République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Frères Mentouri de Constantine
Faculté des Sciences de la Technologie
Département d'Electronique

N° d'ordre: 13/Ds/2018
N° serie:04/Ele/2018

THESE

Présentée Pour obtenir le diplôme de

DOCTORAT EN SCIENCES

Spécialité : Electronique

Option : Contrôle

Par

TITEL Faouzi

THEME

Commande et Identification Par Les Ondelettes Floues et Optimisation Par Les Algorithmes Evolutionnaires

Soutenue le : 07 / 02 / 2018

Devant le jury:

Président : Fella HACHOUF Professeur, Université des frères Mentouri Constantine

Rapporteur : Khaled BELARBI Professeur, ENPC de Constantine

Examinateurs : Zoheir HAMMOUDI Professeur, Université des frères Mentouri Constantine
Lamine MEHENNAOUI MCA, Université du 20 août 1955 Skikda
Fayçal ARBAOUI MCA, Université Badji Mokhtar Annaba

Année 2018

Dédicaces

Je dédie ce travail à mes très chers parents.

Je le dédie également à ma femme et à mes

Enfants Abdelmouiz et Abdelmoumene

Et à toute ma famille.

.

Remerciement

Il m'est particulièrement agréable de témoigner ma reconnaissance à Mr. K. Belarbi , Professeur à l'ENPC de Constantine pour le soutien et l'aide spontanée qu'il n'a jamais manqué de m'apporter .Sa collaboration et ses conseils m'ont énormément aidés .

Je remercie Mme. F. Hachouf , Professeur à l'université de Constantine de l'honneur qu'elle m'a fait en présidant le jury de cette thèse .

Je tiens à remercier Mrs: Z. Hammoudi, Professeur à l'université de Constantine, L. Mehennaoui, Maître de conférences (A) à l'université de Skikda et F. Arbaoui, Maître de conférences (A) à l'université de Annaba, pour l'intérêt qu'ils ont accordé à mon travail en acceptant d'examiner cette thèse .

En fin je tiens à remercier tous ceux qui m'ont aidé de prés ou de loin pour la réalisation de ce travail.

Résumé

Dans cette thèse, deux approches de conception des systèmes d'inférence floue à partir des données numériques ont été développées. Ces approches se caractérisent par leur capacité d'extraire et d'améliorer automatiquement la connaissance à partir des données numériques et de préserver l'interprétabilité des règles floues durant le processus d'optimisation.

Dans la première approche, on a proposé une nouvelle méthode de conception des contrôleurs Neuro-Flous qui implémentent des systèmes d'inférence floue du type Mamdani. Cette structure contient des neurones capables d'accomplir les opérations floues fondamentales. Les connexions entre les neurones du réseau sont pondérées par des poids réels et binaires. Une méthode évolutionnaire dite mixed Binary-Real Non dominated Sorting Genetic Algorithm II (MBR-NSGA II) a été développée pour assurer à la fois une bonne précision et une haute interprétabilité, ceci en minimisant simultanément deux critères de performance. Afin de préserver l'interprétabilité des règles floues tout le long du processus d'optimisation, on a imposé un jeu de contraintes. L'approche développée a été testée pour le contrôle de deux systèmes : le système du pendule inversé et le simulateur d'hélicoptère.

Dans la deuxième approche, une méthode à base d'algorithmes génétiques pour la conception des réseaux d'ondelettes floues a été proposée. Cette approche combine plusieurs techniques du soft computing tel que les systèmes flous Takagi-Sugeno, les réseaux d'ondelettes et les algorithmes génétiques. Ainsi, La structure du réseau d'ondelette flou adoptée consiste en une combinaison de deux parties ; une partie contenant le réseau d'ondelettes et une partie implémentant le mécanisme du raisonnement flou du type Takagi-Sugeno. Une méthode à base d'algorithmes génétiques est employée pour trouver les valeurs optimales des paramètres des deux structures. Cette approche a été testée en simulation, d'une part, à l'identification des systèmes dynamiques non linéaires, deux exemples issus de la littérature ont été étudiés D'autre part, à la commande de deux systèmes dynamiques non linéaires.

Mots clés:
Système d'inférence flou, Les algorithmes évolutionnaires, Réseaux de neurone flou, Optimisation multiobjectif, NSGAII, Contrôleur neuro-flou, Réseaux d'ondelettes, Réseaux d'ondelettes floues, Identification et Contrôle.

Abstract

In this work, two approaches for designing fuzzy inference systems from data are developed. These approaches are characterized by their ability to automatically extract and improve knowledge from numerical data and to preserve interpretability of fuzzy rules during the optimization process.

In the first approach, a Neuro-Fuzzy Controller network, called NFC that implements a Mamdani fuzzy inference system is proposed. This network includes neurons able to perform fundamental fuzzy operations. Connections between neurons are weighted through binary and real weights. Then a new algorithm called mixed Binary-Real Non dominated Sorting Genetic Algorithm II (MBR-NSGA II) is developed to perform both accuracy and interpretability of the NFC by minimizing two objective functions. In order to preserve interpretability of fuzzy rules during the optimization process, some constraints are imposed. The approach is tested on two control examples: the pole and cart system and a helicopter simulator model.

In the second approach, a genetic algorithm based method for designing fuzzy wavelet neural network (FWNN) is presented. The proposed framework combines several soft computing techniques such as fuzzy inference system, wavelet neural network and genetic algorithm. Thus, the structure of the proposed FWNN consists of combination of two network structures, one containing the fuzzy reasoning mechanism and the other containing Wavelet neural networks. Then a genetic algorithm based method is used to find optimal values of the parameters of the both network structures. The ability of the technique in identifying non linear dynamical systems is demonstrated on two examples. Also, this approach is tested for the control of two dynamic plants commonly used in the literature.

Keywords:
Fuzzy inference system, Evolutionary algorithms, Fuzzy neural network, Multiobjective optimization, NSGAII, Neuro-Fuzzy Controller, Wavelet neural network, Fuzzy wavelet network, Identification and Control

مـــلخـــــــص

في هذه الأطروحة ، طوّرت طريقتين جديدتين لتصميم الأنظمة التي تعتمد على المنطق الغامض انطلاقا من معطيات رقمية ، هذه الطرق تسمح في آن واحد باستخراج المعرفة بطريقة آلية و بتحسينها اعتمادا على المعطيات الرقمية مع الحفاظ على قراءة و تأويل القواعد الغامضة طوال مرحلة بحث القيم المثلى.

في الطريقة الأولى ، استعملنا طريقة جديدة من أجل تصميم بنية نظام عصبي غامض يؤدي عمل نظام غامض من نوع مامداني، هذه البنية تحتوي على عقد قادرة على أداء العمليات الغامضة الأساسية ، هذه العقد مرتبطة ببعضها البعض وفق وسائط حقيقية و ثنائية. استعملت طريقة متطورة متعددة الأهداف لأجل ضمان الدّقة الجيدة و التأويل العالي.و هذا بتقليص دالتي الكلفة في آن واحد، و من أجل الحفاظ على ضمان تأويل القواعد الغامضة طوال فترة بحث القيم المثلى ، فرضنا بعض المتطلبات. اختبرت هذه الطريقة في مراقبة نظامين : نظام النواس المعكوس و نظام محاكاة المروحية.

أما في الطريقة الثانية ، فقد استعملنا طريقة تعتمد على الخوارزمات الجينية من أجل تصميم شبكات المويجات الغامضة .هذه الطريقة عبارة عن مزيج من عدّة تقنيات كالأنظمة الغامضة ، شبكات المويجات و الخوارزمات الجينية ، و تتكون بنية الشبكة المويجية الغامضة من جزئين : جزء يحتوي على شبكة المويجات و جزء يحتوي على النظام الغامض من نوع طاكاجي - سوجينو و باستعمال منهجية تعتمد على الخوارزمات الجينية قمنا بإيجاد الوسائط المثلى للجزئين معا.

ومن جهة أخرى فقد أختبرت هذه الطريقة من أجل التعريف بالأنظمة الديناميكية الغير الخطية .و عليه فقد شملت الدّراسة نظامين مستنبطين من الأعمال العلمية ، وكذا من أجل التحكم في نظامين ديناميكين غير خطين.

الكلمات المفتاحية :

أنظمة المنطق الغامض ،الخوارزمات المتطورة ، الأنظمة العصبية الغامضة ، ميكانيزمات البحث المتعددة الأهداف ، المراقب العصبي الغامض ،الشبكة المويجية ، الشبكة المويجية الغامضة ، التعريف و المراقبة.

Sommaire

Chapitre III : Optimisation d'un SIF par l'algorithme MBR-NSGAII

Chapitre IV : Les réseaux d'ondelettes flous

Chapitre V : Optimisation d'un réseau FWNN par les AG

Liste des figures et tableaux

Figures

Tableaux

Liste des Acronymes

ACO : Ant Colony Optimisation.

ADEA: Adaptive Differential Evolution Algorithm.

AE : Algorithmes Evolutionnaires.

AFWNN: Adaptive Fuzzy Wavelet Neural Network.

AG : Algorithme Génétique.

ANFIS: Adaptive Network-based Fuzzy Inference System.

CWT: Continuous Wavelet Transform.

DE: Differential Evolution.

DEMO: Differential Evolution for Multiobjective Optimization.

DWT: Discrete Wavelet Transform.

EQM: Erreur Quadratique Moyenne.

FCM: Fuzzy C-Means.

FLC: Fuzzy Logic Controller.

FNS: Fuzzy Neural System (système neuronal flou).

FWNN: Fuzzy Wavelet Neural Network.

MBR-NSGA: Mixed Binary-Real Nondominated Sorting Genetic Algorithm.

MDEA: Multiobjective Differential Evolution Algorithm

MISO: Multi input Single output.

MOACO: Multiobjective Ant Colony Optimisation.

MODE: Multiobjective Differential Evolution.

MOEA: Multiobjective Evolutionary Algorithms.

MOGA: Multiobjective Genetic Algorithm.

MOMoDE: Multiobjective Modified Differential Evolution.

MOPSO: Multiobjective Particale Swarm Optimisation.

MSE: Mean Square Error.

NN : Neural Network (Réseau de neurones).

NPGA: Niched Pareto Genetic Algorithm.

NSGA: Nondominated Sorting Genetic Algorithm.

PDE: Pareto Differential Evolution.

PDEA: Pareto Differential Evolution Approach.

PSO : Particale Swarm Optimisation.

RBF: Radial Basis Function.

RFNN: Recurrent Fuzzy Neural Network.

RMSE: Root Mean Square Error.

SE : Stratégies d'Evolution.

SIF : Système d'inférence flou.

SPEA: Strength Pareto Evolutionary Algorithm.

TS : Takagi-Sugeno

WNN: Wavelet Neural network.

Introduction

Les systèmes d'inférence floue (SIF) représentent les applications les plus importantes de la logique floue et la théorie des sous-ensembles flous. Ils sont fortement capables de traiter l'imprécision et l'incertitude de l'information existante dans les problèmes complexes réels. Considérés comme des approximateurs universels et étant capables de représenter des concepts linguistiques (connaissance humaine), les SIF ont été largement utilisés avec succès pour la classification, le contrôle et la modélisation des systèmes non-linéaires.

Les systèmes d'inférences flous peuvent être construits soit à partir d'une expertise humaine soit à partir de données numériques [1]. Le premier cas est basé sur l'aptitude de la logique floue à modéliser le langage naturel. Ces SIF contiennent des règles floues obtenues à partir d'une expertise humaine, souvent qualitative, exprimé en langage naturel. Ce type de SIF offre un haut niveau sémantique mais malheureusement souffre de manque de précision dans le cas des systèmes complexes. Dans le deuxième cas, les SIF sont basés sur l'apprentissage automatique à partir des données numériques. La conception peut être décomposée en deux phases principales : la génération automatique des règles du SIF et l'optimisation du SIF. Cependant, lors du développement des SIF à partir des données numériques, deux aspects importants sont à considérer : l'interprétabilité et la précision.

D'une part, le terme interpretabilité consiste à décrire le comportement du système modélisé sous une forme compréhensible. D'autre part, la précision est une caractéristique du système flou qui montre son aptitude à représenter fidèlement le système réel, souvent exprimée sous forme d'un critère, le plus répandus est celui de l'erreur quadratique moyenne.

L'interpretabilté et la précision sont deux caractéristiques contradictoires ; en effet l'une peut être améliorée au profit de l'autre. Dans la littérature, cette situation est dite compromis Interprétabilité-précision (*the interpretability-accuracy trade-off*) [2,3].

De nombreuses approches ont été suggérées pour traiter ce problème afin d'augmenter l'interprétabilité des systèmes flous. Ces approches sont principalement basées sur une structure appropriée du système flou (par exemple : une structure hybride) [4-10] ou sur l'utilisation d'un algorithme d'optimisation spécifique (par exemple : méthodes dans le domaine de l'optimisation multi-objectif ou de l'optimisation évolutionnaire) [11-17].

Le travail effectué dans le cadre de cette thèse repose sur l'utilisation de différentes techniques du soft computing, à savoir, les réseaux de neurones, les réseaux d'ondelettes et les méthodes d'optimisation évolutionnaires pour la conception paramétrique et structurelle des systèmes d'inférence flous à partir des données numériques tout en assurant l'interprétabilité et la précision. L'étude en simulation considérée traite deux types de problèmes ; le problème de l'identification et le problème de commande tout en considérant une grande variété de systèmes.

Dans la première approche, on a proposé une nouvelle méthode de conception des contrôleurs Neuro-flous (CNF) qui implémentent des SIF du type Mamdani. Cette structure contient des neurones capables d'accomplir les opérations floues fondamentales. Les connexions entre les neurones du réseau sont pondérées par des poids réels et binaires. Une méthode évolutionnaire dite Mixed Binary-Real Non dominated Sorting Genetic Algorithm II (MBR-NSGA II) a été développée pour assurer à la fois une bonne précision et une haute interprétabilité, ceci en minimisant simultanément deux critères de performance. Afin de préserver l'interprétabilité des règles floues tout le long du processus d'optimisation, on a imposé un jeu de contraintes. L'approche développée a été testée pour le contrôle de deux systèmes : le système du pendule inversé et le simulateur d'hélicoptère.

Dans la deuxième approche, une méthode à base d'algorithmes génétiques (AG) pour la conception des réseaux d'ondelettes floues (GA-FWNN) a été proposée. Cette approche combine plusieurs techniques du soft computing tel que les systèmes flous Takagi-Sugeno, les réseaux d'ondelettes et les algorithmes génétiques. Cette combinaison a pour objectif de produire des systèmes qui ont une capacité d'apprentissage très rapide, peuvent décrire des non-linéarités, nécessitent un nombre réduit de neurones pour identifier des systèmes d'une grande complexité et offrent une convergence très précise.

La structure du réseau d'ondelette flou adoptée consiste en une combinaison de deux parties ; une partie contenant le réseau d'ondelettes et une partie implémentant le mécanisme du raisonnement flou du type Takagi-Sugeno.

La conception d'un tel réseau nécessite l'optimisation de certains paramètres caractérisant le réseau, à savoir, les paramètres des fonctions d'appartenance gaussiennes (les centres et les largeurs), les translations, les dilatations et les poids de pondération. Vue la complexité du problème d'optimisation, une méthode basée sur les algorithmes génétiques a été utilisée.

Cette approche a été testée en simulation, d'une part, à l'identification des systèmes dynamiques non linéaires, deux exemples issus de la littérature ont été étudiés. D'autre part, à la commande de deux systèmes dynamiques non linéaires.

Ce travail est organisé en cinq chapitres :

- Dans le premier chapitre, on présente et on classifie les différentes méthodes d'optimisation des SIF basées sur les données numériques.
- Dans le deuxième chapitre, on va présenter quelques approches évolutionnaires les plus populaires utilisées pour l'optimisation multi-objectif.
- Le chapitre trois comprend l'optimisation d'un contrôleur Neuro-flou réalisant un SIF du type Mamdani par un nouveau algorithme évolutionnaire dit MBR-NSGAII (Mixed Binary-Real Non dominated Sorting Genetic Algorithm II).
- Dans le chapitre quatre, il y a une description de quelques structures de réseaux d'ondelettes floues qui montrent la fusion des systèmes flous et les réseaux d'ondelettes.
- Le cinquième chapitre est consacré à l'utilisation des GA-FWNN pour résoudre les problèmes d'identification et de contrôle.
- Et enfin, on termine par une conclusion.

Chapitre I :

Conception Des Systèmes d'inférence floue

1.1 Introduction

Les systèmes d'inférence floue sont des approximateurs universels [18] et sont capables de représenter une connaissance humaine. Ces deux propriétés ont leurs limites inhérentes et peuvent devenir antinomiques dans certaines circonstances. En effet, d'une part, la précision dans l'approximation se paye soit par une perte de sémantique soit par l'augmentation considérable de la base de règles, d'autre part, l'expertise humaine ne peut pas être entièrement traduite.

De ce fait, plusieurs techniques basées sur l'apprentissage automatique à partir des données numériques ont été proposées dans la littérature permettant la conception des SIF. Cette conception peut être décomposée en deux phases principales : la génération automatique des règles du SIF et l'optimisation paramétrique et structurelle du SIF. Cependant, lors du développement des SIF à partir des données numériques, deux caractéristiques importantes sont à considérer : l'interprétabilité [1,19,20,21,22] et la précision [23].

L'interprétabilité décrit la capacité du système flou à offrir une bonne compréhension de son comportement en inspectant son fonctionnement ou sa base de règles. La précision est une caractéristique du système flou qui montre sa capacité à représenter fidèlement le système réel. Cependant, l'interprétabilité et la précision sont deux aspects contradictoires dans la conception des systèmes flous.

L'objectif de ce chapitre est de présenter et de classer les différentes méthodes de conception des SIF basées sur les données numériques.

1.2 Description d'un SIF

Un système d'inférence floue est décrit par un ensemble de règles représentées par un ensemble de relations linguistiques liant les variables d'entrée et les variables de sortie, ces règles ont la forme générale suivante :

R : SI {Conditions} ALORS {Actions}

La partie {conditions}, dite prémisses de la règle, est caractérisée par un certain nombre d'expressions du type $\{x_i\ est\ A^i, i = 1,2, \dots n\}$, où x_i sont les variables d'entrée et A^i leurs labels linguistiques, ces expressions sont généralement reliées entre elles par des opérateurs de conjonction ET. Dans le cas des systèmes flous MISO, la partie $\{Actions\}$, dite conséquence de la règle, est caractérisée par l'expression $\{y\ est\ B^i$ où y est la variable de sortie. Suivant la nature de B^i, on distingue deux types de SIF : Les SIF du type Mamdani [24] et ceux du type Takagi-Sugeno [25]. La principale différence entre ces deux types réside

dans le fait que les règles du type Mamdani sont à conclusion symbolique (B^i est une valeur linguistique) tandis que les règles du type Takagi-Sugeno sont à conclusion algébrique (B^i est une valeur numérique).

1.2.1 SIF du type Mamdani

Les règles floues du type Mamdani sont exprimées de la forme suivante :

$$R_i : Si\ x_1\ est\ A_1^i\ et \ldots x_n\ est\ A_n^i\ Alors\ \ y\ est\ B^i, \quad i = 1,2,\ldots,N \qquad (1.1)$$

Où : $x_i\ (i = 1,\ldots,n)$: Les variables d'entrée du SIF.

y : La sortie du SIF.

A_j^i, B^i: Termes linguistiques définis par les fonctions d'appartenance

Correspondantes $\mu A_j^i(x_j), \mu B^i(y)$.

La méthode de Mamdani repose sur l'utilisation de l'intersection floue (l'opérateur ET) pour réaliser l'implication floue, la fonction Min est souvent utilisée :

$$\mu R_i(x_1, x_2, \ldots, x_n, y) = Min\ (\mu A_1^i(x_1), \mu A_2^i(x_2), \ldots, \mu A_n^i(x_n), \mu B^i(y)) \qquad (1.2)$$

Chaque règle est activée séparément et les conclusions sont agrégées pour définir l'ensemble flou associé à la variable de sortie y. L'agrégation des règles est réalisée par l'union floue (l'opérateur OU), la fonction Max est souvent utilisée :

$$\mu R(x_1, x_2, \ldots, x_n, y) = Max(\mu R_i(x_1, x_2, \ldots, x_n, y), i = 1 \ldots N \qquad (1.3)$$

La sortie numérique y est la moyenne pondérée des sorties de toutes les règles, elle est souvent obtenue par la méthode de défuzzification du centre de gravité comme suit :

$$y = \frac{\sum_{i=1}^{N} c_i W_i}{\sum_{i=1}^{N} W_i} \qquad (1.4)$$

Avec : N représente le nombre de règles floues.

c_i est le centre de la fonction d'appartenance caractérisant la valeur linguistique B^i

W_i est la valeur de vérité de la règle R_i donnée par :

$$W_i = Min\ (\mu A_1^i(x_1), \mu A_2^i(x_2), \ldots, \mu A_n^i(x_n)) \qquad (1.5)$$

1.2.2 SIF du type Takagi-Sugeno

Dans ce cas, les règles ont la forme suivante :

$$R_i : Si\ x_1\ est\ A_1^i\ et \ldots x_n\ est\ A_n^i\ Alors\ \ y_i = f_i(x_1, x_2, .., x_n), i = 1,2,\ldots,N \qquad (1.6)$$

Généralement f_i est une fonction linéaire donnée par :

$$f_i(x_1, x_2, \ldots, x_n) = p_{i0} + p_{i1}x_1 + p_{i2}x_2 + \cdots + p_{in}x_n \qquad (1.7)$$

Avec p_{ij} (i=1,2,…,N ; j=0,1,…,n) : Coefficients réels constants.

La sortie finale du système flou est formulée par :

$$y = \frac{\sum_{i=1}^{N} y_i W_i}{\sum_{i=1}^{N} W_i} \quad (1.8)$$

Avec $W_i = ET\ (\mu A_1^i(x_1), \mu A_2^i(x_2), \dots, \mu A_n^i(x_n))$ est la valeur de vérité de la règle i.

1.3 Aspects Généraux De La Conception des SIF

1.3.1 Introduction

Les SIF de type Mamdani, comme ceux de Takagi-Sugeno, possèdent deux caractéristiques principales. D'une part, ils sont capables de représenter des connaissances humaines. D'autre part, ils sont des approximateurs universels. Ces deux propriétés ont été utilisées pour concevoir deux types de SIF : Les SIF basés sur les connaissances expertes (Expert Knowledge Based FIS) et les SIF basés sur les données numériques (Data Based FIS). L'approche de conception des SIF du premier type est basée sur les connaissances acquises par des opérateurs experts. Bien que cette approche offre des SIF à haut niveau sémantique, elle est cependant laborieuse, consomme beaucoup de temps, et dans la plus part des cas, spécifique pour chaque application, en plus cette approche présente d'autres difficultés tel que :

- Les opérateurs ne peuvent pas facilement transformer leurs connaissances et expériences en une forme algorithmique ou base de règles nécessaire pour la conversion en une stratégie de contrôle automatique.
- Toute la connaissance d'un expert ne peut pas être entièrement verbalisée : le savoir-faire et l'intuition ne sont pas transmissibles facilement.
- Le domaine d'expertise n'est pas toujours disponible.
- Dans le cas des systèmes complexes, ces SIF souffrent de manque de précision.

Dans ce qui suit, nous nous intéresserons à la conception des SIF à partir des données numériques. Il s'agit d'extraire des connaissances à partir des données numériques et de les exprimer dans un langage proche du langage naturel.

La méthodologie peut être décomposée en deux phases principales [1] :

- Génération automatique des règles du SIF.
- L'optimisation du SIF.

La base de règles doit satisfaire des conditions d'interprétabilité qui sont : la complétude, la distinction, la consistance et la continuité [26].

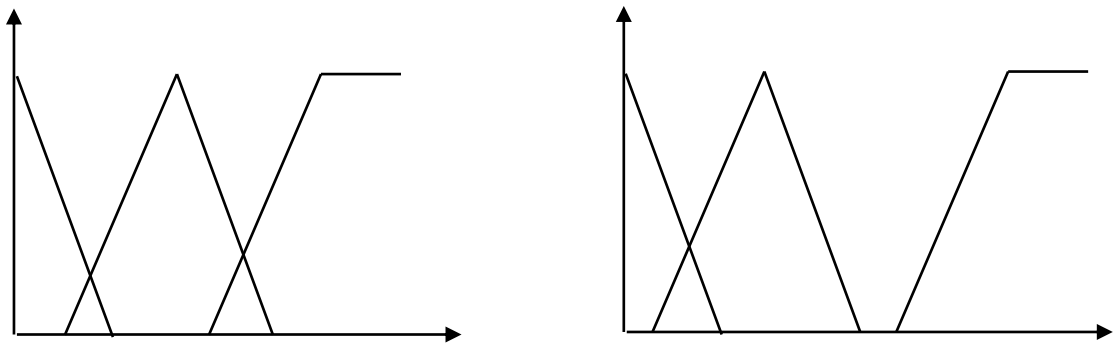
1.3.2 Propriétés De La Base De Règles

a) Complétude (Completeness)

Une base de règles floues est complète si, étant donné un vecteur d'entrée quelconque, il y a au moins une règle qui est active. Une base de règles est incomplète s'il existe une situation de l'espace d'entrée pour laquelle aucune règle n'est activable. La complétude des systèmes flous consiste-en :

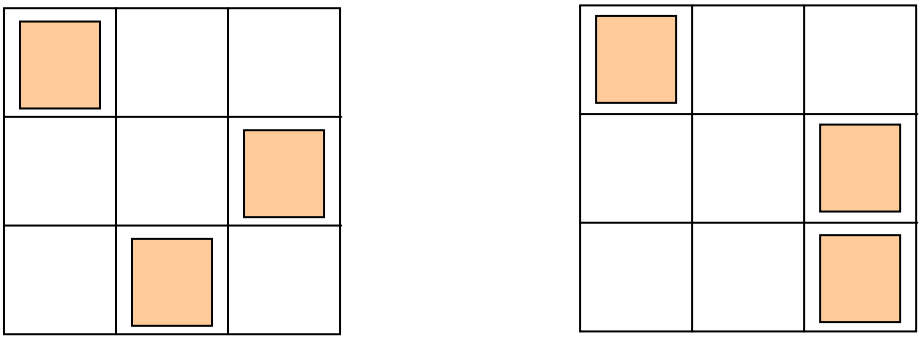
- La complétude des partitions floues
- La complétude de la structure de règle floue.

Un exemple d'une partition floue complète/incomplète, ainsi qu'une structure de règle floue complète/incomplète sont indiquées dans les figures (1.1) et (1.2).



(a) **(b)**

Fig.1.1 : Partition floue (a) Complète
(b) Incomplète



(a) **(b)**

Fig.1.2 : Structure de règle floue (a) Complète / (b) Incomplète
(La case pleine représente une règle floue)

b) Distinguabilité (Distinguishability)

La ditinguabilité est l'une des plus importants aspects d'interprétabilité des systèmes flous. Dans la figure (1.3), des exemples de partitions floues distinguables/indistinguables sont illustrés.

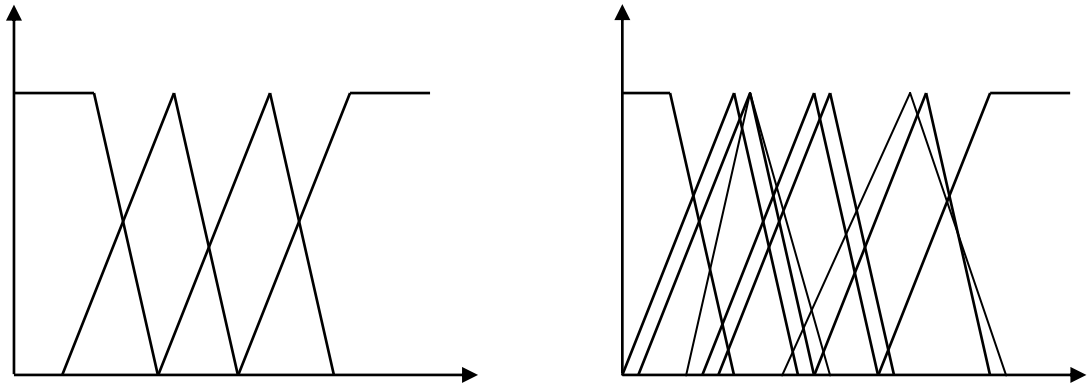


Fig.1.3 : Des partitions floues (a) Distinguables (b)indistinguables

Généralement, une partition floue distinguable dépend de la similarité des sous-ensembles flous et le nombre de sous-ensembles flous dans la partition. En effet, Une partition floue distinguable ne doit pas avoir plusieurs sous-ensembles flous similaires et le nombre de sous-ensembles flous ne doit pas dépasser neuf [27].

La complétude et la distinguabilité d'une partition floue peuvent être garanties en imposant une contrainte sur la similarité entre deux sous ensembles flous adjacents [26] :

$$S1 < S(A,B) < s2 \qquad (1.9)$$

Avec: $0 < s1 < s2 < 1$

$S(A,B)$ mesure de la similarité floue.

$S(A,B) = 0$, signifie que les deux sous-ensembles flous ne se recouvrent pas.

$S(A,B) = 1$, signifie que les deux sous-ensembles flous sont les mêmes (A=B).

c) Consistance (Consistency)

Une base de règles floues est consistante si elle ne contient pas de contradictions. Une base de règles floues est inconsistante s'il existe au moins deux règles qui ont des parties prémisses très similaires (ou identiques), mais possèdent différentes conséquences.

Un exemple d'une telle base de règles est :

$$R1 : Si\ obstacle\ en\ face\ Alors\ aller \text{ à } gauche$$

$$R2 : Si\ obstacle\ en\ face\ Alors\ aller \text{ à } droite$$

d) Continuité (Continuity)

La continuité de la base de règles floues garanti que lorsqu'il y'a de petites variations à l'entrée, ceci ne provoque pas de grandes variations en sortie. En d'autre terme, une base de règles floues est continue si toutes les règles de prémisses adjacentes ont des conséquences adjacentes. De plus il est préférable qu'un système d'inférences floues soit compact, c'est à dire qu'il ait un nombre réduit de conditions dans les prémisses des règles floues et un nombre réduit de règles floues dans la base de règles.

1.4 Génération Automatique des Règles du SIF

Il existe trois types de méthodes de génération de règles. Le premier type consiste à partager l'espace d'entrée en un nombre défini de sous-ensembles flous, ainsi les règles floues sont générées en se basant sur ces sous-ensembles flous. Le deuxième type est le groupement flou, dont lequel les données d'entraînement sont organisées dans des groupes homogènes, ensuite une règle est associée à chaque groupe. Dans le dernier type, on trouve les méthodes basées sur les techniques du Soft Computing telles que les réseaux de neurones, les réseaux d'ondelettes et les méthodes évolutionnaires.

1.4.1 Méthodes Basées Sur Le Partitionnement Flou

Dans cette méthode, chaque variable d'entrée est partitionnée en sous-ensembles flous. Ensuite, la génération de la base de règles est faite selon trois approches :

a) Première approche : Implémentation de toutes les règles

La génération des règles dans cette approche consiste à implémenter toutes les règles qui correspondent à toutes les combinaisons possibles des sous-ensembles flous des variables d'entrée.

Soit x_i la $i^{ème}$ variable d'entrée, on considère que son domaine est découpé en $m_i$sous ensembles flous ayant chacun leur label : $A_i^1, A_i^2, \dots, A_i^{m_i}$.

En prenant toutes les combinaisons possibles, le nombre total des règles pour un système à n entrées est : $\prod_{i=1}^{n} m_i$, ce qui peut entraîner une explosion combinatoire.

b) Deuxième approche : Choix dynamique du nombre des sous ensembles flous

Dans la première approche, le nombre des sous-ensembles flous m_i utilisés pour partitionner l'espace d'entrée est fixé. Ceci pose un problème dans le choix de la valeur de m_i car un m_i petit rend le système incapable de modéliser ou d'approximer un comportement

non linéaire, d'autre part m_i ne peut pas prendre des valeurs assez grandes parce que cela va résulter en un très grand nombre de règles.

Pour remédier à ces problèmes et d'éviter de fixer les valeurs des m_i, des auteurs ont proposé des méthodes qui permettent de dériver les valeurs de ces sous-ensembles flous à partir des données. Parmi ces méthodes, on peut citer la méthode dite "*Partition Refinement"* qui consiste à ajouter à chaque étape de l'algorithme un ensemble flou à l'entrée qui vérifie un certain critère [28]. Une autre méthode consiste à utiliser les algorithmes génétiques pour sélectionner les meilleures partitions souhaitées pour chaque région de l'espace d'entrée [29].

c) Troisième approche : Seulement une règle pour une paire de données

Dans cette méthode, introduite par Wang et Mendel, le nombre des règles ne dépend pas du nombre d'ensembles flous considérés pour chaque variable d'entrée, il dépend du nombre de paires d'entraînement. Dans [30], les auteurs ont proposé une méthode qui consiste en cinq étapes. Cette méthode permet d'avoir une base de règles adaptative : de nouvelles règles sont en compétition avec des règles existantes.

1.4.2 Groupement Flou : (Fuzzy clustering)

Les Algorithmes du groupement flou consistent à organiser les données dans des groupes homogènes et d'associer une règle à chaque groupe. L'algorithme de groupement des C-moyennes floues CMF (ou Fuzzy C-means clustering; FCM Clustering) est probablement le plus utilisé.

Les C-Moyennes Floues (FCM)

On se donne n vecteurs d'entrée $(x_j)_{j=1}^{n}$ que l'on veut associer à C groupes (clusters) de centres $(v_i)_{i=1}^{c}$. Chaque point (x_j) appartient au groupe de centre (v_i) avec un certain degré d'appartenance noté $\mu_{ij} \in [0\ 1]$. La méthode des C-moyennes floues consiste à minimiser le critère :

$$J_{FCM} = \sum_{j=1}^{n}\sum_{i=1}^{c} \mu_{ij}^{m}\, D_{ij} \tag{1.10}$$

$$\text{Ou : } D_{ij} = \left\|x_j - v_i\right\|_A^2 = (x_j - v_i)A(x_j - v_i)^T \tag{1.11}$$

Avec A est une matrice symétrique définie positive

m est l'exponentiel flou

Sous les contraintes :

$$\sum_{i=1}^{c} \mu_{ij} = 1, \forall\, j = 1, \dots, n \tag{1.12}$$

$$\text{Et} \quad m \geq 1 \tag{1.13}$$

La procédure d'optimisation commence par une initialisation aléatoire des degrés d'appartenance et par donner une valeur fixe au nombre de clusters C, ensuite les centres v_i et les degrés d'appartenance μ_{ij} sont itérativement modifiés par les équations suivantes :

$$v_i = \frac{\sum_{j=1}^{n} \mu_{ij}^{m}\, x_j}{\sum_{j=1}^{n} \mu_{ij}^{m}} \tag{1.14}$$

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{D_{ij}}{D_{kj}}\right)^{2/m-1}} \tag{1.15}$$

Ces opérations sont répétées jusqu'à convergence.

Plusieurs variantes de l'algorithme des FCM ont été proposées dans la littérature, tel que l'algorithme de PCM (Possibilistic C-means) [31] et l'algorithme de FCRM (Fuzzy C-regression model) [32].

L'algorithme de groupement des c-moyennes floues ainsi que ses variantes présentent deux grands problèmes :

a. Le choix du nombre de groupes (c), ce problème est connu sous le nom du problème de validité du groupe.
b. Le choix de la valeur de m.

Deux techniques ont été proposées pour résoudre le premier problème, la première consiste à exécuter l'algorithme des FCM avec un nombre croissant de clusters ($c = 2, \dots, n-1$) et de caractériser chaque partition en utilisant des indexes [33]. Dans la deuxième technique, il s'agit d'exécuter une seule fois un algorithme qui peut déterminer le nombre de groupes souhaité, parmi ces algorithmes, on trouve l'algorithme proposé par Chiu [34] dit groupement soustractif. Concernant le choix de m, plusieurs auteurs recommandent une valeur fixe égale à 1,5 ou 2.

1.4.3 Les Méthodes Du Soft Computing

A cette classe, appartiennent les méthodes basées sur les techniques du *soft computing*, les plus utilisées sont les réseaux de neurones, les réseaux d'ondelettes et les algorithmes évolutionnaires.

1.4.3.1 Les systèmes Neuro-flous

Les systèmes Neuro-flous (FNS) sont des systèmes d'inférence floue implémentés dans un réseau de neurones (NN). Ce NN peut être un réseau hybride [35, 36] ; un réseau récurrent lorsqu'il s'agit de modéliser ou de contrôler des systèmes dynamiques [37, 38] ; un réseau adaptatif (ANFIS) [39] ; ou un réseau à fonctions de base radiale (RBF)[40].

a) Les FNS hybrides

Les réseaux hybrides contiennent dans leurs structures des neurones-flous capables d'accomplir les opérations floues fondamentales tel que ET et OU. Ces opérateurs Sont réalisés utilisant la norme-T, la conorme-T ou la norme-S, ceci a permis de développer différentes structures neuronales-floues hybrides [35, 36,41, 42].

Un exemple simple d'architecture d'un tel réseau est indiqué dans la figure (1.4)[36]. Ce réseau consiste en cinq couches : une couche d'entrée, une couche de fuzzification qui utilise des fonctions d'appartenance de forme triangulaire, une couche de neurones ET, une couche de neurones OU et une couche de défuzzificatiuon qui consiste en la méthode du centre de gravité.

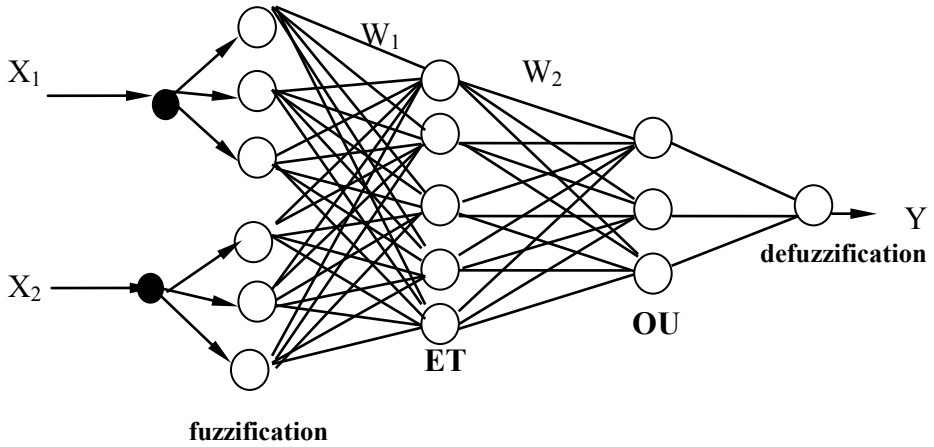


Fig. 1.4 : Architecture d'un FNS hybride
- 2 entrées/1 sortie
- 3 ensembles flous pour les E/S
- 5 neurones ET et 3 neurones OU

Les opérateurs ET et OU dans cette structure sont réalisés par les normes T-S comme suit :

La fonction ET : $y_{ET} = ET(X, W) = T_{i=1}^{I}[x_i sw_i]$ (1.16)

La fonction OU : $y_{ou} = OU(X, W) = S_{i=1}^{I}[x_i tw_i]$ (1.17)

Les normes T et S étant le produit et la somme algébrique respectivement, les fonctions ET et OU s'écrivent :

$$y_{ET} = \prod_{i=1}^{I}(x_i + w_i - x_i w_i) \tag{1.18}$$

$$y_{OU} = 1 - \prod_{i=1}^{I}(1 - x_i w_i) \tag{1.19}$$

Ces sorties sont suivies par un élément non linéaire qui peut être une fonction sigmoïdale f modifiée par les paramètres m et b :

$$f(y) = \frac{1}{1 + e^{-(y-m)b}} \tag{1.20}$$

Ce réseau est entraîné en utilisant le signal d'erreur qui est égal à la différence entre la sortie désirée et la sortie du réseau. L'algorithme de la rétro-propagation est souvent utilisé pour ajuster les poids des neurones ET et OU, une fois le réseau converge, on peut extraire les règles d'inférence.

b) Les FNS récurrents

Les réseaux de neurones flous récurrents (RFNN) s'adaptent mieux pour la description des systèmes dynamiques. Dans [37], les auteurs ont proposé une structure d'un RFNN qui consiste en un réseau connexionniste comprenant quatre couches (Fig.1.5) ; une couche d'entrée ; la deuxième couche contient des neurones qui réalisent des fonctions d'appartenance de type gaussien et agissent comme des unités de mémoire à cause des connexions de retour (feedback connections) ; les neurones de la troisième couche dits neurones de règles, réalisent l'opération ET en calculant le produit algébrique de leurs entrées. Enfin la couche de sortie qui réalise l'opération de défuzzification en considérant une combinaison linéaire des conséquences obtenues à partir de chaque règle.

L'algorithme utilisé pour l'apprentissage est celui de la rétro-propagation qui permet l'ajustement des paramètres du réseau à savoir : les paramètres des fonctions d'appartenance, les poids des connexions feedback et les poids des connexions entre la troisième et la quatrième couche.

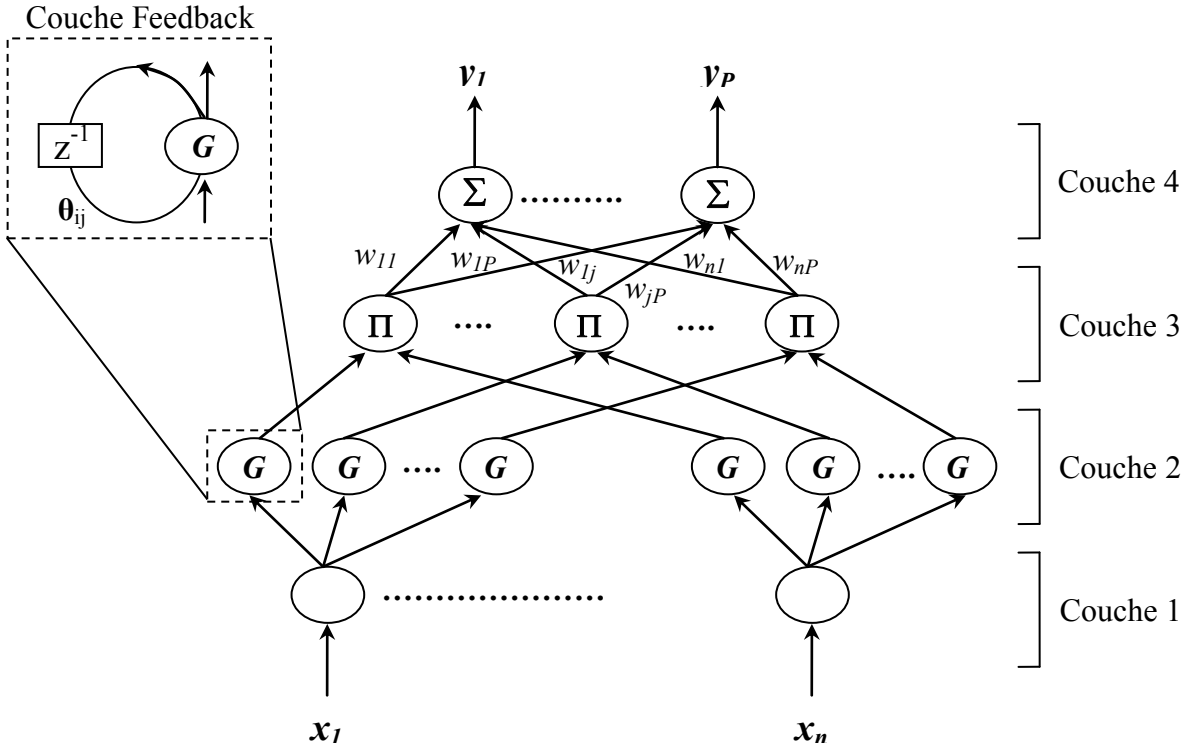


Fig. 1.5 : Configuration du RFNN

c) Architecture ANFIS

L'architecture ANFIS a été proposée par Jang [43] ; c'est un réseau connexionniste à cinq couches réalisant un système d'inférence flou du type Takagi et Sugeno.

Ce réseau contient des nœuds adaptatifs représentés par des carrés et des nœuds fixes représentés par des cercles. La structure d'un tel réseau est illustrée dans la figure (1.6); pour la simplicité, on a considéré un réseau ayant deux entrées (x_1 et x_2), une sortie (y) et deux règles floues.

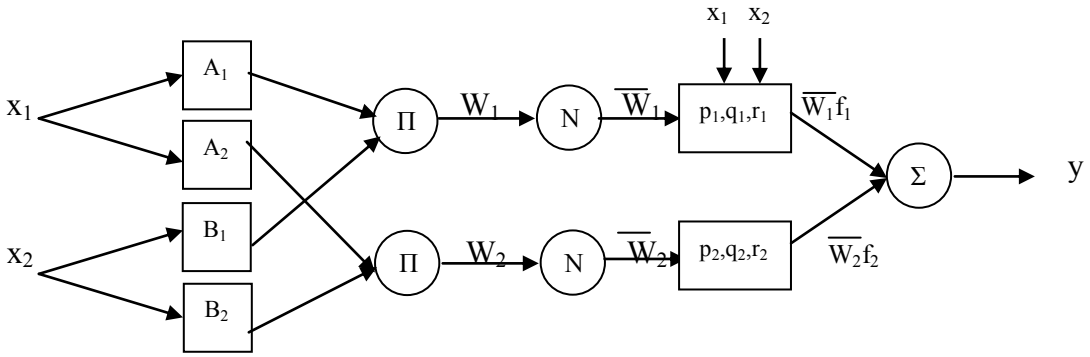


Fig. 1.6 : Architecture ANFIS

La première couche contient des nœuds qui représentent les fonctions d'appartenance associées aux variables linguistiques A_i et B_i. Les nœuds de la deuxième couche calculent les valeurs de vérité W_i des deux règles. Chaque nœud de la troisième couche calcule la valeur de vérité normalisée $\overline{W_i}$. Et enfin la sortie du réseau est donnée par :

$$y = \sum_{i=1}^{2} \overline{W_i} f_i = \sum_{i=1}^{2} \overline{W_i} (p_i x_1 + q_i x_2 + r_i) \qquad (1.21)$$

L'apprentissage du réseau permet d'ajuster les paramètres des fonctions d'appartenance caractérisant les labels A_i et B_i ainsi que l'ensemble des paramètres $\{p_i, q_i, r_i\}$ caractérisant la partie conséquence des règles floues. L'algorithme d'apprentissage utilisé est celui de la rétro-propagation.

d) Les RBF

Les SIF sont fonctionnellement équivalents à des réseaux de neurones à base radiale (RBF) [40]. Un réseau RBF comprend trois couches; une couche d'entrée, une couche cachée comprenant un nombre de nœuds égal au nombre de règles et une couche de sortie (Fig.1.7).

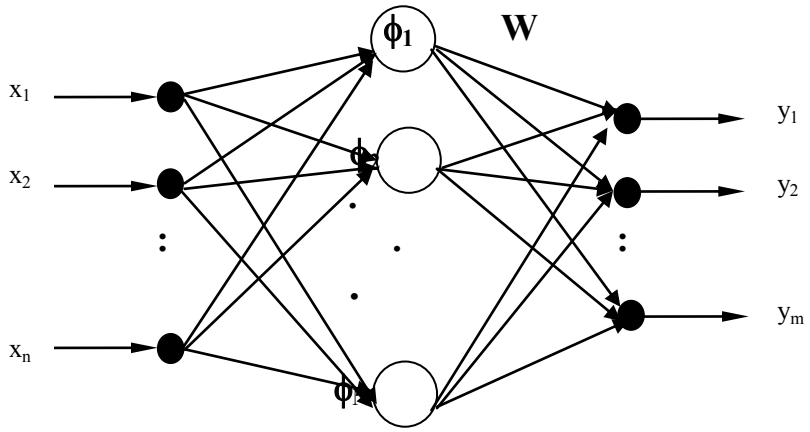


Fig.1.7 : Structure du réseau RBF

Avec une telle structure, le réseau RBF permet une représentation des règles floues : chaque nœud de la couche cachée représente une règle et les fonctions de base sont équivalentes aux fonctions d'appartenance du système flou. Jang [40] a montré que sous certaines conditions, les RBF et les SIF sont équivalents, la principale condition est que les sorties des règles floues doivent être des scalaires.

Un réseau RBF peut approximer n'importe quelle fonction continue $f : \Re^n \longrightarrow \Re^m$ en effectuant une transformation non linéaire au niveau de la couche cachée suivie d'une transformation linéaire au niveau de la couche de sortie. Les sorties y_k du réseau sont données par :

$$y_{k=1,\dots,m} = \sum_{i=1}^{N} \Phi_i \left(\|x - c_i\|_2\right) w_{ik} \qquad (1.22)$$

Ou : $x \in \Re^n$ est le vecteur d'entrée.

c_i est un vecteur de $\Re^n$ caractérisant le nœud i de la couche cachée.

N est le nombre de nœuds de la couche cachée.

Φ_i les fonctions de base.

w_{ik} sont les poids des connexions entre la couche cachée et la couche de sortie.

$\| \quad \|_2$ est la norme euclidienne

Il existe plusieurs formes de la fonction de base Φ, la plus utilisée est la fonction gaussienne exprimée par :

$$\Phi_i(\|x - c_i\|_2) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - c_{ij})^2}{2\sigma_i^2}\right) = \prod_{j=1}^{n} \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_i^2}\right) \qquad (1.23)$$

L'apprentissage consiste à déterminer un nombre minimum de nœuds dans la couche cachée, c.à.d. le nombre de règles, les vecteurs correspondants c_i et σ_i ainsi que les poids w_{ik}.

1.4.3.2 Les Algorithmes Génétiques

Les concepts de base des algorithmes génétiques ont été développés par J. Holland [44]. Ce sont des techniques d'optimisation stochastiques qui tentent d'imiter les processus d'évolution naturelle des espèces et de la génétique, pour cela, ils utilisent un codage des paramètres d'origine du problème d'optimisation et effectuent des opérations génétiques spéciales. Vu leur application avec succès sur une variété de problèmes d'apprentissage et d'optimisation, les AG ont été proposé comme une méthode de conception des contrôleurs flous. Il existe trois approches d'application des AG pour la conception des contrôleurs flous :

- Les AG sont utilisés pour synthétiser la structure du contrôleur flou qui consiste à extraire un ensemble optimal de règles linguistiques et leurs fonctions d'appartenance [45, 46, 47,48, 142].
- Les AG sont utilisés pour trouver des fonctions d'appartenance optimales pour un ensemble de règles floues spécifiées [49, 50].
- Les fonctions d'appartenance des valeurs linguistiques spécifiées sont fixées et l'AG est utilisé pour déterminer un ensemble optimal de règles [51].

1.4.3.3 Les réseaux d'ondelettes-flous

Les réseaux d'ondelettes flous (FWNN) sont l'une des méthodologies du Soft Computing qui ont pris considération dans la dernière décennie. Ces réseaux combinent la logique floue,

les réseaux de neurones et la théorie des ondelettes pour construire un modèle très puissant en termes d'apprentissage. Cependant, La conception d'un tel réseau nécessite l'optimisation de certains paramètres caractérisant le réseau, à savoir, les paramètres des fonctions d'appartenance, les paramètres du réseau d'ondelettes et les poids des règles floues. Plusieurs méthodes ont été proposées dans la littérature, qu'on peut classer en deux grandes catégories : les méthodes basées sur le calcul des dérivées [52-55] et les méthodes employant les algorithmes évolutionnaires [56-59].

En général, la structure d'un FWNN implémente des règles floues du type T-S dans lesquelles des fonctions ondelettes sont intégrées, ainsi la forme de la $i\grave{e}me$ règle floue est donnée par :

$$R^i: Si\ x_1 est\ A_1^i\ ET\ x_2\ est\ A_2^i, \dots, ET\ x_n est\ A_n^i\ \ Alors\ \ y_i = w_i \sum_{j=1}^{n} \psi_{ij}\left(x_j\right) \quad (1.24)$$

Avec : $x_j\ (j = 1, \dots, n)$ représente le signal d'entrée.

$y_i\ (j = 1, \dots, N)$ représente le signal de sortie de la règle R^i

A_j^i est la fonction d'appartenance de la $j\grave{e}me$ entrée de la $i\grave{e}me$ règle

w_i est le poids entre les signaux d'entrée et la $i\grave{e}me$ sortie

ψ_{ij} représente une famille d'ondelettes obtenue par dilatation et translation d'une ondelette mère $\psi(x)$

$$\psi_{ij}\left(x_j\right) = \psi\left(\frac{x_j - t_{ij}}{d_{ij}}\right), \qquad d_{ij} \neq 0 \quad (1.25)$$

Ou t et d représentent les paramètres de translation et de dilatation respectivement.

Une ondelette mère de type chapeau mexicain peut être employée :

$$\psi(x) = \frac{1}{\sqrt{|d|}}(1 - 2x^2)exp\left(\frac{-x^2}{2}\right) \quad (1.26)$$

Dans la partie antécédente des règles floues exprimées dans (1.24), des fonctions d'appartenance gaussiennes peuvent être utilisées et sont données par l'équation suivante :

$$A_j^i\left(x_j\right) = exp\left(-\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right) \quad (1.27)$$

Ou c_j^i et σ_j^i représentent respectivement le centre et la demi-largeur de la fonction d'appartenance.

La valeur de vérité de la règle R^i peut être obtenue par l'équation suivante :

$$\mu_i(x) = \prod_{j=1}^{q} A_j^i\left(x_j\right) \quad (1.28)$$

Finalement, la sortie défuzzifiée est calculée par :

$$u = \frac{\sum_{i=1}^{N} \mu_i y_i}{\sum_{i=1}^{N} \mu_i} \tag{1.29}$$

1.5 Optimisation Du SIF

1.5.1 Introduction

L'optimisation d'un système d'inférence flou consiste à trouver une représentation paramétrique adéquate, de choisir l'algorithme d'optimisation correspondant aux données disponibles et parfois établir un jeu de contraintes permettant de préserver l'interprétabilité des règles floues tout le long du processus d'optimisation, il s'agit alors d'une optimisation sous contraintes. D'une manière générale, l'optimisation d'un SIF se décompose en deux principales catégories :

- **Optimisation paramétrique** :

Dans cette catégorie, il s'agit d'ajuster les paramètres caractérisant les SIF à savoir les paramètres des fonctions d'appartenance (d'entrées et de sorties) et les conclusions des règles

- **Optimisation structurelle :**

Le but essentiel cherché dans cette catégorie est de représenter le SIF avec un nombre réduit de règles en plus, il s'agit de déterminer le nombre de sous-ensembles flous pour chaque entrée ; les opérateurs de conjonction, de disjonction et d'implication ; la technique de déffuzification et la sélection des variables d'entrée. Cependant l'optimisation structurelle est assez complexe et reste un problème ouvert.

1.5.2 Les Méthodes d'Optimisation

On peut classer les différentes méthodes d'optimisation en trois catégories principales :

1.5.2.1 Les méthodes analytiques :

Elles sont basées sur les méthodes du gradient [60], ces méthodes ne sont applicables que si les fonctions considérées sont dérivables, ce qui les rend non appropriées pour l'optimisation structurelle ou l'optimisation sous contraintes. Elles sont donc utilisées pour l'optimisation paramétrique des SIF qui suppose que la structure est définie et qu'il ne reste qu'à optimiser les paramètres des conclusions des règles (SIF du type Takagi-Sugeno) et les paramètres des fonctions d'appartenance. Ces méthodes imposent le choix des fonctions d'appartenance dérivables (Gaussienne) et d'une conjonction dérivable (Le produit).

La méthode analytique la plus utilisée est celle du gradient qui consiste à minimiser un certain critère J égal à la moyenne de l'erreur quadratique entre la sortie désirée et la sortie actuelle du SIF.

Les paramètres du SIF sont incrémentés selon la loi suivante :

$$\Delta p = -\delta \frac{\partial J}{\partial p} \qquad (1.30)$$

Avec p : paramètres du SIF à optimiser

δ : le gain

J : la fonction coût.

L'inconvénient majeur de ces méthodes est que l'interprétabilité (la sémantique) des règles floues n'est pas assurée.

1.5.2.2 Les Méthodes Evolutionnaires :

Les méthodes évolutionnaires ou les algorithmes évolutionnaires (AE) sont des méthodes d'optimisation et de recherche stochastique inspirées par de diverses formes d'évolution (par exemple naturelle, sociale, …, etc.). Ces méthodes sont implémentées par des algorithmes basés sur des populations. Plusieurs variantes d'algorithmes existent en se basant sur le comportement des différentes populations, par exemple les fourmis, les bactéries et les oiseaux.

Les algorithmes évolutionnaires recouvrent principalement les algorithmes génétiques (AG) développés par Holland en 1975, les stratégies d'évolution (SE) développées par Rechenberg en 1973, optimisation par les essaims particulaires (PSO) présenté par Kennedy et Eberhart en 1995, les algorithmes de colonies de fourmis (ACO) proposés par Dorigo et les algorithmes à évolution différentielle (DE) proposés par Storn et Price en 1995.

a) Les Algorithmes génétiques et les stratégies d'évolution

Les AG agissent sur une population d'individus, codés sous forme de chaîne de caractères (Binaire, réelle) assujettis à une sélection darwinienne :

La survie et la reproduction des individus les mieux adaptés à leur environnement.

Chaque individu représente un point de recherche dans l'espace des solutions, à qui on associe une valeur de fonction coût dont on veut obtenir la valeur maximum. A partir d'une population initiale créée aléatoirement, les AG génèrent de nouveaux individus plus performants en effectuant des opérations génétiques. Les principaux opérateurs génétiques sont :

1) **La reproduction :** C'est un processus dans lequel une nouvelle génération de population est formée par une sélection aléatoire des individus d'une population

existante suivant les valeurs de leur fonction coût. En conséquence, les individus ayant des valeurs élevées de leur fonction coût auront plus de copies dans la prochaine génération.

2) **Le croisement :** Il consiste à échanger des gènes entre deux individus ″parents″ arbitrairement choisis dans la population produisant ainsi de nouveaux individus ″enfants″ qui possèdent certaines caractéristiques génétiques de leurs ″parents″ respectifs. Il peut y avoir un, deux ou un nombre quelconque de points de croisement.
3) **La mutation :** Cet opérateur consiste à altérer un ou plusieurs gènes choisis aléatoirement. Cette altération va prendre des formes différentes selon le type des gènes :
   - Une simple inversion, pour un codage binaire.
   - L'ajout d'un bruit gaussien, pour des réels.

Les opérations de reproduction, croisement et mutation sont répétées autant de fois jusqu'à ce qu'un certain critère d'arrêt (critère de convergence) soit satisfait, par exemple le nombre de générations maximum est atteint.

Les stratégies d'évolution (SE) sont similaires aux AG avec quelques différences qui sont résumées dans le tableau ci dessous :

	AG	SE
Chaîne	Binaire	Réelle
Croisement	Elevé	Faible
Mutation	Faible	Elevé

Tableau 1.1 – Différences entre les AG et les SE

L'avantage de ces méthodes par rapport aux méthodes d'optimisation analytiques, c'est qu'ils ne nécessitent que la connaissance de la valeur de la fonction à optimiser et pas sa dérivée ou autre connaissance auxiliaire ; ainsi ils permettent à la fois l'optimisation (principalement hors ligne) paramétrique et structurelle avec ou sans contraintes pour des systèmes flous dérivables ou non [61].

Thrift [51] a utilisé un AG pour trouver une base de règles floues optimale suivant un apprentissage hors-ligne. Karr [62] a appliqué les AG avec un codage binaire pour trouver les meilleurs paramètres des fonctions d'appartenance pour un contrôleur flou appliqué à la commande du système pendule-chariot. Dans [63], Karr a utilisé un AG pour l'ajustement en-

ligne des fonctions d'appartenance produisant ainsi un contrôleur flou adaptatif. Homaifar et McCormick [45] ont utilisés un AG pour la conception simultanée de l'ensemble des règles floues et les paramètres des fonctions d'appartenance des variables d'entrée. Dans leur approche, les centres des fonctions d'appartenance de type triangulaire sont fixés et seulement les largeurs des bases sont ajustées. Linkens et Nyongesa [47, 64] proposent l'optimisation hors-ligne des règles d'inférence et des fonctions d'appartenance en ajustant simultanément les centres et les bases de ces fonctions. L'AG a été aussi utilisé dans la conception automatique de contrôleurs flous dans le domaine de la robotique mobile [65,66], ou il a été employé pour ajuster à la fois les fonctions d'appartenance et la base des règles floues. Dans [67], une méthode basée sur les SE a été utilisée pour déterminer les règles appropriées d'un système d'inférence flou du type Takagi-Sugeno.

b) optimisation par les essaims particulaires (PSO)

PSO est une méthode d'optimisation non-paramétrique inspirée de la formation d'essaims par des animaux tels que les oiseaux et les poissons [68,69]. L'algorithme PSO travaille sur une population appelée essaim de solutions possibles, elles-mêmes appelées particules, aléatoirement placées dans l'espace de recherche.

A chaque itération, le déplacement des particules est influencé non seulement par leur meilleure position, mais également de la meilleure position de leur voisinage.

Soit $x_i(k)$ la position actuelle de la particule P_i à l'instant k, sa nouvelle position peut être calculée par l'équation suivante :

$$x_i(k+1) = x_i(k) + v_i(k+1) \tag{1.31}$$

La vitesse de chaque particule est mise à jour par l'équation suivante :

$$v_i(k+1) = \omega v_i(k) + \omega_1[xp_i - x_i(k)] + \omega_2[xg - x_i(k)] \tag{1.32}$$

Avec:

$v_i(k)$: La vitesse de la particule P_i à l'instant k.

$x_i(k)$: La position de la particule P_i à l'instant k.

xp_i : La meilleure solution trouvée par la particule P_i .

xg : La meilleure solution globale trouvée jusqu'à l'instant k dans la population.

$\omega_1 = c_1 r_1$, $\omega_2 = c_2 r_2$

Les paramètres $\omega, c_1 et\ c_2$ sont des coefficients constants fixés par l'utilisateur :

$$0 \leq \omega \leq 1.2\ , 0 \leq c_1 \leq 2\ et\ 0 \leq c_2 \leq 2 \tag{1.33}$$

$r_1 et\ r_2$ sont des nombres aléatoires distribués entre 0 et 1 tirés à chaque itération.

L'évaluation de la performance des particules est basée sur l'optimisation d'une fonction objective d'un problème spécifique.

Ainsi, dans le cas d'une minimisation d'une fonction $f(x(k))$, le choix de la meilleure particule (xg) globale est obtenu par :

$$\text{Min } f\big(X(\tau)\big), \qquad \tau = 1,2,\dots,k. \tag{1.34}$$

Avec X est un vecteur qui représente les N particules dans l'essaim.

De manière similaire, la meilleure performance (xp_i) d'une particule P_i est déterminée par :

$$\text{Min } f\big(x_i(\tau)\big), \qquad \tau = 1,2,\dots,k. \tag{1.35}$$

Les algorithmes PSO ont été largement utilisés pour l'ajustement des paramètres des contrôleurs flous (FLC) du type Takagi-Sugeno [70-74] et du type Mamdani [75]. Dans [76], une approche basée sur la combinaison d'un AG avec le PSO a été proposée pour optimiser les paramètres d'un FLC.

c) optimisation par les colonies de fourmis (ACO)

L'algorithme ACO est inspiré du comportement collectif des fourmis qui consiste à trouver le plus court chemin qui sépare leur nid de la source de nourriture [77].

Dans l'ACO, le problème d'optimisation est représenté sous forme d'un graphe composé d'un ensemble de sommets et d'un ensemble d'arcs reliant les sommets. Ainsi, une solution particulière S consiste en une concaténation de composantes dénotées c_{ij} indiquant le sommet *i* et l'arc reliant ce sommet avec le sommet *j*. Cette solution forme donc un chemin d'un sommet initial à un sommet final. Dans les problèmes de commande des systèmes, le sommet final correspond au régime permanent désiré.

A chaque arc est associé deux valeurs :

τ_{ij} : Une variable phéromone qui représente la connaissance acquise sur la solution optimale.

η_{ij} : Une variable heuristique qui fournit une information à priori sur la qualité de la composante d'une solution (la qualité du déplacement d'un nœud i à un nœud j)

Le fonctionnement d'un algorithme ACO de base est comme suit ; un ensemble de M fourmis est aléatoirement réparti sur des sommets. Pour chaque fourmis *c*, on associe une solution partielle S_{pc} initialement vide, les variables phéromone sont initialisées à τ_0 et les variables heuristiques sont fixer de façon à favoriser le choix de quelques sommets par rapport à d'autres.

A chaque itération, chaque fourmi choisis la composante c_{ij} à ajouter à sa solution partielle S_{pc} avec une probabilité donnée par :

$$p_c\{j\backslash i\} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l\in\aleph_i}\tau_{il}^{\alpha}\eta_{il}^{\beta}} \ ,\forall j \in \aleph_i \tag{1.36}$$

Avec :

$p_c\{j\backslash i\}$: La probabilité pour qu'une fourmi c sur un sommet i se déplace au sommet j existant dans son voisinage $\aleph_i$.

$\aleph_i$: Voisinage représentant l'ensemble des sommets pas encore visité connectés au sommet i.

$\alpha \ et \ \beta$: Deux paramètres qui contrôlent l'importance relative entre τ_{ij} et η_{ij} respectivement.

En se déplaçant d'un nœud i à un nœud j, chaque fourmi ajoute la composante associée c_{ij} à sa solution partielle S_{pc} jusqu'à atteindre le nœud final produisant ainsi une solution candidate. Cette dernière est évaluée en utilisant une fonction d'adaptation $F(S)$ qui sera par la suite utilisé pour la mise à jour des phéromones selon la formule :

$$\tau_{ij} \longleftarrow (1-\rho)\tau_{ij} + \rho \sum_{S\in\mathcal{L}_{upd}} \Delta\tau_{ij}(S) \tag{1.37}$$

Avec :

ρ : Coefficient d'évaporation des phéromones $\rho \in [0\ 1]$.

$\mathcal{L}_{upd}$: L'ensemble des solutions susceptibles d'être employées pour la mise à jour des phéromones [78].

$\Delta\tau_{ij}(S)$: représente la quantité de phéromones déposée calculée par :

$$\Delta\tau_{ij}(S) = \begin{cases} F(S), & si \ c_{ij} \in S \\ 0 \ , & sinon \end{cases} \tag{1.38}$$

Dans la prochaine itération, chaque fourmi répète les étapes précédentes en utilisant les valeurs des phéromones mis à jour pour mieux choisir les sommets vers lesquels se déplacer.

Ce processus continue jusqu'à ce qu'un certain critère d'arrêt soit atteint, produisant ainsi les phéromones codant la solution du problème d'optimisation.

Plusieurs travaux scientifiques ont considérés la combinaison de l'ACO avec les SIF [78-81]. Dans la plus part de ces travaux, les ACO ont été utilisés pour l'optimisation des paramètres des contrôleurs flous.

d) Les algorithmes à évolution différentielle (DE)

Les algorithmes DE sont des méthodes de recherche stochastiques parallèles qui utilisent un certain nombre de vecteurs formant des solutions candidates du problème d'optimisation multidimensionnel [82-83].

Chaque vecteur d'une génération G est de dimensions D, il contient les paramètres de décision du problème d'optimisation, il est noté par :

$$X_{i,G} = \left[x_{1i,G}, x_{2i,G}, \dots, x_{Di,G}\right] \ , \ i = 1, \dots \ , NP \tag{1.39}$$

Une population initiale de vecteurs est crée aléatoirement. Cette population doit couvrir l'espace de recherche total des paramètres en tenant compte des limites définies pour chaque paramètre. Une fois l'initialisation terminée, l'algorithme DE de base génère de nouveaux vecteurs en employant trois opérations : Mutation, Croisement et Sélection.

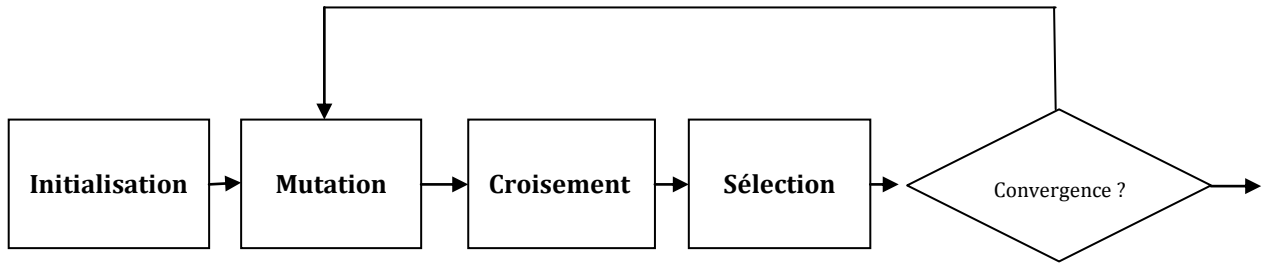


Fig.1.8: Opérations de l'algorithme DE

1) La mutation : Pour chaque vecteur $X_{i,G}$, $i = 1, \dots \ , NP$, un vecteur mutant est crée par :

$$V_{i,G+1} = X_{i,G} + F\left(X_{r1,G} - X_{r2,G}\right) \tag{1.40}$$

Avec :

$r_1 et\ r_2$: Des indices aléatoires appartenant à $\{1,2, \dots, NP\}$ et différent de i.

F : Un facteur réel constant $\in$ $[0\ 2]$.

2) Le croisement : L'opération de croisement consiste à produire de nouveaux vecteurs (U) en mélangeant les vecteurs sources (X) et les vecteurs mutés (V) comme suit :

$$U_{i,G+1} = \left(u_{1i,G+1}, u_{2i,G+1}, \dots, u_{Di,G+1}\ \right) \tag{1.41}$$

Avec :

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & si\ (rand\ (j) \leq CR)\ ou\ j = mbr\ (i) \\ x_{ji,G} & si (rand\ (j) > CR)\ ou\ j \neq mbr\ (i) \end{cases} , i = 1,2, \dots, NP\ et\ j = 1,2, \dots, D. \tag{1.42}$$

Ou :

$rand(j)$: Un nombre aléatoire uniformément distribué dans l'intervalle $[0\ 1]$.

CR : Une constante de croisement $\in$ $[0\ 1]$.

$mbr(i)$: Un indice choisi aléatoirement dans $[1\ D]$.

3) La sélection : L'opération de sélection est décrite par :

$$X_{i,G+1} = \begin{cases} U_{i,G+1}\ , & f(U_{i,G+1}) < f\left(X_{i,G}\right) \\ X_{i,G}\ , & f(U_{i,G+1}) \geq f\left(X_{i,G}\right) \end{cases} \tag{1.43}$$

Avec $f(x)$ est une fonction d'adaptation.

L'utilisation des algorithmes DE pour la conception des SIF a été considérée dans plusieurs travaux [84-86]. Dans [84], l'algorithme DE a été employé pour optimiser les paramètres de pondérations des entrées/sortie d'un contrôleur flou adaptatif. Dans [85], une structure hiérarchique d'un contrôleur flou a été proposée afin de réduire le nombre de règles floues en une fonction linéaire des variables d'entrée, et l'algorithme DE a été utilisé pour la conception automatique du contrôleur flou hiérarchique. Une méthode d'optimisation basée sur les DE pour l'extraction des règles floues d'un FLC a été considérée dans [86].

1.5.2.3 Les méthodes de classification :

Les méthodes de classification peuvent être utilisées à la fois pour l'optimisation supervisée et non supervisée.

- La classification supervisée part d'un étiquetage à priori des vecteurs de la base d'apprentissage. Il s'agit alors de déterminer les prototypes (les vecteurs les plus représentatifs) de chaque classe. Un vecteur d'entrée inconnu est ensuite classé en fonction de sa distance aux différents prototypes.
- La classification non supervisée ne présuppose pas de connaissances sur les classes, ni même sur leur nombre, le problème est de repérer des groupements de points significatifs dans l'espace de données.

L'algorithme de classification, probablement, le plus utilisée est celui des C-moyennes floues (FCM ; Fuzzy C-Means), le principe de cette méthode a été évoqué au paragraphe (1.4.2) de ce chapitre.

1.6 Conclusion

Dans ce chapitre, nous avons vu que la conception des systèmes d'inférence flous à partir des données numériques peut être décomposée en deux phases principales : génération automatique des règles du SIF et l'optimisation du SIF.

Il existe plusieurs méthodes de génération de règles, à savoir ; les méthodes basées sur le partitionnement flou, les méthodes basées sur les groupements flous et les méthodes basées sur les techniques du Soft Computing.

Quant à l'optimisation du système, elle peut être faite en deux niveaux : optimisation paramétrique (ajustement des fonctions d'appartenance et les conclusions des règles) et l'optimisation structurelle (réduction de la base de règles, sélection des variables d'entrée, optimisation du nombre de sous-ensembles flous,…).

Le problème de l'optimisation d'un SIF consiste donc à :

- trouver une représentation paramétrique adéquate pour un SIF.

- Etablir un jeu de contraintes permettant de préserver l'interprétabilité des règles floues tout le long du processus d'optimisation.
- Choisir l'algorithme d'optimisation correspondant aux données disponibles.

Plusieurs techniques d'optimisation des SIF sont disponibles et sont classées en trois catégories : les méthodes analytiques, les méthodes évolutionnistes et les méthodes de classification.

Les méthodes analytiques s'adaptent bien pour l'optimisation paramétrique et sont non appropriées pour l'optimisation structurelle ou l'optimisation sous contraintes, les méthodes de classification sont utilisées pour l'optimisation structurelle dans un contexte supervisé ou non-supervisé, les méthodes évolutionnistes quant à elles, permettent à la fois l'optimisation paramétrique et structurelle avec ou sans contraintes pour des systèmes flous dérivables ou non.

Chapitre II :

Optimisation Multi-objectif Par Les Algorithmes Evolutionnaires

2.1 Introduction

Dans beaucoup de problèmes pratiques, on est souvent confronté à optimiser plus d'un critère de performance à la fois, on se trouve donc face à un problème à objectifs multiples ou à critères multiples. Ces critères peuvent être en conflit tel que maximiser la vitesse et la sécurité dans une voiture, et il n'est pas possible (ni souhaitable) de les combiner en un objectif unique ou de les réduire de manière à n'avoir qu'un seul critère à optimiser.

Ces problèmes attirent depuis longtemps l'attention des chercheurs qui utilisent des techniques d'optimisation et d'exploration traditionnelles. Elles consistent à allouer des poids à chacun des objectifs afin d'indiquer leur importance dans le problème, mais ces méthodes sont très subjectives et il est difficile de trouver des poids qui peuvent refléter avec précision une situation réelle.

L'utilisation des méthodes évolutionnaires et les techniques d'optimisation parallèle tel que les AG, permet de trouver un ensemble de solutions optimales de Pareto. Dans ce chapitre, on va présenter quelques approches évolutionnaires les plus populaires utilisées pour l'optimisation multi-objectif.

2.2 Le problème Multi-objectif

Un problème d'optimisation multi-objectif consiste à optimiser plusieurs objectifs à la fois. Ce problème est considéré comme spécial dans le sens qu'il n'a pas une solution unique, en effet on ne peut pas trouver une solution pour laquelle tous les objectifs sont optimaux. Cependant la solution d'un problème multi-objectif consiste en un ensemble de solutions.

2.2.1 Formulation du problème Multi-objectif

Un problème d'optimisation multi-objectif général peut être décrit par une fonction vecteur f qui lie entre un ensemble de m paramètres (variables de décision) et un ensemble de n critères comme suit :

$$\text{Minimiser/Maximiser} \quad y = f(x) = (f_1(x),\ f_2(x), \dots, f_n(x)) \tag{2.1}$$

$$\text{Avec :} \quad x = (x_1, x_2, \dots, x_m) \in X$$

$$y = (y_1, y_2, \dots, y_n) \in Y$$

Où x : Vecteur de décision

X : L'espace des paramètres

y : Vecteur des critères (objectifs)

Y : L'espace des critères (objectifs)

L'ensemble des solutions de ce problème d'optimisation correspond à tous les vecteurs de décision connus sous le nom de l'ensemble optimal de Pareto telle qu'une solution ne peut

être améliorée pour un des objectifs qu'au détriment de la dégradation d'au moins un autre objectif.

2.2.2 Les Méthodes Classiques pour l'Optimisation Multi-objectif

Dans les techniques d'optimisation classiques, le problème multi-objectif est converti en un problème d'optimisation à objectif unique en regroupant tous les objectifs en une fonction scalaire $S : X \longrightarrow \Re$. Généralement cette fonction consiste en une somme pondérée des objectifs et elle est formulée comme suit :

$$S(x) = \sum_{i=1}^{n} W_i \, f_i(x) \tag{2.2}$$

Avec W_i: Des poids réels.

f_i: Les fonctions objectives.

n : Le nombre d'objectifs.

La méthode d'optimisation essaie donc de minimiser une distance d par rapport à un vecteur objectif désiré y_d :

$$d(x) = \|f(x) - y_d\| \tag{2.3}$$

ou $\| \ . \|$ est la norme euclidienne

Cependant, résoudre un problème d'optimisation multi-objectif par des techniques d'optimisation à objectif unique présente beaucoup d'inconvénients :

- Le résultat d'une telle optimisation donne une solution unique (un point unique), cette solution peut être optimale par rapport à un critère et en même temps moins optimale par rapport à un autre, en plus cette solution peut même être dominée.
- Il est difficile de déterminer avec précision les valeurs des poids, surtout lorsqu'on n'a pas suffisamment d'information ou de connaissance à propos du problème d'optimisation.

Pour ces raisons, l'application de telles méthodes pour la résolution des problèmes multi-objectifs n'est pas toujours efficace.

2.2.3 Concept d'optimalité de Pareto

Mathématiquement le concept d'optimalité de Pareto est défini en terme de dominance comme suit :

Considérons un problème de maximisation et soit deux vecteurs de décision a et $b \in X$, on dit que a domine b et on écrit $a > b$ si et seulement si :

$$\forall\, i \in \{1,2,\dots,n\}: f_i(a) \geq f_i(b)\ et\ \exists\, j \in \{1,2,\dots,n\}: f_j(a) > f_j(b) \tag{2.4}$$

Tous les vecteurs de décision qui ne sont pas dominés par aucun autre vecteur de décision sont dits non-dominés ou efficaces. L'ensemble des solutions non-dominés est appelé l'ensemble optimal de Pareto ou le front optimal de Pareto.

2.3 Optimisation Multi-objectif par les algorithmes évolutionnaires

Du à leur parallélisme inhérent dans l'exploration de l'espace de recherche et à leur aptitude d'agir sur une population de solutions, les algorithmes évolutionnistes (AE) sont capables d'obtenir un ensemble de points optimal de Pareto et sont donc très appropriés pour l'optimisation à objectifs multiples [87-90]. Dans [91], une grande variété d'algorithmes évolutionnaires multi-objectifs (MOEA) a été citée et classifiée, tout en présentant leur développement durant les dernières années.

Dans le paragraphe suivant, on va présenter quelques approches évolutionnaires les plus populaires.

2.4 Optimisation Multi-objectif par les AG

L'utilisation des algorithmes génétiques pour l'optimisation multi-objective implique deux grands problèmes :

- Comment accomplir l'attribution de la fonction d'adaptation (fitness assignment) et la sélection respectivement dans le but de diriger la recherche vers l'ensemble optimal de Pareto.
- Comment préserver la diversité dans la population afin d'éviter une convergence prématurée et d'obtenir un front bien distribué.

2.4.1 Recherche Multimodale et Préservation de la Diversité

Dans le cas d'une optimisation multi-objectif, on cherche un ensemble de solutions non-dominées au lieu d'une solution unique, pour cela l'algorithme génétique multi-objectif doit accomplir une recherche multimodale qui explore le front de Pareto uniformément. Malheureusement, un AG simple tend à converger vers une solution unique et maintes fois, des solutions sont perdues ; ceci est dû aux erreurs stochastiques associées à ses opérateurs génétiques. Afin de surmonter à ce phénomène de convergence des AG, connu sous le nom de la dérive génétique (genetic drift), plusieurs méthodes ont été développées qui tentent de préserver la diversité dans la population. Ces méthodes peuvent être classées en deux catégories : techniques de création de sous populations appelées niches (*Niching Techniques*, *NT*) et techniques directes sans création de niches (*Non Niching Techniques*, *NNT*).

Les NT sont les plus répandues dans le domaine des AG multi-objectif, ceci est du à leur capacité de former et de maintenir des sous-populations ou niches stables.

a) Les NT : Niching techniques

La méthode la plus utilisée est celle du partage de la fonction d'adaptation proposée par Goldberg et Richardson [92]. Elle est basée sur l'idée que les individus dans une niche particulière, ont à partager les ressources disponibles (similaire à la nature). Ainsi, pour une valeur de fonction d'adaptation la plus dégradée d'un certain individu, on trouve la plus part des individus localisés dans son voisinage (à proximité) exprimé en terme d'une mesure de distance $d(i,j)$ et spécifié par le rayon de la niche σ_{Share}. La fonction d'adaptation partagée (*The shared fitness*) S_i d'un individu *i* est donnée par :

$$S_i = \frac{f_i}{m_i} \tag{2.5}$$

Ou f_i: fitness de l'individu *i*

m_i : donné par

$$m_i = \sum_{j=1}^{n} Sh[d(i,j)] \tag{2.6}$$

Avec $d(i,j)$: Distance entre l'individu i et l'individu j

$Sh[d]$: Fonction de partage qui a souvent la forme suivante :

$$Sh[d(i,j)] = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_{Share}}\right)^{\alpha}, d(i,j) \leq \sigma_{Share} \\ 0 \quad , \quad Sinon \end{cases} \tag{2.7}$$

En outre, selon que la distance $d(i,j)$ considère les génotypes (gènes) ou les phénotypes (les paramètres décodés), on distingue le partage génotypique (*genotypic sharing)* et la partage phénotypique (*phenotypic sharing*). Le partage phénotypique peut être réalisé sur les vecteurs de décision ou sur les vecteurs objectifs [93].

b) Les NNT : Non-niching techniques

Parmi ces méthodes, la plus répandue dans l'optimisation multicritère est appelée combinaison restreinte (*restricted mating*). L'idée de base de cette technique est que deux individus ne sont autorisés à se reproduire que seulement s'il y a une certaine distance entre eux. Néanmoins, cette méthode n'est pas très largement utilisée dans les AG multi-objectif.

2.4.2 Sélection et Stratégies d'Attribution De La Fonction d'Adaptation

La sélection dans un AG est un mécanisme qui permet de concentrer la recherche dans des régions prometteuses de l'espace de recherche tout en s'appuyant sur la fonction d'adaptation. Dans le cas d'objectifs multiples, l'opérateur de sélection doit diriger la

recherche vers le front non-dominé, ce qui fait que la valeur de la fonction d'adaptation de chaque individu reflète son utilité à l'égard de l'optimalité de Pareto, ainsi l'attribution de la fonction d'adaptation est l'un des principaux points dans une optimisation multi-objectif [94]. Souvent différentes approches utilisées pour l'optimisation multicritère sont classifiées selon les stratégies d'attribution de la fonction d'adaptation. On distingue trois stratégies : sélection de critère, agrégation des critères et sélection de Pareto.

a) Sélection de critère: (ou fitness evaluation by changing objectives)

Les méthodes utilisant cette stratégie commute entre les différents critères durant la phase de sélection. A chaque instant un individu est choisi pour la reproduction, potentiellement un critère différent va décider quel membre de la population va être copié dans la nouvelle population. Parmi les méthodes qui utilisent cette approche de sélection, on distingue l'algorithme VEGA (Vector Evaluated Genetic Algorithm) développé par Shaffer en 1985 [95]. C'était le premier algorithme qui a traité séparément les critères en exploitant les propriétés parallèles des algorithmes génétiques dans le but de trouver des solutions non-dominées.

VEGA effectue une sélection pour chaque critère séparément. En effet étant donné une population de taille P et un nombre de critères d'optimisation égal à m, chaque membre de la population est évalué pour son adaptation par rapport à chacun des m critères. Une sous-population de taille P/m est extraite de l'ancienne population utilisant une sélection proportionnelle basée sur les fonctions d'adaptation relatives au critère en considération. On obtient ainsi plusieurs sous-populations relatives à chaque critère, ensuite ces sous-populations sont mélangées et groupées dans une nouvelle population qui sera soumise aux opérations génétiques usuelles à savoir le croisement et la mutation. Ce processus est illustré dans la figure (2.1).

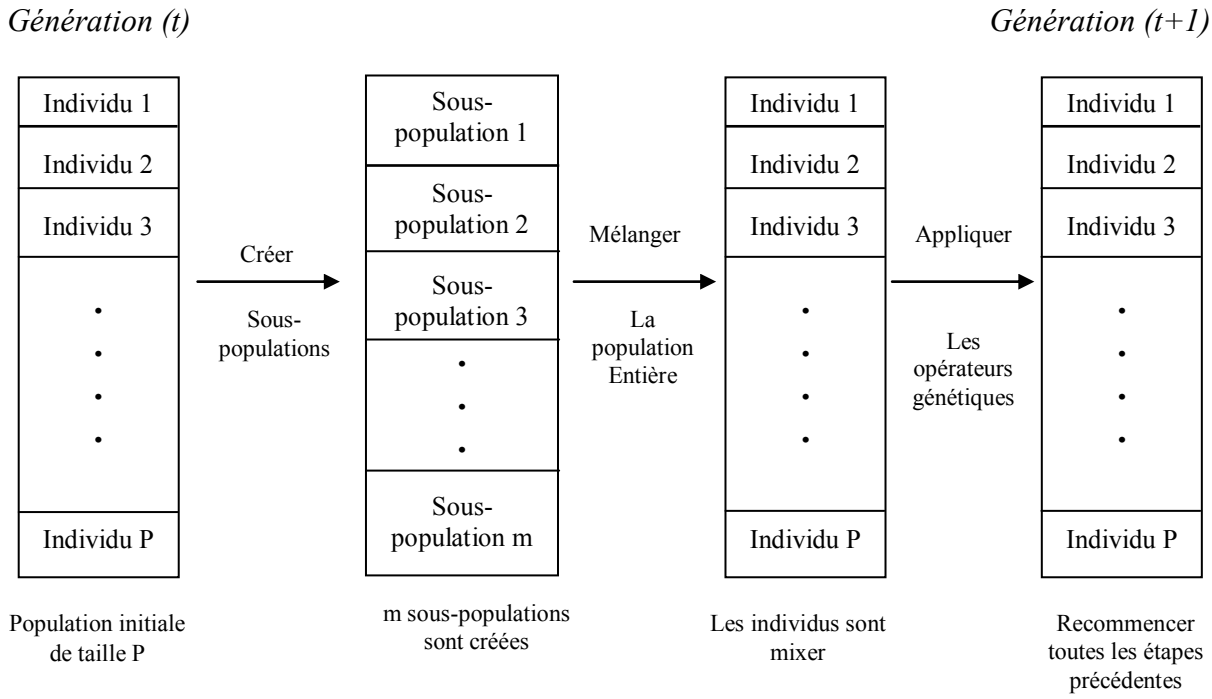


Fig. 2.1 : Schéma de la sélection VEGA

b) Agrégation des critères: (ou Scalarization with changing parameters)

Ce type de sélection est basé sur les approches traditionnelles d'optimisation multi-objectif où les critères multiples sont combinés pour former une fonction objective unique paramétrée. Les paramètres de la fonction résultante sont systématiquement variés durant la même phase d'exécution afin de trouver un ensemble de solutions optimales de Pareto.

Parmis les méthodes utilisant cette sélection, on distingue l'algorithme HLGA (Hajela and Lin's Genetic Algorithm) proposé par Hajela et Lin en 1992 [96]. Dans cet algorithme, la méthode de la somme pondérée est considéré pour l'attribution de la fonction d'adaptation où à chaque critère est associé un poids $W_i = [0, 1]$ tel que $\sum W_i = 1$.

Ainsi la valeur de la fonction d'adaptation scalaire est calculée en effectuant la somme pondérée des fonctions objectives : $\sum W_i \, f_i(x)$.

Pour rechercher de multiples solutions en parallèle, les poids ne sont pas fixés, mais ils sont codés dans le chromosome (génotype). La diversité des combinaisons des poids est ensuite promue par un partage phénotypique de la fonction d'adaptation.

c) Sélection de Pareto

Goldberg [97] était le premier à proposer une stratégie se basant sur le principe de la non-domination de Pareto. Ceci a fait paraître différents types d'algorithmes génétiques multi objectif, connus sous le nom approches Pareto.

Dans la méthode de Goldberg, les individus sont classés d'une manière itérative : Au début, toutes les solutions non-dominées sont identifiées et on leur attribue un rang 1, ces points sont ensuite enlevés de la population et l'ensemble suivant des individus non-dominés est identifié et assigné un rang 2. Ce processus continue jusqu'à ce que tous les individus de la population aient un rang qui sera considéré comme une mesure d'adaptation. Après cela, les comptes de reproduction ou les probabilités de sélection peuvent être calculées selon le rang.

Les approches Pareto les plus populaires sont : MOGA (Multiobjective Genetic Algorithm), NPGA (Niched Pareto Genetic Algorithm), NSGA (Nondominated Sorting Genetic Algorithm), SPEA (Strength Pareto Evolutionary Algorithm) et NSGAII.

1) Multiobjective Genetic Algorithm (MOGA)

Cet algorithme a été proposé par Fonseca et Fleming en 1993 [98], la méthode d'attribution de la fonction d'adaptation utilisée dans cet algorithme est une extension de l'attribution de fonction d'adaptation standard basée sur le classement (*Ranking)* dont le principe est le suivant :

Considérons un individu x_i de la génération t qui est dominé par $p_i^{(t)}$individus dans la population courante, alors la valeur de son rang est donnée par :

$$Rang(x_i, t) = 1 + p_i^{(t)} \tag{2.8}$$

Et tous les individus non-dominés sont assignés le même rang (Fig. 2.2)

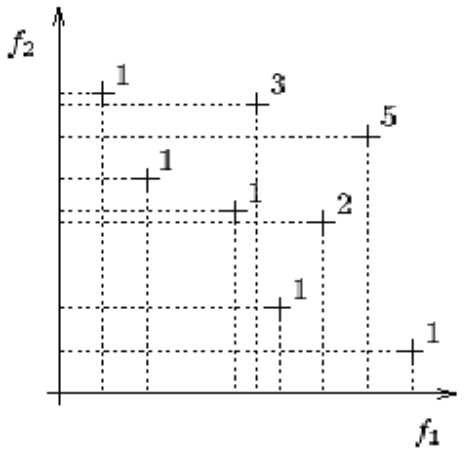


Fig. 2.2 : Classement multiobjectif

Il faut noter qu'à une génération donnée, tous les rangs ne sont pas nécessairement représentés dans la population, comme l'indique la figure (2.2) où le rang 4 est absent.

Une fois que tous les individus auront un rang, l'attribution de la fonction d'adaptation est réalisée de la manière suivante :

(1) Classer la population suivant les rangs des individus.

(2) Assigner une fonction d'adaptation aux individus en réalisant une interpolation du meilleur individu (rang 1) au mauvais (rang $n \leq N$) selon une certaine fonction qui peut être linéaire ou non-linéaire.

(3) Prendre la moyenne des fonctions d'adaptation des individus avec le même rang, afin de s'assurer que tous les individus égaux seront sélectionnés avec la même probabilité.

Cependant, cette méthode peut provoquer une convergence prématurée. Pour éviter cela, Fonseca et Fleming ont introduit dans leur algorithme une technique de formation des niches afin de distribuer la population sur la région optimale de Pareto en utilisant un partage sur les valeurs des fonctions objectives.

Ce qu'il faut noter c'est que MOGA est une approche efficace et facile à implémenter, mais comme toutes les autres techniques de classement de Pareto, sa performance est fortement dépendante du choix approprié de la distance phénotypique σ_{Share}, Fonseca et Fleming ont développé une méthodologie pour calculer une telle valeur.

2) Niched Pareto Genetic Algorithm (NPGA)

Cet algorithme a été proposé par Horn et Nafpliotis en 1993 [93], il combine la sélection par tournoi et le concept de dominance de Pareto. Généralement Dans une sélection par tournoi, un ensemble d'individus est aléatoirement choisi dans la population courante et le meilleur de ces sous-ensembles sera reproduit dans la prochaine population. Dans leur algorithme, ils ont modifié cette technique comme suit :

- Deux individus concurrents (candidats) pour la sélection sont choisis aléatoirement dans la population.
- Un ensemble d'individus de comparaison de taille t_{dom} est aussi choisi dans la population.
- Chacun des deux candidats est ensuite comparé à chaque individu de l'ensemble de comparaison. Si un candidat est dominé par l'ensemble de comparaison et l'autre ne l'est pas, alors le dernier est sélectionné pour la reproduction. Si les deux individus sont dominés ou non-dominés par l'ensemble de comparaison, alors on utilise un partage pour choisir le gagnant du tournoi.
- Le partage consiste en la dégradation de la fonction objective f_i d'un individu par un paramètre m_i calculé pour cet individu. Cette dégradation est obtenue en divisant la fonction objective par (f_i/m_i). Le paramètre m_i est calculé sur l'ensemble des individus de la population courante comme suit :

$$m_i = \sum_{j \in pop} Sh[d(i,j)] \tag{2.9}$$

Avec $d(i,j)$: distance phénotypique (sur le vecteur objectif $(f_1(x), f_2(x), \dots, f_n(x))$ entre les individus i et j.

$Sh[d]$: fonction de partage; c'est une fonction décroissante de d tel que :

$$\begin{cases} Sh[0] = 1 \\ et \\ Sh[d \geq \sigma_{Share}] = 0 \end{cases} \quad (2.10)$$

Typiquement la fonction de partage triangulaire est utilisée où :

$$\begin{cases} Sh[d] = 1 - \dfrac{d}{\sigma_{Share}}, & d \leq \sigma_{Share} \\ Sh[d] = 0 \quad , & d > \sigma_{Share} \end{cases} \quad (2.11)$$

où σ_{Share} est le rayon de la niche, fixé par l'utilisateur selon une certaine estimation de la distance minimale désirée ou prédite entre les pics désirés.

- Le candidat gagnant du partage est celui ayant un nombre réduit d'individus dans sa niche (c.à.d. celui qui a le plus petit m_i).

```
function selection     {retourne un individu de la population courante S}
begin
      mélanger(random_pop_index) ;
      Candidat_1 := random_pop_index[1] ;
      Candidat_2 := random_pop_index[2] ;
      Candidat_1_dominé := faux ;
      Candidat_2_dominé := faux ;
      for comparison_set_index = 3 to t_dom+3 do
         begin
              comparison_individual := random_pop_index[comparison_set_index];
              if S[comparison_individual] domine S[candidat_1]
                then candidat_1_dominé := vrai;
              if S[comparison_individual] domine S[candidat_2]
                then candidat_2_dominé := vrai;
         end
       if (candidat_1_dominé AND NOT candidat_2_dominé )
         then return candidat_2;
      else if (NOT candidat_1_dominé AND candidat_2_dominé )
            then return candidat_1;
      else {do sharing}
          {calcul des m_i }
          if m_i [candidat_1] > m_i [candidat_2]
            then return candidat_2;
          else return candidat_1;
end;
```

Fig. 2.3 :Pseudo-code du tournoi de domination de Pareto

La figure ci-dessus illustre le pseudo-code du NPGA dans le cas d'un problème de maximisation. Dans ce pseudo-code, S est un vecteur de N individus dans la population

courante et $random\_pop\_index$ est un vecteur contenant les N indices dans un ordre aléatoire.

3) Nondominated Sorting Genetic Algorithm (NSGA)

En 1994, Srinivas et Deb ont développé une approche basée sur la méthode de classification de Goldberg et l'ont appelée NSGA [99]. L'attribution de la fonction d'adaptation est effectuée en plusieurs étapes.

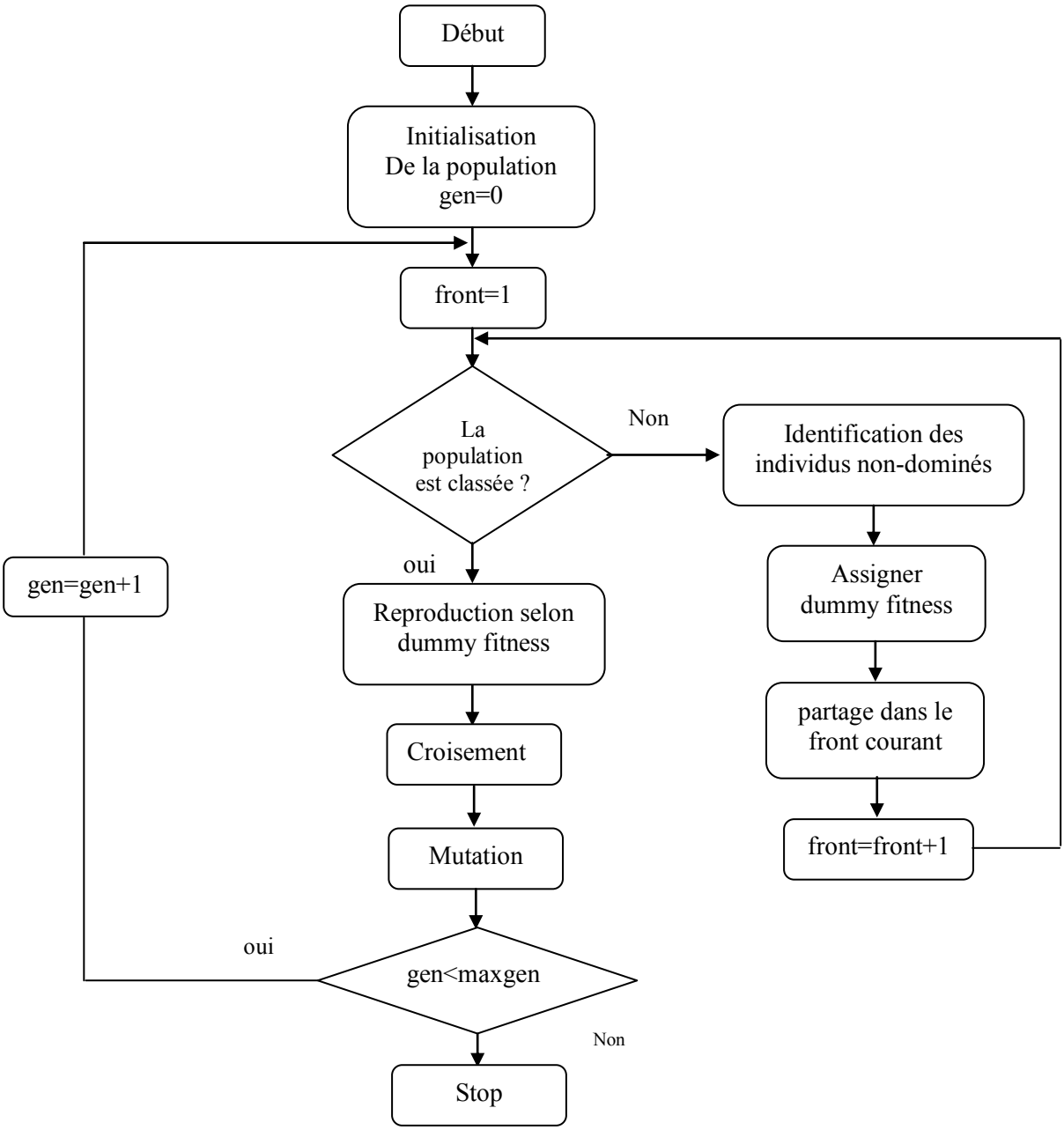


Fig. 2.4 : Organigramme du NSGA

Dans chaque étape, les solutions non-dominées constituant le front non-dominé sont assignées la même valeur de fonction d'adaptation transitoire (*dummy fitness),* et afin de préserver la diversité dans la population, ces solutions sont partagées avec leurs valeurs de fonction

d'adaptation transitoire (partage phénotypique sur le vecteur de décision $(x_1, x_2, \dots, x_n)$) et sont ignorées dans la prochaine phase de classification.

Finalement, la fonction d'adaptation transitoire prend une valeur inférieure à la plus petite valeur de la fonction d'adaptation partagée du front non-dominé courant. Ensuite, le prochain front est obtenu. Cette procédure est répétée jusqu'à ce que tous les individus de la population soient classés. Une sélection à reste stochastique est utilisée pour cette approche. La figure (2.4) illustre l'organigramme de cette approche.

4) Strength Pareto Evolutionary Algorithm (SPEA)

Cet algorithme a été proposé par Zitzler et Thiele en 1999 [94]. Il regroupe les différentes caractéristiques des algorithmes génétiques multi-objectifs précédents dans un seul algorithme, en plus il se caractérise par :

- l'ensemble d'individus contenant des solutions optimales de Pareto générées plus loin est maintenu (sauvegardé).
- Cet ensemble est utilisé pour évaluer la fonction d'adaptation d'un individu selon le principe de dominance de Pareto.
- Une nouvelle méthode de niche est développée afin de préserver la diversité dans la population ; cette méthode est basée sur la dominance de Pareto et ne nécessite aucun paramètre de distance (tel que le rayon de la niche pour le partage).
- Une méthode de groupement est introduite pour réduire l'ensemble de Pareto.

L'algorithme est illustré dans la figure (2.5). Dans cet algorithme, deux populations de taille P et P' sont considérées. A chaque génération, les individus non-dominés dans P sont copiés dans P' et tous les individus dans P' qui sont dominés par un autre individu de P' sont enlevés. Si le nombre des solutions non-dominées externes dépasse un certain maximum N', alors P' est réduite par des groupements. Par la suite, la fonction d'adaptation de chaque individu dans P aussi bien que dans P' est calculée en deux étapes : d'abord, les individus dans l'ensemble non-dominé externe P' sont classés par rapport aux membres de P. Ensuite, les individus dans la population P sont évalués par rapport aux membres de P'.

Le partage de la fonction d'adaptation est inclus dans les deux phases de l'attribution de la fonction d'adaptation avec la principale différence est que les niches ne sont pas définies en terme de distance mais en terme de dominance de Pareto. Finalement, les individus de $(P + P')$ subissent une sélection par tournoi binaire pour la reproduction et enfin les opérateurs de croisement et de mutation sont appliqués.

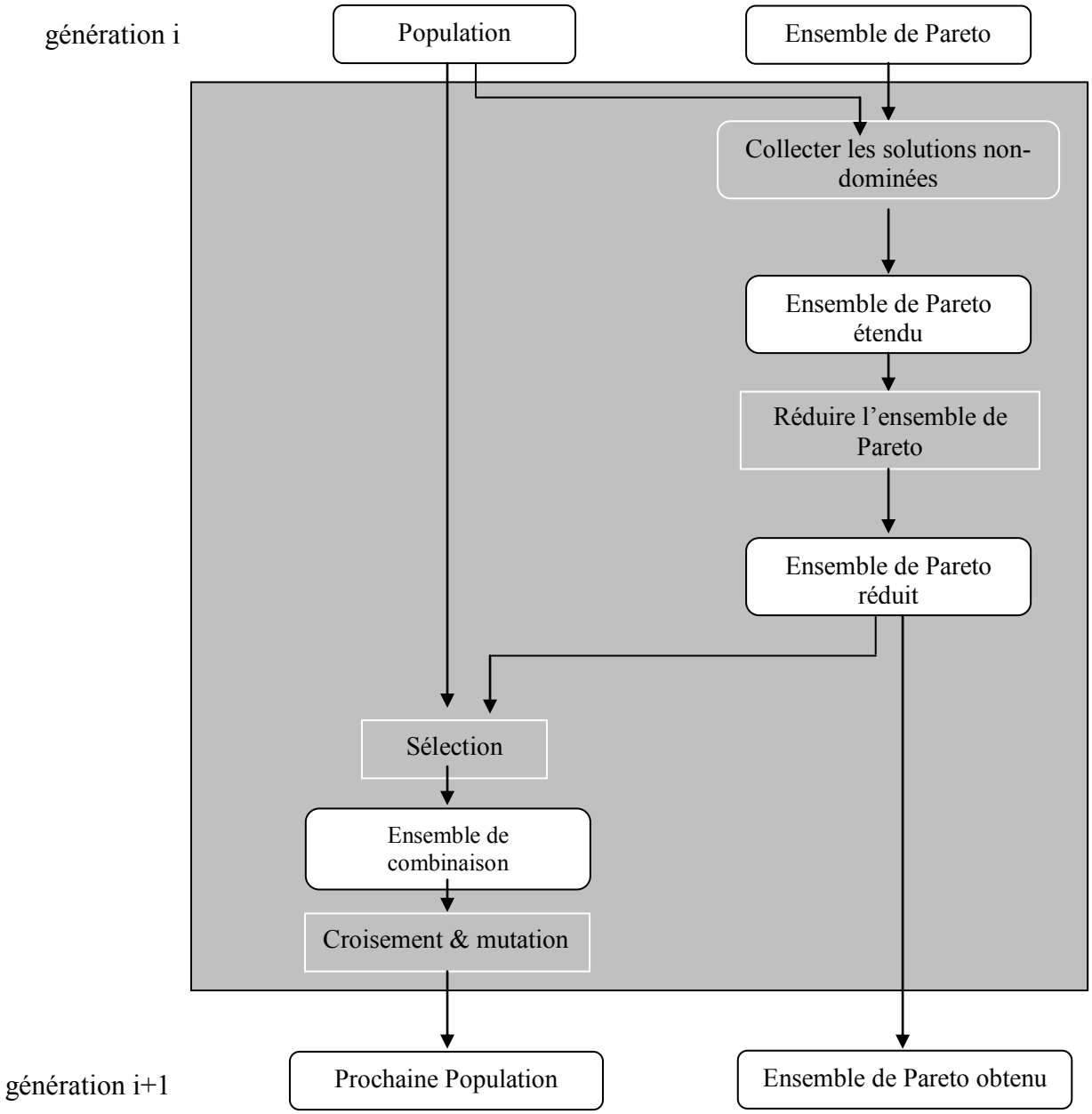


Fig. 2.5: Organigramme du SPEA

5) Nodominated Sorting Genetic Algorithm (NSGA II)

L'algorithme NSGA II [100] est une version modifié du NSGA qui utilise une procédure de tri non-dominée plus rapide et qui assure l'élitisme. Cet algorithme commence par une initialisation usuelle de la population. Ensuite, il y a calcul des différents fronts d'individus non-dominés et la population est triée selon la dominance à chaque front.

Dans cette procédure, deux entités sont calculées pour chaque individu (p); le compte de domination (np) qui indique le nombre d'individu qui domine l'individu (p) et l'ensemble d'individus (Sp) que l'individu (p) domine.

Une fois que le tri non-dominé est terminé, un paramètre appelé encombrement de la distance est calculé pour chaque individu, et alors une sélection par tournoi avec un opérateur de comparaison est faite entre deux individus aléatoirement choisis parmi la population des parents. Ainsi, l'individu avec le nombre de front le plus bas est choisi si les deux individus viennent de différents fronts, et l'individu avec une distance d'encombrement plus élevée est choisi si les deux individus sont du même front. Ensuite, une nouvelle population est produite utilisant les opérateurs génétiques : sélection, croisement et mutation. Enfin, la population combinée est triée selon la non-domination. Dans cet algorithme, l'élitisme est assuré puisque tous les individus précédents et nouveaux sont inclus dans la nouvelle population et seulement les meilleurs individus sont choisis en tant qu'une nouvelle population de parents (Fig.2 .6).

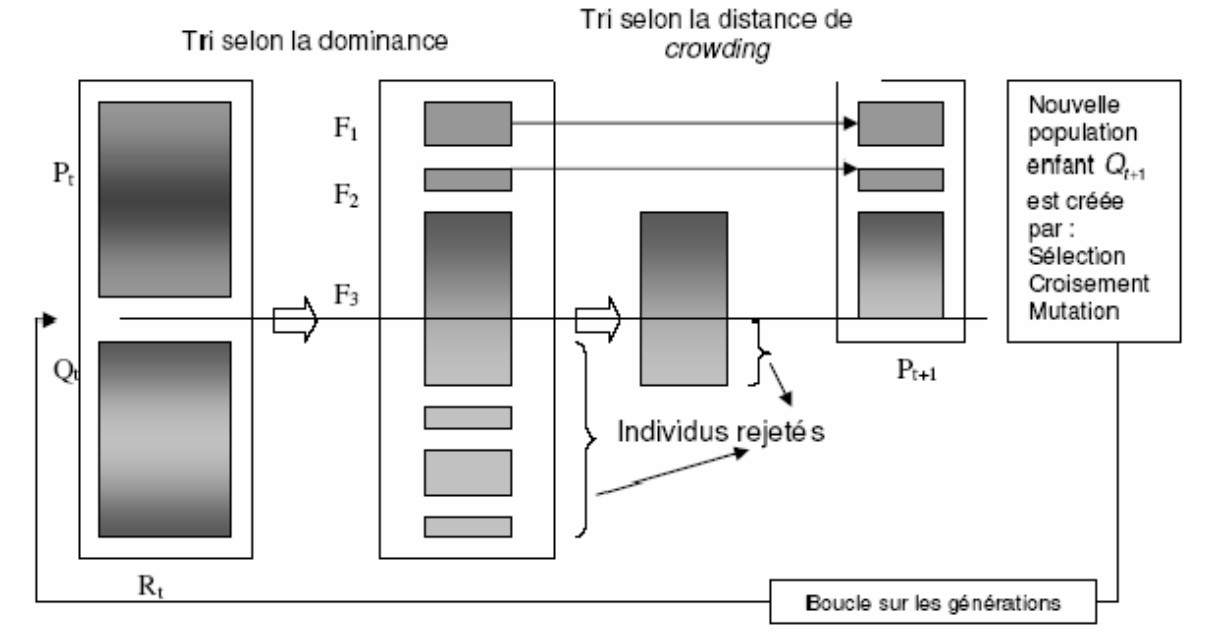


Fig. 2.6 : principe du NSGA II

2.5 Optimisation Multi-objectif par les PSO

Vu leur simplicité et leur succès pour la résolution des problèmes d'optimisation, les PSO ont été considérés pour l'optimisation multi-objective. La première variante des PSO multi-objectif a été proposée par Moore et Chapman [101].

Cependant, l'utilisation des PSO pour l'optimisation multi-objectif implique trois problèmes essentiels :

- Comment sélectionner les meilleures particules (leaders) afin d'avantager les solutions non dominées.
- Comment conserver les solutions non-dominées trouvées durant la phase de recherche.
- Comment maintenir la diversité dans la population.

Plusieurs solutions ont été proposées dans la littérature offrant ainsi plusieurs variantes de MOPSO [102-108]. D'une manière générale, le fonctionnement d'un algorithme MOPSO est illustré par le pseudo-code de la (Fig.2.7) [109].

```
Début
    Initialisation de la population (l'essaim)
    Initialisation des particules Leaders
    Qualité (Leaders)
    G=0
    Tant que G < Gmax
        Pour chaque particule
            Sélectionner Leader
            Mise à jour de la position
            Mutation
            Evaluation
            Mise à jour de P_best
        Fin Pour
        Mise à jour des leaders
        Qualité (Leaders)
        G++
    Fin Tant que
Fin
```

Fig. 2.7 : pseudo-code d'un MOPSO général

Au début, une population initiale d'individus est crée. Ensuite, un ensemble de leaders est initialisé avec les individus non-dominés de la population. Par la suite, un certain critère de performance est calculé pour les leaders. Ainsi, à chaque génération et pour chaque individu, un leader est sélectionné et la mise à jour de la position est effectuée suivie d'une opération de mutation. Ensuite, l'individu est évalué et son P_{best} est mis à jour. Généralement, un nouvel individu remplace son P_{best} si celui-ci est dominé ou s'ils sont incomparables. Une fois que tous les individus sont mis à jour, l'ensemble des leaders est aussi mis à jour. Finalement, l'ensemble des leaders est à nouveau évalué par un certain critère de performance. Ce processus continu jusqu'à atteindre un nombre maximum de générations.

2.6 Optimisation Multi-objectif par les ACO

Plusieurs approches d'ACO ont été proposées pour une large variété de problèmes d'optimisation à objectifs multiples [110-114]. La différence entre ces approches peut être résumée dans les trois points suivants :

- Structure des phéromones.
- Choix des meilleures solutions.
- Définition des variables heuristiques.

L'algorithme générique d'un MOACO est illustré dans la figure (2.8) [115], il est caractérisé par le nombre de colonies de fourmis N_{col} , le nombre des structures de phéromone considérées N_τ et une information heuristique notée η^i définie pour chaque fonction objective $f_i \in F$. De plus, on sait que les solutions sont représentées sous forme d'un graphe composé d'un ensemble de sommets (V) et d'un ensemble d'arcs (E) reliant les sommets noté $G = (V, E)$.

```
Algorithme MOACO (N_col , N_τ ):
  Initialiser les pheromones à τ_max
  repeat
     Pour chaque colonie c = 1: N_col
        Pour chaque fourmi k = 1: N_fourmi
           construire une solution
     Pour i = 1: N_τ
        mettre à jour la i^ème structure de phéromone
        Si un phéromone < τ_min Alors le mettre à τ_min
        Si un phéromone > τ_max Alors le mettre à τ_max
  until Nombre max de cycles atteint
```

Fig. 2.8 : Algorithme générique du MOACO

Initialement, toutes les variables phéromone sont initialisées à une valeur limite τ_{max}. Ensuite, à chaque cycle, chaque fourmi construit une solution et les phéromones sont mis à jour. Afin d'éviter une convergence prématurée, les phéromones sont limités par τ_{min} et τ_{max} Tel que $0 < \tau_{min} < \tau_{max}$. Enfin, l'algorithme s'arrête lorsqu'un nombre maximum de cycle soit atteint.

La (Fig.2.9) montre l'algorithme utilisé par les fourmis pour construire une solution S décrite par un graphe $G = (V, E)$. A chaque itération, un sommet v_i de G est choisi avec une probabilité $p_S(v_i)$ parmi un ensemble de sommet candidat $Cand$; qui sera ajouter par la suite à la solution S. La probabilité $p_S(v_i)$ est définie par :

$$p_S(v_i) = \frac{\left(\tau_S(v_i)\right)^{\alpha}\left(\eta_S(v_i)\right)^{\beta}}{\sum_{v_j \in Cand}\left(\tau_S(v_j)\right)^{\alpha}\left(\eta_S(v_j)\right)^{\beta}} \quad (2.12)$$

Ou $\tau_S(v_i) et\ \eta_S(v_i)$ représentent respectivement les facteurs phéromone et heuristique d'un sommet v_i.

S ←∅
Cand ← V
While Cand ≠ ∅ do
  Choisir $v_i \in$ Cand avec une probabilité $p_S(v_i)$
  Ajouter v_i à la solution S
  Enlever de Cand les sommets violant les contraintes
End while

Fig. 2.9 : Algorithme de construction d'une solution S

La mise à jour des phéromones est réalisée de la même manière que celle dans l'algorithme ACO présenté dans le chapitre précédent.

2.7 Optimisation Multi-objectif par les DE

Les algorithmes DE ont été initialement développés pour traiter les problèmes d'optimisation à objectif unique. Cependant, leur simplicité d'implémentation et leur efficacité les ont rendus des candidats pour l'optimisation multi-objectif. En effet, le premier algorithme DE traitant le problème multi-objectif a été proposé dans [116]. Dans cette approche appelé PDE, l'algorithme DE est employé pour créer de nouvelles solutions et seulement les solutions non-dominées sont considérées pour la création de la prochaine génération. Par la suite, d'autres algorithmes DE multi-objectif ont été développés, à savoir PDEA, MODE, DEMO, ADEA et MOMoDE [117-119].

Le fonctionnement de base d'un algorithme MODE [119] consiste à introduire une approche basée le principe de dominance de Pareto pour sélectionner les meilleurs individus. D'abord, une population de taille NP est générée aléatoirement et les fonctions d'adaptation sont évaluées pour chaque individu. Durant le processus de recherche, la population d'une génération donnée est triée en plusieurs rangs selon le concept de dominance. Ensuite, les opérations DE sont appliqués sur l'ensemble d'individus de la population, et les fonctions d'adaptation sont ainsi évaluées pour les différents individus obtenus.

La différence majeure entre les DE et les MODE réside dans le fait que les individus obtenus et les individus parents correspondants sont combinés pour former une population

globale de taille 2NP. Cette population sera ensuite triée suivi d'un calcul de la distance d'encombrement (crowding distance). La sélection des NP meilleurs individus est faite selon leur rang et la distance d'encombrement. Ces individus seront considérés en tant que parents de la prochaine génération.

Cette procédure continue jusqu'à ce que tous les meilleurs NP individus choisis aient un rang égal à un. Le Pseudo-code de l'algorithme MODE est présenté dans la figure (2.10).

1) Evaluer Les NP individus aléatoires de la population initiale P.
While (le critère d'arrêt n'est pas atteint) **do**
  2) Classer les individus dans des fronts selon la non-dominance.
  3) calculer la distance d'encombrement des individus de chaque front.
  4) appliquer les opérations DE sur les individus de la population triée (P') pour créer de nouveaux individus de la population (P'').
  5) effectuer un Elitisme :
    - former une population(PT) de taille $2NP$ contenant les individus parents et les nouveaux individus crées.
    - trier la population (PT) selon la non-dominance (PT').
    - choisir les meilleurs NP individus de (PT') et les mettre dans (P''').
End While.

Fig. 2.10 : Pseudo-code de l'algorithme MODE

2.8 Conclusion

Du à leur parallélisme inhérent dans l'exploration de l'espace de recherche et à leur aptitude d'agir sur une population de solutions, les algorithmes évolutionnaires (AE) sont capables d'obtenir un ensemble de points optimal de Pareto et sont donc très appropriés pour l'optimisation à objectifs multiples.

Dans ce chapitre, nous avons présenté différentes approches évolutionnaires utilisées pour l'optimisation multicritère. Souvent ces approches sont des extensions des AE utilisés pour les problèmes d'optimisation à objectif unique. Un nombre assez importants d'AE multi-objectif a été proposé en littérature [91].

Chapitre III :

Optimisation d'un SIF par l'algorithme MBR-NSGAII

3.1 Introduction

Le compromis omniprésent qui existe entre la précision du système et sa complexité est particulièrement important dans les SIF [1-3]. La complexité, exprimée en termes d'interprétabilité du système flou, est déterminée d'une part par la complétude mesurée par le nombre de règles floues dans la base de règles et par le nombre des variables d'entrée impliquées dans chaque règle, et d'autre part par la distinction des ensembles flous.

De nombreuses approches ont été suggérées pour traiter ce problème afin d'augmenter l'interprétabilité des systèmes flous. Ces approches sont principalement basées sur une structure appropriée du système flou (par exemple. une structure hybride) [4-10] ou sur l'utilisation d'un algorithme d'optimisation spécifique (par exemple. méthodes dans le domaine de l'optimisation multi-objectif ou de l'optimisation évolutionnaire)[11-17].

Dans ce chapitre, un contrôleur Neuro-flou (CNF) est adopté pour implémenter un SIF du type Mamdani [48]. Le but de notre méthode est non seulement d'assurer la précision appropriée du contrôleur, mais d'assurer également l'interpretabilité des règles floues. Ainsi, le problème d'optimisation est posé comme un problème multi-objectif dont la solution est obtenue par un nouvel algorithme évolutionnaire développé dit MBR-NSGAII (Mixed Binary-Real Non dominated Sorting Genetic Algorithm II).

3.2 Structure du Contrôleur Neuro-Flou

La structure du contrôleur Neuro-flou qui implémente un SIF du type Mamdani est illustrée dans la figure (3.1). Cependant, pour la simplicité de la présentation, on va développer un CNF ayant un nombre d'entrée quelconque et uniquement une seule sortie. La généralisation pour le cas multi-variable est directe. La structure du CNF consiste en cinq couches : une couche d'entrée, une couche de fuzzification, une couche de neurone AND (ET), une couche de neurones OR(OU) et une couche de défuzzification.

- *Couche d'entrée (input layer)*: les nœuds de cette couche transmettent seulement les valeurs d'entrée à la prochaine couche, ils n'effectuent aucune opération:

$$v_i = x_i \tag{3.1}$$

  Les poids des connexions au niveau de cette couche sont égaux à un.

- *Couche de fuzzification (fuzzification layer):* les nœuds de cette couche calculent les valeurs des fonctions d'appartenance des entrées v_i. tous les nœuds qui sont connectés au même nœud d'entrée possèdent le même poids Li correspondant à la largeur de la partie centrale de l'univers de discours des variables d'entrée.

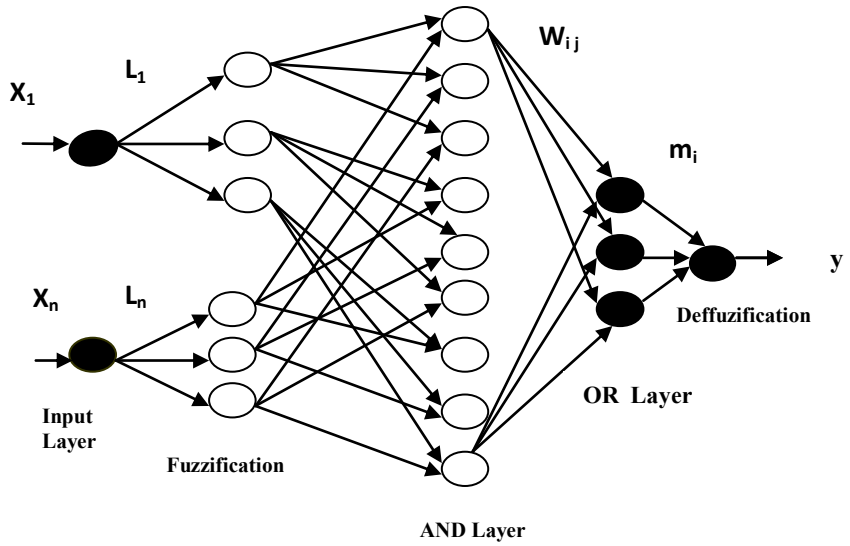


Fig. 3.1 : Architecture du Contrôleur Neuro-flou

Afin de garantir la complétude et la ditinguabilité des partitions floues, des fonctions d'appartenance triangulaires symétriques sont considérées (Fig. 3.2).

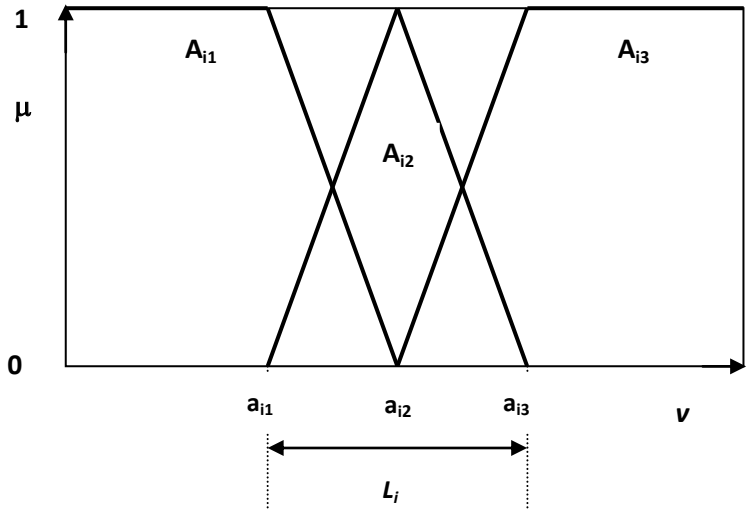


Fig. 3.2 : Partition symétrique triangulaire de l'univers de discours (avec 3 sous-ensembles flous)

La sortie du noeud (i, j) est donnée par:

$$\mu_{A_{ij}}(v_i) = \begin{cases} v_i(N_i - 1/L_i) + (N_i - 1/2) - j + 2\,, \\ \qquad \text{if } a_{ij-1} < v_i < a_{ij} \\ -v_i(N_i - 1/L_i) - (N_i - 1/2) + j, \\ \qquad \text{if } a_{ij} < v_i < a_{ij+1} \\ 1 \quad , \qquad v_i < a_{i1} \ \text{ ou } \ v_i > a_{iNi} \end{cases} \tag{3.2}$$

Avec les sous ensembles flous A_{ij} , $i = 1, \ldots, n$; $j = 1, \ldots, N_i$, est le nombre d'ensembles flous associés à la variable i, Les sommets des ensembles flous sont donnés par :

$$a_{ij} = (-(1/2) + ((j-1)/(N_i - 1)))L_i \,, \quad avec\ i = 1, \ldots, n\ et\ j = 1, \ldots, N_i \tag{3.3}$$

- *Couche AND (AND Layer)*: Les nœuds de cette couche sont dits les nœuds de règles, ils réalisent l'opération logique AND en utilisant le produit algébrique comme suit :

$$y_k^{And} = \mu_{A_{1j1}}(v_1).\mu_{A_{2j2}}(v_2) \ldots \mu_{A_{njn}}(v_n), \quad j_i = 1 \ldots N_i, i = 1 \ldots n, k = 1 \ldots \prod_{i=1}^{n} N_i \tag{3.4}$$

Les poids des neurones de cette couche sont fixés à un et ne sont pas ajustables.

- *Couche OR (OR Layer)* : Au niveau de cette couche, les règles ayant les mêmes conséquences sont intégrées en utilisant l'opérateur flou OR qui est réalisé par :

$$y_l^{OR} = 1 - \prod_{k=1}^{Na}(1 - y_k^{AND} W_{kl}),\ l = 1, \ldots, N_o \tag{3.5}$$

Avec :

W_{kl} Les poids associés à la connexion entre le nœud k de la couche AND et le nœud l de la couche OR,

N_a Le nombre de neurones de la couche AND,

et N_o Le nombre d'ensembles flous associés à la variable de sortie.

Et puisqu'une règle ne peut avoir qu'une seule conséquence, alors W_{kl} doit être binaire:

$$W_{kl} \in \{0,1\}\ \forall\ k, l \tag{3.6}$$

- *Couche de deffuzification (deffuzifacation layer)*: Le nœud de cette couche réalise l'opération de défuzzification selon la règle du centre de gravité

$$u = \sum_{i=1}^{No} m_i y_i^{OR} / \sum_{i=1}^{No} y_i^{OR} \tag{3.7}$$

Ou m_i sont des poids à valeurs réels qui correspondent aux centres des ensembles flous triangulaires de la variable de sortie. Ces poids peuvent être donnés par :

$$m_i = \left(-\frac{1}{2} + \frac{i-1}{Ni-1}\right) Lo, \quad i = 1, \ldots, No \tag{3.8}$$

Avec Lo est la largeur de la partie centrale de l'univers de discours de la variable de sortie

3.3 Optimisation par l'algorithme multi-objectif

3.3.1 Définition du problème

L'idée de base de notre approche consiste à assurer une optimisation structurelle et paramétrique du CNF qui implémente un SIF du type Mamdani. Pour l'optimisation

paramétrique il s'agit, d'une part, de trouver les valeurs optimales des largeurs des parties centrales des univers de discours des variables d'E/S (les poids L_i ,$i = 1, \dots, n$ et L_o) permettant ainsi d'altérer les fonctions d'appartenance associées. D'autre part, il s'agit d'ajuster les poids binaires (W_{kl}) connectant la couche AND et la couche OR, cet ajustement permet de modifier les règles floues.

En ce qui concerne l'optimisation structurelle, il s'agit de minimiser les règles floues ainsi obtenues, c'est-à-dire trouver un nombre réduit de règles floues.

Afin de préserver l'interprétabilité (complétude, distinguabilité, consistance, et continuité) des règles floues tout le long du processus d'apprentissage, on a établi un jeu de contraintes comme suit :

- **Complétude** : La complétude des systèmes flous consiste en la complétude des partitions floues et de la complétude de la structure des règles floues. La complétude des partitions floues des variables d'entrée consiste à ce que tout l'espace d'entrée soit couvert.

Supposons que la variable d'entrée x_i est partitionnée en N_i sous-ensembles flous représentés par $A_{i1}(xi), A_{i2}(xi), \dots, A_{iNi}(xi)$ dans l'univers de discours U, le partitionnement est considéré complet si la condition suivante est vérifiée :

$$\forall x_i \in U, \exists\ 1 \leq j \leq N_i : \mu_{A_{ij}}(x_i) > 0 \tag{3.9}$$

Puisque dans notre méthode on a choisi des fonctions d'appartenance de type triangulaire symétrique, cette contrainte est vérifiée et reste même vérifiée après optimisation parce que l'optimisation des fonctions d'appartenance consiste en l'ajustement du paramètre L_i qui permet toujours d'avoir des fonctions d'appartenance triangulaires symétriques.

Pour la complétude de la structure des règles floues, il faut au moins qu'une règle soit active, cette condition est formulée par la contrainte suivante :

$$\sum_{l=1}^{N_o} {y_l}^{Or} \neq 0 \tag{3.10}$$

- **Distinguabilité** : Pour ajuster les fonctions d'appartenance, on a considéré la variation du paramètre L_i donc on ne risque pas d'avoir des sous-ensembles flous similaires.
- **Consistance** : Un ensemble de règles floues est inconsistant s'il existe au moins deux règles de prémisses identiques mais de conclusions différentes. Afin d'assurer la consistance des règles floues, on a imposé les contraintes suivantes :

$$\sum_{l=1}^{N_o} W_{kl} \leq 1, \forall\ k = 1, \dots, N_a \tag{3.11}$$

Cela signifie que pour des prémisses données d'une règle, on ne peut avoir, au maximum, qu'une seule et unique conséquence.

Par conséquent, les degrés de liberté d'un tel problème sont : le nombre d'ensemble flous pour chaque variable N_q, les paramètres binaires W_{ik} définissant les règles et les paramètres réels Li et Lo des fonctions d'appartenance triangulaires des variables d'entrées et de sortie respectivement.

Ainsi, le problème consiste à trouver tous ces paramètres afin d'assurer un certain degré de précision avec une base de règles compacte. Ceci est formulé en tant qu'un problème d'optimisation à deux objectifs :

Trouver $N_q, q = 1 \dots n,\ M,\ Li,\ Lo$ et W_{ik} tel que :

$$Min\left((J_1 = f(e(t)),\ \left(J_2 = \sum_{i=1}^{NR}\sum_{k=1}^{M} W_{ik} \right)\right) \qquad (3.12)$$

Avec $e(t)$ est l'erreur, et $f()$ est une fonction de l'erreur.

3.3.2 L'algorithme multiobjectif MBR-NSGAII

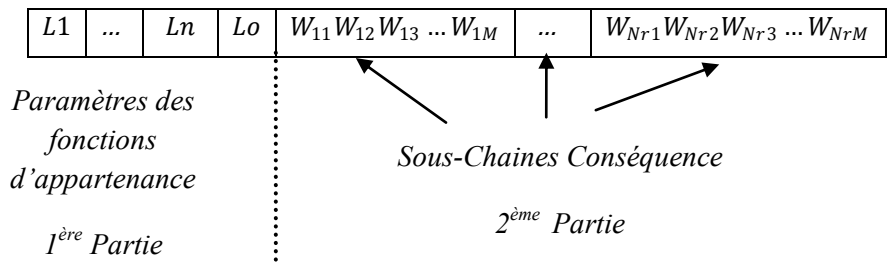
Le problème considéré étant à objectifs multiples, on a développé une méthode évolutionnaire très appropriée pour traiter ce genre de problèmes dite MBR-NSGAII [138]. Afin de satisfaire aux contraintes imposées, on a introduit une légère perturbation au niveau de ses opérateurs génétiques. Par la suite on va décrire la procédure utilisée pour résoudre ce problème.

3.3.2.1 Représentation des individus de la population

Chaque individu de la population est composé de deux parties :

La première contient les paramètres réels des fonctions d'appartenance associées aux variables d'entrée et de sortie.

La deuxième contient les poids binaires W_{ik} qui modélisent la conséquence de la règle.



Avec la sous-chaine : $/\ W_{i1}W_{i2}W_{i3} \dots W_{iM}\ /, W_{ik} \in \{0, 1\}$ defini la conséquence de la règle i et sera appelée dans ce qui suit sous-chaine conséquence.

Ainsi, la contrainte (3.11) implique qu'un seul poids binaire dans une sous-chaine conséquence donnée sera égal à un, cependant si tous les poids binaires sont nuls alors la règle associée n'aura pas de conséquence et ne sera pas incluse dans la base de règles.

3.3.2.2 Croisement mixte binaire-réel

Un croisement à deux points est utilisé: Le premier point tombe dans la première partie de l'individu (croisement réel) et le deuxième point dans la deuxième partie (croisement binaire).

1. croisement réel: les paramètres de la première partie de l'individu utilisent un codage réel et un croisement intermédiaire étendu [120] est utilisé, ainsi deux nouveaux individus ($O1\ and\ O2$) sont obtenus à partir des deux parents P_1 et P_2 comme suit :

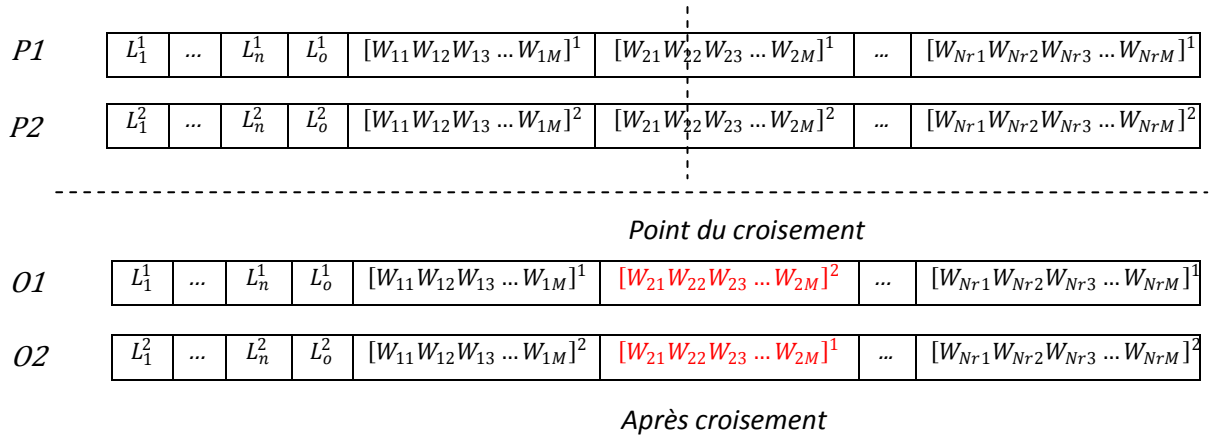
$$O_1 = P_1 + \alpha_1(P_2 - P_1) \tag{3.13}$$

$$O_2 = P_2 + \alpha_2(P_1 - P_2) \tag{3.14}$$

Avec α_i est une valeur choisie aléatoirement dans l'intervalle $[-0.25, 1.25]$.

Ce croisement est effectué pour chaque paramètre de la première partie de l'individu.

2. Croisement binaire: Afin de respecter la contrainte (3.11), le croisement dans la deuxième partie de l'individu est modifié comme suit :



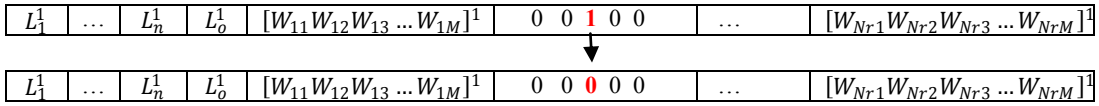
Cela signifie qu'il y a échange des sous-chaines conséquence correspondant au point du croisement.

3.3.2.3 Mutation mixte binaire-réel

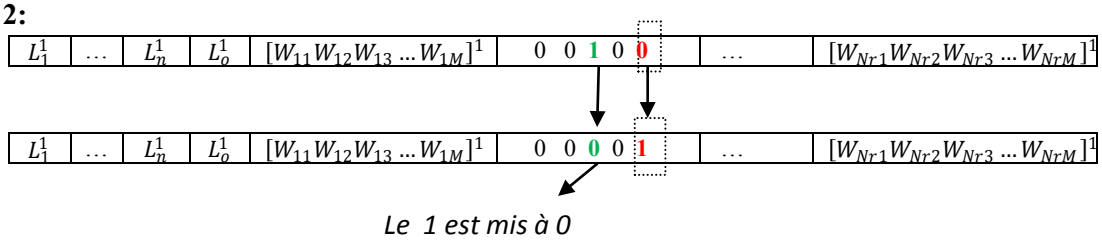
Une mutation non uniforme [120] est appliquée sur la première partie de l'individu. Si la mutation tombe dans la deuxième partie, le 1 est mis à 0 et le 0 est mis à 1. Cependant, dans ce dernier cas, la contrainte (3.11) peut être violée et on risque d'avoir deux 1 dans la même sous chaine conséquence, ainsi la règle associée aura deux conséquences.

Alors pour satisfaire (3.11), et dans le cas ou il y a deux 1 dans la sous-chaine conséquence, le bit qui était égal à 1 avant mutation est mis à zéro.

Cas 1:



Cas 2:



3.4 Application au contrôle des systèmes

Deux exemples d'application ont été considérés : conception d'un CNF pour le système du pendule inversé et un CNF multi-variable décentralisé d'un modèle de simulateur d'hélicoptère.

3.4.1 Contrôle du pendule inversé

3.4.1.1 Système du pendule inversé

La structure du système est composée d'un segment rigide et un chariot sur lequel le segment est placé [139]. Le chariot se déplace à gauche ou à droite suivant la force exercée sur lui. Le segment est lié au chariot via une articulation libre avec frottement offrant uniquement un seul degré de liberté.

Les dynamiques du système du pendule inversé sont caractérisées par deux variables d'état :

θ : L'angle du segment par rapport à l'axe vertical.

$\dot{\theta}$: La vitesse angulaire du segment.

Le comportement de ces deux variables d'état est gouverné par l'équation différentielle du second ordre suivante :

$$\ddot{\theta} = \frac{gsin\theta + cos\theta\left(\frac{-F - ml\dot{\theta}^2 sin\theta}{m_c + m}\right)}{l\left(\frac{4}{3} - \frac{mcos^2\theta}{m_c + m}\right)} \tag{3.15}$$

Où : g, l'accélération gravitationnelle = 9.8 m/s^2.

m_c, masse du chariot = 1 Kg

m, masse du segment = 0.1 Kg

l, demi-longueur du segment = 0.5 m

F, la force appliquée en Newton (N).

Ce système est simulé par la méthode de Range-Kutta d'ordre 4 avec une période d'échantillonnage égale à 10ms.

3.4.1.2 Conception du contrôleur Neuro-flou

L'objectif de contrôle est de stabiliser le pendule en exerçant une force sur la base du chariot. Ce système présente quelques propriétés intéressantes pour tester notre approche. En effet, il est fortement non linéaire lorsqu'il est loin de la verticale et il est sensible aux variations des paramètres tels que les conditions initiales la longueur du segment et la masse.

Le contrôleur Neuro-flou possède deux entrées : l'erreur e(t) et la variation de l'erreur Δe(t), et une sortie : la force F qui va être appliquée sur le chariot pour stabiliser le pendule dans sa position d'équilibre (verticale) qui correspond à un angle et vitesse angulaire nuls. Le nombre des fonctions d'appartenance est fixé à trois (3) (NE : Négative, ZE : Zéro et PO : Positive) pour chacune des variables d'entrée/sortie.

Initialement, on a réseau contenant six neurones dans la couche de fuzzification, neuf nœuds dans la couche AND et trois nœuds dans la couche OR (Fig.3.3)

L'algorithme MBR- NSGA II a pour but de trouver des valeurs optimales pour $Li\ (L1, L2), Lo$ et W_{ij} tel que:

$$Min\left((J_1 = f(e(k)) , \left(J_2 = \sum_{i=1}^{9}\sum_{j=1}^{3} W_{ij} \right) \right) \tag{3.16}$$

$$\text{avec : } f = \sum_{k=1}^{500} |e(k)| + |\Delta e(k)| + \lambda |F(k-1)|$$

Puisque la verticale correspond à un angle et vitesse angulaire nuls alors on a :

$$|e(k)| = |\theta(k)| \ et\ |\Delta e(k)| = |\Delta\theta(k)|$$

On a ajouté la force dans le critère J_1, afin de minimiser l'effort exercé sur le chariot, cet effort est pondéré par un coefficient λ fixé à une valeur de 0.01.

Le problème de minimisation des critères J_1 et J_2 est transformé en problème de maximisation des fonctions d'adaptation f_1 et f_2 définies par :

$$Max\left(f_1 = \frac{\alpha_1}{1+\beta_1 J_1}, \qquad f_2 = \frac{\alpha_2}{1+\beta_2 J_2} \right) \tag{3.17}$$

Avec $\alpha_1, \alpha_2, \beta_1, \beta_2$ sont des facteurs de mise à l'échelle choisis comme suit :

$$\alpha_1 = 10^5, \alpha_2 = 10, \beta_1 = 20\ et\ \beta_2 = 1$$

Il y a 30 paramètres impliqué dans le processus d'optimisation.

Le contrôleur flou est entraîné en simulation en boucle fermée en considérant les paramètres génétiques suivants :

Génération maximale=100,

Taille de la population =100,

Probabilité de croisement=0.8

Probabilité de mutation=0.03.

Les conditions initiales pour l'angle et la vitesse angulaire suivantes :

$$\theta(0) = 20^o \; et \; \dot{\theta}(0) = 0^o/s.$$

La figure (3.3) montre l'ensemble des solutions obtenues dans la dernière génération.

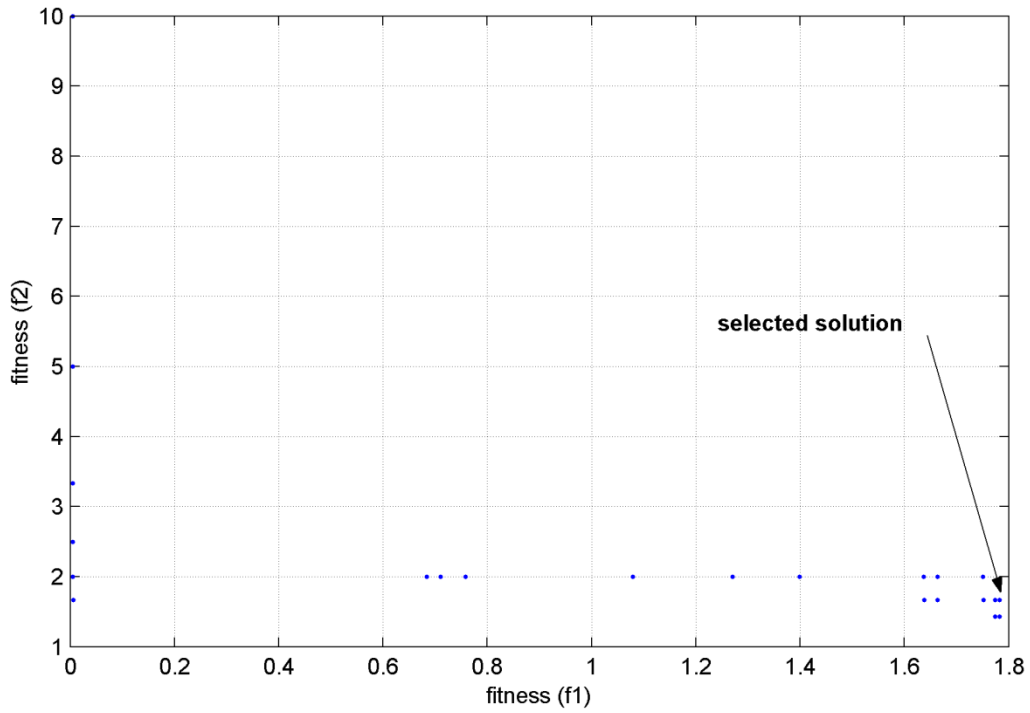


Fig. 3.3 : Distribution de la population finale

Les résultats du processus d'apprentissage obtenus par le MBR-NSGAII sont :

- Fonctions d'adaptation du meilleur individu du front :

$$f1 \; = \; 1.783$$

$$f2 \; = \; 1.667$$

- Valeurs optimales des paramètres réels L_i :

$$L_1 \; = \; L_\theta \; = \; 23.0666$$

$$L_2 \; = \; L_{\dot{\theta}} = 74.9882$$

$$L_O \; = \; L_F \; = \; 606.9333$$

- Valeurs optimales des paramètres binaires W:

$$W = \; [000\ 100\ 000\ 100\ 010\ 000\ 010\ 001\ 000]$$

Ceci correspond à un CNF ayant une structure réduite contenant cinq nœuds dans la couche de fuzzification, cinq nœuds dans la couche AND et trios nœuds dans la couche OR comme illustré dans la figure (3.4).

La structure du CNF obtenue est interprétée par un SIF du type Mamdani ayant cinq règles données dans le tableau (3.1).

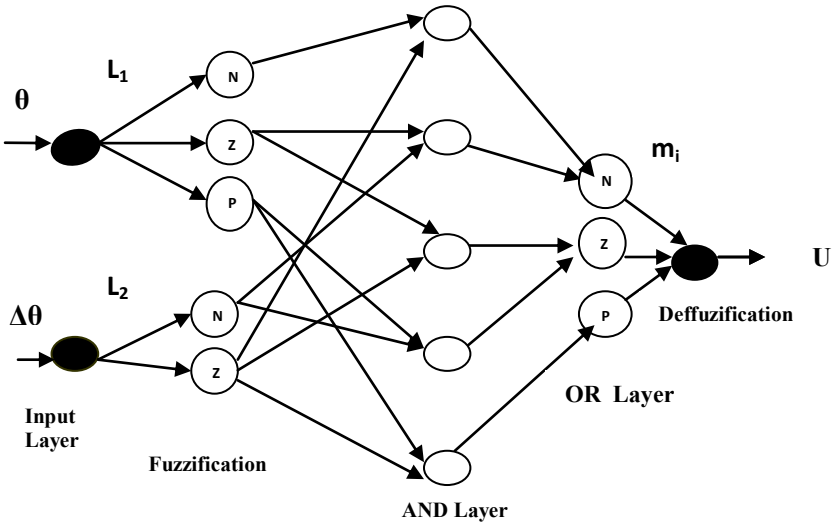


Fig. 3.4: Structure du CNF optimisé

θ(t) / Δθ (t)	NE	ZE	PO
NE		NE	ZE
ZE	NE	ZE	PO

Tableau 3.1: base de règles interprétée par le CNF

Les figures (3.5), (3.6) et (3.7) représentent respectivement les variations de l'angle, de la vitesse angulaire et de la force à partir du point initial (20°, 0°/sec). On constate que le contrôleur flou arrive à stabiliser le pendule sans oscillations pendant un temps inférieur à 1,5sec

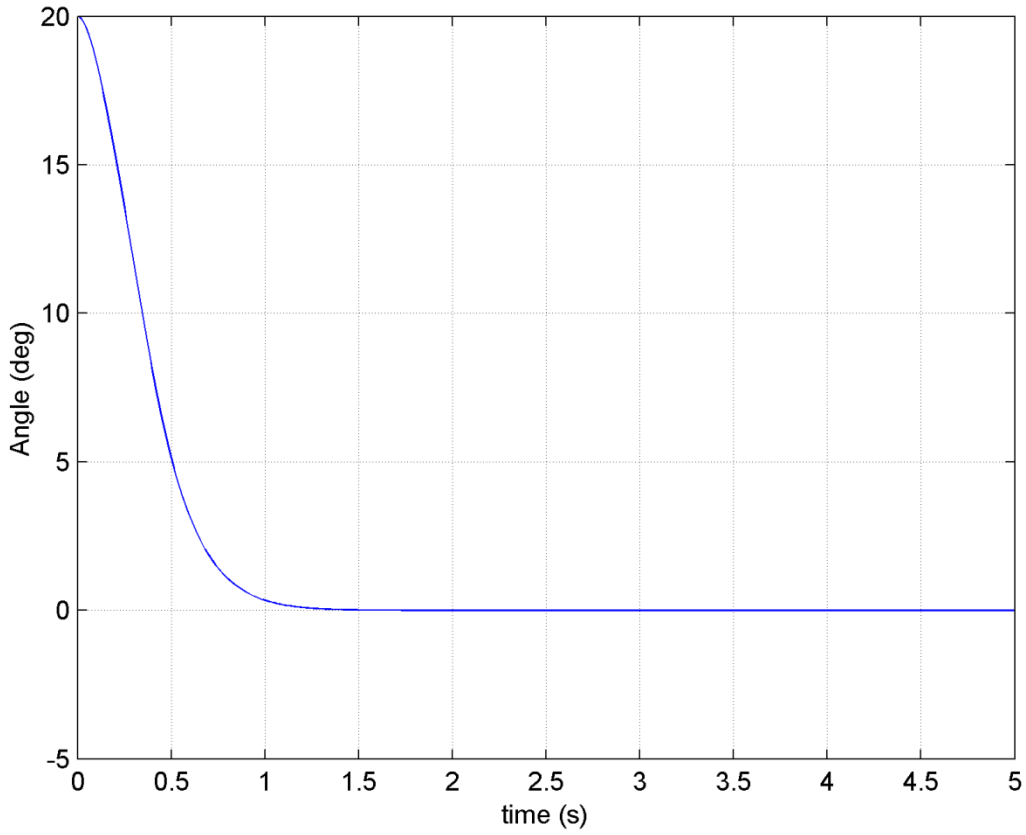


Fig. 3.5 : Variations de l'angle à partir du point (20°, 0°/s)

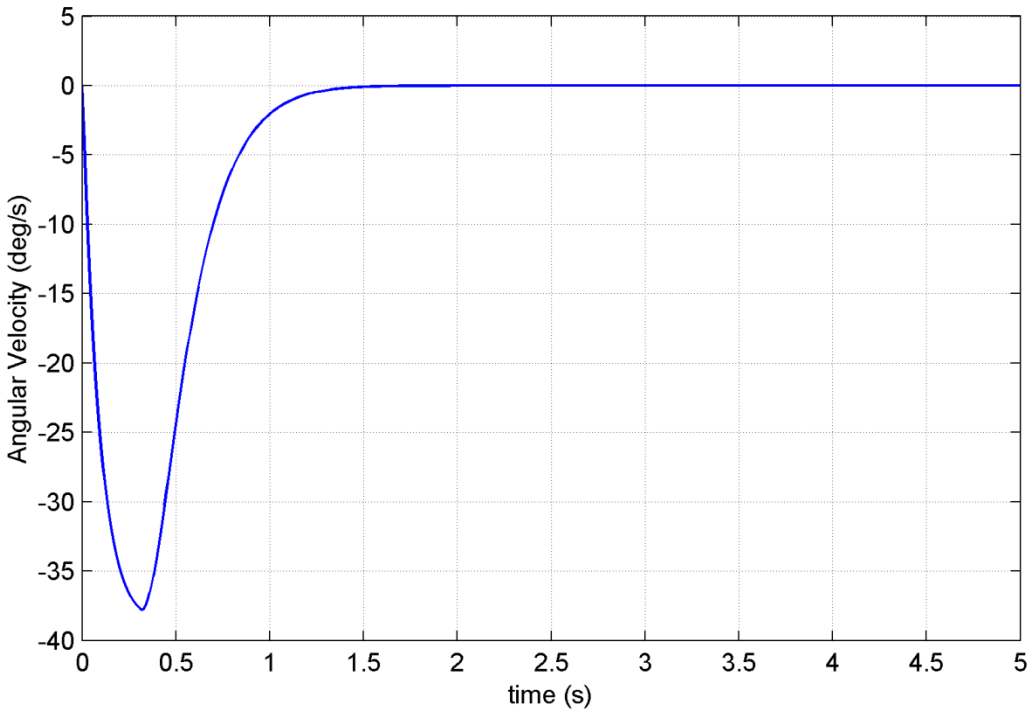


Fig.3.6 : Variations de la vitesse angulaire à partir du point (20°,0°/s)

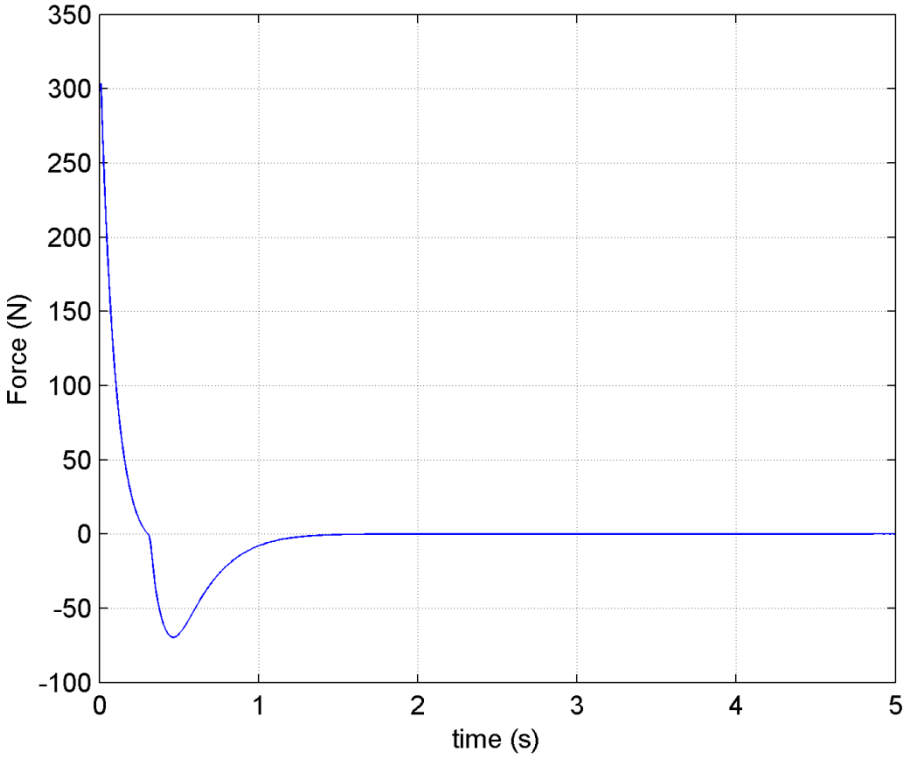


Fig. 3.7: Variations de la force

3.4.1.3 Robustesse du Contrôleur Neuro-Flou Optimisé

a) Variation des Conditions Initiales

Dans la phase d'apprentissage, l'entraînement du contrôleur a été fait en considérant les conditions initiales (20°, 0°/sec), cependant, il est intéressant de voir comment le contrôleur (obtenu à partir de ces conditions) va se comporter pour des conditions initiales différentes. Les figures (3.8, 3.9 et 3.10) illustrent les performances du contrôleur flou optimisé pour des conditions initiales différentes($10° < \theta(0) < 60°$), on constate que le contrôleur arrive toujours à stabiliser le pendule.

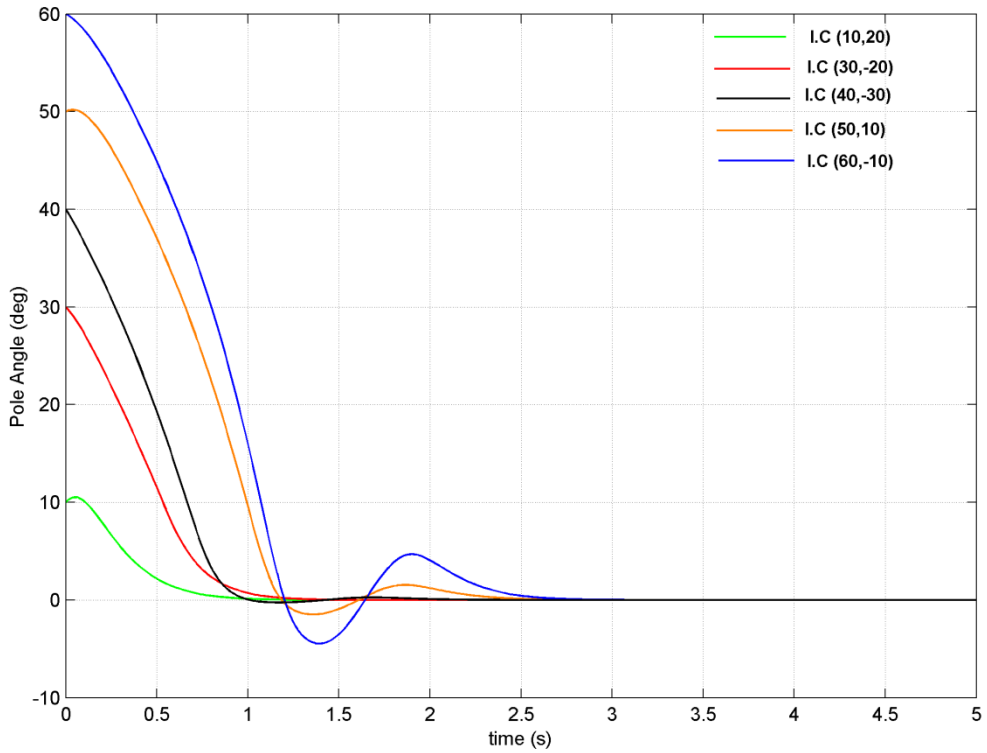


Fig.3.8: Variations de l'angle à partir des conditions initiales différentes

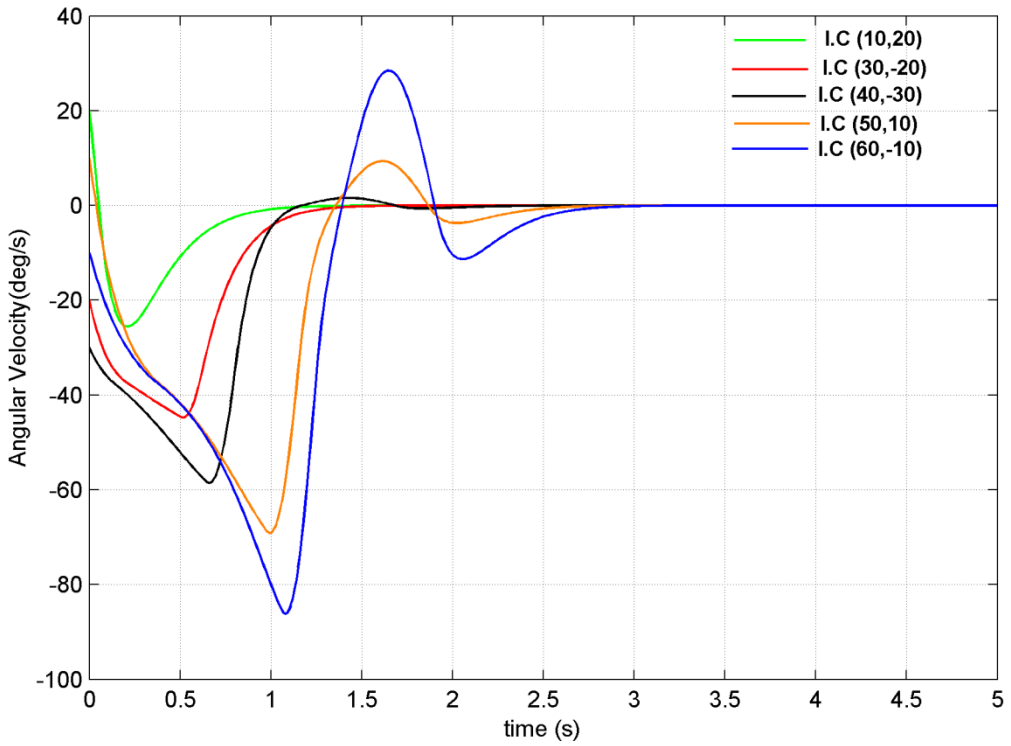


Fig.3.9: Variations de la vitesse angulaire pour des conditions initiales différentes

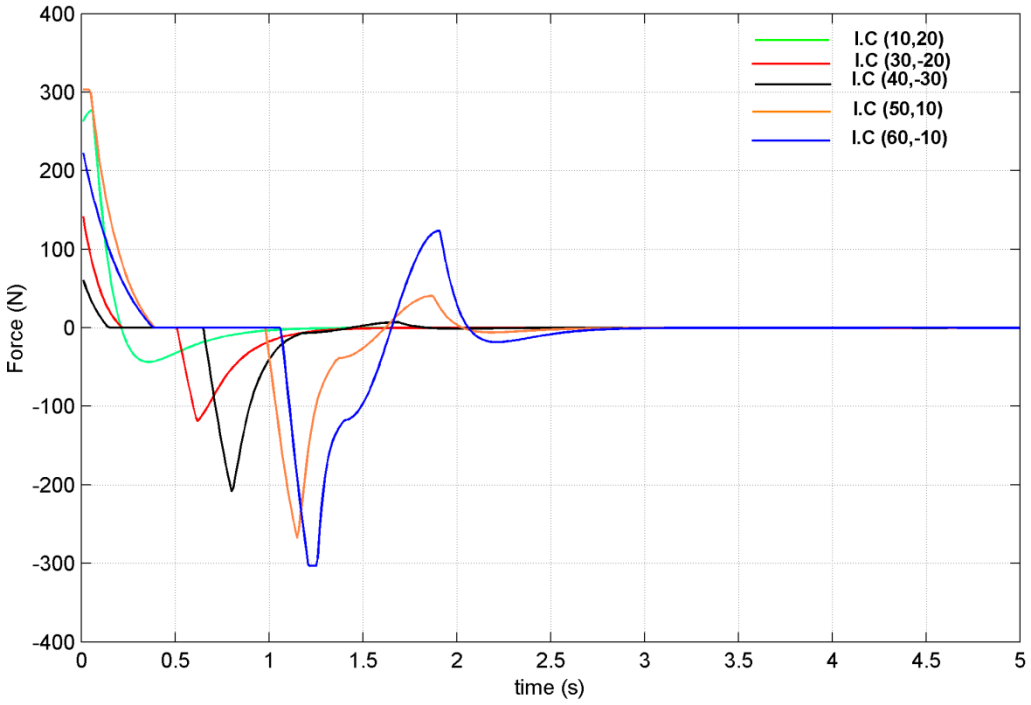


Fig. 3.10: Variations de la force pour des conditions initiales différentes

b) Variation de la Longueur du Pendule

Pour voir comment le contrôleur va réagir aux changements des paramètres du procédé, on l'a testé sur des pendules de longueurs différentes (0.25, 0.5, 1 et 1.5 m). Les résultats sont illustrés dans les figures (3.11, 3.12 et 3.13), ces résultats montrent la robustesse du contrôleur flou optimisé.

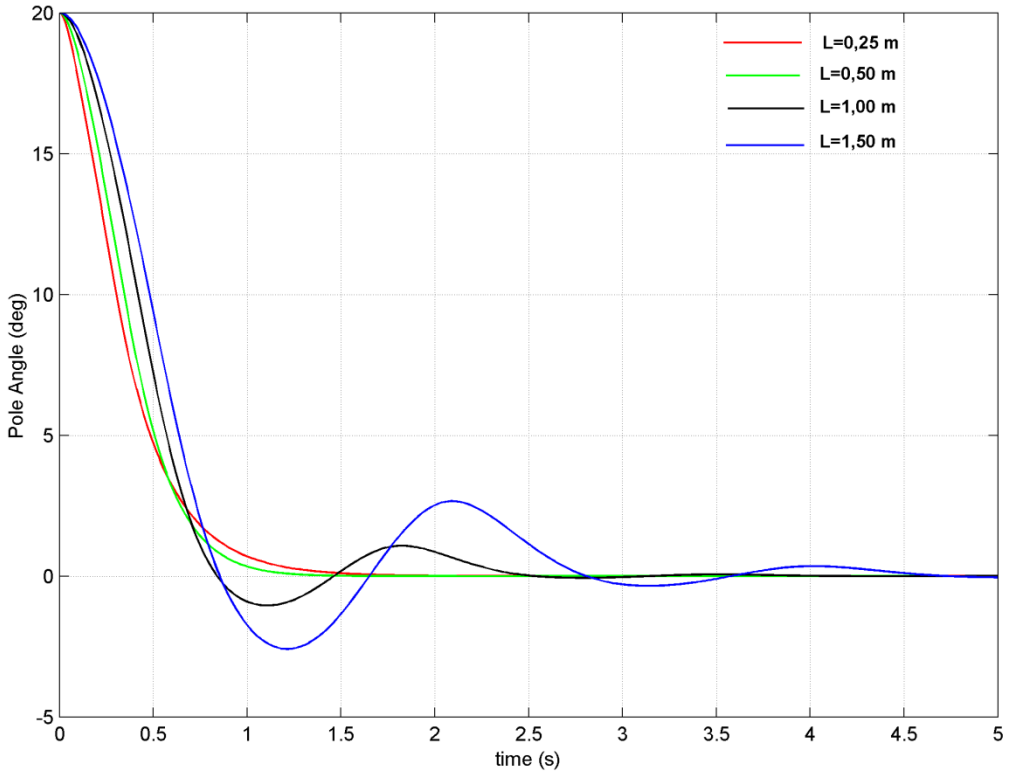


Fig. 3.11: Variations de l'angle pour des pendules de longueur variable

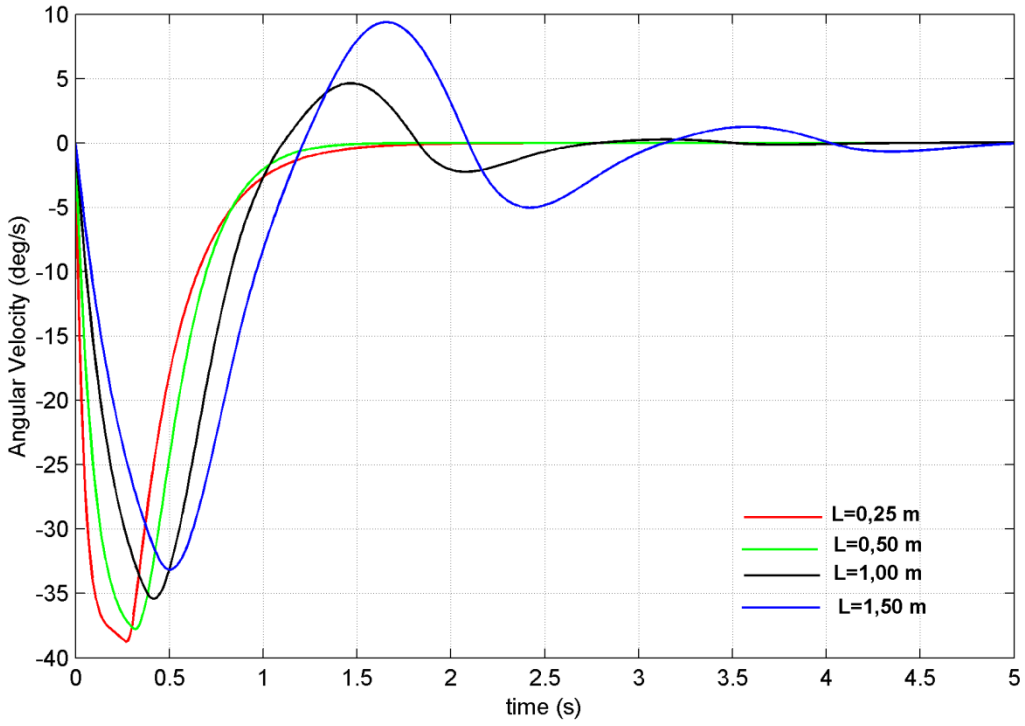


Fig. 3.12: Variations de la vitesse angulaire pour des pendules de longueur variable

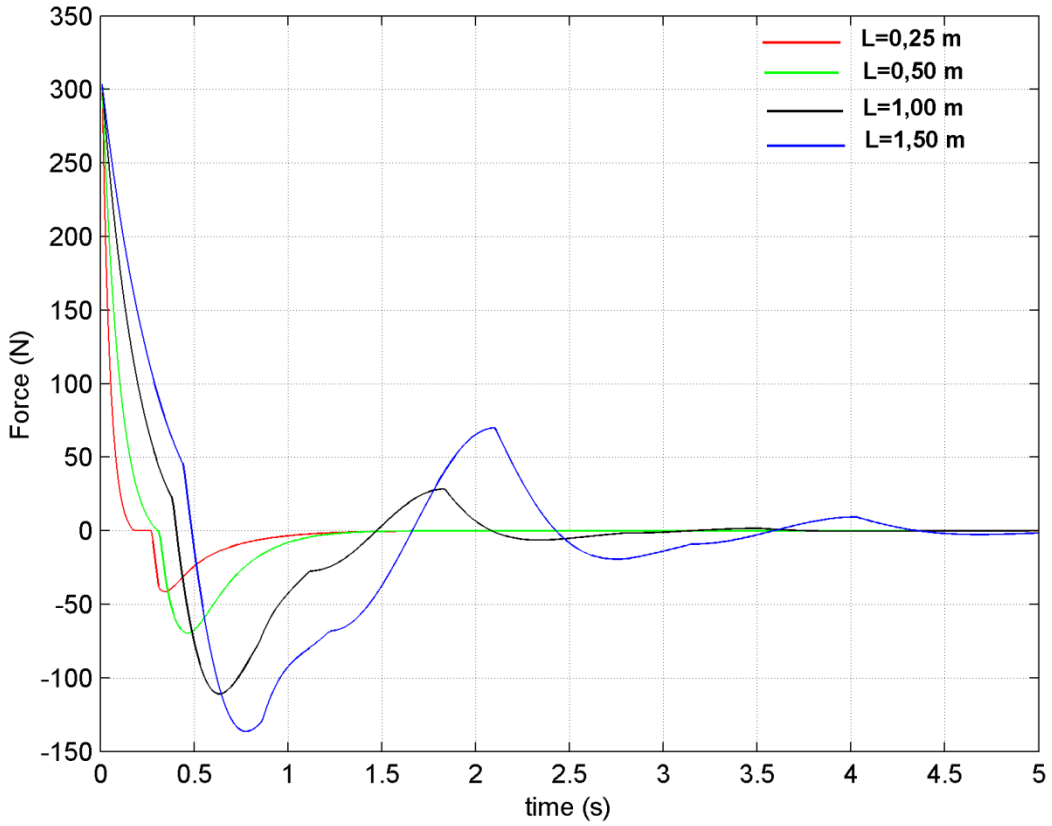


Fig. 3.13: Variations de la force pour des pendules de longueur variable

c) Variation de la de la masse du pendule

Dans ce test de robustesse, on a considéré des pendules de masses différentes (0.05, 0.1, 0.3, 0.5 et 0.7 kg) et on constate d'après les résultats illustrés dans les figures (3.14, 3.15 et 3.16) que le contrôleur se comporte bien face à ces changements.

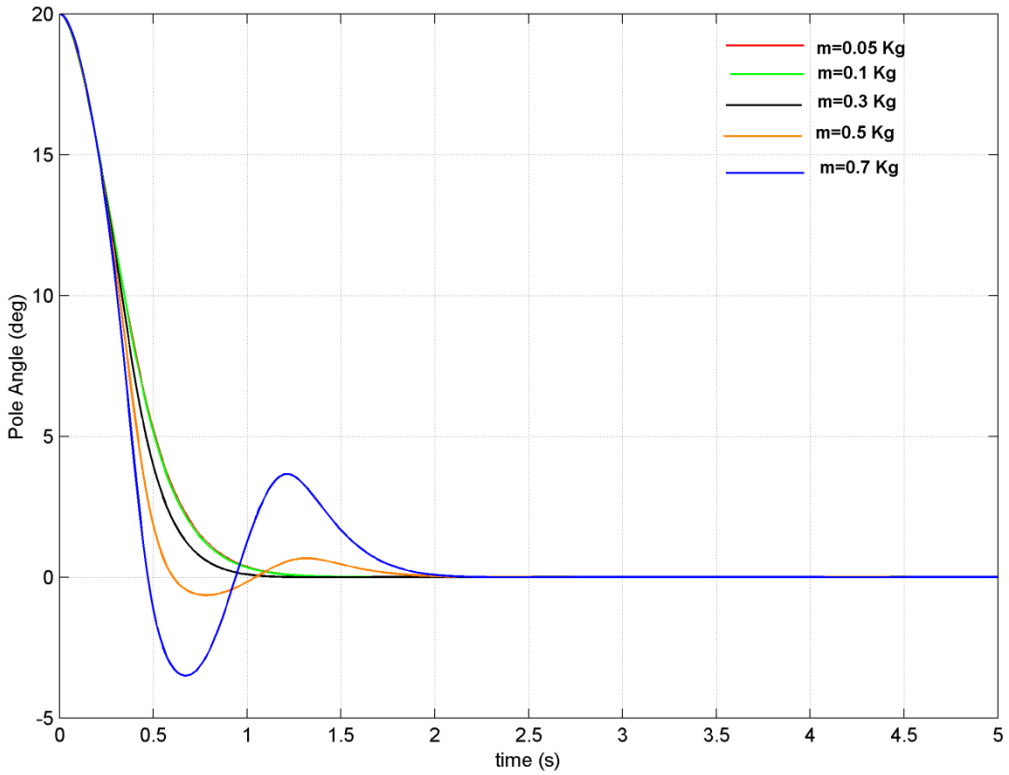


Fig.3.14: Variations de l'angle pour des pendules de masse variable

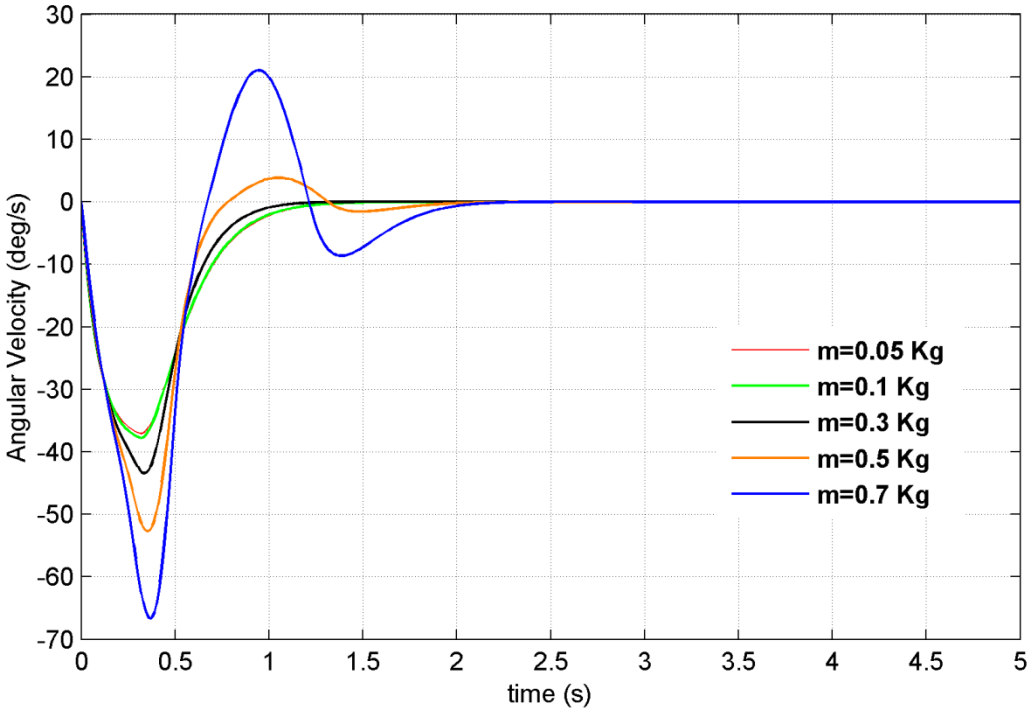


Fig. 3.15: Variations de la vitesse angulaire pour des pendules de masse variable

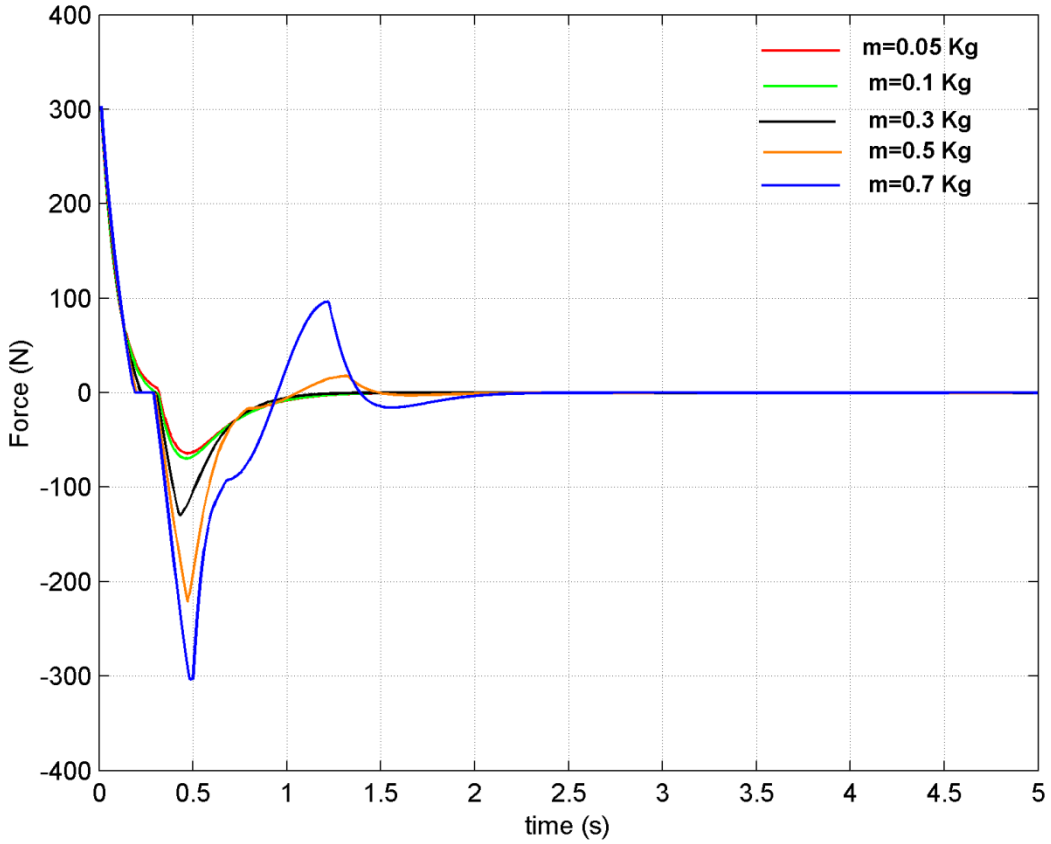


Fig.3.16: Variations de la force pour des pendules de masse variable

3.4.2 Contrôle d'un simulateur d'hélicoptère

3.4.2.1 Modèle du simulateur d'hélicoptère

Le simulateur d'hélicoptère à contrôler est le modèle CE 150 de Humusoft Ltd [121]. Il est composé d'un corps portant deux moteurs DC qui commandent les propulseurs (Fig. 3.17). Le corps de l'hélicoptère possède deux dégrées de liberté; l'angle d'élévation φ, i.e. l'angle entre l'axe vertical et l'axe longitudinal et l'angle d'azimut ψ, i.e. l'angle dans le plan horizontal entre l'axe longitudinal de l'hélicoptère et sa position zéro.

La tension conduisant le moteur principal $u1$ et la tension conduisant le moteur $u2$ de la queue affectent l'angle d'élévation et de l'azimut ; donc nous pouvons dire que les interactions mentionnées rendent le système multivariable. Le modèle d'hélicoptère peut être représenté comme un système (MIMO) multivariable non linéaire avec les entrées $u1$ et $u2$ et les deux sorties φ et le ψ.

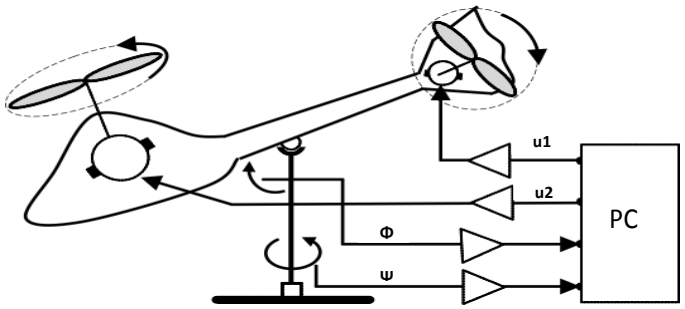


Figure 3.17 : Configuration de l'hélicoptère

Le modèle mathématique du simulateur est donné par les équations suivantes:

$\dot{x}_1 = x_2$

$\dot{x}_2 = -8{,}764x_2 sinx_1 - 3{,}4325x_4u_1 cosx_1 - 0{,}4211x_2 + 0{,}0035{x_5}^2 + 46{,}35{x_6}^2 +$
$0{,}8076x_5x_6 + \ 0{,}0259x_5 + 2{,}9749x_6$

$\dot{x}_3 = x_4$

$\dot{x}_4 = -2{,}1401x_4 + 31{,}8841{x_8}^2 + 14{,}2029x_8 + 21{,}7150x_9 - 1{,}4010u_1$

$\dot{x}_5 = -6{,}6667x_5 - 2{,}7778x_6 + 2u_1$

$\dot{x}_6 = 4x_5$

$\dot{x}_7 = -8x_7 - 4x_8 + 2u_2$

$\dot{x}_8 = 4x_7$

$\dot{x}_9 = -1{,}3333x_9 + 0{,}0625u_1$

Avec $x_1 = \psi, x_2 = \dot{\psi}, x_3 = \varphi, x_4 = \dot{\varphi}$

3.4.2.2 Conception des contrôleurs Neuro-flou

L'objectif de la commande est de stabiliser l'hélicoptère autour d'un point de référence (Ψ_r , Φ_r) , pour cela on adopté une structure de commande décentralisée par deux contrôleurs neuro- flous : un contrôleur pour le sous-système d'élévation et l'autre pour le sous-système d'azimut.

Chaque contrôleur possède deux entrées l'erreur e(k) et la variation de l'erreur Δe(k) et une sortie $u(k)$. Trois fonctions d'appartenance (Negative (NE), Zero (ZE) et Positive (PO)) sont considérées pour les entrées/sorties. L'algorithme MBR- NSGA II a pour rôle de trouver des valeurs optimales pour les largeurs réels des univers de discours des entrées/sorties des deux contrôleurs: $L_{e1}, L_{\Delta e1}, L_{u1}, L_{e2}, L_{\Delta e2}, L_{u2}$ et d'extraire une structure optimale qui correspond à un ensemble réduit de règles floues des deux contrôleurs décrites par les poids binaires ${w_{ij}}^1 and\ {w_{ij}}^2$ respectivement, ainsi 60 paramètres sont impliqués dans le problème d'optimisation qui utilise les valeurs suivantes:

Taille de la Population =100,

Génération maximale=100,

Probabilité de croisement=0.8,

Probabilité de mutation=0.01.

Le problème de minimisation est donné par:

$$\text{Min}\left(J_1 = \sum_{k=1}^{N} |e_1(k)| + |e_2(k)|,\ \ J_2 = \sum_{i=1}^{9}\sum_{j=1}^{3} W_{ij}{}^{1} \ + \sum_{i=1}^{9}\sum_{j=1}^{3} W_{ij}{}^{2} \right) \qquad (3.18)$$

Avec:

$$e_1(k) = \psi_r(k) - \psi(k) \ \text{ et } \ e_2(k) = \varphi_r(k) - \varphi(k)$$

Ce problème est transformé en un problème de maximisation:

$$\text{Max}\left(f_1 = \frac{\alpha_1}{1+\beta_1 J_1},\ f_2 = \frac{\alpha_2}{1+\beta_2 J_2} \right); \alpha_1 = 10^6, \alpha_2 = 10, \beta_1 = 20, \beta_2 = 1 \qquad (3.19)$$

Les tests de simulation sont réalisés pour les références $\psi_r = 1 \ and \ \varphi_r = 1$ pour un temps de simulation de 30 secondes. La population de la dernière génération est illustrée dans la figure (3.18) qui montre le front obtenu après 100 générations. Les solutions extraites sur le front sont comme suit :

- Les paramètres réels optimales:

$$L_{e1} = 9.0989,\ L_{\Delta e1} = 2.1647, L_{u1} = 5.9648,$$
$$L_{e2} = 7.4927,\ L_{\Delta e2} = 1.1852, L_{u2} = 3.3399$$

- Et les poids binaires optimaux:

$$W^1 = [001\ 100\ 001\ 010\ 010\ 010\ 001\ 100\ 010]$$
$$et\ W^2 = [000\ 000\ 010\ 100\ 010\ 000\ 100\ 001\ 001]$$

La structure des deux CNF obtenus interprètent des SIF de type Mamdani ayant les règles floues suivantes (tableau 3.2 et 3.3):

$\Delta e_1(t)$ \ $e_1(t)$	NE	ZE	PO
NE	PO	ZE	PO
ZE	NE	ZE	NE
PO	PO	ZE	ZE

Tableau 3.2 : Base des règles floues optimisées Du CNF1

$\Delta e_2(t)$ \ $e_2(t)$	NE	ZE	PO
NE		NE	NE
ZE		ZE	PO
PO	ZE		PO

Tableau 3.3 : Base des règles floues optimisées Du CNF2

Les Figures (3.19) et (3.20) montrent respectivement les variations de l'angle d'azimut et l'angle d'élévation, on constate que les deux sorties suivent parfaitement les références ($\psi_r = 1$ and $\varphi_r = 1$) et dans un temps inférieur à 5 secondes. L'angle d'élévation présente un dépassement de 10%. Les signaux de contrôle des deux CNF sont illustrés dans les figures (3.21) et (3.22).

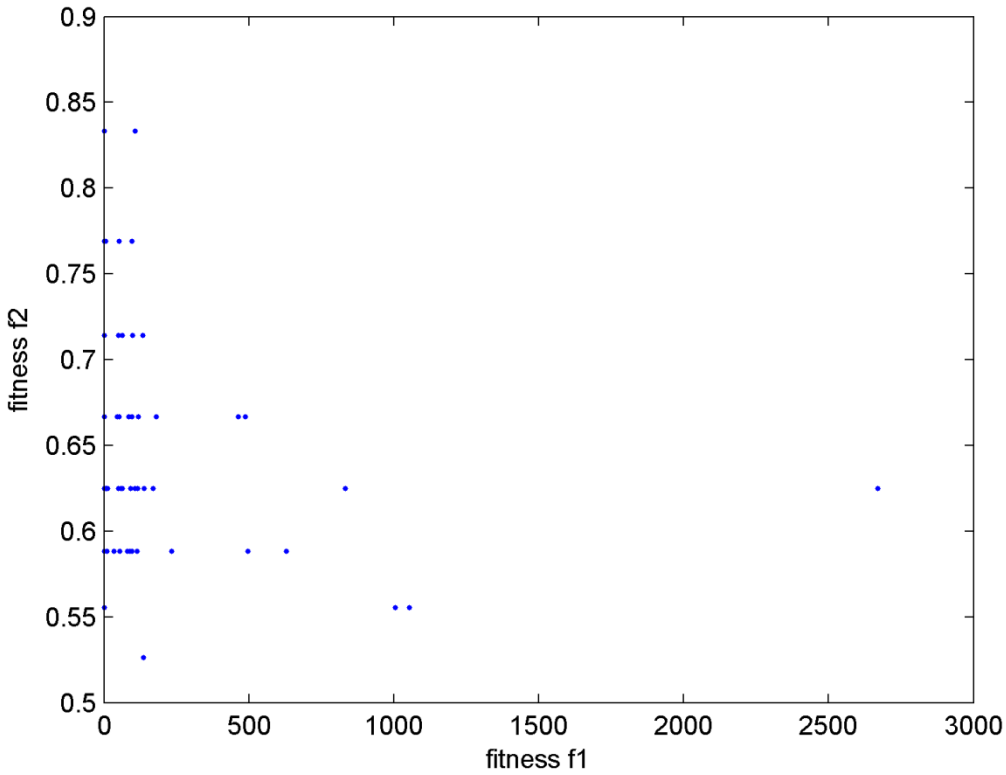


Fig. 3.18 : Distribution de la population finale

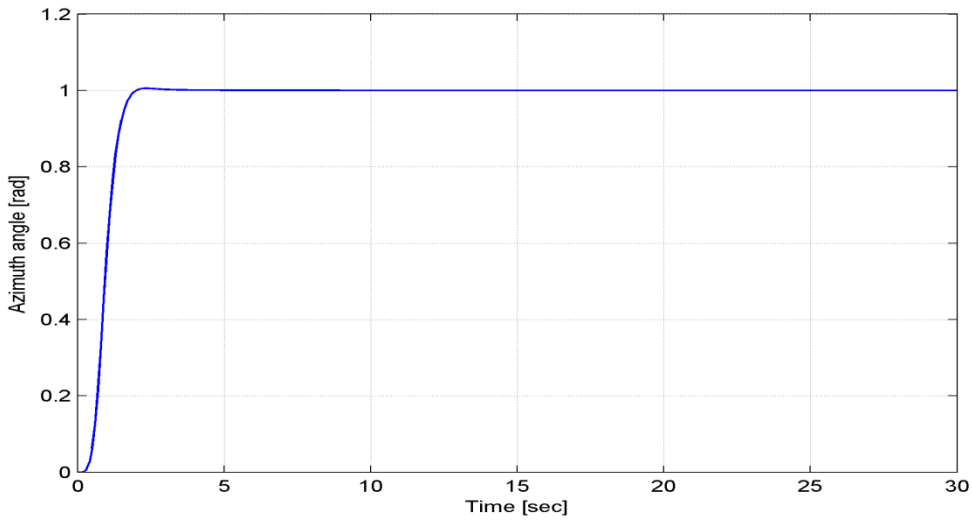


Fig. 3.19: Variations de l'angle d'Azimut (references: $\psi_r = 1$ et $\varphi_r = 1$)

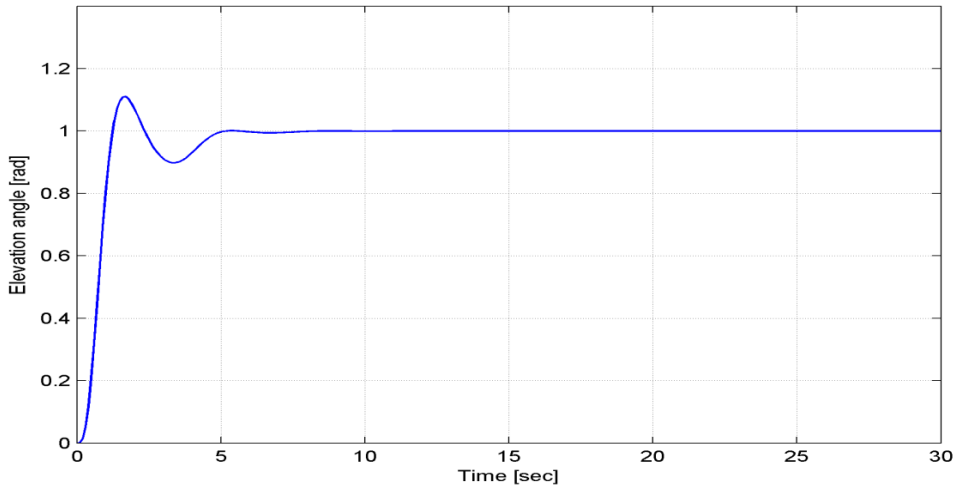


Fig.3.20: variations de l'angle d'élevation (references: $\psi_r = 1$ et $\varphi_r = 1$)

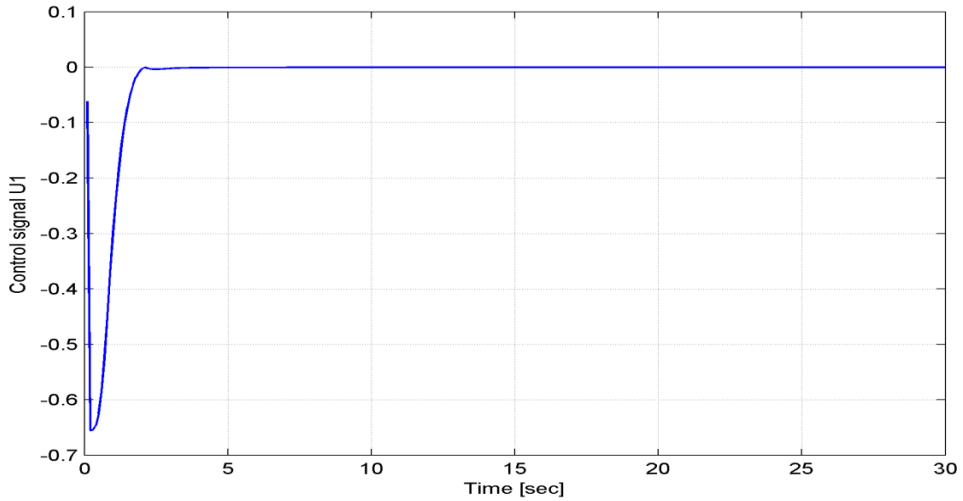


Fig. 3.21 : Signal de contrôle U_1 ($\psi_r = 1$ et $\varphi_r = 1$)

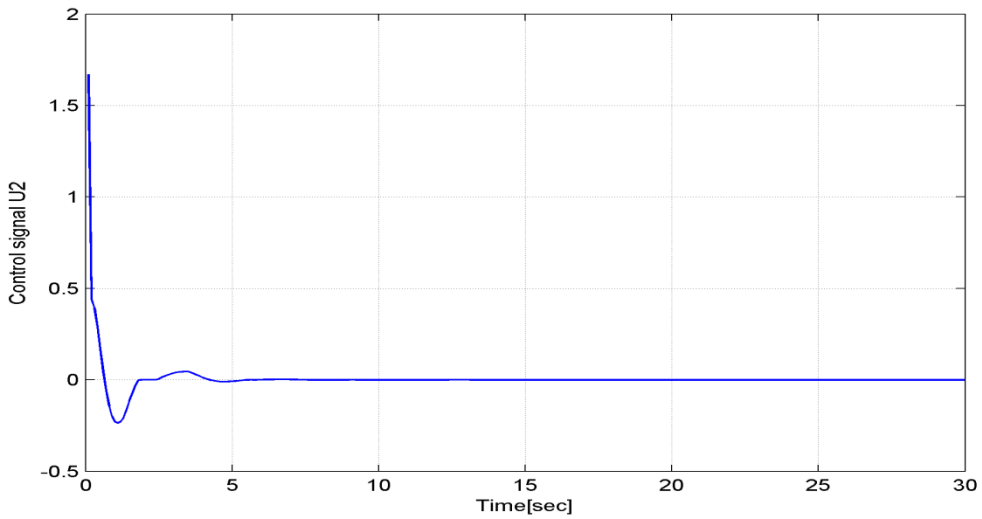


Fig. 3.22 : Signal de contrôle U2 (références: $\psi_r = 1$ et $\varphi_r = 1$)

3.4.2.3 Robustesse des contrôleurs neuro-flou

Les contrôleurs CNF obtenus ont été testés pour des variations des angles de références (l'azimut et l'élévation). D'une part, les figures (3.23, 3.24, 3.25 et 3.26) montrent les résultats de simulation lorsque $\psi_r = 1 \ et \ \varphi_r = 0$, on constate que la variation de l'azimut est très légère et la perturbation agissant sur l'élévation est rapidement éliminée. D'autre part, les figures (3.27, 3.28, 3.29 et 3.30) montrent les résultats de simulation lorsque $\psi_r = 0 \ et \ \varphi_r = 1$, On constate que la variation de l'angle d'élévation n'a pas d'effets sur l'angle d'azimut.

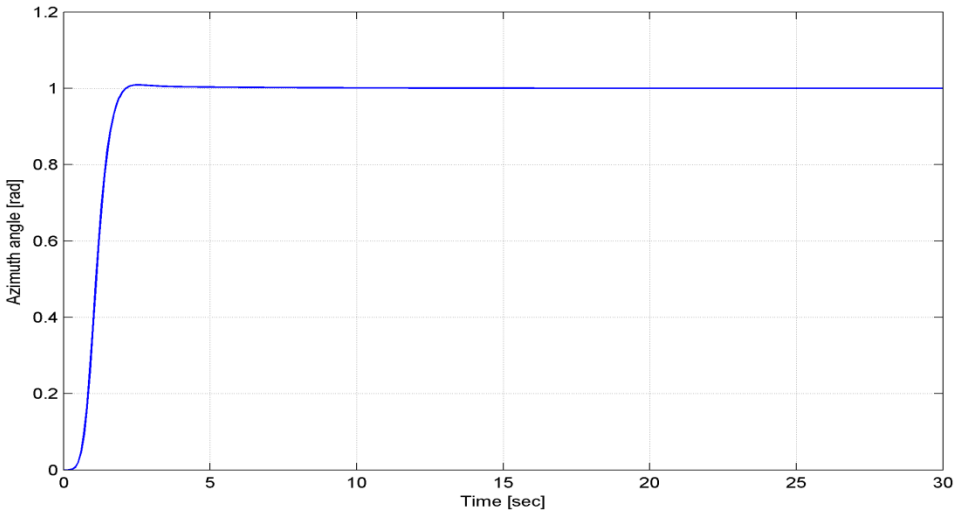


Fig. 3.23:Variations de l'angle d'Azimut (references: $\psi_r = 1 \ et \ \varphi_r = 0$)

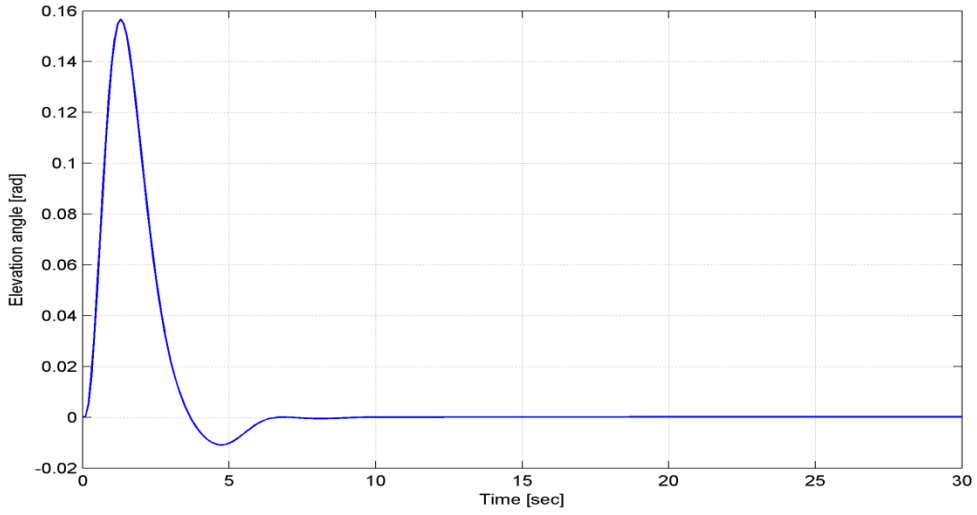


Fig.3.24: variations de l'angle d'élévation (references: $\psi_r = 1 \ et \ \varphi_r = 0$)

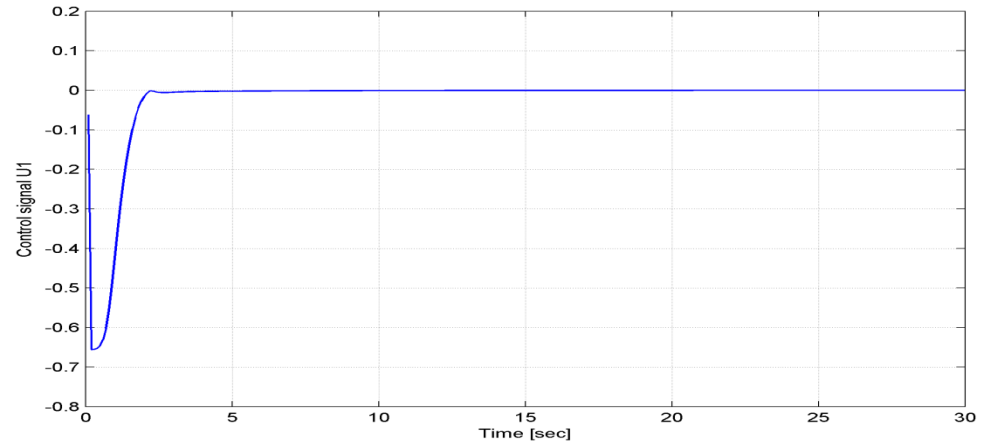


Fig. 3.25 : Signal de contrôle U_1 ($\psi_r = 1$ et $\varphi_r = 0$)

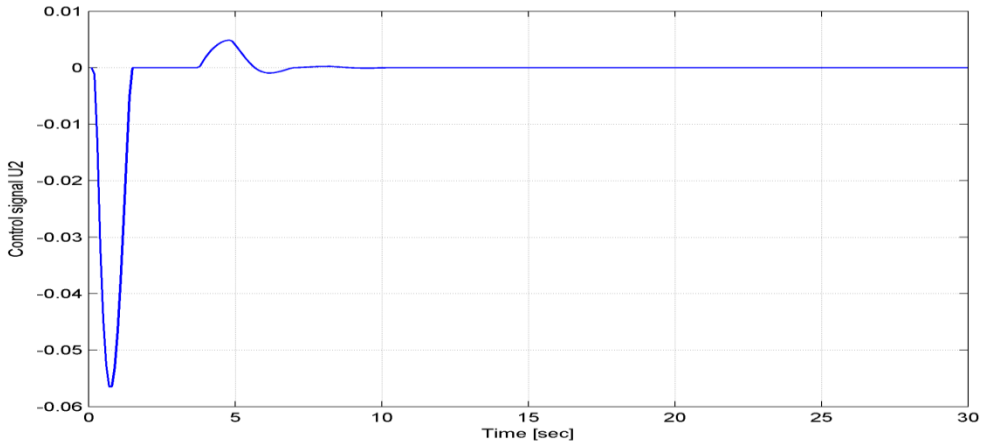


Fig. 3.26 : Signal de contrôle U2 (références:$\psi_r = 1$ et $\varphi_r = 0$)

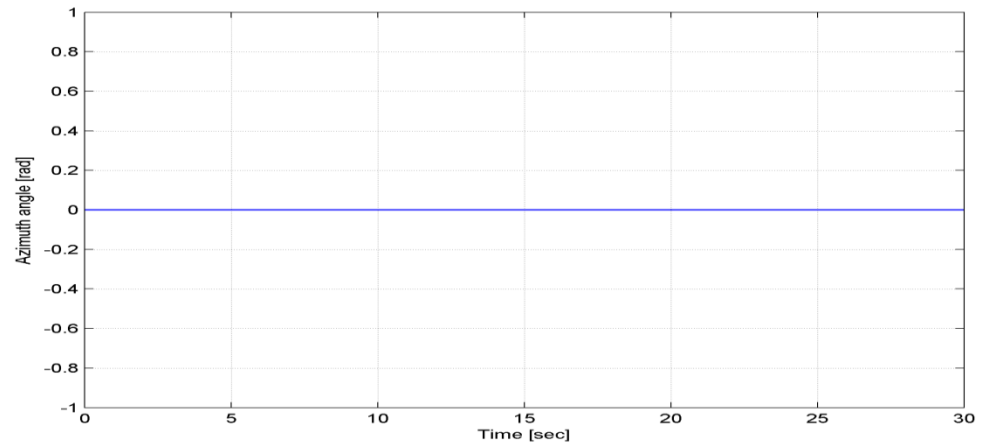


Fig. 3.27:Variations de l'angle d'Azimut (references: $\psi_r = 0$ et $\varphi_r = 1$)

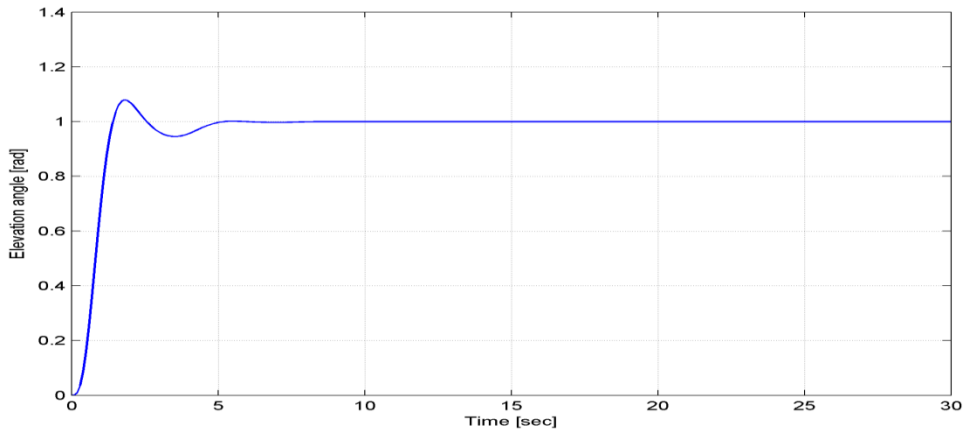


Fig.3.28: variations de l'angle d'élevation (references: $\psi_r = 0$ *et* $\varphi_r = 1$*)*

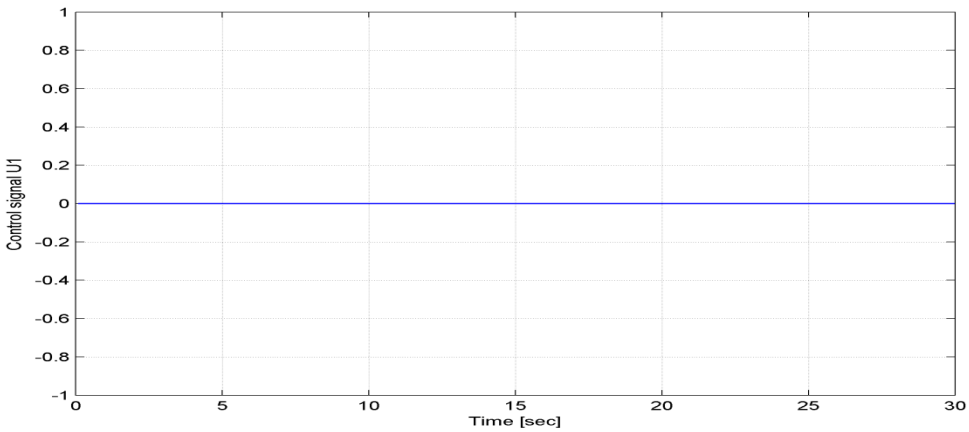


Fig. 3.29 : Signal de contrôle U_1 *(*$\psi_r = 0$ *et* $\varphi_r = 1$*)*

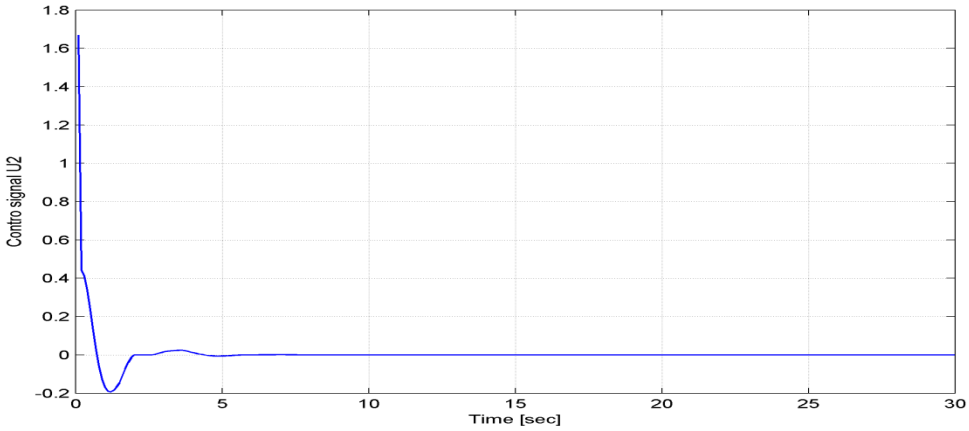


Fig. 3.30 : Signal de contrôle U2 (références:$\psi_r = 0$ *et* $\varphi_r = 1$*)*

3.5 Conclusion

Dans ce chapitre, on a proposé une nouvelle approche de conception des contrôleurs Neuro-flous assurant à la fois une bonne précision et une haute interprétabilité. Ce compromis précision-interprétabilité dans la conception des SIF est traité en tant qu'un problème d'optimisation à deux objectifs.

Pour résoudre ce problème, un nouvel algorithme a été proposé dit MBR-NSGA2 (mixed Binary-Real Non dominated Sorting Genetic Algorithm II), il a été utilisé pour trouver les valeurs optimales des fonctions d'appartenance des variables d'entées/sorties et la base des règles floues modélisées par des poids binaires assujettis à certaines contraintes afin de préserver l'interprétabilité des règles floues durant le processus d'optimisation. Du à la complexité du problème d'optimisation, les opérateurs génétiques ont été améliorés.

Pour vérifier la performance de cette approche, deux exemples de contrôle ont été considérés : contrôle du système du pendule inversé et contrôle décentralisé multivariable d'un simulateur d'hélicoptère. Dans les deux cas, le processus d'optimisation a produit des CNF compacts avec une haute précision et robustesse et une très bonne interprétabilité.

Chapitre IV :

Les réseaux d'ondelettes flous

4.1 Introduction

Une large variété de structures neuronales et flous, appelés les systèmes Neuro-flous, traitant les problèmes d'identification et de contrôle ont été proposées dans la littérature. Les principales raisons derrière l'utilisation des réseaux de neurones résident dans leurs propriétés, à savoir : les capacités d'apprentissage et de généralisation, mappage non linéaire et le calcul parallèle.

Ces dernières années, les réseaux d'ondelette (WNN), qui combine la théorie des ondelettes et les réseaux de neurones sont devenue un sujet très actif dans de nombreux domaines de recherche et deviennent un outil très populaire pour l'approximation des fonctions [122-123].

Les WNN peuvent offrir de meilleures performances dans l'apprentissage que les réseaux de neurones conventionnelles, ceci est du au fait que les fonctions d'activation tel que les fonctions gaussiennes et sigmoïde sont remplacées par des fonctions ondelettes dans les couches cachées du WNN. Ces fonctions réalisent l'analyse du signal dans le domaine temporel et fréquentiel. Cependant, contrairement aux NN qui sont des réseaux globaux, les WNN sont des réseaux locaux dans lesquels la sortie est bien localisée dans les domaines temps et fréquence, ainsi la vitesse d'apprentissage sera plus rapide.

Par conséquent, la combinaison des WNN et les SIF permet de développer des systèmes, appelés les réseaux d'ondelettes flous (FWNN), qui sont caractérisés par une vitesse d'entrainement rapide, requièrent peu de neurones pour identifier des systèmes d'une plus grande complexité et d'obtenir également une convergence plus précise. Dans ce chapitre on va voir quelques structures qui montrent la fusion des systèmes flous et les réseaux d'ondelettes.

4.2 La transformée en ondelette

Les ondelettes sont principalement utilisées pour la décomposition des fonctions. La décomposition d'une fonction en ondelettes consiste à l'écrire comme une somme pondérée de fonctions obtenues à partir d'opérations simples effectuées sur une fonction principale appelée ondelette mère. Ces opérations consistent en des translations et dilatations de la variable. Selon que ces translations et dilatations sont choisies de manière continue ou discrète, on parlera d'une transformée en ondelettes continue ou discrète [124].

4.2.1 la transformée continue

Une transformée en ondelettes continue (CWT) d'une fonction réelle $f(x)$ est définie par le produit scalaire de f et de ϕ :

$$CWT(t,d) = \frac{1}{\sqrt{d}} \int_{-\infty}^{+\infty} f(x)\, \phi\left(\frac{x-t}{d}\right) dx \qquad (4.1)$$

Ou ϕ est l'ondelette mère caractérisée par les paramètres de translation $t \in R$ et de dilatation $d \in R^{*+}$.

4.2.2 la transformée discrète

La transformée en ondelettes discrète (DWT) est une implémentation utilisant un ensemble discret de dilatations et de translations d'ondelettes.

L'expression de la transformée en ondelettes discrète est donnée par :

$$DWT(m,n) = a_0^{-\frac{m}{2}} \int_{-\infty}^{+\infty} f(x)\phi(a_0^{-m}\, x - nb_0)\, dx \qquad (4.2)$$

Ou : $d = a_0^m \; et \; t = nb_0$ avec $a_0 \; et \; b_0 \in Z$

4.2.3 Différents types d'ondelette mère

Plusieurs types d'ondelette mère ont été proposés dans la littérature, les plus connues sont les suivantes [125] :

Ondelette de Morlet (Fig.4.1):

$$\phi(x) = e^{(-x^2)} \cos(5x) \qquad (4.3)$$

Ondelette Chapeau Mexicain (Fig.4.2) :

$$\phi(x) = (1 - 2x^2)e^{-x^2} \qquad (4.4)$$

Ondelette de Shannon (Fig.4.3) :

$$\phi(x) = \frac{\sin(\pi x)}{\pi x} \qquad (4.5)$$

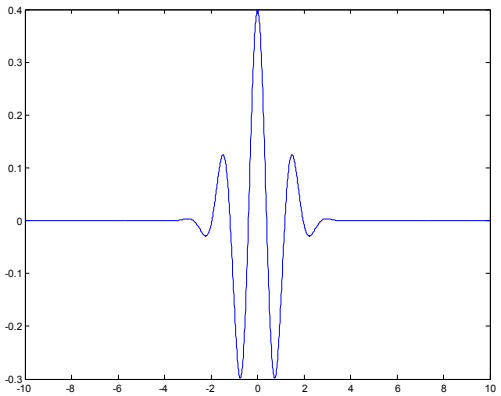


Fig.4.1: ondelette de Morlet

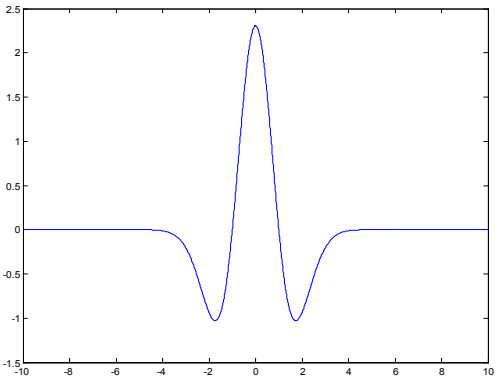


Fig.4.2: Chapeau mexicain

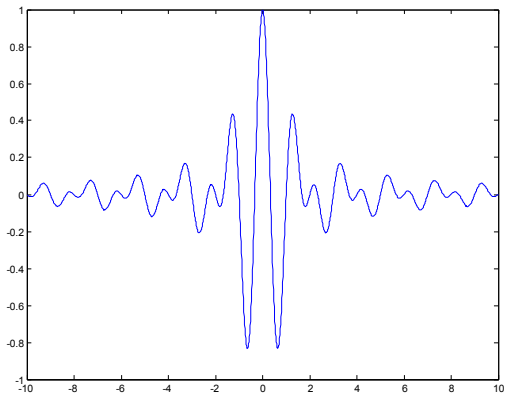


Fig.4.3: Ondelette de Shannon

4.2.4 Les ondelettes multidimensionnelles

La notion des ondelettes multidimensionnelles s'avère très utile lorsqu'on a affaire à des systèmes multivariables. Une ondelette multidimensionnelle est définie comme étant le produit d'ondelettes monodimensionnelles, elle est exprimée par l'expression suivante [124] :

$$\Phi_j(x) = \prod_{k=1}^{N_i} \phi(z_{jk}) \quad , \qquad Avec \quad z_{jk} = \frac{x_k - t_{jk}}{d_{jk}} \tag{4.6}$$

Où x_k représente la composante k du vecteur d'entrée x et z_{jk} la composante centrée par le facteur de translation t_{jk} et dilatée par le facteur d_{jk} (Fig.4.4)

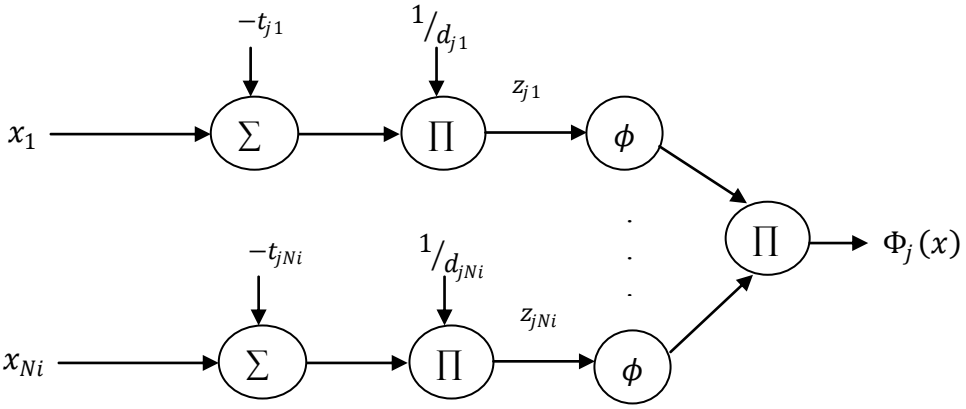


Fig.4.4: Ondelette multidimensionnelle

4.3 Les réseaux d'ondelettes (wavenets)

Les réseaux d'ondelette ont été proposés pour la première fois dans le cadre de la modélisation des systèmes statiques dans [126]. Un réseau d'ondelettes peut être considéré comme étant un réseau direct constitué de trois couches. Une couche d'entrée recevant les N_i entrées du vecteur x, une couche cachée contenant N_w ondelettes et une couche de sortie réalisant une somme pondérée des ondelettes multidimensionnelles Φ_j (Fig.4.5). Ainsi la sortie du réseau est exprimée par :

$$y(x) = \sum_{j=1}^{N_w} C_j \Phi_j(x) \tag{4.7}$$

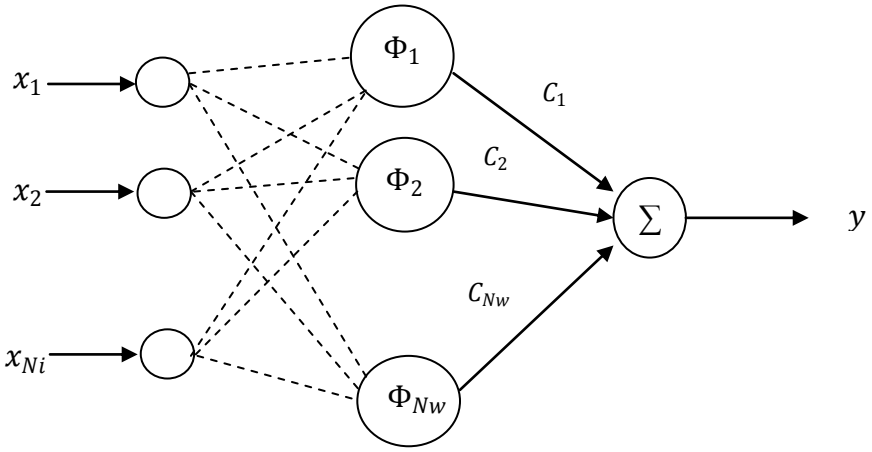


Fig.4.5: structure d'un réseau d'ondelettes

L'apprentissage du réseau d'ondelettes consiste en la minimisation de la fonction coût suivante :

$$J(\theta) = \frac{1}{2}\sum_{n=1}^{N}\left(y_p^n - y^n\right)^2 \tag{4.8}$$

Où θ est un vecteur contenant l'ensemble des paramètres ajustables du réseau : les translations t_{jk}, les dilatations d_{jk} et les coefficients de pondération C_j, y_p^n est la sortie désirée correspondant à l'exemple n et y^n est la sortie du réseau d'ondelettes pour l'exemple n.

4.4 Les réseaux d'ondelettes flous (FWNN)

Plusieurs travaux scientifiques ont proposé et discuté la synthèse des FWNN pour la prédiction des séries temporelles, l'approximation des fonctions, l'identification et le contrôle des systèmes [127-134]. Dans cette partie on présenter quelques architectures de réseaux FWNN.

4.4.1 Le réseau FWNN proposé dans [132]

Dans cette approche, le réseau FWNN considéré combine un système flou de type TS avec des fonctions d'ondelettes. Les règles floues sont exprimées par :

$$Si\ x_1\ est\ A_1^{i_1}\ et\ x_2\ est\ A_2^{i_2}\ et \ldots et\ x_n\ est\ A_n^{i_n}\ \ Alors\ \ \Psi_l(x)$$

Où $x_1, x_2, \ldots, x_n$ sont les variables d'entrée, $A_j^{i_j}$ représente la $j$ème fonction d'appartenance gaussienne de la $j$ème entrée, et $\Psi_l(x)$ représente la sortie l de la règle floue.

Dans leur article, trois structures FWNN ont été proposées suivant la partie conséquence de la règle : FWNN-S(Summation), FWNN-M(Multiplication) et FWNN-R(Radial). L'ondelette mère adoptée est l'ondelette du chapeau mexicain exprimée par :

$$\psi(x) = (1 - x^2)e^{-\frac{1}{2}x^2} \tag{4.9}$$

L'architecture du réseau consiste en six couches comme indiqué dans (Fig.4.6)

Couche 1 : Couche d'entrée recevant les signaux d'entrée $x_1, x_2, \dots, x_n$.

Couche 2 : Couche de Fuzzification contenant des nœuds qui implémentent des fonctions d'appartenance de type gaussien :

$$A_j^{i_j} = exp\left(-\frac{1}{2}\left(\frac{x_j - \mu_{i_j}}{\sigma_{i_j}}\right)^2\right), \quad j = 1,2,\dots,n \; et \; i_j = 1,2,\dots,l_j \tag{4.10}$$

Couche 3 : Chaque nœud de cette couche représente une règle floue dont la sortie est exprimée par :

$$\eta_l = \prod_{j=1}^{n} A_j^{i_j}(x_j), \quad (l = i_1, i_2, \dots, i_n, i_1 = 1, \dots, l_1, \dots, i_n = 1, \dots, l_n). \tag{4.11}$$

Couche 4 : C'est une couche de normalisation, les sorties sont calculées par :

$$\bar{\eta}_l = \frac{\eta_l}{\sum_{i=1}^{m} \eta_i}, \; l = 1, \dots, m. \tag{4.12}$$

Couche 5 : les nœuds de cette couche calculent la valeur de la sortie pondérée de la règle par :

$$f_l = \bar{\eta}_l \Psi_l, \qquad l = 1, \dots, m \tag{4.13}$$

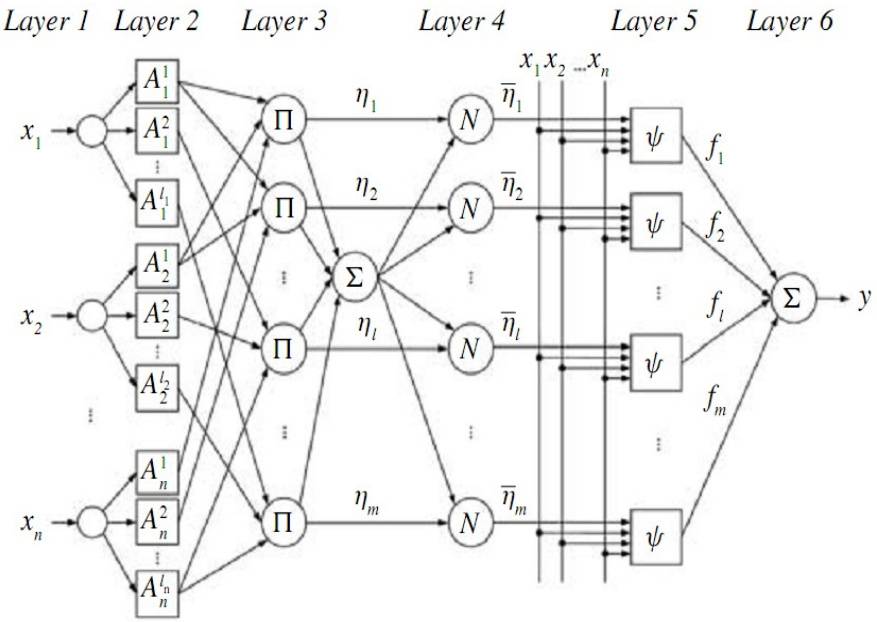


Fig.4.6 : Architecture de la structure FWNN[132]

Pour l'architecture FWNN-S, on a :

$$\Psi_l = \sum_{i=1}^{n} w_{il}\left(1-\left(\frac{x_i - t_{il}}{d_{il}}\right)^2\right) exp\left(-\frac{1}{2}\left(\frac{x_i - t_{il}}{d_{il}}\right)^2\right) \quad (4.14)$$

Pour l'architecture FWNN-M, on a :

$$\Psi_l = w_l \prod_{i=1}^{n}\left(1-\left(\frac{x_i - t_{il}}{d_{il}}\right)^2\right) exp\left(-\frac{1}{2}\left(\frac{x_i - t_{il}}{d_{il}}\right)^2\right) + p_l \quad (4.15)$$

Pour l'architecture FWNN-R, on a :

$$\Psi_l = w_l\left(1-\left(\frac{\|x - t_l\|}{d_l}\right)^2\right) exp\left(-\frac{1}{2}\left(\frac{\|x - t_l\|}{d_l}\right)^2\right) + p_l \quad (4.16)$$

Couche 6 : Cette couche calcule la sortie totale du réseau qui est exprimée par :

$$y = \sum_{l=1}^{m} f_l \quad (4.17)$$

L'ajustement des paramètres $\{\mu, \sigma, t, d, w, p\}$ du réseau FWNN est réalisé par une méthode basée sur le gradient dite BFGS (Broyden–Fletcher–Goldfarb–Shanno).

4.4.2 Le réseau FWNN proposé dans [133-134]

Cette approche utilise des règles floues du type TS de la forme :

$$R^n: Si\ x_1\ est\ A_{n1} et\ x_2\ est\ A_{n2} et, \dots, x_m\ est\ A_{nm}\ \ Alors\ y_n = \sum_{i=1}^{m} w_{in}(1 - z_{in}^2)\, e^{-\frac{z_{in}^2}{2}}$$

Où $x_1, x_2, \dots, x_m$ représentent les variables d'entrée, $y_1, y_2, \dots, y_n$ sont les variables de sortie, A_{ij} représente la fonction d'appartenance gaussienne de la $jème$ entrée da la règle i.

La structure du FWNN proposé dans cet article est illustrée dans (Fig.4.7). Elle comporte sept couches. Les nœuds de la première couche correspondent au nombre des signaux d'entrée. Les nœuds de la deuxième couche implémentent des fonctions d'appartenance de type gaussien :

$$\eta_j(x_i) = exp\left(-\left(\frac{x_i - c_{ij}}{\sigma_{ij}}\right)^2\right), \quad i = 1,2, \dots, m\ et\ j = 1,2, \dots, n \quad (4.18)$$

Où m représente le nombre des entrées et n correspond au nombre des règles floues.

Chaque nœud de la troisième couche représente une règle floue. Les sorties de cette couche sont calculées en utilisant l'opérateur Min ($\prod$) :

$$\mu_j(x) = \prod_{i=1}^{m} \eta_j(x_i), \qquad j = 1, \dots, n. \quad (4.19)$$

La quatrième couche contient n fonctions ondelettes (WF) dont les sorties sont calculées par :

$$y_l = w_l \Psi_l(x) = w_l \sum_{i=1}^{m} |d_{il}|^{-\frac{1}{2}} (1 - z_{il}^2)\, e^{-\frac{z_{il}^2}{2}} \;, l = 1, \dots, n. \tag{4.20}$$

$$\text{Avec} \quad z_{il} = \frac{(x_i - t_{il})}{d_{il}}$$

La défuzzification est réalisée par les couches six et sept et la sortie globale du FWNN est calculée par :

$$u = \frac{\sum_{l=1}^{n} \mu_l(x)\, y_l}{\sum_{l=1}^{n} \mu_l(x)} \tag{4.21}$$

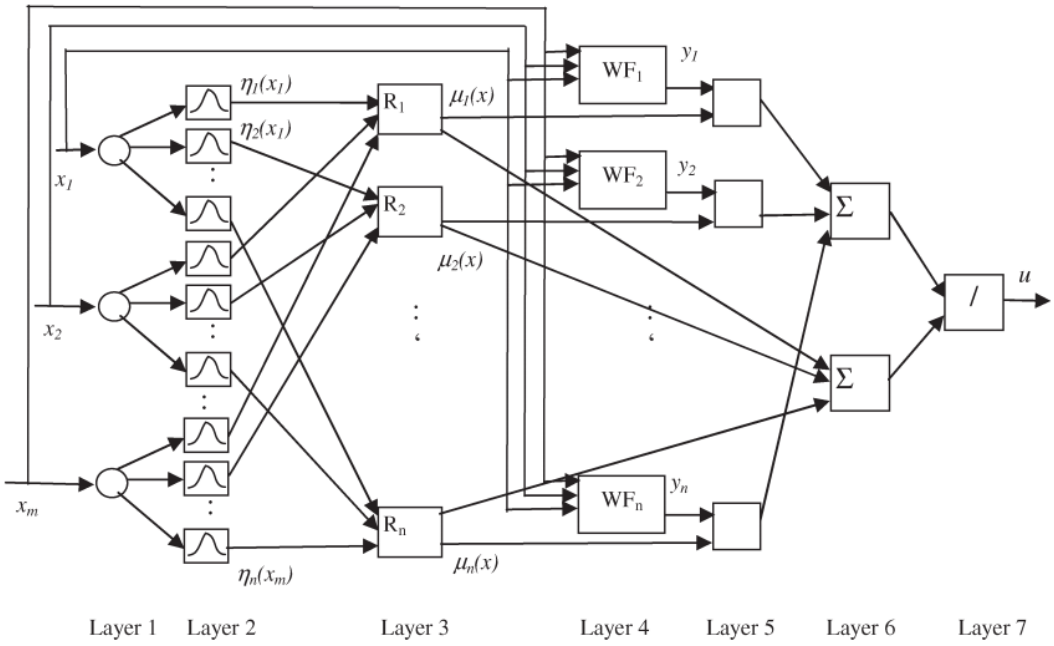


Fig.4.7 : Architecture de la structure FWNN[133]

Un algorithme basé sur la méthode du gradient a été employée pour l'ajustement des paramètres $\{c, \sigma, t, d, w\}$ du FWNN.

4.4.3 Le réseau FWNN proposé dans [131]

Dans cette approche, une structure adaptative dite AFWNN (Fig.4.8) a été proposée pour l'approximation des fonctions non linéaires. Le principe de cette méthode est basé sur l'architecture ANFIS proposée dans [135]. Cette structure implémente des règles floues du type :

$$Si\ x_1\ est\ A_1^{i_1}\ et\ x_2\ est\ A_2^{i_2}\ et \dots et\ x_n\ est\ A_n^{i_n}\ \ Alors\ \ \Psi_l(x)$$

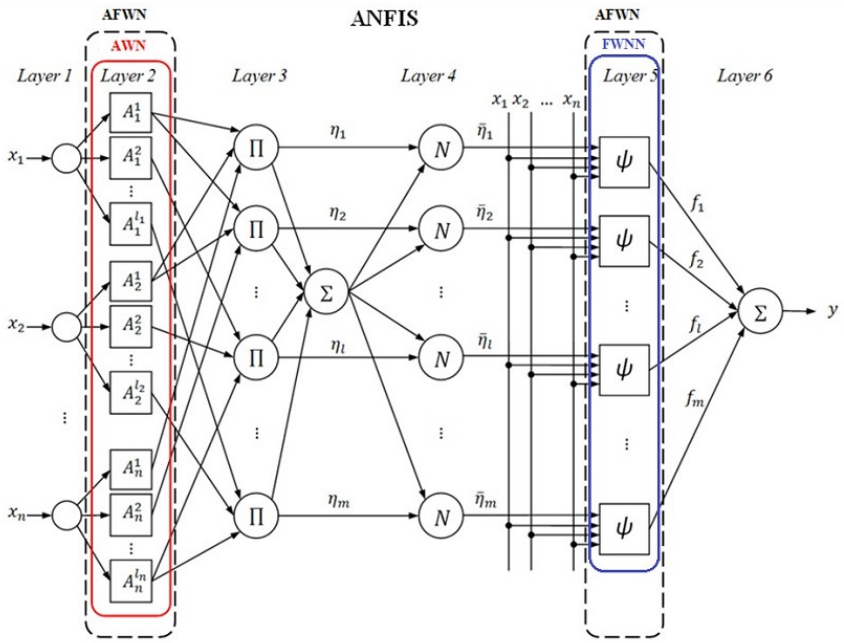


Fig.4.8 : Architecture de la structure AFWNN[131]

La structure du réseau comporte six couches :

Couche 1 : Couche d'entrée correspondant aux signaux d'entrée $x_1, x_2, \dots, x_n$.

Couche 2 : Couche de fuzzification, les nœuds de cette couche utilisent des fonctions d'appartenance sous forme d'ondelette du type chapeau mexicain exprimées par :

$$A_j^i = \left(1 - \left(\frac{x_j - \mu_{i_j}}{\sigma_{i_j}}\right)^2\right) exp\left(-\frac{1}{2}\left(\frac{x_j - \mu_{i_j}}{\sigma_{i_j}}\right)^2\right), j = 1,2, \dots, n \; et \; i_j = 1,2, \dots, l_j \qquad (4.22)$$

Couche 3 : La sortie du nœud l de cette couche est calculée par :

$$\eta_l = \prod_{j=1}^{n} A_j^{i_j}(x_j), \quad (l = i_1, i_2, \dots, i_n, i_1 = 1, \dots, l_1, \dots, i_n = 1, \dots, l_n). \qquad (4.23)$$

Ainsi le nombre total des règles est égal à $m = \prod_{i=1}^{n} l_i$

Couche 4 : couche de normalisation, chaque nœud calcule la valeur normalisée d'une règle par :

$$\bar{\eta}_l = \frac{\eta_l}{\sum_{i=1}^{m} \eta_i}, l = 1, \dots, m. \qquad (4.24)$$

Couche 5 : les nœuds de cette couche calculent la conséquence pondérée d'une règle par :

$$f_l = \bar{\eta}_l \Psi_l, \qquad l = 1, \dots, m \qquad (4.25)$$

Avec

$$\Psi_l = \sum_{i=1}^{n} w_{il} \left(1 - \left(\frac{x_i - t_{il}}{d_{il}}\right)^2\right) exp\left(-\frac{1}{2}\left(\frac{x_i - t_{il}}{d_{il}}\right)^2\right), l = 1, \dots, m \qquad (4.26)$$

Couche 6 : le nœud de cette couche calcule la sortie globale du réseau par la formule :

$$y = \sum_{l=1}^{m} f_l \tag{4.27}$$

Les paramètres ajustables du réseau AFWNN sont les paramètres de translation (μ) et de dilatation (σ) des fonctions d'appartenance ondelette de la partie antécédente de la règle floue ; les paramètres de translation (t) et de dilatation (d) des fonctions d'ondelette et les poids w.

Le mécanisme d'ajustement consiste à minimiser l'erreur quadratique moyenne (EQM) suivante :

$$E = \frac{1}{N}\sum_{k=1}^{N}(y - y_d)^2 \tag{4.28}$$

Où N représente le nombre de paires d'entrée-Sortie de la fonction à approximer, y_d est la sortie désirée et y est la sortie du modèle AFWNN.

L'algorithme d'ajustement utilisé consiste en une méthode basée sur le gradient dite l'algorithme DFP (Davidon-Fletcher-Powel).

4.4.4 Le réseau FWNN proposé dans [127]

Dans [127], une structure d'un FWNN (Fig.4.9) a été proposée pour l'identification et le contrôle des systèmes dynamiques non linéaires. Pour le problème de contrôle, une loi de commande prédictive à base de FWNN dite WFNNPC (Wavelet Fuzzy Neural Networks Predictive Control) a été présentée. Le réseau proposé implémente un SIF ayant les règles du type :

$$R^l: Si\ x_1 est\ F_{1l}\ et\ , \dots , et\ x_n est\ F_{nl}\ Alors\ \hat{y}\ est\ \omega_l$$

Où $x_1, x_2, \dots, x_n$ sont les variables d'entrée, les F_{il} sont les ensembles correspondants aux entrées x_i dans la $l^{ième}$ règle, $\hat{y}$ est la variable de sortie du réseau et ω_l représente le poid de la conséquence de règle l.

Ce réseau est structuré en six couches dont le fonctionnement est comme suit :

Couche 1 : C'est une couche d'entrée ; les noeuds de cette couche transmettent directement les valeurs d'entrée vers la couche suivante.

Couche 2 : Couche DE fuzzification ; chaque nœud de cette couche accompli une fonction d'appartenance de type gaussien :

$$\mu_{il} = exp\left(-\frac{(x_i - m_{il})^2}{2\sigma_{il}^2}\right) \tag{4.29}$$

Couche 3 : Chaque nœud de cette couche effetu le produit de ses signaux d'entrée :

$$\bar{\mu}_l = \prod_{i=1}^{n} \mu_{il} \tag{4.30}$$

Couche 4 : chaque nœud de cette couche est marqué par N dont la sortie est calculée par :

$$v_l = \frac{\bar{\mu}_l}{\sum_{l=1}^{L} \bar{\mu}_l} \tag{4.31}$$

$$\text{Où } 0 < v_l \leq 1 \; et \; \sum_{l=1}^{L} v_l = 1$$

Couche 5 : La fonction ω_l réalisée par chaque nœud de cette couche est donnée par :

$$\omega_l = \left(\sum_{i=1}^{N} \emptyset_{il}\right) w_l = \left(\sum_{i=1}^{N} -v_{il} \; exp\left(\frac{-v_{il}^2}{2}\right)\right) w_l \tag{4.32}$$

$$\text{Où } v_{il} = \frac{x_i - t_{il}}{d_{il}}$$

Avec $\emptyset_{il}$ est l'ondelette mère décrite par la première dérivée de la fonction gaussienne, t_{il} et d_{il} sont les paramètres de translation et de dilatation de cette ondelette et w_l sont les poids associés à ce nœud. Ainsi, la sortie du nœud est exprimée par :

$$o_l = \omega_l v_l \tag{4.33}$$

Couche 6 : Le nœud de cette couche calcule la sortie globale du FWNN par :

$$\hat{y} = \sum_{l=1}^{L} o_l = \sum_{l=1}^{L} \omega_l v_l \tag{4.34}$$

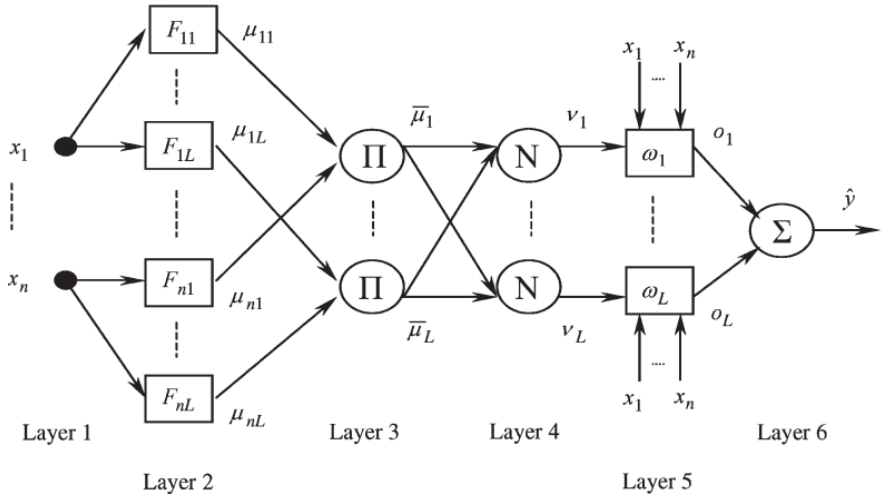


Fig.4.9 : Architecture de la structure FWNN[127]

L'ajustement des paramètres du réseau est assuré par l'algorithme de descente du gradient en considérant la minimisation de la fonction d'erreur suivante :

$$J(k) = \frac{1}{2}\big(y(k) - \hat{y}(k)\big)^2 \tag{4.35}$$

4.5 Optimisation des réseaux d'ondelettes flous

L'optimisation est un problème primordial dans la conception des FWNN. En effet, il s'agit de trouver un ensemble optimal des paramètres caractérisant le réseau, le plus souvent, les paramètres des fonctions d'appartenance, les paramètres des fonctions d'ondelette ainsi que les poids des règles floues. Les méthodes d'optimisation rencontrées dans la littérature peuvent être subdivisées en deux grandes catégories :

- Les méthodes basées sur le calcul des dérivés [52-55].
- Les méthodes employant les méthodes évolutionnaires [56, 57, 59].

Les méthodes basées sur le calcul des dérivés sont généralement des méthodes employant l'algorithme du gradient et sont donc considérées comme des méthodes de recherche locales. Cependant, ces méthodes sont très sensibles aux valeurs initiales des paramètres, de plus la fonction à optimiser doit être différentiable comme elles risquent de tomber éventuellement dans un optimum local. Pour remédier à ces problèmes, l'utilisation des méthodes évolutionnaires s'avère très utile.

4.6 Conclusion

Dans ce chapitre, on a présenté différentes structures de réseaux d'ondelette flous.Ces structures consistent à combiner les réseaux d'ondelettes et les systèmes d'inférence flous. Cette combinaison a pour objectif de produire des systèmes qui ont une capacité d'apprentissage très rapide, qui peuvent décrire des non-linéarités, qui nécessitent un nombre réduit de neurones pour identifier des systèmes d'une grande complexité et qui ont une convergence très précise.

Chapitre V :

Optimisation d'un réseau FWNN par les AG

5.1 Introduction

Récemment, plusieurs travaux scientifiques ont utilisés la structure dite FWNN pour la modélisation des systèmes non linéaires caractérisés par l'incertitude (132), l'approximation des fonctions (56), et pour les problèmes d'identification et contrôle des systèmes (127,137).

Dans ce chapitre, on va étudier l'utilisation des FWNN pour résoudre les problèmes d'identification et de contrôle. L'approche proposée combine plusieurs techniques du soft computing tel que les systèmes flous TS, les réseaux d'ondelettes et les algorithmes génétiques.

La structure du FWNN proposé consiste en une combinaison de deux structures ; une contenant les réseaux d'ondelettes et l'autre contient le réseau implémentant le mécanisme d'inférence flou. Un mécanisme d'optimisation à base d'algorithme génétique est utilisé pour obtenir les valeurs optimales des paramètres de translation, de dilatation et les poids du WNN et les paramètres des fonctions d'appartenance du système d'inférence flou.

5.2 Structure du réseau d'ondelettes

Un réseau d'ondelettes consiste en une structure possédant trois couches, utilisant les ondelettes comme fonctions d'activation. La structure du réseau d'ondelettes qu'on a proposé dans ce travail est schématisée dans la figure (5.1).

La configuration du réseau est comme suit:

- Une sortie y,
- n variables d'entrée $x = \{x_1, x_2, .., x_n\}$
- et N_w nœuds de la couche cachée.

La sortie du réseau est calculée par :

$$y = \sum_{j=1}^{N_w} w_j \Phi_j (Z_{jk}) + a_0 \tag{5.1}$$

Avec a_0 et w_j sont des coefficients de pondération et Φ_j est une fonction ondelette dérivée de l'ondelette mère ψ :

$$\Phi_j = \prod_{k=1}^{n} \psi(Z_{jk}) \tag{5.2}$$

Dans notre étude, la première dérivée de la fonction gaussienne est considérée comme une ondelette mère, elle est exprimée par :

$$\psi(x) = -xe^{-\frac{1}{2}x^2} \tag{5.3}$$

Où

$$Z_{jk} = \frac{x_k - t_{jk}}{d_{jk}} \tag{5.4}$$

Avec t_{jk} et d_{jk} sont les paramètres de translation et de dilatation respectivement. Les paramètres du WNN à optimiser sont a_0, w_j , t_{jk} et d_{jk} .

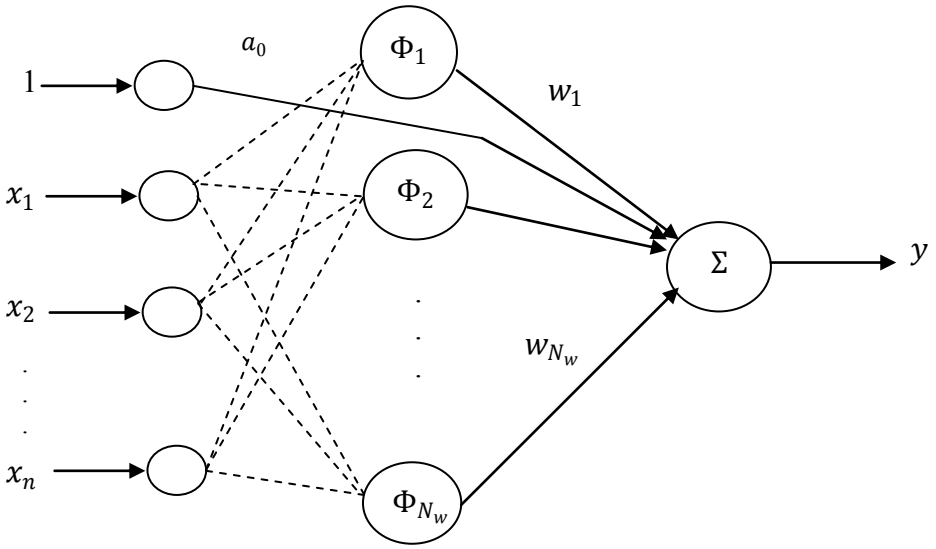


Fig. 5.1: Architecture du WNN proposé

5.3 Structure du réseau d'ondelettes floues

L'architecture du FWNN proposée est illustrée dans la figure (5.2)[140]. C'est un réseau multicouche direct qui réalise un SIF de type Takagi-Sugeno. Les règles floues à base d'ondelettes ont la forme suivante:

$$R_1: Si\ x_1\ est\ A_{11}\ et\ x_2\ est\ A_{21}\ \dots et\ x_n\ est\ A_{n1} \quad Alors\ y_1 = \sum_{j=1}^{N_w} w_{j,1}\Phi_j(Z_{jk,1}) + a_{0,1}$$

$$R_2: Si\ x_1\ est\ A_{12}\ et\ x_2\ est\ A_{22}\ \dots et\ x_n\ est\ A_{n2} \quad Alors\ y_2 = \sum_{j=1}^{N_w} w_{j,2}\Phi_j(Z_{jk,2}) + a_{0,2}$$

.

.

$$R_m: Si\ x_1\ est\ A_{1m}\ et\ x_2\ est\ A_{2m}\ \dots et\ x_n\ est\ A_{nm} \quad Alors\ y_m = \sum_{j=1}^{N_w} w_{j,m}\Phi_j(Z_{jk,m}) + a_{0,m}$$

Où les $x_1, x_2, \dots, x_n$ sont les entrées et les $y_1, y_2, \dots, y_m$ sont les sorties du WNN.

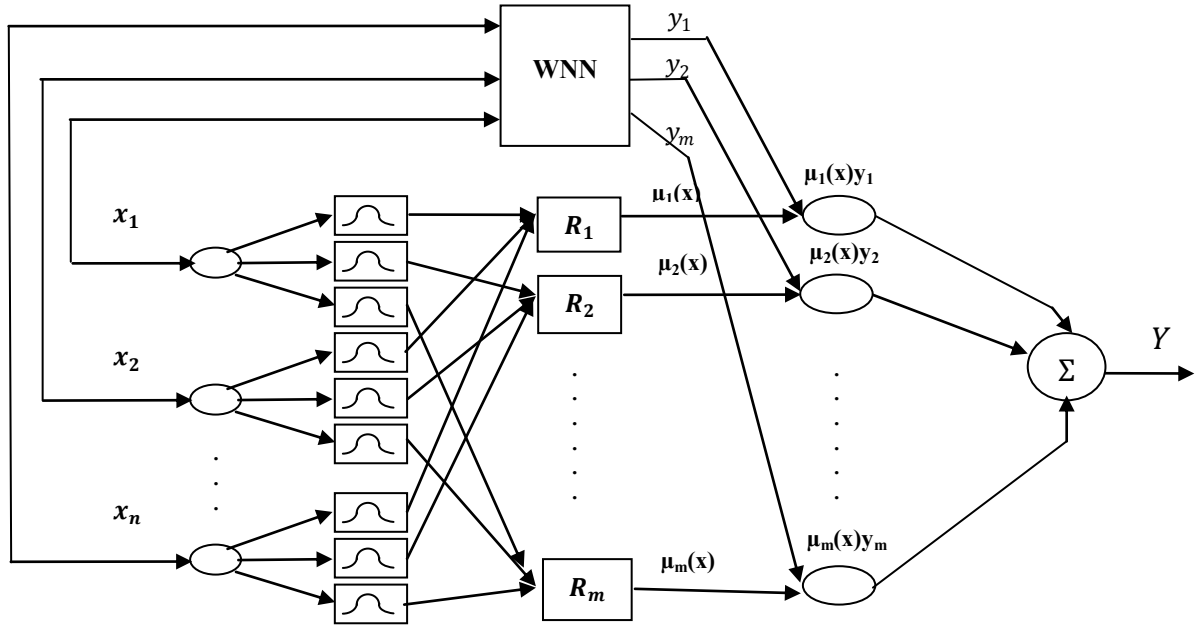


Fig. 5.2: Structure du réseau FWNN

La structure du FWNN consiste en une combinaison de deux réseaux. Le réseau supérieur contient les ondelettes et le réseau inférieur contient la structure du mécanisme de raisonnement flou. Le réseau FWNN consiste en cinq couches :

Couche 1 : C'est une couche d'entrée ; les nœuds de cette couche sont des nœuds d'entrée x_i, le nombre de ces nœuds est égale au nombre des signaux d'entrée.

Couche 2 : couche de Fuzzification ; les nœuds de cette couche calculent le dégrée d'appartenance d'une valeur d'entrée à un ensemble flou. Dans notre cas, des fonctions d'appartenance gaussienne ont été utilisées et sont exprimées par :

$$\mu_{A_{ij}}(x_i) = e^{-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}} \tag{5.5}$$

Avec c_{ij} et σ_{ij} représentent les centres et les largeurs des fonctions d'appartenance gaussiennes.

Couche 3 : Couche ET ; chaque nœud de cette couche représente une règle floue. L'opération ET considérée est exprimée par :

$$\mu_k(x) = \mu_{A_{1j_1}}(x_1).\mu_{A_{2j_2}}(x_2) \dots \mu_{A_{2j_n}}(x_n) \tag{5.6}$$

Où

$$j_i = 1 \dots N_i, i = 1 \dots n, k = 1 \dots \prod_{i=1}^{n} N_i \tag{5.7}$$

Et N_i représente le nombre d'ensembles flous associés à l'entrée i.

Couche 4 : couche conséquence ; les sorties de cette couche sont calculées en multipliant les signaux de sortie du réseau WNN et les sorties de la couche ET.

Couche 5 : couche de Défuzzification ; le nœud de cette couche calcule la sortie globale du réseau par :

$$Y = \sum_{k=1}^{m} f_k(x) y_k \tag{5.8}$$

Avec

$$f_k(x) = \frac{\mu_k(x)}{\sum_{l=1}^{m} \mu_l(x)} \tag{5.9}$$

5.4 Optimisation du FWNN

L'optimisation des paramètres est le problème principal dans la conception des FWNN. Pour résoudre ce problème, on a utilisé un algorithme génétique à codage réel (136) pour obtenir les valeurs optimales de la partie antécédente des règles: les centres (c_{ij}) et les largeurs (σ_{ij}) des fonctions d'appartenance gaussiennes; et de la partie conséquence des règles : translation (t_{jk}) et dilatation (d_{jk}) et les poids a_{0l} et $w_{j,l}$ des fonctions ondelettes.

Dans l'AG, une population de chromosomes est initialisée et puis évolue. Chaque chromosome inclut les paramètres ajustables. Au début, une reproduction sélective est appliquée à la population courante de sorte que les chromosomes tirent un certain nombre de copies proportionnelles à leur propre fonction coût ; ceci est fait par la méthode de la roulette. Alors de nouveaux chromosomes sont créés en utilisant les opérations de croisement et de mutation, qui sont régies par des probabilités. Dans cet algorithme, on a employé un croisement linéaire et une mutation uniforme (136).

Dans le croisement linéaire, trois individus (O1, O2, et O3) sont construits de deux parents P1 et P2 comme suit :

$$O_1 = 0.5.P_1 + 0.5.P_2$$
$$O_2 = 1.5.P_1 - 0.5.P_2$$
$$O_3 = -0.5.P_1 + 1.5.P_2$$

Ce croisement est exécuté pour chaque paramètre du chromosome. Avec ce type de croisement un mécanisme de choix de progéniture est appliqué, qui choisit les deux progénitures les plus prometteuses des trois pour substituer leurs parents dans la population.

Dans la mutation uniforme, nous définissons une direction positive et négative de mutation choisie aléatoirement avec la probabilité 0.5 (50%). Ainsi, si le paramètre :

$$p_i : (a_i < p_i < b_i)$$

est choisi pour la mutation, le paramètre résultant $p_i^{'}$ de l'application de l'opérateur de mutation est donné par :

$p_i^{'}$ est un nombre (uniforme) aléatoire du domaine $[p_i, \min(M, b_i)]$ pour la mutation positive.

$p_i^{'}$ est un nombre (uniforme) aléatoire du domaine $[\max(M, a_i), p_i]$ pour la mutation négative.

Où M représente la variation de la grandeur maximale du paramètre p_i.

Enfin, une nouvelle population est créée après que ce processus soit complété. Une fois que le critère d'arrêt, par exemple, un nombre prédéfini de générations est atteint ; le chromosome avec la valeur la plus élevée de la fonction coût dans la dernière génération est pris comme solution au problème.

En général la fonction coût à maximiser par l'AG est définie par :

$$fitness = \frac{1}{1+J} \tag{5.10}$$

Tel que J est l'indice de performance à minimiser tout en considérant le chromosome suivant:

$$\{t_{jk,l}, d_{jk,l}, a_{0,l}, w_{j,l}, c_{ij}, \sigma_{ij}\}$$

5.5 Résultats de simulation

5.5.1 Application à l'identification des systèmes

Afin d'évaluer la performance de l'approche proposée pour le problème de l'identification des systèmes dynamiques non linéaires, deux exemples ont été considérés [140].

5.5.1.1 Exemple1

Le système dynamique non linéaire à identifier est décrit par:

$$y(k) = 0.3\, y(k-1) + 06y(k-2) + f\big(u(k)\big) \tag{5.11}$$

Avec

$$f\big(u(k)\big) = u(k)^3 + 0.3\, u(k)^2 - 0.4\, u(k) \tag{5.12}$$

ET

$$u(k) = \begin{cases} \sin\left(\frac{2\pi k}{125}\right), & k \le 500 \\ 0.2\sin\left(\frac{2\pi k}{40}\right) + 0.8\sin\left(\frac{2\pi k}{25}\right), & k > 500 \end{cases} \tag{5.13}$$

Il est clair que la sortie actuelle du système dépend des deux sorties précédentes et le signal d'entrée actuel. Cependant, notre FWNN a seulement deux entrées $y(k-1)$ et $u(k)$ et une sortie $y(k)$.

Chacune de ces entrées possède trois fonctions d'appartenance gaussiennes, formant ainsi neuf règles. Par conséquent, il y a 12 paramètres à mettre à jour dans la partie antécédente des règles (paramètres des fonctions d'appartenance gaussiennes). En outre, neuf WNN sont

employés pour réaliser la partie conséquente des règles, résultant à 54 paramètres à mettre à jour. L'ensemble des données considérées pour l'identification contient 1000 échantillons, les 500 premiers sont employés pour la l'entrainement du réseau et les 500 autres sont employés pour la validation. Les paramètres de l'AG utilisés pour ce problème d'optimisation sont récapitulés dans le tableau 5.1.

Paramètre	valeur
Génération	100
Population	100
Probabilité de croisement	0.8
Probabilité de mutation	0.01

Tableau 5.1 : Paramètres de l'AG

La fonction d'adaptation à maximiser par l'AG est donnée par:

$$f = \frac{\alpha}{1 + \beta J} \tag{5.14}$$

Avec $\alpha \; et \; \beta$ sont des facteurs de mise à l'échelle fixés comme suit :

$$\alpha = 100 \; et \; \beta = 1$$

Et J est un critère de performance, l'erreur quadratique moyenne est utilisé avec k=500:

$$J = \frac{\sum_{i=2}^{k} (y_d(i) - y(i))^2}{k} \tag{5.15}$$

La figure (5.3) de la phase d'entrainement montre l'évolution des valeurs minimales de l'EQM à travers les 100 générations. On peut noter que l'entrainement du FWNN est très rapide et la convergence à la valeur optimale est obtenue à la génération 40.

Après 100 générations, la valeur de la meilleure fonction d'adaptation est de 99.99718 correspondant à la valeur $2.81629.10^{-5}$ du MSE, i.e., la sortie estimée par le FWNN est presque égale à la sortie actuelle, comme illustré dans la figure (5.4).

En observant les résultats de validation illustrés dans la figure (5.5), on constate que le FWNN suit parfaitement le système réel.

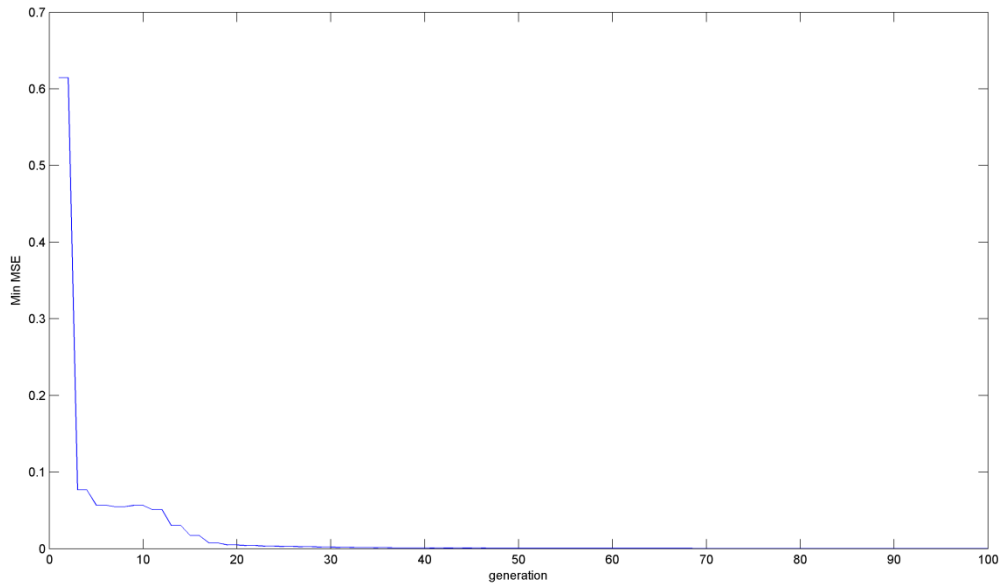


Fig.5.3: Valeurs de l' EQM dans la phase d'entrainement

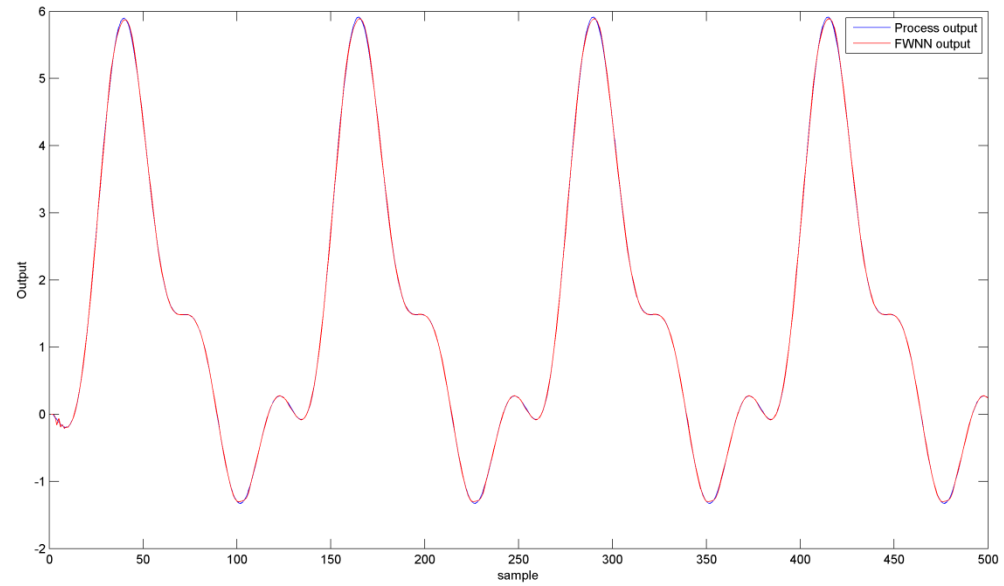


Fig.5.4 : Résultats d'entrainement de l'identification

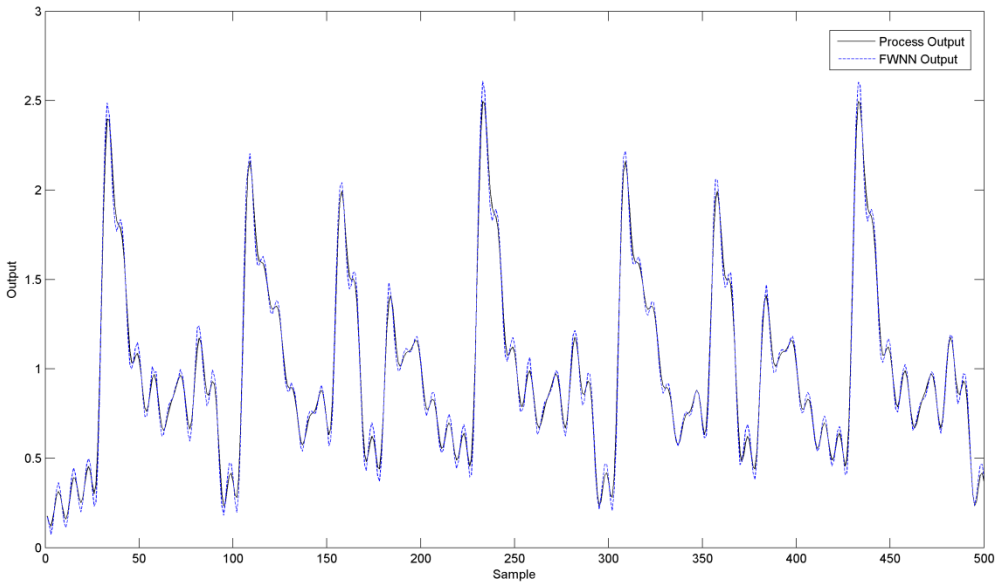


Fig.5.5 : Résultats de validation de l'identification

5.5.1.2 Exemple2

Le système est décrit par les équations suivantes :

$$y(k+1) = f\big(y(k), y(k-1), y(k-2), u(k), u(k-1)\big) \tag{5.16}$$

Avec $u(k)$ est donné par:

$$\begin{cases} \sin\left(\frac{k\pi}{25}\right) & , \quad k < 250 \\ 1 & , 250 \le k < 500 \\ -1 & , 500 \le k < 750 \\ 0.3\sin\left(\frac{k\pi}{25}\right) + 0.1\sin\left(\frac{k\pi}{32}\right) + 0.6\sin\left(\frac{k\pi}{10}\right) & , 750 \le k < 1000 \end{cases} \tag{5.17}$$

ET

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + {x_3}^2 + {x_2}^2} \tag{5.18}$$

Ici, la sortie courante du procédé dépend des trois précédentes sorties et de deux précédentes entrées. Cependant, la structure du FWNN possède deux entrées $y(k)$ et $u(k)$; et une seule sortie $y(k+1)$. Trois fonctions d'appartenance gaussiennes sont considérées pour chaque entrée formant ainsi neuf règles. Aussi, neuf WNN sont utilisées.

L'ensemble des données contient 1000 points de référence, 500 sont employés pour l'entrainement et tous l'ensemble des données est employé pour la validation. Pour

l'optimisation, un AG avec les mêmes paramètres qu'indiqués dans le tableau 5.1 est employé pour maximiser une fonction d'adaptation donnée par :

$$f = \frac{\alpha}{1 + \beta J} \tag{5.19}$$

Avec $\alpha = 100\ et\ \beta = 1$

Et J l'erreur quadratique moyenne avec k=500:

$$J = \frac{\sum_{i=3}^{k} (y_d(i+1) - y(i+1))^2}{k} \tag{5.20}$$

La figure (5.6) montre l'évolution des valeurs minimales de la MSE à travers 100 générations. Après 100 générations, la valeur de la meilleure fonction d'adaptation était 99.9106 correspondant à la valeur 8.9474.10-4 de l'EQM. Les figures (5.7) et (5.8) montrent le comportement du FWNN en comparaison avec le processus réel, il est clair de voir que la sortie du FWNN est assez proche de la sortie désirée.

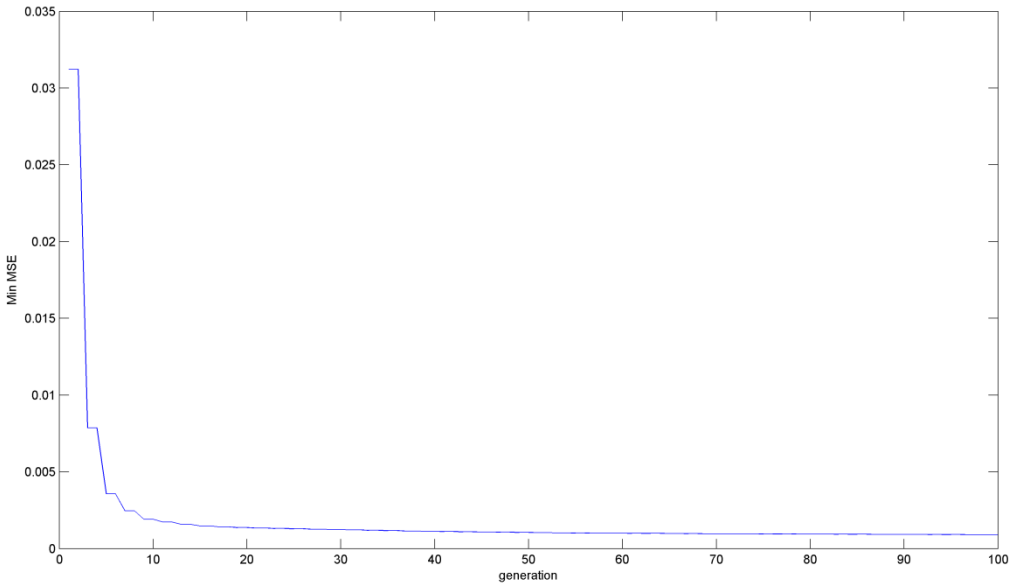


Fig.5.6: Valeurs de l'EQM dans la phase d'entrainement

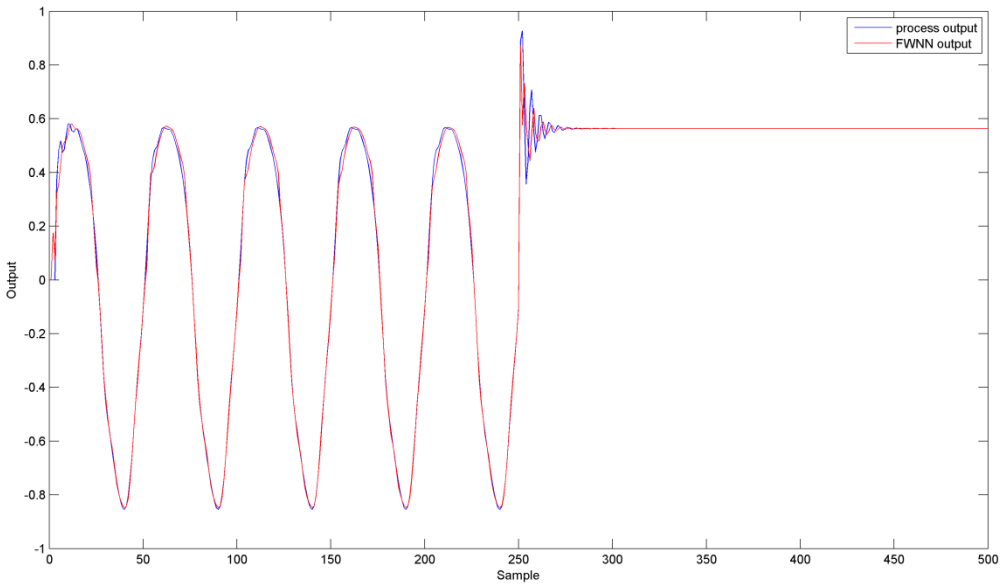


Fig.5.7 : Résultats d'entrainement de l'identification

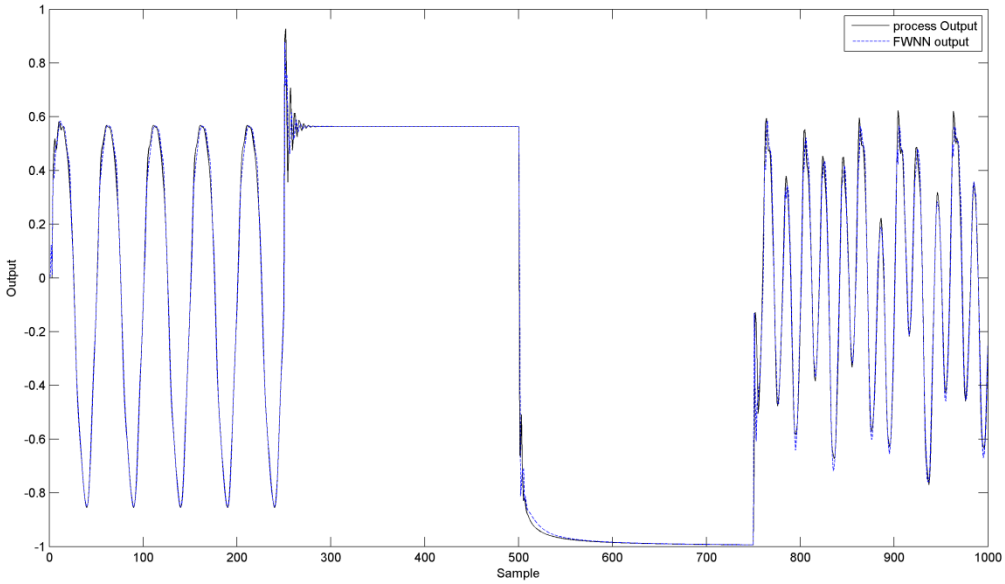


Fig.5.8: Résultats de validation de l'identification

5.5.2 Application au contrôle des systèmes

Dans cette partie, nous étudions l'utilisation de l'approche GA-FWNN développée, pour la commande des systèmes [141]. Nous adoptons la structure de la commande illustrée dans la figure (5.9).

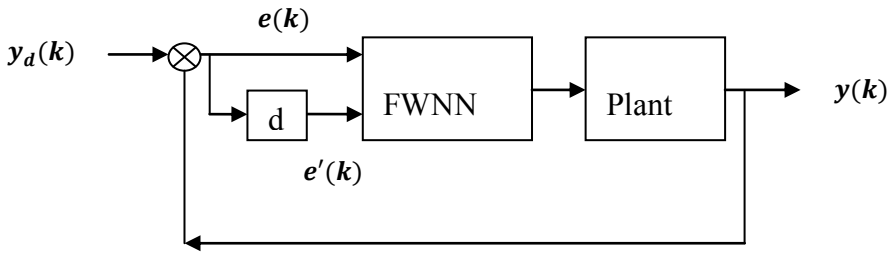


Fig. 5.9: Structure de contrôle

Ici, le signal de sortie du procédé est noté par $y(k)$, $y_d(k)$ représente le signal de référence, $e(k)$ et $e'(k)$ représentent respectivement l'erreur et la variation de l'erreur.

5.5.2.1 Exemple 1

Le système dynamique non linéaire à contrôler par la structure GA-FWNN est décrit par (133):

$$y(k) = \frac{y(k-1)y(k-2)(y(k-1)+2.5)}{\left(1+y^2(k-1)+y^2(k-2)\right)} + u(k) \tag{5.21}$$

Le contrôleur FWNN possède deux entrées l'erreur $e(k)$ et la variation de l'erreur $e'(k)$ et une sortie $u(k)$. Trois fonctions d'appartenance gaussiennes sont considérées pour chaque entrée formant ainsi neuf règles. Par conséquent, il y a 12 paramètres à ajuster dans la partie antécédente des règles. En outre, neuf WNN sont employés pour réaliser la partie conséquente des règles, résultant à 54 paramètres à ajuster. Le nombre de données considéré pour l'entrainement est 2000. Les paramètres de l'AG sont :

Génération = 200;

Taille de la population = 100;

Probabilité de croisement = 0.7;

Probabilité de mutation = 0.01

La fonction d'adaptation à maximiser par l'AG est donnée par:

$$f = \frac{\alpha}{1+\beta J} \ , avec \ \ \alpha = 100 \ et \ \beta = 1 \tag{5.22}$$

Avec J est critère de performance, la racine de l'erreur quadratique moyenne (RMSE) est utilisé avec k=2000:

$$J = \sqrt{\frac{\sum_{i=3}^{k}\left(y_d(i)-y(i)\right)^2}{k}} \tag{5.23}$$

Le résultat de l'entrainement présente une valeur du RMSE égale à 0.2122. Les figures (5.10) et (5.11) montrent les caractéristiques temporelles du système de contrôle de la phase d'entrainement et de la phase de validation respectivement, on constate que le contrôleur GA-FWNN réalise une très bonne performance de contrôle.

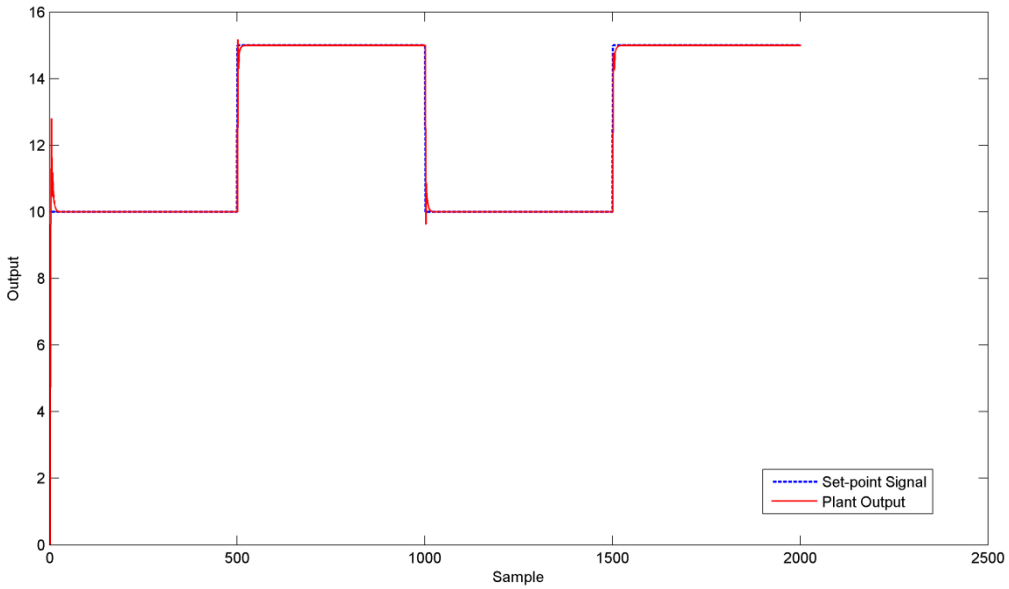


Fig. 5.10: caractéristique de la réponse temporelle du système de contrôle GA-FWNN (phase d'entrainement)

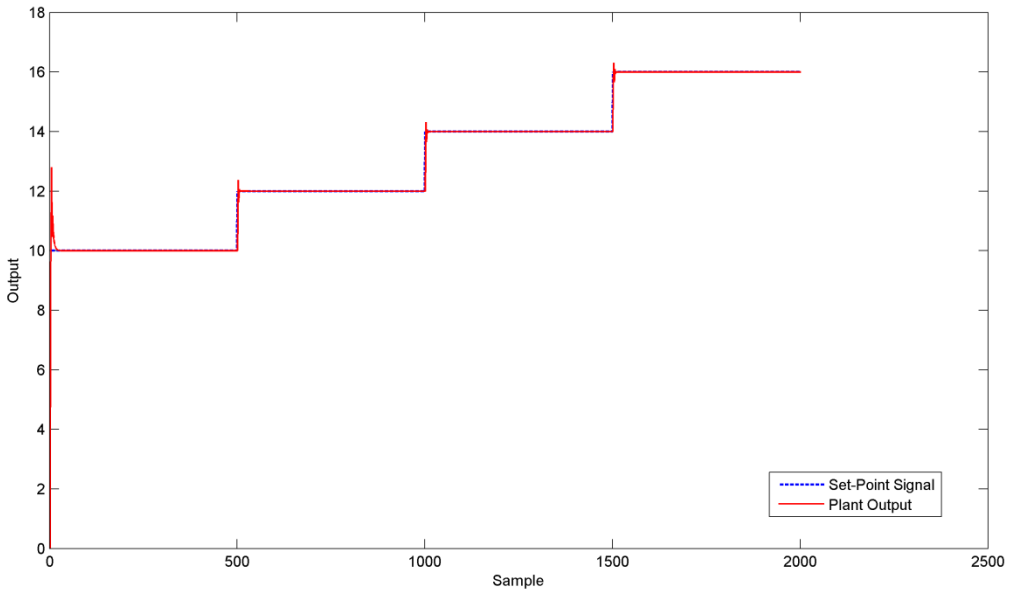


Fig. 5.11: Caractéristique de la réponse temporelle du système de contrôle GA-FWNN (phase de validation)

5.5.2.2 Exemple 2

Le deuxième système à contrôler est décrit par (133) :

$$y(k) = 0.72y(k-1) + 0.025y(k-2)u(k-1) + 0.01u^2(k-2) + 0.2u(k-3) \quad (5.24)$$

Dans ce cas, la sortie actuelle du système dépend de deux sorties précédentes et de trois entrées précédentes. Cependant, la structure du contrôleur GA-FWNN possède uniquement 2 entrées $e(k)$ et $e'(k)$; et une sortie $u(k)$. Identique à la structure précédente, trois fonctions d'appartenance gaussienne sont considérées pour chaque entrée, formant ainsi neufs règles. Aussi, neufs WNN sont utilisées. Le nombre des données d'entrainement est de 2000 valeurs. Pour l'optimisation, un AG avec les mêmes paramètres considérés dans l'exemple précédent est utilisé pour maximiser la fonction d'adaptation donnée par:

$$f = \frac{\alpha}{1+\beta J}, avec \ \alpha = 100 \ et \ \beta = 1 \quad (5.25)$$

Avec J est la racine de l'erreur quadratique moyenne (RMSE) et k=2000:

$$J = \sqrt{\frac{\sum_{i=4}^{k}\left(y_d(i) - y(i)\right)^2}{k}} \quad (5.26)$$

Les résultats d'entrainement donnent un RMSE égale à 0.2567. Les figures (5.12) et (5.13) montrent les performances du système de contrôle GA-FWNN pour la phase d'entrainement et pour la phase de validation respectivement.

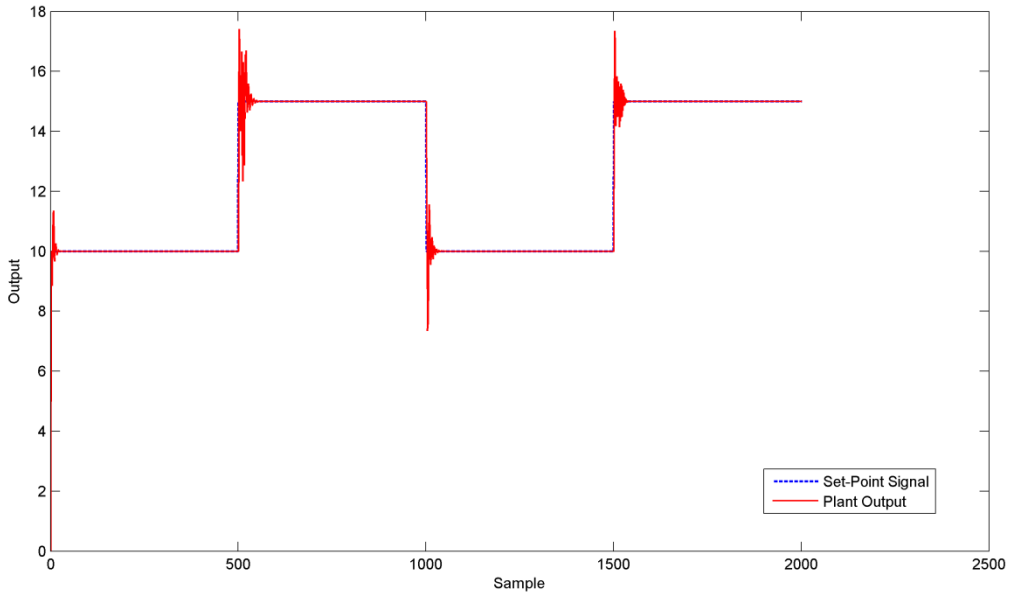


Fig. 5.12: caractéristique de la réponse temporelle du système de contrôle GA-FWNN (phase d'entrainement)

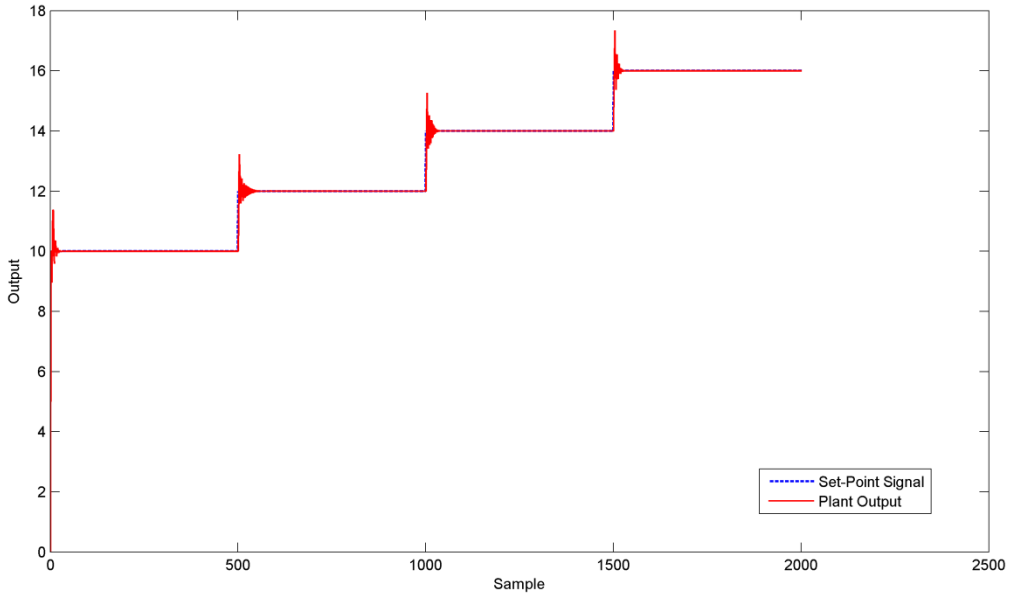


Fig. 5.13: Caractéristique de la réponse temporelle du système de contrôle GA-FWNN (phase de validation)

5.6 Conclusion

L'approche proposée dans ce chapitre, à savoir GA-FWNN, combine plusieurs techniques du soft computing tel que les systèmes flous Takagi-Sugeno (TS), les réseaux d'ondelettes et les algorithmes génétiques. La structure du réseau d'ondelette flou adoptée consiste en une combinaison de deux parties ; une partie contenant le réseau d'ondelettes et une partie implémentant le mécanisme du raisonnement flou du type Takagi-Sugeno.

La conception d'un tel réseau nécessite l'optimisation de certains paramètres caractérisant le réseau, à savoir, les paramètres des fonctions d'appartenance gaussiennes (les centres et les largeurs), les translations, les dilatations et les poids de pondération. Vue la complexité du problème d'optimisation, une méthode basée sur les algorithmes génétiques a été utilisée.

Cette approche a été testée en simulation, d'une part, à l'identification des systèmes dynamiques non linéaires, deux exemples issus de la littérature ont été étudiés. Les résultats obtenus sont très satisfaisants, et montrent que cette approche possède une très bonne vitesse d'apprentissage et offre des modèles très précis. D'autre part, à la commande de deux systèmes dynamiques non linéaires. Les résultats de simulation démontrent que les contrôleurs obtenus peuvent converger rapidement, s'adaptent bien face aux nouvelles données et présentent une erreur quadratique assez faible.

Conclusion

Plusieurs techniques de conception des SIF à partir des données numériques sont disponibles. Le problème général consiste à déterminer le nombre adéquat de règles, à placer les fonctions d'appartenance sur l'espace d'entrée et/ou de sortie et à déterminer les conclusions des règles. Cependant, lors du développement des SIF à partir des données numériques, deux aspects importants sont à considérer : l'interprétabilité et la précision.

Dans la première approche proposée dans cette thèse, un contrôleur Neuro-flou (CNF) est adopté pour implémenter un SIF du type Mamdani. Le but de notre méthode est non seulement d'assurer la précision appropriée du contrôleur, mais d'assurer également l'interpretabilité des règles floues. Ainsi, le problème d'optimisation est posé comme un problème multi-objectif dont la solution est obtenue par un algorithme évolutionnaire amélioré dit MBR-NSGAII (Mixed Binary-Real Non dominated Sorting Genetic Algorithm II).

Cet algorithme est utilisé pour trouver les valeurs optimales des fonctions d'appartenance des variables d'entées/sorties et la base des règles floues modélisées par des poids binaires assujettis à certaines contraintes afin de préserver l'interprétabilité des règles floues durant le processus d'optimisation. Du à la complexité du problème d'optimisation, les opérateurs génétiques ont été améliorés.

Pour vérifier la performance de cette approche, deux exemples de contrôle ont été considérés : contrôle du système du pendule inversé et contrôle décentralisé multivariable d'un simulateur d'hélicoptère. Dans les deux cas, le processus d'optimisation a produit des CNF compacts avec une haute précision et robustesse et une très bonne interprétabilité.

Dans la deuxième approche, une méthode à base d'algorithmes génétiques (AG) pour la conception des réseaux d'ondelettes floues (GA-FWNN) a été proposée. La structure du réseau d'ondelette flou adoptée consiste en une combinaison de deux parties ; une partie contenant le réseau d'ondelettes et une partie implémentant le mécanisme du raisonnement flou du type Takagi-Sugeno.

L'optimisation des paramètres est le problème principal dans la conception des FWNN. Ainsi, un algorithme génétique à codage réel est utilisé pour obtenir les valeurs optimales de la partie antécédente des règles: les centres et les largeurs des fonctions d'appartenance gaussiennes; et de la partie conséquence des règles : translation et dilatation et les poids des fonctions ondelettes.

Cette approche a été testée en simulation, d'une part, à l'identification des systèmes dynamiques non linéaires, deux exemples issus de la littérature ont été étudiés. Les résultats obtenus sont très satisfaisants, et montrent que cette approche possède une très bonne vitesse d'apprentissage et offre des modèles très précis. D'autre part, à la commande de deux systèmes dynamiques non linéaires. Les résultats de simulation démontrent que les contrôleurs obtenus peuvent converger rapidement, s'adaptent bien face aux nouvelles données et présentent une erreur quadratique assez faible.

Cependant, la possibilité d'utiliser d'autres méthodes évolutionnaires (à objectif unique ou multi-objectif) tels que les PSO, ACO, DE,..., avec quelques améliorations, s'avère très intéressant comme un travail future.

Production scientifique

(1) **F.Titel**, K.Belarbi," ***A Mixed Binary-Real NSGA II Algorithm Ensuring both Accuracy and Interpretability of a Neuro-Fuzzy Controller*** ", International Journal of Electrical and Computer Engineering **(IJECE),** Vol. 7, No. 5, pp. 2614~2626, October 2017.

(2) **F.Titel**, K.Belarbi," ***Genetic Algorithm-based fuzzy wavelet neural network for Control of Dynamic Plants***", proceedings of the international conference on control, Engineering and Information Technology (CEIT'2014), Monastir, Tunisia, , March 22-25, 2014. PET Journal-IPCO-2014, pp. 98-102. ISSN 2356-5608.

Cette communication a été publiée dans un journal :

F.Titel, K.Belarbi," ***Genetic Algorithm-Based Fuzzy Wavelet Neural Network for Control of Dynamic Plants*** ", International Journal of Computer Science, Communication & Information Technology **(CSCIT)**, Vol.3, Issue 2, pp1-5, 2016.

(3) **F.Titel**, K.Belarbi," ***Identification of dynamic systems using a Genetic Algorithm-based fuzzy wavelet neural network approach***", proceedings of the *3rd IEEE* international conference on systems and control, ICSC'2013, Algiers, Algeria, , pp.*619-624,*October 29-31, 2013.

(4) **F.Titel**, K.Belarbi, "***Optimal Design of a fuzzy neural network using a multiobjective genetic Algorithm***", Proceedings of the 2nd International Conference on systems and control, ICSC2012, Marrakech, Morocco, pp.232-237, June 20-22, 2012.

(5) K.Belarbi, **F.Titel**, W.Bourebia, K.Benmahammed, "***Design of Mamdani fuzzy logic controllers with rule base minimization using genetic Algorithm***", Journal of Engineering applications of artificial intelligence, Vol.18,Issue7 pp.875-880, October 2005.

(6) K.Belarbi and **F.Titel**, "***Genetic algorithm for the design of a class of fuzzy controllers: An alternative approach*** ", IEEE transactions on fuzzy systems.Vol.8, N°4, pp.398-405, August 2000.

Bibliographie

[1] S. Guillaume, ″Designing Fuzzy Inference systems from data: An Interpretability-Oriented Review,″ IEEE Trans. Fuzzy Systems, Vol.9, N°. 3, pp.426-443, June 2001.

[2] H. Ishibuchi and Y. Nojima, "Discussions on Interpretability of Fuzzy Systems using Simple Examples," Proc. of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference (IFSA/EUSFLAT 2009), July 2009, pp. 1649-1654.

[3] S.M. Zhou and J. Q. Gan, "Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling," Fuzzy Sets and Systems, vol. 159, pp. 3091-3131, Dec. 2008.

[4] F. Afsari, M. Eftekhari, E. Eslami, and P.-Y. Woo, "Interpretability-based fuzzy decision tree classifier a hybrid of the subtractive clustering and the multi-objective evolutionary algorithm," Soft Computing, vol. 17, pp. 1673-1686, Sep.2013.

[5] R. Ishibashi and C. L. Nascimento, "Knowledge extraction using a genetic fuzzy rule-based system with increased interpretability," Proc. IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI 2012), IEEE press, Jan.2012, pp. 247-252.

[6] C.-F. Juang and C.-Y. Chen, "Data-driven interval type-2 neural fuzzy system with high learning accuracy and improved model interpretability," IEEE transactions on cybernetics, vol. 43, pp. 1781-1795, Dec. 2013.

[7] M. I. Rey, M. Galende, M. Fuente, and G. I. Sainz-Palmero, "Checking orthogonal transformations and genetic algorithms for selection of fuzzy rules based on interpretability-accuracy concepts," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 20, pp. 159-186, Oct.2012.

[8] J. Casillas, O. Cordón, M. J. Del Jesus, and F. Herrera, "Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction," IEEE Transactions on Fuzzy Systems, vol. 13, pp. 13-29, Feb.2005.

[9] K. Cpałka, K. Łapa, A. Przybył, and M. Zalasiński, "A new method for designing neuro-fuzzy systems for nonlinear modelling with interpretability aspects," Neurocomputing, vol.135, pp. 203-217, July 2014.

[10] A. Sarkheyli, A. M. Zain, and S. Sharif, "Robust optimization of ANFIS based on a new modified GA," Neurocomputing, vol. 166, pp. 357-366, Oct.2015.

[11]M. J. Gacto, R. Alcalá, and F. Herrera, "Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems," Soft Computing, vol. 13, pp. 419-436, Mar.2009.

[12]M.Galende-Hernández, G. I. Sainz-Palmero, and M. J. Fuente-Aparicio, "Complexity reduction and interpretability improvement for fuzzy rule systems based on simple interpretability measures and indices by bi-objective evolutionary rule selection," Soft Computing, vol. 16, pp. 451-470, Mar.2012.

[13] H. Wang, S. Kwong, Y. Jin, W. Wei, and K.-F. Man, "Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction," Fuzzy sets and systems, vol. 149, pp. 149-186, Jan.2005.

[14]A. Botta, B. Lazzerini, F. Marcelloni, and D. C. Stefanescu, "Context adaptation of fuzzy systems through a multi-objective evolutionary approach based on a novel interpretability index," Soft Computing, vol. 13, pp. 437-449, Mar.2009.

[15] P. K. Shukla and S. P. Tripathi, "A review on the interpretability-accuracy trade-off in evolutionary multi-objective fuzzy systems (EMOFS)," Information, vol. 3, pp. 256-277, July 2012.

[16] M. Antonelli, P. Ducange, B. Lazzerini, and F. Marcelloni, "Multi-objective evolutionary learning of granularity, membership function parameters and rules of Mamdani fuzzy systems," Evolutionary Intelligence, vol. 2, pp. 21-37, Nov.2009.

[17] R. Alcalá, Y. Nojima, F. Herrera, and H. Ishibuchi, "Multiobjective genetic fuzzy rule selection of single granularity-based fuzzy classification rules and its interaction with the lateral tuning of membership functions," Soft Computing, vol. 15, pp. 2303-2318, Dec.2011.

[18] L.X. Wang, "Fuzzy systems are universal approximators," Proc. 1st IEE Conf. On fuzzy systems, pp.1163-1169, San Diego, 1992.

[19]Cassilas, J., Cordon, O. and Herrera, F. "Interpretability Improvements in Linguistic Fuzzy Modelling", Springer, Heidelberg, Germany, 2003.

[20] Gacto, M.J., Alcala, R. and Herrera, F. "Interpretability of linguistic fuzzy rule based systems: an overview of interpretability measures", Information Sciences, Vol. 181,No. 20, pp.4340–4360, 2011.

[21] Shukla, P.K. and Tripathi, S.P. "Interpretability issues in evolutionary multi-objective fuzzy knowledge base systems", J.C. Bansal (Eds.): Proceedings of 7th International Conference on Bio-Inspired Computing: Theories and Applications (BICTA-2012), Advances in Intelligent Systems and Computing, Vol. 201, pp.473–484.2013.

[22] J. Casillas, O. Cordón, M. J. del Jesús, F. Herrera, "Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction", *IEEE Transactions on Fuzzy Systems*, vol. 13, no. 1, pp. 13-29, 2005.

[23] Cassilas, J., Cordon, O., Herrera, F. and Magdalena, L. "Accuracy Improvements in Linguistic Fuzzy Modelling", Springer, New York, USA. 2003.

[24]E.H. Mamdani, "Applications of fuzzy algorithms for control of a simple dynamic process," Proc. IEE, Vol.121, pp.1585-1588, 1974.

[25] T.Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control, " IEEE Trans. On Syst. Man, Cybern., SMC-15(1), January 1985.

[26] Y. Jin, W.V. Seelen and B. Sendhoff, "On generating FC^3 fuzzy rule systems from data using evolution strategies, " IEEE Trans. Syst. Man, Cybern. - Part B: Cybernetics, Vol.29,N°6, pp.829-845, Dec.1999.

[27]Y. Jin, "Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement, " IEEE Transactions on fuzzy systems, Vol.8, N°2, pp.212-221, April 2000.

[28] I. Rojas, H. Pomares, J. Ortega and A. Prieto, "Self-organized fuzzy system generation from training examples, " IEEE trans. Fuzzy Syst. Vol.8, pp.23-26, Feb.2000.

[29] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy If-then rules for classification problems using genetic algorithms, " IEEE Trans. Fuzzy Syst., Vol.3, pp.260-270, Aug. 1995.

[30] L.X. Wang and J.M. Mendel, "Generating fuzzy rules by learning from examples, "IEEE Trans. Syst., Man, Cybern., Vol.22, pp.1414-1427, 1992.

[31] R. Krishnapuram and J. Kim , " A note on the Gustafson-Kessel and adaptive fuzzy clustering algorithms, " IEEE Trans. Fuzzy syst., Vol.7, pp.453-461, Aug.1999.

[32] E. Kim, M. Park, S. Kim, and M. Park, "A transformed input-domain approach to fuzzy modeling, " IEEE Trans. Fuzzy Syst., Vol.6, pp.596-604, Nov.1998

[33] X. Xie and G. Beni, "A validity measure for fuzzy clustering, " IEEE Trans. Pattern Anal. Machine intell., Vol.13, pp.841-847, Aug.1991.

[34] S.L. Chiu, " Fuzzy model identification based on cluster estimation, " J. Intell. Fuzzy Syst., Vol. 2, pp.267-278, 1994.

[35] J. Buckley and Y. Hayashi, "Fuzzy neural networks: a survey, " Fuzzy sets and systems 66 (1994), pp.1-13.
[36]W. Pedrycz , "Fuzzy neural networks and neurocomputations, " Fuzzy Sets and Systems,1993 , 56, pp.1-28 .
[37] C.H Lee and C.C. Teng, " Identification and control of dynamic systems using recurrent fuzzy neural networks, " IEEE Trans. On Fuzzy Syst., Vol.8, N°.4, pp.349-366, Aug.2000.
[38]J. Zhang and J. Morris,"Recurrent neuro-fuzzy networks for nonlinear process modeling," IEEE Trans. On neural networks, Vol.10, N°2, pp.313-326, March 1999.
[39] J.R. Jang, "Self-learning fuzzy controllers based on temporal back propagation, " IEEE Trans. On neural networks, Vol.3, N°5,pp.714-723, Sept. 1992.
[40] J.-S.R. Jang and C.-T. Sun, "Functional equivalence between radial basis function network and fuzzy inference systems, " IEEE Trans. Neural Net., Vol.4, pp.156-159, 1993.
[41] Y. Hayashi, J.J. Buckley and E. Czogala , " Systems engineering applications of Fuzzy neural networks, "Proc. of internat. joint conf. on Neural Networks ,Baltimore ,MD,1992 Vol. 2 ,pp. 412-418 .
[42] C-T. Lin and Y-C. Lu, " A neural fuzzy system with fuzzy supervised learning, " IEEE Trans. On Syst. Man and Cybernetics- Part B: Cybernetics, Vol.26, N°5, pp.744-763, October 1996.
[43] J.R. Jang, "Self-learning fuzzy controllers based on temporal back propagation, " IEEE Trans. On neural networks, Vol.3, N°5,pp.714-723, Sept. 1992.
[44] D.E.Goldberg , "Algorithmes génétiques exploration ,optimisation et apprentissage automatique, " Edition Addison-Wesley 1994.
[45] A. Homaifar and Ed McCormick , " Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms, " IEEE Trans. on Fuzzy Systems, Vol. 3, N^{o} 2, pp.129-139, May 1995.
[46] J. Kim and B.P. Zeigler , " Designing Fuzzy logic controllers using a multiresolutional search paradigm, " IEEE Trans. on Fuzzy Systems, Vol. 4, No 3, pp.213-226, August 1996.
[47] D.A. Linkens and H.O. Nyongesa ,"Genetic Algorithms for Fuzzy control-Part 1 :Off-line system development and application, " IEE Proceedings-Control Theory and applications Vol. 142, N^{o} 3, pp.161-176, May 1995.
[48]K. Belarbi and F.Titel, "Genetic algorithm for the design of a class of fuzzy controllers: An alternative approach, " IEEE Trans. Fuzzy Syst. Vol.8, N°4, pp.398-405, Aug.2000.
[49] C.L. Karr, "Design of an adaptive Fuzzy logic controller using a genetic algorithm, " Proc. of the fourth Int. conf. on Genetic Algorithms, pp.450-457, 1991.
[50]C.L. Karr and E.J. Gentry, " Fuzzy control of pH using genetic algorithms, " IEEE Trans.on Fuzzy Systems ,Vol. 1 ,N^{o} 1, pp.46-53, February 1993.
[51] P. Thrift , " Fuzzy logic synthesis with genetic algorithms, " Proc. of the fourth internat. conf. On Genetic Algorithms, pp.509-513, 1991.
[52] A. Ebadat, N. Noroozi, AA. Safavi, and SH. Mousavi, " New fuzzy wavelet network for modeling and control:The modeling approach," Commun Nonlinear Sci, vol. 16, pp. 3385-3396, 2011.
[53] J. Cordova W. Yu, "Two types of haar wavelet neural networks for nonlinear system identification", Neural Process Lett,35, 283-300, 2012.
[54] Karatepe E, and Alc M, " A new approach to fuzzy wavelet system modeling," Int J Approx Reason, vol. 40, pp. 302-322, 2005

[55] M. Zekri, S. Sadri, and F. Sheikholeslam, "Adaptive fuzzy wavelet network control design for non linear systems," Fuzzy Sets Syst, vol. 159, pp. 2668-2695, 2008
[56]ST. Tzeng, "Design of fuzzy wavelet neural networks using the GA approach for function approximation and system identification," Fuzzy Sets Syst, vol. 161, pp. 2585-2896, 2010
[57] J. Cao, Z. Lin, and G. Huang, "Composite function wavelet neural networks with differential evolution and extreme learning machine," Neural Process Lett, vol. 33, pp. 251-265, 2011
[58] HL. Wei, SA. Billings, YG. Zhao, and LZ. Guo, "An adaptive wavelet neural network for spatiooral system identification," Neural Networks, vol. 23, pp. 1286-1299, 2010
[59] Hojjat Allah Bazoobandi, Mahdi Eftekhari, "Comparing Different Versions of Differential Evolution for Training Fuzzy Wavelet Neural Network", Iranian Conference on Intelligent Systems (ICIS), 2014.
[60] H. Nomura, I. Hayashi, N. Wakami, "A self-tuning method of fuzzy control, " Proc. Of 4th IFSA congress, Brussels, 1991.
[61] P.-Y. Glorennec, "Algorithmes d'apprentissage pour systèmes d'inférence floue, " Paris, France: Hermès, 1999.
[62] C.L. Karr, "Design of an adaptive Fuzzy logic controller using a genetic algorithm," Proc. of the fourth Int. conf. on Genetic Algorithms, pp.450-457, 1991.
[63]C.L. Karr and E.J. Gentry, " Fuzzy control of pH using genetic algorithms, " IEEE Trans.on Fuzzy Systems ,Vol. 1 ,N° 1, pp.46-53, February 1993.
[64] D.A. Linkens and H.O. Nyongesa ,"Genetic Algorithms for Fuzzy control-Part 2 :On-line system development and application, " IEE Proceedings-Control Theory and applications Vol. 142, N° 3, pp.177-185, May 1995.
[65]Pratihar, D.K., Deb, K., Ghosh, A.: A genetic-fuzzy approach for mobile robot navigation among moving obstacles. International Journal of Approximate Reasoning 20(2), 145–172 (1999)
[66] Mucientes, M., Moreno, D.L., Bugarin, A., Barro, S.: "Design of a fuzzy controller in mo bile robotics using genetic algorithms". Applied Soft Computing 7, 540–546 (2007)
[67] Wayan Firdaus Mahmudy, Agus Naba, "Rule Optimization of Fuzzy Inference System Sugeno using Evolution Strategy for Electricity Consumption Forecasting" , International Journal of Electrical and Computer Engineering (IJECE) Vol. 7, No. 4, August 2017, pp. 2241~2252
[68] Kennedy, J., Eberhart, R.: "Particle swarm optimization. In: Proceedings of 1995 IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)
[69] Clerc, M., Kennedy, J.: "The particle swarm - explosion, stability, and convergence in a multidimensional complex space". IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
[70] Karakuzu, C., "Fuzzy controller training using particle swarm optimisation for nonlinear system control", ISA Transactions 47, 229–239 (2008)
[71]Mukherjee, V., Ghoshal, S.P. "Intelligent particle swarm optimized fuzzy PID controller for AVR system. Electric Power Systems Research 77, 1689–1698 (2007)
[72]Lin, C., Hong, S. "The design of neuro-fuzzy networks using particle swarm optimisation and recursive singular value decomposition. Neurocomputing 71, 271–310 (2007)
[73]Esmin, A.A.A., Lambert-Torres, G. "Fitting fuzzy membership functions using hybrid particle swarm optimization", In: Proceedings on 2006 IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, pp. 2112–2119 (2006)
[74]Fang, G., Kwok, N.M., Ha, Q.P. "Automatic Fuzzy Membership Function Tuning Using the Particle Swarm Optimisation". In: IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA 2008), Wuhan, China, vol. 2, pp. 324–328 (December 2008)

[75] Gu Fang1, Ngai Ming Kwok2, and Dalong Wang2, F.L. Wang et al. (Eds.) "Automatic Rule Tuning of a Fuzzy Logic Controller Using Particle Swarm Optimisation: AICI 2010, Part II, LNAI 6320, pp. 326–333, 2010. Springer-Verlag Berlin Heidelberg 2010

[76]Ali, S.F., Ramaswamy, A. "Optimal fuzzy logic controller for MDOF structural systems using evolutionary algorithms", Engineering Applications of Artificial Intelligence (22), 407–419 (2009)

[77]M. Dorigo and C. Blum, "Ant colony optimization theory: a survey, " Theoretical Computer Science, vol. 344, no. 2-3, pp. 243–278, November 2005.

[78] J. van Ast, R. Babuska, and B. De Schutter, "Fuzzy ant colony optimization for optimal control, " Proceedings of the 2009 American Control Conference, St. Louis, Missouri, pp. 1003–1008, June 2009.

[79] J. Casillas, O. Cord´ on, and F. Herrera, "Learning fuzzy rule-based systems using ant colony optimization algorithms," in Proceedings of the ANTS'2000. From Ant Colonies to Artificial Ants: Second Interantional Workshop on Ant Algorithms. Brussels (Belgium), September 2000, pp. 13–21.

[80] B. Zhao and S. Li, "Design of a fuzzy logic controller by ant colony algorithm with application to an inverted pendulum system, " in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics,2006, pp. 3790–3794.

[81] W. Zhu, J. Chen, and B. Zhu, "Optimal design of fuzzy controller based on ant colony algorithms, " in Proceedings of the IEEE International Conference on Mechatronics and Automation, 2006, pp. 1603–1607.

[82] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, " Journal of Global Optimization, vol. 11, Dec.1997, pp. 341-359.

[83]S. Das, P. N. Suganthan, "Differential evolution—A survey of the state-of-the-art", *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4-31, Feb. 2011.

[84] Shuai L., Wei S., "Design of Fuzzy Logic Controller Based on Differential Evolution Algorithm", Computational Intelligence, Networked Systems and Their Applications. LSMS/ICSEE 2014. Communications in Computer and Information Science, vol 462. Springer, Berlin, Heidelberg

[85] France Cheong, Richard Lai. "Designing a hierarchical fuzzy logic controller using the differential evolution approach. "Applied Soft Computing 7. 2007: 481–491.

[86] LI Shuai, SUN Wei, "Optimization of Fuzzy Control Rules based on Differential Evolution Algorithm, " Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference August 8-10, 2014, Yantai, China,pp.2610-2613

[87] M. Gong, L. Jiao, D. Yang, W. Ma, "Research on evolutionary multi-objective optimization algorithms, " Journal Software 20 (2) (2009) 271–289.

[88]C.A. Coello Coello, "Recent trends in evolutionary multiobjective optimization," in: Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, Springer-Verlag, 2005, pp. 7–32.

[89]B.Y. Qu, P.N. Suganthan, "Multi-objective evolutionary algorithms based on the summation of normalized objectives and diversified selection," Information Sciences 180 (17) (2010) 3170–3181.

[90] C. Shi, Z. Yan, Z. Shi, L. Zhang, "A fast multi-objective evolutionary algorithm based on a tree structure, " Applied Soft Computing 10 (2) (2010) 468–480.

[91] Aimin Zhoua,∗, Bo-Yang Qub, Hui Lic, "Multiobjective evolutionary algorithms: A survey of the state of the art" , Swarm and Evolutionary Computation, Volume 1,Issue 1, March 2011, Pages 32-49.

[92] D.E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multi-modal function optimization, " in Proc. 2nd Int. Conf. Genetic algorithms, pp.41-49, 1987.
[93] J. Horn and N. Nafpliotis, "Multiobjective optimization Using the niched pareto genetic algorithm," Univ. Illinois at Urbana-Champaign, Urbana, Il, IlliGAL Rep.93005, 1993
[94] E. Zitzler and L. Thiele, "An evolutionary algorithm for multiobjective optimization: The strength pareto approach," Computer Engineering and communication Network Lab (TIK) Swiss Federal Institute of technology (ETH), Zurich, Switzerland, Tech.Rep.N°43, May 1998.
[95] J.D Schaffer, "Multiple-objective optimization using genetic algorithm," in Proc. 1st Int. Conf. Genetic Algorithms, pp. 93-100, July 1985.
[96] P. Hajela and C.Y. Lin, "Genetic search strategies in multicriterion optimal design," J. Structural Optimization, Vol.4, pp.99-107, June 1992.
[97] D.E.Goldberg , "Algorithmes génétiques exploration ,optimisation et apprentissage automatique, " Edition Addison-Wesley 1994
[98] C.M Fonseca and P. J. Fleming, "Genetic algorithm for multiobjective optimization, formulation, discussion and generalization," In proceedings of the 5th international conference on genetic algorithms (S. Forest, Ed.), San Mateo, CA: Morgan Kaufmann, pp.416-423, July 1993.
[99]N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," Evol. Comput. Vol.2, N° 3, pp. 221-248, 1994
[100] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE transactions on evolutionary computation, vol. 6, pp. 182-197, Apr.2002.
[101] J. Moore, R. Chapman,**"**Application of particle swarm to multiobjective optimization, **"** Tech. Rep., Department of Computer Science and Software Engineering, Auburn University, 1999.
[102] Y.Wang, Y. Yang, **"**Particle swarm optimizationwith preference order ranking for multi-objective optimization, **"** Information Sciences 179 (12) (2009) 1944–1959.
[103] V.L.Huang, P.N. Suganthan, J.J. Liang, **"**Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems**"**, International Journal of Intelligent Systems 21 (2) (2006) 209–226.
[104] N.A.Moubayed, A. Petrovski, J.McCall, **"**A novel smart multiobjective particle swarm optimisation using decomposition, **"** in: Parallel Problem Solving from Nature, PPSN XI, in: LNCS, vol. 6239, 2010, pp. 1–10.
[105] C. A. C. Coello, "An Introduction to Multi-Objective Particle Swarm Optimizers," Soft Computing in Industrial Application, Advances in Intelligent and Soft Computing, vol. 96, pp. 3-12, 2011.
[106] C.A. Coello Coello, G.T. Pulido, M.S. Lechuga, **"**Handling multiple objectives with particle swarm optimization, **"** IEEE Transactions on Evolutionary Computation 8 (3) (2004) 256–279.
[107] W.-F. Leong, G.G. Yen, **"**PSO-based multiobjective optimization with dynamic population size and adaptive local archives**"**, IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) 38 (5) (2008) 1270–1293.
[108] Vipin Kumar and Sonajharia Minz, **"**Multi-Objective Particle Swarm Optimization: An Introduction**"**, Smart Computing Review, vol. 4, no. 5, October 2014
[109] Margarita Reyes-Sierra and Carlos A. Coello Coello, **"**Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art**",** International Journal of Computational Intelligence Research.Vol.2, No.3 (2006), pp. 287–308

[110]D. Angus, "Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem", in: IEEE Symposium on Computational Intelligence in Multicriteria Decision Making, MCDM 2007,2007, pp. 333–340.

[111] C. Garcia-Martinez, O. Cordon, F. Herrera, "A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP", European Journal of Operational Research 127 (1) (2007) 116–148.

[112] D.M. Chitty, M.L. Hernandez, "A hybrid ant colony optimisation technique for dynamic vehicle routing, " in: Conference on Genetic and Evolutionary Computation, GECCO 2004, in: LNCS, vol. 3102, 2004, pp. 48–59.

[113] K.F. Doerner,W.J. Gutjahr, R.F. Hartl, C. Strauss, C. Stummer, " Pareto ant colony optimization with ILP preprocessing in multiobjective project portfolio selection, " European Journal of Operational Research 171 (3) (2006) 830–841.

[114]Barán, B., & Schaerer, M. " A multiobjective ant colony system for vehicle routing problem with time windows. In Proceedings of the 21st IASTED international conference on applied informatics (pp. 97–102).2003.

[115] Ines Alaya , Christine Solnon , Khaled Ghedira , "Ant Colony Optimization for Multi-objective Optimization Problems", 19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2007.

[116] R. Sarker, H. Abbass, **"**Differential evolution for solving multi-objective optimization problems, **"** Asia Pacific Journal of Operational Research 21 (2)(2004) 225–240.

[117] Saha I., Maullik U., Łukasik M., Plewczynski D. "Multiobjective Differential Evolution: A Comparative Study on Benchmark Problems, " Advances in Intelligent Systems and Computing, vol 242.2014,Springer, Cham

[118]Robič T., Filipič B. "DEMO: Differential Evolution for Multiobjective Optimization" Evolutionary Multi-Criterion Optimization. EMO 2005. Lecture Notes in Computer Science, vol 3410. Springer, Berlin, Heidelberg

[119] B. V. Babu, B. Anbarasu , "Multi-Objective Differential Evolution (MODE): An Evolutionary Algorithm for Multi-Objective Optimization Problems (MOOPs),2005.

[120] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," Artificial intelligence review, vol. 12, pp. 265-319, Aug.1998.

[121] User's manual: CE 150 helicopter model, Humusoft, Prague, 2002.

[122]M.thuillard, "A review of wavelet networks, wavenets, fuzzy wavenets and their applications ," ESIT 2000,pp.394-399, Aachen, Germany, September 2000.

[123]S. Postalcioglu, Y. Becerikli, "Wavelet networks for nonlinear system modeling," Neural Computing & Applications, Vol.16, pp. 433–441, May 2007.

[124] Yacine OUSSAR, "Réseaux d'ondelettes et réseaux de neurones pour la modélisation statique et dynamique de processus". Thèse de doctorat en Robotique de l'université pierre et marie curie, 1998,

[125] A. Banakar, M.F. Azeem, "Artificial Wavelet Neural Network and Its Application in Neuro-Fuzzy Models, Applied Soft Computing Journal (2007),

[126] Qingling Zhang, "wavelet networks", IEEE Transactions on Neural Networks,Vol. 3(6):889-98. February 1992

[127]Lu, C.-H. "Wavelet Fuzzy Neural Networks for Identification and Predictive Control of Dynamic Systems ", IEEE Trans. Industrial Electronics, 58(7):3046-3058, 2011.

[128] Linhares, Leandro L.S. , Araújo Jr., José M. ,Araújo, Fábio M.U. , Yoneyama, Takashi "A nonlinear system identification approach based on fuzzy wavelet neural network", Journal of Intelligent & Fuzzy Systems, vol. 28, no. 1, pp. 225-235, 2015

[129] mehrnoosh davanipoor, maryam zekri , farid sheikholeslam, "Fuzzy wavelet neural network with an accelerated hybrid learning algorithm", IEEE Transactions on fuzzy systems, vol. 20, no. 3, june 2012

[130] AfroozEbadat, NavidNoroozi,"New fuzzy wavelet network for modeling and control: The modeling approach, " Communications in Nonlinear Science and Numerical Simulation Volume 16, Issue 8, August 2011, Pages 3385-3396

[131] Yusuf Oysal, Sevcan Yilmaz , "An adaptive fuzzy wavelet neural network with gradient learning algorithm for nonlinear function approximation, 10th IEEE International Conference on Networking, Sensing and Control (ICNSC), 2013

[132]Yilmaz, S., and Oysal, Y. "Fuzzy Wavelet Neural Network Models for Prediction and Identification of Dynamical Systems", IEEE Transactions on Neural Networks, 21(10): 1599-1609. (2010).

[133]Abiyev, R.H., and Kaynak, O. "Fuzzy Wavelet Neural Networks for Identification and Control of Dynamic Plants- A Novel Structure and a Comparative Study", IEEE Transactions on Industrial Electronics, 55(8):3133-3140. (2008)

[134] Rahib Hidayat Abiyev , "Fuzzy Wavelet Neural Network for Control of Dynamic Plants" International journal of computational intelligence vol.1 (2) 2004.

[135] J.-S.R. Jang, " ANFIS: Adaptive-network-based fuzzy inference systems, " IEEE Trans. Syst. Man, Cybern., Vol.23, pp.665-685, 1993.

[136] Wright, A. H. Genetic Algorithm for Real Parameter Optimization. Foundation of Genetic Algorithms, pages 205–218. Morgan Kaufmann.1991.

[137] Zekri, M., Sadri, S., and Sheikholeslam, F. , Adaptive Fuzzy Wavelet Network Control Design for Nonlinear Systems. Fuzzy Sets and systems, 159(20):2668- 2695, 2008.

[138] F.Titel, K.Belarbi," A Mixed Binary-Real NSGA II Algorithm Ensuring both Accuracy and Interpretability of a Neuro-Fuzzy Controller ", International Journal of Electrical and Computer Engineering (IJECE), Vol. 7, No. 5, pp. 2614~2626, October 2017.

[139] K. Yoshida, "Swing-up control of an inverted pendulum by energy-based methods," In: Proceedings of the American Control Conference, San Diego, pp. 4045-4047, 1999.

[140] F.Titel, K.Belarbi," Identification of dynamic systems using a Genetic Algorithm-based fuzzy wavelet neural network approach", proceedings of the 3rd IEEE international conference on systems and control, ICSC'2013, Algiers, Algeria, , pp.619-624,October 29-31, 2013.

[141] F.Titel, K.Belarbi," Genetic Algorithm-based fuzzy wavelet neural network for Control of Dynamic Plants", proceedings of the international conference on control, Engineering and Information Technology (CEIT'2014), Monastir, Tunisia, , March 22-25, 2014. PET Journal- IPCO-2014, pp. 98-102. ISSN 2356-5608.

[142]K.Belarbi, F.Titel, W.Bourebia, K.Benmahammed, "Design of Mamdani fuzzy logic controllers with rule base minimization using genetic Algorithm", Journal of Engineering applications of artificial intelligence, Vol.18,Issue7 pp.875-880, October 2005.