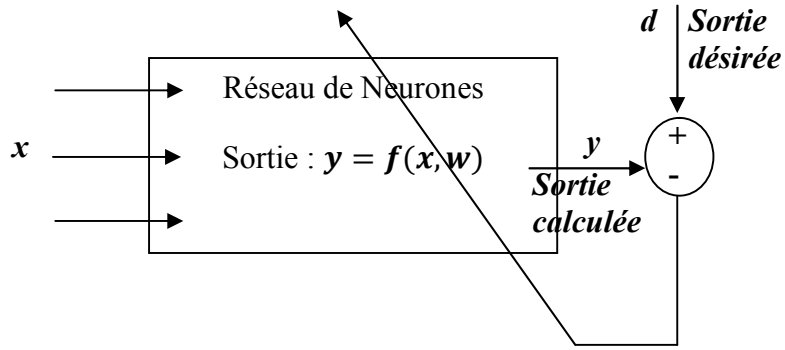


Figure (4.8): Apprentissage supervisé

b) ***Apprentissage non-supervisé:***

Dans ce cas, l'apprentissage est basé uniquement sur les vecteurs d'entrées sans fournir de réponse désirée. Le réseau s'auto-organise de façon à optimiser une fonction de coût.

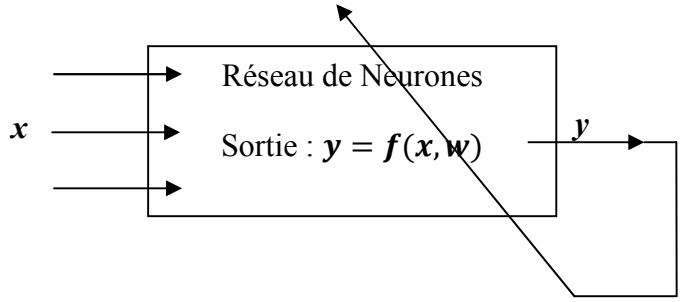


Figure (4.9): Apprentissage non-supervisé

L'algorithme qui a approuvé son efficacité dans la pratique est l'algorithme de « Back-Propagation ».

c) ***Principe de l'algorithme de « Back-Propagation » (B-P) :***

- L'algorithme est appliqué pour les réseaux multicouches (un à deux couches cachées).
- L'apprentissage est supervisé
- Fonction d'activation : une fonction dérivable (Sigmoïde)
- La méthode est basée su l'algorithme de la descente de gradient (minimisation de l'erreur quadratique entre les sorties obtenues et celles désirées)

Principe :

- L'algorithme de (B-P) consiste à rétro-propager l'erreur commise en sortie vers les couches cachées et cela après propagation du signal d'entrée vers la couche de sortie. Trouver les poids optimaux, qui minimisent cette erreur et constitue l'entrainement du réseau.
- On dispose d'un ensemble d'exemples (vecteurs d'entrainement) qui sont des couples entrées-sorties
- A chaque itération, un exemple est présenté au réseau, le résultat du traitement est ensuite comparé avec la sortie désirée
- On calcule l'erreur, sa dérivée, le gradient de la dérivée
- On ajuste les poids en se basant sur le gradient correspondant à chacun
- Cette procédure est répétée couche par couche
- Le processus est répété en représentant successivement chaque exemple, jusqu'à ce que le réseau converge (l'erreur de sortie se stabilise à un seuil acceptable).

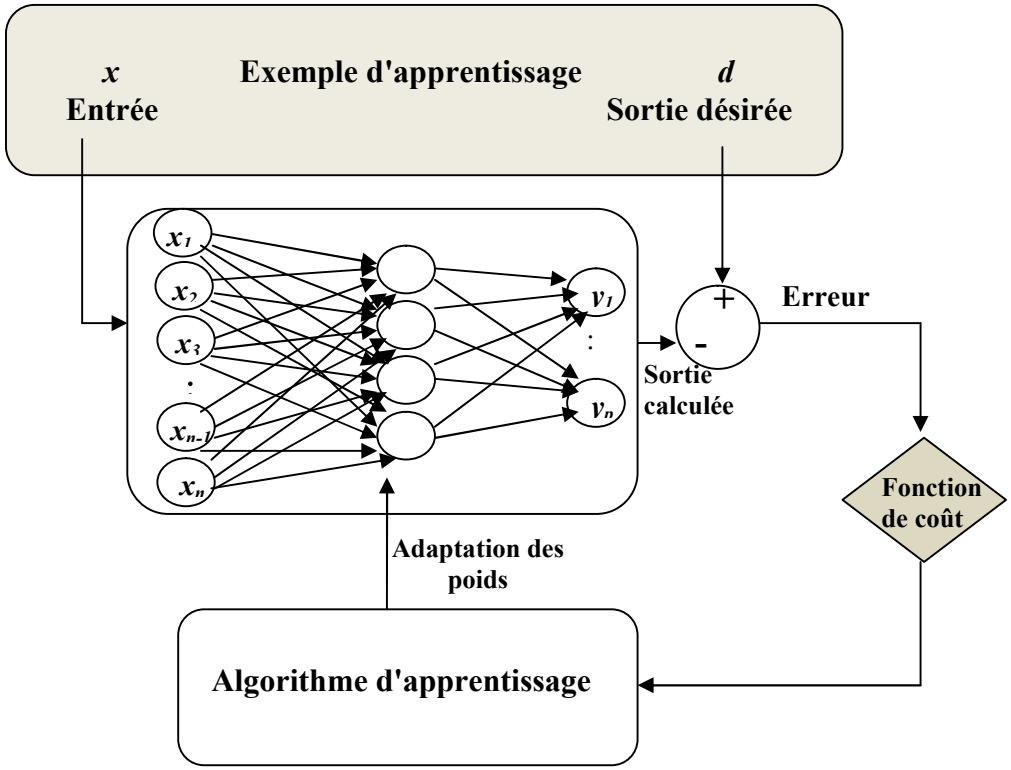
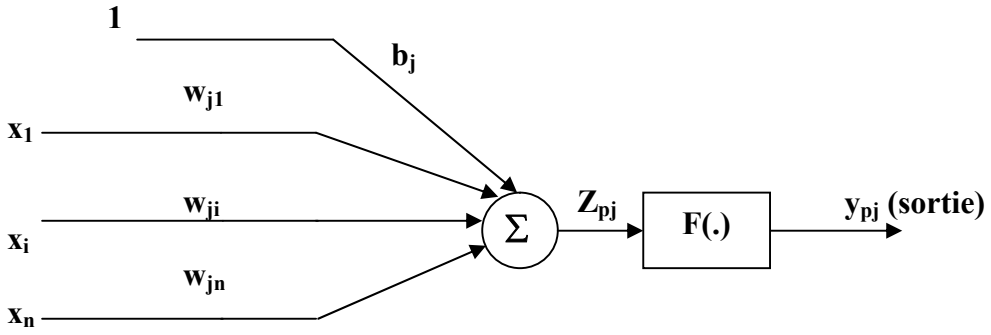


Figure (4.10): Principe de l'algorithme de B-P

L'algorithme de « Back-Propagation » (B-P):

Soit le vecteur d'entrée : $X_p = [x_{p1}\ x_{p2}\ \dots\ x_{pn}]$ est représenté au j^{eme} neurone dont le vecteur poids $w_j = [w_{j1}\ w_{j2}\ \dots\ w_{jn}]$.

La sortie du (P.E) est : $Z_{pj} = X_p{}^t w_j + b_j$ **ou** $Z_{pj} = \sum_{i=1}^{n} x_{pi} w_{ji} + b_j$, b_j : le biais (le seuil)



Avec : $y_{pj} = F(Z_{pj})$

- Les neurones appartenant à la couche de sortie, la fonction d'activation est linéaire
- Les autres neurones (appartenant aux autres couches), la fonction d'activation est sigmoïde.

La fonction de coût à minimiser est : $\boldsymbol{E(w) = \sum_{p=1}^{M} E_p(w)}$

Avec : ***M*** : le nombre d'exemples d'entrainement

$\boldsymbol{E_p(w)}$: le carré de l'erreur associée au $\boldsymbol{p^{ème}}$ exemple.

$$\boldsymbol{E_p(w) = \frac{1}{2}\sum_{j=1}^{N}\left(d_{pj} - y_{pj}\right)^2}$$

N **:** le nombre de neurones dans la couche de sortie

$\boldsymbol{d_{pj}}$: la sortie désirée du nœud de sortie *« j »*

$\boldsymbol{y_{pj}}$: la sortie du réseau au nœud *« j »*

On calcule le gradient :

$$\boldsymbol{\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial y_{pj}} \cdot \frac{\partial y_{pj}}{\partial Z_{pj}} \cdot \frac{\partial Z_{pj}}{\partial w_{ji}}}$$

$\boldsymbol{\frac{\partial E_p}{\partial y_{pj}}}$: selon la couche à laquelle appartient le nœud en question (couche de sortie ou couche cachée)

$$\boldsymbol{\frac{\partial y_{pj}}{\partial Z_{pj}} = F'(Z_{pj})}$$

$$\boldsymbol{\frac{\partial Z_{pj}}{\partial w_{ji}} = x_{pi} \;\; avec: w_{ji}: poid\ du\ neurone} \text{ "j" } \boldsymbol{correspondant} \text{ à } \boldsymbol{l'entr}\text{ée "}\boldsymbol{i}\text{"}$$

L'équation d'ajustement des poids est:

$$\boldsymbol{w_{ji}(k+1) = w_{ji}(k) - Lr.\frac{\partial E_p}{\partial w_{ji}}}$$

Avec : Lr : est le pas d'apprentissage (de l'ordre de **0.05 à 0.25**)

On peut écrire :

$$\boldsymbol{w_{ji}(k+1) = w_{ji}(k) + Lr.\Delta j.x_{pi} \quad et\ \Delta j = -\left(\frac{\partial E_p}{\partial y_{pj}}\right).F'(Z_{pj})}$$

Pour augmenter la rapidité de l'algorithme, on peut utiliser la technique du « momentum » :

$$\boldsymbol{w_{ji}(k+1) = w_{ji}(k) + Lr.\Delta j.x_{pi} + \alpha.\left(w_{ji}(k) - w_{ji}(k-1)\right), \quad 0 < \alpha < 1}$$

Algorithme :

1- Initialisation des poids et des biais (entre ± 0.5)

2- Présenter un vecteur à l'entrée du réseau : $X_p = [x_{p1}\ x_{p2}\ \dots\ x_{pn}]$ et la sortie désirée associée : $D_p = [d_{p1}\ d_{p2}\ \dots\ d_{pn}]$

3- Calculer la sortie y_p du réseau en propageant X_p à travers ce dernier en utilisant les équations nécessaires (les équations des sorties)

4- Calculer Δj associé à chaque neurone du réseau

5- Ajuster les poids associé à chaque nœud ($w_{ji}(k+1)$)

6- Revenir à l'étape **2**

On présente les vecteurs d'entrées successivement jusqu'à ce que l'erreur de sortie se stabilise

VIII. EXEMPLES D'APPLICATION:

Exemple N°1: Régulation d'un système décrit par fonction de transfert

Soit le système décrit par la fonction de transfert:

$$G(s) = \frac{3}{(s+1)(s+2)(s+3)}$$

On veut réaliser la régulation de ce système en utilisant trois types de régulateurs: régulateur classique PID, régulateur flou (FLC) et un régulateur neuronale. Une comparaison des performances de ces trois régulateurs est effectuée.

La simulation sous Simulink est réalisée en utilisant le schéma bloc suivant:

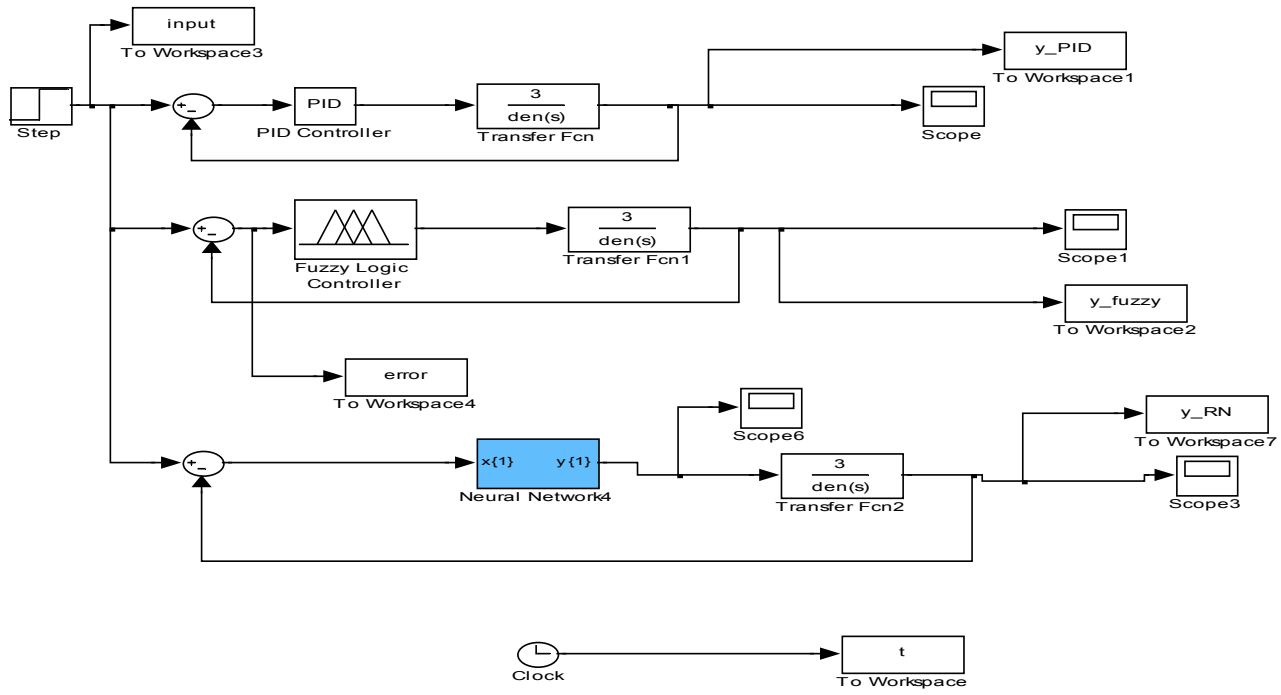


Figure (4.11): Schéma bloc du système simulé en utilisant les trois régulateurs: PID, FLC et neuronale

La conception du contrôleur flou est déjà expliquée dans le chapitre précédent tandis que le contrôleur neuronal est réalisé par conception d'un réseau de neurone multicouche feed-forward, constitué d'une seule couche d'entré, une couche cachée de 20 neurones et une couche de sortie. L'entrée du réseau est l'erreur et la sortie est la commande. L'algorithme utilisé pour l'optimisation des poids est l'algorithme de Back-propagation.

Les instructions utilisées pour la conception du réseau sont représentés par le script suivant:

```
u=2.*ones(length(error),1)';
net=newff([1 0],[20 1],{'tansig','purelin'});
net1=train(net,error',u);
yout=sim(net1,error)
gensim(net1,-1)
```

La simulation du bloc Simulink a permet de tracer les sorties obtenues par les trois régulateurs:

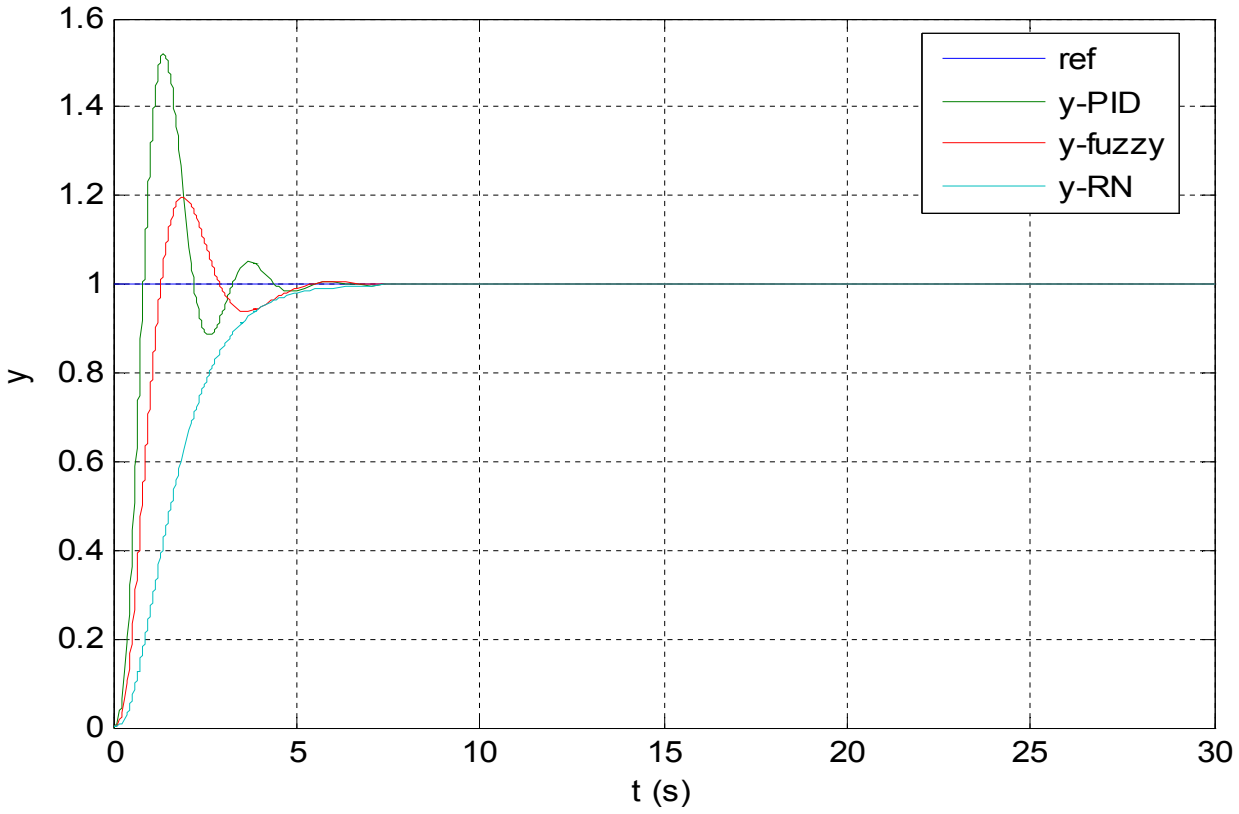


Figure (4.12): Les sorties obtenues par les trois régulateurs

Les caractéristiques des trois régulateurs sont mentionnées dans le tableau suivant:

Caractéristiques PID	**Caractéristiques FLC**	**Caractéristiques RN**
RiseTime (temps de montée): **0.4946**	RiseTime: **0.8012**	RiseTime: **2.7434**
SettlingTime (temps d'établissement): **4.2012**	SettlingTime: **4.6906**	SettlingTime: **5.0053**
Overshoot: **51.8689**	Overshoot: **19.8373**	Overshoot: **0**

IX. LES RESEAUX A FONCTION DE BASE RADIALE (RBF):

1. *Définition et architecture du réseau RBF:*

Les réseaux RBF est une architecture de réseaux Feedforward puissante. Ce type de réseau a été introduit par Hardy en 1971, et la théorie correspondante a été développée par Powell en 1985. Ces réseaux étaient à l'origine appliqués aux problèmes d'interpolation multivariable et ils ont pris le terme de réseaux de neurones grâce à Broomhead et Lowe.

Ces réseaux de neurones ont trouvé des domaines d'applications variés comme le traitement de la parole, traitement de l'image et traitement du signal.

Les réseaux RBF ("Radial Basis Function") sont des réseaux à couches qui ont comme origine une technique d'interpolation nommée la méthode d'interpolation à base de fonction radiale BFR. Ce réseau comporte une seule couche cachée dont la fonction d'activation est appelée fonction-noyau ou gaussienne et une couche de sortie avec une fonction d'activation linéaire. La méthode RBF est particulière par ses réponses utiles pour un domaine de valeurs restreint.

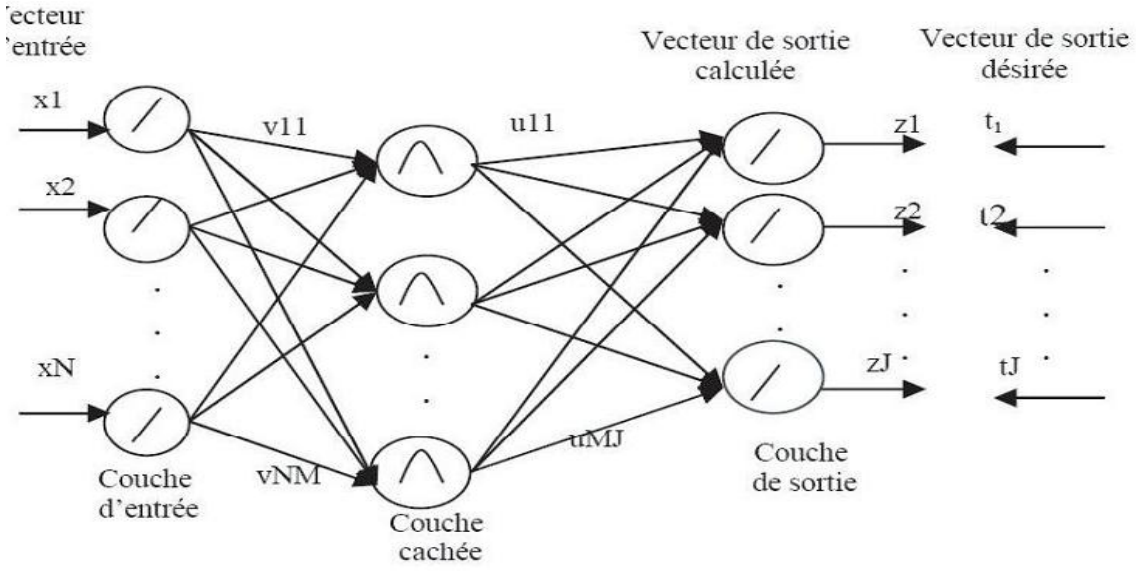


Figure (4.13): Architecture d'un réseau de neurones RBF

Le réseau RBF représenté par la figure ci-dessus, est constitué de ***N*** neurones d'entrée, ***M*** neurones cachés et ***J*** neurones de sortie.

La sortie du m^{ième} neurone de la couche cachée est donnée par:

$$y_m^q = exp\left[-\frac{\|x^q - v_m\|^2}{2\sigma_m^2}\right]$$

Avec: v_m est le centre m^{ième} neurone de la couche cachée et σ_m est la largeur de la m^{ième} gaussienne.

La sortie du j^{ième} neurone de la couche de sortie est donnée par:

$$y_j^q = \frac{1}{m}\sum_{m=1}^{M} w_{mj} y_m$$

w_{mj} sont les poids reliant les neurones de la couche cachée à ceux de la couche de sortie.

2. *Algorithme d'apprentissage d'un réseau RBF:*

L'apprentissage du réseau RBF consiste à régler quatre paramètres principaux:

- le nombre de neurones dans la couche cachée (le nombre de gaussiennes)
- la position des centres des gaussiennes
- la largeur des gaussiennes
- les poids de connexions entre les neurones cachés et les neurones de sorties.

Le réseau RBF consiste à minimiser l'erreur quadratique totale EQT calculée entre les sorties calculées et celles désirées:

$$EQT = \sum_{q=1}^{Q}\sum_{j=1}^{J}\left(t_j^q - z_j^q\right)^2$$

Pour le réseau RBF, l'ajustement des poids w_{mj} est donné par l'équation suivante:

$$w_{mj}^{i+1} = w_{mj}^{i} + \mu(t_j - z_j)y_m$$

μ est le pas d'apprentissage dont la valeur est comprise entre 0 et 1.

L'avantage principal du réseau RBF est qu'il est possible de simplifier l'apprentissage en divisant le travail en trois phases:

- positionnement des centres
- détermination des largeurs des noyaux gaussiens
- adaptation des poids

Il existe plusieurs stratégies d'apprentissage selon la façon dont les centres sont spécifiés:

- Centres fixes sélectionnés de façon aléatoire: choisir les centres et les dispersions suite à une analyse de données, les poids sont la solution des équations linéaires.
- Centres obtenus par apprentissage supervisé: estimer les centres par catégorisation Kmeans, les dispersions peuvent êtres choisies comme la distance moyenne entre les centres voisins, et les poids sont la solution des équations linéaires.
- Centres obtenus par apprentissage supervisé: tous les paramètres sont ajustés pour minimiser une fonction d'erreur.

Les Kmeans sont des méthodes itératives permettant de séparer une série de vecteurs en différents clusters et chaque cluster est représenté par un centre.

Algorithme de Kmean: les étapes de l'algorithme sont les suivantes:

1. Choisir au hasard le centre de chacun des K clusters.
2. Attribuer chaque objet au cluster dont le centre lui est le plus proche.
3. Recalculer les positions des nouveaux centres.
4. Répéter les étapes 2 et 3 jusqu'à convergence (nombre maximal d'itérations atteint).

X. LES RESEAUX DE NEURONES RECURRENT (RNN):

Contrairement à un réseau de neurones acyclique (Feed-Forward Neural Network -FFNN), un réseau de neurones récurrent (Recurrent Neural Network - RNN) est un réseau de neurones dont le graphe de connexion contient au moins un cycle. Comme pour les FFNN, de nombreux types de RNN ont été développés au cours des 30 dernières années tels que les réseaux d'Elman, les réseaux de Jordan et les Echo State Networks.

RNN sont une classe de réseaux de neurones qui sont naturellement adaptés au traitement de données de séries chronologiques et d'autres données séquentielles.

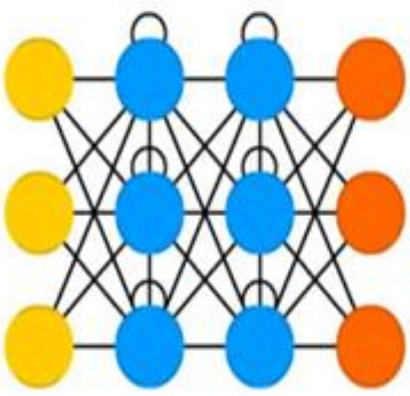


Figure (4.14): Réseau de neurones récurrent (RNN)

CHAPITRE V: RESEAUX ADAPTATIFS ET RESEAUX NEURO-FLOUS

I. DEFINITION D'UN SYSTEME EVOLUTIF HYBRIDE:

Le développement d'un système hybride peut être nécessaire pour obtenir les solutions optimales. La figure 5.1 représente la notion de systèmes évolutifs hybrides, comprenant trois courants majeurs (par exemple, systèmes d'inférence floue, RNA et les algorithmes évolutifs).

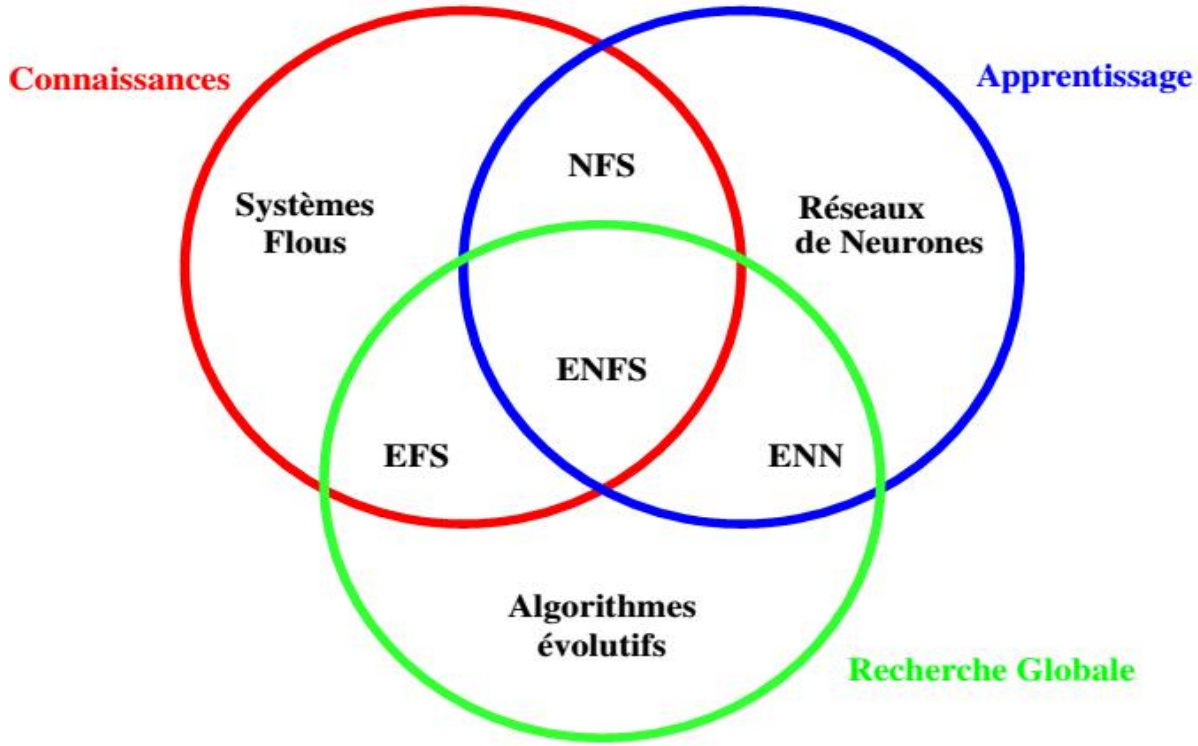


Figure (5.1): Système évolutif hybride

II. SYSTEME NEURO-FLOU:

Les systèmes neuro-flous sont largement connus pour leur capacité à modéliser des systèmes dynamiques complexes. Fusion de RNA et de système d'inférence floue le neuro-flou a attiré beaucoup d'attention dans de nombreuses disciplines d'ingénierie en raison de leur nature complémentaire.

Les réseaux de neurones flous (Fuzzy Neural Networks) combinent alors les réseaux de neurones et les systèmes flous. La méthode choisie d'entraînement des réseaux de neurones permet à

ces systèmes d'apprendre par l'ajustement de leurs paramètres (les poids de connexion) en donnant un paquet de données (Entrées/Sorties) d'entraînement.

III. MODELES D'HYBRIDATION NEURO-FLOU:

Il existe quatre catégories de combinaisons des réseaux de neurones avec la logique floue.

1. Systèmes neuro-flous concurrents:

Les réseaux de neurones et les systèmes flous collaborent sur la même tâche sans qu'il y ait aucune influence de l'un sur l'autre; aucun des systèmes ne détermine les paramètres de l'autre. De tels réseaux servent à l'apprentissage des fonctions d'appartenances, au calcul de l'inférence, à la réalisation de la phase d'agrégation et de défuzzification. Ils peuvent réaliser l'extraction de règles floues, en analysant les corrélations qui existent entre les entrées et le sorties du réseau de neurones et explicitent alors les connaissances obtenues par un réseau de neurones en établissant une interface entre connaissances numériques et connaissances symboliques.

2. Réseaux Neuronaux flous

Dans ces réseaux, les techniques floues sont employées pour augmenter les possibilités du processus d'apprentissage et d'exécution des réseaux de neurones. Ceci s'effectue par la création d'un réseau de neurones, qui opère sur des données floues en entrée.

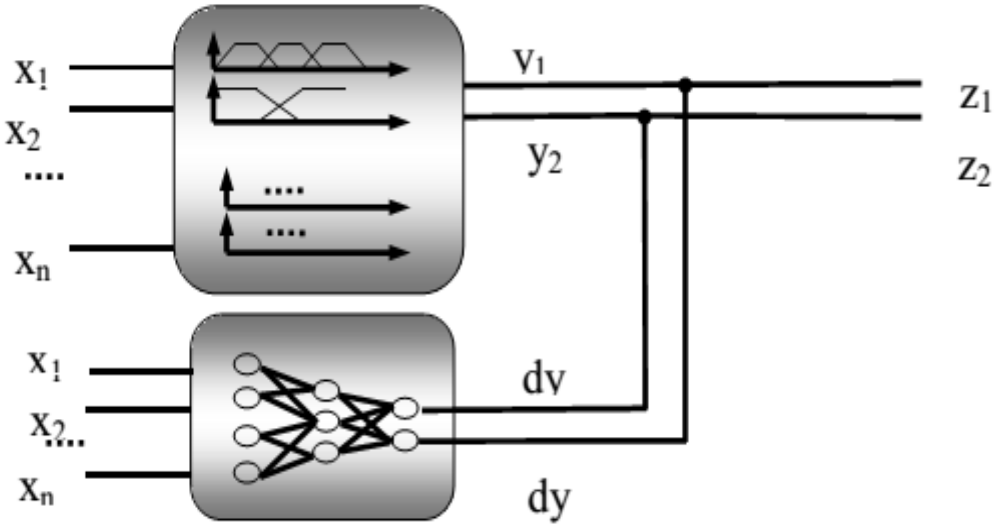


Figure (5.2): Réseaux de neuro-flou réalisé en parallèle

3. Systèmes neuro-flous en coopération:

Le réseau de neurones est employé pour déterminer les paramètres (les règles et les ensembles flous) d'un système flou. Après la phase d'apprentissage, le système flou fonctionne sans réseau de neurone. C'est une forme simple des systèmes neuro-flous.

Il existe plusieurs possibilités d'associations:

- un réseau de neurones qui fonctionne en amont d'un système flou. Les variables d'entrée d'un système de commande flou sont déterminées à partir de la sortie d'un réseau de neurones lorsqu'elles ne sont pas mesurables directement.
- un réseau de neurones effectuant une tâche de classification ou de reconnaissance de formes, qui est suivie par un système d'aide à la décision floue.
- un réseau de neurones qui fonctionne en aval d'un système flou par exemple dans le but d'ajuster les sorties d'un système de commande flou à de nouvelles connaissances obtenues, les variables de sortie étant les erreurs sur les variables de sortie du système flou.

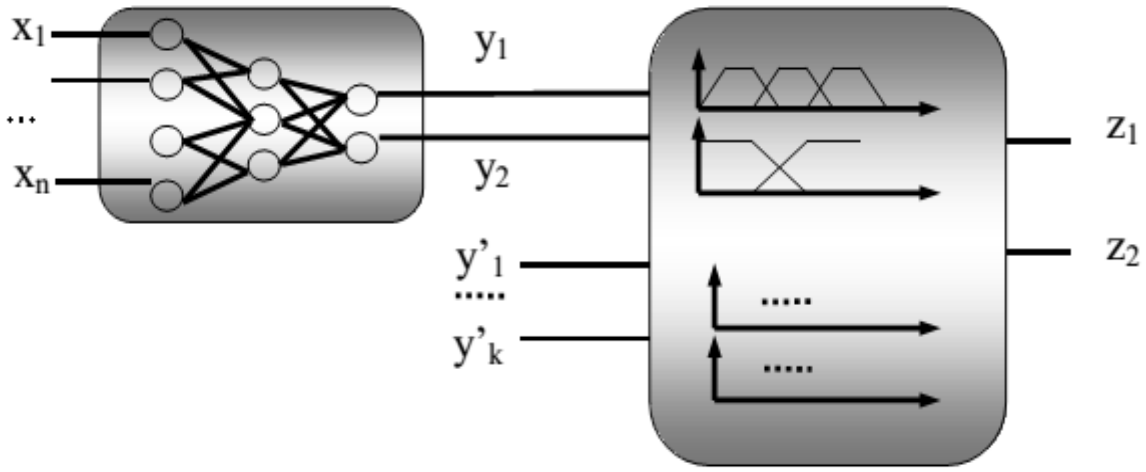


Figure (5.3): Réseaux de neuro-flou réalisé en série

4. <u>Systèmes neuro-flous hybrides:</u>

Les approches neuro-floues modernes sont de cette forme. Un réseau neuronal et un système flou sont combinés dans une architecture homogène. Le système peut être interprété comme un réseau neuronal spécial avec des paramètres flous ou comme un système flou mis en application sous une forme distribuée parallèle.

Le système d'inférence floue est mis sous la forme d'un réseau multicouche dans lequel les poids correspondent aux paramètres du système. L'architecture du réseau dépend du type de règles et des méthodes d'inférence, d'agrégation et de défuzzification choisie. Par exemple, pour des règles de la forme : « si V_1 est A_{11} et V_2 est A_{21} alors W est W_1», on fait appel à un réseau de neurones qui admet pour entrées les valeurs X_1 et X_2 prises par les variables V_1 et V_2 et dont les deux couches cachées correspondent respectivement au calcul de la valeur des fonctions d'appartenance de A_{11} pour l'entrée X_1 et de A_{21} pour X_2 et à celui de la valeur prise par la conjonction des conditions de chaque règle utilisant un opérateur de conjonction adéquat.

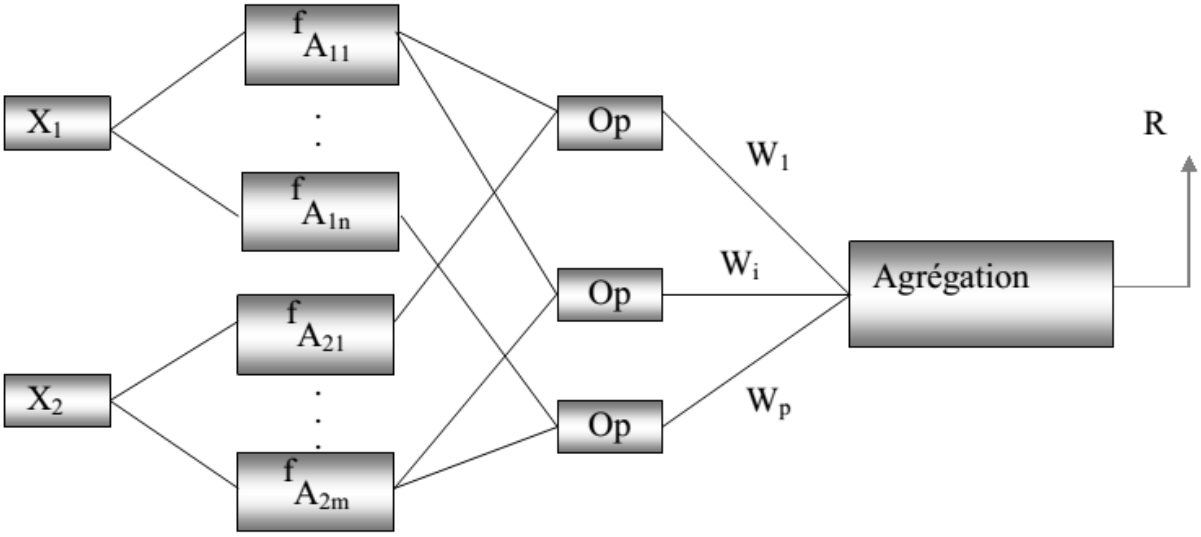


Figure (5.4): Réseaux de neuro-flou

5. <u>Les modèles neuro-flous hybrides:</u>

On peut classer les architectures Neuro-flous deux classes:

- Les modèles de réseaux neuro-flous à apprentissage supervisé
- Les modèles de réseaux neuro-flous à apprentissage par renforcement

a. ***Les modèles de réseaux neuro-flous à apprentissage supervisé:*** On peut citer les archotectures:

ANFIS: Adaptive Network Fuzzy Inference System.

NNDFR: Neuronal Network Driven Fuzzy Reasoning.

NEFCLASS: Neuro Fuzzy CLAASSification.

NEFPROX: Neuro-Fuzzy Function Proximation.

FuNe: Neuro Fuzzy model.

FRN: Fuzzy Rule Net.

b. ***Les modèles de réseaux neuro-flous à apprentissage par renforcement:*** On peut citer les architectures:

ARIC: Approximation Reasoning-based Intelligent Control.

NEFCON: Neuro-Fuzzy CONtroller.

Dans ce qui suit, on va intéresser par le système d'inférence adaptatif neuro-flous (ANFIS).

6. Système d'inférence adaptatif neuro-flous (ANFIS):

ANFIS est un réseau neuro-flou typique, utilisant le système d'inférence flou de Takagi et Sugeno avec 5 couches.

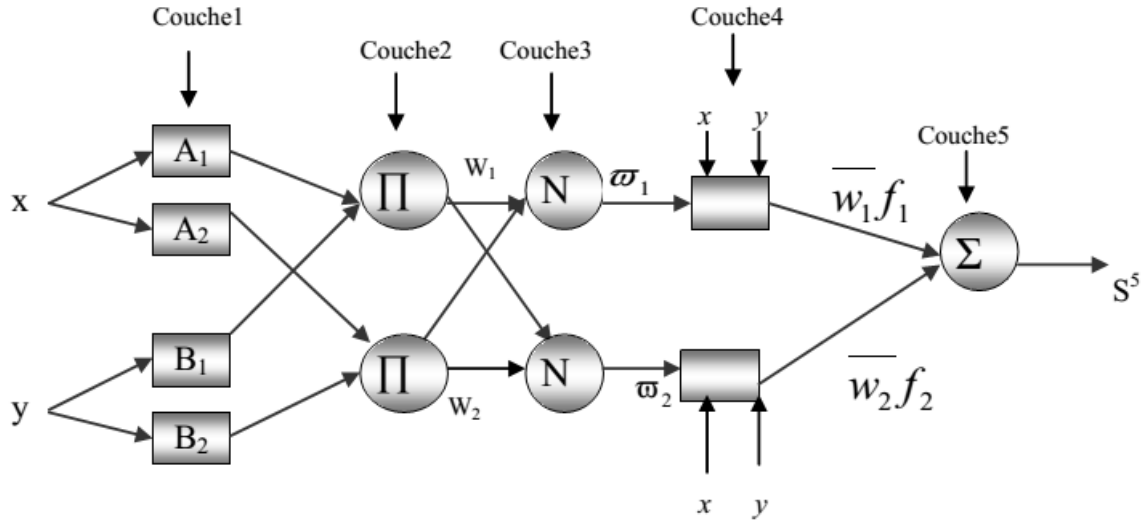


Figure (5.5): Modèle ANFIS

Couche1: Les nœuds expriment les conditions d'appartenance, ils remplissent le rôle de fuzzification. Chaque nœud est plus ou moins activé en fonction de degré d'appartenance à un sous ensemble flou donné (le nombre de nœuds associés à chaque entrée est le nombre de sous ensembles flous qui définissent cette variable floue).

$$S_i^1 = \mu_{A_i}(x): \textbf{est la fonction d'appartenanceau sous ensemble } A_i$$

$$S_i'^1 = \mu_{B_i}(y): \textbf{est la fonction d'appartenanceau sous ensemble } B_i$$

Avec la fonction d'appartenance est de type Gaussienne:

$$\mu_{A_i}(x) = e^{-\frac{1}{2}\left[\frac{x-c_i}{S_i}\right]^2}$$

Couche2: Les nœuds expriment les règles d'inférences. Les liaisons de cette couche remplissent le rôle de la partie pré-condition des règles.

$$S_i^2 = w_i = \mu_{A_i}(x)\mu_{B_i}(y) \;\; i = 1,2$$

Couche3: Chaque nœud i dans cette couche correspond à une règle floue Sugeno. Il reçoit les sorties des neurone de fuzzyfication et calcule son activation en fonction de la sortie précédente w_i comme suit:

$$S_i^3 = \overline{w_i} = \frac{w_i}{\sum_i w_i}$$

Couche 4: Les liaisons de cette couche remplissent le rôle de la partie conséquence des règles, chaque nœud i calcule le degré de vérité normalisé d'une règle floue donnée. La valeur obtenue

représente la contribution de la règle floue au résultat final. Ainsi la sortie du neurone ***i*** de la couche 4 est calculée en fonction des variables d'entrées comme suit :

$$S_i^4 = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i)$$

Couche 5: La sortie est obtenue en sommant l'ensemble des sorties de la couche 4.

$$S_i^5 = f = \sum_i \overline{w_i} f_i$$

7. Entraînement d'un réseau ANFIS:

Le modèle ANFIS utilise généralement l'algorithme de rétro-propagation pour modifier les paramètres des fonctions d'appartenance (centre et largeur de la fonction gaussienne) afin de minimiser l'erreur entre la sortie du système calculée et la sortie désirée.

C'est un algorithme à deux temps où on estime d'abord les paramètres des conséquents par une technique de moindres carrés et ensuite les poids du réseau par une descente de gradient. Chaque époque d'entraînement comprend un passage en avant et un passage en arrière. Durant le passage en avant, les patrons d'entrée servent à déterminer les sorties des neurones couche par couche, permettant de déterminer les valeurs des paramètres de conséquents en bout de compte. Durant le passage en arrière, l'algorithme de rétro-propagation d'erreur est appliqué pour régler les poids des différentes couches, avec l'objectif de minimiser l'erreur entre la valeur calculée et celle désirée :

$$e = \frac{1}{2}\sum (f - out_d)^2$$

CHAPITRE VI: ALGORITHMES DE CALCUL EVOLUTIONNAIRE ET D'INTELLIGENCE COLLECTIVE

I. DEFINITIONS:

1. *Les algorithmes évolutionnaires (AE):*

Les principes de base des algorithmes évolutionnaires (AE), dont les plus connus sont les algorithmes génétiques (AG), sont directement inspirés de la théorie de l'évolution selon Darwin. Ces méthodes de résolution de problèmes, d'optimisation stochastique, copient de façon très simplifiée la capacité de populations d'organismes vivants à s'adapter à leur environnement à l'aide de mécanismes de sélection et d'héritage génétique.

2. *L'intelligence collective:*

L'intelligence collective (Swarm Intelligence en Anglais) désigne les capacités cognitives d'une communauté résultant des interactions multiples entre les membres (ou agents) de la communauté. Des agents, au comportement très simple, peuvent ainsi accomplir des tâches complexes grâce à un mécanisme fondamental appelé synergie. Sous certaines conditions particulières, la synergie créée, par la collaboration entre les individus, fait émerger des possibilités de représentation, de création et d'apprentissage supérieures à celles des individus isolés. Les formes d'intelligence collective sont très diverses selon les types de communauté et les membres qu'elles réunissent. Les systèmes collectifs sont en effet plus ou moins sophistiqués.

L'intelligence collective est observée notamment chez les insectes sociaux (fourmis, termites et abeilles) et les animaux en mouvement (oiseaux migrateurs, bancs de poissons). En conséquence, plusieurs algorithmes basés sur le principe d'intelligence collective ont été introduits : on peut citer les colonies de fourmis, les essaims particulaires et les colonies des abeilles.

II. LES ALGORITHMES GENETIQUES:

1. *DEFENITION:*

Les Algorithmes Génétiques (algorithmes évolutionnistes) sont des techniques d'optimisation stochastiques basés sur les principes de la sélection naturelle génétique. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable.

Un AG permet une population composée de nombreux individus d'évoluer selon des règles de sélections spécifiées à un état qui maximise la «fitness» (où minimise la fonction de coût). La méthode a été développée par John Holland **(1975)** au cours des années **1960** et **1970** puis popularisée par un de ses élèves, **David Goldberg** en **(1989).**

2. *PRINCIPE DES ALGORITHMES GENETIQUES :*

Le but de ces algorithmes génétiques est d'optimiser une fonction prédéfinie, appelée fonction objectif, ou fitness; ils travaillent sur un ensemble de solutions candidates, appelé "population" d'individus ou chromosomes, chaque chromosome représente le codage d'une solution potentielle au problème à résoudre.

Une première population est choisie soit aléatoirement, soit par des heuristiques ou par des méthodes spécifiques au problème, soit encore par mélange de solutions aléatoires et heuristiques. Cette population doit être suffisamment diversifiée pour que l'algorithme ne reste pas bloqué dans un optimum local.

A chaque itération, appelée génération, une nouvelle population est crée avec le même nombre de chromosomes.

La création d'une nouvelle population à partir de la précédente se fait par application des opérateurs génétiques que sont : la sélection, le croisement et la mutation. Ces opérateurs sont stochastiques.

Pour mettre en œuvre un algorithme génétique, il est nécessaire de disposer :

• une représentation génétique du problème, c'est-à-dire un codage de solutions utilisé sous la forme de chromosomes.

• un mécanisme de génération de la population initiale. Ce mécanisme est indispensable de construire une population d'individus non homogène.

• une fonction qui permet d'évaluer l'adaptation d'un chromosome à son environnement, ce qui offre la possibilité de comparer des individus. Cette fonction est construite à partir du critère que l'on désire optimiser. L'application de cette fonction à un élément de la population donne sa fitness ;

• un mode de sélection des chromosomes à reproduire. Cette sélection est basée sur la reproduction et sur le codage génétique, qui stocke les informations décrivant l'individu sous forme de gènes ;

• d'opérateurs de croisement et de mutation permettant de diversifier la population au cours des générations et d'explorer l'espace d'état.

• de paramètres qu'utilise l'algorithme : taille de la population, probabilité de croisement et de mutation, nombre total de générations.

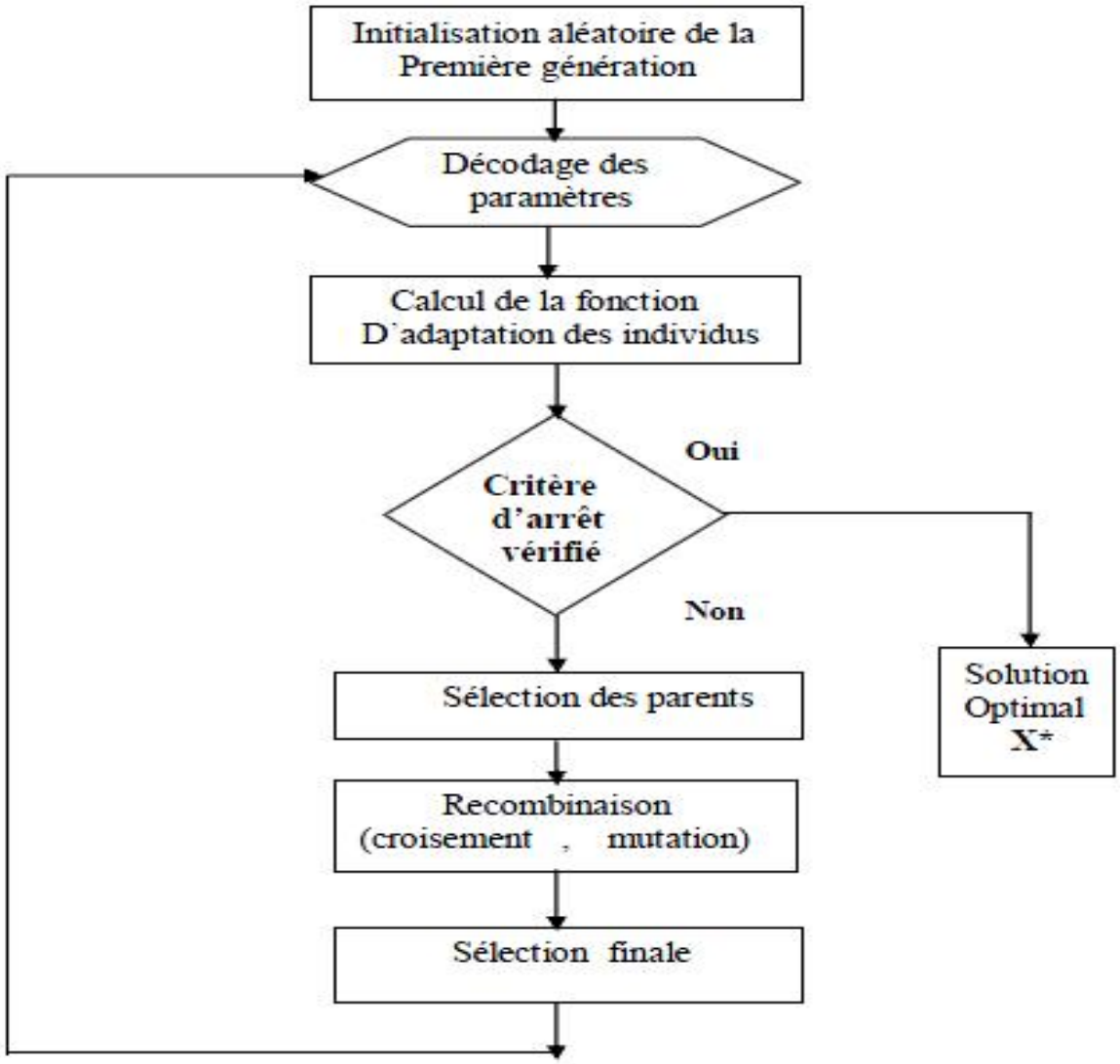
3. COMPARAISON ENTRE LES METHODES D'OPTIMISATION CLASSIQUES ET LES ALGORITHMES GENETIQUES :

Type	Techniques d'optimisation classiques	Algorithmes Génétiques
Applicabilité	Applicable à un domaine de problème particulier	Applicable à divers domaines problématiques
Nombre de points étant simultanément recherché	Un seul point	Multiples points
Transition depuis le point courant à l'autre	Déterministe	Probabiliste
Conditions préalables	Connaissances auxiliaire nécessaire	Une fonction de fitness à optimiser.
Estimation initiale	Fourni par l'utilisateur	Automatiquement généré par l'algorithme
Temps requis pour le calcul	Court	relativement long
La qualité des résultats	Optimum local dépend de l'estimation initiale	Optimum global est plus probable

4. MECANISME DE FONCTIONNEMENT D'UN ALGORITHME GENETIQUE:

Structure générale de l'algorithme génétique :

La structure de l'algorithme génétique est donnée par l'organigramme suivant:



a) <u>La création de la population (initialisation):</u>

Il faut créer une population à faire évoluer. Une population d'individus (chromosomes) est générée. Ces nombreuses solutions individuelles sont générées aléatoirement selon une distribution uniforme pour former une population initiale. La taille de la population dépend sur la nature du problème mais contient typiquement plusieurs centaines ou des milliers de solutions possibles.

b) <u>Le codage:</u>

Le codage est une partie très importante des algorithmes génétiques. Il permet de représente l'individu sous la forme d'un chromosome. A chaque variable d'optimisation x_i correspond un gène. Un chromosome sera donc un ensemble de gènes, chaque point est représenté par un individu doté d'un génotype constitué d'un ou de plusieurs chromosomes. La population est un ensemble de N individus qui vont évoluer d'une génération à une autre.

On peut distinguer deux types de codages les plus utilisables, et on peut passer de l'un à l'autre relativement facilement :

- ***Codage binaire:***

C'est le plus utilisé, Chaque gène dispose du même alphabet binaire **{0, 1}**. Les individus sont représentés sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace. Ce type de codage a pour intérêt de permettre la création d'opérateurs de croisement et de mutation simples.

- ***Codage réel :***

Davis Janikow et Michalewicz ont effectué une comparaison entre la représentation binaire et la représentation réelle. Ces auteurs ont trouvé que la représentation réelle donne de meilleurs résultats d'après leur problème à résoudre. Dans ce codage l'individu est un vecteur réel. Cette représentation est aujourd'hui très utilisée dans les problèmes d'optimisation car dans de nombreuses applications du monde réel, ces problèmes sont naturellement formulés sous forme paramétrique.

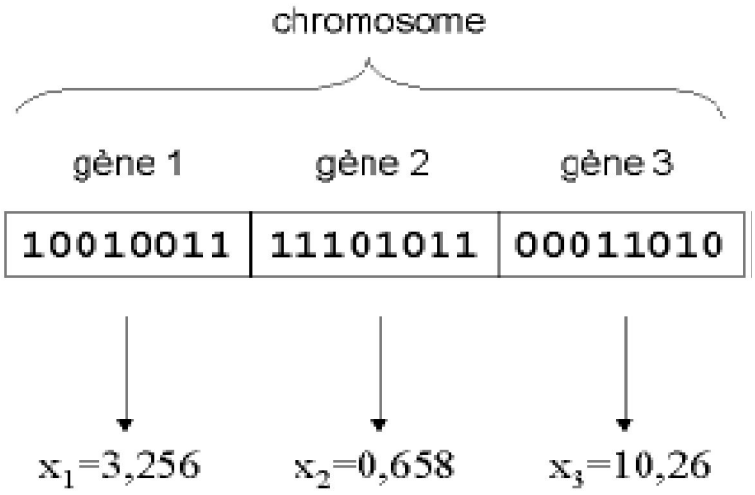


Illustration schématique du codage des variables réelles

c) Fonction d'évaluation:

Une fonction d'évaluation est utilisée pour mesurer séparément les performances de chaque individu, qui correspond à une solution donnée du problème à résoudre. Cette fonction permet d'évaluer la capacité d'un individu à survivre en lui affectant un poids appelé fitness.

La fonction d'adaptation peut être soit mono critère ou multicritère. Une fonction d'adaptation mono critère signifie que la fonction dépend d'une seule fonction objective par contre la fonction d'adaptation multicritère dépend d'une combinaison de plusieurs fonctions objectives d'où la notion objective unique et objectives multiples.

La fonction d'adaptation doit exprimer le plus fidèlement possible, la problématique posée sous forme mathématique. Sa définition peut être simplement analytique, ou elle peut éventuellement faire appel au jugement de l'utilisateur.

d) La sélection:

La sélection des parents a pour but de calculer les individus de la population courante ***P*** qui seront autorisés à se reproduire afin de donner une nouvelle génération ***P'*** plus convenable et optimal que la génération précédente. Les individus les mieux adaptés sont sélectionnés alors que les moins adaptés sont écartés. La sélection doit favoriser les meilleurs éléments selon le critère à optimiser.

Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. En règle générale, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population**.**

Il existe plusieurs méthodes pour la reproduction. On citera :

• La méthode de la "loterie biaisée" (roulette wheel) de GoldBerg,

• La sélection par tournois,

• La sélection universelle stochastique.

• La sélection à reste stochastique.

• La sélection par le rang.

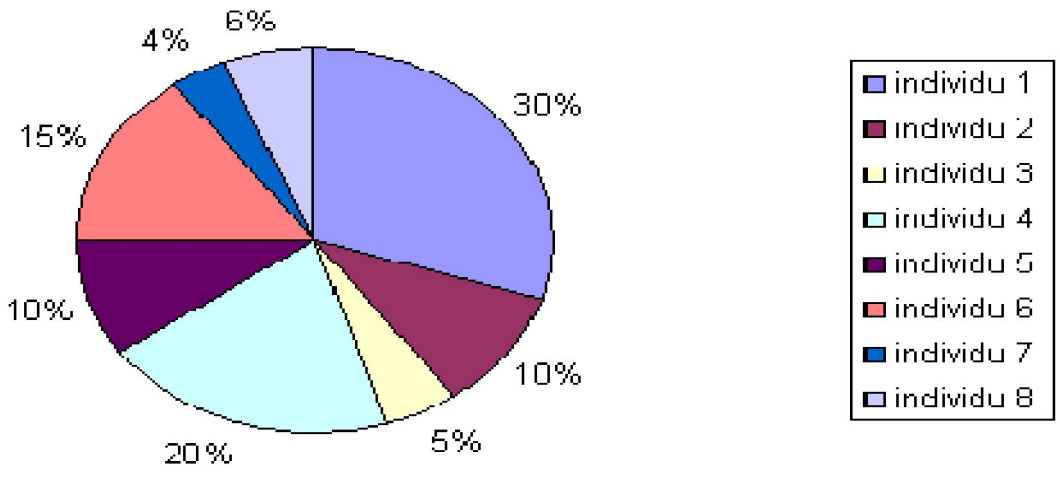
Les méthodes les plus utilisées sont :

- ***La sélection par roulette:***

Cette méthode est la plus connue et la plus utilisée. Avec cette méthode chaque individu a une chance d'être sélectionné proportionnelle à sa performance, donc plus les individus sont adaptés au problème, plus ils ont de chances d'être sélectionnés.

Plus simplement elle consiste à créer une roue de loterie biaisée pour laquelle chaque individu de la population occupe une sélection de la roue proportionnelle à sa valeur d'évaluation. On fait tourner la roue et quand elle cesse de tourner on sélectionne l'individu correspondant au secteur désigné.

Cette méthode n'assure pas la sélection des meilleurs individus et peut être une cause de la convergence prématurée.



- ***La Sélection par tournoi:***

Cette méthode est celle avec laquelle on obtient les résultats les plus satisfaisants.

Le principe de cette méthode est le suivant : on effectue un tirage avec remise de deux individus de ***P***, et on les fait "combattre". Celui qui a la fitness la plus élevée l'emporte avec une probabilité ***p*** comprise entre **0.5** et **1**. On répète ce processus ***n*** fois de manière à obtenir les ***n*** individus de ***P'*** qui serviront de parents.

Avec le tournoi binaire, sur deux individus en compétition, le meilleur gagne avec une probabilité ***p*** comprise entre **0.5** et **1**. Cette méthode choisit toujours une valeur de la fonction d'adaptation plus élevée par rapport à la technique de la roulette biaisée.

- ***La sélection uniforme :***

C'est une technique très simple qui consiste à sélectionner un individu aléatoirement de la population ***P***. La probabilité P_i pour qu'un individu soit sélectionné est définie par:

$$P_i = {}^{1}/_{taille\ de\ population}$$

e) La recombinaison génétique:

On distingue deux opérateurs principaux : Le croisement et La mutation qui permette d'explorer l'ensemble des solutions possibles. Ces opérations sont appliquées aléatoirement, à l'aide de deux paramètres, la probabilité de croisement et la probabilité de mutation. Ces probabilités sont des paramètres très importants, qui influent de façon considérable sur la convergence.

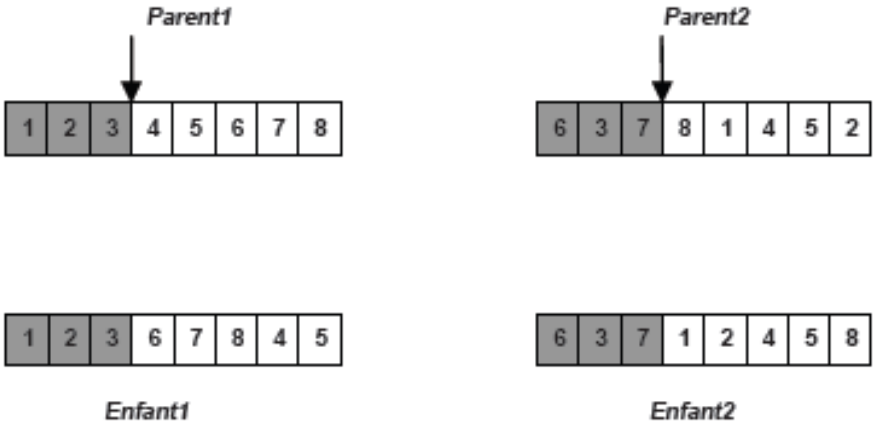
1) Le Croisement (ou crossover):

L'opérateur de croisement permet la création de nouveaux individus par combinaison des gènes des deux individus parents pour donner deux nouveaux chromosomes d'individus enfants (descendants) possédant des caractéristiques issues des deux parents. Il a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes.

Selon la littérature, plusieurs opérateurs de croisement sont proposés :

- ***Croisement en 1-point:***

Consiste à diviser chacun des deux parents en deux parties à la même position, choisie au hasard et à recopier la partie inférieure du parent à l'enfant et à compléter les gènes manquants de l'enfant à partir de l'autre parent en maintenant l'ordre des gènes.



- ***Croisement Arithmétique:***

Le croisement arithmétique combine linéairement deux vecteur des chromosomes parent pour produire deux nouveaux enfants selon (utilisé pour le codage réel) :

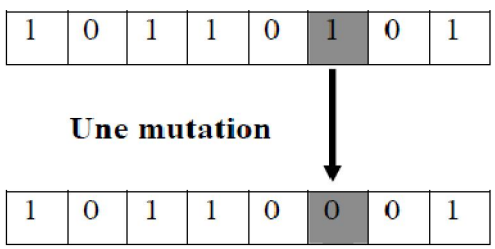
$$\boldsymbol{Enfant\ 1\ = \alpha * \ parent\ 1 + \ (1 - \alpha) * parent\ 2}$$
$$\boldsymbol{Enfant\ 2 = \ (1 - \alpha) * parent\ 1 + \alpha * parent\ 2}$$

Où : $\boldsymbol{\alpha}$ est un facteur aléatoire du poids choisi avant chaque opération du croisement.

2) Mutation:

Comme les individus les mieux adaptés sont les plus susceptibles d'être choisis lors de la sélection, la perte de certains gènes est inévitable avec le temps. La mutation est l'operateur qui permet d´éviter la dégénérescence de la population et d'enrichir le pool de gènes. Cette dégénérescence peut se traduire par une convergence des individus vers un optimum local, Elle permet d'assurer une recherche aussi bien globale que locale et garantit la convergence vers l'optimum.

L'une des plus efficaces est la mutation uniforme qui consiste à muter chaque paramètre de la fonction à optimiser avec une probabilité dépendant des informations contenues dans les gènes des individus. Dans le cas du codage binaire, chaque bit est remplacé selon une probabilité $\boldsymbol{P_m}$ par son inverse.



Comme pour le croisement, de nombreuses méthodes de mutation ont été présentées dans la littérature. Dans le cas d'un codage réel, on peut citer :

- ***La mutation uniforme:***

Remplace la valeur de gène (aléatoirement choisi) par une valeur aléatoire choisi entre les bornes spécifies (par l'utilisateur) $[x_{max}x_{min}]$.

- ***La mutation Gaussienne:***

Un opérateur de mutation qui ajoute une unité (aléatoire) F tiré d'une distribution gaussienne (de moyenne nulle et une variance adaptative) du gène choisi. Le gène résultant est tronquée s'il tombe en dehors des limites inférieure ou supérieure spécifiées par l'utilisateur pour ce gène.

$$x' = x + F$$

Exemple:

Le but est de trouver le maximum de la fonction: $f(x) = x$ sur l'intervalle **[0, 31]** où x est un entier.

Le codage utilisé est un codage binaire, chaque individu (x) est codé sur **5** bits. Nous recherchons donc le maximum d'une fonction « Fitness » dans un espace de **32** valeurs possibles de x. Dans notre exemple, nous allons fixer la taille de la population à $N = 4$. Nous tirons donc de façon aléatoire **4** chromosomes sachant qu'un chromosome est composé de **5** bits.

La population initiale est la suivant:

Numéro	Individu	Fitness	% du total
1	00101	5	14.3
2	10000	16	45.7
3	00010	2	5.7
4	01100	12	34.3
Total		35	100

Une nouvelle population est trouvée après l'application d'une sélection par roue de loterie biaisée. La nouvelle population est décrite dans le tableau suivant:

Numéro	Individu
1	10000
2	01100
3	00101
4	10000

Les parents sont sélectionnés au hasard. Nous choisissons aléatoirement un site de croisement. Le tableau ci-dessous donne les conséquences de cet opérateur en supposant que les chromosomes **1** et **3**, puis **2** et **4** sont appariés et qu'à chaque fois le croisement s'opère (par exemple avec $p_c = 1$).

	l = 3	l = 2
Les parents	**100|00** **001|01**	**01|100** **10|000**
Les enfants (après croisement)	**10001** **00100**	**01000** **10100**

Dans cet exemple à codage binaire, la mutation est la modification aléatoire occasionnelle (de faible probabilité) de la valeur d'un bit (inversion d'un bit). Nous tirons ainsi pour chaque bit un chiffre aléatoire (p) entre **0** et **1** et si ce chiffre est inférieur à pm ($p < p_m$) alors la mutation s'opère. En choisissant une probabilité de mutation $p_m = 0,05$, nous obtenons la population après mutation:

Ancien chromosome	Tirage aléatoire	Nouveau chromosome
10001 **00100** **01000** **10100**	**10001** **00100** **01000** **10100**	**10011** **00100** **01000** **11100**

L'évaluation de la nouvelle population:

Numéro	Individu	Fitness	% du total
1	**10011**	**19**	**32.2**
2	**00100**	**4**	**6.8**
3	**01000**	**8**	**13.6**
4	**11100**	**28**	**47.5**
Total		**59**	**100**

Le maximum est maintenant de **28.** Donc, nous sommes passés de **16** à **28** après une seule génération. Bien sûr, nous devons recommencer la procédure à partir de l'étape de sélection jusqu'à ce que le maximum global (**31**), soit obtenu, ou bien qu'un critère d'arrêt ait été satisfait.

5. Application:

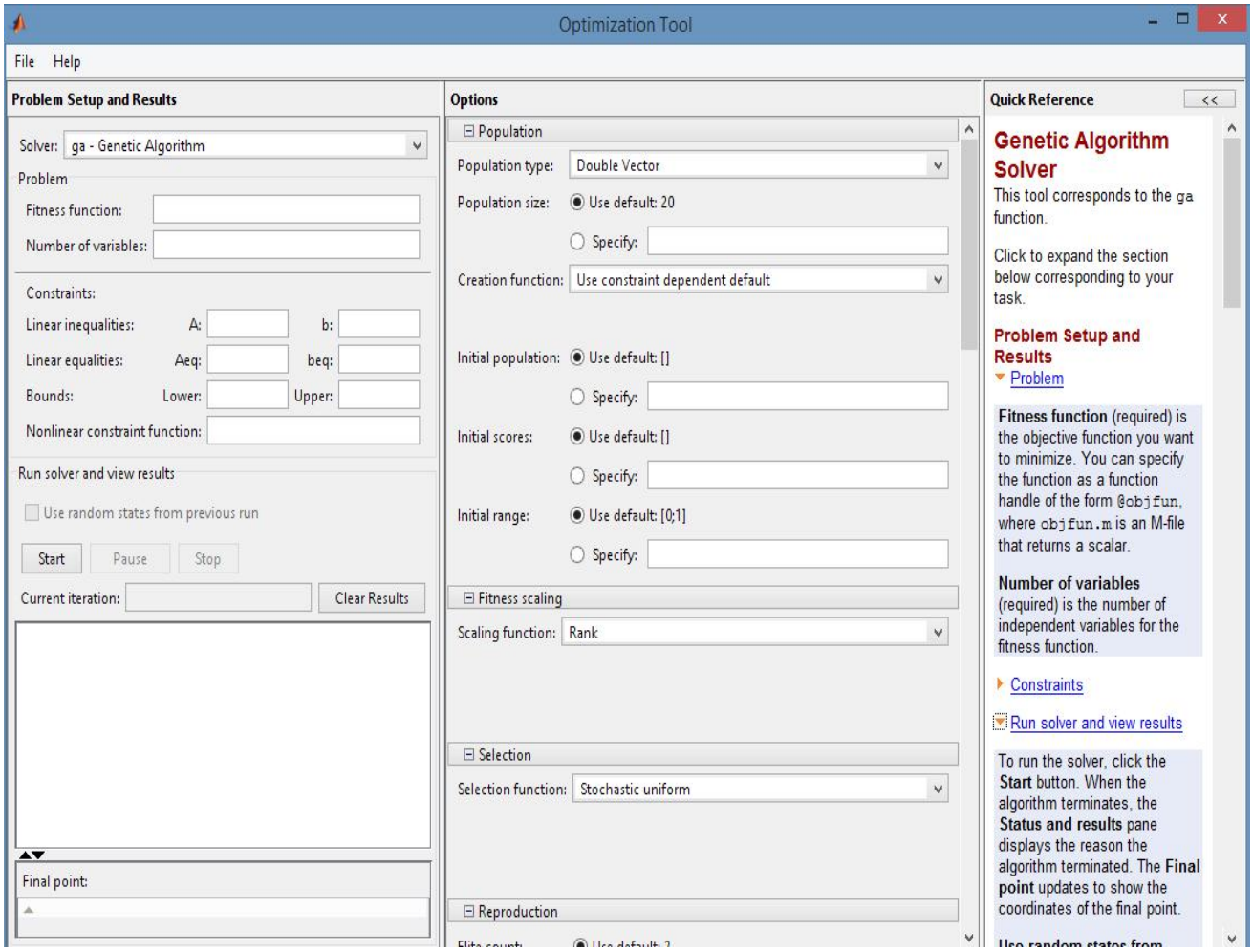
$F = 5* x_1 + 2.4* x_2 + 7.5* x_3 + 0.8* x_4$

F=10 : est la valeur optimale.

On cherche à trouver les valeurs de ***xi*** tel que : $0 \leq x_1, x_2, x_3, x_4 \leq 1$

■ **Fitness = |F - opt| = |F - 10| , Objectif: Min Fitness**

On utilise une simulation de l'AG en utilisant "optimtool".

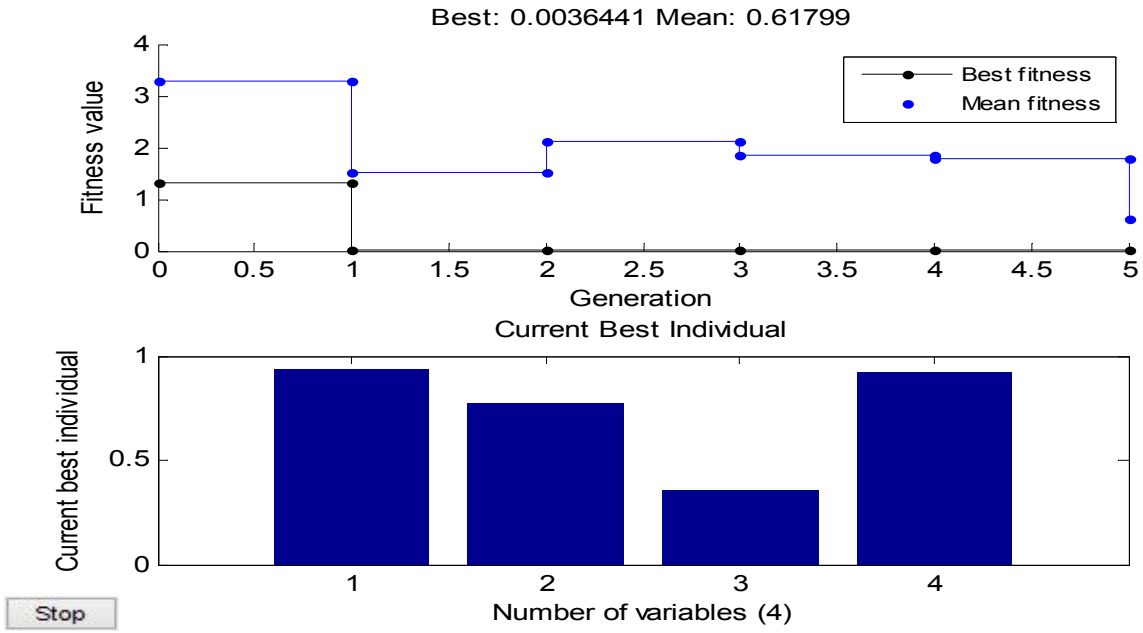



```
function f=fitness_exemple(x)

F=5*x(1)+2.4*x(2)+7.5*x(3)+0.8*x(4);

f=abs(F-10);
```

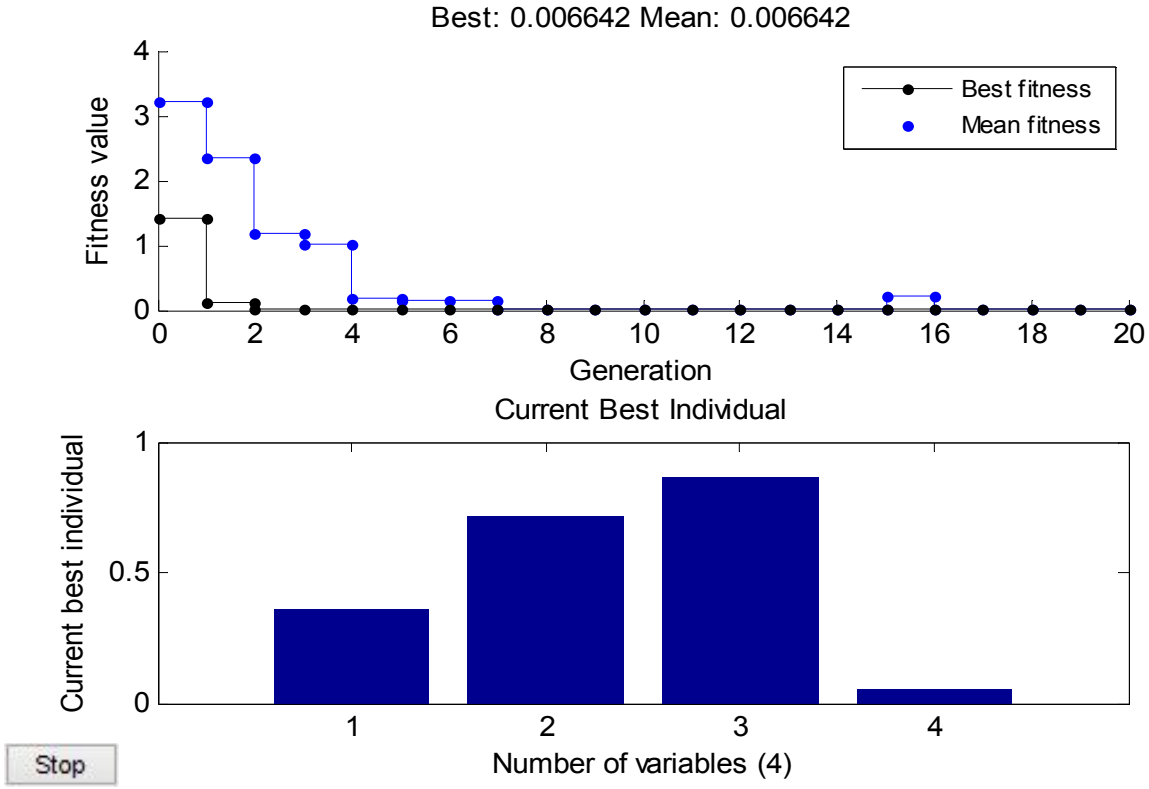
On obtient:



Le meilleur individu: x = [0.9417 0.7732 0.3601 0.9237]

avec: f= 0.0036 et F= 10.0036

En augmentant le nombre de générations, on obtient les résultats suivants:



X = [0.3606 0.7144 0.8598 0.0509]

f= 0.0066; F = 10.0066

- Minimisation de la fonction:

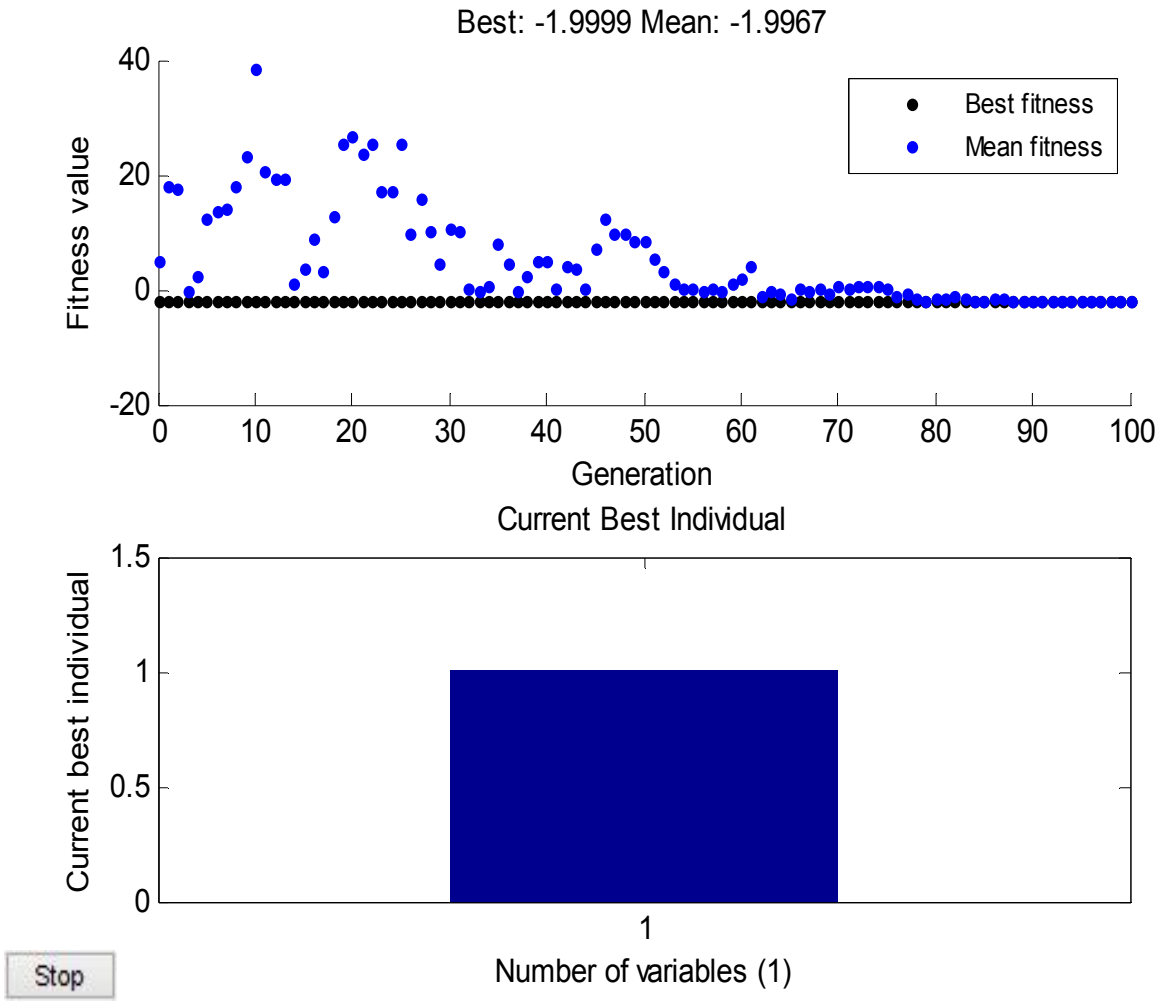
$$f(x) = 2x^2 - 4x, \qquad x \in [0; 5]$$

- Création de la fonction fitness:

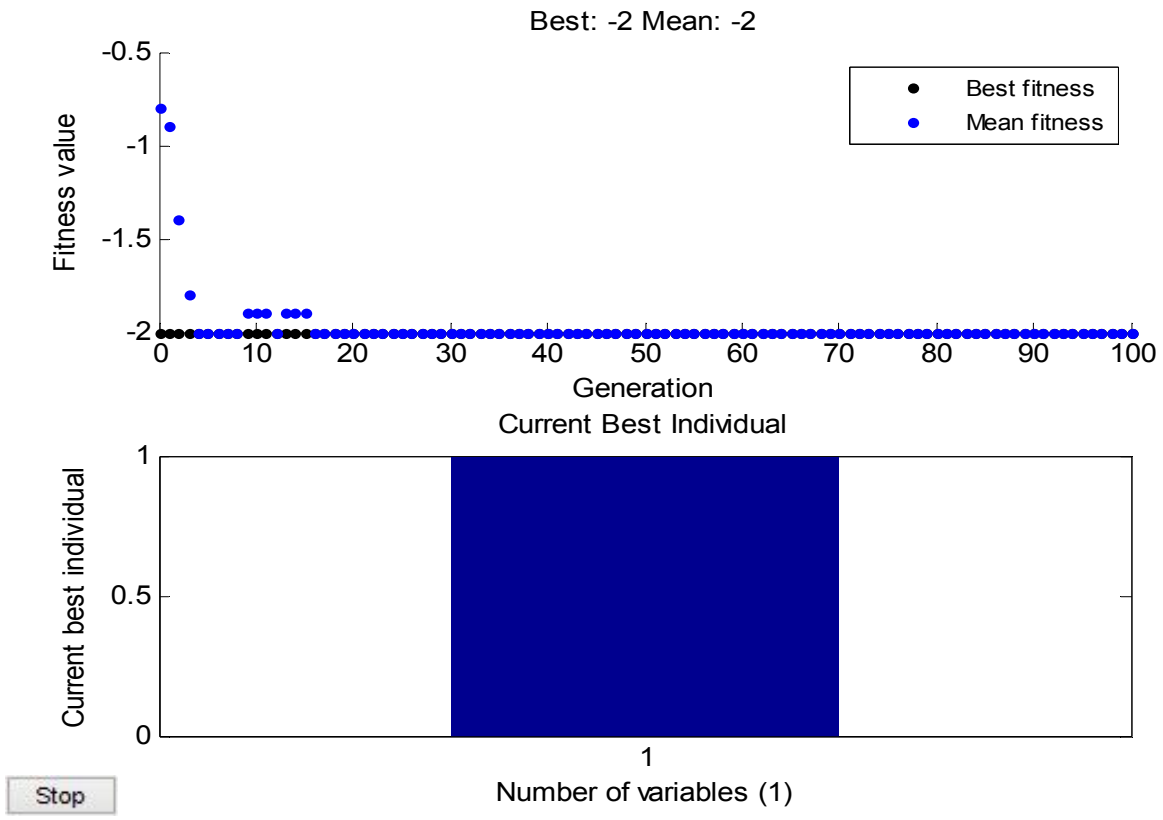
function [f, x] = fit_1(x)

F = 2*(x^2)-(4*x);

Codage réel:



Codage binaire:



- En utilisant la fonction «ga »

[f,x]=ga(@fit_1,1,[ ],[ ],[ ],[ ], 0, 5)

f =

1.0000

x =

-2.0000

On peut définir les paramètres de l'AG en utilisant la fonction: « gaoptimset » comme suit:

options=gaoptimset('populationtype','bitstring','generations',100)

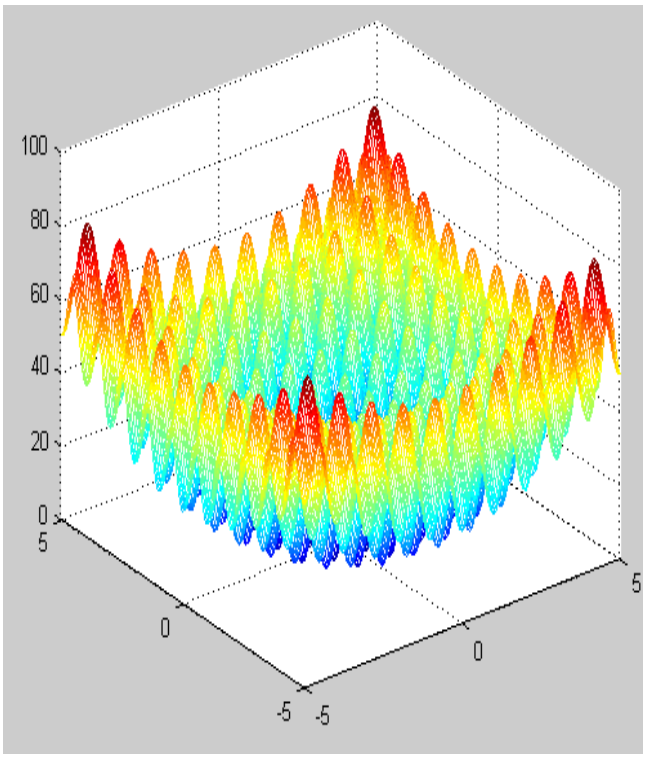
[f, x]=ga(@fit_1,1,[ ],[ ],[ ],[ ], 0, 5, [ ], options)

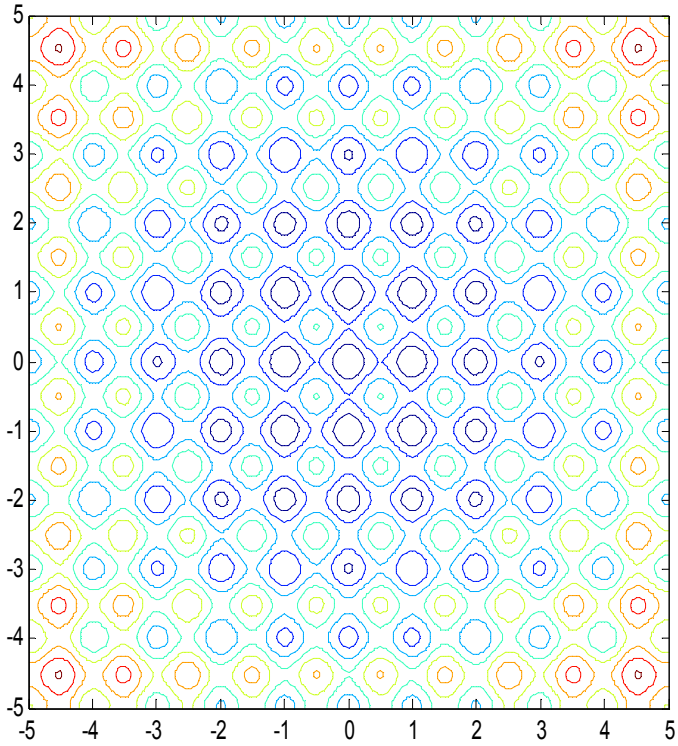
- La fonction ***Rastrigin***

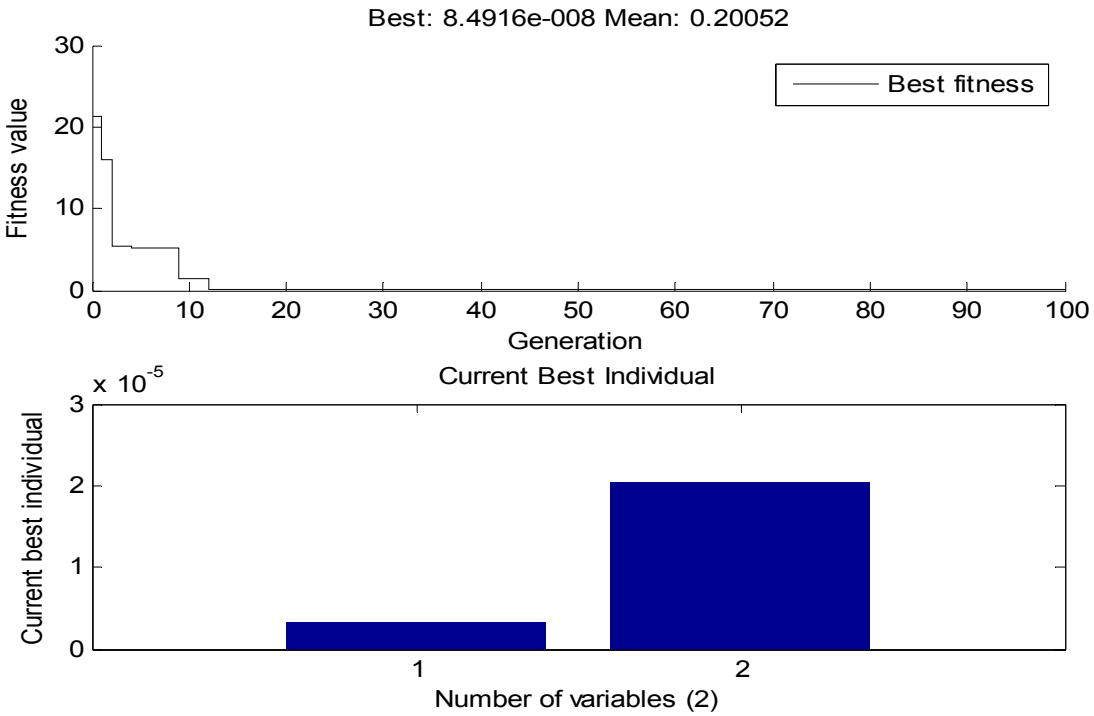
$$f(x) = 20 + {x_1}^2 + {x_2}^2 - 10(cos2\pi x_1 + cos2\pi x_2); \quad x_1 \in [-5;5] \; et \; x_2 \in [-5;5]$$

$$({x_1}^*, {x_2}^*) = (0,0) \; et \; f(x^*) = 0$$

L'allure de la fonction Rastrigin montre qu'elle contient plusieurs optimums locaux.





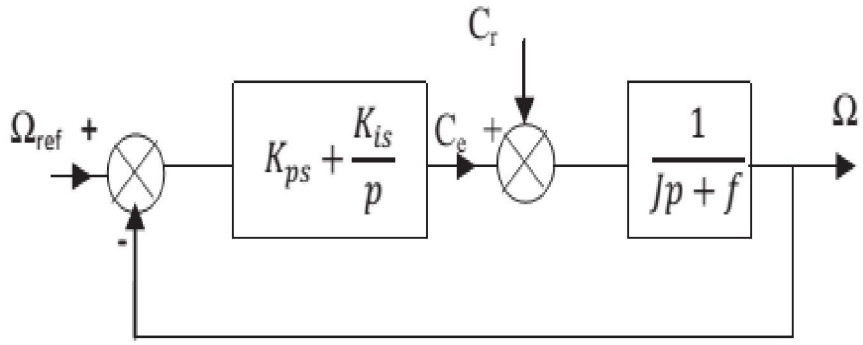


L'optimum obtenu est:

x = 1.0e-004 *[0.0313 0.2045]

***Application N°2:** Optimisation des paramètres d'un régulateur PI pour le contrôle de vitesse de rotation D'une MAS*

Le schéma bloc du régulateur de vitesse de rotation est représenté par la figure ci-dessous:

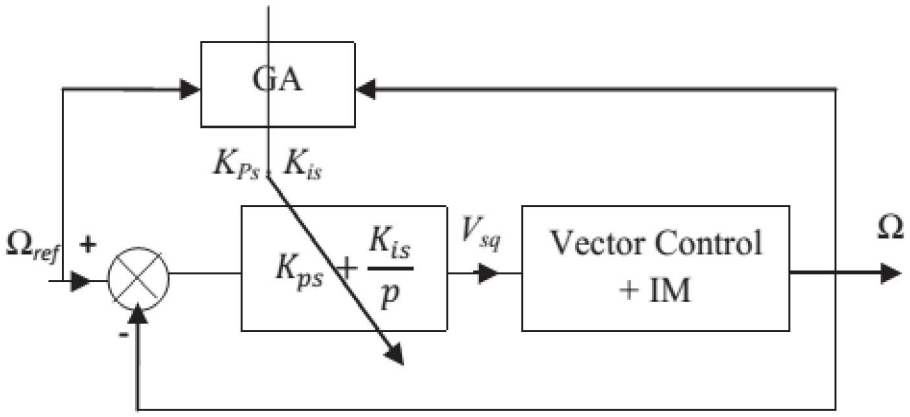


$$\begin{cases} K_{is} = J\omega_n{}^2 \\ K_{ps} = \dfrac{2\zeta}{\omega_n} K_{is} - f \end{cases}$$

L'optimisation des paramètres du régulateur PI de vitesse consiste à la détermination du coefficient du gain proportionnel (K_{ps}) et le coefficient du gain intégral (K_{is}) par minimisation de la fonction objective représentée par l'équation:

$$Fit = \int_0^{t_{sim}} e^2(t)dt = \int_0^{t_{sim}} \left(\Omega_{ref}(t) - \Omega(t)\right)^2 dt$$

Le principe d'optimisation des paramètres du régulateur PI par AG est expliqué par la figure ci-dessous:



Les paramètres du régulateur PI trouvé en utilisant l'AG sont mentionnés dans le tableau suivant:

Controller	K_P	K_i
Speed controller (Conventional PI)	0.5990	1.8000
Speed controller (PI-GA)	0.7656	1.3638

Application N°3: ***Optimisation des paramètres d'une MAS***

L'identification des paramètres de la machine asynchrone (MAS) consiste à la détermination du vecteur: ***X = [Rs Rr Ls Lr M]*** par minimisation de l'erreur quadratique moyenne entre les valeurs estimées (vit_c) et les valeurs mesurées (vit_m) de la vitesse de rotation mécanique, qui est représenté par l'équation:

$$MSE = Fit = \frac{1}{n}\sum_{i=1}^{n}[vit_m(i) - vit_c(i)]^2$$

Avec les valeurs mesurées de la vitesse de rotation mécanique (vit_m) sont obtenues par simulation du modèle de la machine par utilisation du schéma bloc développé par Simulink. Les valeurs des paramètres utilisées dans la simulation sont:

$$R_s = 1\ \Omega,\ R_r = 93\text{e-}3\ \Omega,\ L_s = 191\text{e-}3\text{H},\ L_r = 15.8\text{e-}3\text{H},$$
$$M = 52\text{e-}3\text{H},\ J = 50\text{e-}3\ \text{N.m.s}^2,\ P = 2.$$

Les résultats de simulation sont les suivants:

Method	Vector of parameters $X = [R_s\ R_r\ L_s\ L_r\ M]$	Fitness function
GA	1.005 Ω 0.095 Ω 0.189 H 0.016 H 0.052 H	1.034

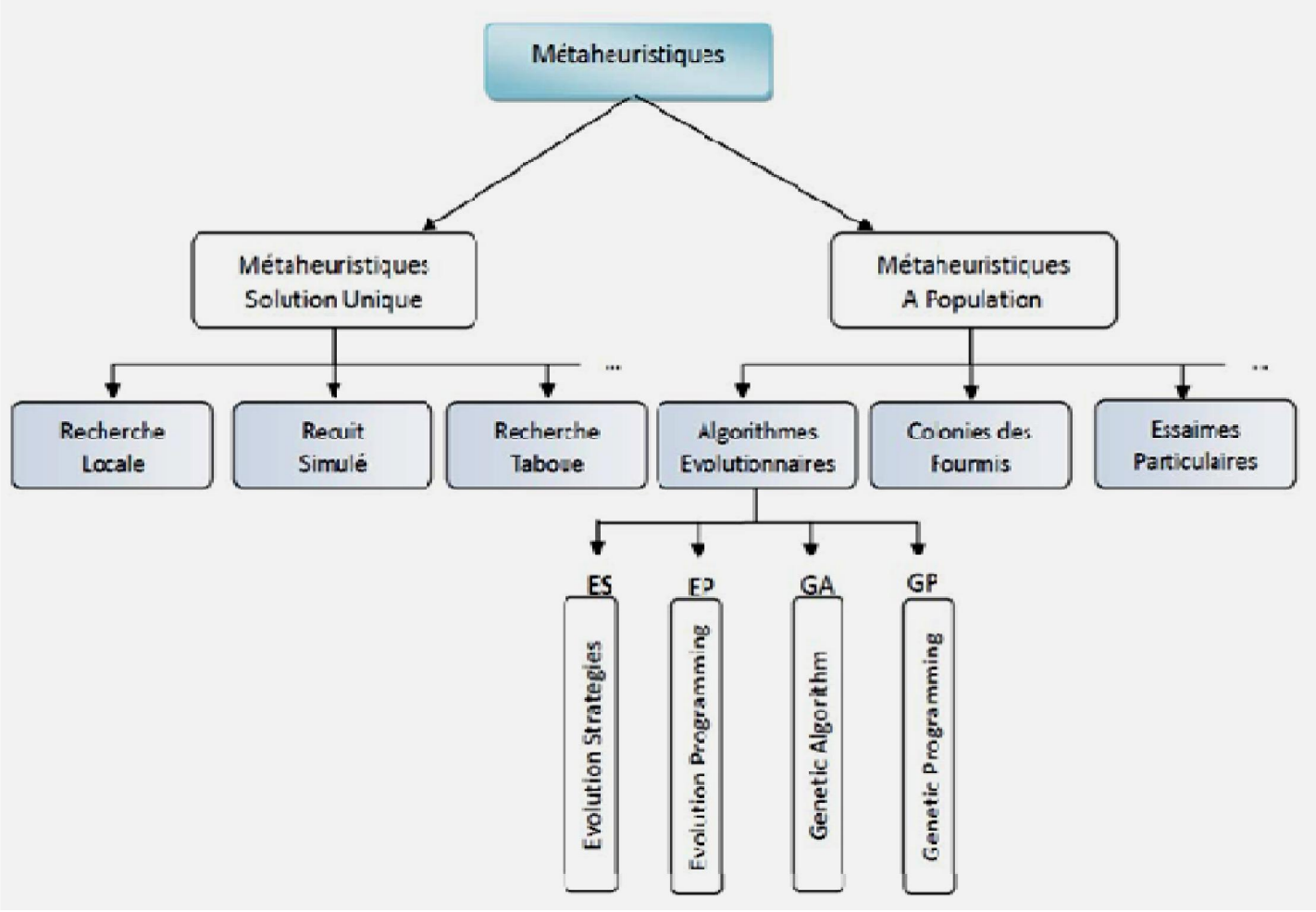
Les paramètres de la MAS obtenus après application de l'AG sont proches de ceux utilisés en simulation avec une erreur quadratique moyenne minimale.

III. ALGORITHMES D'OPTIMISATION PAR ESSAIMS DE PARTICULES:

Pour de nombreux problèmes, il n'existe pas de solution déterministe qui donne le résultat en un temps raisonnable, et ceci malgré la création d'ordinateurs de plus en plus performants. Pour pallier à ce problème, on a recours à des méthodes dites heuristiques, c'est-à-dire des méthodes qui fournissent une solution approchée. Toutefois, il faut reproduire le processus sur plusieurs itérations pour tendre vers une solution acceptable.

On retrouve parmi ces heuristiques, certains algorithmes qui possèdent un principe générique adaptable et qui s'applique donc à plusieurs problèmes d'optimisation. On les appelle des méta-heuristiques.

Classes des méta-heuristiques:



1. *DEFINITION:*

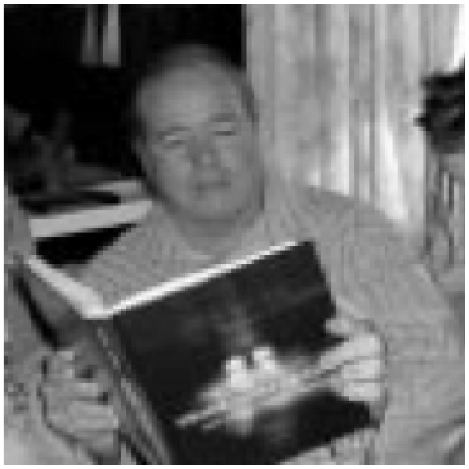
En 1995, Russel Eberhart, ingénieur en électricité et James Kennedy, socio-psychologue, s'inspirent du monde du vivant pour mettre en place une méta-heuristique : l'optimisation par essaim

particulaire (l'OEP). Cette méthode se base sur la collaboration des individus entre eux : chaque particule se déplace et à chaque itération, la plus proche de l'optimum communique aux autres sa position pour qu'elles modifient leur trajectoire. Cette idée veut qu'un groupe d'individus peu intelligents puisse posséder une organisation globale complexe.

L'optimisation par essaim particulaire, s'inspire fortement des relations grégaires des oiseaux migrateurs qui doivent parcourir des longues distances et qui doivent donc optimiser leurs déplacements en termes d'énergie dépensée.



Russell Eberhart



James Kennedy

Cette technique est souvent décrite comme une sorte d'algorithme évolutionnaire, avec une population d'individus (les agents), dans laquelle, à chaque pas de temps, les « meilleurs » (selon un critère prédéfini) sont plus ou moins imités par les autres. Un aspect essentiel, qui la différencie, par exemple, des algorithmes génétiques classiques, est l'existence d'une mémoire, à laquelle ne contribuent que les meilleurs éléments. Le modèle est censé expliquer et reproduire certains comportements sociaux, mais s'est révélé un peu simpliste.

Par contre, convenablement modifié, il peut résoudre des problèmes d'optimisation, car le processus fait apparaître des individus de plus en plus performants : c'est ainsi que l'OEP est née, sous le nom, donc, de Particle Swarm Optimization (PSO), grâce à la collaboration de deux personnes de profils très différents, Russ Eberhart et Jim Kennedy.

2. ***LES ELEMENTS DE LA PSO:***

Pour appliquer la PSO il faut définir un espace de recherche constitué de particules et une fonction objective à optimiser. Le principe de l'algorithme est de déplacer ces particules afin qu'elles trouvent l'optimum.

Chacune de ces particules est dotée :

- D'une **position**, c'est-à-dire ses coordonnées dans l'ensemble de définition.
- D'une **vitesse** qui permet à la particule de se déplacer. De cette façon, au cours des itérations, chaque particule change de position. Elle évolue en fonction de son meilleur voisin, de sa meilleure position, et de sa position précédente. C'est cette évolution qui permet de tomber sur une particule optimale.
- D'un voisinage, c'est-à-dire un ensemble de particules qui interagissent directement sur la particule, en particulier celle qui a le meilleur critère.

A tout instant, chaque particule connait :

- Sa meilleure position visitée. On retient essentiellement la valeur du critère calculée ainsi que ses coordonnées.
- La position du meilleur voisin de l'essaim qui correspond à l'ordonnancement optimal.
- La valeur qu'elle donne à la fonction objectif car à chaque itération il faut une comparaison entre la valeur du critère donnée par la particule courante et la valeur optimale.

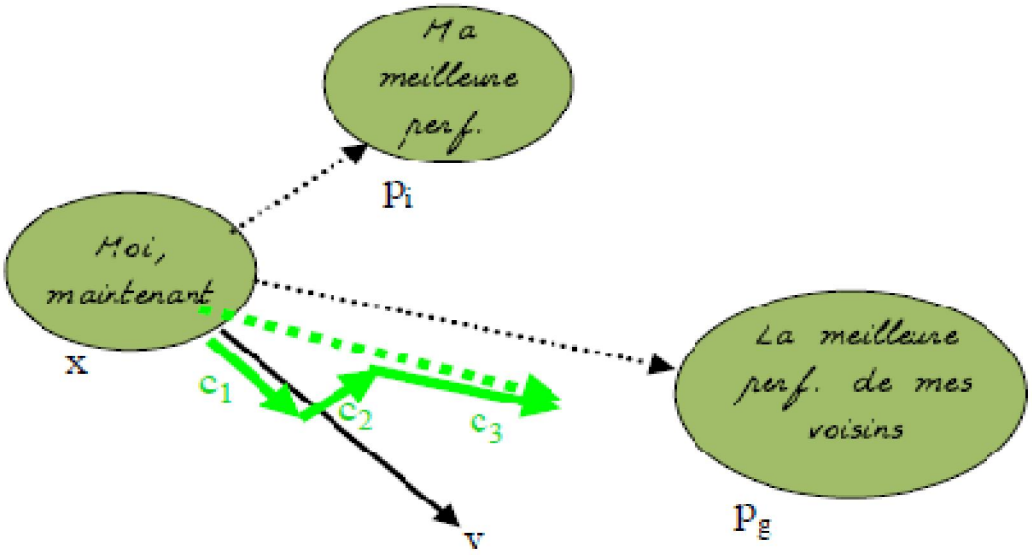
A partir des quelques informations dont elle dispose, une particule doit décider de son prochain mouvement, c'est-à-dire décider de sa nouvelle vitesse.

Pour ce faire, elle combine linéairement trois informations :

- sa vitesse actuelle
- sa meilleure performance
- la meilleure performance de ses voisines (ses informatrices).

À l'aide de trois paramètres parfois appelés *coefficients de confiance*, qui pondèrent trois tendances :

- tendance à suivre sa propre voie
- tendance conservatrice (revenir sur ses pas)
- tendance « panurgienne » (suivre le meilleur voisin)



3. *PRINCIPE FONDAMENTAL :*

Nous allons donc nous intéresser à cette méta-heuristique dans le cadre où elle a, avant tout, été définie à savoir l'optimisation de fonctions continues.

L'algorithme de base de la PSO travaille sur une population appelée *essaim* de solutions possibles, elles-mêmes appelées *particules*. Ces particules sont placées aléatoirement dans l'espace de recherche de la fonction objective.

A chaque itération, les particules se déplacent en prenant en compte leur meilleure position (déplacement égoïste) mais aussi la meilleure position de son voisinage (déplacement panurgien).

Dans les faits, on calcule la nouvelle vitesse à partir de la formule suivante :

$$\begin{cases} \boldsymbol{v(k+1) = c_1 v(k) + c_2\big(p_i - x(k)\big) + c_3(p_g - x(k))} \\ \boldsymbol{x(k+1) = x(k) + v(k+1)} \end{cases}$$

Où : v_{k+1} et v_k : sont les vitesses de la particule aux itérations k et $k+1$.

p_i : est la meilleure position de la particule

p_g : est la meilleure position de son voisinage à l'itération k

x : est la position de la particule à l'itération k

c_1, c_2, c_3 : sont des coefficients fixés, c_2 est généré aléatoirement à chaque itération et, en général,

$c_3 = c_2$

La nouvelle vitesse est calculée comme combinaison linéaire de trois éléments, comme vu auparavant, puis appliquée à la position courante pour donner la nouvelle position.

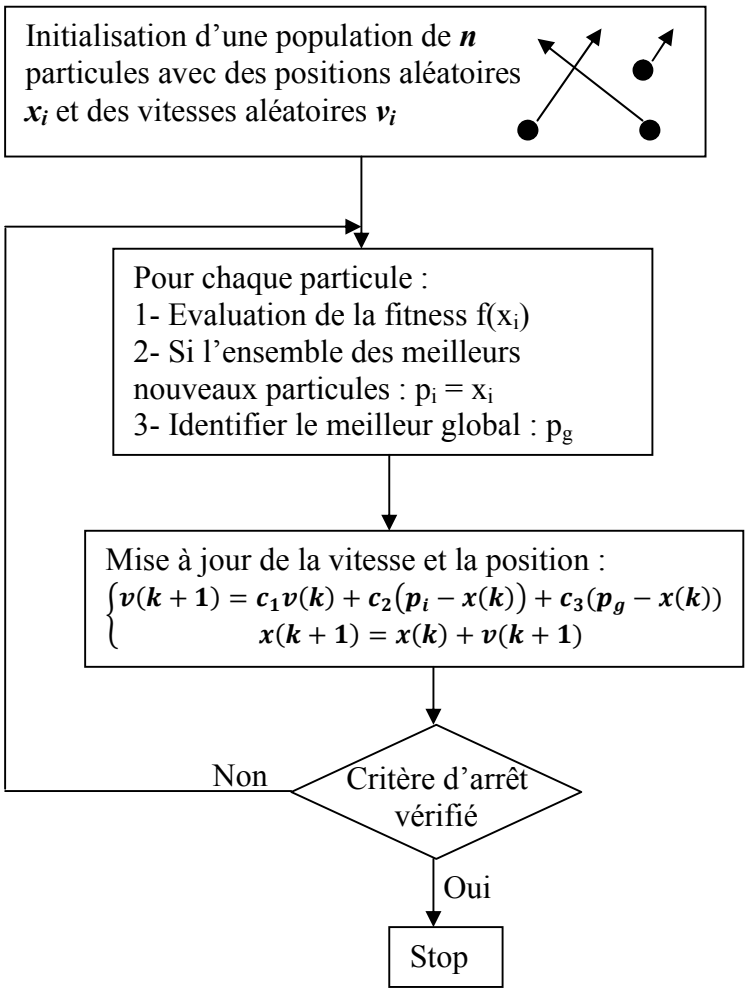
L'efficacité de la méthode tient en particulier au caractère aléatoire des deux derniers coefficients.

Algorithme de principe :

L'algorithme de base est très simple :

Initialisation aléatoire de l'essaim;
Calcul de la fonction Fitness de chaque particule;
$p_i = x_i$, calcul de p_g
Tant que le critère d'arrêt n'est pas satisfait faire
  Répéter
    Mise à jour des vitesses et des positions
    Calcul de la fitness de chaque particule
    Mise à jour des p_i, i=*1….N* et mise à jour de p_g
  Jusqu'à la création d'un résultat suffisant ;
Fin Tant que

Ou l'organigramme correspondant:



IV. ALGORITHME DES COLONIES DE FOURMIS:

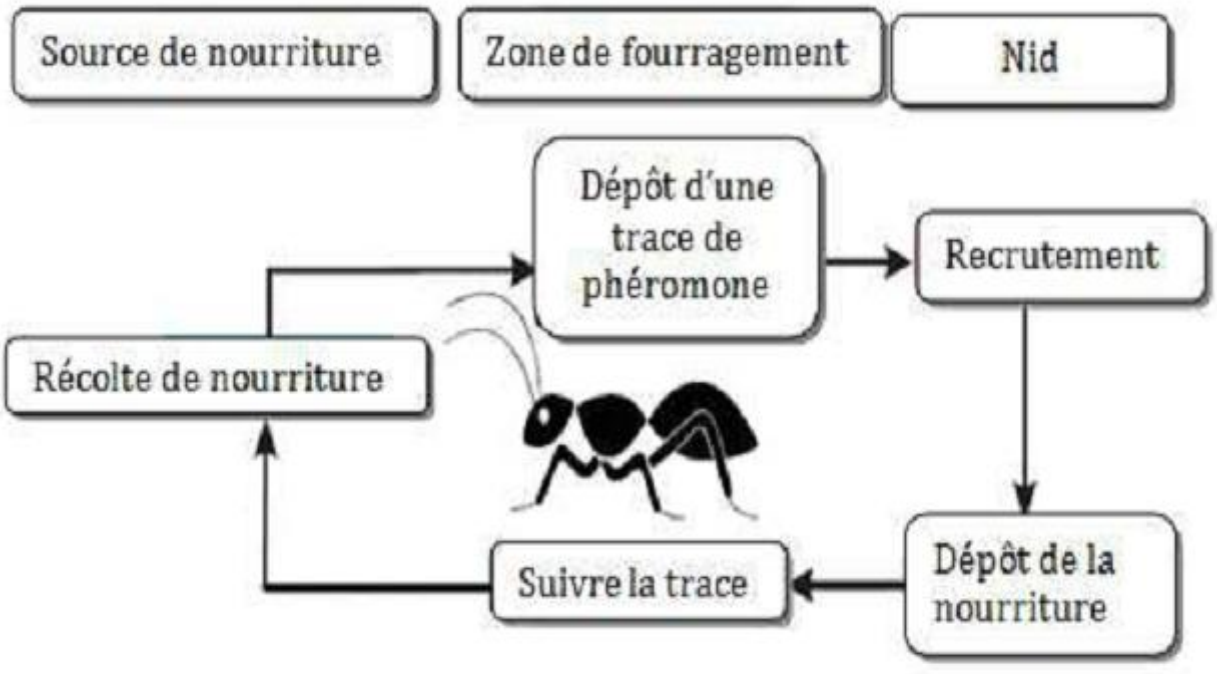
1. *INTRODUCTION:*

La méthode des colonies de fourmis est une méthode méta-heuristique, qui est relativement récente. Elle a été introduite en 1991 par Colorni, Dorigo et Maniezzo pour résoudre le problème du Voyageur de commerce. Elle s'est popularisée, puis a été l'objet d'améliorations dès 1995 et a été appliquée avec succès à d'autres problèmes d'optimisation combinatoire dès 1994.

2. *PRINCIPE:*

Les fourmis sont capables de résoudre collectivement des problèmes complexes, comme trouver le plus court chemin entre deux points dans un environnement accidenté. Pour cela, elles communiquent entre elles de façon locale et indirecte, grâce à une hormone volatile, appelée phéromone : au cours de sa progression, une fourmi laisse derrière elle une trace de phéromone qui augmente la probabilité que d'autres fourmis passant à proximité choisissent le même chemin.

L'optimisation d'un problème par colonies de fourmis (Ant Colony Optimization ACO) consiste à représenter le problème à résoudre sous la forme de la recherche d'un meilleur chemin dans un graphe, appelé graphe de construction, puis d'utiliser des fourmis artificielles pour rechercher de bons chemins dans ce graphe. Le comportement des fourmis artificielles est inspiré des fourmis réelles : elles déposent des traces de phéromone sur les composants du graphe de construction et elles choisissent leurs chemins relativement aux traces de phéromone précédemment déposées ; ces traces sont évaporées au cours du temps.



Comportement d'une fourmi naturelle lors de la recherche de nourriture

3. LES FOURMIS ARTIFICIELLES

Les fourmis artificielles sont des agents artificiels et par facilité son nom de fourmi. La programmation informatique moderne permet de faire fonctionner en parallèle divers « agents », par exemple des agents fourmis qui interagissent entre eux et avec leur environnement. Chaque agent répond en fonction d'une ou d'un petit nombre de règles simples, sans qu'il soit nécessaire de faire appel à des capacités cognitives élaborés. Les fourmis modélisées sont alors des agents réactifs et on parle alors d'intelligence en essaim.

4. ALGORITHME ACO:

Le premier algorithme s'inspire du comportement des fourmis recherchant un chemin entre leur colonie et une source de nourriture. La source d'inspiration des ACO est la trace des phéromones et le comportement des fourmis réelles qui utilisent les phéromones comme moyen de communication. Par analogie à la biologie, les ACO sont basés sur la communication indirecte d'une colonie de simples agents appelés fourmis artificielles en utilisant des traces de phéromones artificielles. Les fourmis artificielles dans un ACO mettent en œuvre une heuristique de construction aléatoire qui construit des décisions probabilistes en fonction des traces de phéromones artificielles et les données d'entrées du problème pour le résoudre.

La première étape de l'application d'un ACO est la définition d'un modèle adéquat: le modèle de phéromones. En générale, tout algorithme de colonies de fourmis doit contenir les éléments suivants:

- traces de phéromone,
- information heuristique (visibilité),
- règle de mise à jour des traces de phéromone,
- construction de solutions,
- probabilité de sélection
- condition d'arrêt.

Les traces de phéromones: les traces de phéromones sont représentées en fonction de la valeur de la fonction objective pour l'arc considéré. La définition de la quantité de phéromone est spécifique à chaque problème étudié et dépend de la variante d'ACO utilisée. Pour l'algorithme de base « ant system » utilisé pour le problème du voyageur de commerce.

La quantité de phéromones $\Delta\tau^{k}_{ji}$ déposée par une fourmi est donnée par :

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L^{k}} & \text{si la fourmi } k \text{ est passée par l'arc } (i,j) \\ 0 & \text{sinon} \end{cases}$$

Où Q est une constante et L^k la longueur du chemin parcouru par la fourmi k où F^k valeur de la solution obtenue par la fourmi k (la meilleure solution).

La construction des solutions: Chaque fourmi k est placée sur un nœud source et recherche un chemin en sélectionnant les nœuds i à ajouter à sa mémoire $tabou_k$. À chaque fois qu'une fourmi se trouve sur un nœud i, elle choisit le nœud j suivant selon une règle probabiliste donnée dans le cas du « **Ant System** » par:

$$p_{ij}^{k} = \begin{cases} \frac{{\tau_{ij}}^{\alpha}.{\eta_{ij}}^{\beta}}{\sum_{l \notin tabou_k} {\tau_{il}}^{\alpha}.{\eta_{il}}^{\beta}} & \text{si } j \in tabou_k \\ 0 & \text{sinon} \end{cases}$$

Où τ_{ij} représente les traces de phéromones et η_{ij} la visibilité. α et β sont respectivement l'importance relative des traces de phéromones et de la visibilité l'une par rapport à l'autre.

Mise à jour des traces: Une fois que les solutions ont été évaluées, elles peuvent influencer la matrice de phéromones à travers un processus de mise à jour des traces de phéromones. La mise à jour des traces de phéromones est effectuée à la fois localement et globalement. Le but de la mise à jour des phéromones est d'augmenter les valeurs de phéromones associées aux solutions « bonnes » ou prometteuses et de diminuer la valeur de phéromones associées aux mauvaises solutions.

Cette mise à jour est donc opérée par :

- La diminution des valeurs de toutes les phéromones à travers l'évaporation des phéromones
- L'augmentation du taux de phéromones associées à un ensemble de « bonnes » solutions:

$$\tau_{ij} \leftarrow \tau_{ij}\,(1-\rho) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Avec : ρ coefficient d'évaporation des traces de phéromones, $\Delta\tau^{k}_{ji}$ renforcement de l'arc (i, j) pour la fourmi k. L'évaporation des phéromones est nécessaire pour éviter une convergence trop rapide de l'algorithme.

Algorithme général de colonie de fourmis:

1. Initialisation des traces de phéromones
2. Tant que le critère d'arrêt n'est pas atteint, faire

   Pour toutes les fourmis, faire

   - Construire une nouvelle solution en utilisant les traces de phéromones et la visibilité

   - Évaluer les solutions construites

   Fin pour
3. Mise à jour des traces de phéromones
4. Fin tant que

Exemple: structure globale d'un ACO pour un problème de voyageurs de commerce.

Structure globale d'un ACO

Initialisation des pistes de phéromone ;

Boucler tant que critère d'arrêt non atteint :

Construire les solutions composant par composant, en traversant le graphe disjonctif.

Utilisation (facultative) d'une heuristique.

Mise à jour des pistes de phéromone, la mise à jour peut se faire localement ou globalement.

Fin de la boucle.

CHAPITRE VII: PROBABILITE ET RAISONNEMENT PROBABILISTE

I. RAISONNEMENT PROBABILISTE:

1. *RAPPELS DE PROBABILITES:*

Probabilité conditionnelle:

A et ***M*** deux événements, l'information a priori sur ***A*** est ***P(A).*** M s'est produit : ***P(M) ≠ 0***. S'il existe un lien entre ***A*** et ***M***, cet évènement va modifier notre connaissance sur ***A***.

L'information a posteriori:

$$\boldsymbol{P(A|M) = P(A,M).P(M)}$$

Indépendance:

A et ***B*** sont indépendants ssi :

P(A,B) = P(A) × P(B)

P(A|B) = P(A)

P(B|A) = P(B)

Indépendance conditionnelle:

A et B sont indépendants conditionnellement à **C** ssi :

$$\boldsymbol{P(A|B, C) = P(A|C)}$$

$\{M_i\}$ ensemble complet d'´ev´enements mutuellement exclusifs.

Marginalisation : $\boldsymbol{P(A) = \sum_i P(A, M_i)}$

Théorème des probabilités totales:

Un évènement A peut résulter de plusieurs causes M_i. Quelle est la probabilité de A connaissant :

- les probabilités élémentaires $P(M_i)$ (a priori)
- les probabilités conditionnelles de A pour chaque M_i

$$\boldsymbol{P(A) = \sum_i P(A|M_i)P(M_i)}$$

Mais comment répondre à la question inverse ?

Théorème de Bayes:

Un évènement A s'est produit. Quelle est la probabilité que ce soit la cause M_i qui l'ait produit ?

$$\boldsymbol{P(M_i|A) = \frac{P(A|M_i) \times P(M_i)}{P(A)}}$$

- $\boldsymbol{P(M_i|A)}$: probabilité a posteriori
- $\boldsymbol{P(A)}$: constante (pour chaque $\boldsymbol{M_i}$)

Théorème de Bayes généralisé (Chain rule):

$$\boldsymbol{P(A_1 \ldots A_n) = P(A_1)P(A_2|A_1)P(A_3|A_1, A_2) \ldots P(A_n|A_1 \ldots A_{n-1})}$$

II. RESEAUX BAYESIENS:

1. Définition:

Un réseau bayésien est un système représentant la connaissance et permettant de calculer des probabilités conditionnelles apportant des solutions à différentes sortes de problématiques. La structure de ce type de réseau est simple : un graphe dans lequel les nœuds représentent des variables aléatoires, et les arcs (le graphe est donc orienté) reliant ces dernières sont rattachées à des probabilités conditionnelles. Notons que le graphe est acyclique : il ne contient pas de boucle. Les arcs représentent des relations entre variables qui sont soit déterministes, soit probabilistes. Ainsi, l'observation d'une ou plusieurs causes n'entraîne pas systématiquement l'effet ou les effets qui en dépendent, mais modifie seulement la probabilité de les observer.

Le fait d'indiquer un arc entre deux variables implique une dépendance directe entre ces deux variables : l'une est le parent, et l'autre l'enfant. Il faut fournir le comportement de la variable enfant au vu du comportement de son ou ses (s'il y en a plusieurs) parents.

Pour cela, chaque nœud du réseau possède une table de probabilités conditionnelles. Une table de probabilités conditionnelles associée à un nœud permet de quantifier l'effet du ou des nœuds parents sur ce nœud : elle décrit les probabilités associées aux nœuds enfants suivant les différentes valeurs des nœuds parents. Pour les nœuds racines (sans parents), la table de probabilité n'est plus conditionnelle et fixe alors des probabilités a priori concernant les valeurs de la variable.

2. Construction d'un graphe:

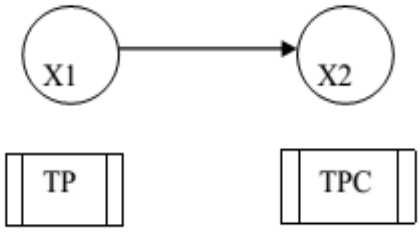
Construire un réseau bayésien c'est donc :

- Définir le graphe du modèle: c'est un graphe acyclique orienté $\boldsymbol{G}$, $\boldsymbol{G = (V, E)}$, ou $\boldsymbol{V}$ est l'ensemble des nœuds de $\boldsymbol{G}$, et $\boldsymbol{E}$ est l'ensemble des arcs de $\boldsymbol{G}$, ainsi $\boldsymbol{G = (V, E)}$. $\boldsymbol{V}$ symbolise un ensemble de variables aléatoires $\boldsymbol{X}$, $\boldsymbol{X = \{X1, X2\}}$.

- Définir les tables de probabilités (la composante quantitative ***μ***) de chaque variable, conditionnellement à ses causes. Des tables de probabilités (TP) pour les nœuds parents et des tables de probabilités conditionnelles (TPC) pour les nœuds descendants,

μ = {μi} = {P (Xi/Parents(Xi))}.

Un exemple ordinaire d'un réseau bayésien à deux variables est représenté par la figure ci-dessous:



3. <u>Domaines d'application:</u>

L'intérêt particulier des réseaux bayésiens est de tenir compte simultanément de connaissances a priori d'experts (dans le graphe) et de l'expérience contenue dans les données. Domaines d'utilisation principaux : diagnostic (médical et industriel), analyse de risques, détection de spams, datamining, détection de fraudes, exploitation du retour d'expérience, modélisation et simulation de systèmes complexes, détection d' intrusions, TextMining, analyse de BioPuces, analyse de trajectoires de santé. Pour résumer, un réseau bayésien est un modèle probabiliste graphique permettant d'acquérir, de capitaliser et d'exploiter des connaissances, né du besoin de créer des systèmes experts à base de probabilités.

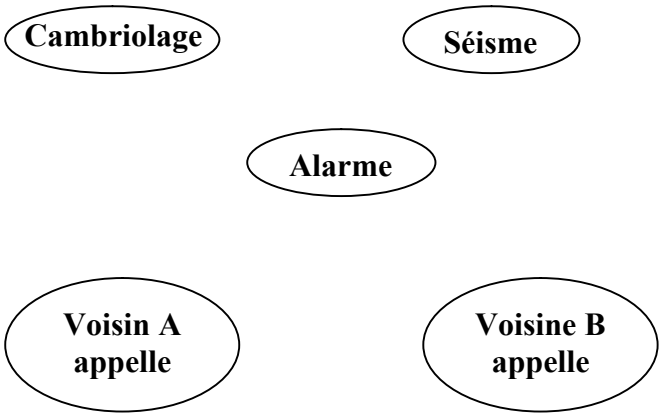
4. <u>Exemple:</u>

● Considérons la situation suivante :

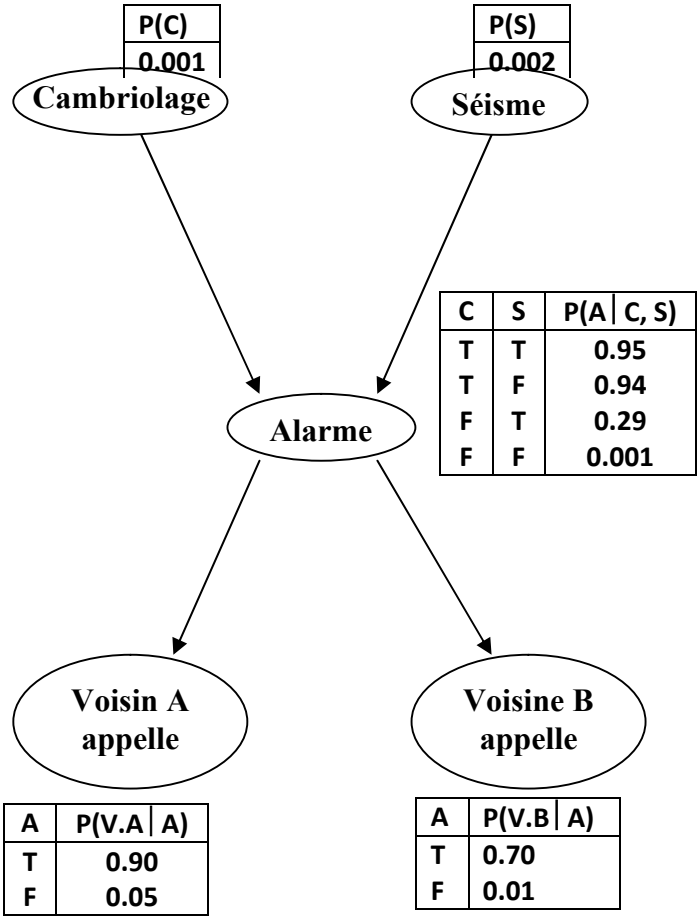
- je suis au travail, et mes voisins (A et B) m'ont promis de m'appeler chaque fois que mon alarme sonne
- mon voisin A m'appelle pour me dire que mon alarme sonne
  » Parfois il confond l'alarme avec la sonnerie du téléphone
- par contre ma voisine B ne m'appelle pas toujours
  » Parfois elle met la musique trop fort
- parfois mon alarme se met à sonner lorsqu'il y a de légers séismes
- comment conclure qu'il y a ou non un cambriolage chez moi?

● On peut représenter ce problème par un RB

- Variables aléatoires
  - ♦ Cambriolage
  - ♦ Séisme
  - ♦ Alarme
  - ♦ Voisin A appelle
  - ♦ Voisine B appelle



- La topologie RB modélise les relations de causalités
- Un arc d'un nœud X vers un nœud Y signifie que la variable X influence la variable Y.
  - ♦ Un cambriolage peut déclencher l'alarme
  - ♦ un séisme aussi
  - ♦ L'alarme peut inciter A à appeler
  - ♦ L'alarme peut inciter B à appeler
- Une table de probabilités conditionnelle (TPC) donne la probabilité pour chaque valeur du nœud étant donnée les combinaisons des valeurs des parents du nœud (c'est l'équivalent d'une distribution)



C	S	P(A | C, S)
T	T	0.95
T	F	0.94
F	T	0.29
F	F	0.001

A	P(V.A | A)
T	0.90
F	0.05

A	P(V.B | A)
T	0.70
F	0.01

Définitions:

- S'il y a un arc d'un nœud Y vers un nœud X, cela signifie que la variable Y influence la variable X
  - ♦ Y est appelé le **parent** de X
  - ♦ Parents (X) est l'ensemble des parents de X

- Si X n'a pas de parents, sa distribution de probabilités est dite **inconditionnelle** ou **a priori**
- Si X a des parents, sa distribution de probabilités est dite **conditionnelle**

Calcul de la probabilité conjointe:

$$\boldsymbol{P(X_1, \dots X_n) = \prod_{i=1}^{n} P(X_i | \, Parents(X_i))}$$

$$\boldsymbol{P(V.A, V.B, A, \neg c, \neg s) = P(V.A|A)P(V.B|A)P(A|\neg c, \neg s)P(\neg c)P(\neg s)}$$

$$\boldsymbol{= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.00062}$$

Avec: C = ¬c et S = ¬s.

Calcul de la probabilité marginale:

$$\boldsymbol{P(\neg c, A) = \sum_{V.A} \sum_{V.B} \sum_{s} P(V.A, V.B, A, \neg c, S = s)}$$

$$\boldsymbol{= \sum_{V.A} \sum_{V.B} \sum_{s} P(V.A|A)P(V.B|A)P(A|\, \neg c, s)P(\neg c)P(s)}$$

$$\boldsymbol{= \sum_{s} \sum_{V.A} \sum_{V.B} P(V.A|A)P(V.B|A)P(A|\, \neg c, s)P(\neg c)P(s)}$$

$$\boldsymbol{= \sum_{s} \sum_{V.A} P(V.A|A)P(A|\, \neg c, s)P(\neg c)P(s) \underbrace{\sum_{V.B} P(V.B|A)}_{= 1}}$$

$$\boldsymbol{\sum_{s} P(A|\, \neg c, s)P(\neg c)P(s) \underbrace{\sum_{V.A} P(V.A|A)}_{= 1}}$$

$$\boldsymbol{= \sum_{s} P(A|\, \neg c, s)P(\neg c)P(s) + P(A|\neg c, \neg s)P(\neg c)P(\neg s)}$$

$$\boldsymbol{= 0.29 \times 0.999 \times 0.002 + 0.001 \times 0.999 \times 0.998 = 0.0016}$$

Donc,

$$\boldsymbol{P(\neg c, A) = \sum_{V.A} \sum_{V.B} \sum_{s} P(V.A, V.B, A, \neg c, S = s) = \sum_{s} P(A|\neg c, S = s)P(\neg c)P(S = s)}$$

CHAPITRE VIII: SYSTEMES EXPERTS ET LEURS APPLICATIONS

I. DEFINITION:

D'une manière générale, un système expert est un outil capable de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier. Il s'agit de l'une des voies tentant d'aboutir à l'intelligence artificielle.

En outre, Un système expert est un système d'aide à la décision basé sur un moteur d'inférence et sur une base de connaissances. Il est la transcription logicielle de la réflexion d'un expert dans un domaine donnée.

Plus précisément, un système expert est un logiciel capable de répondre à des questions, en effectuant un raisonnement à partir de faits et de règles connus. Il peut servir notamment comme outil d'aide à la décision.

Il existe plusieurs systèmes experts crées et appliqués dans plusieurs domaines, médical, industriel, chimie, …. Parmi eux :

• SEDIAG, DIAGNEX, MAINTEX, SOLVEUR/AMIDEC : des systèmes experts pour le diagnostic.

• DENDRAL : 1969, chimie, recherche la formule développée d'un corps organique à partir de la formule brute et du spectrogramme de masse du corps considéré. Identifier les constituants chimiques d'un matériau.

• CRYSALIS : 1979, chimie, recherche la structure de protéines à partir de résultats d'analyse cristallographique.

• MOLGEN : 1977, biologie, engendre un plan de manipulations génétiques en vue de construire une entité biologique donnée.

• PROSPECTOR : 1978, géologie, aide le géologue à évaluer l'intérêt d'un site en vue d'une prospection minière. (1600 règles)

• AM : 1977, mathématiques, proposition de conjectures, de concepts intéressants. (500 règles)

• MUSCADET : 1984, mathématiques, démonstration de théorèmes.

• MYCIN : 1974, médecine, système d'aide au diagnostic et au traitement de maladies bactériennes du sang (Denis 2006).

II. STRUCTURE ET FONCTIONNEMENT D'UN SYSTEME EXPERT:

L'architecture d'un système expert typique est constituée de plusieurs modules qui s'interagissent (voir figure ci-dessous) :

- L'interface utilisateur sert à simplifier la communication, elle peut utiliser la forme question-réponse, le menu, le langage naturel….. etc.
- La base de connaissances contient les connaissances concernant la résolution du problème.
- Le moteur d'inférence est un mécanisme qui permet d'inférer des connaissances nouvelles à partir de la base de connaissances du système. Le moteur d'inférence applique une stratégie de résolution en utilisant les connaissances et ceci pour en dériver une nouvelle information.
- La base de faits contient les données spécifiques liées à l'application traitée. Elle peut contenir aussi les solutions intermédiaires ou les conclusions partielles trouvées lors de l'inférence. C'est pourquoi la base des faits constitue la mémoire de travail du système expert c'est - à - dire l'état du processus de résolution du problème.
- Le module d'explication permet au système expert d'expliquer son raisonnement.
- L'éditeur permet l'édition des connaissances dans la base.
-

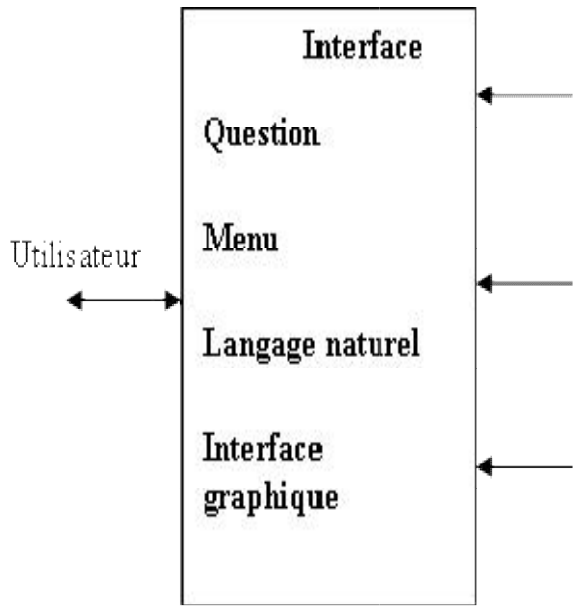


Figure (8.1): Ar

III. DOMAINES D'APPLICATION

Les systèmes experts ont été conçus pour résoudre certains types de problèmes comme en médecine, en droit, en chimie, en éducation,…. etc. Les catégories de problèmes abordés par les systèmes experts sont :

– L'interprétation ou la construction d'une description abstraite à partir de données.

– Le prédiction des conséquences à partir de situations données.

– Le diagnostic d'une défaillance à partir d'un ensemble d'observations.

– La conception d'une configuration de composants à partir d'un ensemble de contraintes.

– La planification d'une séquence d'actions pour l'accomplissement d'un ensemble de buts à partir de certaines conditions de départ et en présence de certaines contraintes.

– La réparation d'un dysfonctionnement.

– le contrôle du comportement d'un environnement complexe.

IV. SYSTEMES EXPERTS FLOUS:

Grâce à la logique floue, un système expert flou est capable d'évaluer les faits de manière plus nuancée. Lors de l'évaluation de la situation, les règles ne sont plus soit vraies, soit fausses. Elles sont plus ou moins vérifiées, et une action plus adaptée à la situation est suggérée.

La structure générale d'un système expert flou est le suivant:

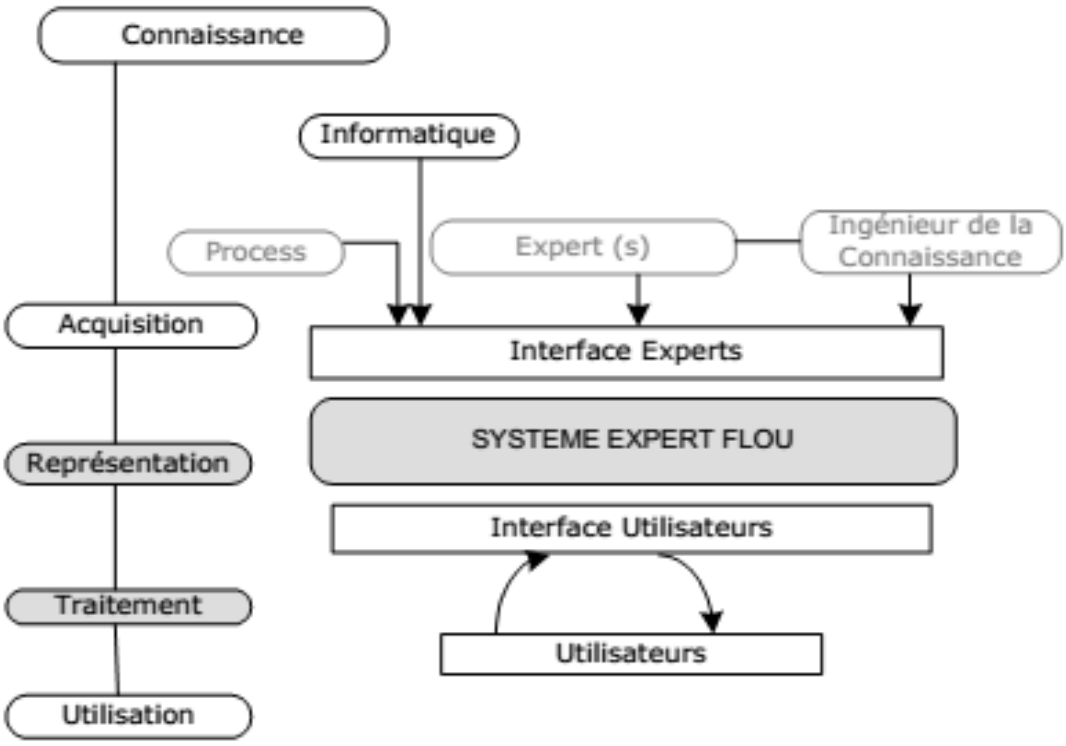


Figure (8.2): Structure d'un système expert flou

La structure interne du système expert flou dépend de son domaine d'application et de la nature des données à représenter.

Des systèmes experts flous ont été conçus dans le domaine de l'électronique de puissance, essentiellement pour faire le diagnostic des différentes pannes qui peuvent arriver à un circuit de transmission. La logique floue a été utilisée, également, dans le domaine de diagnostic pour l'agriculture pour la détection et l'identification du type de maladies d'insecte à partir des symptômes constatés sur les plantes . Il a été intéressant, certainement, d'utiliser un système expert flou dans la conception d'architecture d'un réseau informatique. Enfin, le domaine du diagnostic médical avait beaucoup utilisé les systèmes experts flous notamment pour l'analyse du sang de la corde embryonnaire et pour le diagnostic des symptômes du cancer de la peau.

V. SYSTÈMES EXPERTS EN INGÉNIERIE ÉLECTRIQUE:

Durant la dernière décennie, les systèmes experts ont trouvés un grand intérêt dans l'ingénierie électrique. Ils ont étés appliqués dans plusieurs domaines tels que: le diagnostic et la maintenance de problèmes temps réel.

Les diverses applications des systèmes experts aux domaines de l'ingénierie électrique sont très nombreuses, on peut citer quelques unes:

Système Expert	Rôle
ASDEP (Auxiliary System Design and Evaluation Program: Programme de conception et d'évaluation de systèmes auxiliaires)	Conception des réseaux électriques d'une centrale de production d'énergie électrique. Pour la conception de ce système, on doit prendre en considération plusieurs facteurs en même temps: - les données de la centrale et de la charge - le matériel nécessaire pour réaliser le réseau (transformateurs, câbles et disjoncteurs) - Fiabilité, maintenance, flexibilité, et prix.
Diagnostic des pannes dans les machines d'induction (en ligne)	La détection en ligne des différents types de défauts qui peut affecter une machine à induction par simple contrôle des courants d'alimentation. Les pannes peuvent être d'origine électrique, mécanique ou électromécanique. La détection de défaut est effectuée par visualisation et analyse en temps réel des spectres des courants en utilisant des capteurs. Ces spectres indiquent la présence des pics secondaires autour du fondamental en cas de présence de défaut. Le système expert visualise le défaut qui est le nœud terminal d'un arbre de défaillance résultant de l'ensemble de l'historique du moteur, qui à son tour est lié à un arbre d'observations fait à propos des états passés du moteur en question.
ESIMT (Expert System for Induction Motor Testing: Système expert pour les tests des moteurs à induction)	Tester en temps réel un moteur à induction en fonctionnement sur site. Il a été développé au Département de Génie Électrique de Naples (Italie) en utilisant la logique floue. Les paramètres du moteur à tester sont principalement: le courant, la tension, la puissance et la fréquence. Ces paramètres vont êtres comparés aux valeurs admissibles en utilisant une carte d'acquisition.

Décision pour le contrôle de la puissance réactive et de la tension	Détecter les problèmes de variations anormales de la tension d'un système de puissance en proposant les actions de contrôles appropriées (branchement de shunts de capacités, ajustement des rapports de transformations des transformateurs ou utilisation de compensateurs synchrones).
Estimation des sections défaillantes	Utilise une base de données constituée par des schémas de connections des relais et des disjoncteurs sous forme de règles de productions et par des faits concernant la configuration du système sous des nœuds et des branches
Diagnostic et la maintenance des grandes machines électriques à vitesse variable	Conception de convertisseurs pour les grandes machines synchrones: C'est pour permettre un prototypage rapide et une estimation de prix des convertisseurs conçus pour des grandes machines à vitesse variable (supérieur à lMW). Donner rapidement le convertisseur optimal, en se basant sur une base de données constituée d'informations sur les différents types de thyristors avec leurs caractéristiques et prix, les types de refroidissement de la machine (contraintes techniques de conception) et les formules de calcul. Ce système a été élaboré au Laboratoire d'Électrotechnique de GRENOBLE en collaboration avec JEUMONT SCHNEIDER INDUSTRIES
DAMOCLES (Design of Asynchronous Motor Controlled and Led by Expert System: Conception d'un moteur asynchrone contrôlé et piloté par un système expert)	Conception des moteurs asynchrones en prenant en charge tous les problèmes d'itération et de rebouclage fréquents.
Conception d'appareillage électrique (Système expert flou)	Conception des chambres de coupures des disjoncteurs à arcs tournants.

VI. APPLICATION AU DIAGNOSTIC:

Exemple d'un système de diagnostic industriel:

Le générateur de systèmes experts permet la création de systèmes experts d'ordre 0 et à base de logique floue dédiés au diagnostic. En fait, le diagnostic industriel repose sur le principe de cause et effet qui existe entre une panne et ses origines probables. Lorsqu'une panne ou un défaut survient, plusieurs scénarios sont envisageables. Afin que les systèmes experts puissent identifier le vrai scénario de panne, les relations causes et effets seront représentées par des relations floues et le calcul de la valeur d'appartenance permet d'identifier le poids de la possibilité. Un coefficient sera assigné à toutes les hypothèses de défauts et celle qui aura la pondération la plus importante sera la plus proche de la réalité.

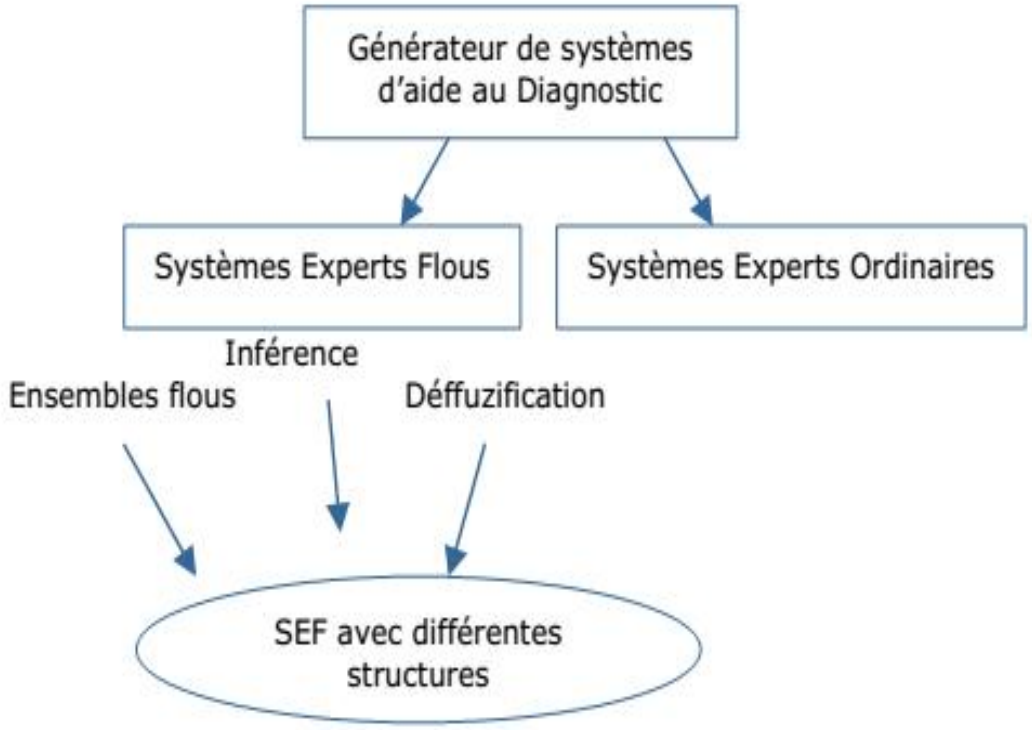


Figure (8.3): Architecture d'un générateur de système d'aide au diagnostic

CONCLUSION GENERALE

Ce polycopié de cours a permet de donner des généralités et les principes de bases des différents techniques de l'intelligence artificielle utilisés en Génie électrique

Nous avons commencé ce polycopié par une introduction sur l'intelligence artificielle qui explique la différence entre l'intelligence naturelle et artificielle, introduit ses domaines d'application et les techniques utilisées en Génie électrique. Puis, nous avons donné des généralités sur le "soft computing" et ses composantes essentielles qui sont: le système d'inférence flou, les réseaux de neurones et les algorithmes évolutionnaires. La théorie de la logique floue, ses applications et la constitution d'un système d'inférence floue avec un exemple d'application sont expliqués dans le troisième chapitre. Tandis que, le quatrième chapitre donne des généralités sur les réseaux de neurones (définition, structure d'un neurone formel, les différents types des réseaux de neurones), principe d'apprentissage supervisé et non supervisé et le principe de l'algorithme de Back-propagation. En utilisant les connaissances de bases obtenues par l'introduction des notions de logique floue et réseaux de neurones, nous avons expliqué le principe des réseaux de neurones adaptatifs et des réseaux neuro-flous. Le sixième chapitre est dédié aux algorithmes génétiques. Le septième chapitre donne des généralités sur les algorithmes évolutionnaires: recuit simulé, recherche tabou,… en détaillant la technique d'optimisation par essaims de particules. Les derniers chapitres donnent des généralités sur les probabilités, le raisonnement probabiliste et les systèmes experts avec ses applications (système expert floue et la prise de décision)

Ce polycopié de cours contient plusieurs exemples d'application avec leurs simulations sous Matlab/Simulink pour permettre aux étudiants de mieux comprendre les notions théoriques et savoir programmer les différentes techniques introduites.

BIBLIOGRAPHIE

[1] M. Djelti, B.Kouninef, ***"L'impact de l'intelligence artificielle sur le système éducatif***", Les ouvrages du CRASC, p. 187-21,. 2022

[2] Ph. Leray, ***"Réseaux bayésiens - introduction et apprentissage modélisation et découverte de connaissances"***, 8èmes Journées Francophones Extraction et Gestion des Connaissances Sophia Antipolis, 29 janvier 2008. http://www-sop.inria.fr/axis/egc08.

[3] O. Parent and J. Eustache, " ***Les Réseaux Bayésiens***", Master2 Recherche Connaissance et Raisonnement, 2006 – 2007 Cours Cognition et connaissance - Alain MILLE, Université Claude Bernard Lyon 1.

[4] S. Zeghdani, ***" Modélisation de l'état d'un système de production sur la base d'une approche Bayésienne",*** mémoire de Magister, Batna, 2015.

[5] G.Cormier, "***CHAPITRE 11. LOGIQUE FLOUE***".

[6] H. Mohamed Assaad, ***"Chapitre I: Introduction à la commande par logique floue".***

[7] B. Ould Bouamama, ***"Ingénierie des systèmes automatisés",*** Polytech.Lill.

[8] T. Laamayad, ***"Intelligence artificielle",*** Documents pédagogiques et didactiques, Université de Ouargla, 2015/2016.

[9] Cours: ***"les outils par l'intelligence artificielle",*** département d'informatique, université de Annaba

[10] COSTANZO Andrea, LUONG Thé and Van MARILL Guillaume, ***" Optimisation par colonies de fourmis"***, 19 mai 2006.

[10] A. Benyamina, ***"Application des algorithmes de colonies de fourmis pour l'optimisation et la classification des images***", Thèse de doctorat en informatique, Université des sciences et de la technologie d'Oran Mohamed Boudiaf, 2013.

[11] J.L Deneubourg, S Aron, S. Goss et J.M Pasteels, ***"The self-organizing exploratory pattern of the argentine ant"***. Dans Journalon insect Behavior, 3: 159-168, 1990.

[12] M .Dorigo, T. Stuzle, ***"Ant Colony Optimization",*** MIT Press, Cambridge, MA, 2004.

[**13] A.Taleb**, ***"Contribution des réseaux neuro-flous et algorithmes génétiques à la reconnaissance de la parole",*** Thèse de doctorat, université des sciences et de la technologie d'Oran Mohamed Boudiaf, 2010

[14] F.Hamidia, ***"Introduction à l'intelligence artificielle***", polycopié de cours, Université de Médéa, 2021.

[15] Ph. Leray, ***" Réseaux bayésiens - introduction et apprentissage modélisation et découverte de connaissances"***, 8èmes Journées Francophones Extraction et Gestion des Connaissances Sophia Antipolis 29 janvier 2008. http://www-sop.inria.fr/axis/egc08

[16] E. Bonabeau, M. Dorigo, G.Theraulaz, ***"Swarm Intelligence: From Natural to Artificiel Systems",*** NEW York, Oxford University Press, 1999.

[17] E. Bonabeau et G. Theraulaz, ***"L'intelligence en essaim, pour la science",*** 282 (3): pp. 66-73, N° 271 mai 2000.

[18] H. Zermane, ***" Cours et travaux dirigés Intelligence Artificielle pour l'Ingénierie",*** polycopié de cours pour Master en Génie industriel Management industriel, université Mostapha Ben-Boulaid Batna 2.

[19] A. Zouaidi, ***"Conception et réalisation d'un système expert pour l'évaluation des performances et l'analyse de stabilité des machines à réluctance variable",*** Thèse de doctorat, Université des sciences et technologies de Lille, 1994.