République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A.MIRA-BEJAIA

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Faculté de Technologie
Département Génie Electrique
Laboratoire de Technologie Industrielle et de l'Information LTII

THÈSE

EN VUE DE L'OBTENTION DU DIPLOME DE DOCTORAT

Domaine : Sciences et Technologies Filière :
Automatique Spécialité : Automatique et systèmes

Présentée par
BRIKH Lamine

Thème

Contribution à la conception des systèmes flous de type TSK en utilisant les algorithmes évolutionnaires : Application à la modélisation et la commande des systèmes non-linéaires

Soutenue le : 27 février 2025 ***Devant le Jury composé de :***

Nom et Prénom	**Grade**		
Mr BERRAH Smail	Professeur	Univ. de Bejaia	Président
Mr GUENOUNOU Ouahib	Professeur	Univ. de Bejaia	Rapporteur
Mr LEHOUCHE Hocine	MCA	Univ. de Bejaia	Co-Rapporteur
Mr MENDIL Boubekeur	Professeur	Univ. de Bejaia	Examinateur
Mr LAMAMRA Kheireddine	Professeur	Univ. d'Oum El Bouaghi	Examinateur
Mr BOUBERTAKH Hamid	Professeur	Univ. de Jijel	Examinateur
Mr YAHIAOUI Fatah	MCA	Univ. de Bejaia	Invité
Mr BAKIR Toufik	Professeur	Univ. de Dijon-France	Invité

Année Universitaire : 2024/2025


بسم الله الرحمن الرحيم

Remerciements

Je remercie Dieu le tout puissant de nous avoir donné la force, la volonté et le courage à mettre en lumière ce modeste travail. Je tiens à remercier les membres de jury d'avoir accepté de participer et d'examiner ma thèse en commençant par le Pr BERRAH Smail pour l'honneur qu'il m'a fait en présidant le jury de cette thèse. Egalement mes vifs remerciements s'adressent au Pr MENDIL Boubeker, Pr LAMAMRA Kheureddine et Pr BOUBERTAKH Hamid pour avoir consacré du temps à la lecture de ma thèse en tant qu'examinateurs de ce travail. Mes remerciements les plus vifs à mon Directeur de thèse Pr GUENOUNOU Ouahib, mon Co-Directeur Pr LEHOUCHE Hocine et Pr BAKIR Toufik qui m'ont soutenu tout au long de mon travail.

Je remercie beaucoup mes collègues dans l'équipe techniques avancées de commande TAC du laboratoire de technologie de Technologie Industrielle et de l'Information LTII pour m'avoir aidé et leurs contributions en particulier KACIMI Mohand Akli, YAHIAOUI Fatah et OUAMRI Mohamed Amine. Je remercie toutes ma famille pour leurs confiances et encouragements tout le long de cette thèse notamment dans les moments les plus difficiles, en particulier ma mère et ma femme.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

Acronyme	Signification
TSK	Takagi-Sugeno-Kang
PSO	Particle Swarm Optimization
EQ	Equilibrium optimizer
GLSA	Gravitational Local Search Algorithm
ACO	Ant Colony Optimization
EQM	Erreur quadratique Moyenne
FCM	Fuzzy C-Means
Par	Particle
Vel	Velocity

Introduction générale

Les défis modernes en science et technologie, notamment la gestion de systèmes non linéaires et dynamiques, sont de plus en plus complexes, nécessitant des approches novatrices pour leur modélisation et optimisation. Les systèmes non linéaires présentent des comportements multiformes, rendant leurs analyses traditionnelles insuffisantes et incertaines [1]. Selon [2], ces systèmes sont souvent caractérisés par une grande sensibilité aux conditions initiales et des réponses complexes aux perturbations. De plus, dans [3] les auteurs soulignent que les approches traditionnelles basées sur des modèles linéaires simplifiés ne capturent pas adéquatement la dynamique intrinsèque de ces systèmes. En conséquence, il est crucial d'explorer des méthodes avancées comme les systèmes flous et les algorithmes évolutionnaires pour améliorer la précision des modèles et l'efficacité des solutions. Cette nécessité est également mise en évidence par les travaux de [4], qui démontrent que des approches modernes permettent de mieux gérer l'incertitude et la complexité, offrant des perspectives nouvelles pour surmonter les limites des méthodes conventionnelles.

La logique floue, développée par Lotfi Zadeh en 1965 [5], est une approche puissante pour traiter des informations imprécises et incertaines, en imitant le raisonnement humain. Contrairement aux systèmes traditionnels qui utilisent des variables binaires et des règles strictes, la logique floue utilise des ensembles flous et des règles « Si-Alors » pour modéliser des situations où les informations sont vagues ou incomplètes. Cette flexibilité permet de mieux représenter la complexité des systèmes réels et de gérer des situations où les données sont qualitatives ou ambiguës [6]. En intégrant des concepts de logique floue, les modèles peuvent s'adapter plus

facilement à des environnements dynamiques et offrir des solutions plus proches de la réalité.

Les modèles Takagi-Sugeno-Kang (TSK) sont une extension significative de cette approche floue. Introduits par Takagi et Sugeno en 1985 [7], les modèles TSK combinent des règles floues avec des fonctions linéaires dans les parties conséquentes du modèle, ce qui permet de capturer des relations non linéaires complexes tout en conservant une structure analytique et interprétable [8]. Cette approche hybride est particulièrement efficace pour modéliser des systèmes non linéaires où les relations entre les variables ne peuvent pas être décrites simplement par des équations linéaires. Récemment, les modèles TSK ont bénéficié de l'intégration des techniques d'apprentissage profond et d'optimisation hybride, offrant des avantages significatifs dans diverses applications pratiques. Par exemple, [9] ont démontré que l'utilisation des réseaux de neurones profonds pour améliorer les systèmes TSK permet une meilleure gestion des incertitudes et une précision accrue dans la prédiction de systèmes dynamiques complexes. Leur approche, combinant l'apprentissage profond avec des règles floues, a permis de développer des modèles TSK plus robustes et adaptatifs pour la commande de processus industriels.

En combinant les avantages des réseaux de neurones pour l'extraction de caractéristiques avec la flexibilité des règles floues, les chercheurs ont développé des modèles TSK plus robustes et adaptatifs. Par ailleurs, l'utilisation des algorithmes évolutionnaires, tels que les algorithmes génétiques (GA) [10], le Particle Swarm Optimization (PSO) [11], et d'autres méthodes comme l'Optimisation par Colonies de Fourmis (ACO) [12] et les Algorithmes d'Optimisation par Essaim de Particules Améliorés (E-PSO), a montré des résultats prometteurs pour l'optimisation des paramètres et des structures des modèles TSK. Les chercheurs dans le travail [13] ont illustré comment les algorithmes génétiques peuvent être utilisés pour affiner les paramètres des fonctions d'appartenance floues, tandis que les travaux ([14], [15]) ont démontrés que le PSO peut améliorer les performances des modèles TSK en optimisant simultanément les paramètres flous et les coefficients de sortie linéaires. De plus, l'ACO, [16] a été utilisé pour optimiser les structures des systèmes flous en simulant le comportement des colonies de fourmis, permettant ainsi une exploration plus efficace des espaces de solutions complexes. Les améliorations apportées par

ces algorithmes d'optimisation offrent également une meilleure convergence et une gestion plus fine des paramètres dans les systèmes TSK.

Problématique et approche proposée

Comme évoqué précédemment, la construction des systèmes flous de type TSK est un problème assez complexe à résoudre, car elle comporte deux grandes parties : la définition de sa structure (nombre de fonctions d'appartenance, nombre de règles, etc.) et la détermination du jeu de paramètres définissant le système flou.

Nous abordons la question formulée principalement comme suit : comment concevoir un modèle ou un contrôleur flou de type TSK qui prenne en charge simultanément l'optimisation des fonctions d'appartenance (les paramètres) et des règles floues ?

À cette question, nous avons proposé une approche d'optimisation du système flou Takagi-Sugeno-Kang en utilisant une nouvelle version améliorée de l'optimisation par essaims de particules. Cette nouvelle version introduit deux paramètres adaptatifs découplés dans la fonction de mise à jour de la vitesse. Cet algorithme est appliqué à une structure neuro-floue, créée sous la forme d'un réseau à cinq couches, pour exploiter divers paramètres des fonctions d'appartenance ainsi que ceux des conclusions des règles. Le nouveau PSO amélioré a été utilisé pour évaluer ces paramètres et choisir le meilleur modèle parmi tous les individus de l'essaim. Deux systèmes non linéaires sont utilisés pour démontrer la précision de l'approche.

Structure de la thèse

Dans **le premier chapitre**, nous avons introduit les notions fondamentales de la logique floue, ce qui nous a permis de maîtriser cette approche et de mettre en évidence ses avantages par rapport à la logique booléenne. Par la suite, les différentes structures des systèmes flous ont été présentées, notamment celle de Takagi-Sugeno-Kang, que nous avons utilisée dans notre approche. La fin de ce chapitre propose un aperçu de diverses architectures de systèmes neuro-flou, conçus tant pour la modélisation que pour la commande.

Le deuxième chapitre de cette thèse explore les fondements de l'optimisation, ainsi que les algorithmes évolutionnaires, en mettant l'accent sur leurs capacités à résoudre des problèmes complexes. Les algorithmes évolutionnaires, inspirés des processus naturels d'évolution, sont particulièrement adaptés pour optimiser des

problèmes avec des espaces de recherche vastes et non linéaires. Parmi ces algorithmes, ceux basés sur la population, comme l'Optimiseur d'Équilibre (EQ) [17], la Recherche Locale Gravitationnelle (GLSA) [18], et l'Optimisation par Essaims de Particules, se distinguent par leurs capacités à explorer efficacement l'espace de recherche et à éviter les optima locaux.

Le troixième chapitre se concentre sur l'apprentissage des paramètres des systèmes flous de type TSK, en utilisant des techniques d'optimisation avancées pour ajuster ces paramètres dans des contextes complexes [19]. Les modèles TSK, avec leurs capacités à approximer des comportements non linéaires par des équations linéaires locales, nécessitent des méthodes d'optimisation adaptées pour ajuster un grand nombre de paramètres et améliorer leurs performances . L'optimisation par essaims de particules est choisie pour sa flexibilité et sa capacité à localiser l'optimum global dans des environnements difficiles.

Le quatrième chapitre aborde l'apprentissage de la structure des modèles TSK. Bien que les méthodes Mamdani et TSK soient les principales approches en logique floue, le modèle TSK est particulièrement pertinent pour modéliser des systèmes non linéaires complexes avec une précision accrue. Ce chapitre détaille l'approche hybride proposée pour obtenir des modèles flous de type TSK, offrant à la fois une grande précision et une structure optimale.

CHAPITRE 1

Système à inférence floue

1.1 Introduction

Depuis son introduction en 1965 par le professeur lotfi zadah [5], la théorie des ensembles flous a trouvé des applications dans une grande variété de problèmes : la classification, la modélisation et la commande des systèmes complexes et fortement non-linéaires. En effet, de nombreux systèmes ne se prêtent pas aux approches de modélisation classiques en raison soit du manque de connaissances précises sur le système, du comportement fortement non linéaire, ou des caractéristiques temporelles variables [20]. La théorie des ensembles flous permet de countenir toutes ces difficultés à côté d'autres techniques comme les réseaux de neurones [21] qui sont connus aujourd'hui pour être des approximateurs universels. Tout comme les réseaux de neurones, la logique floue offre la possibilité d'intégrer des informations en provenance de différentes sources [22], mais ce qui fait la particularité de cette théorie est sa similitude au raisonnement humain. Ce qui la rend plus proche à la pensée purement humaine la rendant l'outil capable de transférer la connaissance humaine aux machines et inversement.

Un modèle flou est un modèle logique utilisant des règles causales de type « Si-Alors ». Ces règles sont formulées avec des ensembles flous dits d'entrée et d'autres dits de sortie. Les ensembles d'entrée sont une interprétation linguistique des variables numériques réelles issues du système étudié tandis que les ensembles de sortie. Cette nouvelle façon de formuler les règles logiques à base des ensembles flous, rend l'assimilation de l'expertise du modèle plus facile humainement. On parle alors de modèles transparents car ils peuvent être lus et exprimés en langage

naturel. Au niveau du calcul, les modèles flous peuvent être considérés comme des structures mathématiques flexibles, qui peuvent rapprocher une grande classe de systèmes non linéaires complexes à un degré de précision souhaité.

Selon la structure des règles floues, et surtout leurs formulation, on peut distinguer trois types de modèle flou :

— Le modèle linguistique flou ([23], [24]), où l'antécédent et les conséquences sont des propositions floues.

— Modèle relationnel flou ([25], [26]), qui peut être considéré comme une généralisation du modèle linguistique, permettant d'associer une proposition antérieure particulière à plusieurs propositions différentes via une relation floue.

— Modèle flou de Takagi-Sugeno-Kang [7], où le conséquent est une fonction nette des variables antérieures plutôt qu'une proposition floue.

1.2 Théorie des ensembles flous

Par la notion d'ensemble flou, Zadeh introduit un caractère graduel de l'appartenance d'un individu à un ensemble donné. Cela permet une meilleure représentation des termes et des connaissances vagues que nous, les humains, manipulons au quotidien. Mathématiquement, chaque ensemble flou A appartenant à un univers de discours U est associé à une fonction d'appartenance notée μ_A à valeur dans l'intervalle [0,1]. Cette fonction associe à chaque indivdu x de A un degré d'appartenance noté : $\mu_A(x)$ indiquant le niveau d'appartenance de x à A. $\mu_A(x) = 1$ et $\mu_A(x) = 0$ correspondent respectivement à l'appartenance totale et la non-appartenance.

Cette nouvelle formulation permet une meilleure similitude à l'appréhension humaine de l'environnement contrairement à la logique booléenne. Pour illustrer d'avantage cette différence prenant le cas de la température ressentie montré sur la figure 1.1. Il est évident qu'une température de 0°C est perçue comme étant froide et qu'une température de 40°C comme étant chaude, mais qu'en est-il de 15°C?. En logique booléenne on est obligé de considérer cette température soit froide ou chaude, mais en logique floue, on peut l'exprimer de façon plus similaire à notre ressenti, car cette température n'est ni froide ni chaude, mais plutôt froide que

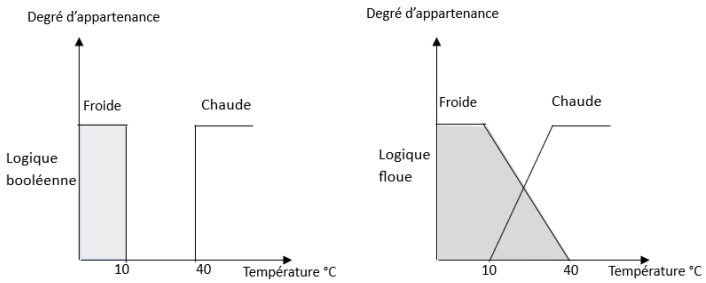

FIGURE 1.1 – Degrès d'appartenance en logique Booléenne et en logique floue

chaude.

Cet exemple permet d'illustrer le fait qu'une logique binaire classique soit, dans certains cas, trop limitative. Donc, il est nécessaire de faire appel à une autre logique multiévaluée qui sera vue comme une extension de la précédente, c'est bien que la logique floue.

Cette théorie d'ensemble flou a grandement contribué dans divers domaines scientifiques en particulier en automatique. Souvent, lors de la synthèse des lois de commande, on se retrouve face à des systèmes mal définis et difficiles à décrire par des modèles mathématiques précis, rendant ainsi la synthèse de la loi de commande très difficile, voire impossible. Mais grâce à cette nouvelle théorie, la commande et la modélisation des systèmes deviennent nettement moins complexes. Elles requièrent principalement l'intervention d'un expert connaissant bien le système, qui pourra élaborer les modèles et lois de commande en s'appuyant sur ses connaissances.

1.2.1 Variable linguistique

Une variable linguistique est définie par le triplet (x, U, T_x) où x est l'information réelle observée et définie sur un univers de discours U représentant la plage de variation de cette dernière. À cet univers de discours est associé une partition floue T_x composée d'un nombre fini d'ensembles flous $T_x = A_1, A_2, \ldots$, chaque ensemble flou A_i correspond à un terme linguistique utilisé pour la description de l'information x. Dans l'exemple précédent, la température correspond à l'information décrite dans un univers de discours allant de 0°C à 40°C comme indiqué sur la figure 1.2.

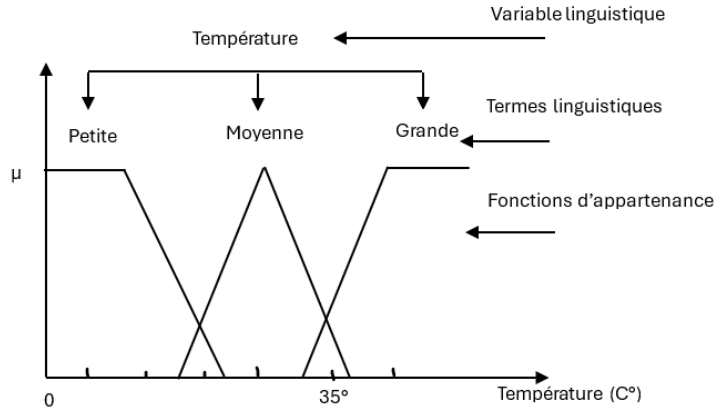

FIGURE 1.2 – Variable linguistique

1.2.2 Modificateur linguistique

En logique floue, les modificateurs linguistiques jouent un rôle essentiel pour représenter la gradation ou l'incertitude. Voici quelques exemples de modificateurs linguistiques en logique floue :

1. **Très** : Le modificateur **Très** est utilisé pour indiquer une forte appartenance à un ensemble flou. Par exemple, "Il fait très chaud aujourd'hui" indique que la chaleur est fortement présente.

2. **Un peu** : Par exemple, "la vitesse de la voiture un peu lente" indique que la vitesse est moyenement lente

1.2.3 Fonction d'appartenance

Soit X un ensemble d'individus x_i. Un ensemble flou A dans X est défini par la fonction d'appartenance $\mu_A(x_i)$ à valeurs dans l'intervalle $[0, 1]$ comme le montre l'équation 1.1.

$$\forall\, x_i \in \mathbb{X}, \mu_A(x_i) \in [0, 1] \tag{1.1}$$

1.3 Concepts des ensembles flous

1.3.1 Support

Le support d'un sous-ensemble flou A noté : $Supp(A)$ est défini comme étant l'ensemble classique des éléments x_i pour lesquels $\mu_A(x)$ est non nulle comme le montre l'équation 1.2.

$$\forall x_i \in \mathbb{X}, supp(A) = \{x_i / \mu_A(x_i) > 0\} \tag{1.2}$$

1.3.2 Hauteur

La hauteur d'un sous-ensemble flou A notée : $h(A)$, correspond à la hauteur maximale de la fonction d'appartenance $\mu_A(x)$. Cette hauteur est généralement normalisée dans l'intervalle [0, 1].

$$\forall x_i \in \mathbb{X}, h(A) = max(\mu_A(x_i)) \tag{1.3}$$

1.3.3 Noyau

Le noyau d'un ensemble flou A noté : $N(A)$ est l'ensemble classique de tous les éléments x_i de X qui appartiennent totalement à A comme le montre l'équation 1.4.

$$\forall x_i \in \mathbb{X}, N(A) = \{x_i / \mu_A(x_i) = 1\} \tag{1.4}$$

1.3.4 Le point de croisement

Le point de croisement d'un ensemble flou A de X est le sous-ensemble des éléments de X pour lesquels la fonction d'appartenance prend une valeur égale à 0.5. C'est l'ensemble des éléments de X qui appartiennent autant à A qu'à son complémentaire comme le montre l'équation 1.5.

$$\forall x_i \in \mathbb{X}, C(A) = \{x_i / \mu_A(x_i) = 0.5\} \tag{1.5}$$

1.3.5 La cardinale

La cardinalité d'un ensemble flou A de X, noté $|A|$ est le nombre d'éléments appartenant à A pondéré par leur degré d'appartenance comme le montre l'équation 1.6.

$$|A| = \begin{cases} \int_X f(x)dx, & \text{si } X \textit{ est continu} \\ \sum_{x_i \in X} \mu_A(x) & \text{si } X \textit{ est descret} \end{cases} \tag{1.6}$$

où l'expression :

$$\|A\| = \frac{|A|}{|X|} \tag{1.7}$$

est appelée cardinalité relative de A. On peut alors l'interpréter par la fraction d'éléments dans A, pondérés par leurs degré d'appartenance en A.

1.3.6 Coup de niveau α ou $\alpha - coupe$

$\alpha - coupe$ c'est l'ensemble booléen des éléments de X qui appartiennent à A avec un degré d'appartenance au moins égal à α.

$$\forall\, x_i \in \mathbb{X} \wedge \forall\, \alpha \in [0,1], \alpha - coupe(A) = \{x_i / \mu_A(x_i) > \alpha\} \tag{1.8}$$

Toutes les caractéristiques mentionnées ci-dessus sont illustrées par la figure 1.3

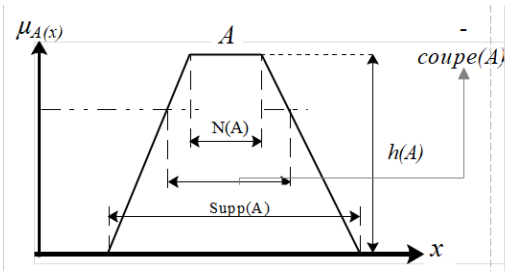

FIGURE 1.3 – Caractéristiques des sous-ensombles flous

Les fonctions d'appartenance les plus utilisées pour définir les ensembles flous sont :

1.3.7 Fonction triangulaire

L'allure de cette fonction est complètement définie par trois paramètres $\{a, b, c\}$:

$$f(x; a, b, c) = max(min(\frac{x-a}{b-a}, \frac{c-x}{c-b}), 0) \tag{1.9}$$

où : a, b et c sont les projections des trois sommets du triangle.

1.3.8 Fonction trapézoide

Cette fonction nécessite un paramètre de plus par rapport à la fonction triangulaire, elle est donnée par : $\{a, b, c, d\}$.

$$f(x; a, b, c, d) = max(min(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}), 0) \tag{1.10}$$

où : a, b, c et d sont les projections des sommets du trapèze

1.3.9 Fonction gaussienne

Elle est définie par deux paramètres $\{\sigma, c\}$ comme le montre l'équation 1.11.

$$f(x; \sigma, c) = \exp \frac{-(x-c)^2}{2\sigma^2} \tag{1.11}$$

La figure 1.4 donne une illustration graphique de ces différentes formes de fonctions d'appartenance.

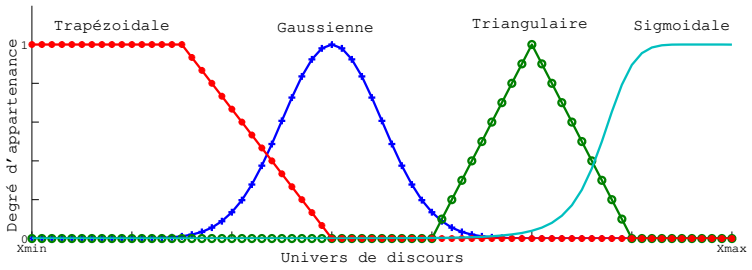

FIGURE 1.4 – Types de fonctions d'appartenance

1.4 Opérateurs flous

On retrouve pratiquement les mêmes opérateurs que dans la logique classique, sauf qu'en logique floue, leurs définitions diffèrent.

Soit A et B deux ensembles flous définis dans le référentiel U et ayant comme fonctions d'appartenance μ_A et μ_B respectivement.

1.4.1 Union

L'union de deux ensembles flous A et B est l'ensemble flou $\mu_{A\cup B}$ défini par l'équation 1.12.

$$\forall x \in \mathbb{U}, \mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x) \tag{1.12}$$

Le + représente souvent l'opérateur *Max*.

$$\mu_A(x) + \mu_B(x) = max\{\mu_A(x), \mu_B(x)\} \tag{1.13}$$

1.4.2 Intersection

L'intersection de deux ensembles flous A et B est l'ensemble flou $\mu_{A\cap B}$ défini par l'équation 1.14.

$$\forall x \in \mathbb{U}, \mu_{A\cap B}(x) = \mu_A(x) * \mu_B(x) \tag{1.14}$$

Le $*$ représente souvent l'opérateur *Min* défini par l'équation 1.15.

$$\mu_A(x) * \mu_B(x) = min\{\mu_A(x), \mu_B(x)\} \tag{1.15}$$

1.4.3 Complémentation

La fonction d'appartenance μ_A du complément de l'ensemble A, noté $\bar{A}$, est définie par l'équation 1.16.

$$\forall x \in \mathbb{U}, \mu -_A (x) = 1 - \mu_A(x) \tag{1.16}$$

La figure 1.5 résume ces opérations de façon graphique.

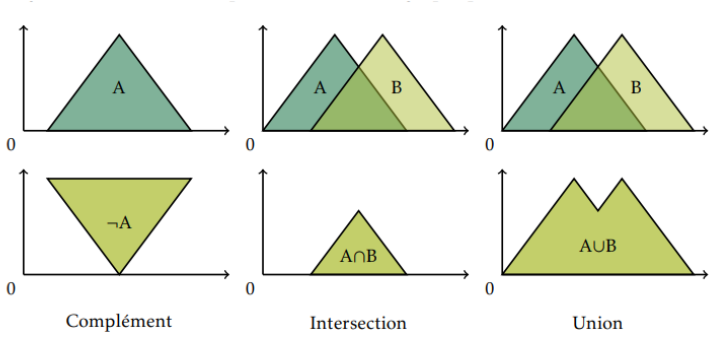

FIGURE 1.5 – Représentation graphique des opérateurs flous

1.4.4 Distance entre ensembles flous

La notion des distances entre ensembles flous peut être utile pour définir des relations de type « à peu près égal à » ou « très supérieur à ». Soit A, B deux ensembles flous de X, la distance entre ces deux ensembles A et B est définie par l'équation 1.17

$$d_p(A,B) = \begin{cases} (\int_X |\mu_A(x) - \mu_B(x)|^p)^{\frac{1}{p}} dx, & \text{si } X \textit{ est continu} \\ (\sum_{x_i \in X} (\mu_A(x) - \mu_A(x))^p)^{\frac{1}{p}} & \text{si } X \textit{ est descret} \end{cases} \tag{1.17}$$

En particulier si $p = 1$ on parle alors de la distance de Hamming donnée par l'équation 1.18

$$d_p(A,B) = \begin{cases} (\int_X |\mu_A(x) - \mu_B(x)|) dx, & \text{si } X \textit{ est continu} \\ (\sum_{x_i \in X} (\mu_A(x) - \mu_A(x))) & \text{si } X \textit{ est descret} \end{cases} \tag{1.18}$$

1.5 Proposition floue

Une proposition floue élémentaire est définie à partir d'une variable linguistique dans l'univers de discours T_X, elle est sous la forme X *est* A. Par exemple, *la température est froide* est une proposition élémentaire définie à partir de la variable linguistique *la température*.

La valeur de vérité d'une proposition élémentaire (X *est* A) est égale à $\mu_A(x)$ où x correspond à la valeur numérique exacte de X.

En général, une proposition floue est définie à partir de propositions élémentaires et d'opérateurs logiques. Il existe plusieurs méthodes pour calculer la valeur de vérité de telles propositions. Le tableau 1.1 présente les opérateurs communément utilisés.

Norme	Opérateur	Concepteur
Conjonction (x_1 est A) et (x_2 est B)	$\min(\mu_A(x_1), \mu_B(x_2))$	Logique Zadeh
	$\max(\mu_A(x_1) + \mu_B(x_2) - 1, 0)$	Logique Lukasiewicz
	$\mu_A(x_1) * \mu_B(x_2)$	Logique probabiliste
Disjonction (x_1 est A) ou (x_2 est B)	$\max(\mu_A(x_1), \mu_B(x_2))$	Logique Zadeh
	$\max(\mu_A(x_1) + \mu_B(x_2), 1)$	Logique Lukasiewicz
	$\mu_A(x_1) + \mu_B(x_2) - \mu_A(x_1) * \mu_B(x_2)$	Logique probabiliste

1.6 Règle floue

Une règle floue est généralement composée des éléments suivants :

Antécédent flou : C'est la partie de la règle qui précise les conditions, dite aussi prémisse ; elle correspond à la condition associée à la règle en question. L'antécédent flou est généralement composé de propositions floues. Par exemple, dans un système de contrôle de climatiseur, une règle floue pourrait avoir un antécédent tel que "Si la température est chaude ET l'humidité est élevée".

Conséquent flou : C'est la partie de la règle qui précise les actions ou les conclusions à prendre si les conditions de l'antécédent sont satisfaites. Le conséquent flou est également exprimé en termes de propositions floues. Par exemple, le résultat pourrait être "Alors augmenter la puissance du climatiseur".

Fonctions d'implication : Les fonctions d'implication déterminent comment les valeurs de vérité des propositions floues de l'antécédent influencent les valeurs de vérité des propositions floues du conséquent. Différentes fonctions d'implication peuvent être utilisées en fonction des caractéristiques spécifiques du problème.

La figure 1.6 présente un modèle d'une règle floue.

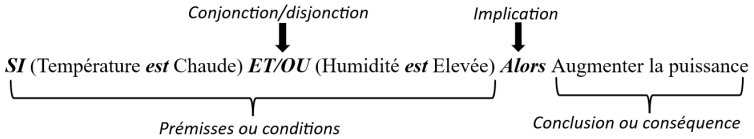

FIGURE 1.6 – Règle floue

1.7 Système à inférence floue

En se basant sur ce qui a été présenté jusqu'à maintenant, le système flou peut être défini comme une unité de traitement de l'information symbolique avec un support physique et numérique. Cette hybride capable de réagir comme le raisonnement des humains tout en étant des machines. Le paragraphe précédent donnait un aperçu de ses composantes. Cependant, pour qu'un système flou fonctionne correctement, en plus de la représentation mathématique des opérateurs flous, il est essentiel de définir des partitions floues uniques pour chaque variable d'entrée et de sortie. L'ajout d'un ensemble de règles floues constitue le premier élément topologique du système flou, appelé base de connaissances. Cette base est connectée à deux interfaces de conversion. La première s'appelle l'interface de fuzzification et est chargée de convertir tous les nombres saisis en valeurs linguistiques. La seconde s'appelle l'interface de défuzzification. Cette dernière est chargée de convertir les résultats des calculs flous en valeurs numériques pouvant être plus facilement traitées par les machines.

La figure 1.7 présente une structure générale d'un système à inférence floue.

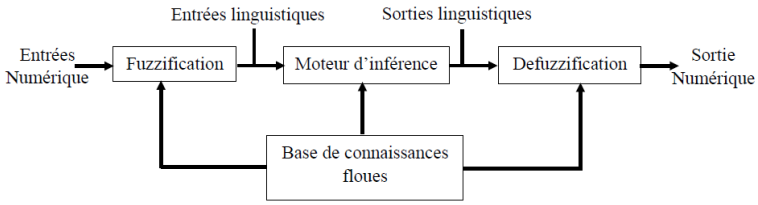

FIGURE 1.7 – Architecture d'un système à inférence floue

1.7.1 Base de règles

C'est un ensemble de règles floues qui capturent les connaissances et les stratégies de prise de décision. Chaque règle est généralement exprimée en termes d'antécédents flous (conditions) et de conséquences floues (actions). Par exemple, une règle pourrait être : "Si la température est chaude ET l'humidité est élevée, alors activez le climatiseur."

1.7.2 Fuzzification

La fuzzification est l'opération de rendre une entrée précise en valeur linguistique. Les valeurs d'entrée sont converties à partir d'ensembles flous, où chaque élément est associé à un degré d'appartenance spécifique. Les fonctions d'appartenance sont appliquées aux mesures et des degrés de vérité sont établis pour chaque proposition.

1.7.3 Moteur d'inférence

Le moteur d'inférence est le composant central du système à inférence floue. Il utilise les règles floues et les données d'entrée pour calculer le degré de vérité de chaque règle. L'opérateur ($\wedge$) dans l'équation 1.19 présente une conjonction entre les parties d'une prémisse.

$$w_{R_i}(x_1, x_2) = \mu_{Aj}(x_1) \wedge \mu_{Bj}(x_2) \tag{1.19}$$

Le moteur d'inférence affecte le degré de vérité de chaque règle sur la conclusion associée à la prémisse afin de déterminer le résultat flou de la règle en cours. Par exemple, la hauteur de l'ensemble flou de la conclusion peut être modulée par le degré d'activation précédemment calculé, réduisant ainsi cette hauteur. Dans une autre méthode, l'ensemble flou de la conclusion est simplement tronqué à la valeur du degré d'activation. Dans le premier cas, on parle d'inférence *PRODUIT* tandis que dans le second on parle d'inférence *MINIMUM*. La figure 1.8 montre une illustration graphique des deux méthodes.

1.7.4 Composition de règles floues

Cette étape combine toutes les conclusions des règles pour obtenir un ensemble flou global de la variable de sortie. Les opérations *SOMME* et *MAXIMUM* forment ses principales formulations. La première consiste à caractériser l'ensemble de sortie par une fonction d'appartenance égale à la somme des fonctions d'appartenance des sous-ensembles flous, par contre la seconde cherche le maximum entre les fonctions d'appartenance des sous-ensembles flous. La figure 1.9 regroupe une composition des résultats de deux règles floues par les deux méthodes discutées.

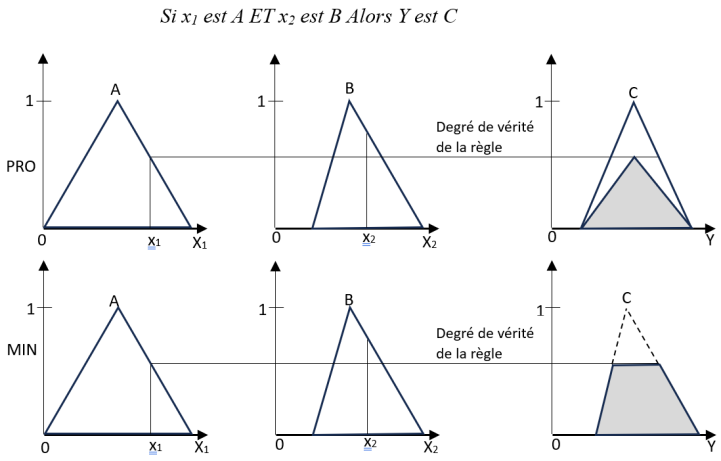

FIGURE 1.8 – Illustration graphique d'inférence produit et minimum

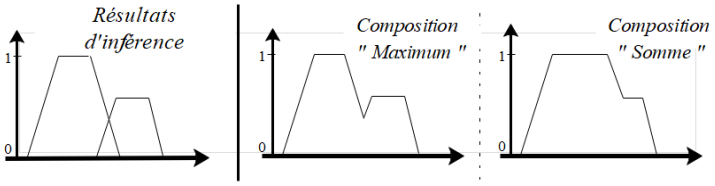

FIGURE 1.9 – Composition des ensembles flous

1.7.5 Défuzzification

La défuzzification est le processus final qui convertit les sorties floues en valeurs numériques précises pour les variables de sortie. Il existe différentes méthodes de défuzzification, telles que la méthode par centre de gravité. Il s'agit de déterminer la coordonnée en y du centre de gravité [27] du sous-ensemble flou A résultant de l'inférence comme le montre l'équation 1.20

$$COG(y) = \frac{\sum_{x=a}^{b} \mu_A(x) * x}{\sum_{x=a}^{b} \mu_A(x)} \tag{1.20}$$

La défuzzification peut aussi être réalisée par d'autres méthodes, par exemple en prenant la valeur maximale de la fonction d'appartenance résultante ou bien la moyenne des abscisses du maximum [28]. À partir d'une interprétation géométrique de cette dernière, la méthode *MOM* peut être utilisée par exemple dans des applications de prise de décisions pour privilégier la sortie la "plus plausible". La méthode du centre de gravité (COG) est utilisée avec l'inférence max-min de Mamdani. Cela est nécessaire, car la méthode d'inférence de Mamdani n'a pas d'effet interpolateur, et l'utilisation de la méthode *MOM* dans ce cas-ci fournit une sortie abrupte avec

des changements par morceaux.

Les règles floues jouent un rôle clé dans la représentation des connaissances et de l'expérience des experts en matière de contrôle/modélisation et dans l'établissement de lien entre les variables d'entrée des contrôleurs/modèles flous et la (ou les) variable(s) de sortie. Il existe deux grands types de règles floues, à savoir les règles floues de Mamdani et les règles floues de Takagi-Sugeno-Kang (TSK, en abrégé). Premièrement, nous commençons par les systèmes flous familiers de Mamdani.

1.7.6 Inférence de Mamdani (linguistique)

Le mécanisme d'inférence dans le modèle linguistique montré par la figure 1.10 est basé sur la règle compositionnelle d'inférence [29]. L'idée de base consiste à dériver un ensemble flou de sortie qui représente une conclusion globale issue de l'évaluation de l'ensemble des règles (agrégation) et du constat d'un certain fait d'entrée. Par simplicité, considérons ici une seule règle avec antécédent et conséquent scalaires de la forme donnée par l'équation 1.21.

$$R_i \; SI \; (x_1 \; est \; A) \; ET \; (x_2 \; est \; B) \; Alors \; y \; est \; C \tag{1.21}$$

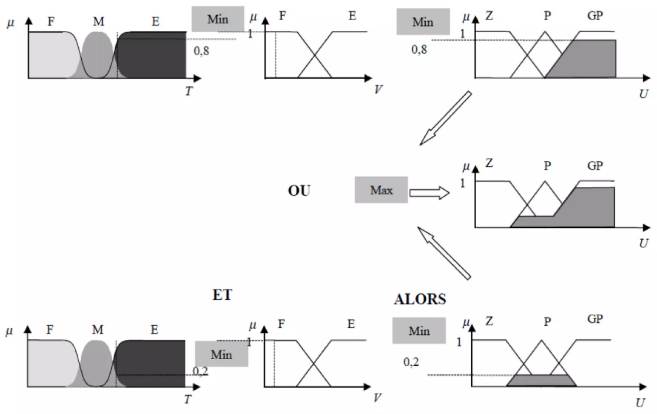

FIGURE 1.10 – Inférence de Mamdani

1.7.7 Inférence de TSK (précis)

D'une manière générale, un modèle de type Takagi-Sugeno (TS) [7] est basé sur une collection de règles R_i de la forme indiquée par l'équation 1.22.

$$R_i \; SI \; (x_1 \; est \; A) \; ET \; (x_2 \; est \; B) \; Alors \; y = f_i / \; f_i = a_{0i} + a_{1i}x_1 + a_{2i}x_2 \tag{1.22}$$

La méthode de Sugeno permet de simplifier le calcul de l'aggrégation, afin d'obtenir plus rapidement une solution nette. Cette méthode est souvent utilisée pour des applications à temps réel, où le temps de calcul est important. La méthode de Sugeno utilise un seul pic comme fonction d'appartenance, plutôt qu'un polygone. Ce pic est un ensemble flou ayant une fonction d'appartenance égale à 1 à un point particulier de l'espace et 0 ailleurs. Le résultat de l'évaluation des règles devient l'amplitude du pic.

La particularité du modèle TSK est qu'il utilise une fonction linéaire pour calculer la valeur de sortie. Plus précisément, chaque règle définit une fonction linéaire de la forme :

$f_i(x_1, x_2, \ldots, x_r) = a_{0i} + a_{1i} * x_1 + a_{2i} * x_2 + \cdots + a_{ri} * x_r$

où r est le nombre de variables d'entrée, $x_1, x_2, \ldots, x_r$ sont les variables d'entrée et a_0, a_1 sont des coefficients constants.

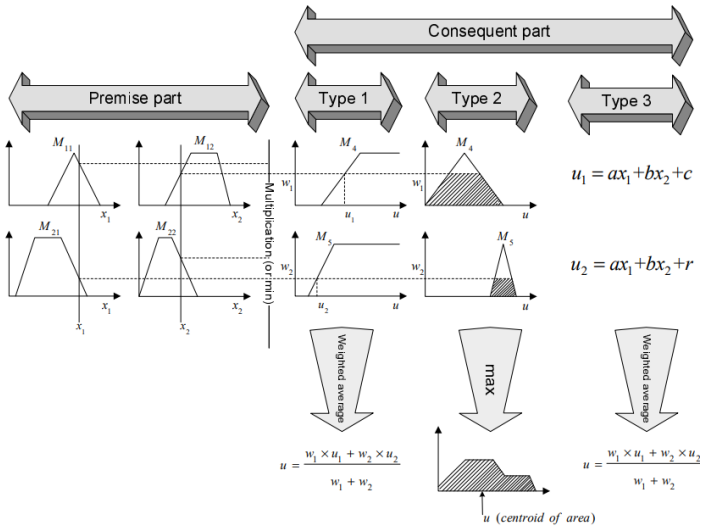

FIGURE 1.11 – Exemple d'inférence Sugeno avec operateur MAX

La figure 1.11 illustre un exemple de fonction de type Sugeno pour un système à deux entrées et deux sous-ensembles flous.

Pour faire l'aggrégation des règles, on utilise une moyenne pondérée donnée par l'équation 1.23.

$$Y_{sugeno} = \frac{\sum_i \mu_A(k_i) * k_i}{\sum_i \mu_A(k_i)} \tag{1.23}$$

1.8 Les architectures neuro-flous

Les systèmes neuro-flous ([30], [31], [32]) également connus sous le nom de réseaux neuronaux flous (FNN pour Fuzzy Neural Networks en anglais), combinent des concepts de réseaux neuronaux artificiels (RNA) [33] avec la logique floue pour résoudre des problèmes impliquant des données incertaines, non linéaires ou floues. Les systèmes neuro-flous exploitent la puissance des réseaux neuronaux pour la modélisation des données avec la flexibilité de la logique floue pour traiter l'incertitude et la non-linéarité. Ils sont particulièrement adaptés aux domaines où les données sont complexes, incomplètes ou imprécises, et où des modèles robustes sont essentiels pour la prise de décision et le contrôle.

Voici quelques hybridations neur-flou :

1.8.1 Système neuro-flou coopératif

Un système Neuro-flou coopératif peut être considéré comme préprocesseur où le mécanisme d'apprentissage de réseaux de neurones artificiels (RNA) détermine les fonctions d'appartenance de système d'inférence flou (SIF) ou les règles floues à partir de données d'apprentissage. Une fois les paramètres du SIF établis, le rôle du RNA devient secondaire. Les règles basées sur les données sont généralement déterminées par un algorithme de clustering flou, tandis que les fonctions d'appartenance sont approximées par le RNA à partir des données d'apprentissage. La figure 1.12 représente le modèle Neuro-Flou coopératif [34].

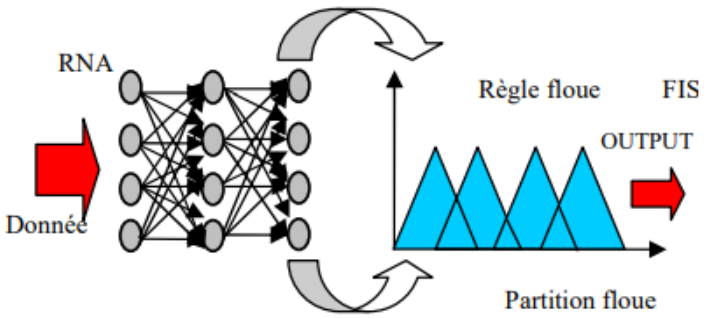

FIGURE 1.12 – Architecture neuro-flou coopérative

1.8.2 Système neuro-flou concourant

Dans un système neuro-flou concourant, le RNA assiste en continu le SIF pour ajuster les paramètres, en particulier lorsque les variables d'entrée du contrôleur ne peuvent pas être mesurées directement. Dans certains cas, les sorties du SIF peuvent ne pas être directement applicables au processus. La figure 1.13 représente le modèle Neuro-Flou concourant [34].

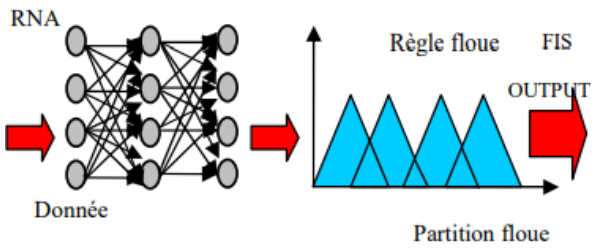

FIGURE 1.13 – Architecture neuro-flou concourante

1.8.3 Falcon (Fuzzy Adaptive Learning Control Network)

FALCON a une architecture à cinq couches [35], comme il est représenté dans la figure 1.14. Il y a deux neurones pour chaque variable de sortie. Une pour les données d'apprentissage (sortie désirée) et l'autre est pour la sortie de FALCON. La première couche cachée sert à fuzzifier les variables d'entrées. Chaque neurone dans cette couche représente une fonction d'appartenance. La deuxième couche cachée définit les parties antécédentes des règles floues suivies par les parties conséquences des règles dans la troisième couche cachée. FALCON emploie un algorithme d'apprentissage hybride comportant l'apprentissage non supervisé pour initialiser les fonctions d'appartenance et les règles floues, et l'apprentissage supervisé pour un ajustement optimal de l'ensemble des paramètres du système flou.

1.8.4 Le NEFCON (Neuro-Fuzzy Control)

NEFCON [36] est conçu pour mettre en application le système d'inférence flou de type Mandani . Il est constitué de 2 couches dont les poids représentent les ensembles flous et les règles floues comme le montre sur la figure 1.15. La couche d'entrée assure la tâche de l'interface de fuzzification, la logique d'inférence est représentée par les

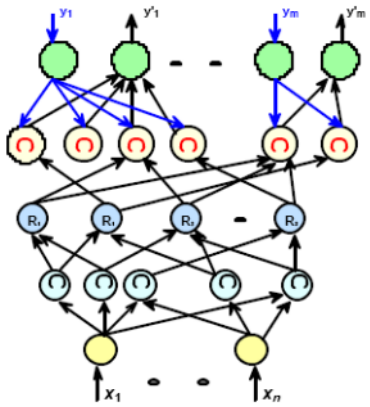

FIGURE 1.14 – Architecture FALCON

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

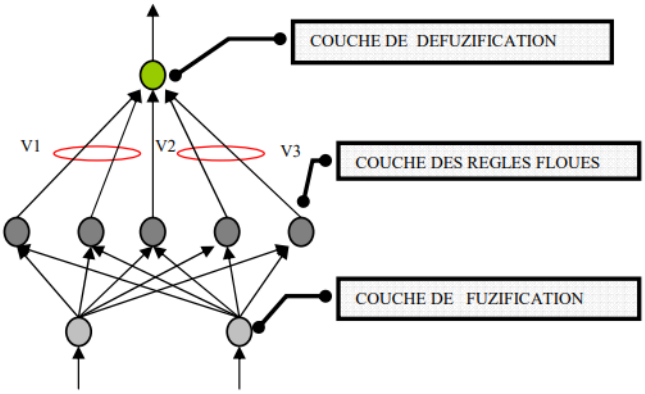

FIGURE 1.15 – Architecture NEFCON

1.8.5 Le ANFIS (Adaptive Network Based Fuzzy Inference System)

ANFIS (Adaptive Network Based Fuzzy Inference System) [37] c'est un système d'inférence adaptatif neuro-flou qui consiste à utiliser un réseau de neurones de type MLP à 5 couches pour lequel chaque couche correspond à la réalisation d'une étape d'un système d'inférence floue de type Takagi-Sugeno. Pour la simplicité, nous supposons que le système d'inférence flou a deux entrées x et y, et une seule sortie f. Supposons que la base de règles contient deux règles floues de type Takagi-Sugeno.

$$Regle\ 1\ : Si\ x\ est\ A_1\ et\ y\ est\ B_1\ Alors\ f_1 = p_1x + q_1y + r_1 \tag{1.24}$$

$$Regle\ 2\ : Si\ x\ est\ A_2\ et\ y\ est\ B_2\ Alors\ f_2 = p_2x + q_2y + r_2 \tag{1.25}$$

L'ANFIS correspondant à une architecture composée de cinq couches, comme représenté sur la figure 1.16.

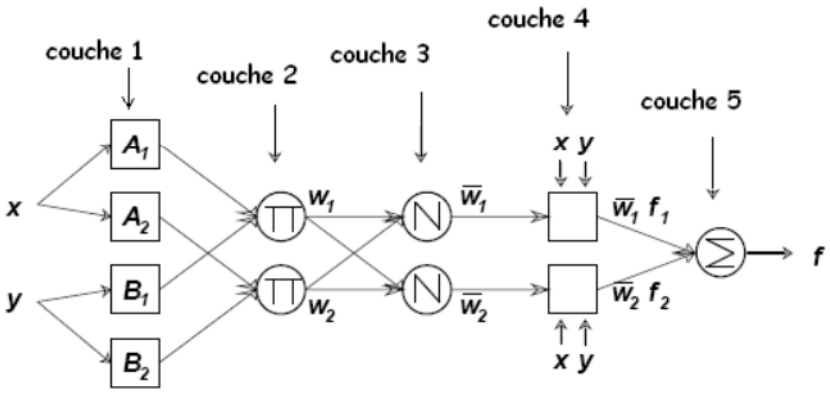

FIGURE 1.16 – Architecture ANFIS

1.9 Conclusion

Dans ce chapitre, nous avons exploré les systèmes flous d'un point de vue théorique. Nous avons défini les concepts de base tels que les variables linguistiques, les sous-ensembles flous et les opérateurs flous, ce qui a permis de rapprocher la pensée humaine de celle de la machine. À travers cette étude, nous avons constaté que les modèles de type Mamdani et Sugeno sont les plus couramment utilisés. Nous avons ensuite présenté les architectures des systèmes flous les plus populaires et leurs domaines d'application. Enfin, nous avons examiné les diverses combinaisons de la logique floue avec les réseaux de neurones, telles qu'elles sont abordées dans la littérature.

La complexité et les difficultés rencontrées pour l'ajustement et le raffinage des paramètres des systèmes flous nécessitent l'utilisation de méthodes d'optimisation, qui seront détaillées dans le chapitre suivant.

CHAPITRE 2

Optimisation et algorithmes évolutionnaires

2.1 Introduction

Les scientifiques sont confrontés à des problèmes extrêmement variés, dont la complexité ne cesse d'augmenter avec les années. En optimisation, les problèmes sont classés en plusieurs grandes catégories : combinatoires ou continus, mono-objectifs ou multi-objectifs, avec ou sans contraintes [38]. Selon ces catégories, diverses méthodes de résolutions sont élaborées pour s'adapter aux propriétés intrinsèques des problèmes afin d'être les plus performantes possible. En optimisation, un problème est exprimé par une formulation mathématique dite fonction objectif ou fonction de coût. Le but est de minimiser ou de maximiser la fonction considérée, c'est-à-dire de trouver l'optimum global ([39], [40]). Pour illustrer ce qu'est un problème d'optimisation, on peut prendre l'exemple des entreprises qui veulent minimiser les coûts de production tout en maximisant leurs profits. Bien que le problème d'optimisation semble purement mathématique, il est omniprésent et sollicité dans différents domaines tels que le traitement d'image, l'informatique, l'industrie, la physique, etc.

Ce second chapitre sera entièrement consacré au problème d'optimisation ainsi qu'aux différents algorithmes d'optimisation qui lui sont associés. Une formulation mathématique du problème d'optimisation sera donnée avant de présenter une taxonomie des algorithmes connus. La suite sera dédiée aux algorithmes plus récents et les plus populaires qui seront utilisés dans le cadre de ce travail.

2.1.1 Problème d'optimisation et formulation

Mathématiquement, un problème d'optimisation est décrit par ses fonctions objectifs à minimiser ou à maximiser, ses contraintes d'égalités et/ou d'inégalités, ainsi que son espace de recherche. Ce problème peut être formellement décrit par l'équation 2.1

$$\begin{cases} Max/Min(\sum_{m=1}^{M} f_m(x)) \\ \sum_{j=1}^{J} g_j(x) \leq 0 \\ \sum_{k=1}^{K} h_k(x) = 0 \\ x_{min} \leq x \leq x_{max} \end{cases} \tag{2.1}$$

où : f_m exprime la m^{ieme} fonction objectif, M représente le nombre total d'objectifs à optimiser. g_j et h_k : désignent respectivement les contraintes d'inégalité et d'égalité imposées aux variables. x représente le vecteur des variables de décision, tandis que x_{min} et x_{max} définissent les limites de l'espace de recherche du problème à maximiser ou à minimiser.

2.2 Taxonomie des méthodes d'optimisation

Dans la littérature, il existe une large panoplie d'algorithmes d'optimisation divers et variés. Ces algorithmes d'optimisation imitent généralement les règles de la physique. Parmi les algorithmes les plus populaires, citons l'Algorithme Optimiseur d'Équilibre [17], l'Algorithme de Recherche Gravitationnelle (GSA) [18], l'Optimisation de la Force Centrale (CFO) [41], l'Algorithme d'Optimisation des Réactions Chimiques Artificielles (ACROA) [42]. Le mécanisme de ces algorithmes est basé sur un ensemble aléatoire d'agents de recherche qui communiquent et qui se déplacent dans l'espace de recherche selon des règles physiques. Ces mouvements sont mis en œuvre, par exemple, en utilisant la force gravitationnelle, la projection de rayons, la force électromagnétique, la force d'inertie, les poids, etc. Cependant, les algorithmes basés sur une population sont aujourd'hui très connus et utilisés.

Ces derniers parcourent l'espace de recherche afin de trouver la meilleure solution possible. Ils requièrent souvent des processus itératifs qui vont améliorer une ou plusieurs solutions à la fois. Ainsi, la recherche s'oriente progressivement vers la solution optimale. Ces algorithmes, généralement qualifiés de méta-heuristiques, se distinguent aussi par leurs origines, mécanismes et formulations.

2.2.1 Méta-heuristiques

Les méta-heuristiques sont des techniques d'optimisation qui offrent une alternative aux méthodes exactes, particulièrement lorsque ces dernières deviennent difficiles à appliquer à des problèmes complexes ou de grande taille. Leur popularité n'a cessé de croître ces dernières décennies en raison de leur efficacité dans de tels contextes. Certaines approches, comme les Algorithmes Génétiques [10], l'Optimisation par Colonies de Fourmis [12] et l'Optimisation par Essaims de Particules [11], sont largement reconnues, non seulement en informatique, mais aussi dans divers domaines scientifiques. En plus d'un riche corpus de recherches théoriques, ces méthodes ont été appliquées avec succès à de nombreux champs d'étude.

Mais qu'est-ce qui rend les méta-heuristiques si répandues ? Quatre atouts majeurs expliquent cet engouement : leur simplicité, leur flexibilité, leur indépendance vis-à-vis des dérivées et leur capacité à éviter les optima locaux.

D'abord, leur simplicité repose sur des concepts inspirés de phénomènes naturels, qu'ils soient physiques, biologiques ou évolutionnaires. Cette accessibilité facilite leur apprentissage, favorisant ainsi l'émergence de nouvelles variantes, l'hybridation de plusieurs approches et l'amélioration des techniques existantes. De plus, leur compréhension aisée permet aux chercheurs de divers domaines de les adapter rapidement à leurs propres problématiques.

Ensuite, contrairement aux méthodes basées sur le gradient, les méta-heuristiques ne nécessitent pas le calcul des dérivées. Elles fonctionnent par exploration stochastique, en débutant avec un ensemble de solutions générées aléatoirement. Cette caractéristique les rend particulièrement adaptées aux problèmes pour lesquels les informations dérivées sont difficiles à obtenir ou inexistantes. De plus, elles permettent d'éviter la stagnation dans des solutions locales en explorant plus largement l'espace de recherche, qui est souvent complexe et rempli d'optima locaux.

Toutefois, il n'existe pas de méta-heuristique universelle capable de résoudre efficacement tous les problèmes d'optimisation. Un algorithme performant sur un type de problème peut être inefficace sur un autre, ce qui alimente un champ de recherche dynamique visant à améliorer continuellement les méthodes existantes et à en concevoir de nouvelles.

Globalement, les méta-heuristiques se divisent en deux grandes catégories : celles basées sur une solution unique et celles exploitant une population de solutions. La première approche, illustrée par le recuit simulé [43], démarre avec une solution unique qui s'affine progressivement. À l'inverse, les méthodes basées sur une population, comme les Algorithmes Génétiques, optimisent un ensemble de solutions évoluant au fil des itérations. Ces dernières offrent certains avantages tels que :

— Les solutions candidates multiples partagent des informations sur l'espace de recherche, ce qui permet de faire des sauts vers les régions prometteuses de cet espace.
— Les solutions candidates multiples collaborent pour éviter les solutions localement optimales.
— Les méta-heuristiques basées sur la population offrent généralement une plus grande capacité d'exploration que les algorithmes basés sur une solution unique.

L'une des branches intéressantes des méta-heuristiques basées sur la population est l'intelligence en essaim (Swarm Intelligence). Selon les chercheurs [44], l'intelligence en essaim est "l'intelligence collective émergente de groupes d'agents simples". Les techniques d'intelligence en essaim s'inspirent principalement des colonies naturelles, des troupeaux et des écoles de poissons. Parmi les techniques d'intelligence en essaim les plus populaires, on trouve l'optimisation par colonies de fourmis [17], l'optimisation par essaims de particules [11] et l'algorithme de colonie d'abeilles artificielles (ABC) [45]. Parmi les avantages des algorithmes d'intelligence en essaim, on peut citer les suivants :

— Les algorithmes d'intelligence en essaim (SI) conservent des informations sur l'espace de recherche au cours des itérations, tandis que les algorithmes évolutionnaires (EA) ne conservent pas généralement les informations des générations précédentes.

— Les algorithmes d'intelligence en essaim utilisent souvent la mémoire pour conserver la meilleure solution obtenue jusqu'à présent.

— Les algorithmes d'intelligence en essaim nécessitent généralement moins de paramètres à ajuster.

— Les algorithmes d'intelligence en essaim utilisent généralement moins d'opérateurs que les approches évolutionnaires, telles que le croisement, la mutation, l'élitisme, etc.

— Les algorithmes d'intelligence en essaim sont faciles à mettre en œuvre.

Algorithm 1 Structure générale d'un algorithme d'évolution.

1: Générer aléatoirement une population de solutions candidates P et les évaluer ;
2: **Tant que** la condition d'arret n'est pas vérifiée **faire :**
3: Produire une population de nouvelles solutions candidates P' en effectuant des modifications aléatoires sur des éléments sélectionnés dans P,
4: Evaluer les solutions de P',
5: Remplacer certains éléments de P par des éléments de P',
6: **Fin Tant que**
7: Retourner la meilleure solution de P

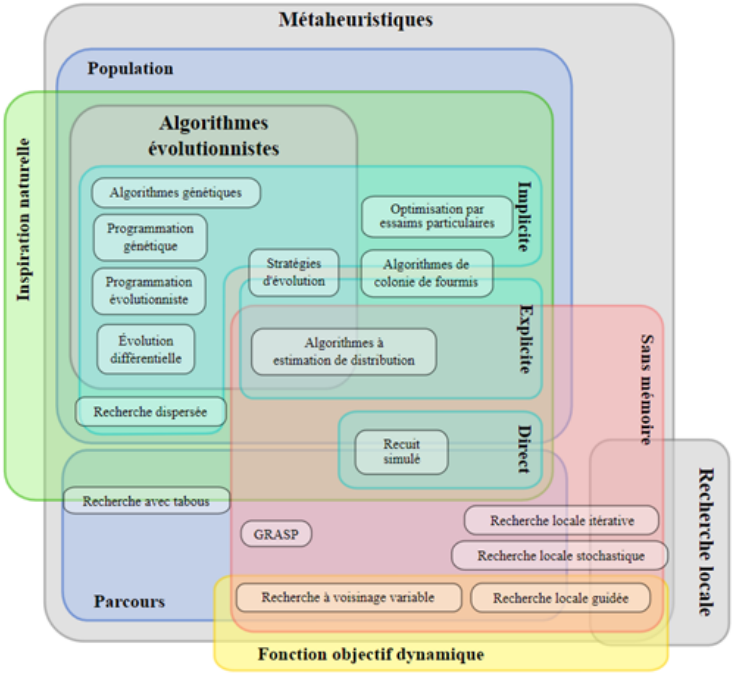

FIGURE 2.1 – Différentes classes des méta-heuristiques

La figure 2.1 offre une vue d'ensemble de la classification des méta-heuristiques en fonction de leurs origines et de leurs mécanismes.

2.2.2 Mécanismes des méta-heuristiques

Les méta-heuristiques s'articulent autour de trois mécanismes fondamentaux : l'exploration, l'exploitation et la mémorisation.

Le premier mécanisme, l'exploration, se réfère au processus d'exploration approfondie des zones prometteuses de l'espace de recherche. Pour soutenir cette phase, un algorithme doit disposer d'opérateurs stochastiques permettant une recherche aléatoire et globale dans l'espace de recherche.

Le second mécanisme, l'exploitation, fait référence à la capacité de recherche locale autour des régions prometteuses identifiées lors de la phase d'exploration.

Le troisième mécanisme, la mémorisation, soutient l'apprentissage en permettant à l'algorithme de se concentrer sur les zones où l'optimum global est susceptible de se trouver, évitant ainsi les optima locaux.

2.2.3 Algorithme génétique

Cet algorithme a été proposé par Holland [46] et simule les concepts de l'évolution darwinienne. Les applications techniques de l'algorithme génétique (AG) ont été largement étudiées par Goldberg [10]. L'optimisation est réalisée en faisant évoluer une population de solutions à travers les générations. Chaque nouvelle génération est créée par la combinaison et la mutation des individus de la génération précédente. Étant donné que les meilleurs individus ont une probabilité plus élevée de participer à la génération de la nouvelle population, cette dernière est généralement meilleure que les générations précédentes. Cela permet d'optimiser la population initiale aléatoire au fil des générations.

Dans sa formulation, l'algorithme génétique travaille avec une population de chromosomes, où chaque chromosome est constitué de gènes représentant les variables du problème, généralement codés en binaire ou en valeurs réelles. Ces chromosomes sont ensuite évalués à l'aide d'une fonction objectif pour déterminer la performance des solutions qu'ils représentent. Pour ce faire, l'algorithme génétique utilise trois mécanismes principaux.

Sélection

La sélection est une étape cruciale qui détermine quels individus d'une population vont être choisis pour transmettre leurs gènes à la génération suivante. L'objectif est de privilégier les individus les plus aptes (selon une fonction d'évaluation ou de fitness) afin d'améliorer la qualité des solutions au fil des générations. Dans le processus de sélection, les individus sont choisis parmi la population de parents $P = \{s_1, s_2, ..., s_N\}$ où chaque individu s_k a une probabilité $P_k(s_k) = \frac{f(s_k)}{\sum_{i=1}^{N} f(s_i)}$ d'être sélectionné, la population P^1 est ainsi constituée. Chaque individu est codé sous forme de chromosomes, ce qui permet de créer un lien entre la valeur de la variable et l'individu dans la population, imitant ainsi la transcription génotype-phénotype observée dans le monde vivant.

Les principales méthodes de sélection utilisées sont :

Sélection par roulette

Chaque individu a une chance proportionnelle à sa fitness d'être sélectionné. Les individus avec une meilleure fitness ont donc plus de chances d'être choisis, mais même ceux avec une faible fitness ont une petite chance d'être sélectionnés.

Sélection par tournoi

Un sous-ensemble d'individus est sélectionné aléatoirement et le meilleur individu de ce groupe est choisi. Cela peut augmenter la précision de sélection.

Sélection par classement

Les individus sont classés en fonction de leur fitness, et la probabilité d'être sélectionné dépend de leur rang dans ce classement plutôt que de leur fitness absolue.

Élitisme

Un certain pourcentage des meilleurs individus est directement transféré à la génération suivante sans modification, garantissant ainsi que les solutions les plus performantes ne soient pas perdues.

Codage

Il existe principalement trois types de codage : le codage binaire, le codage en base n et le codage réel.

Le premier type, le codage binaire, est le plus simple à mettre en œuvre grâce à un alphabet réduit à deux éléments : 0 et 1. De plus, il repose sur des fondements

théoriques solides, comme la théorie des schémas, et bénéficie de la facilité de mise en œuvre des opérateurs génétiques.

Le deuxième type de codage, le codage en base n, utilise un alphabet $\{0, 1, 2, ..., n-1\}$ pour représenter les différentes valeurs de la variable d'optimisation. Ce codage est particulièrement adapté aux problèmes d'optimisation impliquant des variables entières, tels que le problème du voyageur de commerce et la conception des règles floues de type Mamdani.

Le codage réel diffère considérablement des deux codages précédents et est généralement plus simple à utiliser. Lorsqu'on cherche à optimiser une fonction de n variables $f(x_1, x_2, ..., x_n)$, alors le chromosome H est représenté sous la forme :

$H = \{x_1, x_2, ..., x_n\}$

L'évaluation du chromosome est accélérée grâce à l'absence de procédure de décodage. En outre, le codage réel offre souvent une meilleure précision et un gain notable en temps d'exécution.

Croisement

La liste des individus est parcourue et chaque individu a une probabilité P_c d'être sélectionné pour le croisement. Lorsque deux individus sont choisis, un point de coupure est tiré aléatoirement, et les chaînes binaires sont échangées à partir de ce point. Cela donne lieu à une nouvelle population d'enfants . Ce mécanisme est illustré par la figure 2.2.

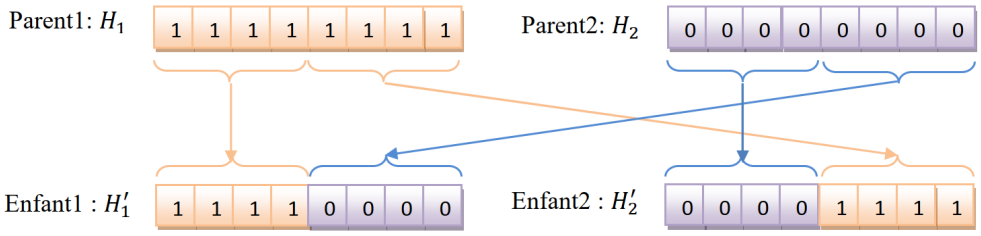

FIGURE 2.2 – Principe du croisement binaire en un point

On trouve également d'autres variantes de ce type de croisement, telles que le croisement multipoint [47]. Dans cette méthode, plusieurs points de croisement sont sélectionnés le long du chromosome, et les fragments des chromosomes parentaux sont copiés en alternance pour créer de nouveaux chromosomes pour les enfants. Cette technique peut être appliquée aussi bien au codage en base n qu'au codage

réel. Toutefois, pour le codage réel, d'autres formes de croisement sont souvent privilégiées, tels que le croisement barycentre [48] et le croisement Laplacien [49].

Mutation

Les individus de la population issue du croisement sont soumis à une phase de mutation. Chaque individu a une faible probabilité P_m de voir un ou plusieurs de ses gènes modifiés de manière aléatoire. Cette étape est essentielle pour maintenir la diversité génétique et prévenir la stagnation dans des optima locaux. L'opérateur de mutation diffère suivant le type de codage utilisé, ainsi dans le codage binaire, la mutation est très simple à élaborer, il suffit de complémenter à 1 tous les allèles des chromosomes des enfants lorsque la probabilité est inférieure à P_m. La figure 2.3. donne un exemple illustratif pour la mise en œuvre de cet opérateur.

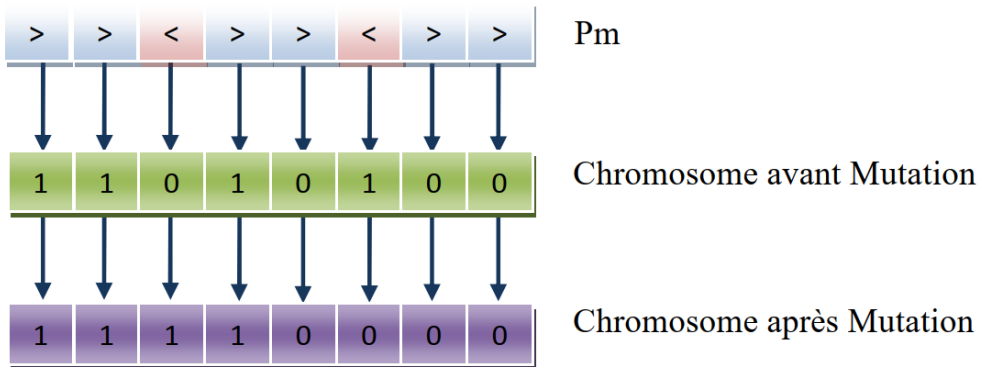

FIGURE 2.3 – Principe de la mutation binaire

Les étapes du processus de fonctionnement de l'algorithme génétique sont résumées dans l'algorithme 2.

Algorithm 2 Pseudo-code d'un algorithme génétique.

1: Définition de la fonction objectif $F(x)$;
2: Définition des opérateurs génétiques et leurs probabilités respectives (P_s, P_c, P_m) ;
3: Initialisation des chromosomes et mettre le compteur de génération à 1 ;
4: **Faire :**
5: Décodage des chromosomes ;
6: Evaluation de la fitness des individus représentés ;
7: Sélection des parents ;
8: Croisement des parents ;
9: Mutation des enfants ;
10: **Jusqu'à ce qu'un critère d'arrêt soit atteint**

2.2.4 Algorithme optimiseur d'équilibre

L'approche de l'OE s'inspire d'un simple bilan de masse dynamique dans un volume de contrôle bien mélangé, dans lequel une équation de bilan de masse est utilisée pour décrire la concentration d'un constituant non réactif dans un volume de contrôle en fonction de ses divers mécanismes de source et de puits. L'équation du bilan massique fournit la physique sous-jacente pour la conservation de la masse entrant, sortant et générée dans un volume de contrôle. Une équation différentielle ordinaire du premier ordre, qui exprime le bilan de masse générique, montre que le changement de masse dans le temps est égal à la quantité de masse entrant dans le système plus la quantité générée à l'intérieur, moins la quantité quittant le système. Cette relation est décrite par l'équation 2.2.

$$V\frac{dC}{dt} = QC_{eq} - QC + G \tag{2.2}$$

C est la concentration dans le volume de contrôle (V) , $V\frac{dC}{dt}$ est le taux de variation de la masse dans le volume de contrôle, Q est le débit volumétrique à l'entrée et à la sortie du volume de contrôle, C_{eq} représente la concentration à un état d'équilibre dans lequel il n'y a pas de génération à l'intérieur du volume de contrôle, et G est le taux de génération de masse à l'intérieur du volume de contrôle. Lorsque $V\frac{dC}{dt}$ atteint zéro, un état d'équilibre stable est atteint. Un réarrangement de l'équation 2.2 permet de résoudre $\frac{dC}{dt}$ en fonction de $\frac{Q}{V}$; où $\frac{Q}{V}$ représente l'inverse du temps de résidence, appelé ici γ, ou le taux de renouvellement $\gamma = \frac{Q}{V}$. Par la suite, l'équation 2.2 peut également être modifiée pour résoudre la question de la concentration dans le volume de contrôle (C) en fonction du temps (t) comme le montre l'équation 2.3.

$$\frac{dC}{\gamma C_{eq} - \gamma C + \frac{G}{V}} = dt \tag{2.3}$$

L'équation 2.4 montre l'intégration de l'équation 2.3 au fil du temps

$$\int_{C_0}^{C} \frac{dC}{\gamma C_{eq} - \gamma C + \frac{G}{V}} = \int_{t_0}^{t} dt \tag{2.4}$$

Le résultat est donné par l'équation 2.5.

$$C = C_{eq} + (C_0 - C_{eq})F + \frac{G}{\gamma V}(1 - F) \tag{2.5}$$

Dans l'équation 2.5, F est calculé par léquation 2.6.

$$F = \exp^{-\gamma(t-t_0)} \tag{2.6}$$

où t_0 et C_0 sont respectivement le temps initial et la concentration initiale. L'équation 2.5 peut être utilisée pour estimer la concentration dans le volume de contrôle avec un taux de renouvellement connu ou pour calculer le taux de renouvellement moyen à l'aide d'une simple régression linéaire avec un taux de génération connu et d'autres conditions.

La figure 2.4 et l'algorithme 3 résument le mécanisme de fonctionnement de l'algorithme optimiseur d'équilibre.

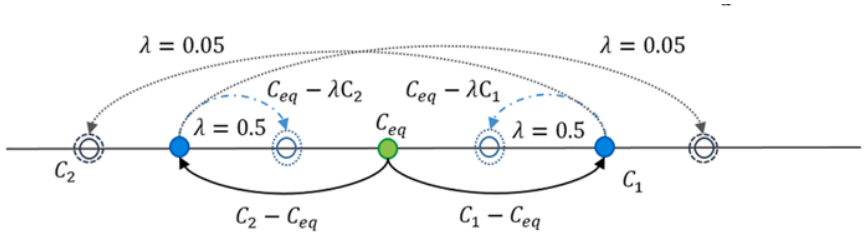

FIGURE 2.4 – Mécanisme de mise à jour pour EQ

Algorithm 3 Pseudo-code d'un algorithme Equilibrium Optimizer.

Initialisation des variables ; population initiale ;
Calculer la fitness ;
De $i = 1 : L$ **Faire :**
Si ($fit(C_i < fit(C_{eq1})$) mis à jour C_{eq1} utilisant la mise à jour $fit(C_i)$;
Répéter la comparaison $fit(C_x)$ et $fit(C_{eqx})$;
Mise à jour C_{eqx} utilisant $fit(C_x)$;
Fin De ;
$C_{avg} = \frac{(C_{eq1}+C_{eq1}+...+C_{eqn})}{n}$;
Si $iter \geq 1$ mis à jour le mécanisme de mémorisation ;
De $i = 1 : n$ **faire** ;
Choisir aléatoirement un condidat de $EQ0_{pool}$;
Génération aléatoire des vecteurs γ ; r ;

2.2.5 Optimisation par essaims de particules

Comme son nom l'indique, l'algorithme PSO (Particle Swarm Optimization) développé par [11] est un algorithme de recherche stochastique basé sur la population qui simule le comportement social des oiseaux au sein d'une volée. Récemment, il a été largement utilisé dans plusieurs domaines d'application grâce à son efficacité. Les chercheurs de [50], [51] et [52] ont utilisé le PSO dans différents domaines tels que la classification et les problèmes d'ingénierie. Cet algorithme évolutionnaire a également été utilisé pour améliorer les performances et réduire la complexité des modèles dédiés à l'identification de systèmes non linéaires [19] et [53]. L'un des principaux avantages de l'algorithme PSO est sa capacité à trouver un optimum global ou quasi-optimum global en explorant correctement l'espace de recherche. En règle générale, dans le PSO, tous les membres de la population survivent jusqu'à la fin du processus. Leur interaction se traduit par une amélioration itérative de la qualité de la solution au fil du temps.

Après la distribution aléatoire des particules, chacune se déplace vers une solution dans l'espace de recherche appelée optimum local ($local_{par}$). L'ensemble de l'essaim se dirige vers la meilleure solution parmi toutes les solutions trouvées, désignée comme optimum global ($global_{par}$). La position de chaque particule est évaluée comme une solution potentielle.

Au fil des itérations successives, chaque particule met à jour sa position en utilisant sa meilleure position historique ($local_{par}$) en vue d'atteindre la meilleure position globale ($global_{par}$) selon les équations 2.7 et 2.8.

$$Vel_q(k+1) = (Vel_q(k) + C_1 r_1(local_{Par_q}(k) - Par_q(k)) + C_2 r_2(global_{par}(k) - Par_q(k))) \tag{2.7}$$

$$Par_q(k+1) = Par_q(k) + Vel_q(k+1) \tag{2.8}$$

où

— C est un facteur de constriction, C_1 et C_2 sont les coefficients d'accélération. Les deux paramètres C_1 et C_2 sont empiriques selon la relation $C_1 + C_2 \geq 4$ et $C = \frac{2}{C_1+C_2-2+\sqrt{(C_1+C_2)^2-4(C_1+C_2)}}$ telle que définie dans [54].

— r_1 et r_2 sont des nombres aléatoires uniformément distribués dans [0 1].

— $local_{par_q}$ est la meilleure position de la q^{ime} particule par rapport à son historique jusqu'à la k^{ime} itération.

— $global_{par}$ est la meilleure position de l'essaim jusqu'à la k^{th} itération.

L'évolution de déplacement de chaque particule est illustrée par la figure 2.5.

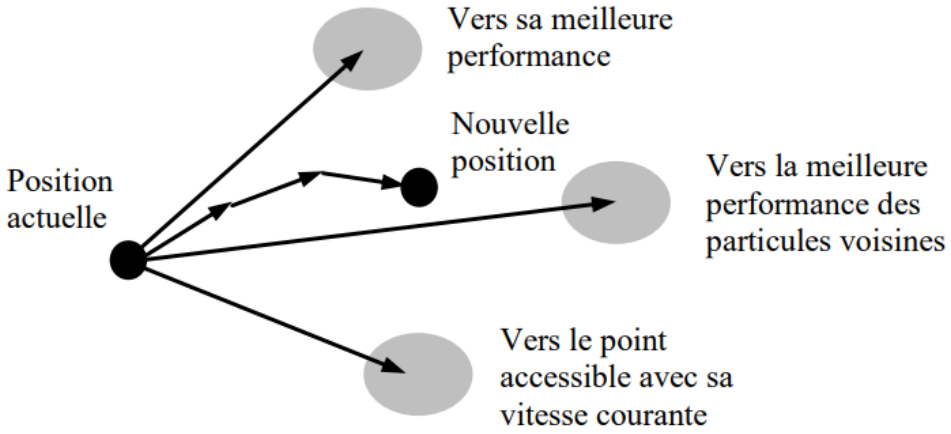

FIGURE 2.5 – Stratégie de déplacement d'une particule

Topologie de voisinage

La topologie de voisinage a un impact significatif sur la performance de l'algorithme PSO. Selon diverses études, la structure en anneau est souvent considérée comme plus efficace que d'autres configurations. La figure 2.6 illustre les différentes topologies existantes.

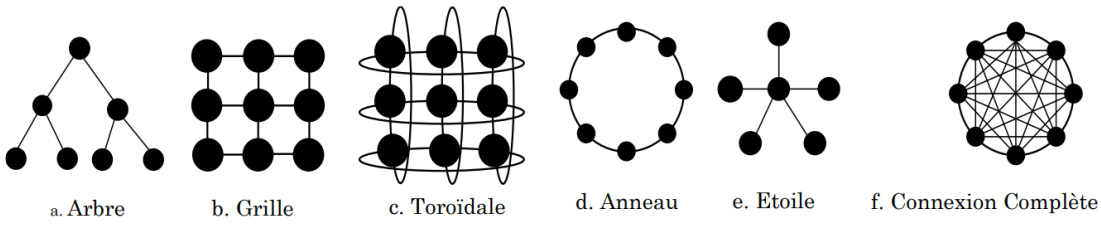

FIGURE 2.6 – Différentes topologies de voisinage

Le pseudo-code donné par l'algorithme 4 résume la manière globale de l'évaluation de l'algorithme PSO.

Algorithm 4 Pseudo-code d'un algorithme PSO

1: Définir la fonction objectif $F(x)$;
2: Initialiser les positions et les vitesses pour chaque particule et mettre l'indice d'itération à 1 ;
3: Initialiser les meilleures positions personnelles des particules à leurs positions initiales ;
4: Evaluer toutes les positions des particules ;
5: Mettre à jour les vitesses de toutes les particules ;
6: Mettre à jour la meilleure position personnelle de chaque particule ainsi que la meilleure position globale de l'ensemble de l'essaim.. Incrémenter l'indice d'itération
7: Jusqu'à ce qu'un critère d'arret soit atteint ;

2.3 Conclusion

Dans ce chapitre, nous avons abordé le problème d'optimisation en nous concentrant sur les algorithmes évolutionnaires. Nous avons particulièrement étudié le fonctionnement des algorithmes génétiques et du PSO, que nous utiliserons dans le cadre de cette thèse.

Dans le chapitre suivant, nous présenterons une version améliorée du PSO que nous avons développée pour l'apprentissage des paramètres d'un système neuro-flou.

CHAPITRE 3

Apprentissage des paramètres du système flou TSK

3.1 Introduction

En automatique, on est souvent confronté à des problèmes mathématiques complexes, que ce soit lors de la modélisation ou de la commande. Les méthodes déjà existantes, bien qu'elles aient fait leurs preuves, restent assez difficiles à appliquer pour des systèmes hautement non linéaires, complexes ou mal connus [55]. C'est dans cette optique que des techniques intelligentes [56] ont été introduites pour surmonter ces difficultés et faire face à la polyvalence des systèmes rencontrés.

Parmi ces techniques figure la logique floue avec son modèle TSK, très utilisé tant pour la commande que pour la modélisation [57]. Grâce à sa capacité d'approximation, similaire à celle des réseaux de neurones, et à la possibilité d'intégrer l'expertise d'un expert humain, les systèmes flous TSK sont devenus très populaires dans la littérature.

Le modèle TSK tente de décomposer l'espace d'entrée en régions floues, puis d'approximer le système dans chaque région par une simple équation linéaire [58]. Le principal avantage de ce modèle réside dans son pouvoir représentatif et sa capacité à décrire un système hautement non linéaire à l'aide de données d'entrée/sortie.

Cependant, le modèle TSK comprend trop de paramètres, rendant sa conception et son utilisation assez difficile. C'est la raison pour laquelle l'optimisation de ce type de modèle nécessite des méthodes d'optimisation plus adaptatives ([59], [60]), pour lesquelles nous utilisons l'optimisation par essaim de particules qui ne nécessite pas de description mathématique du problème d'optimisation et qui est capable de localiser l'optimum global dans un environnement complexe. En effet, la flexibilité

du PSO permet d'ajuster tous les paramètres des règles floues ([61], [62]).

3.2 Structure du système neuro-flou

La structure du modèle neuro-flou utilisée dans le cadre de cette thèse est organisée sous la forme d'un réseau neuronal à cinq couches, comme le montre la figure 3.1.

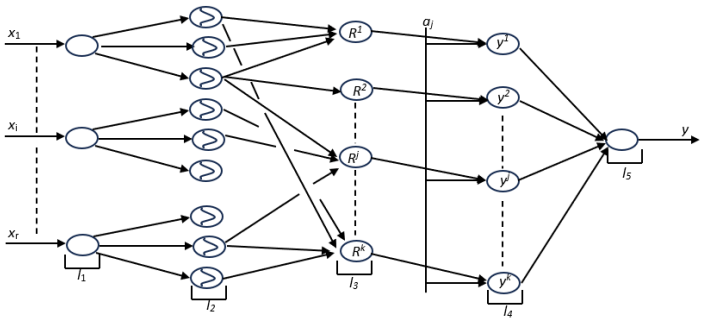

FIGURE 3.1 – Structure du réseau de neuro-flou

3.2.1 Première couche

Aucune fonction n'est effectuée par les neurones de cette couche ; chaque neurone se contente de transmettre le signal d'entrée à la couche suivante, comme précisé par l'équation 3.1.

$$Q_i^1 = u_i^1,\ u_i^1 = x_i \tag{3.1}$$

3.2.2 Deuxième couche

Les neurones de cette couche représentent les fonctions d'appartenance (ensembles flous). La sortie de chaque neurone indique le degré d'appartenance d'une variable d'entrée à l'ensemble flou associé à ce neurone. Ce degré d'appartenance est calculé selon l'équation 3.2.

$$Q_i^2 = \exp \frac{u_i^2 - c_i^2}{\sigma_i^2} \tag{3.2}$$

$$u_i^2 = \begin{cases} Q_1^1 \\ Q_2^1 \\ Q_3^1 \end{cases} \tag{3.3}$$

où c_i^2 et σ_i^2 sont les paramètres de la i^{ieme} fonction gaussienne associée au i^{ieme} neurone (ce sont les paramètres à ajuster).

3.2.3 Troisième couche

Chaque neurone de cette couche correspond à une règle floue dans la base du modèle flou. Ses entrées proviennent de tous les nœuds de la couche 2 qui forment la partie prémisse de cette règle. La sortie de chaque nœud représente le degré d'activation de la règle calculé comme indiqué en équation 3.4.

$$Q_i^3 = \prod_i u_i^3, \;\; u_i^3 = Q_i^2, \;\; i = 1, 2, \ldots, k \tag{3.4}$$

où k est le nombre de règles floues, égal à $n_1 * n_2 * \ldots * n_r$. Les poids de liaison dans cette couche sont fixés à 1.

3.2.4 Quatrième couche

Chaque nœud de cette couche effectue une sommation linéaire des variables d'entrée, comme le montre l'équation 3.5.

$$Q_i^4 = a_0^i + \sum_{j=1}^{r} a_j^i x_j \tag{3.5}$$

Dans lequel a_j^i ($j = 1, 2, \ldots, r$ *et* $i = 1, 2, \ldots, k$) sont les paramètres à ajuster. Les liens entre cette couche et la couche de sortie sont égaux à l'unité.

3.2.5 Cinquième couche

Le nœud de cette couche représente la sortie globale du modèle. L'équation 3.6 est utilisée pour calculer la sortie.

$$Q_i^5 = \frac{\sum_{j=1}^{k} Q_i^3 * Q^4}{\sum_{j=1}^{k} Q_i^3} \tag{3.6}$$

3.3 PSO amélioré

À partir de la première version établie par Eberhart [11], plusieurs chercheurs ont proposé des versions améliorées pour en accroître l'efficacité du PSO. Les auteurs de [19] ont proposé une version améliorée en ajoutant un terme externe λ à l'équation de la vitesse 3.8, permettant de réduire la probabilité d'être piégé dans des optima locaux.

$$Par_q(k+1) = Par_q(k) + Vel_q(k+1) \tag{3.7}$$

$$Vel_q(k+1) = C(Vel_q(k) + C_1 r_1(local_{Par_q}(k) - Par_q(k))$$

$$+C_2 r_2(global_{par}(k) - Par_q(k))) + C_3\lambda(k) \tag{3.8}$$

où λ est appelée constante d'apprentissage additive, C_3 est un vecteur de nombres distribués aléatoirement.

Une autre version a été proposée par [50], où le poids d'inertie dans le PSO est utilisé pour équilibrer la recherche globale et locale. En effet, un poids d'inertie élevé facilite la recherche globale, tandis qu'un poids d'inertie faible facilite la recherche locale. En ajustant dynamiquement le poids d'inertie, la capacité de recherche est adaptée, ce qui améliore significativement le processus de recherche. Cela est mis en œuvre par l'introduction d'un nouveau paramètre appelé poids d'inertie adaptatif W, comme le montre l'équation 3.9.

$$Vel_q(k+1) = C(W * Vel_q(k) + C_1 r_1(local_{Par_q}(k) - Par_q(k))$$

$$+C_2 r_2(global_{par}(k) - Par_q(k))) \tag{3.9}$$

La version décrite dans ce chapitre est la version publiée dans notre article [63] (algorithme 5) qui est une combinaison des approches proposées dans [19] et [50], visant à tirer parti des avantages de chacune. Elle utilise le paramètre d'inertie adaptative W de l'équation 3.9 pour contrôler la vitesse de convergence et intègre

un terme externe λ (équation 3.8) pour réduire la probabilité de se retrouver piégé dans un minimum local. La fonction de mise à jour de la vitesse est alors donnée par l'équation 3.10

$$Vel_q(k+1) = C(W * Vel_q(k) + C_1 r_1(local_{Par_q}(k) - Par_q(k))$$

$$+C_2 r_2(global_{par}(k) - Par_q(k))) + C_3\lambda(k) \tag{3.10}$$

3.4 Optimisation des paramètres du modèle TSK par le PSO amélioré

3.4.1 Algorithme d'apprentissage

La figure 3.2 illustre la structure d'identification utilisée pour obtenir le modèle neuro-flou. L'algorithme d'apprentissage 5 suit les étapes classiques d'un PSO standard, les principales différences étant au niveau de l'équation de la mise à jour des vitesses des particules (équation 3.10), de la fonction objectif et de la structure des particules.

Algorithm 5 algorithme PSO amélioré

1: Initialiser $C_1 = C_2; C_3 = C; r_1; r_2$
2: $\lambda = \frac{1}{max(eig(X*X^T))}$
3: $min_{cost} = min(min(E)); mean_{cost} = mean(E)$
4: $global_{min} = min_{cost}; local_{par} = Par; local_{par} = E$
5: $k = 0$
6: **while** $k < maxit$ **do**
7: $k = k + 1$
8: $W = (maxit - k)/maxit$ ⇒équation 3.10)⇒(équation 3.7)
9: $local_{cost} = local_{cost} * not(better_{cost}) + E * better_{cost}$
10: $[temp, t] = min(local_{cost})$
11: **if** $temp < global_{cost}$ **then**
12: $global_{par} < Par_i(t, :) \rightarrow index = t \rightarrow global_{cost} = temp$
13: **end if**
14: $[k\ global_{par}\ global_{cost}]$
15: $min_{cost}(k+1) = min(E) \rightarrow global_{min}(k+1) = global_{cost}$
16: $mean_{cost}(k+1) = mean(E)$
17: **end while**

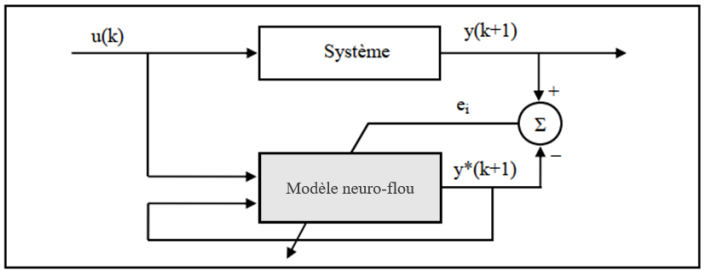

FIGURE 3.2 – Schéma de la structure d'identification floue

Fonction objectif

Le critère d'erreur quadratique moyen du signal de sortie de la dernière couche est utilisé pour évaluer les performances du modèle, comme indiqué dans l'équation 3.11

$$EQM = \frac{\sum_0^M (y^*(k+1) - y(k+1))^2}{M} \tag{3.11}$$

où $y^*(k+1)$ est la sortie souhaitée, $y(k+1)$ est la sortie réelle du modèle flou TSK et M est le nombre d'échantillons.

Structure des particules

Dans un essaim de N particules, la structure de la i^{ieme} particule est définie comme un vecteur tel qu'il est donné dans l'équation 3.12.

$$Par_i = [c_1, c_2, \ldots, c_n, \sigma_1, \sigma_2, \ldots, \sigma_n, a_1, a_2, \ldots, a_r] \tag{3.12}$$

où $c_1, c_2, \ldots, c_n$ et $\sigma_1, \sigma_2, \ldots, \sigma_n$ sont respectivement les centres et les variances des fonctions d'appartenance (paramètres antécédents) et $a_1, a_2, \ldots, a_r$ sont les paramètres des conclusions. n et r sont respectivement le nombre de fonctions d'appartenance et le nombre d'entrées.

3.4.2 Résultats et discussions

Dans cette section, afin de montrer les performances de la version du PSO que nous avons proposée, nous l'appliquons à deux problèmes d'identification classiques présentés par [64]. Dans ce contexte, les deux systèmes sont décrits par les équations aux différences 3.13 et 3.14.

$$y(k) = \frac{y(k-1)y(k-2)(y(k-1)+2.5)}{1+y^2(k-1)+y^2(k-2)}u(k-1) \tag{3.13}$$

$$y(k) = \frac{24+y(k-1)}{30}y(k-1) - 0.8\frac{u^2(k-1)}{1+u^2(k-1)}y(k-2) + 0.5u(k-1) \tag{3.14}$$

La conception du modèle flou pour chaque système est réalisée de la même manière et est décrite par le pseudo-code 5. Ce dernier intègre un ensemble de paramètres initialisés comme indiqué dans le tableau 3.1.

Tableau 3.1 – Paramètres de simulation

Paramètres	Valeurs
Maxit	1500
MF	3
Popsize	60
C_1	2.1
C_2	1.9
C_3	4
C	0.76

La conception du modèle flou se déroule en deux étapes principales : la phase d'apprentissage et la phase de validation. Lors de la phase d'apprentissage, le modèle flou est entraîné à partir d'un ensemble de données d'apprentissage, où les paramètres des fonctions d'appartenance et des règles floues sont optimisés à l'aide du PSO proposé. Une fois l'apprentissage terminé, la phase de validation évalue la performance du modèle flou sur un autre ensemble de données, appelé ensemble de validation. Cette étape permet de vérifier la capacité du modèle flou à généraliser sur de nouvelles données et à éviter le surapprentissage. Ensemble, ces deux étapes garantissent que le modèle flou obtenu est à la fois précis et robuste dans ses prédictions. Dans ce qui suit, les deux phases seront détaillées.

3.4.3 Phase d'apprentissage

Les données d'entrée comprennent 300 points générés par une séquence d'impulsions d'amplitude aléatoire dans la plage [-1, 1] et une période aléatoire entre 1 et 4 pour le premier modèle. Pour le second modèle, 200 points sont utilisés avec une période aléatoire dans l'intervalle [1, 20], comme illustré dans les figures 3.3 et 3.4.

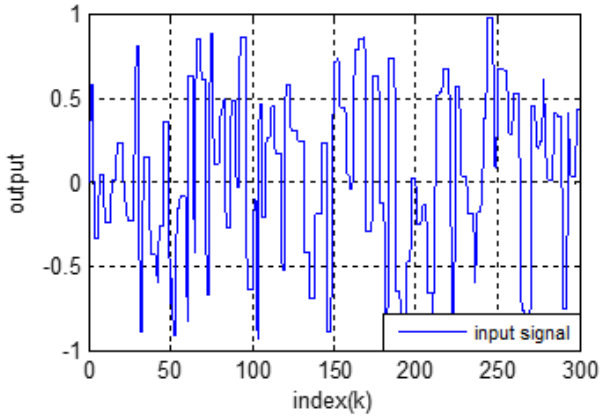

FIGURE 3.3 – Signal d'entrée pour la phase d'apprentissage du premier modèle

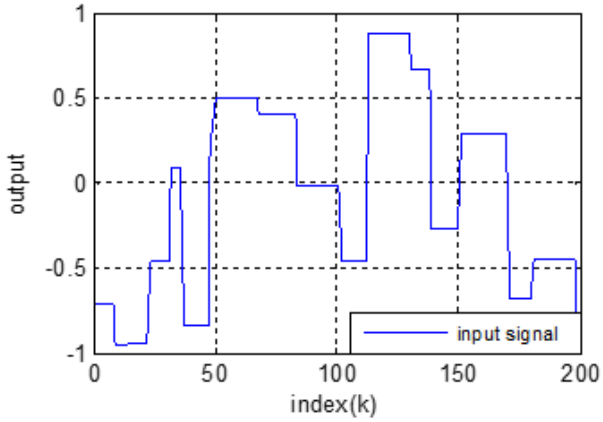

FIGURE 3.4 – Signal d'entrée pour la phase d'apprentissage du deuxième modèle

Les figures 3.5 et 3.6 fournissent une comparaison des sorties des systèmes et des modèles flous correspondants. On peut constater à travers ces deux figures que le signal prédit par chaque modèle flou est quasiment identique à la sortie du système associé.

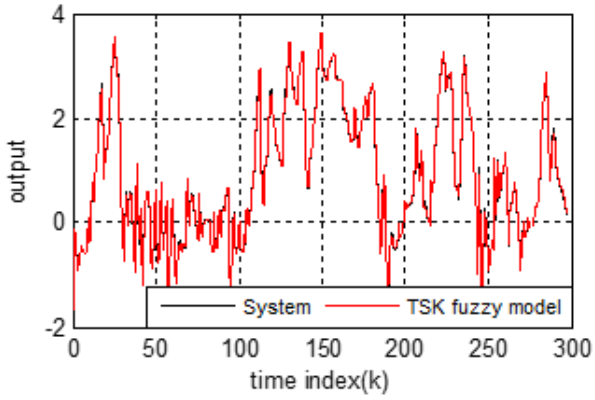

FIGURE 3.5 – Signaux de sortie du premier système et modèle flou

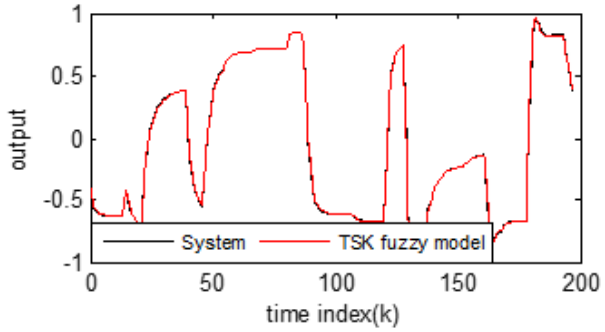

FIGURE 3.6 – Signaux de sortie du deuxième système et modèle flou

Les figures 3.7 et 3.8 montrent l'évolution de la fonction objectif EQM_{Train} pour les différentes versions du PSO. Le tableau 3.2 résume les résultats finaux obtenus par chacune des versions simulées pour les deux systèmes durant la phase d'apprentissage.

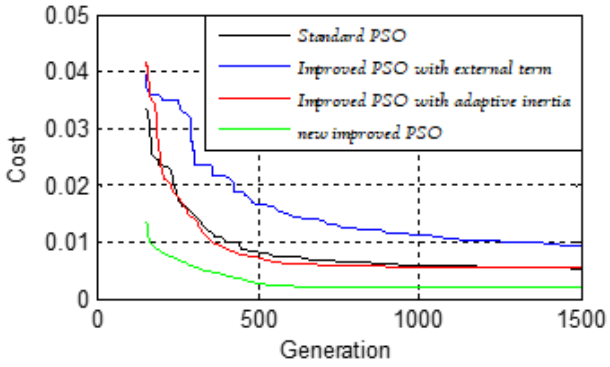

FIGURE 3.7 – Signaux de sortie du premier système et modèle flou

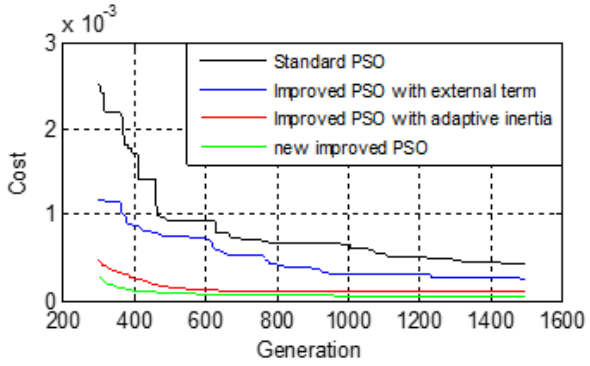

FIGURE 3.8 – Signaux de sortie du deuxième système et modèle flou

Les figures 3.7 et 3.8 montrent une amélioration très significative de la vitesse de convergence avec la nouvelle version proposée, tant pour le premier système que

Tableau 3.2 – Résultats de simulation pour la phase d'apprentissage

	Standard PSO	PSO améliorée avec terme externe	PSO amélioré avec inertie adaptative	Nouveau PSO améliorée
Premier système	0.0053	0.0092	0.0091	0.0019
Deuxième système	0.0037	0.0019	$4.96 * 10^{-3}$	$4.51 * 10^{-4}$

pour le deuxième système. En outre, cette version permet d'obtenir des modèles flous bien plus précis, comme le met en évidence le tableau 3.2.

3.4.4 Phase de test

Une fois le processus d'apprentissage achevé, un signal d'entrée sinusoïdal de la forme : $u_k = \sin(2\pi k/25)$ a été appliqué à chaque système ainsi qu'aux modèles correspondants. Les figures 3.9 et 3.10 illustrent la sortie de chaque système, accompagnée de celle du modèle flou correspondant. Là encore, comme lors de la phase d'apprentissage, on constate une très bonne similarité entre chaque système et son modèle flou. Le tableau 3.3 présente l'erreur quadratique moyenne des résultats de la simulation à ce stade.

Tableau 3.3 – Résultats de simulation pour la phase de test

	Standard PSO	PSO améliorée avec terme externe	PSO amélioré avec inertie adaptative	Nouveau PSO améliorée
Premier système	0.0042	0.0078	0.0091	0.0019
Deuxième système	0.0037	0.0019	$4.96 * 10^{-3}$	$4.51 * 10^{-4}$

D'après les valeurs de l'erreur quadratique moyenne présentées dans le tableau 3.3, il est clair que la nouvelle version du PSO proposée surpasse les autres versions, et ce, pour les deux systèmes étudiés.

3.5 Commande d'un pendule inversé

Dans cette partie, afin de ne pas nous limiter à la simulation, nous allons mettre en œuvre un contrôleur flou TSK optimisé par l'algorithme PSO pour la commande d'un banc d'essai de pendule inversé. Ce banc, conçu par la firme Feedback, est

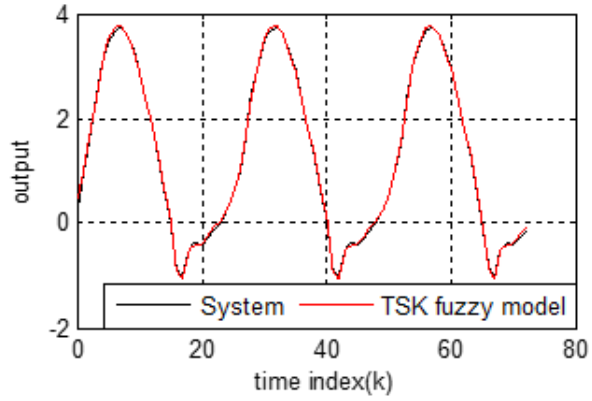

FIGURE 3.9 – Signaux de sortie du premier système et son modèle flou

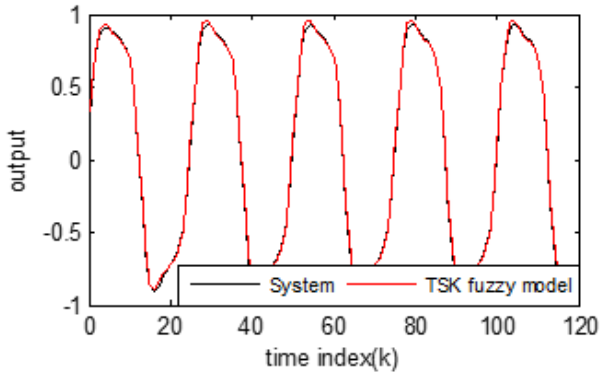

FIGURE 3.10 – Signaux de sortie du deuxième système et son modèle flou

disponible dans notre laboratoire LTII de l'Université de Béjaïa. L'objectif de cette application est de piloter le système de manière à stabiliser le pendule en position verticale tout en respectant les contraintes liées au mouvement du chariot. La figure 3.11 illustre la structure mécanique du système utilisé pour cette expérience.

Le modèle mathématique descriptif du système chariot-pendule est exprimé par l'équation 3.15 :

$$\begin{cases} F = (M+m)\ddot{x} + ml\cos(\theta)\ddot{\theta} - ml\sin(\theta)\dot{\theta}^2 \\ 0 = (ml^2 + I)\ddot{\theta} + ml\ddot{x}\cos(\theta) + d\dot{\theta} - mgl\sin(\theta) \end{cases} \tag{3.15}$$

où les variables $(x,\ \dot{x},\ \theta,\ \dot{\theta})$ correspondent respectivement à la position et la vitesse du chariot, ainsi qu'à l'angle et la vitesse angulaire du pendule. Le tableau 3.4 présente l'ensemble des paramètres physiques du banc d'essai.

FIGURE 3.11 – Structure mécanique du pendule inversé du laboratoire LTII

TABLEAU 3.4 – Paramètres du pendule inversé

Paramètres	définition	Valeurs
M	Masse du chariot	2.4 kg
m	Masse du pendule	0.23 kg
l	Longueur du pendule	0.36
I	Moment d inertie	0.099 $kg * m^2$
g	Acceleration gravitationnelle	9.81 m/s^2
d	Coefficient de frottement du pendule	0.005 Nms/rad
b	Coefficient de frottement du chariot	0.00005 N/m
Te	Période d'échantionnage	0.001 s
	région de stabilité	$\theta \subset$ [-0.2 0.2]

3.5.1 Structure du controleur flou TSK

Pour éviter une structure complexe du contrôleur TSK, qui pourrait entraîner un grand nombre de paramètres, les auteurs dans [61] proposent une architecture simplifiée reposant sur deux entrées ($E1, E2$) et une seule sortie (F). La première entrée se compose d'une combinaison de l'erreur linéaire et de sa variation, tandis que la seconde intègre l'angle de rotation et sa vitesse, comme indiqué dans l'équation 3.16. Ces combinaisons permettent une supervision globale du mouvement du chariot tout en réduisant le nombre de variables d'entrée.

La figure 3.12 illustre la structure du système de commande. Dans cette figure, le pendule reçoit la force F appliquée par le contrôleur, tandis que les sorties sont

représentées par la position x du chariot et l'angle θ du pendule. De plus, les facteurs d'échelle ($G1$, $G2$) inclus dans le système de commande permettent d'adapter les variables d'entrée à leurs univers de discours respectifs.

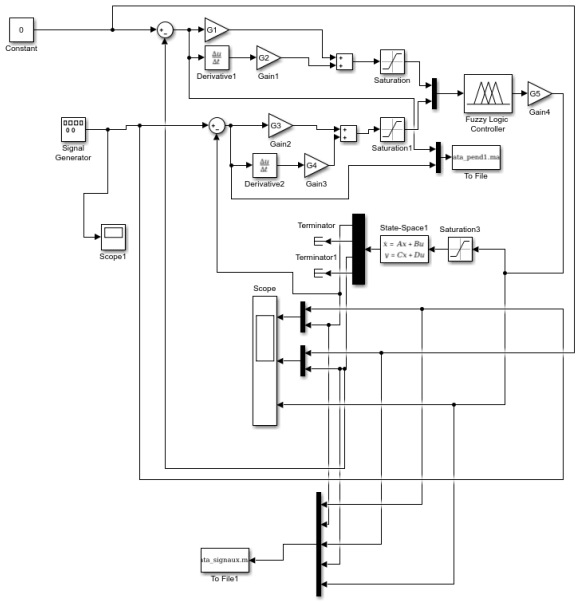

FIGURE 3.12 – Structure de la commande floue appliquée sur le pendule

$$\begin{cases} E_1 = G_1 e_x + G_2 \dot{e}_x \\ E_2 = G_3 \theta + G_4 \dot{\theta} \end{cases} \tag{3.16}$$

Nous utilisons un partitionnement avec trois fonctions gaussiennes asymétriques pour chacune des variables d'entrée $E1$ et $E2$, ce qui nous donne un total de neuf règles floues de la forme : Si $E1$ *est* $A1$ *et* $E2$ *est* $A2$ Alors : $F = a0 + a1 * E1 + a2 * E2$.

Dans ce cas, la structure de la i^{ieme} particule est définie comme un vecteur tel qu'il est donné dans l'équation 3.17.

$$Par_i = [c_1, c_2, \ldots, c_n, \sigma_1, \sigma_2, \ldots, \sigma_n, a_1, a_2, \ldots, a_r, G_1, G_2, G_3, G_4, G_5] \tag{3.17}$$

3.5.2 Phase d'apprentissage

L'apprentissage du système s'effectue sur 100 itérations, avec pour objectif l'optimisation de la fonction exprimée par l'équation 3.18. Cette fonction objectif est définie comme une agrégation de l'intégrale du carré de l'erreur liée à la position linéaire et à l'angle de rotation. Elle évalue l'efficacité du contrôleur flou à imposer

une trajectoire optimale au chariot tout en maintenant le pendule inversé en position verticale.

Le processus d'apprentissage est mené à l'aide d'un signal carré composé de deux périodes. Comme l'illustre la figure **??** , le contrôleur flou optimisé génère un signal de commande adapté, correspondant à la tension d'alimentation du moteur (proportionnelle à la force exercée). Ce signal assure un suivi précis de la trajectoire de référence du chariot tout en stabilisant efficacement le pendule inversé en position verticale.

$$ISE = \sum_{k=1}^{k} (e_x)^2 + \sum_{k=1}^{k} (\theta)^2 \tag{3.18}$$

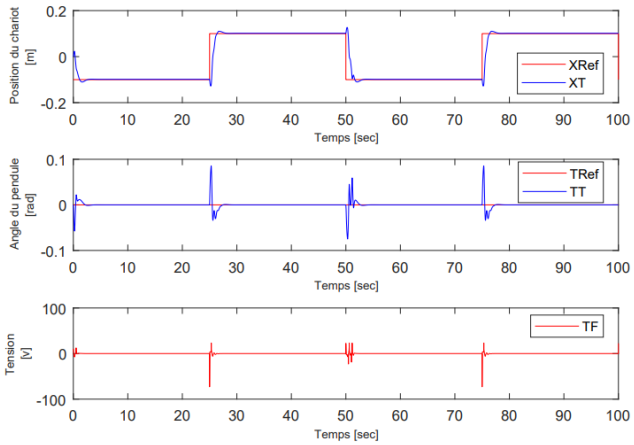

FIGURE 3.13 – Résultats pratique pour la phase d'apprentissage

3.5.3 Phase de test

Afin de tester la fiabilité du contrôleur optimisé face aux variations de la consigne, nous avons réalisé deux tests en utilisant des trajectoires de référence différentes pour le mouvement du chariot. Comme le montrent les figures 3.14 et 3.15, nous observons également des performances appréciables, ce qui démontre la capacité du contrôleur à maintenir la stabilité du pendule tout en suivant la trajectoire souhaitée.

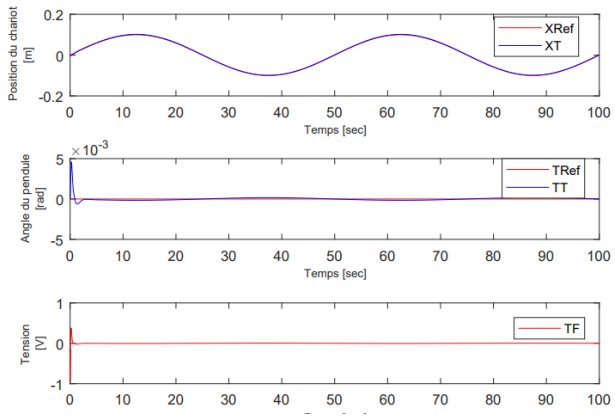

FIGURE 3.14 – Test par un signal sinusoidale

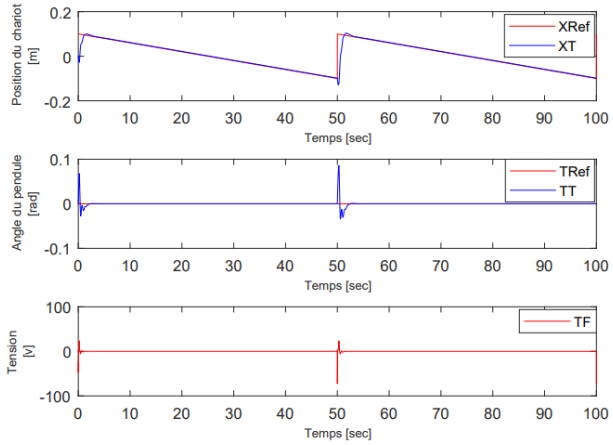

FIGURE 3.15 – Test par un signal en dents de scie

3.6 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approche d'optimisation des paramètres d'un système neuro-flou en utilisant une version améliorée de l'algorithme PSO. Cette version intègre deux paramètres adaptatifs : un facteur d'inertie qui diminue progressivement à chaque génération pour favoriser une convergence rapide et un terme externe supplémentaire conçu pour réduire le risque de piègeage dans un minimum local. Les résultats des simulations, tant pour l'entraînement que pour les tests, confirment l'efficacité de cette version améliorée du PSO.

Dans le chapitre suivant, cette version améliorée du PSO sera combinée avec l'algorithme FCM dans le but d'optimiser d'avantage la structure d'un système flou de type TSK.

CHAPITRE 4

Apprentissage de la structure du TSK

4.1 Introduction

Dans le cadre des algorithmes d'optimisation, notamment les métaheuristiques comme le PSO, l'initialisation aléatoire des solutions de départ peut souvent influencer la qualité et la convergence de la solution finale [66]. En effet, une initialisation inappropriée peut conduire l'algorithme vers des optima locaux, limitant ainsi son efficacité. Pour pallier ce problème, ce chapitre présente un algorithme hybride développé au cours de ces travaux de thèse, combinant le PSO et la méthode de Clustering Flou (FCM) [67].

4.2 Algorithme Fuzzy C-Means

L'algorithme FCM (Fuzzy C-Means) ([67], [68]) est une méthode de clustering flou qui permet de regrouper des données en plusieurs clusters en fonction de leurs similarités. Contrairement aux méthodes de clustering traditionnelles comme le k-means [69], où chaque donnée appartient strictement à un seul cluster, l'approche floue du FCM permet à chaque point de données d'appartenir à plusieurs clusters avec des degrés d'appartenance différents, représentés par des valeurs de probabilité. Cette flexibilité est particulièrement utile lorsque les frontières entre les clusters ne sont pas clairement définies.

4.2.1 Principe de fonctionnement de FCM

— Initialisation : on commence par choisir le nombre de clusters c et on initialise aléatoirement les centres des clusters ou les degrés d'appartenance de chaque point de données à chaque cluster.

— Degrés d'appartenance : chaque point de données est associé à chaque cluster avec un degré d'appartenance, généralement noté cu_{ij} qui indique dans quelle mesure le point x_i appartient au cluster j. Ces degrés d'appartenance sont compris entre 0 et 1, et leur somme, pour chaque point de données, est égale à 1.

— Mise à jour des centres de clusters : les centres des clusters sont recalculés en fonction des degrés d'appartenance, de manière à minimiser la distance pondérée entre les points de données et les centres de clusters décrite par l'équation 4.1. Le centre de chaque cluster est déterminé par la moyenne pondérée des points de données, pondérée par leurs degrés d'appartenance.

$$J_m = \sum_{j=1}^{M} \sum_{r=1}^{R} \mu_{jr}^{m} \|x_j - c_r\| \tag{4.1}$$

où

— J_m est un critère à minimiser (fonction objective).

— M est le nombre de points de données. On suppose que les M données peuvent être regroupées en R clusters (fonction d'appartenance) $R < M$.

— m est l'exposant de la matrice de partition floue pour contrôler le degré de recouvrement flou, avec $m > 1$.

— x_j est le j^{th} élément du vecteur de données$X$,

— c_r est le centre du r^{th} cluster

— μ_{jr} est le degré d'appartenance de x_j au x_j cluster.

— Mise à jour des degrés d'appartenance : les degrés d'appartenance des points de données à chaque cluster sont mis à jour en fonction des distances entre les points de données et les centres des clusters. Plus un point est proche du centre d'un cluster, plus son degré d'appartenance à ce cluster sera élevé.

— Itération : Les étapes de mise à jour des centres de clusters et des degrés d'appartenance sont répétées jusqu'à ce que la variation des centres de clus-

ters entre deux itérations consécutives soit inférieure à un seuil prédéfini, indiquant la convergence de l'algorithme.

Le principe de fonctionnement de l'algorithme FCM est donné par l'algorithme 6

Algorithm 6 Algorithme Fuzzy C-Means

1: Initialiser le nombre de cluster R
2: Initialiser de manière aléatoire les valeurs d'appartenance de clusterμ_{jr}
3: Calculer les centres des clusters : $c_r = \frac{\sum_{r=1}^{R}(\mu_{jr})^m x_j}{\sum_{r=1}^{R}(\mu_{jr})^m}$
4: mise à jour μ_{jr} en fonction de : $\mu_{jr} = \frac{1}{\sum_{k=1}^{R}(\frac{\|x_j - c_r\|}{\|x_j - c_k\|})^{\frac{2}{m-1}}}$
5: calculer la fonction objective J_m définie dans l'équation 4.1.
6: repeter l'étape $3-5$ jusqu'à J_m s'améliore en dessous d'un seuil minimal spécifié ou jusqu'à un nombre maximal d'itérations spécifié
7: Calculer les paramètres de normalisation pour chaque terme linguistique
8: $\sigma_{jr} = \frac{1}{j}\sum(x_j - c_r)^T(x_j - c_r)$

4.3 Algorithme hybride

Après avoir généré les données d'entrée et de sortie à l'aide des fonctions de référence, l'algorithme d'apprentissage proposé est présenté en trois étapes :

1. Générer les paramètres initiaux des fonctions d'appartenance et les conclusions en fonction des données d'entrée-sortie à l'aide de l'algorithme Fuzzy C-Means.
2. Construire la structure du modèle flou TSK.
3. Éliminer les règles inutiles en suivant un critère (voir l'équation 4.2) basé sur les poids de chaque règle.

$$\begin{cases} w_i = \dfrac{\sum_{j=1}^{M} w_j^i}{M} \\ if\ w_i < \gamma\ then\ w_i = 0 \\ else \\ w_i = w_i \end{cases} \tag{4.2}$$

où w_i présente le poids de la i^{eme} règle floue

4.3.1 Première étape

Dans cette étape, le Fuzzy C-Means (FCM), en tant que méthode de clustering flou, a été utilisé pour répartir efficacement les ressources d'entrée-sortie et pour déterminer les centres des clusters en tenant compte des similarités entre les données et en attribuant des degrés d'appartenance à plusieurs clusters en plusieurs sous-ensembles flous (fonctions d'appartenance). Tout cela, pour construire les paramètres initiaux des fonctions d'appartenance pour chaque particule de l'essaim dans le but d'avoir un bon départ.

4.3.2 Seconde étape

Une fois que les paramètres des fonctions d'appartenance ont une bonne initialisation, la deuxième étape sera de procéder à la construction de notre modèle flou. Le nombre de règles floues est égal au produit entre les nombres de fonctions d'appartenance pour chaque entrée. La structure générale du modèle flou est présentée par la figure 4.1.

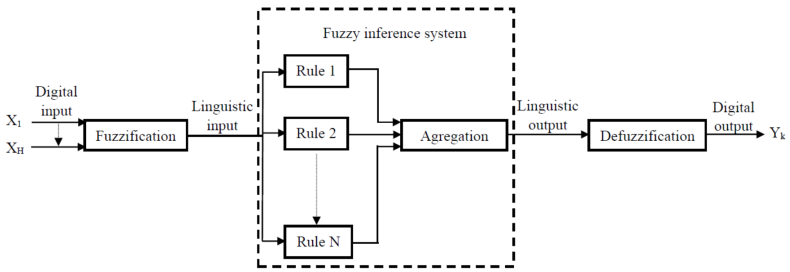

FIGURE 4.1 – Structure du modèle flou

Dans la fuzzification, la fonction d'appartenance est utilisée pour fuzzifier la valeur d'entrée et obtenir l'appartenance floue correspondante. Le système d'inférence floue calcule le poids w_i de chaque règle ainsi que sa sortie y_i. Enfin, la défuzzification est utilisée pour calculer et contrôler la sortie totale du système flou.

4.3.3 Troisième étape

Après avoir ajusté les paramètres initiaux, toutes les particules sont évaluées par la fonction 3.10. Nous calculons la nouvelle position de chaque particule à l'aide de l'équation 3.7. La principale contribution de cette nouvelle approche réside dans

l'adaptation dynamique des paramètres du modèle flou présenté par l'équation 3.12, ainsi que dans la minimisation de son architecture. À chaque itération, les paramètres du modèle flou sont évalués par le PSO afin de rechercher le meilleur modèle (la meilleure particule).

Le PSO utilise l'opérateur *min* pour extraire la valeur minimale du vecteur de coût *EQM* et calculer sa moyenne. Nous considérons toutes les particules comme des minima locaux et leurs coûts comme des minima locaux. L'extraction de la valeur minimale du vecteur de coût est utilisée pour localiser la meilleure particule (le minimum global) $global_{par}$ dans la matrice de particules *Par* de l'algorithme 7.

Algorithm 7 Algorithm PSO

Initialization $C1 = C2, C3, C, r1, r2, maxit$
$\lambda = \frac{1}{max(eig(XX^T))}$
charger les paramèters des MFs trouvé par algorithme 6
$min_{cost} = min(EQM); mean_{cost} = mean(EQM)$
$global_{cost} = min_{cost}; local_{par} = Par_q; local_{cost} = EQM$
$[global_{cost}, index] = min(EQM) \rightarrow global_{par} = Par_q(index, :)$
$k = 0$
while $k < maxit$ **do**
    $k = k + 1$
    $W = (maxit - k)/maxit \Rightarrow$ (équation 3.10)$\Rightarrow$(équation 3.7)
    $local_{cost} = local_{cost} * not(better_{cost}) + E * better_{cost}$
    $[temp, t] = min(local_{cost})$
    if $temp < global_{cost}$ **then**
        $global_{par} < Par_q(t, :) \rightarrow index = t \rightarrow global_{cost} = temp$
    end if
    élimination des règles utilisant critère de l'équation 4.2
    $[k\ global_{par}\ global_{cost}]$
    $min_{cost}(k + 1) = min(EQM) \rightarrow global_{min}(k + 1) = global_{cost}$
    $mean_{cost}(k + 1) = mean(EQM)$
end while

4.4 Fonction objectif

Soit $[X_1, \cdots, X_H] \in \mathbb{R}^{M*H}$, ($X_1, \cdots, X_H$ sont des vecteurs lignes). Le j^{me} élément de chaque vecteur X est collecté à l'instant $j \times Te$ (Te : période d'échantillonnage).

Dans le cas de la modélisation d'un système à l'aide d'un algorithme itératif, l'objectif est de minimiser le critère de l'erreur quadratique moyenne.

L'erreur est la différence entre les mesures collectées y_m et le modèle y à l'itération

k donnée par l'équation :

$$EQM_k = \frac{\sum_{j=1}^{M}(y_k(j) - y_m(j))^2}{M} \tag{4.3}$$

Ce chapitre propose de simplifier la complexité du modèle en réduisant le nombre de règles floues *SI−ALORS*. Conformément à [70] et [71], qui ont montré l'importance du poids de chaque règle floue dans la base des règles floues, le mécanisme proposé fonctionne comme suit :

A chaque itération k, la moyenne d'activation w_i pour chaque règle R^i est calculée pour tous les vecteurs d'entrée X du meilleur modèle $global_{par}$ trouvé jusqu'à présent. Nous éliminons les règles dont la valeur de poids est inférieure à la valeur seuil γ en utilisant l'équation 4.2

4.5 Resultats et discussions

Tous les résultats de simulation et les figures sont obtenus à l'aide de MATLAB 2016.

Cette section montre les performances de l'approche proposée dans notre article [72] à l'aide des deux problèmes classiques d'identification proposés par [64] et étudiés dans le chapitre 3. Pour chaque système, le modèle flou TSK a trois entrées $u(k)$, $y(k-1)$ et $y(k-2)$.

Pour comparer les performances de la technique développée avec des contributions antérieures ([37], [73], [74], [19], [75], [76], [77] et [78]), l'implémentation à l'aide de MATLAB a été réalisée avec les mêmes paramètres : taille de la population = 60 et nombre de générations = 1000. Après plusieurs essais, nous avons déterminé que la meilleure valeur pour γ est 0.1.

4.5.1 Premier test

$$y(k) = \frac{y(k-1)y(k-2)(y(k-1)+2.5)}{1+y^2(k-1)+y^2(k-2)} + u(k) \tag{4.4}$$

Dans ce test, chaque variable d'entrée est partitionnée en trois ensembles flous (A_i, B_i, C_i) avec une forme gaussienne. Le partitionnement des ensembles flous est

effectué au début de notre algorithme en utilisant l'algorithme Fuzzy C-Means pour chaque entrée. Avec une structure de trois fonctions d'appartenance, le nombre total de règles est de 27, avec 126 paramètres à identifier (18 paramètres pour les antécédents et 108 pour les conclusions des règles). Les données d'apprentissage, illustrées par la figure 4.2, consistent en 300 points générés par une séquence d'impulsions avec une amplitude aléatoire dans l'intervalle [−1,1] et une période aléatoire entre 1 et 4.

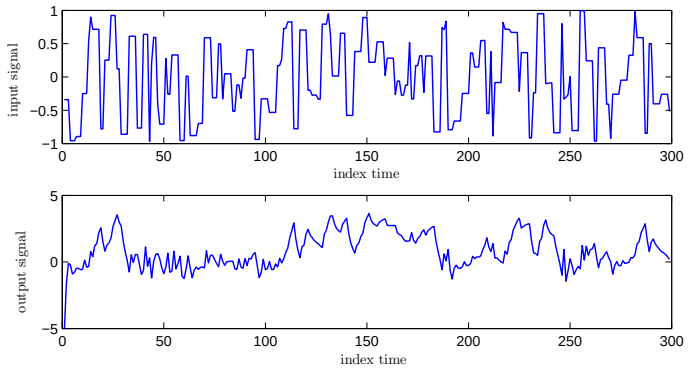

FIGURE 4.2 – Signaux d'entrée et de sortie des données d'apprentissage

Cette phase d'apprentissage est conçue pour permettre l'ajustement automatique de tous les paramètres des règles floues.

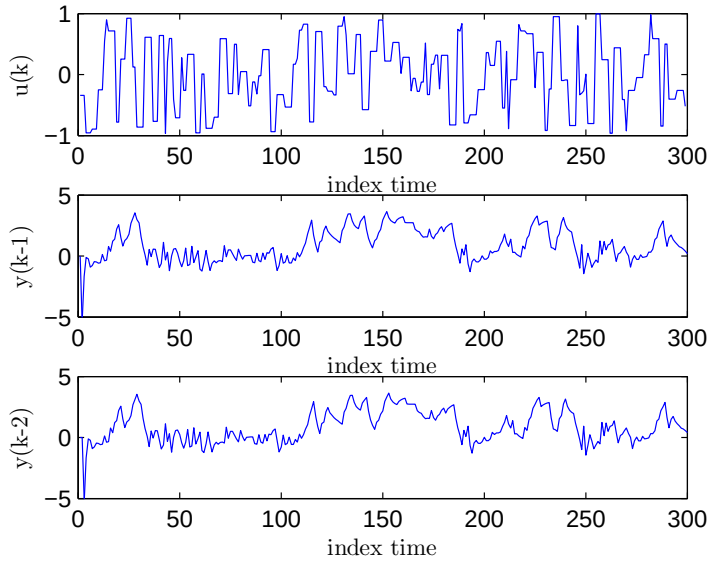

FIGURE 4.3 – Signaux d'entrée u(k), y(k-1) et y(k-2)

Nous constatons que la phase d'initialisation utilisant l'algorithme Fuzzy C-Means est meilleure que l'initialisation aléatoire. En effet, pour 10 exécutions suc-

cessives, l'erreur quadratique moyenne (EQM) dans le cas de l'algorithme Fuzzy C-Means prend ses valeurs entre 0.5196 et 0.6880, alors que l'EQM dans le cas de l'initialisation aléatoire prend ses valeurs entre 0.8361 et 1.223. On constate que la valeur minimale de l'EQM dans le cas de l'initialisation aléatoire est supérieure à la valeur maximale de l'EQM dans le cas de l'algorithme Fuzzy C-Means, ce qui indique que le pire cas de l'algorithme Fuzzy C-Means est meilleur que le meilleur cas de l'initialisation aléatoire. La figure 4.3 montre les signaux d'entrée.

La figure 4.4 montre les fonctions d'appartenance générées par l'algorithme FCM.

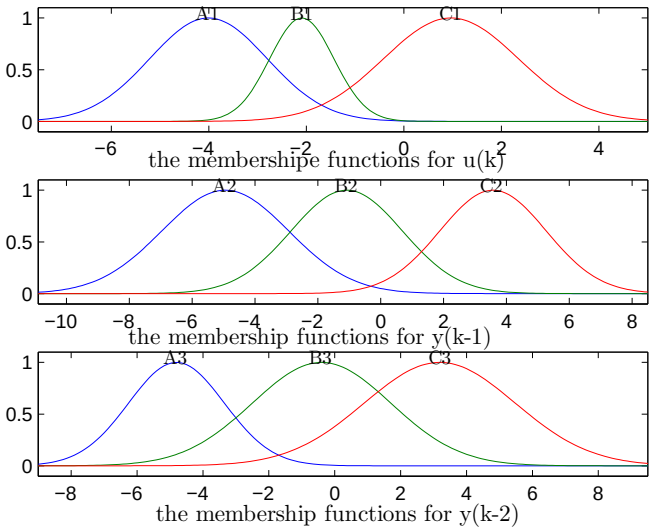

FIGURE 4.4 – Fonctions d'appartenance initiales avant l'apprentissage

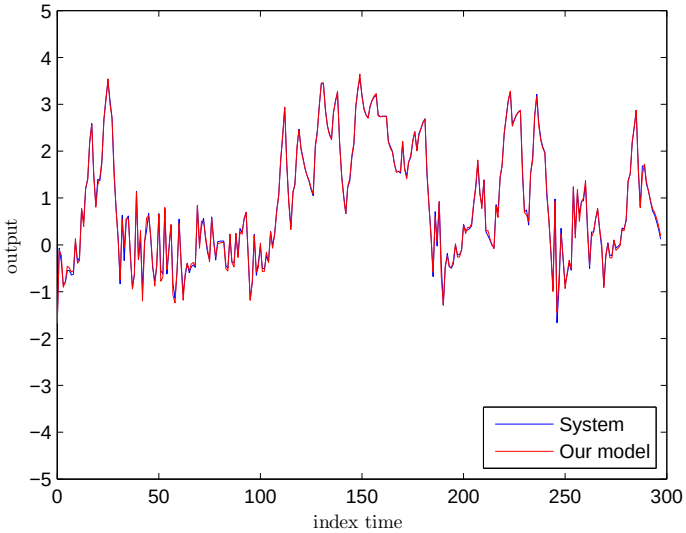

FIGURE 4.5 – Sortie du système et de notre modèle après l'apprentissage

Dans ce qui suit, l'algorithme hybride proposé exploite les paramètres initiaux générés par l'algorithme Fuzzy C-Means. Les paramètres utilisés par le PSO sont comme suit : C=0.7, C_1=2.1, C_2=1.9.

D'après la figure 4.5, le signal de sortie du modèle flou identifié par l'approche proposée correspond entièrement et exactement à la sortie du système réel simulé et démontre une plus grande précision de prédiction supérieure par rapport aux modèles présentés dans le tableau 4.1.

TABLEAU 4.1 – Comparaison de l'approche proposée avec d'autres approches pour le premier test

réference	Nombre de règle	Nombre de paramèters	taille de la population	Nombre de générations	EQM_{train}	EQM_{test}
[37]	8	40	60	1000	0.0132	0.01200
[73]	8	80	60	1000	0.0098	0.01150
[74]	22	32	60	1000	0.0077	0.00514
[19]	27	126	60	1000	0.0096	0.00648
[75]	5	27	60	1000	0.00636	0.0026
[76]	4	40	60	1000	0.0059	0.0072
[77]	8	44	60	1000	0.005	0.025
[78]	4	40	60	1000	/	0.0235
Notre modèle	4 ($\gamma = 0.1$)	28	60	1000	0.0029	0.0015
	18 ($\gamma = 0.5$)	78	60	1000	$9.5 * 10^{-4}$	$7.3 * 10^{-4}$
	27 ($\gamma = 1$)	126	60	1000	$6.8 * 10^{-4}$	$4.7 * 10^{-4}$

Pour vérifier l'efficacité de la stratégie proposée, l'algorithme hybride a été simulé pour trois valeurs différentes de γ (0.1, 0.5 et 1) comme indiqué dans les trois dernières lignes du tableau 4.1. On constate à chaque fois une meilleure précision de nos modèles par rapport à ceux de la littérature. La figure 4.6 donne l'évolution de la fonction objectif à travers les générations pour le cas γ=0.1.

Les dixième (γ=0.5) et onzième (γ=1) lignes présentent les résultats d'entraînement et de test obtenus à l'aide de l'algorithme hybride. Bien que ces résultats soient meilleurs en termes de précision, le nombre de règles et de paramètres intégrés dans leurs modèles respectifs reste relativement élevé, ce qui peut entraîner un temps de calcul considérable. Pour comparer notre algorithme avec les précédents dans les mêmes conditions (nombre de règles et de paramètres), nous considérons le cas de γ=0.1. Les valeurs de EQM_{train}=0.0029 indiquées à la neuvième ligne du tableau 4.1

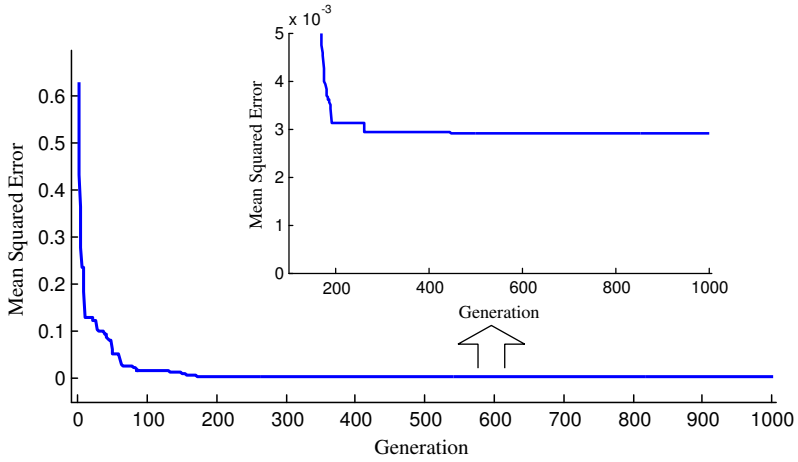

FIGURE 4.6 – Sortie du système et de notre modèle après l'apprentissage

montrent que l'algorithme hybride FCM-PSO peut extraire un modèle flou précis avec un nombre approprié de règles floues où il ne conserve que 4 règles floues avec 28 paramètres comme indiqué dans le tableau 4.1, où les règles floues restantes sont :

— R^3 : If $u(k)$ is $A_1(c = -0.1566, \sigma = 0.2020)$ and $y(k-1)$ is $A_2(c = -0.3983, \sigma = 0.0246)$ and $y(k-2)$ is $A_3(c = 3.2670, \sigma = 2.2187)$

Then $y^3 = 0.0559 + 0.0047 * u(k) - 1.0563 * y(k-1) + 0.8186 * y(k-2)$

— R^6 : If $u(k)$ is $A_1(c = -0.1566, \sigma = 0.2020)$ and $y(k-1)$ is $B_2(c = -0.6736, \sigma = 0.7767)$ and $y(k-2)$ is $C_3(c = -0.2799, \sigma = 1.1058)$

Then $y^6 = 1.2950 + 0.9761 * u(k) + 0.0191 * y(k-1) + 0.3325 * y(k-2)$

— R^7 : If $u(k)$ is $A_1(c = -0.1566, \sigma = 0.2020)$ and $y(k-1)$ is $C_2(c = -0.2010, \sigma = 4.5633)$ and $y(k-2)$ is $A_3(c = 3.2670, \sigma = 2.2187)$

Then $y^7 = 0.1464 - 0.2646 * u(k) - 1.8230 * y(k-1) + 0.0633 * y(k-2)$

— R^9 : If $u(k)$ is $A_1(c = -0.1566, \sigma = 0.2020)$ and $y(k-1)$ is $C_2(c = -0.2010, \sigma = 4.5633)$ and $y(k-2)$ is $C_3(c = -0.2799, \sigma = 1.1058)$

Then $y^9 = 0.5564 - 0.0288 * u(k) + 1.3792 * y(k-1) - 0.3642 * y(k-2)$

En effet, l'approche proposée est clairement la meilleure par rapport à l'ensemble des 8 algorithmes mentionnés dans le tableau 4.1, où l'erreur d'apprentissage est réduite de moitié par rapport à celle obtenue par [77].

Lorsque le processus d'apprentissage est terminé, un signal d'entrée sinusoïdal $u(k) = \sin(2\pi k/25)$ est appliqué pour tester la capacité de généralisation du modèle obtenu. La sortie du système et celle du modèle sont illustrées à la figure 4.7.

La figure 4.7 montre que le modèle obtenu est capable de reproduire la sortie du

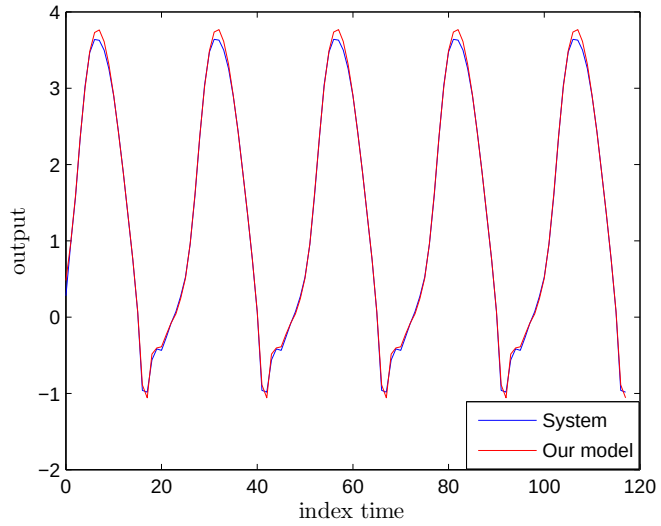

FIGURE 4.7 – Signaux de sortie du système et du modèle après le test

système avec une erreur EQM_{test} de 0.0015, bien inférieure à celles des travaux récents ([75], [76] et [78]) qui sont respectivement de 0.0026, 0.0072 et 0.0235. Il convient de souligner que le nombre de règles et de paramètres est très similaire dans ce travail (4 règles et 28 paramètres) et dans ceux de [76] et [78] (4 règles et 40 paramètres).

4.5.2 Deuxième test

Un deuxième système est utilisé pour montrer que l'approche proposée peut être appliquée à des systèmes qui ont une forte non-linéarité telle que :

$$y(k) = \frac{24 + y(k-1)}{30} y(k-1) - 0.8 \frac{u^2(k-1)}{1 + u^2(k-1)} y(k-2) + 0.5u(k-1) \qquad (4.5)$$

Les 400 données d'entrée/sortie sont générées à partir d'une série d'impulsions aléatoires continues $u(k)$ avec une période d'échantillonnage comprise entre [1 et 20] et une amplitude comprise entre [-5 et 5], les 200 premiers points étant utilisés pour la phase d'apprentissage et les 200 points restants pour la phase de test, comme le montrent les figures 4.8 et 4.9, respectivement.

Dans ce cas, nous avons divisé les trois entrées comme suit : deux entrées $y(k-1)$, $y(k-2)$ sont divisées en trois sous-ensembles flous tandis que l'entrée $u(k-1)$ est répartie en 5 ensembles flous.

La base de règles du modèle flou contient $5 * 3 * 3 = 45$ règles floues et 202

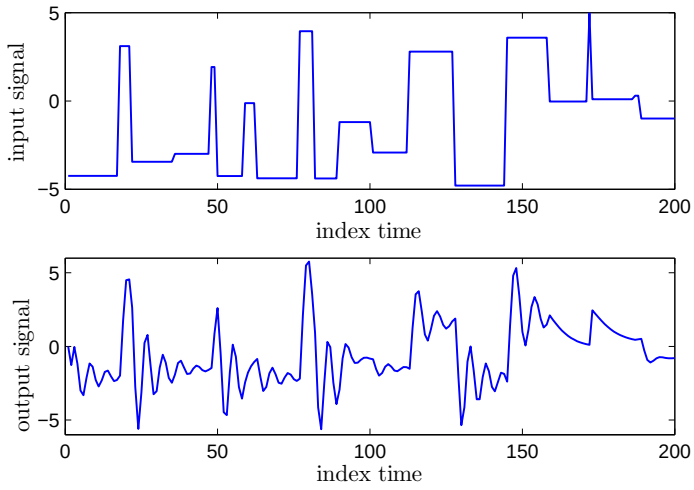

FIGURE 4.8 – Données d'entraînement

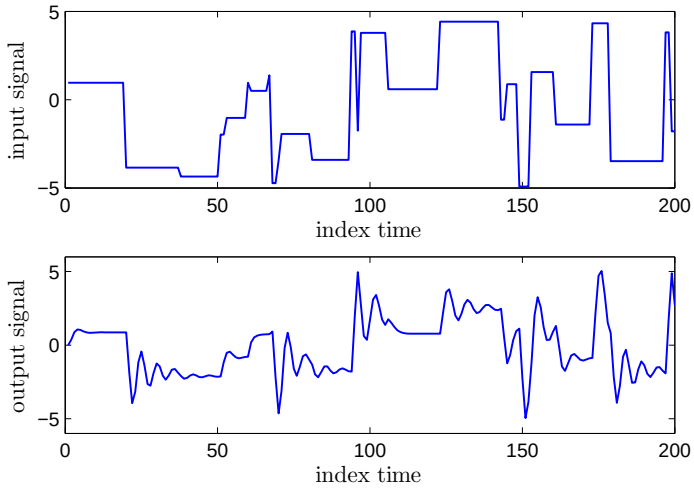

FIGURE 4.9 – Données d'essais

paramètres,$45 * 4$ ce sont les paramètres des conclusions, et $(5 + 3 + 3) * 2 = 22$ ce sont les paramètres des fonctions d'appartenance.

Dans le tableau 4.2, la dixième ligne (γ=1) montre une très bonne performance de l'approche proposée. Cependant, et comme mentionné ci-dessus, la comparaison avec les travaux précédents doit être réalisée dans les mêmes conditions (nombre de règles et paramètres proches).

La neuvième ligne (γ=0.5) présente également de meilleurs résultats par rapport à ceux de [75], avec un nombre de règles (15) et de paramètres (66) moins importants. Les tests en septième et huitième lignes ont été réalisés avec (γ=0.3) et (γ=0.1) et ont donné de bonnes performances par rapport à [75] et [19], avec un très faible nombre de règles (3) et de paramètres (18) dans le cas de γ=0.1. D'après la figure 4.10, on

TABLEAU 4.2 – Comparaison de l'approche proposée avec d'autres tests (deuxième test)

référence	Nombre de de règle	Nombre paramèters	la taille de la population	Nombre de générations	EQM_{train}	EQM_{test}
[37]	8	40	60	1000	0.0201	0.0884
[73]	8	80	60	1000	0.0153	0.0822
[74]	30	36	60	1000	0.0083	0.0221
[19]	45	202	60	1000	0.0149	0.0510
[75]	17	82	60	1000	0.0061	0.0440
[77]	8	44	60	1000	0.013	0.301
Notre modèle	3 ($\gamma = 0.1$)	18	60	1000	0.0520	0.088
	7 ($\gamma = 0.3$)	34	60	1000	0.00905	0.0182
	15 ($\gamma = 0.5$)	66	60	1000	0.0041	0.0078
	45 ($\gamma = 1$)	202	60	1000	$6.78 * 10^{-6}$	$8.54 * 10^{-5}$

peut conclure qu'il y a une bonne concordance entre la sortie du modèle flou et la sortie originale dans la phase d'apprentissage.

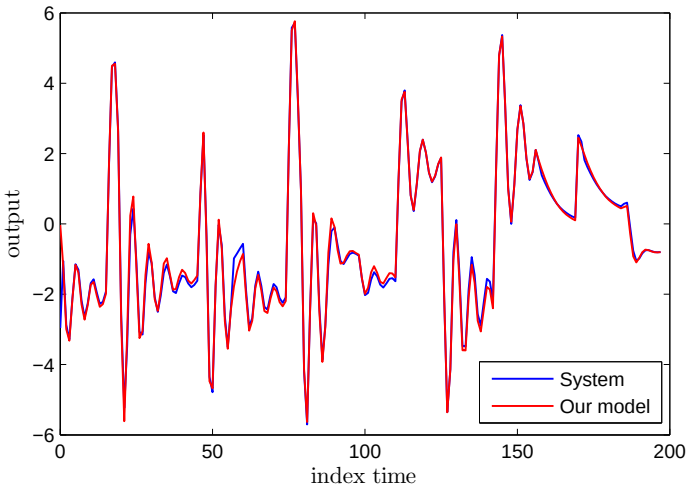

FIGURE 4.10 – Apprentissage de la sortie

En outre, la figure 4.11 montre de très bonnes performances de l'approche proposée pour les phases de test pour γ=0.5.

La performance de l'EQM présentée dans la figure 4.12 du modèle flou TSK avec 15 règles est meilleure que celles obtenues par les autres modèles. Dans notre méthode, l'EQM atteint la valeur de $4.1 * 10^{-3}$, alors que la meilleure valeur donnée par les autres méthodes est de $6.1 * 10^{-3}$.

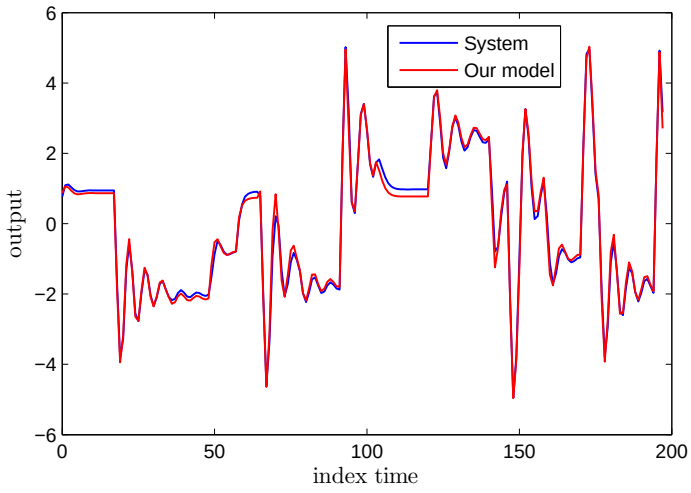

FIGURE 4.11 – Test de la sortie

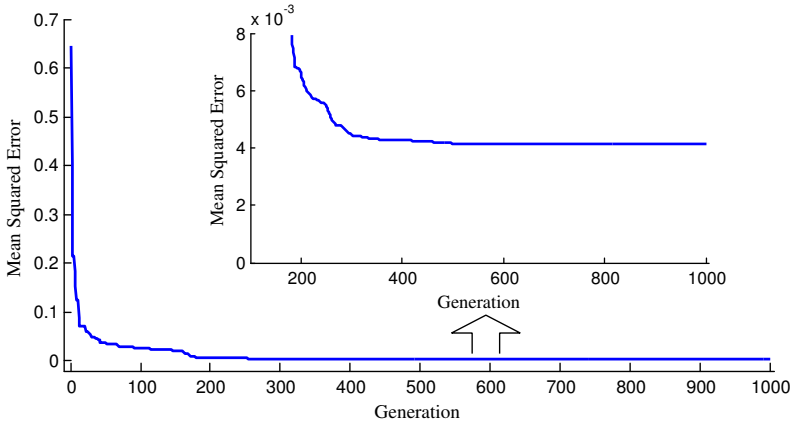

FIGURE 4.12 – Sortie du système et de notre modèle après l'apprentissage

4.6 Conclusion

Ce chapitre présente une nouvelle approche combinant l'algorithme Fuzzy C-Means (FCM) avec l'algorithme PSO. L'utilisation de l'algorithme FCM permet une meilleure initialisation des paramètres des fonctions d'appartenance par rapport à une initialisation aléatoire. Les résultats d'apprentissage et de test, obtenus à l'aide de deux systèmes de référence, montrent que l'algorithme PSO modifié est plus efficace, facile à mettre en œuvre, et capable de traiter des systèmes caractérisés par une forte non-linéarité ou un vaste espace de recherche.

Le critère de sélection des règles modifié, basé sur le poids de chaque règle floue, s'avère à la fois simple et plus efficace que les critères utilisés dans les algorithmes précédents mentionnés dans ce travail. Ce critère permet d'éliminer les règles inutiles tout en conservant un sous-ensemble réduit de règles à identifier avec l'algorithme

PSO, ce qui entraîne un temps de calcul réduit tout en maintenant une précision élevée du modèle.

En d'autres termes, notre algorithme offre des performances supérieures sans compromettre le temps de calcul, ce qui le rend adapté à la modélisation pratique des systèmes. Cette approche permet d'extraire un modèle flou TSK qui atteint simultanément une grande précision et un nombre minimal de règles, faisant ainsi de cet outil un choix efficace pour la modélisation de systèmes non linéaires complexes.

Conclusion générale

Cette thèse porte sur la problématique de l'apprentissage automatique et du contrôle des systèmes non linéaires complexes. Elle vise spécifiquement à concevoir des systèmes flous de type TSK dotés d'une grande précision, en exploitant les algorithmes évolutionnaires. La démarche adoptée consiste en un traitement progressif et structuré, abordant les différentes facettes du problème de manière graduelle afin d'en faciliter la compréhension et l'assimilation.

Dans un premier temps, une description approfondie des systèmes flous a été réalisée, mettant en avant leur forme fondamentale ainsi que les différents outils et principes guidant leur conception. Ensuite, une exploration a été menée sur les diverses combinaisons possibles entre les systèmes flous et les réseaux de neurones, soulignant les approches hybrides et leurs avantages dans le traitement des systèmes complexes.

Dans un second temps, ce manuscrit explore les outils dédiés à la conception automatisée des systèmes flous. Il met en lumière comment les algorithmes évolutionnaires peuvent être exploités pour améliorer les processus d'apprentissage et de contrôle automatique des systèmes flous de type TSK. Ces méthodes reposent sur plusieurs variantes largement répandues des algorithmes mono-objectifs décrites dans la littérature, offrant ainsi des solutions robustes et adaptatives pour optimiser les performances des systèmes flous.

Une version optimisée de l'algorithme PSO, reposant sur deux paramètres adaptatifs indépendants, est détaillée dans le troisième chapitre. Cette amélioration offre une exploration plus approfondie de l'espace de recherche, permettant une optimisation simplifiée et efficace des paramètres des règles floues. Les résultats obtenus démontrent un potentiel significatif, aussi bien pour des applications théoriques, en

soulignant l'efficacité et la flexibilité de cette méthode.

Dans le dernier chapitre, nous avons abordé non seulement la minimisation des paramètres des règles floues, mais également l'optimisation du nombre de règles. Pour ce faire, un algorithme hybride FCM-PSO a été proposé, reposant sur deux étapes clés : une initialisation optimisée des positions des particules du PSO et l'élimination des règles floues superflues. L'efficacité de cet algorithme a été démontrée à travers des applications théoriques et une application pratique emblématique : le contrôle du pendule inversé.

Bibliography

[1]. Yong-ming Li, Xiao Min, and Shaocheng Tong. Adaptive fuzzy inverse optimal control for uncertain strict-feedback nonlinear systems. IEEE Transactions on Fuzzy Systems, 28(10) :2363–2374, 2019.

[2]. Zhifu Cao, Qingguo Fei, Dong Jiang, Rui Zhu, and Hui Jin. Sensitivity analysis of nonlinear transient response based on perturbation in the complex domain. Journal of Computational and Nonlinear Dynamics, 16(1) :011001, 2021.

[3]. Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. Nature communications, 9(1) :4950, 2018.

[4]. Giuseppe Quaranta, Walter Lacarbonara, and Sami F Masri. A review on computational intelligence for identification of nonlinear dynamical systems. Non-linear Dynamics, 99(2) :1709–1761, 2020.

[5]. Lotfi A Zadeh. Fuzzy sets. Information and control, 8(3) :338–353, 1965.

[6]. Ali Guidara and Ali Guidara. Artificial intelligence and fuzzy logic. Policy Decision Modeling with Fuzzy Logic : Theoretical and Computational Aspects, pages 47–67, 2021.

[7]. Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. IEEE transactions on systems, man, and cybernetics, (1) :116–132, 1985.

[8]. Rafał Scherer and Rafał Scherer. Takagi-sugeno fuzzy systems. Multiple Fuzzy Classification Systems, pages 73–79, 2012.

[9]. Xiangming Gu and Xiang Cheng. Distilling a deep neural network into a takagi-sugeno-kang fuzzy inference system. arXiv preprint arXiv :2010.04974, 2020.

[10]. David E Goldberg. Genetic algorithms in search, optimisation and machine learning, 1989. Reading, Addison, Wesley, 1989.

[11]. Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In MHS’95. Proceedings of the sixth international symposium on micro machine and human science, pages 39–43. Ieee, 1995.

[12]. Marco Dorigo. Ant colony optimization. Scholarpedia, 2(3) :1461, 2007.

[13]. Mehmet Kaya and Reda Alhajj. Utilizing genetic algorithms to optimize membership functions for fuzzy weighted association rules mining. Applied Intelligence, 24 :7–15, 2006.

[14]. Chi-Chung Chen. Optimization of zero-order tsk-type fuzzy system using enhanced particle swarm optimizer with dynamic mutation and special initialization. International Journal of Fuzzy Systems, 20 :1685–1698, 2018.

[15]. Elsaid Md Abdelrahim. Binary particle swarm optimization-based ts fuzzy predictive controller for nonlinear automotive application. Neural Computing and Applications, 33(7) :2803–2818, 2021.

[16]. Dinh Sinh Mai, Kien-Trinh Thi Bui, and Chinh Van Doan. Application of interval type-2 fuzzy logic system and ant colony optimization for hydropower dams displacement forecasting. International Journal of Fuzzy Systems, 25(5) :2052–2066, 2023.

[17]. Afshin Faramarzi, Mohammad Heidarinejad, Brent Stephens, and Seyedali Mirjalili. Equilibrium optimizer : A novel optimization algorithm. Knowledge-based systems, 191 :105190, 2020.

[18]. Esmat Rashedi, Hossein Nezamabadi-Pour, and Saeid Saryazdi. Gsa : a gravitational search algorithm. Information sciences, 179(13) :2232–2248, 2009.

[19]. Mehmet Ali Cavuslu, Cihan Karakuzu, and Fuat Karakaya. Neural identification of dynamic systems on fpga with improved pso learning. Applied Soft Computing, 12(9) :2707–2718, 2012.

[20]. Fabien Lauer and Gérard Bloch. Hybrid system identification : Theory and algorithms for learning switching models, volume 478. Springer, 2018.

[21]. Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications : A survey. Heliyon, 4(11), 2018.

[22]. Simone Spolaor, Marco S Nobile, Giancarlo Mauri, Paolo Cazzaniga, and Daniela Besozzi. Coupling mechanistic approaches and fuzzy logic to model and simulate complex systems. IEEE Transactions on Fuzzy Systems, 28(8) :1748–1759, 2019.

[23]. Lofti Zadeh. Optimality and non-scalar-valued performance criteria. IEEE transactions on Automatic Control, 8(1) :59–60, 1963.

[24]. Mamdani. Application of fuzzy logic to approximate reasoning using linguistic synthesis. IEEE transactions on computers, 100(12) :1182–1191, 1977.

[25]. Witold Pedrycz. Identification in fuzzy systems. IEEE transactions on systems, man, and cybernetics, (2) :361–366, 1984.

[26]. Soo Yeong Yi and Myung Jin Chung. Identification of fuzzy relational model and its application to control. Fuzzy sets and systems, 59(1) :25–33, 1993.

[27]. Homayun Motameni, Ali Ebrahimnejad, Javad Vahidi, et al. Morphology of composition functions in persian sentences through a newly proposed classified fuzzy method and center of gravity defuzzification method. Journal of Intelligent and Fuzzy Systems, 36(6) :5463–5473, 2019.

[28]. Bin-Da Liu, Chuen-Yau Chen, and Ju-Ying Tsao. Design of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 31(1) :32–53, 2001.

[29]. Ebrahim H Mamdani and Sedrak Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. International journal of man-machine studies, 7(1) :1–13, 1975.

[30]. Puyin Liu and Hong-Xing Li. Fuzzy neural network theory and application, volume 59. World Scientific, 2004.

[31]. Detlef D Nauck and Andreas Nürnberger. Neuro-fuzzy systems : A short historical review. In Computational intelligence in intelligent data analysis, pages 91–109. Springer, 2013.

[32]. Noureen Talpur, Said Jadid Abdulkadir, Hitham Alhussian, Mohd Hilmi Hasan, Norshakirah Aziz, and Alwi Bamhdi. Deep neuro-fuzzy system application trends, challenges, and future perspectives : A systematic survey. Artificial intelligence review, 56(2) :865–913, 2023.

[33]. Anders Krogh. What are artificial neural networks ? Nature biotechnology, 26(2) :195–197, 2008.

[34]. Ajith Abraham. Neuro fuzzy systems : Sate-of-the-art modeling techniques. arXiv preprint cs/0405011, 2004.

[35]. WL Tung and Chai Quek. Falcon : neural fuzzy control and decision systems using fkp and pfkp clustering algorithms. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 34(1) :686–695, 2004.

[36]. Andreas Nürnberger, Detlef Nauck, and Rudolf Kruse. Neuro-fuzzy control based on the nefcon-model : recent developments. Soft Computing, 2 :168–182, 1999.

[37]. J-SR Jang. Anfis : adaptive-network-based fuzzy inference system. IEEE transactions on systems, man, and cybernetics, 23(3) :665–685, 1993.

[38]. Nyoman Gunantara. A review of multi-objective optimization : Methods and its applications. Cogent Engineering, 5(1) :1502242, 2018.

[39]. Thomas Weise. Global optimization algorithms-theory and application. Self-Published Thomas Weise, 361 :153, 2009.

[40]. Urmila M Diwekar. Introduction to applied optimization, volume 22. Springer Nature, 2020.

[41]. Seyed Hamed Javadi and Seyed Hamid Zahiri. A central force optimization approach for data clustering. IETE Journal of Research, pages 1–9, 2018.

[42]. Janmenjoy Nayak, Sura Paparao, Bighnaraj Naik, N Seetayya, P Pradeep, HS Behera, and Danilo Pelusi. Chemical reaction optimization : a survey with application and challenges. In Soft Computing in Data Analytics : Proceedings of International Conference on SCDA 2018, pages 507–524. Springer, 2019.

[43]. Dimitris Bertsimas and John Tsitsiklis. Simulated annealing. Statistical science, 8(1) :10–15, 1993.

[44]. Eric Bonabeau. Swarm intelligence. A Primer on Multiple Intelligences, page 211, 2003.

[45]. Jagdish Chand Bansal, Harish Sharma, and Shimpi Singh Jadon. Artificial bee colony algorithm : a survey. International Journal of Advanced Intelligence Paradigms, 5(1-2) :123– 59, 2013.

[46]. Holland John. Holland. genetic algorithms. Scientific american, 267(1) :44–50, 1992.

[47]. William M Spears and Kenneth A De Jong. An analysis of multi-point crossover. In Foundations of genetic algorithms, volume 1, pages 301–315. Elsevier, 1991.

[48]. Hamza Khalfi, Nour Eddine Alaa, and Mohammed Guedda. Period steady-state identification for a nonlinear front evolution equation using genetic algorithms. International Journal of Bio-Inspired Computation, 12(3) :196–202, 2018.

[49]. Joseph L Pachuau, Arnab Roy, and Anish Kumar Saha. An overview of crossover techniques in genetic algorithm. Modeling, Simulation and Optimization : Proceedings of CoMSO 2020, pages 581–598, 2021.

[50]. DP Kanungo, Bighnaraj Naik, Janmenjoy Nayak, Sarada Baboo, and HS Behera. An improved pso based back propagation learning-mlp (ipso-bp-mlp) for classification. In Computational Intelligence in Data Mining-Volume 1 : Proceedings of the International Conference on CIDM, 20-21 December 2014, pages 333–344. Springer, 2015.

[51]. Zenadji Sylia, Cédric Gueguen, Brikh Lamine, Talbi Larbi, and Abdelkrim Khireddine. New strategy for resource allocation using pso-pfs hybrid. International Journal of Wireless and Mobile Computing, 18(2) :175–182, 2020.

[52]. Mohand Akli Kacimi, Ouahib Guenounou, Lamine Brikh, Fateh Yahiaoui, and Nouh Hadid. New mixed-coding pso algorithm for a self-adaptive and automatic learning of mamdani fuzzy rules. Engineering Applications of Artificial Intelligence, 89 :103417, 2020.

[53]. Ahmed A Adeniran and Sami El Ferik. A reinforced combinatorial particle swarm optimization based multimodel identification of nonlinear systems. AI EDAM, 31(3) :327–358, 2017.

[54]. Hilal Labdelaoui, Fares Boudjema, and Djamel Boukhetala. A multiobjective tuning approach of power system stabilizers using particle swarm optimization. Turkish Journal of Electrical Engineering and Computer Sciences, 24(5) :3898–3909, 2016.

[55]. Raheleh Jafari and Wen Yu. Fuzzy modeling for uncertainty nonlinear systems with fuzzy equations. Mathematical problems in Engineering, 2017(1) :8594738, 2017.

[56]. Adrian A Hopgood. Intelligent systems for engineers and scientists : a practical guide to artificial intelligence. CRC press, 2021.

[57]. Mohammad Biglarbegian, William W Melek, and Jerry M Mendel. On the stability of interval type-2 tsk fuzzy logic control systems. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 40(3) :798–818, 2009.

[58]. John Yen, Liang Wang, and Charles Wayne Gillespie. Improving the interpretability of tsk fuzzy models by combining global learning and local learning. IEEE Transactions on fuzzy Systems, 6(4) :530–537, 1998.

[59]. Babak Rezaee and MH Fazel Zarandi. Data-driven fuzzy modeling for takagi–sugeno–kang fuzzy system. Information Sciences, 180(2) :241–255, 2010.

[60]. Radu-Emil Precup, Marius-Csaba Sabau, and Emil M Petriu. Nature-inspired optimal tuning of input membership functions of takagi-sugeno-kang fuzzymodels for anti-lock braking systems. Applied Soft Computing, 27 :575–589, 2015.

[61]. Tianhua Chen, Qiang Shen, Pan Su, and Changjing Shang. Induction of quantified fuzzy rules with particle swarm optimisation. In 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pages 1–7. IEEE, 2015.

[62]. Jaouher Chrouta, Abderrahmen Zaafouri, and Mohamed Jemli. An improved heterogeneous multi-swarm pso algorithm to generate an optimal ts fuzzy model of a hydraulic process. Transactions of the Institute of Measurement and Control, 40(6) :2039–2053, 2018.

[63]. Lamine Brikh, Ouahib Guenounou, Fatah Yahiaoui, MA Kacimi, and Ahmed Ouaret. Optimization of tsk fuzzy model using new improved pso. International Journal of Control Theory and Applications, pages 323–333, 2016.

[64]. KS Narendra. Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks, 1(1) :183–192, 1989.

[65]. Mohand Akli KACIMI. Contribution à la conception des systèmes flous de type MAMDANI par les algorithmes d'optimisation multi-objectif. PhD thesis, Université de Béjaia-Abderrahmane Mira.

[66]. Eneko Osaba, Esther Villar-Rodriguez, Javier Del Ser, Antonio J Nebro, Daniel Molina, Antonio LaTorre, Ponnuthurai N Suganthan, Carlos A Coello Coello, and Francisco Herrera. A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. Swarm and Evolutionary Computation, 64 :100888, 2021.

[67]. James C Bezdek, Robert Ehrlich, and William Full. Fcm : The fuzzy c-means clustering algorithm. Computers and geosciences, 10(2-3) :191–203, 1984.

[68]. Janmenjoy Nayak, Bighnaraj Naik, and HSr Behera. Fuzzy c-means (fcm) clustering algorithm : a decade review from 2000 to 2014. In Computational Intelligence in Data Mining-Volume 2 : Proceedings of the International Conference on CIDM, 20-21 December 2014, pages 133–149. Springer, 2015.

[69]. Bao Chong et al. K-means clustering algorithm : a brief review. vol, 4 :37–40, 2021.

[70]. Hisao Ishibuchi and Takashi Yamamoto. Rule weight specification in fuzzy rule-based classification systems. IEEE transactions on fuzzy systems, 13(4) :428–435, 2005.

[71]. Sharifa Rajab and Vinod Sharma. An interpretable neuro-fuzzy approach to stock price forecasting. Soft Computing, 23 :921–936, 2019.

[72]. Brikh Lamine, Guenounou Ouahib, and Bakir Toufik. Selection of minimum rules from a fuzzy TSK model using a PSO–FCM combination. Journal of Control, Automation and Electrical Systems, 34(2) : 384-393, 2023.

[73]. Liang Wang and R Langari. Complex systems modeling via fuzzy logicieee transactions on systems. Man, and Cybernetics-Part B : Cybernetics26, 100106, 1996.

[74]. Wael A Farag, Victor H Quintana, and Germano Lambert-Torres. A genetic based neuro-fuzzy approach for modeling and control of dynamical systems. IEEE Transactions on neural Networks, 9(5) :756–767, 1998.

[75]. Ouahib Guenounou, Boutaib Dahhou, and Ferhat Chabour. Tsk fuzzy model with minimal parameters. Applied Soft Computing, 30 :748–757, 2015.

[76]. Chaoshun Li, Jianzhong Zhou, Li Chang, Zhengjun Huang, and Yongchuan Zhang. T–s fuzzy model identification based on a novel hyperplane-shaped membership function. IEEE Transactions on Fuzzy Systems, 25(5) :1364–1370, 2016.

[77]. Haydar Kilic, Ugur Yuzgec, and Cihan Karakuzu. A novel improved antlion optimizer algorithm and its comparative performance. Neural Computing and Applications, 32(8) :3803–3824, 2020.

[78]. Wei Li, Junfei Qiao, Xiao-Jun Zeng, and Shengli Du. Identification and simplification of ts fuzzy neural networks based on incremental structure learning and similarity analysis. Fuzzy Sets and Systems, 394 :65–86, 2020.

Résumé

Cette thèse se concentre sur la conception et l'optimisation des systèmes flous de type TSK (Takagi-Sugeno-Kang), en mettant l'accent sur la modélisation et le contrôle des systèmes non linéaires à l'aide de techniques d'apprentissage automatique et d'algorithmes évolutionnaires. Deux contributions principales y sont présentées.

La première consiste en une version améliorée de l'algorithme Particle Swarm Optimization (PSO), spécialement adaptée pour un apprentissage plus efficace des paramètres des systèmes flous. La seconde propose une hybridation novatrice entre deux algorithmes distincts, visant à réduire et à optimiser la structure des systèmes flous.

Les résultats expérimentaux et théoriques, présentés en conclusion, démontrent que l'intégration des algorithmes évolutionnaires dans la conception des systèmes flous TSK constitue une solution alternative prometteuse pour la gestion des systèmes non linéaires, qu'il s'agisse de cas théoriques ou d'applications en temps réel.

Mots clé : Logique floue, optimisation, algorithmes évolutionnaires, intelligence artificielle, méta-heuristiques, systèmes non-linéaires

ملخص

تركز هذه الأطروحة على تصميم أنظمة تاكاجي سوجينو كانج (TSK) الضبابية وتحسينها، مع التركيز على نمذجة الأنظمة غير الخطية والتحكم فيها باستخدام تقنيات التعلم الآلي والخوارزميات التطورية. يتم تقديم مساهمتين رئيسيتين.

الأولى هي نسخة محسّنة من خوارزمية تحسين سرب الجسيمات (PSO) ، والتي تم تكييفها خصيصًا لتعلم معلمات النظام الضبابي بكفاءة أكبر. أما الثانية فتقترح تهجينًا مبتكرًا بين خوارزميتين متميزتين، تهدفان إلى تقليل وتحسين بنية الأنظمة الضبابية.

تُظهر النتائج التجريبية والنظرية المعروضة في الخاتمة أن دمج الخوارزميات التطورية في تصميم الأنظمة الضبابية TSK هو حل بديل واعد لإدارة الأنظمة غير الخطية، سواء في الحالات النظرية أو في تطبيقات الوقت الحقيقي.

الكلمات المفتاحية : المنطق الغامض-التحسين-الخوارزميات المتطورة-الذكاء الاصطناعي-الاستدلال الفوقية-الأنظمة الغير خطية

Abstract

This thesis focuses on the design and optimization of Takagi-Sugeno-Kang (TSK) fuzzy systems, with an emphasis on the modelling and control of non-linear systems using machine learning techniques and evolutionary algorithms. Two main contributions are presented.

The first is an improved version of the Particle Swarm Optimization (PSO) algorithm, specially adapted for more efficient learning of fuzzy system parameters. The second proposes an innovative hybridization between two distinct algorithms, aimed at reducing and optimizing the structure of fuzzy systems.

The experimental and theoretical results presented in the conclusion show that the integration of evolutionary algorithms in the design of TSK fuzzy systems is a promising alternative solution for the management of non-linear systems, both in theoretical cases and in real-time applications.

Mots clé : fuzzy Logic, optimization, evolutionnary algorithms, artificial intelligence, meta-heuristics, non-linear systems