TABLE DES MATIERES

4 Optimisation des paramètres du contrôleur PID par l'algorithme immunitaire artificiel

Liste des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Liste des tableaux

Liste des abréviations et symboles

AG	Algorithmes génétiques
AIA	Algorithme immunitaire artificiel
EKF	Extended Kalman Filter
FL	Fuzzy Logic
GAP	Growing and Pruning
MIMO	Multi-entrées multi-sorties
MRAC	Commande adaptative à modèle de référence
PID	Proportionnelle, intégrale et dérivée
RBF	Réseau à fonctions de bases radiales
RMS	Root Mean Square
SEF	Sous -ensembles flous
SI	Système immunitaire
SIA	Système immunitaire artificiel
SIF	Système d'inférence floue
SISO	Mono-entrée mono-sortie
K_d	Gain dérivé d'un PID
K_i	Gain intégral d'un PID
K_p	Gain proportionnel d'un PID
m	La masse de la tige d'un pendule inversé
$M(q)$	Matrice d'inertie d'un robot
$C(q,q)$	Matrice de Coriolis
x	Entrée d'un réseau
$x(t)$	Vecteur d'état
$x_d(t)$	État désirée
$y(t)$	Sortie d'un système ou d'une fonction
Tm	Temps de montée d'un système
Tr	Temps de réponse d'un système
$u(t)$	Signal de commande
$U_{A(x)}$	Fonction d'appartenance d'une variable floue
Nx	Nombre d'entrées d'un réseau de neurones
Ny	Nombre de sorties d'un réseau de neurones
e	Erreur de suivi
σ	Largeur d'une fonction gaussienne
Θ	Angle du pendule inversé
a	Poids du réseau de neurones

Introduction générale

La recherche dans la théorie de la commande des systèmes non linéaires a été motivée par les caractéristiques non linéaires inhérentes aux systèmes physiques que nous essayons souvent de commander. Bien que les systèmes linéaires soient très bien maitrisés et commandés, la commande linéaire n'est pas assez efficace pour bien adapter la stabilité et la performance des systèmes non linéaires.

Les méthodes classiques de l'automatique ont été largement appliquées à de nombreux problèmes de régulation industrielle. Cependant, la plupart des systèmes physiques présentent des non linéarités et leurs paramètres sont souvent mal connus et/ou variables dans le temps (LEWIS, 1993).

Pour la commande de telles classes de systèmes, les méthodes conventionnelles de l'automatique ont montré leurs limites en termes de stabilisation et de performances.

Avec le développement considérable des calculateurs numériques, les automaticiens ont adopté de plus en plus de nouvelles approches telles que la commande adaptative, la commande prédictive et la commande robuste ainsi que les techniques basées sur l'intelligence artificielle (SLOTINE, 1991).

La présence des perturbations et les variations paramétriques appliquées aux systèmes commandés, amènent à l'apparition des commandes intelligentes comme une bonne solution permettant d'assurer la stabilité, la robustesse, la précision, pour les systèmes perturbés. En informatique, les avancées récentes ont permis l'apparition des méthodes intelligentes : la logique floue, les modèles informatiques des systèmes biologiques tels que l'intelligence en essaims, les réseaux de neurones artificiels, les systèmes immunitaires artificiels etc. Ces outils sont regroupés sous le concept de l'intelligence artificielle. En outre les algorithmes évolutionnaires sont dotés d'outils intelligents pour résoudre des problèmes complexes du monde réel comme l'approximation des paramètres des systèmes non linéaires et l'optimisation des paramètres des contrôleurs (BRECHT, 2003).

Objectif et contributions

Dans cette thèse, nous avons travaillé sur trois contributions différentes pour exploiter des approches de commande à base des réseaux de neuro-flous et des algorithmes évolutionnaires.

La première repose sur le contrôle par un réseau neuro-flou, qui est actuellement dans la littérature l'une des préoccupations des chercheurs. Le travail présenté pour cette contribution concerne la commande neuro-floue intelligente, basée sur le réglage automatique des règles du système d'inférence floue via STFIS (Self tuning Fuzzy Inference System) afin de commander un système non linéaire. L'objectif principal est l'interprétation en ligne et l'optimisation de la base des règles, avec un minimum de temps pour assurer la robustesse du contrôleur où l'adaptabilité de la logique floue, la capacité d'apprentissage et de la généralisation des réseaux de neurones sont combinées.

Dans la deuxième contribution nous utilisons un système d'inférence flou avec un apprentissage séquentiel adaptatif du réseau neurones (Séquentiel Adaptif Fuzzy Inference System, SAFIS). Cette dernière est basée sur la combinaison en parallèle des réseaux de neurones RBF avec apprentissage par l'algorithme GAP-EKF et d'un contrôleur flou, dont le but est de développer une loi de commande adaptative pour commander un système non linéaire. Il convient de noter que SAFIS est un algorithme d'apprentissage vraiment séquentiel, et en ligne qui est appliqué pour estimer les différents paramètres de modèle du système non linéaire. Nous injectons les paramètres estimés dans une loi de commande couple calculée (computed torque) par cet algorithme pour contrôler un système qui possède une forte non linéarité.

Une étude sur l'impact du choix des paramètres de l'algorithme SAFIS est effectuée pour tester la robustesse de la loi basée sur l'approche proposée.

La troisième approche est consacrée à l'utilisation d'un algorithme évolutionnaire AIA (algorithme immunitaire artificiel) pour optimiser les paramètres d'un contrôleur classique PID. Dans la littérature ces paramètres sont ajustés arbitrairement afin de contrôler un système non linéaire. Ce type d'algorithmes évolutionnaires sont des méta-heuristiques d'inspiration biologique basés sur la sélection naturelle néo-Darwinienne. Ils génèrent une population de solutions candidates et tentent d'évoluer en appliquant des opérateurs d'évolution et la meilleure solution est celle qui a la valeur optimale d'une fonction objectif. Dans le contexte de robustesse, une étude sur l'impact des paramètres de l'AIA a été intégrée dans la fin de ce chapitre.

Ces contributions ont fait l'objet des productions scientifiques suivantes :

- Sahraoui, M.; Salem, M.; Khelfi, M.F. "***Intelligent Control of nonlinear systems via Self tuning neuro-fuzzy architecture*** ", Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on Year: 2012 Pages: 912 - 916, DOI: 10.1109/SETIT.2012.6482036 IEEE Conference Publications
- **Publication 1**: Sahraoui Mustapha, Khelfi Mohamed Fayçal, Salem Mohammed. "***Sequential Adaptive Fuzzy Inference System Based Intelligent Control of Robot Manipulators***",International Journal of Intelligent Systems and Applications (**IJISA**) ISSN Print: 2074-904X, ISSN Online: 2074-9058 Volume 6, Number 11, October 2014
- **Publication 2**: Sahraoui Mustapha, Khelfi Mohamed Fayçal, Salem Mohammed ***"Application of artificial immune algorithm-based optimisation in tuning a PID controller for nonlinear systems"***, International Journal of Automation and Control (**IJAAC**), Print ISSN: 1740-7516 Online ISSN: 1740-7524 Vol. 9, No. 3, 2015

Organisation de la thèse

Le présent manuscrit est organisé comme suit :

Le premier chapitre présente les éléments de base nécessaires à la justification et la compréhension du reste du travail. Il dresse un état de l'art non exhaustif des techniques de l'intelligence artificielle : réseaux de neurones, logique floue et les réseaux neuro-flou, ainsi quelque type de commande des systèmes non linéaires.

Le deuxième chapitre commence par introduire un contrôleur neuro-flou pour avoir une commande intelligente afin de contrôler des systèmes non linéaires (masse ressort et pendule inversé). Nous passons ensuite à détailler l'architecture de ce contrôleur intelligent nommé STFIS en se concentrant sur l'algorithme d'apprentissage de ce type de réseau.

Le troisième chapitre présente une architecture de commande neuro-floue adaptative SAFIS pour estimer les paramètres du modèle dynamique du robot manipulateur SCARA afin de les injecter dans loi de commande couple calculée. Cette architecture introduit les réseaux RBF avec un algorithme GAP-EKF pour concevoir une commande intelligente pour le robot manipulateur SCARA.

Le quatrième chapitre est consacré à l'optimisation par la méta-heuristique, comme l'algorithme évolutionnaire AIA pour optimiser les

paramètres du contrôleur PID en ligne afin de commander un système non linéaire masse ressort amortisseur.

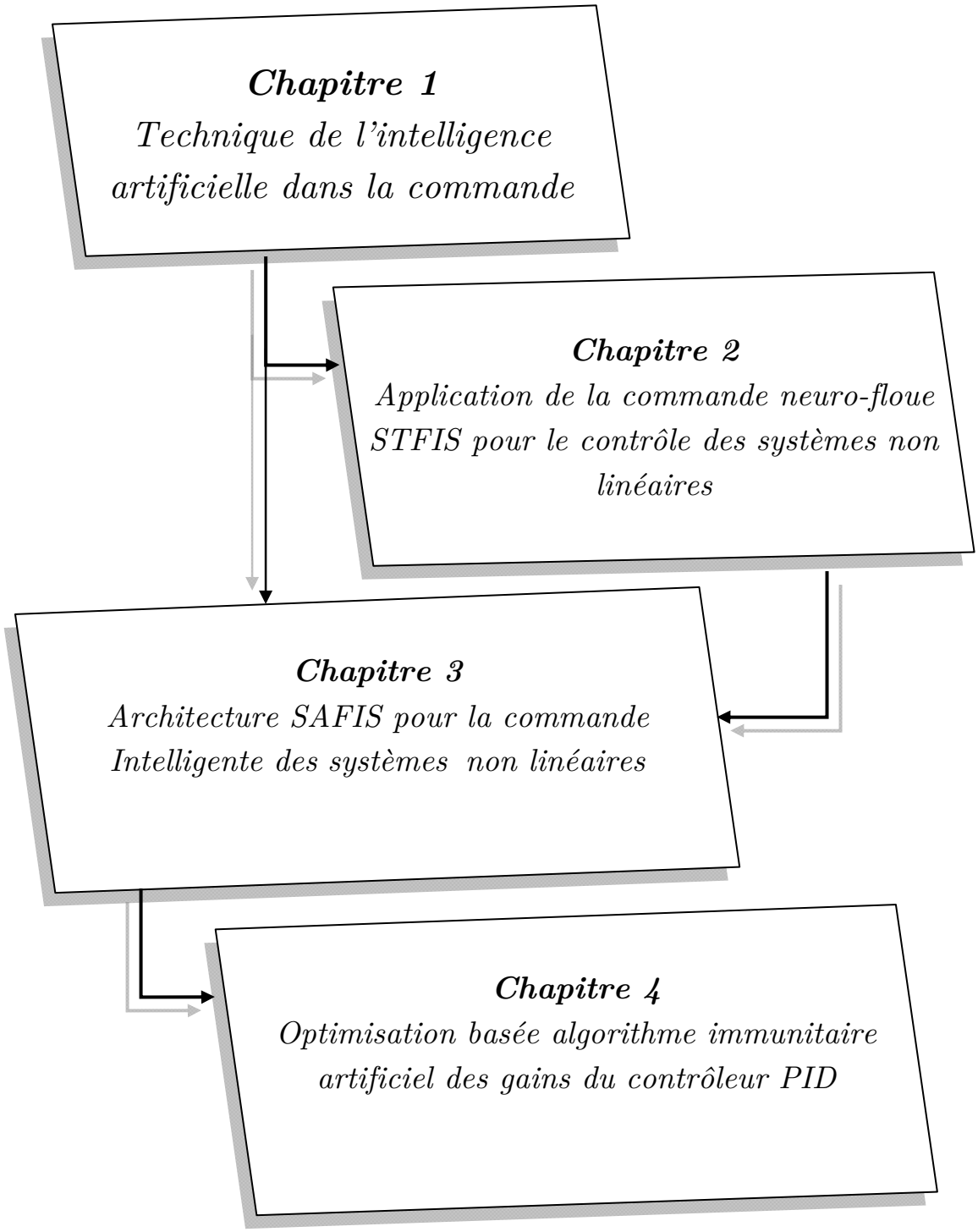


Figure0 : Navigation à travers la thèse

Chapitre 1

Techniques de l'intelligence artificielle dans la commande

1. Introduction

Dans le domaine de la recherche scientifique actuelle, l'un des défis des chercheurs aujourd'hui est de copier la nature et de reproduire des modes de raisonnement et de comportement qui lui sont propre. Dans notre monde la majorité des systèmes dynamiques utilisés sont des systèmes non linéaires. Ce type de système est généralement sujet aux variations et incertitudes structurelles et non structurelles causées par l'environnement, d'où le recours aux techniques de l'intelligence artificielle pour résoudre ces problèmes tel que les réseaux de neurones, la logique floue, les réseaux neuro-floue dont l'adaptation aux changements environnementaux est l'un des propriétés majeur de ces techniques.

Dans ce chapitre nous allons présenter un état de d'art sur les techniques de l'intelligence artificielle. Cette dernière est un domaine de l'informatique qui fait l'étude des mécanismes d'adaptation.

En premier lieu nous allons survole sur ces techniques en commençant par les réseaux de neurones, logique floue et les réseaux neuro-floue. Ces derniers combinent la capacité d'adaptation de la logique floue et la puissance des capacités d'apprentissage et de généralisation des réseaux de neurones, puis leurs applications dans la commande des systèmes non linéaires.

2. Réseaux de neurones

2.1 Neurone formel

Dans la littérature un neurone formel est un automate très simple imitant grossièrement la structure et le fonctionnement d'un neurone biologique, il représente une fonction algébrique non linéaire et bornée, dont la valeur dépend de paramètres appelés coefficients ou poids. Les variables de cette fonction sont habituellement appelés ''entrées '' du neurone et la valeur de la fonction est appelée sa sortie (DREYFUS, 2002).

Un neurone est avant tout un opérateur mathématique numérique qui réalise deux opérations. La première est la somme pondérée du neurone par les poids synaptiques ; cette somme est appelée potentiel neuronal. La seconde génère la sortie du neurone, image par une fonction f appelée généralement fonction d'activation ou d'évaluation (Figure 1.1).

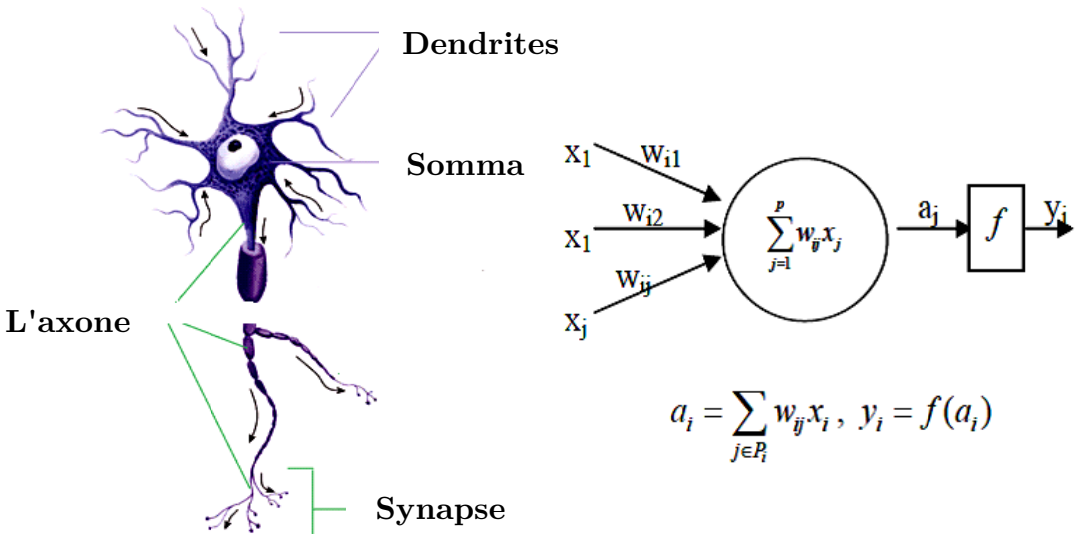


Figure 1.1.Neurone biologique et neurone formel

avec x_i : paramètres d'entrées
w_{ij} : poids synaptiques
a_i : potentiel neuronal
y_i : sortie du neurone i
f : la fonction d'activation

Une collection de neurones formels est appelée réseaux de neurones. Ces réseaux sont souvent constitués de couches de neurones interconnectés

(MCCULLOCH, 1943) et le type de connections, la nature de la fonction d'activation et d'autres paramètres détermineront le type du réseau de neurones (Perceptron multicouches, Réseaux à fonctions de base radiales, Réseau de Kohonen, etc..). Même si les réseaux de neurones ne présentent, dans la majorité des cas, qu'une lointaine parenté avec l'architecture anatomique, physiologique et biologique du tissu nerveux dont ils sont inspirés, ils remportent actuellement un succès important, parce qu'ils présentent des propriétés intéressantes dans des domaines comme: la commande des processus, le diagnostic, la compression, la prévision, l'interpolation, l'optimisation, la classification, la reconnaissance de formes, la reconnaissance de la parole, etc.

2.2 Architecture des réseaux de neurones

2.2.1 Réseaux non bouclés

Ce sont des réseaux unidirectionnels sans retour en arrière (feedforward). Le signal de sortie est directement obtenu après l'application du signal d'entrée. Si tous les neurones ne sont pas des organes de sortie, on parle de neurones cachés ,(Figure 1.2), (POGGIO, 1990).

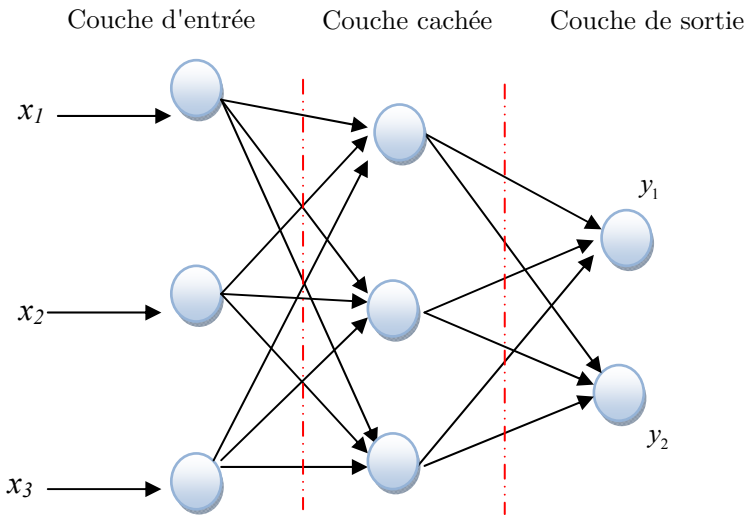


Figure 1.2 Réseau non bouclé.

2.2.2 Réseaux bouclés

Il s'agit de réseaux de neurones avec retour en arrière (feedback network ou récurrent network) (POGGIO, 1990), (Figure 1.3).

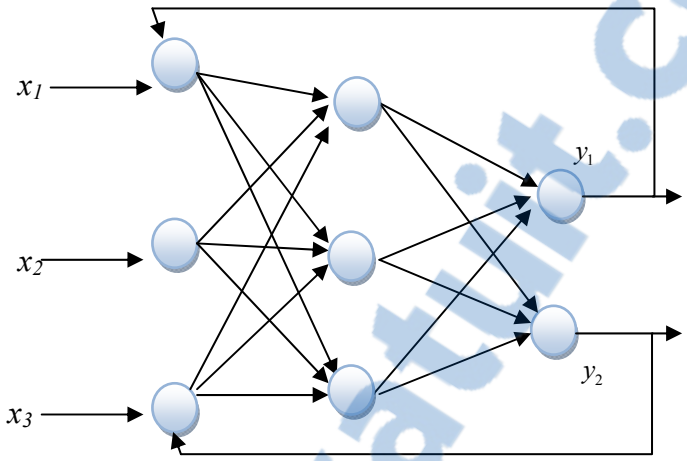


Figure 1.3 Réseau bouclé

2.3 Propriétés des réseaux de neurones

Les réseaux de neurones à couches, présentés au paragraphe précédent, ont la propriété générale d'être des approximateurs universels parcimonieux. Il s'agit en fait de deux propriétés distinctes détaillées ci-dessous (HAGAN, 1996).

2.3.1 Propriétés d'approximation des réseaux de neurones

Les résultats qui présentent un intérêt pour la modélisation et la commande des processus sont ceux qui concernent l'approximation de fonctions à valeurs continues. Les travaux de (CYBENKO, 1989) et (FUNAHASHI, 1989) ont prouvé la possibilité d'approcher des fonctions continues, au sens de la norme uniforme sur les compacts, par des réseaux de neurones. Les réseaux considérés sont de type à une couche cachée à fonction d'activation non linéaire, et à neurones de sortie linéaires.

2.3.2 Propriété de parcimonie

Lorsque nous cherchons à modéliser un processus à partir des données, nous nous efforçons toujours d'obtenir les résultats les plus satisfaisants possibles avec un nombre minimum de paramètres ajustables. Dans cette

optique, (HORNIK, 1994) ont montré que si le résultat de l'approximation (c'est-à-dire la sortie du réseau de neurones) est une fonction non linéaire avec des paramètres ajustables, elle est plus parcimonieuse que si elle est une fonction linéaire de ces paramètres. De plus, pour des réseaux de neurones à fonction d'activation sigmoïdale, l'erreur commise dans l'approximation varie comme l'inverse du nombre de neurones cachés, et elle est indépendante du nombre de variables de la fonction à approcher (HORNIK, 1994). Par conséquent, pour une précision donnée, pour un nombre de neurones cachés donné, le nombre de paramètres du réseau est proportionnel au nombre de variables de la fonction à approcher.

Cette propriété montre l'intérêt des réseaux de neurones par rapport à d'autres approximateurs comme les polynômes dont la sortie est une fonction linéaire des paramètres ajustables

2.4 Apprentissage des Réseaux de Neurones

L'apprentissage est considéré comme une tâche de construction de nouvelles connaissances ou amélioration des connaissances existantes dont le rôle est de définir les poids de chaque connexion (MOODY, 1995). Le but de cette phase est d'améliorer les performances du système en tenant compte des ressources et des compétences dont il dispose. L'apprentissage et l'adaptation constituent deux caractéristiques essentielles des réseaux de neurones. Lorsque la phase d'apprentissage est achevée, le réseau doit être capable de faire les bonnes associations pour les vecteurs d'entrées qu'il n'aura pas appris. C'est l'une des propriétés importante dans les réseaux de neurones.

2.4.1 Type d'apprentissage des Réseaux de Neurones

Des différents algorithmes d'apprentissage et différentes types d'apprentissage ont été développés dans la littérature dans le but de réaliser l'adaptation et l'optimisation des poids d'un réseau de neurone. L'apprentissage par correction d'erreur (retro-propagation), l'apprentissage type Boltzmann, l'apprentissage compétitif peuvent être mentionné parmi les algorithmes les plus cités dans la littérature (HAYKIN, 1999), (WINDROW, 1996), (IRWIN, 1995). Il existe trois types principaux d'apprentissages. (NARENDRA, 1990), (HUNT, 1991), (DREYFUS, 2008).

a) Apprentissage supervisé

Un superviseur, ou professeur, fournit au réseau des couples d'entrées-sorties. Il fait apprendre au réseau l'ensemble de ces couples, par une méthode d'apprentissage. Dans le cas de la rétro-propagation du gradient de l'erreur, en comparant pour chacun d'entre eux la sortie effective du réseau et la sortie désirée. L'apprentissage est terminé lorsque tous les couples entrées-sorties sont reconnus par le réseau.

b) Apprentissage non supervisé

Cet apprentissage consiste à détecter automatiquement des régularités qui figurent dans les exemples présentés et à modifier les poids des connexions pour que les exemples ayant les mêmes caractéristiques de régularité provoquent la même sortie. Les réseaux auto-organisateurs de Kohonen sont les réseaux à apprentissage non supervisé les plus connus.

c) Apprentissage par renforcement

Ce mode d'apprentissage, suppose qu'un comportement de référence n'est pas possible, mais en revanche, il est possible d'obtenir des indications qualitatives (vrai, faux, ...) sur les performances du réseau.

Outre la classification présentée ci-dessus, les méthodes d'apprentissage sont souvent différenciées de par leur caractère hors ligne (hors ligne) ou en ligne (on-line) :

- **Apprentissage hors ligne**

Ce mode d'apprentissage consiste à accumuler les erreurs instantanées consécutives, et à n'effectuer l'adaptation des poids synaptiques que lorsque l'ensemble des données d'apprentissage ont été présentées au réseau. Ce mode permet de mieux estimer le gradient réel de la fonction coût, puisqu'il est à présent calculé à partir d'un ensemble d'exemples, plutôt qu'à partir d'un seul.

- **Apprentissage en ligne**

Il consiste à modifier les valeurs des poids synaptiques immédiatement après la présentation d'un exemple. Dans ce cas, seul le gradient instantané de la fonction coût est utilisé pour l'adaptation des paramètres du système. Sous la condition que les exemples soient présentés au réseau de neurones de

manière aléatoire, l'apprentissage en-ligne rend la recherche du minimum de la fonction coût stochastique, ce qui rend moins probable, pour l'algorithme d'apprentissage, de tomber dans un minimum local. L'efficacité relative des modes d'apprentissage en ligne et hors ligne dépend essentiellement du problème considéré. L'apprentissage en-ligne présente cependant l'avantage que pour une seule présentation de l'ensemble de la base de données. Il implique de multiples phases d'adaptations des poids synaptiques lorsque des données similaires se représentent. Ce qui se produit fréquemment pour des bases de données très étendues (HAGAN, 1995).

2.4.2 Méthodes d'apprentissage

Les connaissances de l'expert ont une forme énumérée : elles sont exprimées sous forme de règles. Dans le cas des réseaux de neurones, les connaissances ont une forme distribuée : elles sont codées dans les poids des connexions, la topologie du réseau, les fonctions de transfert de chaque neurone, le seuil de ces fonctions, la méthode d'apprentissage utilisée. Il existe un certain nombre de méthodes d'apprentissage (RACOCEANU, 2006).

a) Règle de Hebb

C'est la méthode d'apprentissage la plus ancienne (1949), elle est inspirée de la biologie. Elle traduit le renforcement des connexions liant deux neurones activés. Si un des deux neurones au moins n'est pas activé, le poids de la connexion n'est pas modifié. La loi de Hebb s'applique aux connexions entre neurones. Elle s'exprime de la façon suivante : "Si deux cellules sont activées en même temps, alors la force de la connexion augmente".

La modification des poids dépend de l'activation des neurones présynoptique et post synaptique, ainsi comme le montre la table 1. x_i et x_j sont respectivement les valeurs d'activation des neurones i et j, ∂w_{ij} (dérivée partielle du poids) correspond à la modification de poids réalisée.

x_i	x_j	∂w_{ij}
0	0	0
0	1	0
1	0	0
1	1	+

Table 1.1 La loi de Hebb

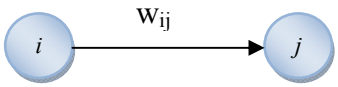


Figure 1.4 : i le neurone amont, j le neurone aval et w_{ij} le poids de la connexion

b) Rétro-propagation du gradient de l'erreur

Cet algorithme est utilisé dans les réseaux de type feedforward, dont l'objectif est de modifier les poids du réseau dans le sens contraire du gradient du critère de performance. Le principe de la rétro-propagation consiste à présenter au réseau un vecteur d'entrées, de procéder au calcul de la sortie par propagation à travers les couches, de la couche d'entrées vers la couche de sortie n passant par les couches. Cette sortie obtenue est comparée à la sortie désirée, une erreur est alors obtenue. A partir de cette erreur, est calculé le gradient de l'erreur qui est à son tour propagé de la couche de sortie vers la couche d'entrée, d'où le terme de rétro-propagation. Cela permet la modification des poids du réseau et donc l'apprentissage. L'opération est réitérée pour chaque vecteur d'entrée et cela jusqu'à ce que le critère d'arrêt soit vérifié (WIDROW, 1990).

3. La logique floue

3.1 Généralités sur la logique floue

La logique classique ne permet pas de raisonner de manière souple sur des connaissances imprécises et/ou incertaines. La logique floue a été proposée et introduite par (ZADEH, 1965) pour modéliser le langage naturel et pour rendre compte du caractère vague des connaissances que nous, les humains, manipulons au quotidien à travers la notion de fuzzy sets ou sous-ensembles flous ; généralisant ainsi l'algèbre de la théorie classique des ensembles.

Cette théorie est basée sur la représentation et la manipulation des connaissances imprécises et la connaissance de l'environnement lorsqu'elle est assez incomplète et incertaine. (ZADEH, 1965) a proposé d'associer à un concept imprécis (concept humain e.g. grand, chaud,...) une fonction d'appartenance à un ensemble. Cette fonction d'appartenance définit le degré de représentation d'une instance vis-à-vis du concept. Il a été montré qu'une telle fonction est

utile pour représenter l'influence des modificateurs linguistiques, comme «très», «plus ou moins», «peu» sur la signification des concepts. L'approche par sous-ensembles flous a été très étudiée et a trouvé de nombreuses applications, en particulier, dans le contrôle des systèmes complexes, permettant de décrire ces systèmes avec un ensemble d'expressions conditionnelles floues de façon similaire au raisonnement humain.

3.1.1 Variable floue

La logique floue permet de tenir compte de la nature imprécise du monde réel, grâce à des termes flous ou variables linguistiques (e.g. "Petit", "Moyen", "Grand"). Chaque terme représente un sous-ensemble de valeurs numériques et caractérise ainsi la variable floue (e.g."Température"). Le domaine sur lequel ces termes et ces variables sont définies constitue l'univers de discours ou le domaine de définition (référentiel), (MAARAF, 2002).

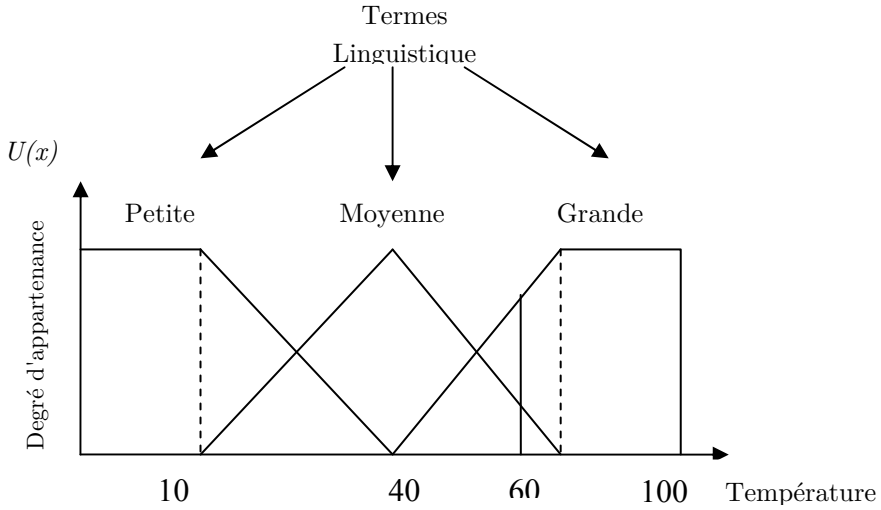


Figure 1.5 : Exemple Partition floue de l'univers de discours de la variable linguistique "Température".

3.1.2 Sous-ensembles flous

Les sous-ensembles flous (SEF) sont une classe d'objet où la transition entre l'appartenance et le non appartenance à l'ensemble est graduelle. Un sous-ensemble flou. A, est défini par :

- Un intervalle convexe A de $\Re$, auquel est associé un label linguistique (e.g. "Petit","Moyen", "Grand").

- Une fonction réelle $U_A(x)$ à valeurs dans [0, l], qui donne le degré d'appartenance d'une variable x au sous-ensemble flou $U_A(x)$. La fonction d'appartenance modélise les situations où un même élément peut être classé dans plusieurs catégories avec des degrés divers (MAARAF, 2002).

3.1.3 Fonction d'appartenance

Afin de permettre un traitement numérique des variables linguistiques dans la prise des décisions floues, une définition des variables linguistiques à l'aide de fonctions d'appartenance est imposée. Dans ce contexte, nous associons à chaque valeur de la variable linguistique une fonction d'appartenance définie par $U_{A(x)}$, qui sera désignée par le degré ou le facteur d'appartenance.

Le plus souvent, nous utilisons pour les fonctions d'appartenance les fonctions suivantes (Figure 1.6) (KARTALOPOULOS, 2004).

- **Fonction triangulaire** : elle est définie par trois paramètres a,b,c qui déterminent les coordonnées des trois sommets (Figure 1.6 (a)).

$$\mu_A(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{1.1}$$

- **Fonction trapézoïdale** : elle est définie par quatre paramètres a,b,c,d (Figure 1.6(b)):

$$\mu_A(x) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{c-x}{d-x}\right), 0\right) \tag{1.2}$$

- **Fonction gaussienne** : elle est définie par deux paramètres σ, m (Figure 1.6 (c)):

$$\mu_A(x) = \exp\left\{-\frac{(x-m)^2}{2\sigma^2}\right\} \tag{1.3}$$

- **Fonction sigmoïde** : une fonction sigmoïde est définie par deux paramètres a,c (Figure 1.6 (d)):

$$\mu_A(x) = \frac{1}{1+\exp\{-a(x-c)\}} \tag{1.4}$$

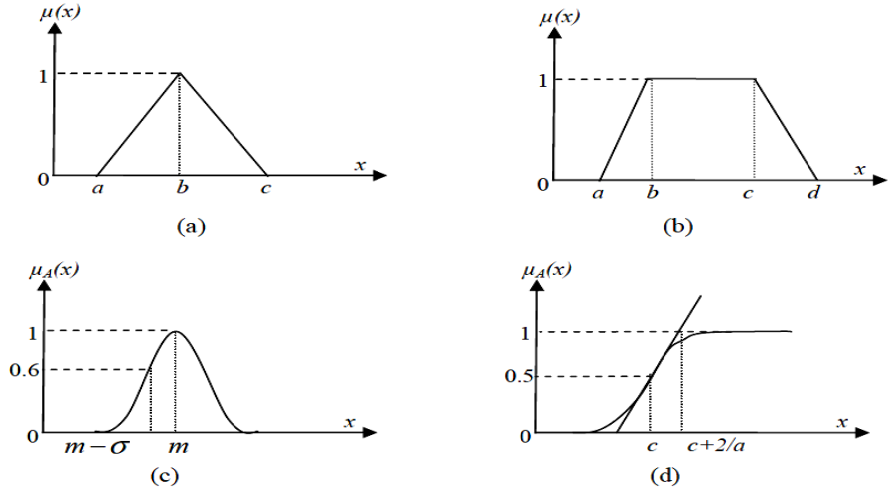


Figure 1.6 : Fonction d'appartenances

3.1.4 Base de règles floues

Une base de règles est un ensemble de règles conditionnelles floues "Si «condition» alors «action»", composées de deux parties : prémisse et conclusion "Si «prémisse» alors «conclusion»". Ces règles floues peuvent être classées en 3 types selon la forme de leur partie conclusion.

Type 1 : conséquence est une constante

Règle i : Si x_i est A_n et... et x_n est A_m alors y est w_j.
Type 2 : conséquence est une fonction linéaire de premier ordre

Règle j : Si x_i est A_{ji} et... et x_n est A_{jn} alors $g_i(x_i,\ldots,x_n)=b_0+b_ix_i+\ldots b_nx_n$.
Type 3 : conséquence est un ensemble flou

Règle k : Si x_i est A_n et... et x_n est A_{kn} alors y est B_k.

Dans ces règles, les x_i et y représentent respectivement les variables d'entrée et de sortie, les A_{sl} et B_K sont des sous-ensembles flous, w_j est une valeur constante, et $g_i(x_i,\ldots,x_n)=b_0+b_ix_i+\ldots b_nx_n$ est fonction linéaire des entrées où les b_j sont des coefficients constants.

3.2 Système d'inférence floue

Un système d'inférence floue est composé d'une base de connaissance, une partie d'entrée représenté par une interface de Fuzzification et une sortie

par une interface de Défuzzification. (Figure 1.7) illustre la structure générale du cœur d'un système d'inférence floue (SIF) (ABRAHAM, 2001) :

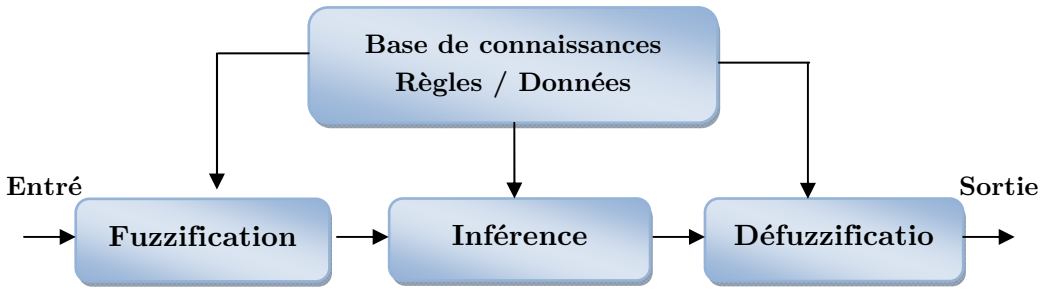


Figure 1.7 : Structure générale d'un système d'inférence flou

Le système inférence floue est un mécanisme de décision, il permet à partir d'un fait observé de la base des règles floues une décision en exploitant le raisonnement approximatif.

3.2.1 Fuzzification

La fuzzification est appliquée pour convertir les valeurs numériques relatives aux entrées en une série de degrés d'appartenance à chaque SEF : (MAARAF, 2002)

-Par union de tous les SEF locaux, l'opération pour réaliser l'union est la T-conorme de Zadeh (Max). (KLIR, 1994)

- Par le calcul du barycentre des conclusions inférées, quand celles-ci sont de nature précise (non floue) (MAARAF, 2002)

3.2.2 Inférence floue

L'inférence floue est l'application de la caractérisation symbolique du système aux règles floues et la déduction d'un certain nombre de résultats locaux ; également exprimés sous forme symbolique concernant les variables de sortie du système. Le but de cette étape est d'arriver à déterminer des sorties Floues ; en partant d'entrées floues et en utilisant une base de règles (WANG, 1994).

Pour pouvoir utiliser cette base de règles, nous avons besoin de trois opérateurs, mathématiques, la conjonction (ET) qui s'applique aux variables à l'intérieur d'une règle tandis que l'opérateur ; l'implication (Si ...Alors) et l'agrégation (Sinon).

Il existe plusieurs méthodes pour réaliser ces opérateurs dans une inférence et qui s'appliquent aux fonctions d'appartenance à savoir :

- Méthode d'inférence max-min (Mamdani) : l'implication floue et l'operateur ET sont réalisés par la fonction minimum et l'opérateur logique OU est réalisé par la fonction maximum ; elle nécessite un temps de calcul trop élevé. Cette méthode est également dite l'implication de Mamdani, d'où la fonction d'appartenance résultante correspond au maximum des deux fonctions d'appartenance partielles puisque les règles sont liées par l'opérateur OU.
- Méthode d'inférence max-prod (Larsen) : la seule différence avec la méthode précédente est la réalisation de l'implication floue le produit est utilisé. Les opérateurs logiques ET et OU sont toujours réalisés respectivement par les fonctions minimum et maximum. La méthode max-prod est également dite implication de (ZADEH, 1972)
- Méthode d'inférence somme-prod (Sugeno) : la méthode d'inférence somme -prod réalise, au niveau de la condition, l'opérateur OU par la formation somme, et l'opérateur ET par la formation produit. Au niveau de la conclusion, elle réalise également l'opérateur ALORS par la formation produit.

3.2.3. Défuzzification

L'interface de défuzzification génère une valeur chiffrée (non floue : crisp) dans la sortie qui doit traduire au mieux l'ensemble flou obtenu de l'opération précédente. Plusieurs stratégies sont utilisées pour réaliser l'étape de défuzzification, les plus répandues sont :

- La méthode du centre de gravité (COG) : la sortie défuzzifiée x^* correspond à l'abscisse du centre de gravité de la surface de la fonction d'appartenance résultante μ(x) (Figure 1.8). Mathématiquement, cela se traduit par :

$$x^* = \frac{\sum_{i=1}^{n} \mu(x_i) x_i}{\sum_{i=1}^{n} \mu(x_i)} \tag{1.5}$$

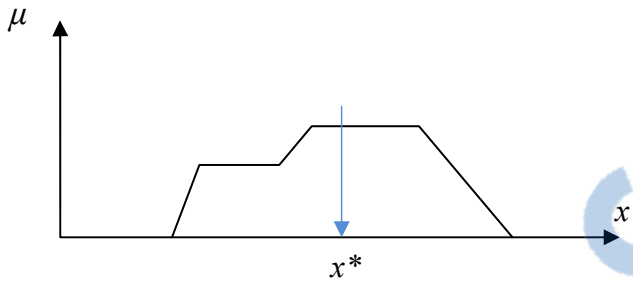


Figure 1.8 : Défuzzification par centre de gravité

- La méthode de la moyenne des maxima, (MAARAF, 2002) : elle été introduite afin de remédier à la lourdeur des calculs de la méthode du centre de gravité. La sortie défuzzifiée est obtenue par le calcul de la moyenne des abscisses pour lesquelles la fonction d'appartenance est maximale :

$$x^* = \frac{\sum_{i=1}^{n} \mu(x_i)}{n} \tag{1.6}$$

- Ou x_i sont définis par : $\mu(x_i)$=max (μ(x))

3.3 Types des contrôleurs flous

Un contrôleur flou est un système d'inférence flou employé dans la technique d'automatisation. Son but est de trouver une valeur numérique à appliquer au système à partir d'un jeu de variables physiques. Les principaux types de contrôleurs qui ont été développés portent le nom des chercheurs qui les ont proposés, il s'agit du contrôleur de Mamdani et du contrôleur de Sugeno. Pour exposer le principe de fonctionnement de chacun d'eux, nous considérons l'exemple d'une base de règles de la forme (WANG, 1994) :

Règle i : si x_1 est A_i et x_2 est B_i Alors y est C_i .

Où A_i, B_i et C_i sont des sous-ensembles flous.

3.3.1 Méthode de Mamdani

Pour exposer le principe de fonctionnement de cette méthode, on considère l'exemple d'une base de règles de la forme (MAMDANI, 1975) :

Règle i : Si x_1 est A_i ET x_2 est B_i Alors y est C_i

où A_i, B_i et C_i sont des sous-ensembles flous.

La méthode de Mamdani est historiquement la première à avoir été proposée, elle repose sur le raisonnement suivant (MAARAF, 2002):

-Calcul de la valeur de vérité de chaque règle :

$$\alpha_i(x) = Min\left(U_{A_i(x_1)}, U_{B_i(x_2)}\right) \tag{1.7}$$

- Calcul de la contribution de chaque règle :

$$\alpha(y) = Min\left(\alpha_i(x), U_{c_i(y)}\right) \tag{1.8}$$

-L'agrégation des règles :

$$\alpha(y) = Max\left(\alpha_i(y)\right) \tag{1.9}$$

- La défuzzyfication pour obtenir une conclusion « nette ».

3.3.2 Méthode de Takagi-Sugeno

Cette méthode a été proposée par Takagi-Sugeno, (TAKAGI et SUGENO, 1985), (LEE, 2003), elle se caractérise par une sortie des règles non floues. A chaque règle, nous associons une sortie définie sous forme numérique comme étant une combinaison linéaire des entrées. Les règles utilisées d'ordre zéro sont du type :

Règle i : Si x_1 est A_i ET x_2 est B_i Alors $y = C_i$

où les valeurs C_i sont des valeurs réelles (non floues). Cette méthode se base sur le raisonnement suivant :

- Calcul de la valeur de vérité de chaque règle :

$$\alpha_i = ET\left(\left(x_1 \; est \; A_i\right), \left(x_2 \; est \; B_i\right)\right) \tag{1.10}$$

- La conclusion de la règle i se calcule : $\alpha_i \; C_i$

- Calcul de la sortie du SIF :

$$y = \frac{\sum_{i=1}^{n} \alpha_i C_i}{\sum_{i=1}^{n} \alpha_i} \tag{1.11}$$

Les valeurs de α_i représentent ici le degré de vérité de chaque règle. Les SIF de type Sugeno permettent une meilleure représentation des fonctions numériques et des mécanismes d'inférence plus rapides.

3.4 Caractéristiques des systèmes d'inférences flou (SIF)

Comme il est noté précédemment, un système flou est en premier lieu caractérisé par son type (Mamdani, Takagi-Sugeno,…) et par les partitions floues qu'il met en œuvre. Cependant, la définition totale d'un système flou passe par la spécification d'un ensemble de caractéristiques dites structurelles et paramétriques.

3.4.1 Les caractéristiques structurelles du SIF

Ces caractéristiques spécifient tous les éléments du SIF qui influent sur sa structure. Ces éléments sont constitués par :

- le type de fonction d'appartenance utilisé (triangle, trapèze, TPE, forme de cloche,…)
- pour chaque terme linguistique, le nombre de termes linguistiques pour chaque variable,
- le nombre optimal de règles,
- les variables principales à ces règles,
- les opérateurs de conjonction, de disjonction et d'implication, et la technique de défuzzification.

3.4.2 Les caractéristiques paramétriques du SIF

Une fois la structure du SIF est choisie, le problème est alors le placement optimal des fonctions d'appartenance d'entrées et de sorties ou des singletons de sorties. Les caractéristiques paramétriques se situent au plus bas niveau de spécification d'un SIF. Elles représentent en fait l'aspect purement numérique du système flou et définissent les sous ensembles qui le constituent :

- Les paramètres des fonctions d'appartenance : variables d'entrée (point modal, base, écart type …)
- Les paramètres des fonctions d'appartenance : variables de sortie (conclusions de règles floues) pour les SIF de type Takagi-Sugeno. (TAKAGI, 1985).

3.4.3 Approximation des fonctions par les systèmes flous

Deux raisons principales amènent à utiliser les systèmes flous comme contrôleurs des systèmes dynamiques. Premièrement, ce type de systèmes flous à la propriété d'approximateur universel de fonctions continues avec un degré de précision quelconque à condition d'utiliser un nombre suffisant de règles floues. Deuxièmement, les systèmes flous sont construits à partir de règles floues de la forme Si-Alors, de ce fait, les informations linguistiques ou mathématiques disponibles d'un expert peuvent éventuellement être incorporées dans le contrôleur. Dans la littérature consacrée aux systèmes flous, on dispose d'un nombre important de publications montrant que les systèmes flous sont des approximateurs universels (KOSKO, 1993), (WANG, 1994), (GLORENNEC, 1999), c'est-à-dire, pour toute fonction réelle continue f définie sur un compact C de Rn , et pour tout 0 , il existe un SIF tel que :

$$\forall\ x\ \in\ C\ ,\ \left\| f\ (\ x\ )\ -\ SIF\quad (\ x\ )\right\| <\ \varepsilon \tag{1.13}$$

Notons, cependant que la propriété d'approximation universelle ne donne pas une méthode de construction du système flou SIF(x), mais elle garantisse seulement son existence. De plus, pour un degré de précision quelconque, il faut utiliser un nombre important de règles floues.

Un système d'inférence floue peut être classifié selon quartes catégories présentées dans le table 1.2 suivant : (SHARMA, 2011)

Classe	***Paramètres***
Logique	Le mécanisme de raisonnement Les opérateurs flous Les types des fonctions d'appartenance La méthode de défuzification
Structurelle	Le nombre des fonctions d'appartenance, des règles, des variables linguistiques

Connective	Antécédents et conséquences des règles, ..
Opérationnel	Les valeurs des fonctions d'appartenances

Table 1.2 : La classification des paramètres du SIF (SHARMA, 2011)

4. Les réseaux neuro-flous

La logique floue et les réseaux de neurones artificiels sont des technologies complémentaires. Ces techniques ont chacune leurs forces et leurs faiblesses .La combinaison de ces deux approches dans un système intégré semble plus prometteuse pour le développement des systèmes intelligents. (MAGALY, 2001).

Réseaux de neurones	**Logique floue**
Avantages	
- Capacité d'apprentissage	- Représentation des connaissances incertaines
- Capacité de génération	- Facilité d'interaction
- Robustesse	- Facilité d'interprétation des résultats
-	- Facilité d'extension de la base de connaissance
Inconvénients	
- Boite noire (manque d'interprétabilité)	- Incapacité de généralisation
- Difficulté de déterminer le nombre de Couches/ neurones	- Dépend de l'exigence d'un expert pour déterminer les règles d'inférence

Table 1.3 Comparaison entre la logique floue et les réseaux de neurones

Afin de résumer l'apport des approches neuro-floues, la table 1.3 regroupe les avantages et les inconvénients des réseaux de neurones et de la logique floue. Cette table montre clairement que la combinaison de ces deux techniques permet de tirer profits des avantages de chacune des deux approches. D'un coté, les réseaux de neurones peuvent améliorer leur transparence, ce qui les rend plus proche des systèmes flous. D'un autre coté, les systèmes flous s'auto-adaptent (réglage automatiques des paramètres), ce qui les rendra plus proche des réseaux connexionnistes (LIN, 1996).

Les systèmes neuro-flous ont suscité l'intérêt croissant des chercheurs dans les domaines scientifiques et d'ingénierie (LIN, 1996), (JANG, 1997). En particulier, les systèmes neuro-flous hybride semblent avoir un intérêt croissant dans le domaine de reconnaissance des formes (MAGALY, 2001), (BARALDI, 1998).

4.1 Définition du système Neuro-Flou

Une définition des systèmes neuro-flous (NF) est donnée dans et reprise dans (RACOCEANU, 2006) selon laquelle : Les systèmes neuro-flous sont des systèmes flous formés par un algorithme d'apprentissage inspiré des réseaux de neurones. La technique d'apprentissage opère en fonction de l'information locale et produit uniquement des changements locaux dans le système flou d'origine. Les règles floues codées dans le système neuro-flou représentent les échantillons imprécis et peuvent être vues en tant que prototypes imprécis des données d'apprentissage (Figure 1.9). On peut aussi noter que les systèmes neuro-flous peuvent être utilisés comme des approximateurs.

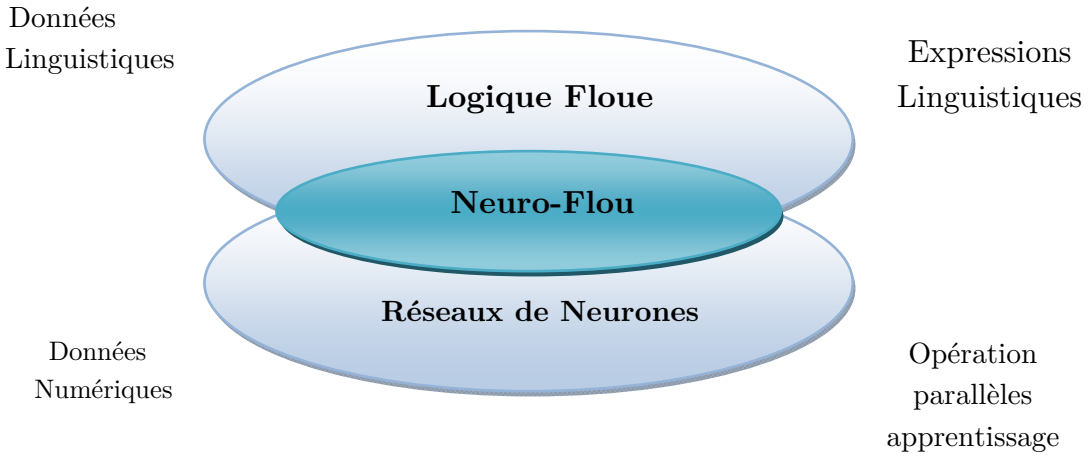


Figure 1.9 Représentation d'un système neuro-flou

Les règles floues codées dans un système neuro-flou représentent les échantillons imprécis et peuvent être vues en tant que prototypes imprécis des données d'apprentissage. Un système neuro-flou ne devrait par contre pas être vu comme un système expert (flou), et il n'a rien à voir avec la logique floue dans le sens stricte du terme. On peut aussi noter que les systèmes neuro-flous peuvent être utilisés comme des approximateurs universels. (LEE, 2005).

Les développements actuels sur ce type de problématique montrent que les performances des NF dépassent celles d'autres méthodes aussi bien en termes de précision des résultats que d'efficacité d'apprentissage (WANG, 2004).

Généralement, les réseaux neuro-flous remplacent les différentes couches cachées des réseaux de neurones par des règles floues (i.e. règles linguistiques). Ils utilisent ensuite des algorithmes d'apprentissage pour définir et optimiser ces paramètres. De plus, les règles d'un système neuro-flou sont transparentes, ce qui permet la validation et la manipulation par un expert. Enfin, les systèmes NF sont très prometteurs dans les cas où les données disponibles sont limitées (MAHABIR, 2006).

4.2. Architecture des systèmes neuro-flous

Diverses associations de ces deux méthodes (réseaux de neurones, logique floue) ont été développées depuis 1988 à nos jours et sont le plus souvent orientées vers la commande de systèmes complexes et les problèmes de classification. Il existe quatre grandes catégories de combinaisons des réseaux de neurones avec la logique floue (RACOCEANU, 2006) : réseau flou neuronal, système neuronal/flou simultanément, modèles neuro-flous coopératifs et modèles neuro-flous hybrides (RACOCEANU, 2006). Ces techniques floues sont utilisées pour augmenter les possibilités d'apprentissage ou l'exécution d'un réseau neuronal.

4.2.1 Système neuronal/flou simultanément

Le réseau neuronal et le système flou fonctionnent ensemble sur la même tâche, mais sans s'influencer. Habituellement le réseau neuronal traite les entrées, ou post-traite les sorties du système flou (Figure 1.10)

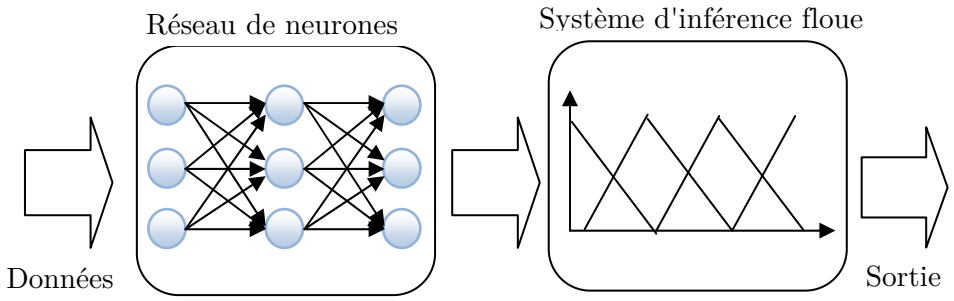


Figure1.10 : Exemple d'association en série d'un réseau de neurone et d'un système

4.2.2 Modèles neuro-flous coopératifs

Le réseau neuronal est employé pour déterminer les paramètres (les règles et les ensembles flous) d'un système flou. Après la phase

d'apprentissage, le système flou fonctionne sans le réseau neuronal. C'est une forme simple des systèmes neuro-flous.

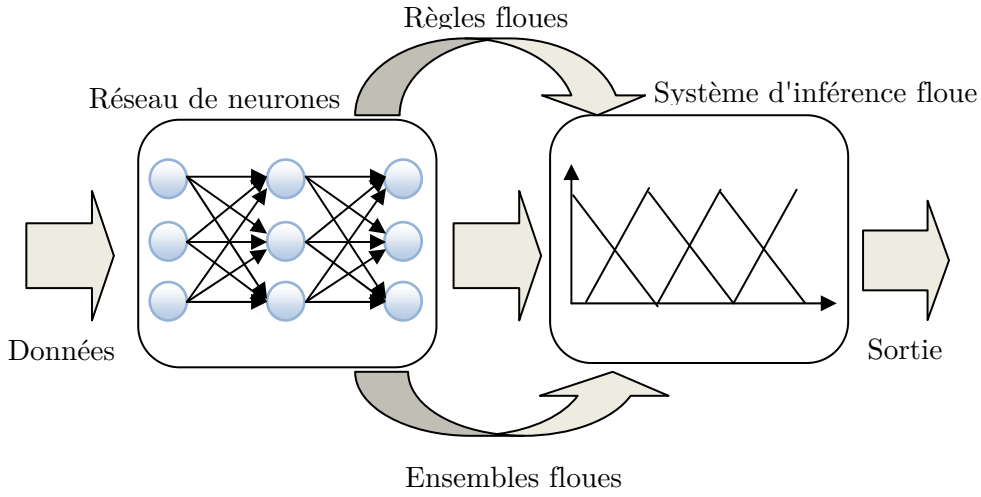


Figure 1.11 : Exemple d'association en parallèle d'un réseau de neurone et d'un système Flou.

4.2.3 Modèles neuro-flous hybrides

Le réseau neuronal et le système flou sont combinés dans une architecture homogène. Il peut être interprété comme un réseau neuronal spécial avec des paramètres flous ou comme un système flou mise en application sous une forme distribuée et parallèle. Plusieurs architectures mettant en œuvre cette approche hybride, sont décrites dans la littérature (LEE, 2005). Parmi ces architectures les plus utilisées on peut citer l'architecture ANFIS

- **ANFIS (Adaptif Neural Fuzzy Inference System):**

Le réseau neuro-flou adaptatif (ANFIS : Adaptif Neural Fuzzy Inference System) est composé d'un ensemble de neurones connectés entre eux par des connexions directes. Chaque neurone modélise une fonction paramétrée ; le changement des valeurs de ses paramètres entraîne le changement de la fonction, de même que le comportement total du réseau adaptatif.

Les nœuds sont de deux types différents selon leur fonctionnalité : des noeuds adaptatifs (carrés) et des nœuds fixes (circulaires) (JANG, 1992). La conception du modèle ANFIS peut être subdivisée en trois étapes : le développement du modèle, apprentissage du réseau et validation et test du

système (OPEYEMI, 2012). D'après (PATNAIK, 2012), (HOSSEINI, 2011), ANFIS présente les avantages suivants :

- Raffine les règles floues Si-Alors pour décrire le comportement d'un système complexe.
- Présente un grand choix d'usage des fonctions d'appartenance.
- Temps de convergence rapide.
- Expertise préalable non requise.

Cependant, le modèle ANFIS est couteux en calcul due au problème de la dimension (DU, 2006) .Autrement l'apprentissage est off-line.

- **Architecture de l'ANFIS**

Dans un ANFIS, les connexions entre neurones sont seulement utilisées pour spécifier le sens de la propagation des stimulations provenant des autres neurones. Pour la structure de ANFIS est composée de cinq couches, et les règle de type si prémisse alors conséquent. ANFIS est l'un de tous premiers systèmes neuro-flou qui existent. Il est très cité dans la littérature car il a prouvé son efficacité avec son algorithme d'apprentissage simplifié : la méthode de descente de gradient et la méthode des moindres carrés. (JANG, 1992)

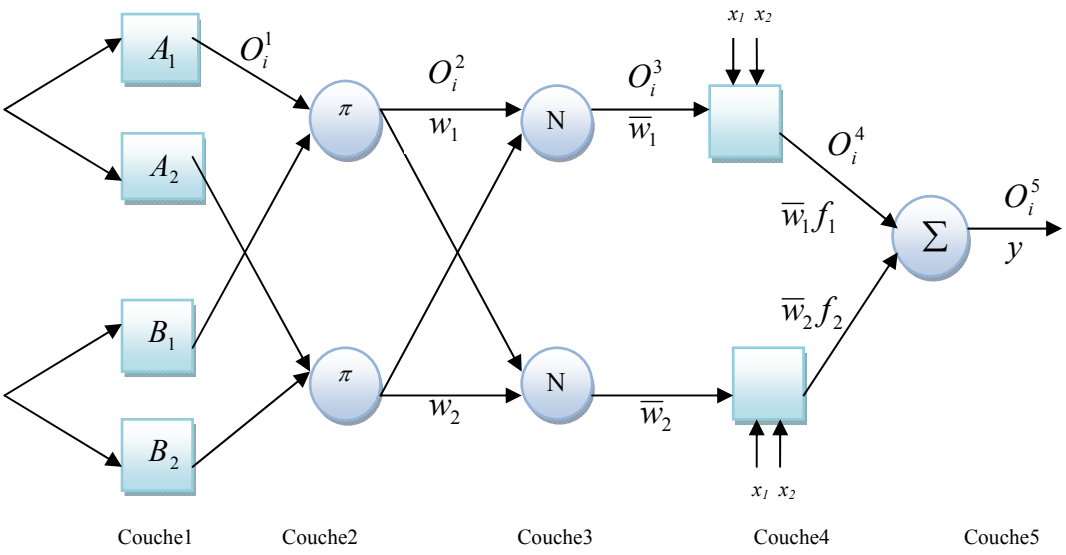


Figure1.12 Architecture d'ANFIS.

La figure présente l'architecture d'un ANFIS formalisant le raisonnement de Sugeno du premier ordre, à deux entrées et une sortie et une base de règles constituée de deux règles, dont une règle est exprimée par :

Règle i : si x est A_i et y est B_i alors $f_i=p_i+q_i+r_i$

Correspondant à l'architecture d'ANFIS qui se compose de cinq couches.

Couche .1 :
Les neurones adaptatifs A_i (B_i) calculent les degrés d'appartenance, l'ensemble des paramètres caractérise les fonctions A_i (B_i). Les paramètres correspondant sont appelés paramètres de la prémisse $\{a_i, b_i, c_i\}$

$$O_i^1 = U_{Ai}(x) \tag{1.14}$$

Généralement $U_{Ai}(x)$ est choisi sous forme de cloche avec son maximum égal à 1 et le minimum égal à 0.

$$U_{Ai}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{\alpha_i}\right)^2\right]^b} \tag{1.15}$$

Où la fonction gaussienne

$$U_{Ai}(x) = \exp\left[-\left(\frac{x-c_i}{\alpha_i}\right)^2\right] \tag{1.16}$$

Couche.2 :
Les neurones fixes modélisent l'opérateur "Et" et calculent la valeur de vérité de chaque règle.

$$w_i = U_{Ai}(x) \times U_{Bi}(x) \tag{1.17}$$

Couche.3 :
Les neurones N sont des neurones fixes, ils effectuent la normalisation de la valeur de vérité de la règle (poids).

$$\overline{w} = \frac{w_i}{w_1 + w_2}, i = 1,2 \tag{1.18}$$

Couche.4 :
Chaque neurone de cette couche est un neurone adaptatif tel que :

$$O_i^4 = \overline{w_i} f_i = \overline{w_i}\left(p_i x + q_i y + r_i\right) \tag{1.19}$$

Les paramètres $\{p_i \; q_i, r_i\}$ sont appelés paramètres de la conséquence.

Couche.5 :

Le neurone de la couche 5 est un neurone fixe, à une entrée donnée, il délivre la réponse du réseau donnée par

$$O_i^5 = \sum_i \overline{w_i} \; f_i \tag{1.20}$$

L'architecture ANFIS est une classe de réseaux adaptatifs proposés par (JANG, 1992). Il peut être vu comme un réseau de neurones non bouclé pour lequel chaque couche est un composant d'un système neuro-flou.

- **Apprentissage de l'ANFIS**

L'ajustement des paramètres de L'ANFIS est réalisé lors de la phase d'apprentissage. Cette étape commence par la construction d'un réseau initial, ensuite applique une méthode d'apprentissage par rétro-propagation Jang a proposé d'appliquer une méthode hybride.

L'algorithme d'apprentissage hybride est une association de la méthode de descente de gradient et de la méthode d'estimation des moindres carrés. La méthode de descente de gradient permet d'ajuster les prémisses en fixant les paramètres conséquents alors que la méthode des moindres carrés ajuste les paramètres conséquents en fixant les prémisses (JANG, 1997).

5. Commande des systèmes non linéaires

5.1 Représentation des systèmes non-linéaires

Un système non linéaire est un ensemble d'équations (différentielles par exemple) non linéaires, d'écrivant l'évolution temporelle des variables constitutives du système sous l'action d'un nombre fini de variables indépendantes appelées entrées ou variables de commande, ou simplement commandes. Les entrées peuvent être choisies en boucle ouverte, qui ne dépendent que du temps, ou en boucle fermée, comme des fonctions des variables mesurées, appelées observations, qui rendent compte de l'état du

système à chaque instant. Aucun système physique n'est complètement linéaire, les méthodes linéaires ne sont donc applicables que dans un domaine de fonctionnement restreint. Certains systèmes sont impossibles à modéliser, même localement, à des systèmes linéaires (SLOTINE, 1991).

La représentation générale d'un système non linéaire est de la forme (1.21) :

$$\begin{cases} \dot{x} = f(x) + g(x)u(t) \\ y = h(x) \end{cases} \tag{1.21}$$

où y est la sortie du système, x est le vecteur d'état et u est le vecteur de commande *.f(x), g(x) et h(x)* sont des fonctions non linéaires du vecteur d'état décrivant le système (LEWIS, 1993).

5.2 Commande des systèmes non linéaires

Le passage automatiquement d'un procédé d'un état particulier à un autre état désiré exprime un ensemble des opérations nommé la commande (LEWIS, 1993).

En réalité n'importe système commandé est soumis à des perturbations et à des variations de paramètres, tel que les bruits, les frottements, ou des erreurs de mesure (Figure 1.13).

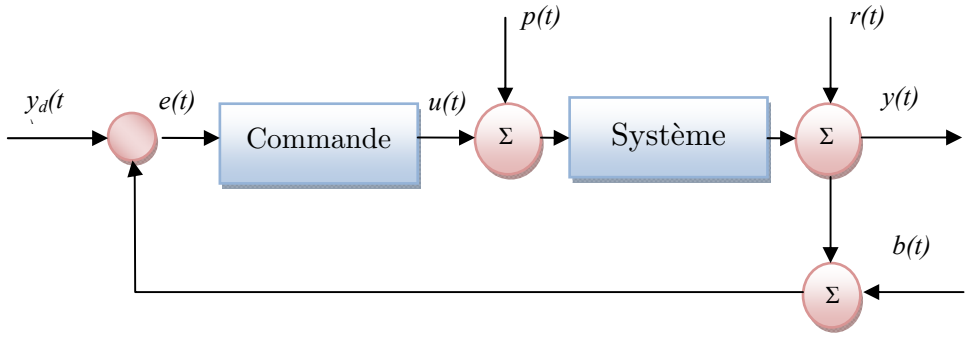


Figure1.13 Structure de commande d'un système non linéaire

où $y_d(t)$: consigne ou signal de référence
$y(t)$: signal de sortie ou réponse
$e(t)$: erreur de suivi
$u(t)$: signal de commande
$p(t)$: perturbation de la commande

$r(t)$: perturbation de la sortie
$b(t)$: bruit de mesure

Nous parlerons de régulation si la consigne $y_d(t)$ est constante dans le temps, dans le cas contraire on dit que la commande est un asservissement ou poursuite de trajectoires. L'objectif majeur de la commande d'un système est d'atteindre un ensemble de performances :

1. Rapidité : elle dépend du temps de montée et du temps d'établissement du régime stationnaire. Elle est basée sur :

Temps de réponse (Tr) : le temps de réponse est le temps nécessaire pour que le régime transitoire ait totalement disparu. (JOHNSON, 2005).

Temps de montée (Tm) : Temps pour lequel la réponse atteint pour la première fois la valeur finale. Il caractérise la vitesse de réaction du système aux premiers instants (JOHNSON, 2005).

2. Précision : elle est caractérisée par :
Erreur statique ($e_s(t)$) : Elle est définit par l'écart entre la consigne et la sortie lorsque le système est en régime stationnaire (JOHNSON, 2005).

$$e_s = \lim_{t \to \infty} (y(t) - y_d(t)) \tag{1.22}$$

Dépassement (O_{max}) : en pratique il est recommandé pour avoir un système agile un dépassement de 10% (JOHNSON, 2005).

$$O_{\max} = \frac{y_{\max} - y_\infty}{y_{\max}} \tag{1.23}$$

3. Robustesse : Elle est sans doute le paramètre le plus important et le plus délicat. Nous disons qu'un système est robuste si la régulation fonctionne toujours en dépit des variations paramétriques. Un régulateur doit être capable d'assurer sa tâche même avec ces changements afin de s'adapter à des usages non prévus/testés (dérive de production, vieillissement mécanique, environnements extrêmes) (JOHNSON, 2005).

4. Stabilité : Nous disons d'un système qu'il est stable si à toute entrée bornée (en amplitude) il répond par une sortie bornée (SLOTINE, 1991).

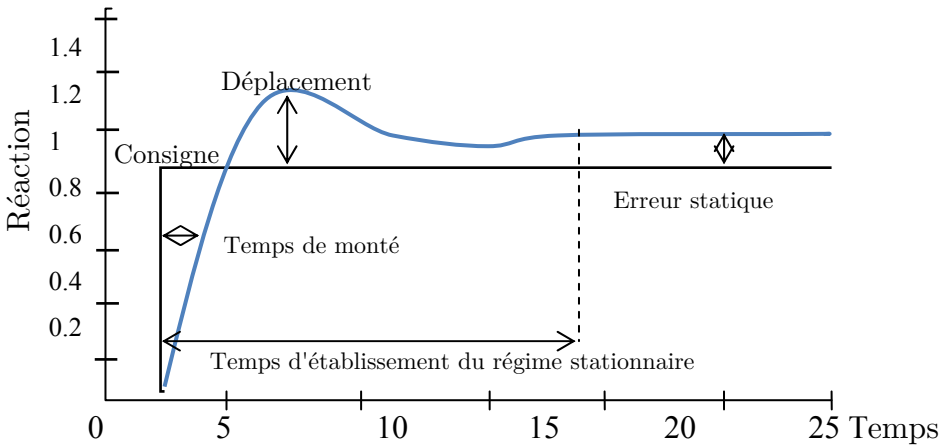


Figure 1.14 Représentation explicative des performances d'un système

5.2.1 Commande proportionnelle, intégrale et dérivée (PID)

Un régulateur PID (Proportionnel, Intégral, Dérivé) est un organe de contrôle généralement utilisé dans le domaine de l'industrie, permettant d'effectuer une régulation en boucle fermée d'une grandeur physique d'un système industriel ou « procédé ». (JOHNSON, 2005).

La commande PID est la somme de trois actions, proportionnelle, intégrale et dérivée, elle est donnée par la relation suivante :

$$u(t) = K_p e(t) + K_d \dot{e}(t) + K_i \int e(t)dt \qquad (1.24)$$

où K_p, K_d *et* K_i sont des gains constants à déterminer.

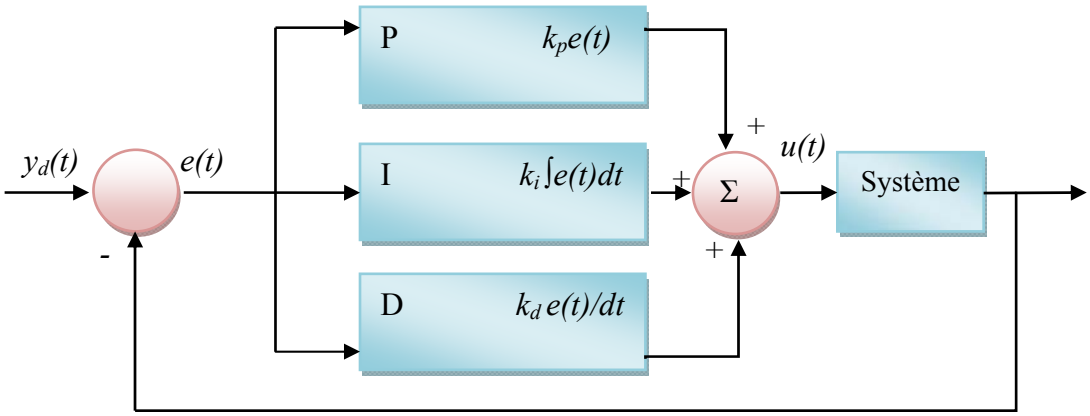


Figure 1.15 : Architecture d'un contrôleur PID.

- **Réglage d'un PID :**

Il consiste à déterminer les gains du PID afin d'obtenir une réponse adéquate du procédé. L'objectif est d'être robuste, rapide et précis. Il faut pour cela diminuer le temps de montée, éliminer l'erreur statique et diminuer le dépassement maximum. L'influence des trois gains d'un PID sur ces performances est représentée dans (JOHNSON, 2005). Il existe des méthodes analytiques permettant de calculer les composantes du correcteur PID, mais elles sont assez complexes et sont peu utilisées pour les systèmes non linéaires. Des méthodes empiriques pour les systèmes linéaires permettent de faciliter amplement la détermination des gains du régulateur PID comme la méthode de Ziegler-Nichols et méthode de Chien-Hrones Reswick. Le problème est que ces méthodes n'ont pas prouvé leurs efficacités pour les systèmes non-linéaires (JOHNSON, 2005).

- **Limitation et Faiblesse du correcteur PID**

Les régulateurs PID répondent à plus du 90% des besoins industriels. Malheureusement, malgré l'expérience acquise au fil des ans, les valeurs choisies pour les paramètres P, I et D ne sont pas toujours satisfaisantes, ni adaptées au processus à régler. Ils existent des méthodes analytiques permettant de calculer les composantes du correcteur PID, mais elles sont assez complexes et sont peu utilisées pour les systèmes non linéaires.

Des méthodes empiriques existent pour les systèmes linéaires et elles permettent de faciliter amplement la détermination des gains du régulateur PID comme la méthode de Ziegler-Nichols et méthode de Chien-Hrones Reswick. Le problème est que ces méthodes n'ont pas prouvé leurs efficacités pour les systèmes non-linéaires (JOHNSON, 2005).

5.2.2 Commande adaptative

La commande adaptative est un ensemble de techniques permettant de fournir une approche systématique pour l'ajustement automatique d'un régulateur en temps réel, dans le but d'achever ou de maintenir des performances désirées pour le système de commande lorsque la dynamique du procédé est inconnue et/ou change au cours du temps (LANDAU,1998), (ASTRÖM, 1989), (JACQUES, , 1991).

Le système de commande adaptative mesure un certain indice de performance (*IP*) du système à commander à partir de l'écart entre l'indice de performance désiré et l'indice de performance mesuré. Le mécanisme d'adaptation commande certains paramètres du système ajustable ou introduit un signal supplémentaire de commande d'après une certaine stratégie afin de minimiser l'IP. (Figure.1.16) représente le principe général d'un système dans une plage donnée de commande adaptative (ISE, 1992).

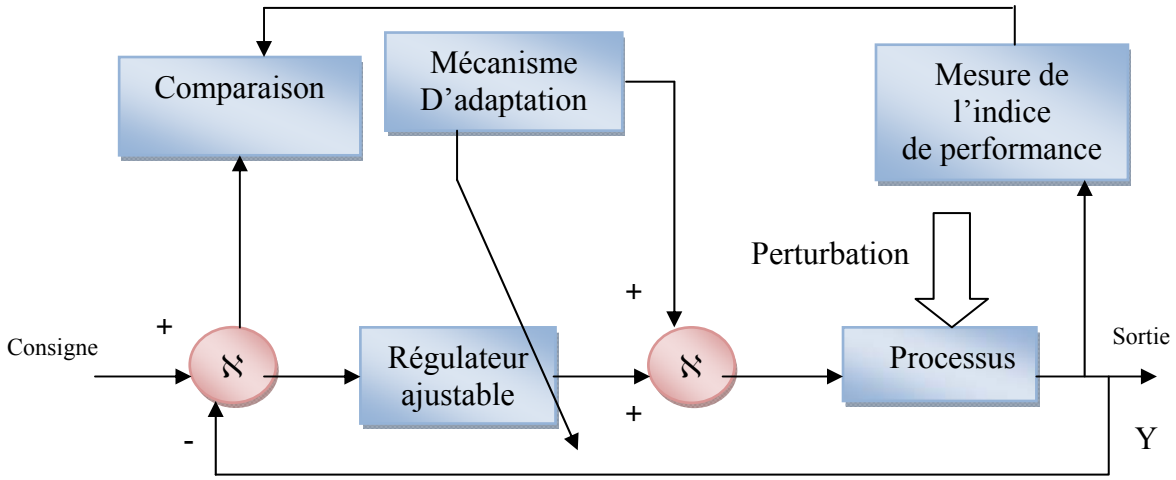


Figure 1.16 : Principe d'un système de commande adaptative

Trois approches ont été essentiellement considérées pour le développement des stratégies de commande adaptative destinées aux procédés à paramètres inconnus et/ou variables dans le temps.

- Approximation des stratégies de commande optimale stochastique «Duale ».

- Système de commande auto - ajustable (self- tunning control).

- Commande adaptative à modèle de référence (MRAC) (ISE, 1992).

5.2.3 Commande robuste

La commande robuste est un type de commande qui vise à garantir les performances et la stabilité d'un système face à des perturbations du milieu et les incertitudes du modèle. En effet, le modèle mathématique qui modélise un système réel est une représentation qui vise à approximer au mieux, avec des hypothèses simplificatrices, le système qu'on veut commander. Il existe donc un écart entre le comportement observé du système réel et son modèle interne. La commande robuste vise à déterminer une loi de commande qui soit capable

de garantir des critères de performance et stabilité pour un système dont le modèle varie autour du modèle théorique ou nominal (SLOTINE, 1991).

Par la même approche, on peut rendre le système robuste face aux perturbations extérieures (par exemple, une rafale de vent sur un avion de ligne) qui en somme peuvent être considérées comme une modification du modèle interne. Parmi les commandes robustes utilisées dans la littérature est la loi de commande par mode glissement (sliding mode).

6. Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les techniques de l'intelligence artificielles en particulier les réseaux de neurones, logique floue et l'hybridation entre les deux techniques afin d'obtenir un réseau neuro-flou plus puissant qui combine les propriétés des deux techniques .

Nous avons aussi présenté l'un des réseaux neuro-flous ANFIS qui est assujetti à quelques limites que les chercheurs ont essayé de résoudre par d'autres techniques intelligentes dans le domaine de la commande des systèmes non linéaires. Une attention particulière a été prêtée au contrôleur PID ainsi que le principe de la commande des systèmes non linéaires. Cet aperçu sur le PID et le réglage de ces paramètres sera pris en compte dans le quatrième chapitre. La dernière section de ce chapitre a été consacrée aux types des commandes, notamment la commande adaptative qui sera le noyau des commandes intelligentes présenté dans ce travail. L'étude de ces techniques nous aide dans les chapitres qui suivent pour exploiter des approches intelligentes dans la commande des systèmes non linéaires.

Les chapitres 2 ,3 et 4 présentent une application de trois approches différentes. Les deux premières basées sur des commandes intelligentes pour la commande des systèmes non linéaires, tandis que le troisième chapitre est basé sur un algorithme évolutionnaire pour un objectif d'optimisation.

Chapitre 2

Application de la commande Neuro-Floue STFIS pour le contrôle des systèmes non linéaires

1. Introduction

Actuellement les systèmes à inférence floue sont employés dans de nombreux domaines. Utilisés principalement lorsque le modèle mathématique du système physique est difficile à élaborer. Ils exploitent des règles floues tirées d'une expertise humaine pour modéliser le comportement dynamique du système. Ces règles sont du type :"SI ALORS ". Les principaux avantages des techniques floues sont l'approche naturelle de la modélisation et la bonne interprétabilité de la description, en employant des règles linguistiques.

Cependant, comme il n'y a aucune méthode formelle pour déterminer ses paramètres (ensembles et règles floues), l'exécution d'un système flou peut prendre beaucoup de temps. Dans ce sens, il serait intéressant de disposer d'algorithmes permettant l'apprentissage automatique de ces paramètres.

L'une des méthodes qui permet de répondre à ces exigences est la combinaison des deux techniques qui nous donne les systèmes neuro-flou.

L'utilisation des contrôleurs neuro-flou pour le contrôle des systèmes non linéaires a connu une croissance rapide ces dernières années. La plupart des contrôleurs flous développés jusqu'à maintenant ont été à base des règles. En effet un système d'inférence floue peut désormais non seulement de prendre des informations linguistiques des experts humains, mais aussi de s'adapter à l'utilisation des données numériques (paires d'entrée / sortie) pour obtenir de meilleures performances.

Le travail présenté dans la deuxième contribution de la thèse concerne une commande neuro-floue intelligente. Il s'agit d'un le système d'inférence floue auto ajustable (Self Tunable Fuzzy Inference System STFIS) appliqué à un système non linéaire. L'objectif principal est l'interprétation en ligne et l'optimisation de la base des règles, avec un minimum de temps pour s'adapter au système qui sera comparé avec un contrôleur classique PID.

2. Architecture neuro flou hybride

L'utilisation conjointe des réseaux de neurones et de la logique floue, permet de tirer les avantages des deux méthodes : les capacités d'apprentissage de la première et la lisibilité et la souplesse de la seconde.

En effet plusieurs architectures, mettant en œuvre cette approche hybride, sont décrites dans la littérature (LEE, 2005). Ces architectures peuvent être classées en trois groupes (WANG, 2001) selon le type de règles floues qu'elles intègrent :

$$si(X_1 \ est \ A_1)et(X_2 \ est \ A_2)ET...ET(X_N \ est \ A_N)Alors(Y \ est \ C)$$

$$C = \begin{cases} B \\ f(x_1, x_2, ..., x_n) \\ \theta \end{cases} \tag{2.1}$$

où x_i $(i = 1,2,...,n)$, Y représentent respectivement les variables d'entrée et de sortie, A_j les ensembles flous d'entrée; B, $f(x_1,x_2,...,x_n)$ et θ représentent respectivement, l'ensemble flou de sortie, une fonction linéaire des variables d'entrée et un composant singleton.

2.1 Variables d'entrées du contrôleur neuro-flou

Un contrôleur flou permet de déterminer la commande et de l'appliquer à un processus à partir de la valeur u ,(Figure 2.1), qui est la variable de sortie de ce contrôleur ; celle-ci est elle-même déterminée à partir des valeurs des variables d'entée e, Δe, (Figure 2.1) du contrôleur par des relations floues, ou règles floues. Théoriquement, le nombre d'entrée n'est pas limité. En pratique il n'est pas rationnel d'utiliser plus de trois variables d'entrée car la détermination des règles devient trop complexe.

Le gain de la sortie u du régulateur flou normalise sa sortie dans l'univers de discours de la commande. De plus, il joue un rôle dans la stabilité système et l'élimination des erreurs en régime permanent. Alors que les gains d'entrées e, Δe ont un rôle de la normalisation des variables linguistiques en vue de leurs utilisations par le contrôleur. Ces gains affectent aussi les performances de la réponse du système en régime transitoire.

Le choix de ces gains peut se faire d'une manière subjective (essais /erreurs) de sorte à obtenir la meilleure performance possible.

Pour le contrôleur décrit dans la Figure 2.1, nous n'avons fait intervenir que deux variables d'entrée :

$$\begin{aligned} e(t) &= y_d(t) - y(t) \\ \Delta e(t) &= e(t) - e(t-1) \end{aligned} \tag{2.2}$$

avec :$e(t)$ est l'écart entre la consigne $y_d(k)$ et le signal de sortie du processus $y(k)$, $e(t)$ est la variation du signal d'écart à l'instant t.

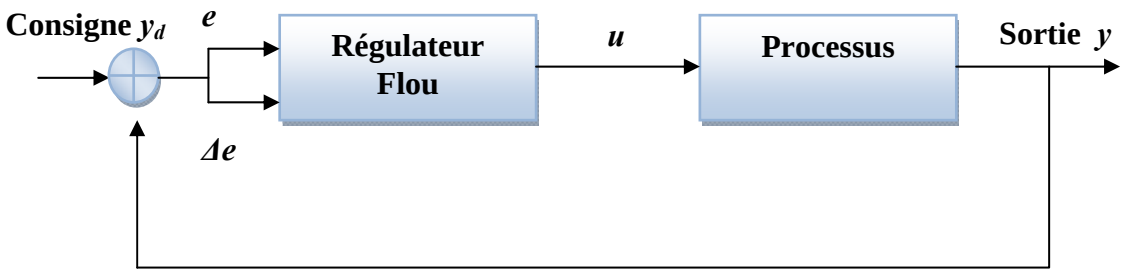


Figure 2.1 Schéma d'un système asservi par un régulateur flou

2.2 Architecture du "control JEAN et Mini-JEAN"

Dans le contrôle des processus dynamiques, le contrôleur doit reproduire la fonction de transfert inverse du système pour déterminer la commande à fournir. Cela est aisé dans certains cas simples, par exemple quand le système est modélisé sous une forme linéaire, mais le plus souvent la structure du système est inconnue, soit elle présente des non linéarités ou bien des paramètres non modélisables. Nous nous plaçons dans cette hypothèse d'avoir aucune connaissance à priori sur le processus à contrôler. Dans ce cas, (JORDAN, 1991) propose la méthode du 'distal control' sous le nom de JEAN (Jordan method Extended for Adaptive Neuro-control).

Cette architecture est présentée dans la figure.2.2 qui nécessite la présence de deux systèmes neuro-flous.

-Un premier système pour identifier le processus (modèle).

-Un second système pour contrôler le processus (Contrôleur).

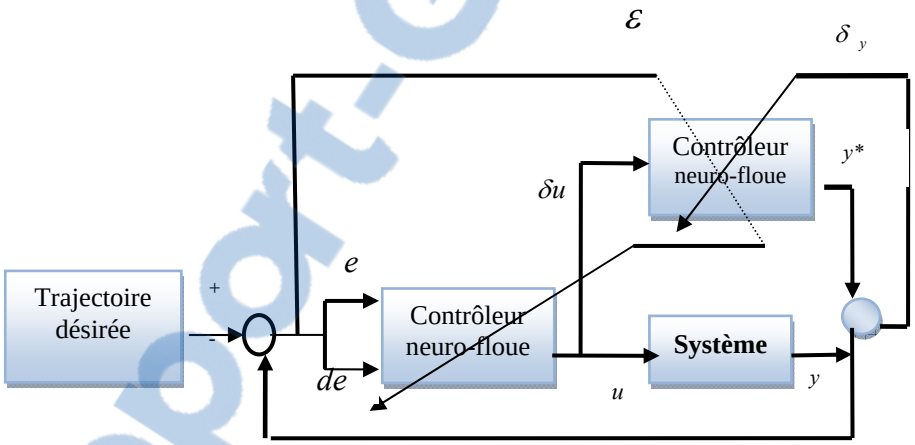


Figure 2.2.Architecture JEAN

Cette méthode nécessite deux étapes successives : l'identification du processus par le réseau modèle, puis l'optimisation du réseau contrôleur. L'ajustement des poids synaptiques (ou des conclusions des règles) de ce dernier est effectué en rétropropageant la valeur u obtenue par la rétropropagation à travers une fonction coût basée sur l'erreur en sortie.

$$e = y - y^* \tag{2.2}$$

Souvent il n'est pas nécessaire de connaître précisément le modèle du processus pour obtenir un contrôle correct.

En effet, un modèle imprécis du système n'altère pas la minimisation de la fonction coût, il modifie seulement le trajet pris par l'algorithme d'optimisation dans l'espace des actions. On ne suivra sans doute pas la plus forte pente, mais on restera sur un chemin descendant. Nous arrivons à une architecture ''mini-JEAN'' avec un seul réseau contrôleur, dont l'optimisation s'effectue en rétropropageant directement l'erreur de sortie.

Nous avons comparé l'architecture mini-JEAN avec l'architecture JEAN. Des performances équivalentes ont été obtenue, tant pour la vitesse d'optimisation que pour l'erreur moyenne en généralisation. Par contre, le temps de calcul est nettement en faveur de mini-JEAN.

Les chercheurs ont étendu ces résultats en remplaçant dans ces architectures le réseau de neurones par un SIF dont certains paramètres sont à optimisés. Ils ont également observé que malgré quelques différences dans la conduite de l'optimisation et dans la commande fournie le recours à la version simplifiée qui n'induit pas de dégradation de performances. C'est pourquoi, notre préférence va cette méthode (mini-JEAN), (ZEMALACHE, 2008).

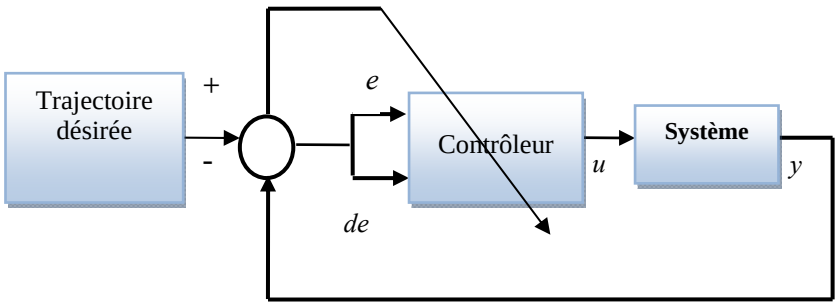


Figure2.3.Architecture mini-JEAN.

3. Système d'inférence floue avec un réglage automatique(STFIS)

Le STFIS (Self tunig Fuzzy Infernce System) est un contrôleur neuro-flou qui présente une analogie structurelle complète avec un système d'inférence floue de type Takagi Sugeno d'ordre zéro (TAKAGI, 2005). C'est-à-dire dans lequel les conclusions des règles sont nettes, et donc la nécessité de défuzzyfication disparaître. Ce système d'inférence floue peut être schématisé dans (Figure 2.4) sous la forme d'un réseau de quatre couches (TAKAGI, 2005).

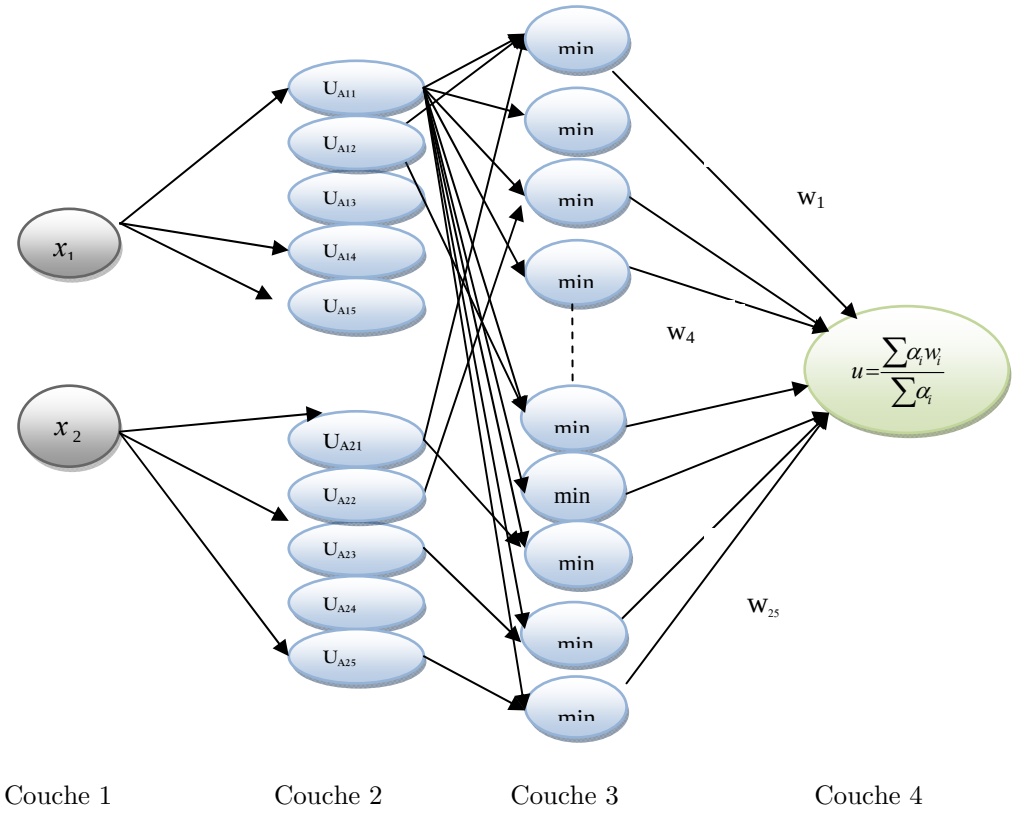


Figure 2.4 : Architecture STFIS

Le STFIS se compose de quatre couches (BERENJI, 1992) :

Couche 1 : reçoit des entrées,

Couche 2 : calcule les degrés de ces entrées d'adhésion à leur sous-ensemble flou. Les coefficients de pondération du réseau entre la première couche et cette couche correspondent aux paramètres définissant les fonctions d'appartenance.

Couches 3 : calcule les valeurs de vérité. Les poids entre deux couches cachées définissent l'opérateur ET choisi (RACOCEANU, 2006).

Couche 4 : la couche de sortie. Le poids w_i du réseau entre la troisième et quatrième couche correspond à des parties de conclusion des règles.

3.1 Apprentissage du réseau STFIS

L'apprentissage de ce type de réseau est un apprentissage supervisé, il se fait par étapes (BRUNET, 1996). Au début, la descente de gradient ne porte que sur les poids w_i de la troisième couche, avec un gain relativement fort (de l'ordre de l'unité) pour la première itération de façon à se rapprocher

rapidement de la solution, puis le gain est réduit . La convergence est souvent très rapide à ce niveau, compte tenu des connaissances déjà intégrées dans les poids des autres couches. Suivant le but recherché et si la décroissance de la fonction coût choisie le permet. On peut limiter l'apprentissage à cette seule couche. Ce procédé permet d'apprendre un jeu de règles, avec des fonctions d'appartenances figées.

Selon la structure utilisée de la méthode de descente du gradient, a pour but d'ajuster les poids de la dernière couche du réseau (Figure 2.4). Le principe général de cette méthode peut être résumé comme suit :

- à chaque itération, nous modifions les poids de la couche de sortie.
- la modification des poids se fait dans le sens opposé au gradient de la fonction coût.
- le processus se répète jusqu'à ce que les poids de la dernière couche aient convergé.

La fonction coût est donnée par :

$$E = \|y - y_d\| \tag{2.3}$$

Si on choisit la norme euclidienne, on trouve :

$$\begin{aligned} E &= \frac{1}{2}\varepsilon^2 \\ \varepsilon &= y(x_1, x_2) - y_d(x_1, x_2) \end{aligned} \tag{2.4}$$

Avec est l'erreur d'approximation, l'équation de descente de gradient sur les poids s'écrit sous la forme :

$$w(k+1) = w(k) - \eta \frac{\partial E}{\partial w} \tag{2.5}$$

avec η : le gain ou le pas du gradient ou vitesse d'apprentissage entre deux échantillons. Cette valeur gère la vitesse à laquelle les paramètres vont converger vers leur valeur finale. Plus il est grand, plus la convergence est rapide, par contre si cette valeur est trop faible, la convergence risque d'être extrêmement longue.

Pour éviter l'inflation des poids et permettre au réseau de converger, on ajoute à la fonction coût E un terme proportionnel à la somme des carrés des poids : Σw_i^2, cette méthode est extrapolée de méthodes de régression non

linéaire qui permet de gérer le compromis entre l'erreur d'apprentissage et l'erreur de généralisation. La méthode consiste à modifier la fonction coût, au terme quadratique, la nouvelle fonction coût devient :

$$J = E + \lambda \sum w_i^2 \tag{2.6}$$

λ est un paramètre qui doit être ajusté de manière empirique. Il contrôle l'augmentation des paramètres. La dérivée de la fonction coût devient :

$$\frac{\partial J}{\partial w} = \frac{\partial E}{\partial w} + 2\lambda w \tag{2.7}$$

$\partial E/\partial w$ est le terme calculé précédemment.
On obtient alors :

$$w(k+1) = w(k) - \eta \frac{\partial E}{\partial w} - \beta w(k) \frac{\alpha_i}{\sum \alpha_i} \tag{2.8}$$

où $\beta=2\eta\lambda$ est le coefficient du terme "*Weight decay*". S'il est trop petit par rapport au gain, l'effet de la régression des poids ne se fera pas sentir. En revanche, s'il est trop important, les poids restent proches de zéro, car l'effet de diminution des poids sera trop important par rapport à l'augmentation due à la descente de gradient normale. Dans ce cas, le système d'inférence floue n'apprendra rien. Il est également important que le terme de régression soit proportionnel au déclenchement de la règle. Si nous n'introduisons pas ce coefficient, la valeur du paramètre w_i baisserait quand la règle n'est pas déclenchée. Lorsque, par exemple, le système évoluait suffisamment longtemps sans déclencher une règle, le paramètre correspondant tendrait vers zéro, et l'information est perdue. Les résultats de la méthode restent toujours dépendants du choix *empirique* des paramètres du réseau.

4. Application de la commande neuro floue basée STFIS sur les systèmes non linéaires

L'approche proposée a été utilisée pour la commande de deux systèmes non linéaires, masse ressort et pendule inversé dont l'architecture adoptée est donnée dans (Figure 2.5) suivante :

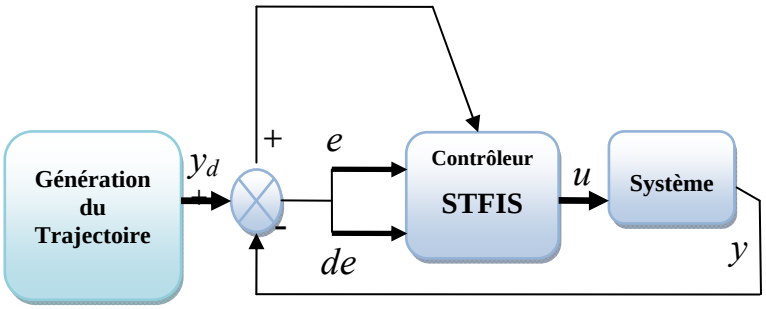


Figure. 2.5 .Architecture adopté.

4.1 Système masse-ressort

Le premier système non linéaire masse-ressort considéré dont le modèle dynamique est donné en (2.9).

$$y''(t) + y'(t) + y(t) + y^3(t) = u(t) \tag{2.9}$$

Dans cette étape, nous forçons les systèmes (2.9) pour suivre le point de consigne défini ($m = 2$ *metre*). Nous utilisons l'architecture connue sous le nom (mini-JEAN) comme dans (Figure 2.3). Pour les paramètres réseau de STFIS sont indiqués dans le tableau 2.1, les cinq fonctions d'appartenance de type sigmoïde et gaussiennes sont tracées dans (Figure 2.6). Classiquement ces coefficients sont choisis par essais successifs.

β	η	λ
0.00005	1	0.9

Table 2.1 : Paramètres du réseau STFIS

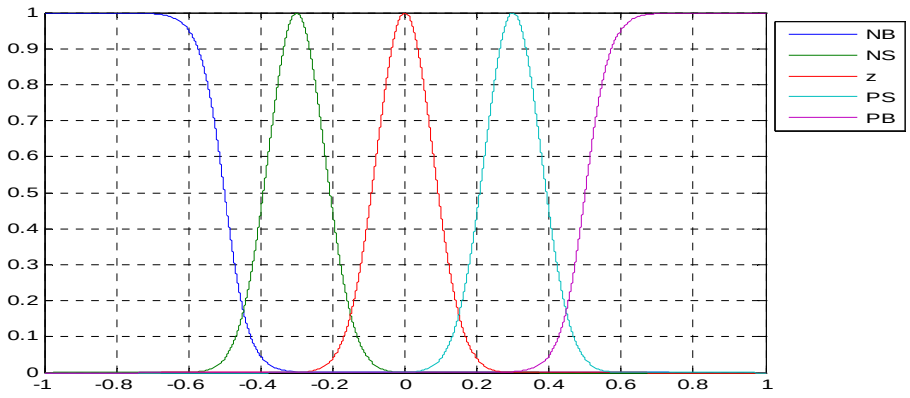


Figure 2.6 : Fonctions d'appartenances (sigmoïde Gaussiennes)

L'erreur de position pour le processus à commander est donnée dans la (Figure 2.7). Le même système est commandé par le contrôleur PID, les résultats de l'erreur de position sont montrés dans la (Figure 2.8). De ces deux figures, l'erreur converge vert zéro rapidement (après < 2.7s) pour le STFIS par rapport au PID (> 10s). Ce qui explique la fiabilité du contrôleur intelligent proposé STFIS. Notre contrôleur présente une fonctionnalité de suivi très intéressante et est en mesure de répondre aux différentes conditions dynamiques.

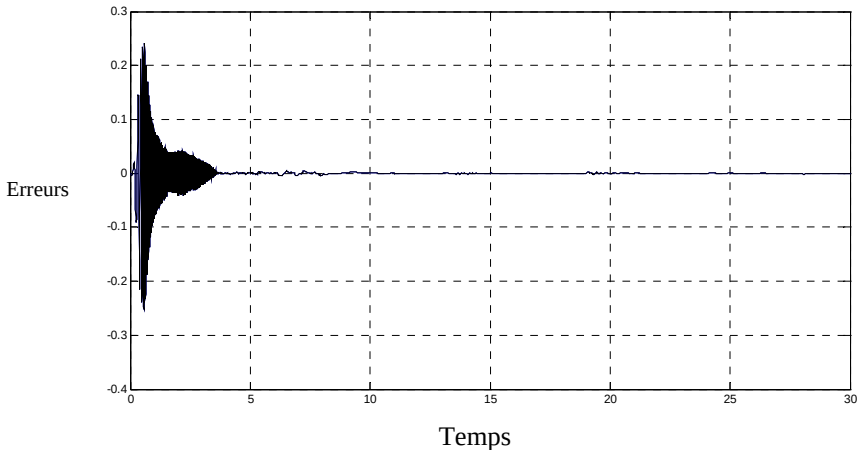


Figure 2.7 : Erreur en position du système commandé par le contrôleur STFIS

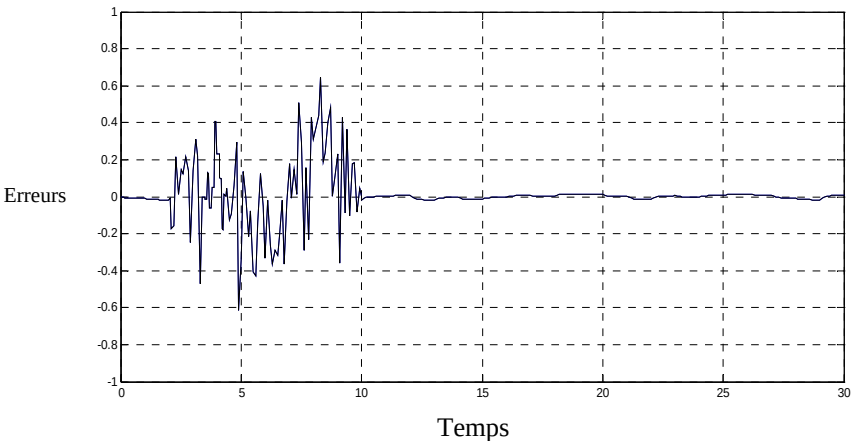


Figure 2.8 : Erreur en position du système commandé par le contrôleur PID

Un bruit blanc est injecté à la sortie du système pour tester l'adaptation et la robustesse du contrôleur (STFIS). Le système bruité masse-ressort est contrôlé par STFIS et aussi par le contrôleur PID. Selon les résultats dans (Figure 2.9) et (Figure 2.10), le dispositif de commande proposée a adapté ses règles face au bruit injecté (erreur de position est inférieur à 10^{-4}), tandis que les erreurs obtenues par le contrôleur PID ont augmenté nettement.

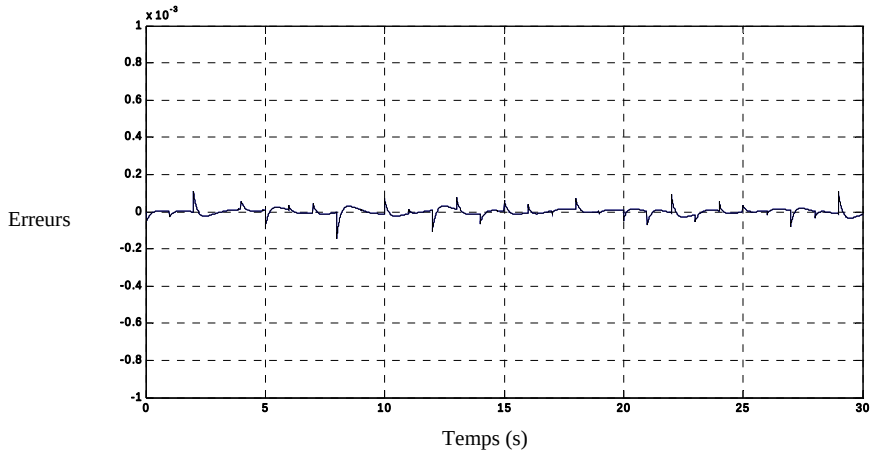


Figure 2.9 : Erreur en position du système commandé par le contrôleur STFIS avec bruit

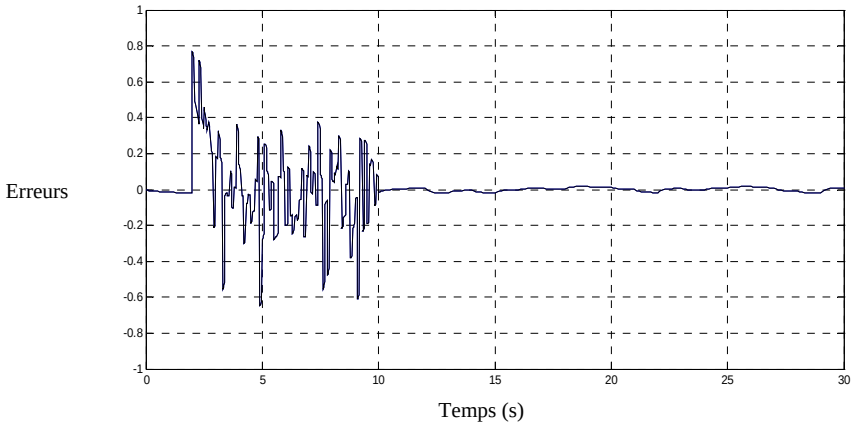


Figure 2.10 : Erreur en position du système commandé par le contrôleur PID avec bruit

4.2 Pendule inversé

Pour donner plus de performance au contrôleur STFIS, la deuxième commande du système se caractérise par leur forte non-linéarité du pendule inversé. L'équation dynamique de pendule inversé est décrite dans l'annexe.

L'objectif de la commande est de garder l'angle $\boldsymbol{\theta}$ nulle pendant le mouvement du charriot ($\boldsymbol{\theta}_d = 0$).

Dans cette étape, nous forçons le système dans la figure A.2 pour suivre la trajectoire sinusoïdale. L'erreur en position angulaire du processus contrôlé est donnée à (Figure 2.12) qui est presque nulle ; à partir de ce résultat, il est

clair que la l'approche intelligente STFIS est entièrement satisfaisante avec une performance remarquable.

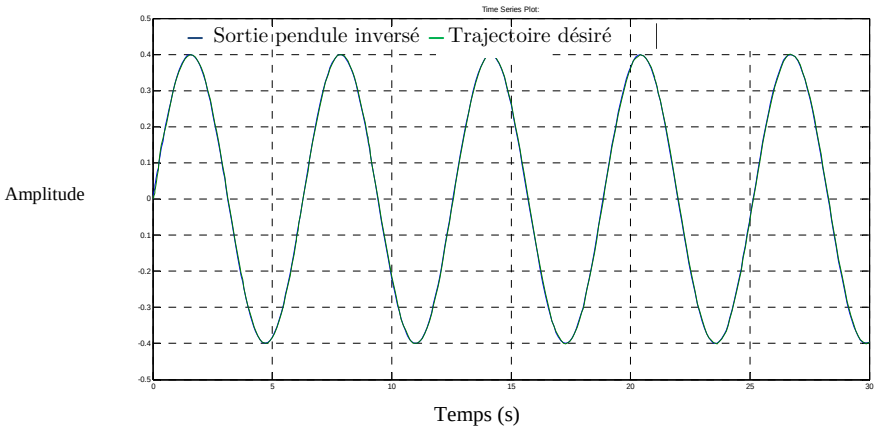


Figure 2.11 : Positions réelle et désirée du Pendule inversé par STFIS

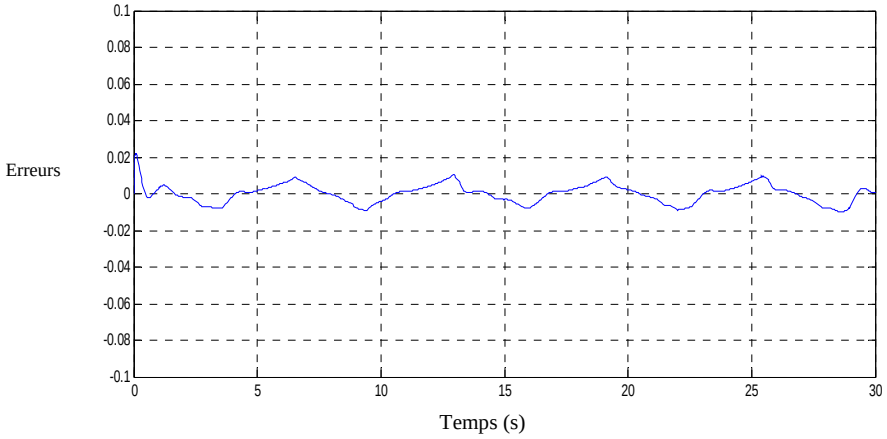


Figure 2.12 : Erreurs en position angulaire par STFIS avec un pendule inversé

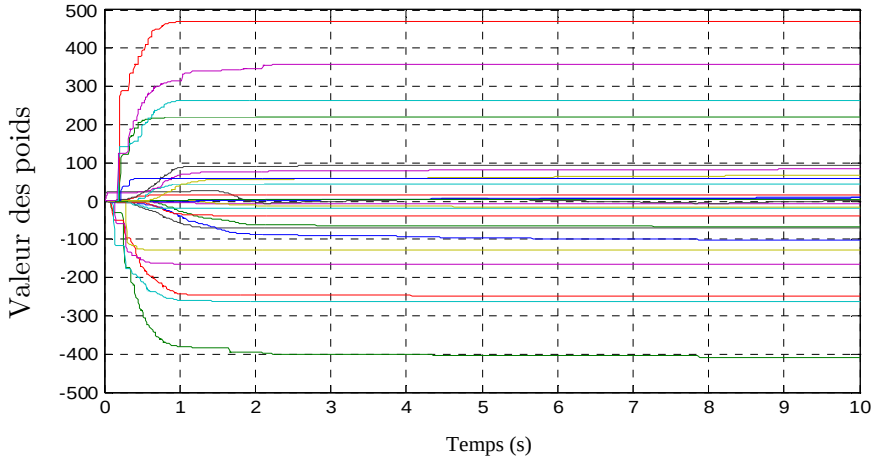


Figure 2.13 : Evolution des poids du réseau STFIS (weights)

Nous pouvons voir clairement la performance du contrôleur intelligent STFIS proposé. (Figure 2.13) montre la convergence et la stabilité des paramètres (poids) de la dernière couche obtenue par optimisation en ligne après un certain temps d'apprentissage.

5. Test de robustesse

Le problème des réseaux dont l'apprentissage est réalisé en ligne est qu'ils ne peuvent pas approximer avec exactitude les données, plus précisément les paramètres de l'algorithme.

Pour tester la robustesse de la loi intelligente basée STFIS, nous avons donné des valeurs différentes au paramètre λ qui doit être ajusté d'une manière empirique pour contrôler l'augmentation des poids.

	Valeurs de λ							
	0.9	0.7	0.5	0.3	0.1	0.09	0.008	0.005
Erreur	0.0102	0.0707	0.1082	0.1353	0.1620	0.1993	0.2041	0.2150

Table 2.2 : Impact du paramètre λ sur l'erreur de position via STFIS

Le résultat décrit dans la table 2.2 après la fixation du paramètre (β= 0.0009), explique clairement l'impact de ce paramètre sur la moyenne quadratique de l'erreur.

6. Conclusion

Nous avons présenté et mis en œuvre dans ce chapitre une technique d'optimisation permettant un ajustement en ligne des paramètres du réseau d'un contrôleur neuro-flou STFIS pour la commande intelligente d'un système non linéaire. En effet, nous avons obtenu une optimisation en ligne d'un système d'inférence floue de type Takagi Sugeno d'ordre zéro. Une comparaison entre le contrôleur PID et STFIS montre l'amélioration de la capacité d'adaptation même en présence d'un bruit pour les deux contrôleurs.

Pour tester la robustesse de cette loi intelligente, nous avons remarqués après une étude que les paramètres de l'algorithme d'apprentissage ont un rôle important. Ils influent directement sur l'erreur de position, ce qui indique que la robustesse du système est assurée avec le bon choix de ces paramètres. D'ou le recourt à l'optimisation des paramètres du STFIS.

Chapitre 3

Architecture SAFIS pour la commande intelligente des systèmes non linéaires

1. Introduction

Dans la littérature le contrôle des processus complexes est mal défini. Les dernières années ont vu une croissance rapide dans l'utilisation des régulateurs flous. Nous notons que la plupart des systèmes non linéaires sont caractérisés par des paramètres incertains, ce qui complique leur contrôle et pour améliorer leur performance. Ces incertitudes conduisent souvent à des différentes lois de commande. Dans cette situation il est recommandé d'utiliser les techniques de l'intelligence artificielle.

La robotique est devenue une réalité indispensable dans le monde de la production industrielle, et aussi dans des nouvelles applications et des secteurs non industriels.

En particulier, le modèle d'un système non linéaire n'est pas toujours disponible et les paramètres font l'objet d'incertitudes causées par les frottements.

Dans le même contexte, les algorithmes de commande ne sont pas restés à l'écart du développement, en ayant pour objectif d'améliorer au maximum les caractéristiques de vitesse et précision. Une solution consiste à introduire une commande neuro-floue pour assurer la robustesse de ces systèmes non linéaires. Les réseaux neuro-flous par leur propriété d'approximation et d'adaptation sont un atout majeur pour être utilisé dans les lois de commande.

Parmi les lois de commande (SLOTINE, 1991) la loi de commande couple calculée (Computed Torque) qui est une stratégie de contrôle de mouvement efficace pour des systèmes non linéaires.

Nous présenterons dans ce chapitre le principe de l'approche d'un système d'inférence flou avec un mécanisme d'apprentissage séquentiel dans une architecture nommée SAFIS (Sequential Adaptive Fuzzy Inference System). L'apprentissage du réseau de cet algorithme est séquentiel qui repose sur l'approche GAP (HUANG, 2004) et le filtre de KALMAN. La structure de loi de commande couple calculée basée sur les réseaux SAFIS est utilisée pour la commande adaptative des systèmes non linéaires (HAI-JUN, 2006).

L'algorithme SAFIS a été appliqué dans la commande d'un robot manipulateur SCARA trois degrés de liberté (poursuite de trajectoire) par une commande adaptative basée sur cette approche, en particulier pour l'estimation des paramètres du modèle du robot manipulateurs afin de les injecter dans la loi de commande adaptative couple calculée (computed torque).

Une étude sur l'impact de la sélection des paramètres prédéfinis de l'algorithme SAFIS a été intégrée pour tester la robustesse de la loi de commande et du système.

2. Notions sur les robots manipulateurs

2.1 Constitution géométrique des robots manipulateurs

Un robot manipulateur est une chaîne cinématique constituée de *n+1* corps, possédant deux corps particuliers et des actionneurs. Le premier corps

particulier, appelé base, est le socle du robot ; le second, appelé terminal, est le support d'un préhenseur ou d'un outil de travail (ANGELES, 2002). Les n corps sont liés entre eux par des liaisons (articulations), suivant une structure de chaîne, ces liaisons sont de nature rotoïde **R** ou prismatiques **P**. Les corps et les liaisons du robot manipulateur sont numérotés de *0* à n dans un ordre croissant en partant du socle (Figure 3.1)

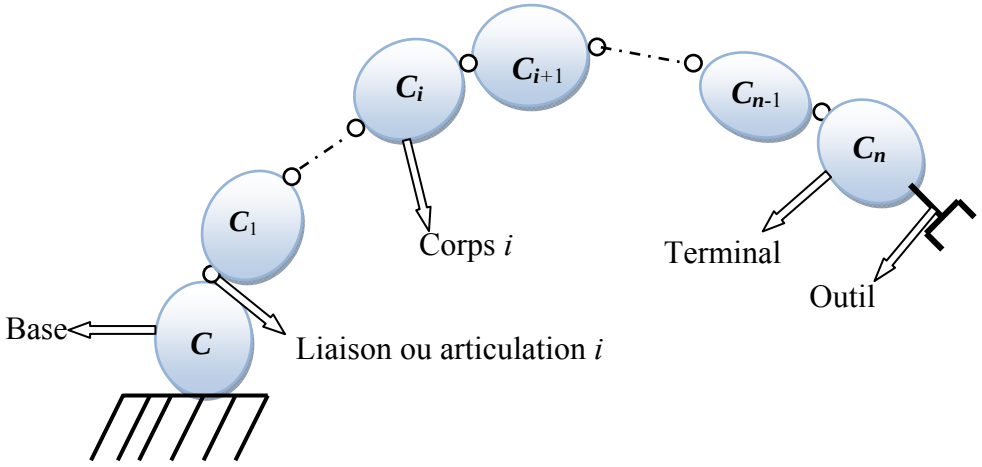


Figure 3.1. Robot à chaîne ouverte simple

2.2 Modélisation dynamique des robots manipulateurs

Le robot manipulateur, quelle que soit la fonction qui lui est attribuée, doit positionner et orienter son organe terminal dans l'espace opérationnel. Ce qui nécessite une modélisation de ce dernier, une modélisation qui consiste à représenter le comportement du robot par des équations algébriques, soit du point de vue des positions, c'est le modèle géométrique, soit du point de vue des vitesses, c'est le modèle cinématique, soit encore en considérant les efforts mis en jeu, c'est le modèle dynamique (KHALIL, 2002).

2.3 Modèle dynamique

Le modèle dynamique exprime les couples des différents bras du robot en fonction des positions, vitesses et accélérations articulaires et les efforts exercés par l'organe terminal sur l'environnement. (LEWIS, 1993)

Il est représenté par la relation suivante :

$$\Gamma = f(q, \dot{q}, \ddot{q}, F_t) \tag{3.1}$$

où : Γ= (n x 1) vecteur des couples/forces des actionneurs selon que l'articulation est rotoïde ou prismatique.

$\boldsymbol{q}$: (n x 1) vecteur des positions articulaires.

$\dot{q}$: (n x 1) vecteur des vitesses articulaires

$\ddot{q}$: (n x 1) vecteur des accélérations articulaires.

F_t : Effort extérieur (forces et couples) à exercer par l'organe terminal.

La relation (3.3) est appelée modèle dynamique inverse ou tout simplement le modèle dynamique. Le modèle dynamique direct utilisé dans la stimulation est celui qui exprime les accélérations en fonction des positions, vitesses et couples des actionnaires. Il est alors représenté par la relation suivante :

$$\ddot{q} = f(q, \dot{q}, \Gamma, F_t) \tag{3.2}$$

Plusieurs formalismes ont été utilisés pour obtenir le modèle dynamique des robots (DOMBRE, 1988), (COIFFET.1992). Une forme générale du modèle dynamique est donnée par :

$$\Gamma = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) \tag{3.3}$$

où: $M(q)$: Matrice d'inertie du robot de dimension (n×n)

$C(q, \dot{q})\dot{q}$: Matrice des termes de Coriolis et centrifuges de dimension (n×n)

$G(q)$: Vecteur de couple dus aux forces de gravitation de dimension n

F(n x 1) Vecteur des frottements

En réalité, les différents constituants du robot sont toujours affectés pour des frottements et des perturbations. Par conséquent, le modèle dynamique du robot manipulateur à chaîne ouverte simple peut être généralisé sous la forme suivante :

$$\Gamma = M(q)\dot{q} + C(q, \dot{q})\dot{q} + G(q) + F(\dot{q}) + \Gamma_d \tag{3.4}$$

avec

$$F(\ddot{q}) = F_v \dot{q} + F_s(\dot{q})q \tag{3.5}$$

Où F_v désigne la matrice des coefficients de frottements visqueux et F_s désigne la matrice des coefficients de frottements secs. Le terme Γ_d ajouté, est le vecteur des couples dû aux perturbations extérieures. (LEWIS, 1993)

3. Commande adaptative des robots manipulateurs

Dans le cadre de la commande des robots manipulateurs, nous avons motivé l'usage de la loi de commande adaptative pour contrôler ces systèmes non linéaire incertains. La dynamique du robot manipulateur donnée par :

$$\Gamma = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) \tag{3.6}$$

En réalité, nous n'avons jamais la connaissance exacte du modèle du robot en raison à des nombreux problèmes associés à la formulation du modèle (variations dans les masse, les frottements les bruits de mesures…). La solution consiste à utiliser le contrôleur adaptatif couple calcule avec une estimation des paramètres inconnus au lieu des paramètres réels. Ce contrôleur approximatif couple calculée aurait la forme (LEWIS, 1993):

$$\Gamma = \hat{M}(q)(\ddot{q}_d + K_v \dot{e} + K_p e) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}(\dot{q}) \tag{3.7}$$

où l'indice « $\hat{}$ » désigne la dynamique estimée, ou les paramètres réels sont remplacés par leur estimés. k_p et k_v sont les matrices de gain, q_d sont utilisés pour désigner la trajectoire désirée, et l'erreur de poursuite est définie par :

$$e = q_d - q \tag{3.8}$$

3.1 Loi de commande adaptive Couple calculé (Computed-Torque)

La commande couple calculé (LEWIS, 1993) repose sur la compensation des non-linéarités du robot manipulateur. Parmi les lois de commande classiques des robots manipulateurs, la commande du couple calculé est la meilleure en termes de résultats. Des procédures adaptatives peuvent être

employées pour compenser les caractéristiques du système non considérées dans la dynamique modélisée.

La stratégie de contrôle adaptatif peut être motivée par un raisonnement que l'on pouvait s'attendre à une meilleure performance si l'estimation du paramètre a été ajustée. Car il semble raisonnable de tenter de changer notre paramètre des estimations basées sur une règle d'adaptation qui serait en fonction du modèle.

La première stratégie de contrôle adaptative est une méthode décrite dans (CRAIG, 1985). Le contrôleur adaptatif couple calculé est utilise pour l'estimation des paramètres. Cette adaptation est basée sur le fait que les paramètres apparaissent linéairement dans le modèle de robot, (Figure 3.2).

La dynamique du robot (3.4) peut être écrite sous la forme :

$$W(q,\dot{q},\ddot{q})\varphi = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) \tag{3.9}$$

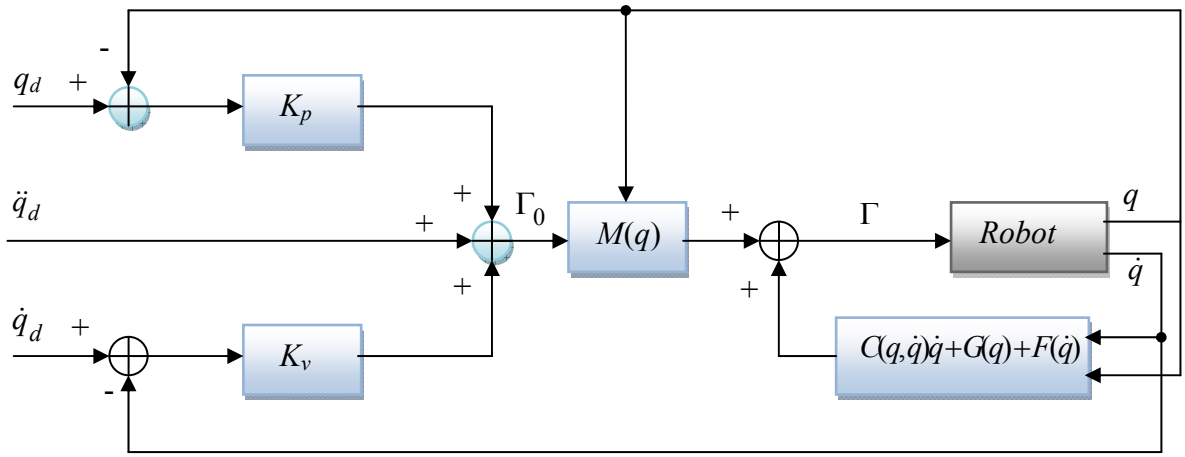


Figure 3.2 : Schéma du contrôleur adaptatif couple calculé.

où $W(q,\dot{q},\ddot{q})\varphi$ est une matrice (n×r) et φ est un vecteur (r×1) de paramètres inconnus. Cette propriété est cruciale pour le type de la commande adaptative que Craig a formulée en ce sens, qu'elle illustre la séparation de paramètres inconnus et les fonctions de temps connu. La raison pour laquelle la dynamique du robot peut être séparé sous cette forme est que la dynamique du robot est linéaire dont les paramètres sont exprimés sous la forme de vecteurs. Cette séparation des paramètres inconnus et les fonctions des temps connus sera utilisée dans la formulation de la commande adaptation

et aussi dans l'analyse de la stabilité de ce dernier. Notez qu'en utilisant (3.9), nous pouvons écrire la dynamique du robot (3.8) par :

$$\Gamma = W(q,\dot{q},\ddot{q})\varphi \tag{3.10}$$

De (CRAIG, 1985), la loi de commande adaptative couple calculé est donnée par :

$$\Gamma = \hat{M}(q)(\ddot{q}_d + K_v\dot{e} + K_p e) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}(\dot{q}) \tag{3.11}$$

Il est facile de voir à partir de notre définition de l'erreur de suivi comment (3.7) peut être écrit comme

$$\Gamma = \hat{M}(q)(\ddot{e} + K_v\dot{e} + K_p e) + \hat{M}(q)\ddot{q} + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}(\dot{q}) \tag{3.12}$$

En utilisant (3.9), (3.11) peut être écrite comme

$$\Gamma = \hat{M}(q)(\ddot{e} + K_v\dot{e} + K_p e) + W(q,\dot{q},\ddot{q})\tilde{\varphi} \tag{3.13}$$

où $\hat{\varphi}$ est un vecteur (n × 1) utilisé pour représenter une estimation variante dans le temps des paramètres inconnu constants. En substituant (3.13) dans (3.10), nous pouvons obtenir l'erreur du système

$$\ddot{e} + K_v\dot{e} + K_p e = \hat{M}^{-1}(q)W(q,\dot{q},\ddot{q})\tilde{\varphi} \tag{3.14}$$

où le paramètre d'erreur est

$$\tilde{\varphi} = \varphi - \hat{\varphi} \tag{3.15}$$

En réécrivant (3.14) sous forme d'espace d'état

$$\dot{e} = Ae + BM^{-1}(q)W(q,\dot{q},\ddot{q})\tilde{\varphi} \tag{3.16}$$

où le vecteur d'erreur de suivi est

$$e = \begin{bmatrix} e \\ \vdots \\ e \end{bmatrix} \quad \text{et} \quad B = \begin{bmatrix} O_n \\ I_n \end{bmatrix}, \quad A = \begin{bmatrix} O_n & I_n \\ -K_p & -K_v \end{bmatrix} \tag{3.17}$$

avec I_n matrice d'identité (n×n), et O_n la matrice nul (n×n).

4. Système d'inférence floue séquentielle adaptive (SAFIS)

Récemment, l'équivalence fonctionnelle entre un réseau de neurones à fonctions de bases radiales (RBF) et un système d'inférence flou a été exploitée par de nombreux chercheurs (LEE.2003), (CHO.1996), (JUANG.2002), (WANG.2000), (LENG.2005). Le SAFIS est un système d'apprentissage séquentiel qui utilise les capacités d'apprentissage du réseau RBF pour l'ajustement des paramètres du réseau qui sont liées à des règles floues. Les données arrivent un par un, après l'apprentissage de chacune des données, ils sont écarté et la notion d'itération n'existe pas, donc l'apprentissage ne nécessite pas de recyclage chaque fois qu'une nouvelle donnée reçue. L'algorithme SAFIS, à les caractéristiques distinctives suivantes (HUANG,2005) :

1. Toutes les opérations d'apprentissage sont séquentiellement (un par un) présentés au système.
2. À tout moment, une seule opération d'apprentissage est vue et appris
3. Une opération d'apprentissage est écartée dès que la procédure d'apprentissage est terminée.
4. Le système d'apprentissage n'a pas de connaissances préalables sur le nombre total des opérations d'apprentissage présentées.

Le SAFIS est développé pour réaliser un système compact avec moins de règles floues. Il utilise le mécanisme RBF-GAP proposée par Huang (HUANG, 2004). L'algorithme SAFIS se compose de :

- la détermination des règles floues
- l'ajustement des paramètres de la prémisse et les paramètres conséquent des règles floues.

SAFIS utilise la notion l'influence d'une règle floue pour ajouter et supprimer les règles lors de l'apprentissage. Il commence sans règles floues. Au cours de l'apprentissage, les données actuelles sont utilisées et il n'est pas

nécessaire de stocker toutes les données du passé. L'influence d'une règle floue est définie selon sa contribution à la sortie du système. Alors que l'ajustement des paramètres se fait en utilisant le mécanisme Extended Kalman filter (EKF). Il convient de noter que SAFIS est un algorithme vraiment à apprentissage séquentiel.

4.1 Réseaux de neurones RBF

Un réseau à fonctions de bases radiales (RBF) est un réseau neurologique artificiel qui emploie des fonctions radiales de base comme fonctions d'activation. C'est une combinaison linéaire des fonctions radiales de base. Elles sont employées dans l'approximation de fonction, la prévision de série chronologique, et la commande.

Le réseau RBF (Radial Basis Function) est basé sur une architecture qui s'organise en deux couches seulement ; une couche cachée et une couche de sortie comme le montre la figure 3.3 tel que :

La couche cachée : constituée des noyaux (ou neurones) RBF effectue une transformation non linéaire de l'espace d'entrée.

La couche de sortie : calcule une combinaison linéaire des sorties de la couche cachée. Chaque noyau élémentaire calcule la distance entre l'entrée et son centre qu'il passe ensuite dans une non linéarité concrétisée par une fonction d'activation (.) qui est généralement de type gaussienne.

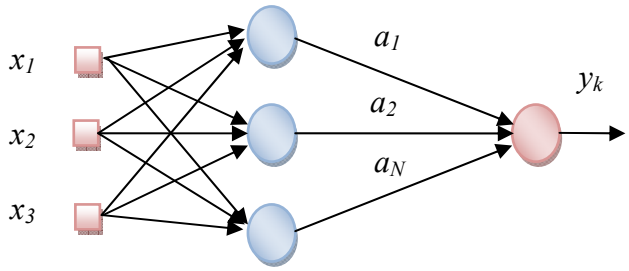


Figure 3.3 : Architecture du réseau RBF

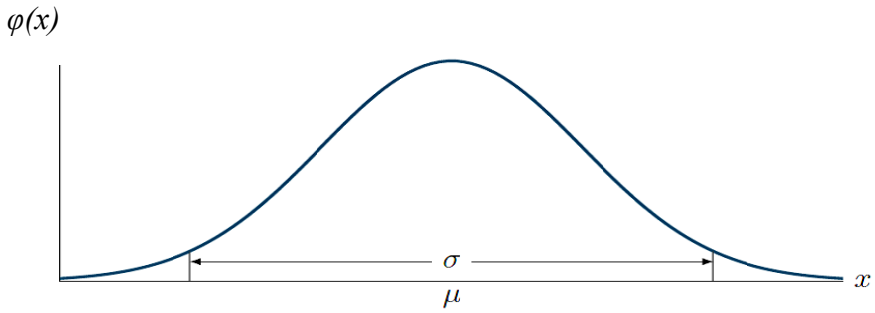


Figure 3.4 : Fonction gaussienne du réseau RBF.

La valeur que prend la sortie du noyau gaussien est d'autant plus importante que l'entrée est plus proche de son centre et tend vers zéro, lorsque la distance entrée centre devient importante. La sortie du réseau RBF est donnée par (HAGAN, 1995) :

$$\hat{y}_k(x) = \sum_{j=1}^{N} a_{kj}\varphi_j(x) + a_{k0} \tag{3.18}$$

Où sa fonction gaussienne est donnée par la relation suivante :

$$\varphi_j(x) = \exp\left[\frac{-\|x_i - \mu_j\|^2}{2\sigma_j^2}\right] \tag{3.19}$$

Avec $\|.\|$ dénotant la norme euclidienne, x est le vecteur d'entrée des éléments x_i, et μ_j : le vecteur déterminant les centres de la fonction φ_j par rapport aux entrées et σ est sa largeur (Figure 3.4) ; a_{k0} est le poids d'un neurone dont l'entrée est toujours 1, appelé le biais (HAGAN, 1995).

4.1.1 Apprentissage des réseaux RBF

L'apprentissage d'un réseau RBF consiste à déterminer son architecture (le nombre N de fonctions radiales) et à fixer les valeurs des paramètres. La plupart des utilisateurs déterminent empiriquement la valeur de N en recourant à des techniques de validation croisée.

L'apprentissage d'un réseau RBF est de type supervisé : on dispose d'un ensemble d'apprentissage constitué de couples (vecteur d'entrée, valeur cible):

$$(x_1;y_1);...;(x_m;y_m) \quad x_i \in R^d; y_i \in R \tag{3.20}$$

et du coût associé à chaque exemple :

$$J_i = \frac{1}{2}(y_i - F(x_i))^2 \tag{3.21}$$

L'objectif de l'apprentissage est de minimiser une quantité J, donnée par l'équation (3.21). Une caractéristique intéressante des modèles RBF est que l'on peut diviser les paramètres en trois groupes : les centres, les largeurs et les poids *w*. L'interprétation de chaque groupe permet de proposer un algorithme d'apprentissage séquentiel, simple et performant (YINGWEI, 1998).

4.1.2 Approche séquentielle

Cette technique d'apprentissage proposée dès la fin des années 1980 (HAGAN, 1995) est très couramment utilisée. Elle consiste à optimiser successivement les trois jeux de paramètres (μ ; σ ; w). Cette technique a l'avantage d'être simple à mettre en œuvre, de demander peu de calculs et de donner des résultats acceptables. La solution obtenue n'est cependant pas optimale.

Dans un premier temps, on estime les positions des centres μ et des largeurs à l'aide d'un algorithme non supervisé de type k-moyennes. Une fois ces paramètres fixés, il est possible de calculer les poids w_j optimaux par une méthode de régression linéaire. C'est certainement la simplicité et l'efficacité de cette méthode qui a fait le succès des RBF (MOODY, 1995).

$$\hat{y}(x) = \sum_{j=0}^{N} a_{kj}\varphi_j(\|x - \mu_j\|, \sigma_j) = \sum_{j=0}^{N} a_j h_j(x) \tag{3.22}$$

4.1.3 Rétro-propagation du gradient de l'erreur

La rétro-propagation du gradient de l'erreur est utilisée pour ajuster les poids et les biais du réseau afin de minimiser l'erreur quadratique entre la sortie du réseau et la sortie réelle (WIDROW, 1990).

A chaque couple entrée/sortie, une erreur est calculée, le gradient, ou pente de l'erreur est déterminé. Ensuite les poids et les biais sont modifiés sur le réseau. Nous réitérons ces calculs jusqu'à l'obtention du critère d'arrêt. Une alternative à l'apprentissage séquentiel décrite dans la section précédente consiste à optimiser les paramètres du modèles RBF par descente de gradient,

comme nous le faisons pour d'autres modèles connexionnistes. Il faut pour cela calculer les dérivées du coût par rapport aux différents paramètres. L'erreur d'un seul couple d'entrée/sortie

$$E_i = \frac{1}{2}(y_{di} - \hat{y}(x_i))^2 \tag{3.23}$$

4.2 Architecture du réseau SAFIS

Généralement, une large classe des systèmes dynamiques non linéaires peut être représentée par le modèle non linéaire discret par la description des entrées-sorties :

$$y(n) = f[y(n-1), y(n-2),..., y(n-k+1); u(n), u(n-1),..., u(n-p+1)] \tag{3.25}$$

où y est un vecteur contenant N_y sorties du système, u est un vecteur de N_u entrées du système ; f est une fonction non linéaire, et k et p sont le maximum des retards de l'entrée et de sortie respectivement. Les x_n;y_n désignent respectivement l'entrées et les sorties du système flou. L'équation ci-dessus peut être mise sous forme :

$$y_n = f(x_n) \tag{3.26}$$

L'objectif du nouveau algorithme SAFIS est de rapprocher f telle que

$$\hat{y}_n = \hat{f}(x_n) \tag{3.27}$$

où $\hat{y}_n$ est la sortie de SAFIS. L'objectif est de minimiser l'erreur entre la sortie du système et la sortie de SAFIS, $\| y_n - \hat{y}_n \|$

La structure de SAFIS illustrée par (Figure 3.5) est constituée de cinq couches pour réaliser des règles floues de la forme :

$$la \text{ règle } k : si\ (x_1\ \ est\ \ A_{1k})...(x_{Nx}\ \ est\ A_{N_x k}), alors\ (\hat{y}_1\ est\ a_{1k})...(\hat{y}_{Ny}\ \ est\ a_{Nyk})$$

où a_{jk} (j = 1, 2,.., N_y, k =1 , 2,.., N_h) est un paramètre conséquent constant dans la règle k, A_{ik} (i = 1, 2,..., N_x) est la valeur de la fonction d'appartenance de la $i^{ème}$ variable d'entrée x_i dans la règle k,N_x est la dimension du vecteur d'entré x (x=[x_1,...,x_{Nx}]^T), N_h est le nombre des règles floues, N_y est la dimension du

vecteur de sortie $\hat{y}$ ($\hat{y}=[\hat{y}_1,.....,\hat{y}_{Ny}]^T$). En SAFIS, le nombre de règles floues N_h varie. Au départ, il n'y a pas de règles floues, puis lors de l'apprentissage les règles floues sont ajoutées ou supprimées. (HAI, ,2006)

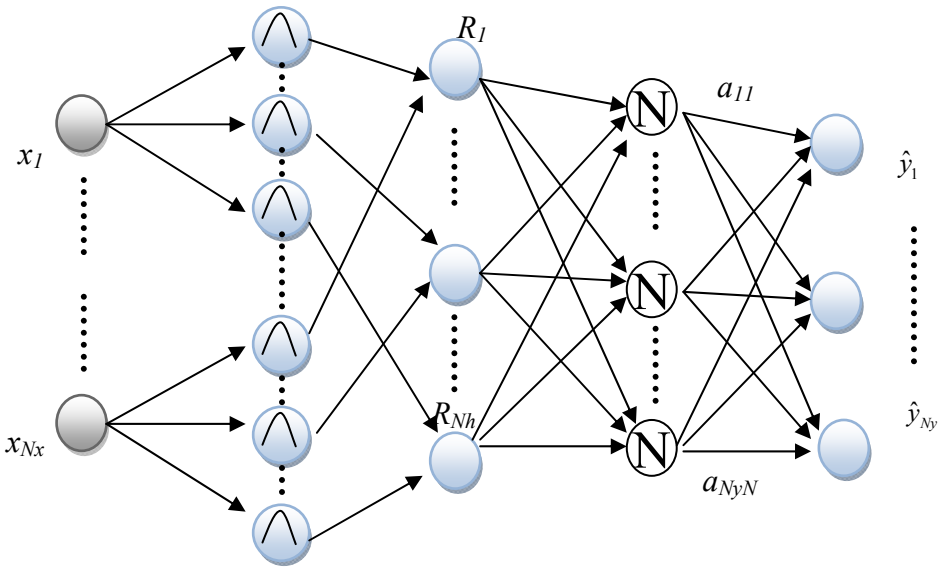


Figure 3.5 : Structure du SAFIS

Couche 1 : Dans la couche 1, chaque nœud représente une variable d'entrée qui est transmise directement à la couche 2

Couche 2 : Dans cette couche, chaque nœud représente la valeur de la fonction d'appartenance de chaque variable d'entrée qui se base sur le principe d'équivalence entre un réseau RBF et un FIS et ainsi les règles floues sont réalisées par une fonction gaussienne a base radiale.

La valeur de la fonction d'appartenance $A_{ik}(x_i)$ de la $i^{éme}$ variable d'entrée x_i dans la $k^{ième}$ fonction gaussienne est donnée par :

$$A_{ik}(x_i) = \exp(-\frac{(x_i - \mu_{ik})^2}{\sigma_k^{\,2}}), k = 1,2,..., Nh \qquad (3.28)$$

où N_h est le nombre de fonctions gaussiennes, μ_{ik} est le centre de la $k^{ième}$ fonction gaussien pour l'entrée x_i, σ_k est la largeur de la $k^{ième}$ fonction gaussien.

Couche 3 : Chaque nœud dans la couche représente la partie si-alors de la règle obtenue, et le nombre total de ces règles est Nh . La valeur de la partie Si de la k^{ième} règle est donnée par :

$$R_k(x) = \prod_{i=1}^{Nx} A_{ik}(x_i) = \exp(-\sum_{i=1}^{Nx} \frac{(x_i - \mu_{ik})^2}{\sigma_k^{\,2}}) = \exp(-\frac{\| x_i - \mu_{ik} \|^2}{\sigma_k^{\,2}}) \qquad 3.29)$$

Couche 4 : Les nœuds dans cette couche sont nommés normalisés dont le nombre est égal au nombre des nœuds dans la troisième couche. Le $k^{ième}$ nœud normalisée est donnée par :

$$\overline{R}_k = \frac{R_k(x)}{\sum_{k=1}^{Nh} R_k(x)} \qquad (3.30)$$

Couche 5 : Chaque nœud de cette couche correspond à une variable de sortie, qui est donnée par la somme pondérée de la production de chaque règle normalisée. La sortie du système est calculée par :

$$\hat{y} = \frac{\sum_{k=1}^{Nh} R_k(x) a_k}{\sum_{k=1}^{Nh} R_k(x)} \qquad (3.31)$$

avec, $\hat{y} = [\hat{y}_1, ..., \hat{y}_{Ny}]^T, a_k = [a_{1k}, a_{2k}, ... a_{Nyk}]^{\mathrm{T}}$

4.3 Opération effectuées par l'architecture SAFIS

4.3.1 Influence d'une règle floue

De l'équation (3.31), la contribution de la $k^{ième}$ règle dans la sortie globale pour l'entrée x_l est donnée par :

$$E(k,l) = |a_k| \frac{R_k(x_l)}{\sum_{k=1}^{Nh} R_k(x_l)} \qquad (3.32)$$

Ensuite, la contribution de la $k^{ième}$ règle à la sortie basée sur les N données d'entrée reçues à ce est obtenue par :

$$E(k) = |a_k| \frac{\sum_{l=1}^{N} R_k(x_l)}{\sum_{k=1}^{Nh} \sum_{l=1}^{N} R_k(x_l)} \tag{3.33}$$

En divisant le numérateur et le dénominateur par N dans l'équation. (3. 33), l'équation devient :

$$E(k) = |a_k| \frac{\sum_{l=1}^{N} R_K(x_l)/N}{\sum_{k=1}^{N_h} \sum_{l=1}^{N} R_K(x_l)/N} \tag{3.34}$$

En utilisant le concept de GAP-RBF (HUANG, 2005), l'influence de la $k^{ième}$ règle floue est définie comme une contribution statistique à la sortie globale de SAFIS, l'influence de la $k^{ième}$ règle est donnée par :

$$E_{\inf}(k) = |a_k| \frac{(1.8\sigma_k)^{N_x}}{\sum_{k=1}^{N_h} (1.8\sigma_k)^{N_x}} \tag{3.35}$$

4.3.2 Ajout d'une règle

SAFIS commence sans règles floues. Comme des entrées x_n; y_n, (n est l'indice de temps) sont reçus successivement au cours de l'apprentissage des règles sont ajoutées en se basant sur deux critères suivants, qui sont le critère de distance et de l'influence de la (N_h +1): nouvelle règle floue a ajouter

$$\begin{cases} \|x_n - \mu_{nr}\| > \varepsilon_n \\ E_{\inf}(N_h + 1) = |e_n| \dfrac{(1.8K\|x_n - \mu_{nr}\|)^{N_x}}{\sum_{k=1}^{N_{h+1}} (1.8\sigma_k)^{N_x}} > e_g \end{cases} \tag{3.36}$$

où ε_n, e_g sont des seuils choisis de façon appropriée, x_n est les dernières données d'entrée, u_{nr} est le centre de la règle floue la plus proche de x_n , e_g est le seuil de croissance choisi en fonction de la précision souhaitée de SAFIS. L'erreur est donnée par

$$e_n = y_n - \hat{y}_n \tag{3.37}$$

avec y_n est la valeur de sortie, $\hat{y}_n$ est la valeur approximative, n est le seuil de distance qui décroît de façon exponentielle est donnée par :

$$\varepsilon_n = \max\{\varepsilon_{\max} \times \gamma^n \varepsilon_{\min}\} \tag{3.38}$$

où ε_{max},, ε_{min} , γ sont la des constantes a choisir .

Si les conditions de (3.36) sont remplies la nouvelle N_h *+1* règle floue est ajoutée, son prédécesseur et les paramètres correspondants sont attribués comme suit :

$$\begin{cases} a_{Nh+1} = e_n \\ \mu_{Nh+1} = x_n \\ \sigma_{Nh+1} = k\|x_n - \mu_{nr}\| \end{cases} \tag{3.39}$$

k est un facteur de chevauchement qui détermine le chevauchement de règles floues dans l'espace d'entrée.

4.3.3 Ajustement des paramètres du réseau SAFIS

Dans le cas de l'insatisfaction des critères d'ajout, nous procéderons à un ajustement des paramètres par les réseaux SAFIS par une stratégie comme dans (HUANG, 2004). L'idée de cette stratégie consiste à ajuster seulement les paramètres relatifs à la règle gagnante sélectionnée par le moyen d'un filtre de Kalman étendu à chaque itération. La règle gagnante est définie comme la règle la ***plus proche*** (au sens de la distance euclidienne) des entrées courantes.

Le vecteur de paramètre de toutes les règles floues est donnée par

$$\theta = \left[\theta_1 \ldots \theta_{nr} \ldots \theta_{Nh}\right]^T = \left[a_1, \mu_1, \sigma_1, \ldots a_{nr}, \mu_{nr}, \sigma_{nr}, \ldots a_{Nh}, \mu_{Nh}, \sigma_{Nh}\right]^T \tag{3.40}$$

où θ_{nr}=[a_{nr},u_{nr},σ_{nr}] est le vecteur des paramètres de la règle floue la plus proche et son gradient est calculé comme suit :

$$\dot{a}_{nr} = \frac{\partial \hat{y}_n}{\partial a_{nr}} = \frac{\partial \hat{y}_n}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial a_{nr}} = \frac{R_{nr}}{\sum_{k=1}^{N_h} R_k}, \dot{\mu}_{nr} = \frac{\partial \hat{y}_n}{\partial \mu_{nr}} = \frac{\partial \hat{y}_n}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial \sigma_{nr}} = \frac{a_{nr} - \hat{y}_n}{\sum_{k=1}^{N_h} R_k} \frac{\partial R_{nr}}{\partial \mu_{nr}},$$

$$\dot{\sigma}_{nr} = \frac{\partial \hat{y}_n}{\partial \sigma_{nr}} = \frac{\partial \hat{y}_n}{\partial R_{nr}} \frac{\partial R_{nr}}{\partial \sigma_{nr}} = \frac{a_{nr} - \hat{y}_n}{\sum_{k=1}^{N_h} R_k} \frac{\partial R_{nr}}{\partial \sigma_{nr}}, \frac{\partial R_{nr}}{\partial \mu_{nr}} = 2R_{nr} \frac{X_n - \mu_{nr}}{\sigma_{nr}^2},$$

$$\frac{\partial R_{nr}}{\partial \sigma_{nr}} = 2R_{nr} \frac{\|X_n - \mu_{nr}\|^2}{\sigma_{nr}^3} \tag{3.41}$$

$$\begin{aligned} K_n &= P_{n-1}B_n\left[R_n + B_n^T P_{n-1}B_n\right]^{-1}, \\ \theta_n &= \theta_{n-1} + K_{n\ell n}, \\ P_n &= \left[I_{Z\times Zn} - K_n B_n^T\right]P_{n-1} + qI_{Z\times Z}, \end{aligned} \tag{3.42}$$

Après avoir obtenu le vecteur du gradient $B_{nr} = \left[\dot{a}_{nr}, \dot{\mu}_{nr}, \dot{\sigma}_{nr}\right]$ de la règle floue la plus proche l'algorithme EKF est utilisé pour mettre à jour les paramètres comme dans (3.42).

où q est un scalaire qui détermine le pas dans la direction du vecteur gradient, z est la dimension des paramètres à ajuster. Quand une nouvelle règle est ajoutée, la dimension de P_n est augmentée :

$$\begin{pmatrix} P_{n-1} & 0 \\ 0 & p_0 I_{Z1\times Z1} \end{pmatrix} \tag{3.43}$$

Z_1 est la dimension des paramètres introduits par la nouvelle règle a ajouté, P_0 est une valeur initiale de l'incertitude affectée à cette règle nouvellement allouée.

4.3.4 Suppression d'une règle floue

Si l'influence de la règle $k^{ième}$ est inférieure au seuil e_p (3.44), alors la règle est insignifiante à la sortie qui doit être retirée. Le seuil de suppression e_p est choisi a priori.

$$E_{\inf}(K) = |ak|\frac{\left(1.8_{\sigma k}\right)^{N_x}}{\sum_{k=1}^{N_h}\left(1.8_{\sigma k}\right)^{N_x}} \langle e_p \tag{3.44}$$

4.4 Algorithme SAFIS

L'approche SAFIS se déroule comme il est illustré dans la Figure 3.6 qui représente un organigramme de l'algorithme SAFIS :

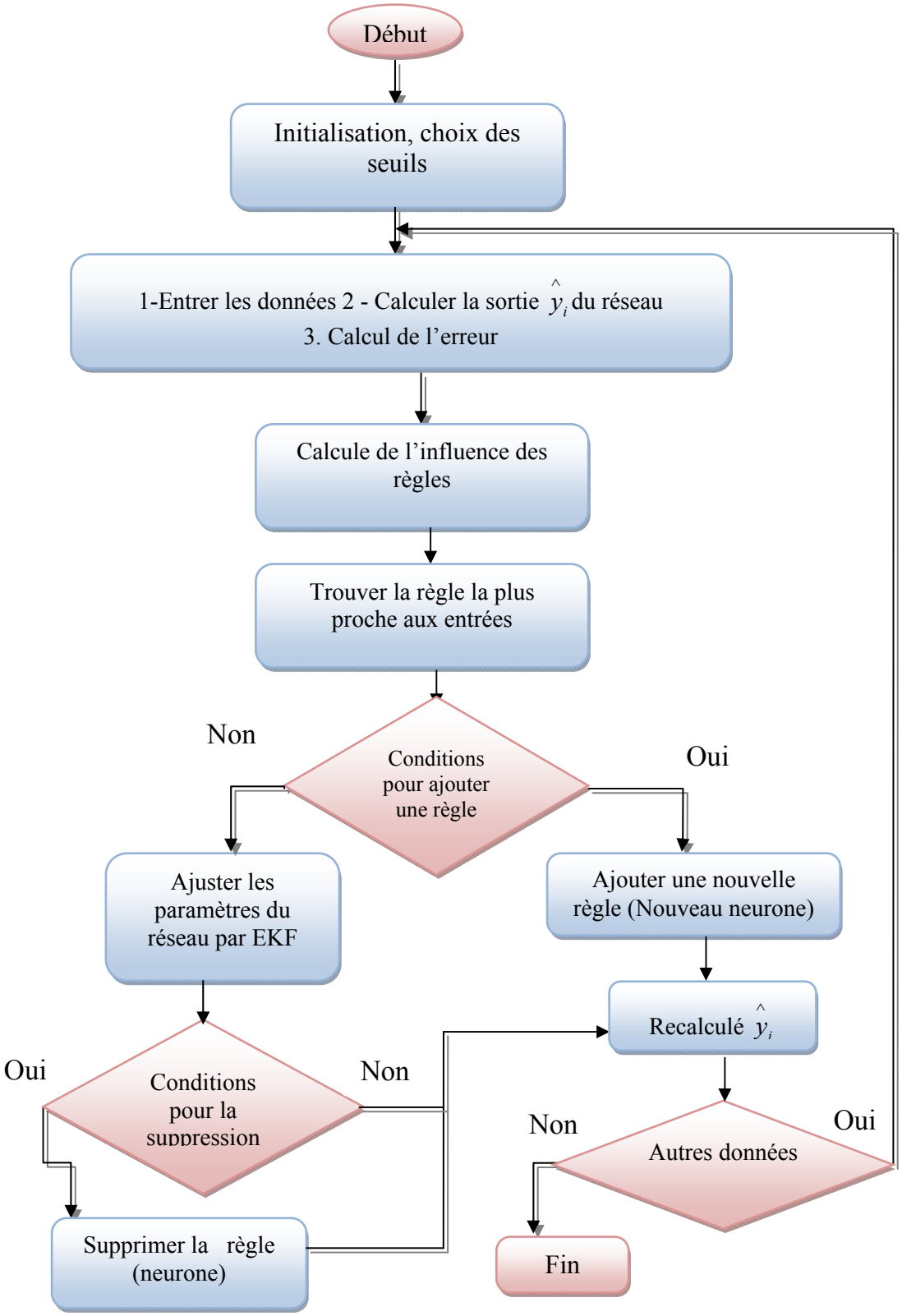


Figure 3.6 : Organigramme du l'algorithme **SAFIS**

5. Application de la commande adaptative basée SAFIS pour un robot manipulateurs

La commande adaptative couple calculée (Figure 3.2), utilise la technique du retour linéarisant, qui consiste à compenser les non-linéarités de sorte que le système dynamique ait un comportement linéaire en boucle fermé. La difficulté pour mettre en œuvre un contrôleur de couples calculés consiste en l'estimation des paramètres du modèle. La loi de commande couple calculé est donné par :

$$\Gamma = \hat{M}(q)\Gamma_0 + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}(\dot{q}) \tag{3.47}$$

où : $\hat{M}$ est la matrice de sortie de l'estimateur de la matrice d'inertie M, $\hat{C}$ est la matrice de sortie de l'estimateur de la matrice de Coriolis, $\hat{G}$ est le vecteur de sortie de l'estimateur du vecteur de gravite. $\hat{F}$ est le vecteur de sortie de l'estimateur du vecteur de frottements. (SAHRAOUI, 2015).

L'estimation des paramètres du robot manipulateur est effectuée à l'aide des réseaux neuro-flous basés sur l'approche SAFIS. La complexité du robot manipulateur et son fort non linéarité rendent l'utilisation d'un seul réseau très difficile à réaliser. Dreyfus (DREYFUS, 2002) privilège la prise en compte des connaissances à priori du procédé à identifier lors de la modélisation.

Donc, nous proposons d'utiliser quatre réseaux neuro-flous pour estimer les paramètres du robot manipulateur. Le premier réseau pour estimer la matrice d'inertie $M(q)$, le deuxième réseau sert à estimer les termes la matrice des Coriolis et forces centrifuges $C(q,q)$ et le troisième sert à estimer le vecteur de frottement $f(q)$ alors le quatrième pour estimer le vecteur de gravité. (SAHRAOUI, 2014). Nous pouvons écrire :

$$\begin{aligned} \hat{M} &= f(q) \\ \hat{C} &= g(q,\dot{q}) \\ \hat{G} &= j(q) \\ \hat{F} &= h(\dot{q}) \end{aligned} \tag{3.48}$$

avec f,g,h,j des réseaux SAFIS. Pour pouvoir obtenir de bonnes performances, nous devons avoir :

$$\begin{aligned} \hat{M} - M &= \varepsilon m \cong 0 \\ \hat{C} - C &= \varepsilon c \cong 0 \\ \hat{G} - G &= \varepsilon g \cong 0 \\ \hat{F} - F &= \varepsilon f \cong 0 \end{aligned} \tag{3.49}$$

Ces conditions sont satisfaisantes à travers un bon choix des paramètres de l'application (seuils).

5.1. Résultats des simulations de la Commande d'un robot SCARA sans bruit

Pour valider l'approche utilisée nous avons effectué des simulations pour la commande adaptative d'un robot manipulateur de type SCARA à 3 degré de liberté, dont les paramètres du modèle sont estimés par quatre réseaux SAFIS.

Une étape d'initialisation de paramètres table 3 (seuils, erreur, facteur de chevauchement..) est nécessaire pour l'accomplissement de l'apprentissage. La détermination de l'architecture du réseau utilisé (nombre de neurone) est faite en appliquant une technique (GROWING AND PRUNING GAP) qui consiste à considérer un réseau sans architecture préalable (nombre de neurones nul) ; et l'ajustement des paramètres du réseau est réalisé par un filtre de Kalman.

Nous considérons en premier lieu un environnement idéal (sans perturbation et incertitudes) et ensuit nous abordons des tests de robustesse de la commande proposer en introduisant des variations dans les paramètres de la matrice d'inertie, et de bruits. (Figure 3.7) montre la structure de la commande adaptative utilisée dans notre application.

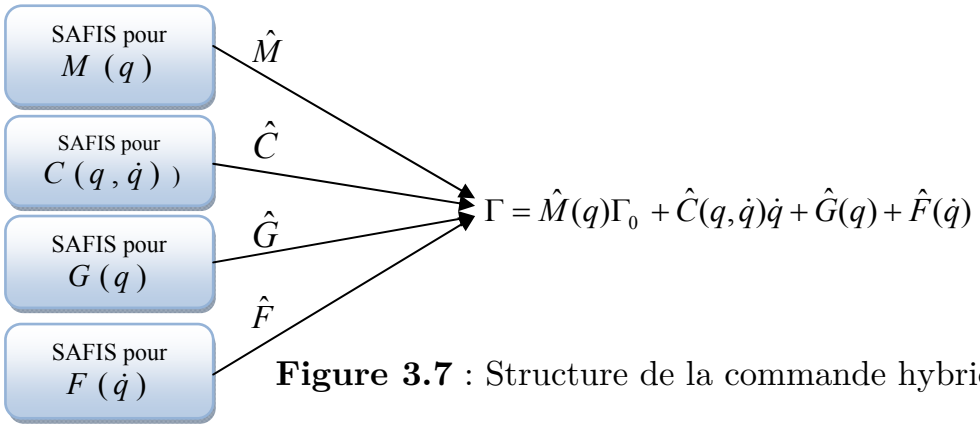


Figure 3.7 : Structure de la commande hybride

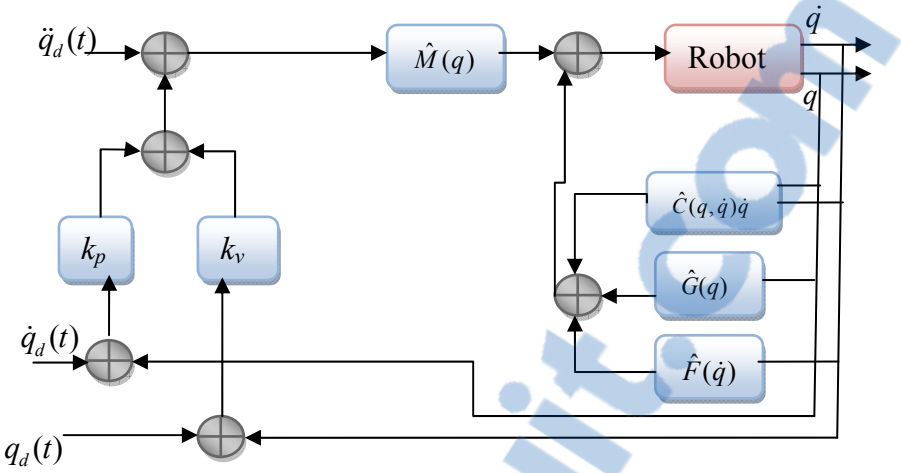


Figure 3.8 : Schéma de la commande du couple calculé

Paramètre	SAFIS-M	SAFIS-C	SAFIS-F
Entrées	q	$q, \dot{q}$	$\dot{q}$
Nombre des entrées	*3*	*9*	*3*
Sorties	$\hat{M}$	$\hat{C}$	$\hat{F}$
Seuil d'ajout e_g	*2.0e-3*	*2.0e-3*	*2.0e-3*
Facteur de chevauchement	*1.0*	*1.0*	*1.0*
Seuil de suppression e_p	*2.0e-4*	*2.0e-4*	*2.0e-4*

Table 3.1 : Paramètres de l'algorithme d'apprentissage pour les 3rèseaux SAFISM, SAFISC, SAFISF

Après avoir initialisé les seuils de l'algorithme d'apprentissage des quatre réseaux SAFIS, dans notre approche les paramètres estimés du robot manipulateur SCARA sont injectés dans la commande couple calculé en utilisant le système SAFIS. (Figure 3.8) montre la structure de la commande adaptative utilisée dans notre application. Les résultats de la commande et les erreurs de l'identification des composantes du modèle sont présentées comme suit :

5.1.1 Résultats de la commande

La commande adaptative à base SAFIS a été utilisée pour commander le robot manipulateur pour un objectif de régulation. Les positions désirées et les positions calculées sont présentées dans (Figure 3.9), (Figure 3.10).

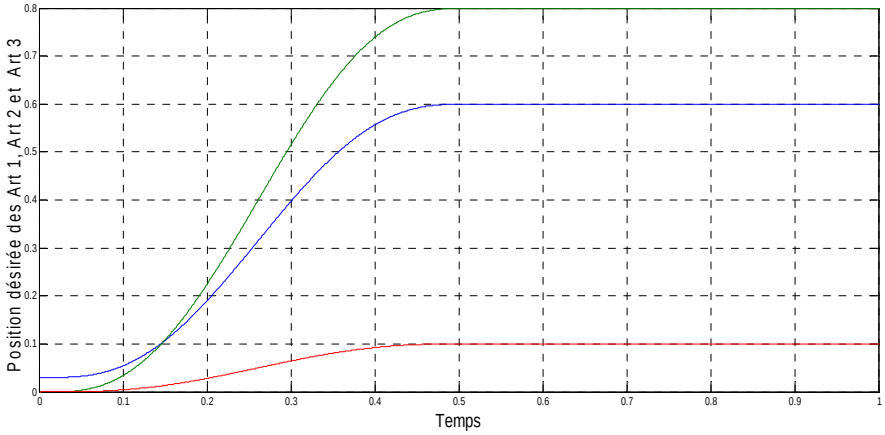


Figure 3.9. : Position désire des Art 1, Art 2 et Art3

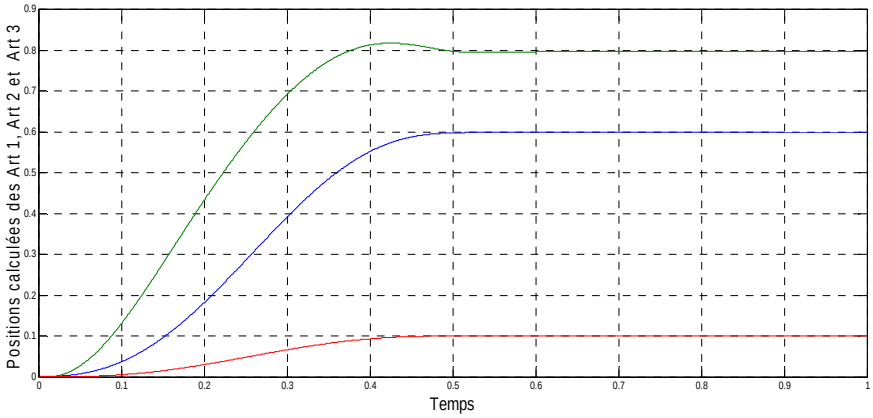


Figure3.10. : Position calculée des Art 1, Art 2 et Art3

Alors que les erreurs en positions sont données dans (Figure 3.11) et dans (Figure 3.12). L'erreur en vitesse des trois articulations présentées dans (Figure 3.13), (Figure 3.14).

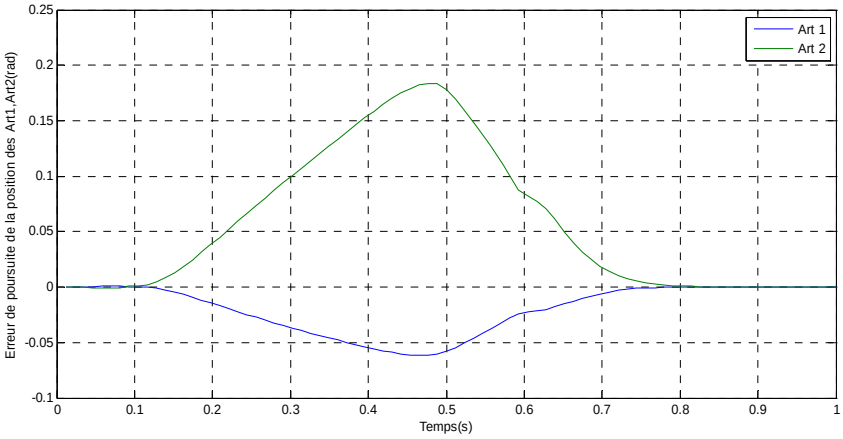


Figure 3.11 : Erreur de poursuite de la position des Art 1, Art 2

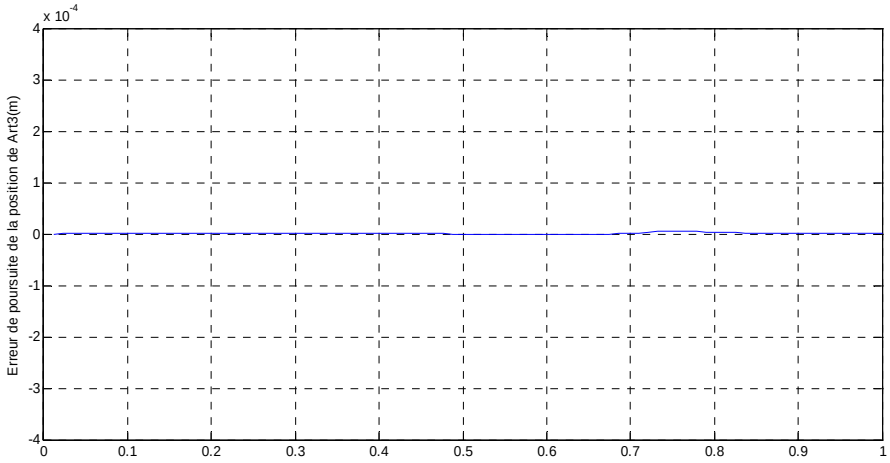


Figure 3.12 : Erreur de poursuite de la position de l'Art 3

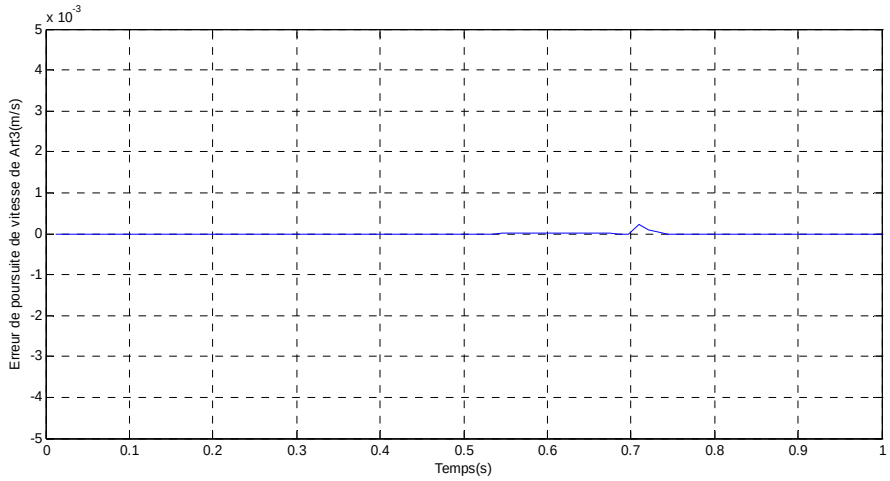


Figure 3.13 : Erreur de poursuite de la vitesse de l'articulateur Art3

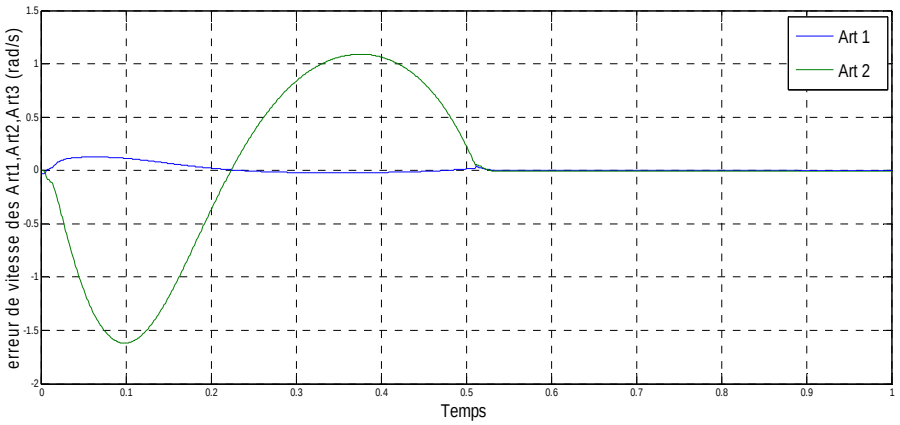


Figure 3.14 : Erreur de poursuite de la vitesse des Art 1, Art 2

Les résultats obtenus sont encourageants par la convergence de l'erreur vert zéro d'où l'utilisation massive des réseaux de SAFIS dans la commande des robots manipulateurs et très importantes.

5.1.2 Estimation de la matrice d'inertie

Nous avons procédé un apprentissage, en utilisant un réseau SAFIS. La (Figure 3.15) montre les erreurs d'identification des éléments de la matrice d'inertie en absence de bruit et perturbation sur le robot, où on remarque une convergence est acceptable de l'erreur ($\varepsilon_m \approx 0$).

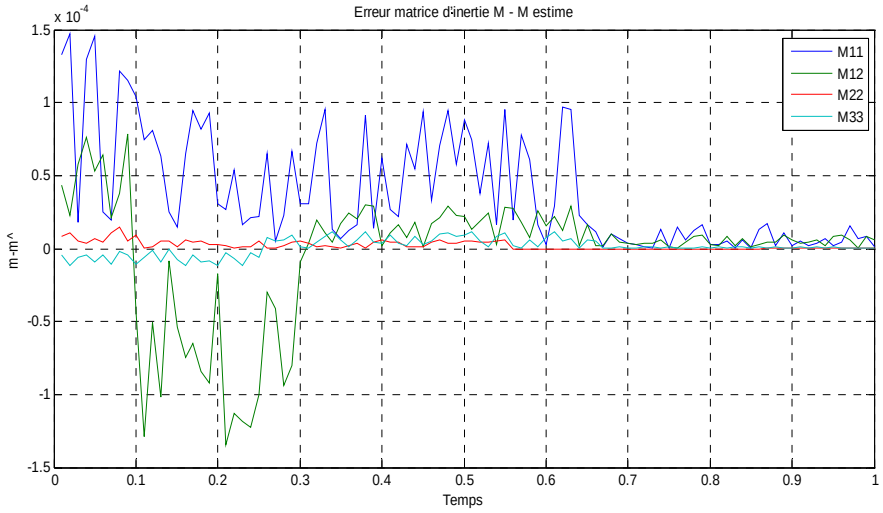


Figure 3.15 : Erreur d'identification des éléments de la matrice d'inertie

5.1.3 Estimation de la matrice Coriolis

Pour l'estimation de la matrice des termes de Coriolis et de forces centrifuges, le même principe que dans le cas de la matrice d'inertie. L'apprentissage un réseau SAFIS donne des résultats affichés dans (Figure 3.16).

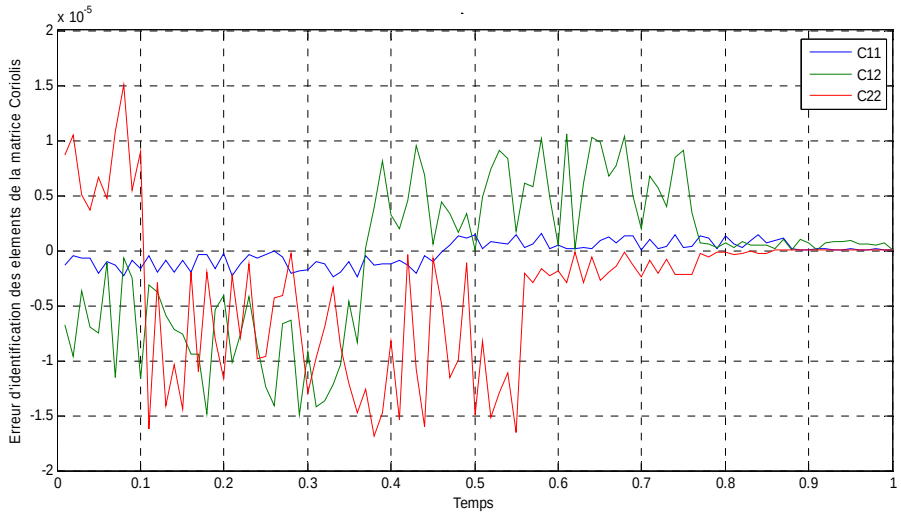


Figure 3.16 : Erreur d'identification des éléments de la matrice Coriolis

Nous constatons que l'erreur d'identification de la matrice Coriolis converge vert zéros. Elle est d'ordre 10^{-6} .

5.1.4 **Estimation du vecteur de frottement**

Les résultats de l'apprentissage du réseau SAFIS pour l'estimation du vecteur de frottement, donnés en (Figure 3.17).

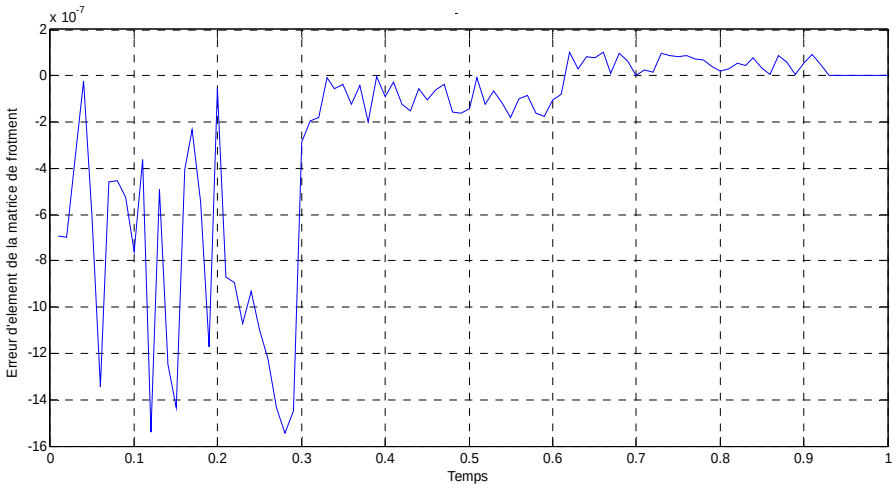


Figure 3.17 : Erreur d'identification d'élément de la matrice de froments

Nous constatons que l'erreur d'identification du vecteur converge vert zéro après une certaine durée d'apprentissage causée par la complexité des frottements.

5.2 Commande d'un robot SCARA en présence de bruit

5.2.1 Résultats de la simulation

Pour tester la robustesse de la commande proposée on a introduit volontairement un bruit blanc dans un intervalle de temps [0.4 0.6], et nous avons altéré des composantes de la matrice d'inertie. Les résultats obtenus sont :

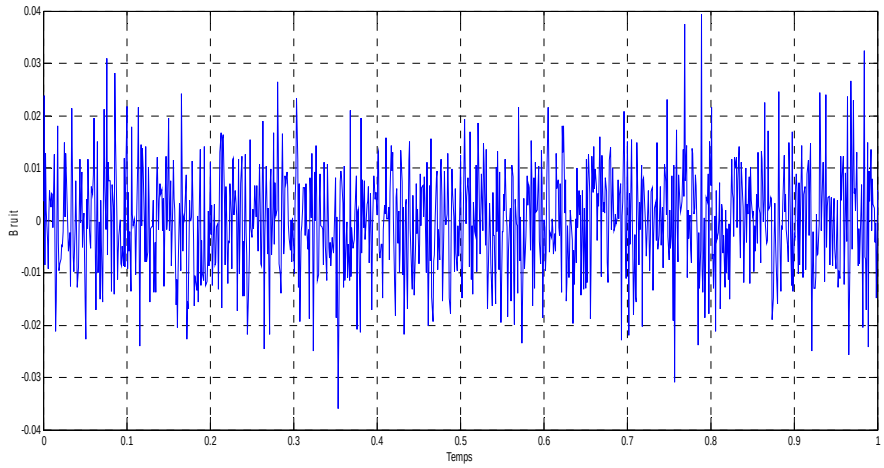


Figure 3.18 : Bruit blanc

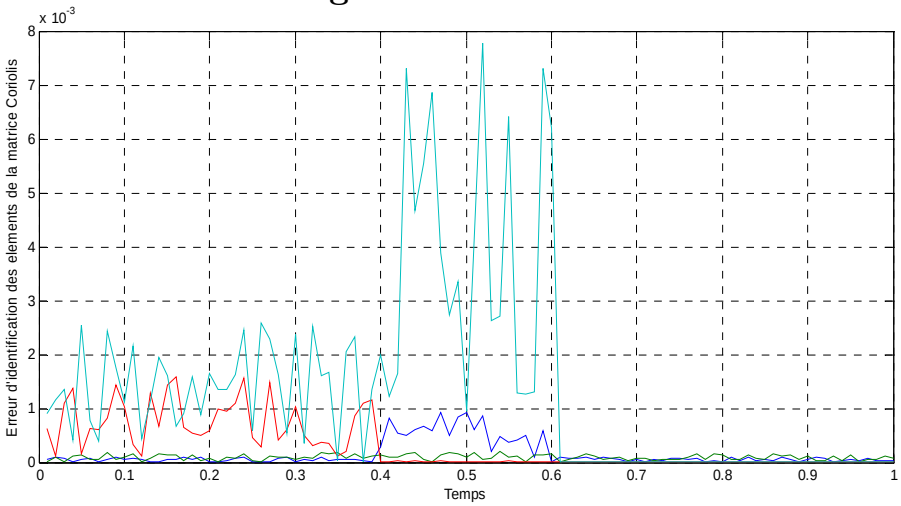


Figure 3.19 : Erreur d'identification des éléments de la matrice d'inertie

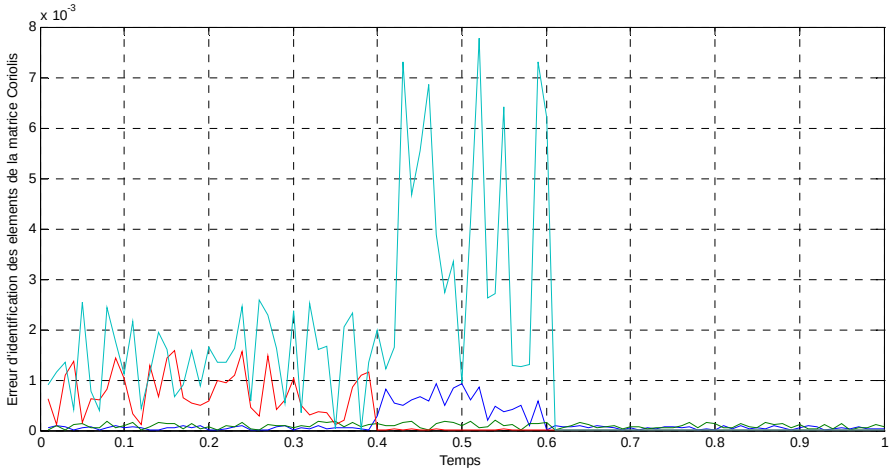


Figure 3.20 *:* Erreur d'identification des éléments de la matrice Coriolis

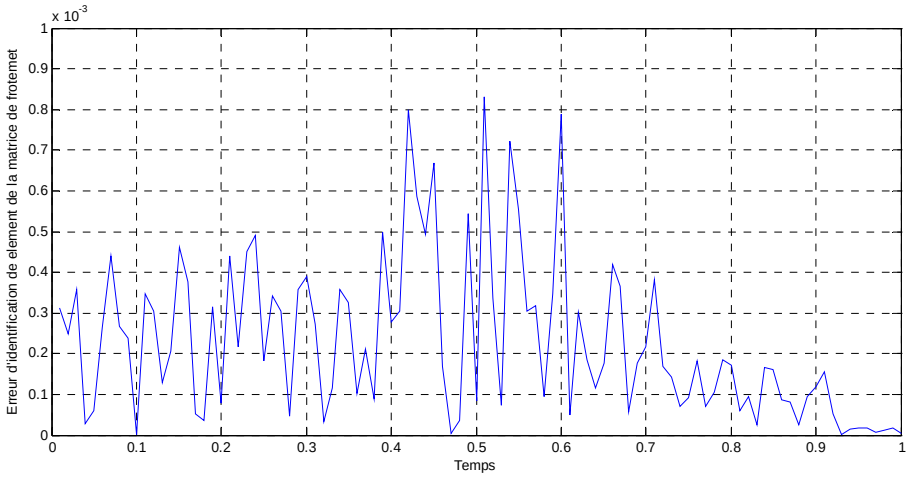


Figure 3.21 : Erreur d'identification d'élément de la matrice de frottements

D'après les résultats obtenus en (Figure 3.19),(Figure 3.20) et (Figure 3.21). Nous constatons qu'ils sont très satisfaisants car l'erreur en position est de l'ordre de 1e-05, ce qui explique la robustesse de la loi de commande basée sur l'architecture SAFIS en présence de bruit.

6. L'impact de la sélection des paramètres prédéfinis pour la robustesse du système

Dans SAFIS, certains paramètres doivent être décidés à l'avance selon les problèmes considérés. Ils comprennent les seuils de distance (ε_{max}, ε_{min} , γ), Le facteur de chevauchement (k) pour déterminer la largeur de la nouvelle règle à ajouter, le seuil d'ajout e_g (growing) pour une nouvelle règle et le seuil d'élagage e_p (pruning) pour enlever une règle insignifiante. Nous avons proposé une procédure générale de sélection pour les paramètres prédéfinis est donnée

comme suit : ε_{max} est réglée à environ la limite supérieure de variables d'entrée, ε_{min} est réglée à environ 10% du ε_{max}; γ est fixé à 0,99. e_p est fixé à environ 10% du e_g. Le facteur de chevauchement (k) est suggéré à choisir dans l'intervalle [1.0, 2.0]. Le seuil e_g est choisi en fonction de la performance du système.

En premier lieu dans la simulation, plusieurs paires de données (ε_{max}, e_g) on été attribués afin de constater l'effet du paramètre e_g sur les performances du système (nombre de règles et le test d'erreur RMS - erreur quadratique -) sous différentes valeurs de ε_{max} et e_g avec k = *1,0.* Les résultats sont détaillés dans la Table 3.2.

ε_{max}	e_g			
	0.001	0.005	0.01	0.05
1.0	(51, 0.0198)	(16, 0.0385)	(10, 0.0535)	(4, 0.0912)
5.0	(35, 0.0233)	(16, 0.0385)	(10, 0.0535)	(4, 0.0912)
10.0	(31, 0.0249)	(15, 0.0386)	(8, 0.0461)	(4, 0.0912)
15	(28, 0.0287)	(14, 0.0389)	(7, 0.0412)	(4, 0.0910)

Table 3.2 : Effets de paramètre, e_g en fonction de ε_{max} et k=1

On constate d'après cette étude que le nombre des règles et la moyenne quadratique de l'erreur est influée généralement par la valeur de e_g .

En deuxième lieu, on va analyser les effets du paramètre e_g sur les performances du système (nombre de règles et le test d'erreur RMS) sous différentes valeurs de k et ε_{max} = 1,0. La table 3.2 montre ces résultats.

k	e_g			
	0.001	0.005	0.01	0.05
1.0	(41, 0.0249)	(14, 0.0386)	(9, 0.0461)	(2, 0.0912)
1.5	(50, 0.0350)	(18, 0.0586)	(15, 0.0598)	(3, 0.1382)
2.0	(52, 0.0557)	(25, 0.0902)	(15, 0.1384)	(3, 0.1391)
2.5	(55, 0.0613)	(31, 0.0985)	(16, 0.1756)	(3, 0.1401)

Table 3.3 : Effets de paramètre, e_g en fonction de k et ε_{max} =1

La table 3.3 montre clairement l'effet du paramètre *eg* sur les performances du système en termes de nombre de règles et l'erreur des tests de RMS sous différentes valeurs de k et ε_{max}.

A partir des deux tables (3.3,3.4), il est facile de constater que l'augmentation de e_g implique que le nombre de règles est diminué et également les performances du système (test d'erreur RMS) devient défavorables avec le

même k et ε_{max}. Ceci, s'explique par le fait que l'optimisation des paramètres de l'algorithme devient une nécessité pour la robustesse de la loi de commande basée sur des approches intelligentes.

7. Conclusion

Dans ce chapitre nous avons présenté une approche intelligente basée sur les techniques de l'intelligence artificielle, pour l'estimation des paramètres du modèle d'un robot manipulateur qui est exposée à des incertitudes paramétriques due à sa forte non linéarité afin d'établir une loi de commande adaptative couple calculé (computed torque). Le noyau de cette approche est les réseaux de neurones RBF. Nous avons considéré l'algorithme d'apprentissage (du SAFIS) en ligne GAP-EKF des réseaux RBF qui utilise une stratégie Growing and Pruning et un filtre de Kalman étendu (EKF) pour modifier les paramètres du réseau (poids, centres et largeurs).

L'algorithme en question commence avec un réseau vide (avec 0 neurones) et tente de construire l'architecture du réseau au fur et à mesure que les couples entrées/sorties seront présentées. Il procède soit en modifiant les paramètres (poids, centres et largeur) du neurone le plus proche de l'entrée présentée si celle-ci est assez proche. Sinon il ajoute un nouveau neurone pour pouvoir réagir à la nouvelle entrée.

Après chaque présentation de couples, l'algorithme élague le réseau en supprimant le neurone le moins signifiant pour empêcher le réseau de s'accroitre exponentiellement et ainsi réduire le temps de calcul.

La loi de commande adaptative computed torque est appliquée au robot SCARA 3 dll. Elle donne des bons résultats dans des conditions parfaites (sans bruits) puis dans des conditions incertaines. A ce stade, la commande adaptative par un réseau SAFIS a fait ses preuves. Les résultats obtenus dans ce chapitre justifient bien le recours aux techniques d'intelligence artificielle dont les réseaux SAFIS font partie. D'un autre coté, le choix des valeurs des paramètres de l'algorithme SAFIS a un impact très important dans la robustesse de la loi de commande robuste et du système.

Les travaux présentés dans ce chapitre ont été validés par une publication internationale ***IJISA 2014***.

Chapitre 4

Optimisation des paramètres du contrôleur PID par l'algorithme immunitaire artificiel

1. Introduction

Les études actuelles dans le domaine de l'automatique ont été témoins d'une croissance rapide de l'utilisation des outils de l'intelligence artificielle, dans le but est de résoudre divers problèmes de contrôle des processus complexes. L'idée est de s'inspiré des mécanismes naturels qui ont été exploités pour développer des méthodes d'analyse heuristique.

La complexité des systèmes non linéaires donne lieu à plusieurs techniques d'intelligence artificielle apportée par la nature biologique afin de développer des algorithmes évolutionnaires. Ces derniers se basent principalement sur l'extraction des métaphores utiles à partir des systèmes biologiques afin de créer des solutions informatiques efficaces. Les chercheurs ont proposé beaucoup de structures de commande qui ont été développées pour commander des systèmes non linéaires, mais la plupart des méthodes sont difficiles et complexes pour être conçues et mises en œuvre.

Dans la littérature les contrôleurs PID sont une solution facile et efficace en raison de leur faible coût de mise en œuvre en l'absence de la connaissance préalable du processus de ces types de contrôleurs.

Les trois principales composants sont impliquées Proportionnelle, Intégrale et Dérivée en se référant à (SOLIHIN, 2011). Dans ce cadre, la procédure de réglage des paramètres PID est considérée comme un problème d'optimisation.

De nombreuses techniques différentes ont été développées pour acquérir les paramètres de contrôle optimal pour les contrôleurs PID. Les problèmes d'optimisation ont été résolus par différents algorithmes évolutionnaires, y compris l'optimisation par des particules essaim comme dans (SOLIHIN, 2011), colonies de fourmis (CHIHA, 2012), algorithmes génétiques proposés par (TENG, 2003) et de l'optimisation biogéographie comme dans (SALEM, 2012).

En outre, le système immunitaire naturel qui défend le corps contre les maladies graves et les infections fait l'objet d'un mécanisme d'optimisation. Ce dernier est un paradigme récent qui tente à assimiler les caractéristiques intéressantes du système immunitaire naturel (SIA), comme la mémorisation, l'apprentissage, le traitement parallèle. Le premier livre sur les Systèmes Immunitaires Artificiels a été écrit par (DASGUPTA, 1999). Les travaux sur le SIA on été remarqués dans (DECASTRO, 2002), (STEVEN, 2000), (TARAKANOV, 2000), (MICHAIL, 2000), (LEANDRO, 2002).

Nous considérons une nouvelle façon de régler les paramètres du régulateur PID, où ces paramètres sont optimisés en utilisant le procédé d'optimisation de l'algorithme immunitaire artificiel (AIA). Cet algorithme est utilisé aussi pour la reconnaissance des formes, l'apprentissage, optimisation les réseaux informatiques, la robotique, ainsi l'identification du modèle.

Dans ce chapitre, nous allons présenter un état de l'art sur l'optimisation et les algorithmes évolutionnaire, dont l'objectif est de présenter le système immunitaire naturel, afin d'entamer l'algorithme immunitaire artificiel AIA qui sera appliquée pour obtenir un réglage optimal des trois termes du contrôleur PID pour contrôler un système non linéaire.

Cette partie est menée par une étude sur le choix des paramètres de l'algorithme AIA et leurs influence sur la robustesse du système et la loi de commande.

2. Optimisation

2.1 Définition

Dans la vie courante, nous sommes fréquemment confrontés à des problèmes qui peuvent être exprimés sous la forme générale des problèmes d'optimisation plus ou moins complexes. (PASCHOS, 2005)

L'optimisation est l'art de comprendre un problème réel, de pouvoir le transformer en un modèle mathématique que nous pouvons étudier afin d'en extraire les propriétés structurelles et de caractériser les solutions du problème. D'après (ANDREASSON, 2005), le terme « optimisation », trouve son origine dans le mot latin « optimum » qui signifie « l'idéal ultime » et « optimus » qui veut dire «le meilleur».

D'une manière simple, résoudre un problème d'optimisation consiste à trouver l'optimum d'une fonction dans un temps d'exécution raisonnable parmi un nombre fini de choix. (WEISE, 2009), (BAECK, 1997). Il s'agit, en général, de maximiser (problème de maximisation) ou de minimiser (problème de minimisation) une fonction objective sous certaines contraintes.

On définit alors une fonction objective, que l'on cherche à optimiser (minimiser ou maximiser) par rapport à tous les paramètres concernés.

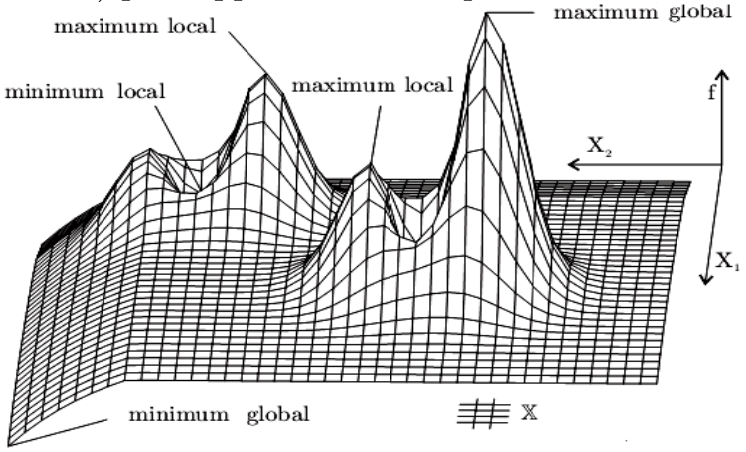


Figure 4.1 : Optimum local et global d'une fonction (deux-dimension).

2.2 Principe d'optimisation et la fonction objective

Mathématiquement, un problème d'optimisation s'écrit sous la forme :

$$\begin{aligned} &\text{minimiser} \quad f_i(x), i = (1,...,M) \\ &x \in \Re^n \ avec \ x = (x_1, x_2, ..., x_n) \\ &Sujet\ à \\ &L_j(x) = 0,\ j = (1.....J) \\ &\varphi_k(x) \le 0, k = (1....K) \end{aligned} \tag{4.1}$$

x est un vecteur d'éléments réels ou entiers de dimension n. Ses composantes x_i sont appelées variables de décision, elles peuvent être continues ou discrètes, dans ce dernier cas, nous parlerons alors d'optimisation combinatoire. Les valeurs possibles des x_i représentent l'espace de recherche noté S dans R_n.

La fonction $f_i(x)$ avec $(i = 1,2,....,M)$ est appelée la fonction coût, ou la fonction objective. Elle attribue à chaque instance de l'espace de recherche une valeur. Les inégalités L_j et φ_k sont appelées les contraintes du problème, qui peuvent être linéaires ou non linéaires.

Pour un problème de minimisation, l'objectif est de rechercher les vecteurs qui minimisent la fonction f tout en respectant les contraintes représentées par les fonctions L_j *et* φ_k.

Les variables de cette fonction sont souvent contraintes d'évoluer dans une certaine partie de l'espace de recherche. Cette fonction à des optima locaux et globaux (CHELOUAH, 2000) (MITCHELL, 1998) :

- Le voisinage d'une solution x est un sous-ensemble de solutions appartenant à S (l'ensemble des solutions possibles) atteignable à partir d'une transformation $V(x)$. Nous pouvons dire qu'une solution x' est voisine d'une solution x, si $x' \in V(x)$. L'application V est appelée structure de voisinage.
- Un point x* est appelé minimum global de la fonction f si :

$$\forall\ x;\ x \neq x^* \Rightarrow\ f(x^*) \langle f(x) \tag{4.2}$$

- Un point x est appelé minimum local fort de la fonction f si (PASCHOS, 2005) :

$$\forall\ x \in V(x^*);\ x \neq x^* \Rightarrow\ f(x^*) \langle f(x) \tag{4.5}$$

- Un point x est appelé minimum local faible de la fonction f si (Figure 4.1) :

$$\forall\ x \in Vx\ (x^*);\ x \neq x^* \Rightarrow\ f(x^*) < f(x) \tag{4.6}$$

Il est à noter que pour les problèmes de maximisation, il suffit de multiplier la fonction coût par (-1).

2.3 Différents problèmes d'optimisation

Il existe plusieurs classes d'optimisation (CHELOUAH, 2000), dont nous citons les plus connues (Figure 4.2) :

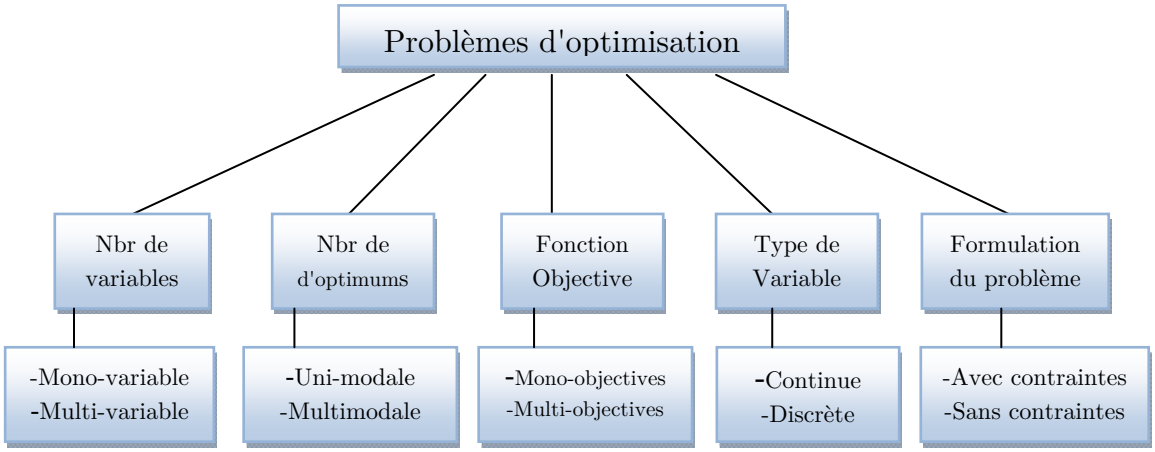


Figure 4.2 Classification des problèmes d'optimisation

- Optimisation Monovariable et Multivariable : Les problèmes d'optimisation mono-variable désignent habituellement tous les problèmes dont la fonction objective ne dépend que d'une seule variable. Les problèmes d'optimisation multivariable rassemblent les problèmes dont la fonction objective est définie par plus d'une variable.
- Optimisation Uni-modale et Multimodale : Un problème dont f ne compte qu'un seul optimum est appelé problème uni-modale. Dans ce cas, un optimum local est aussi un optimum global. A l'inverse, lorsqu'une fonction admet plusieurs optima locaux, elle est dite multimodale
- Optimisation Mono Objectif et Multi Objectif : Les problèmes d'optimisation mono-objective regroupent les problèmes n'ayant qu'un

objectif unique à optimiser. Les problèmes d'optimisation multi-objectif possèdent plus d'un objectif, a optimiser.

- Optimisation continue et discrète : à la différence de l'optimisation continue qui recherche une solution dans un ensemble infini d'objets, l'optimisation discrète regroupe les problèmes dont la solution recherchée est une combinaison d'éléments parmi un ensemble fini d'objets.
- Optimisation avec et sans contraintes : Les problèmes d'optimisation sans contraintes sont des problèmes pour lesquels toute solution s appartenant à S est considérée comme une solution faisable et acceptable. (PASCHOS, 2005).

2.4 Méthodes d'optimisation

Généralement, pour résoudre un problème d'optimisation combinatoire, nous utilisons les méthodes exactes mais lorsque nous sommes confrontés à un problème difficile nous avons recours aux méthodes approchées, dans ce cas le choix est parfois possible entre une heuristique spécialisée, dédiée au problème considéré, et une méta-heuristique qui est une méthode générale (MITCHELL, 1998).

Parmi les méta-heuristiques, nous pouvons différencier les méta-heuristiques à base de voisinage, et les méta-heuristiques à base de population et enfin les méthodes hybrides qui associent souvent une méta-heuristique et une méthode locale.

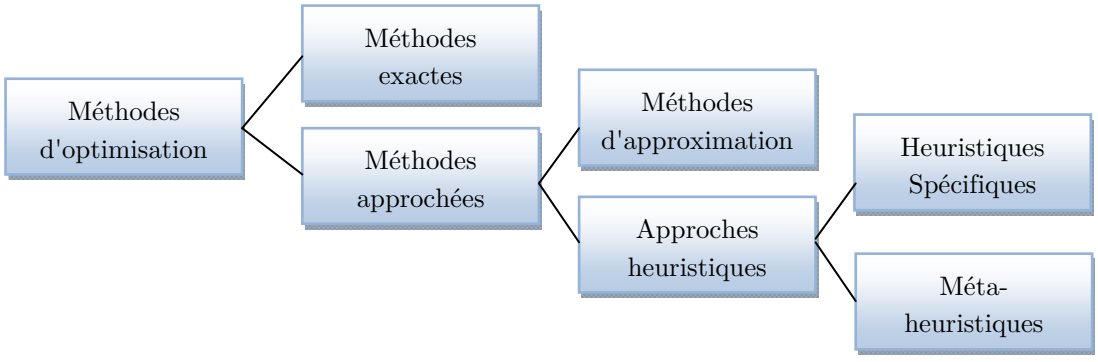


Figure 4.3 Méthodes d'optimisation

2.4.1 Méthodes exactes

Les méthodes qui garantissent la complétude de la résolution sont les méthodes exactes. Ces méthodes donnent à tous les coups la solution optimale. Le temps de calcul nécessaire de telles méthodes augmente en général exponentiellement avec la taille du problème à résoudre. Il faut être conscient que ces méthodes peuvent prendre beaucoup de temps, surtout lorsque les problèmes sont de grande taille (PASCHOS, 2005).

2.4.2 Méthodes à base de voisinage

Dans les problèmes d'optimisation, nous savons que les bonnes solutions sont souvent dans le voisinage d'autres bonnes solutions, l'idée est qu'une bonne stratégie consisterait à se déplacer à travers l'espace de recherche en effectuant de petits pas dans des directions qui améliorent la fonction objective. Cette idée est la base d'une grande famille d'algorithmes appelée méthodes à base de voisinage ou de recherche locale (PASCHOS, 2005). Les méthodes de voisinage diffèrent essentiellement entre eux par le voisinage utilisé et la stratégie de parcours de ce voisinage (MITCHELL, 1998).

2.4.3 Méthodes approchées

Contrairement aux méthodes exactes, les méthodes approchées ne procurent pas forcément une solution optimale, mais seulement une bonne solution (de qualité raisonnable) en un temps de calcul aussi faible que possible. Une partie importante des méthodes approchées est désignée sous le terme de méta-heuristiques qui sont des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé (MITCHELL, 1998).

Plusieurs classifications des méta-heuristiques ont été proposées, la plupart font ressortir globalement trois catégories. Les méthodes à base de solution courante unique, qui travaillent sur un seul point de l'espace de recherche à un instant donné, appelées méthodes à base de voisinage comme la méthode de la descente, le recuit simulé et la recherche tabou, et les méthodes à base de population, qui travaillent sur un ensemble de points de l'espace de recherche, comme les algorithmes évolutionnaires et les colonies de fourmis. Enfin, les méthodes hybrides (MITCHELL, 1998) (Figure 4.4).

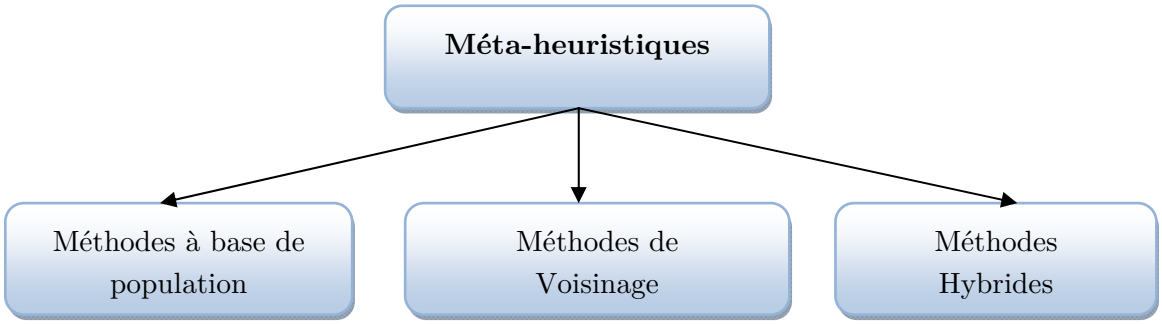


Figure 4.4 : Classification des méta-heuristiques.

2.4.5 Méthodes à base de population

Les méthodes à base de population comme leur nom l'indique, travaillent sur une population de solutions et non pas sur une solution unique comme dans les méthodes à base de voisinage. Nous pouvons distinguer deux grandes classes de ces techniques :

- Les algorithmes évolutionnaires inspirés des concepts issus de la théorie de l'évolution naturelle de Darwin, un exemple typique de ces algorithmes nous trouvons les algorithmes génétiques.
- Les algorithmes inspirés de l'éthologie (l'étude du comportement des diverses espèces animales) comme les colonies de fourmis (MITCHELL, 1998).

2.4.6 Méthodes hybrides

Le mode d'hybridation qui semble la plus fécond concerne la combinaison entre les méthodes de voisinage et les approches d'évolution. L'idée essentielle de cette hybridation consiste à exploiter pleinement la puissance de recherche des méthodes de voisinage et de recombinaison des algorithmes évolutifs sur une population de solutions. (JOG, 1991).

Les algorithmes hybrides sont sans doute parmi les méthodes les plus puissantes. Mais le temps de calcul nécessaires peut devenir prohibitif à cause du nombre d'individus manipulés dans la population. Une voie pour résoudre ce problème est la parallélisation de ces algorithmes sur des machines parallèles ou sur des systèmes distribués (VERHOEVEN, 1995).

3. Algorithmes évolutionnaires

Parmi l'ensemble des techniques de recherche et d'optimisation, le développement des algorithmes évolutionnaires (AE) a été très remarquable dans la dernière décennie (DREO, 2006) (EIBEN, 2003). Les AE font partie du champ de l'Intelligence Artificielle (IA). Ils sont basés sur le concept de la sélection naturelle élaborée par Charles Darwin (KOZA, 1999). Les AE sont des techniques d'optimisation stochastiques globale et itératives, ils simulent un processus d'évolution sur une population d'individus, dans le but de les faire évoluer vers les optima globaux du problème considéré (PASCHOS, 2005).

3.1 Composantes des algorithmes évolutionnaires

Dans la littérature le fonctionnement d'un algorithme évolutionnaire se base sur trois points :

1- Une *population* constituée de plusieurs individus représentant des solutions candidates du problème d'optimisation.
2- Une *fonction d'adaptation* pour évaluer la performance d'un individu (fonction coût).
3- Un *mécanisme d'évolution* de la population composé de plusieurs opérateurs de modification et de sélection permettant d'éliminer certains individus et d'en créer de nouveaux (MICHALEWICZ, 1996). Le cycle d'évolution d'un algorithme évolutionnaire débute avec une population initiale générée d'une façon aléatoire puis répète la procédure suivante (Figure 4.5) :

   - Mesurer la qualité de chaque individu de la population par le mécanisme d'évaluation,
   - Sélectionner une partie des individus,
   - Produire de nouveaux individus par recombinaisons des individus sélectionnés (Mécanisme d'évolution).

Ce processus se termine quand la condition d'arrêt est vérifiée (MITCHELL, 1998).

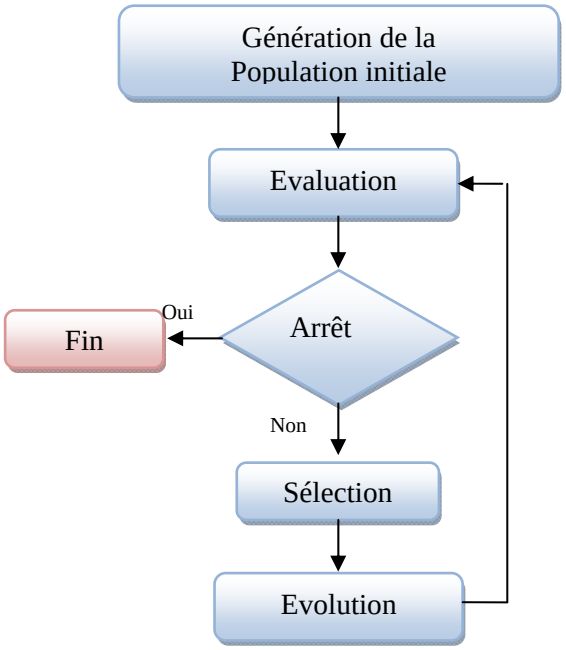


Figure 4.5 Principe général d'un algorithme évolutionnaire

3.2 Conditions de performance des algorithmes évolutionnaires

Les performances des algorithmes évolutionnaires sont conditionnées par les notions de l'exploitation et de l'exploration et comment trouver un bon compromis entre elles (CREPINSEK, 2013).

- L'exploitation (Intensification) insiste sur la capacité d'une méthode de voisinage d'examiner en profondeur des zones de recherche particulières situées à proximité des meilleures solutions. (CREPINSEK, 2013)
- L'exploration (Diversification) met en avant la capacité de découvrir de nouvelles parties prometteuses de l'espace de recherche.

Dans les algorithmes évolutionnaires, la sélection à pour effet de concentrer la recherche autour des configurations de meilleure performance. Du fait de la sélection, la population finit par n'être constituée que d'individus tous similaires. L'une des préoccupations majeures dans les algorithmes évolutionnaires consiste d'ailleurs à préserver le plus longtemps possible une diversité suffisante dans la population. Face à ce type de difficulté, la solution consiste à diriger la poursuite de la recherche vers de nouvelles zones c'est à dire recourir à l'exploration (CREPINSEK, 2013).

4. Système immunitaire naturel

4.1 Définition

Le système immunitaire est un système de défense remarquablement adaptatif. Il est capable de créer une très grande variété de cellules et de molécules susceptibles de reconnaître et d'éliminer spécifiquement un nombre pratiquement illimité d'envahisseurs étrangers. Ces cellules et ces molécules interviennent ensemble dans un réseau dynamique très précisément adaptable, dont la complexité rivalise avec celle du système nerveux (KOZA, 1994).

4.2 Architecture du système immunitaire

Le système immunitaire possède une architecture multicouche (TIMMIS, 2000) qui est constituée de deux couches inter-liées qui sont le système immunitaire inné et le système immunitaire adaptatif ou acquis (Figure 4.6).

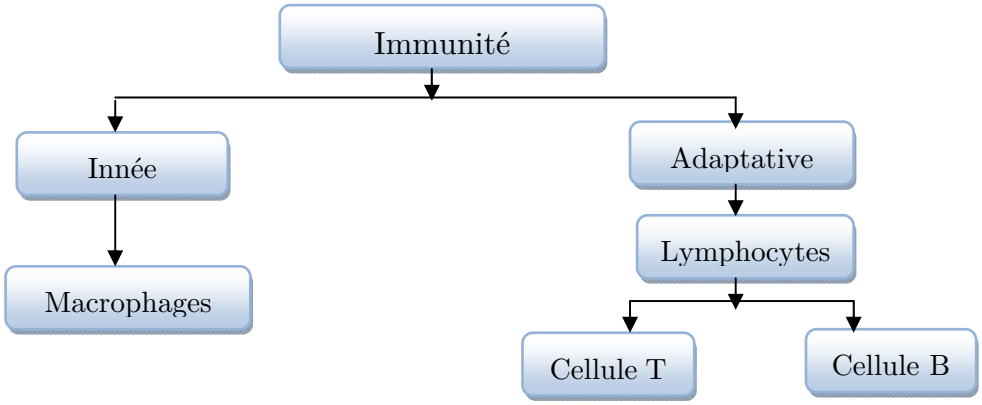


Figure 4.6 : Architecture du système immunitaire

4.2.1 Système immunitaire inné

Le système immunitaire inné est composé d'un ensemble de cellules spécialisées dont le rôle principal est la liaison avec des modèles moléculaires trouvés dans des micro-organismes.

4.2.2 Système immunitaire adaptatif

Le système immunitaire adaptatif est constitué de types différents de cellules dont chacun joue un rôle important. Le rôle central est assuré par les lymphocytes qui sont composés de deux types de cellules : cellule B et cellule T

a) Cellules B

La fonction principale des cellules B consiste à produire et à sécréter des molécules appelées anticorps comme une réponse aux corps étrangers (PERELSON, 1979). Chaque cellule B produit un anticorps spécifique. Le récepteur de cellule B est appelé un anticorps (Ab), (Figure 4.7).

b) Cellules T

Les cellules T peuvent être classées selon deux types : les cellules T d'aide (T helper) et les cellules T cytotoxiques (T killer). Les cellules T d'aide assurent des fonctions essentielles pour la régularisation de la réponse immunitaire par exemple l'activation ou la suppression du développement de certain type de réponse immunitaire. Par contre, les cellules T cytotoxiques assurent des fonctions de suppression des envahisseurs microbiens, des virus ou les cellules cancéreuses. Ainsi, les cellules T présentent des récepteurs sur leur surface (Figure 4.8).

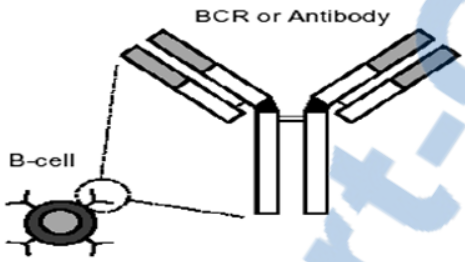


Figure 4.7 : Cellule B

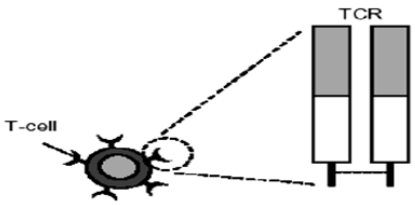


Figure 4.8 Cellule T**.**

4.3 Processus de base d'un système immunitaire

4.3.1 Identification

La reconnaissance d'un antigène est assurée par les lymphocytes. La réaction de chaque lymphocyte est limitée au nombre de cellules étrangères connues comme antigènes. Un antigène est identifié s'il y'a une correspondance entre les récepteurs des cellules et l'épitope de l'antigène (STEVEN, 2000).

Les cellules B sont capables de reconnaître les antigènes libres (Figure 4.9 (a)), tandis que les cellules T ont la possibilité de reconnaître l'antigène qui est présenté par les molécules MHC3 (Figure 4.9 (b)). (STEVEN, 2000), (ALEXENDER, 2000).

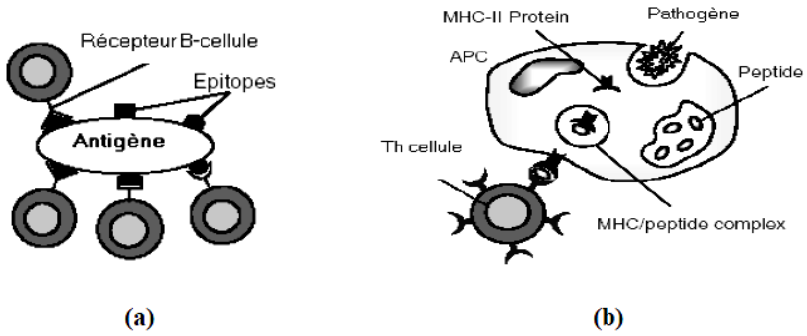


Figure 4.9 L'identification dans le système immunitaire

4.3.2 Activation

L'identification de l'antigène est l'étape préalable du système immunitaire humain qui emploie l'appariement approximatif pour déclencher la réponse (ALEXENDER, 2000). L'appariement entre un récepteur d'un lymphocyte et un épitope de l'antigène détermine l'affinité entre un lymphocyte et un antigène dont l'affinité est le degré de liaison entre le récepteur d'une cellule et l'antigène. (MICHAIL, 2000). Si l'appariement entre un récepteur et l'épitope est fort alors l'affinité est grande sinon elle est petite.

Les anticorps de cellules B matures seront activés d'une manière directe ou indirecte, le type d'activation est déterminé en fonction d'un *seuil d'affinité*. Quand une cellule B correspond à un antigène avec une affinité forte au-dessus du seuil d'affinité, alors elle sera activée directement pour se développer et se différencier. Sinon, si une cellule B correspond à l'antigène avec une affinité faible au-dessous du seuil d'affinité, elle a besoin de l'aide de cellule T d'aide (T helper) pour qu'elle puisse être activée (l'activation indirecte).

4.3.3 Mécanisme de la sélection clonale

En effet, les modèles antigéniques changent constamment. L'efficacité de la détection est maintenue par l'apprentissage dynamique de ces changements qui est assuré par la sélection clonale (PERELSON, 1979). La théorie de la sélection clonale décrit les conséquences de la réponse immunitaire suite à un stimulus antigénique en assurant que seules les cellules qui reconnaissent l'antigène subissent des proliférations et différentiations.

Quand un antigène envahit le corps, des cellules immunitaires reconnaissent cet antigène avec des degrés d'affinité différents. La réponse des

cellules B est la production des anticorps dont chaque cellule sécrète un seul type d'anticorps qui est relativement spécifique à l'antigène. L'appariement fort entre les récepteurs des anticorps et l'antigène produit la stimulation des cellules B c'est-à-dire la prolifération (clone) et la maturation en des cellules de plasma. Le taux de prolifération d'une cellule est directement proportionnel à son affinité avec l'antigène (MATZINGER, 2000) les cellules qui ont les plus grandes affinités seront les plus proliférées et réciproquement. En plus, les lymphocytes qui ont une forte affinité peuvent se différencier en des cellules mémoires. (Figure 3.5).

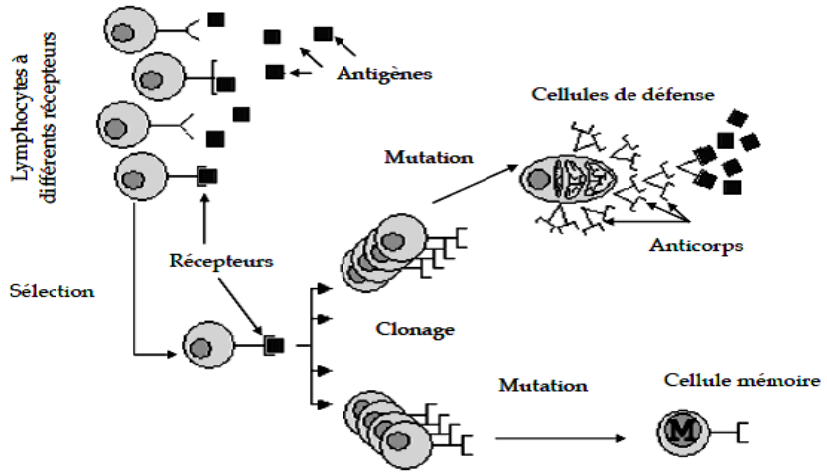


Figure 4.10 Le principe de la sélection clonale

4.3.4 Mémoire immunitaire

L'identification de l'antigène n'est pas suffisante, le système immunitaire humain possède des capacités supplémentaires pour avoir une réponse efficace contre les pathogènes (PERELSON, 1979). Le système immunitaire humain possède plusieurs types de réponses immunitaires qui sont : (MATZINGER, 2000)

- La réponse primaire.
- La réponse secondaire.
- La réponse réactive croisée.

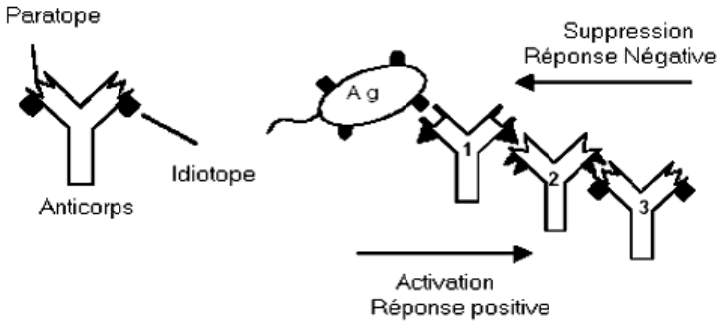
4.3.5 Maturation d'affinité

La maturation d'affinité est le processus qui garantit que le système immunitaire possède de plus en plus des cellules immunitaires spécialisées pour la reconnaissance des modèles antigéniques (JONES, 1998). Ce processus est le résultat du mécanisme de l'hypermutation somatique suivi par une sélection. La mutation qui affecte les parties des récepteurs qui lient avec l'antigène suivi

par une sélection qui garantit la préservation des solutions candidate de hautes qualités. Le récepteur qui possède la plus haute affinité permet d'avoir le plus fort appariement et ainsi la meilleure identification, ce qui permet d'avoir une réponse immunitaire exacte et efficace.

4.4 Réseau immunitaire

La théorie du réseau immunitaire de (JERNE, 1974) suggère que les interactions au sein du système immunitaire ne se limitent pas aux anticorps et antigènes, mais aussi entre les anticorps même en absence d'un stimulus antigénique. Cette interaction est assurée par des récepteurs spécialisés présents sur la surface des anticorps appelés idiotope.



.

Figure 4.11 : La représentation du réseau immunitaire idiotypique

4.5 Caractéristique du système immunitaire

La section précédente a été consacrée pour présenter le système immunitaire humain et son processus d'identification et d'activation, ainsi que certains processus immunitaires de base. Cette partie sera consacrée à une récapitulation des propriétés intéressantes du système immunitaire qui constituent d'un point de vue informatique une source d'inspiration très riche :

- ***Parallélisme*** **:** Système immunitaire (SI) capable de produire plusieurs réponses immunitaires en même temps à des endroits dispersés.
- ***Tolérance au soi*** : SI peut différencier entre les cellules de soi et celles de non soi.
- ***Apprentissage*** **:** SI augmente la capacité d'identification des anticorps à un antigène sélectif. Il apprend continuellement les structures de pathogènes.

- ***Adaptabilité* :** SI permet la production des cellules de plus en plus spécialisées pour l'identification des antigènes. Cela est garantit par la théorie de la sélection clonale.
- ***Dynamique* :** SI change continuellement par la création de nouvelles cellules et molécules, l'élimination des cellules vieilles ou endommagées.
- ***Mémorisation* :** Après une réponse immunitaire à un antigène donné, un ensemble de cellules constituent l'ensemble de cellules mémoires afin de fournir des réponses immunitaires plus rapides et plus puissantes aux rencontres suivantes d'un même antigène.

5. Système immunitaire artificiel (SIA)

5.1 Définitions

Les systèmes immunitaires artificiels (SIA) sont une nouvelle branche de l'intelligence artificielle destinés à résoudre des problèmes divers, inspiré des propriétés et concepts remarquables du système immunitaire biologique (TIMMIS, 2003).

Les SIA sont une implémentation mathématique ou informatique du fonctionnement du système immunitaire naturel. Cette implémentation reprend les plus grandes lignes de son fonctionnement. Cependant il reste presque impossible de modéliser le comportement complet des systèmes biologiques. Plusieurs définitions a été injectée dans littérature, parmi elles :

- ***Définition* 1** Selon (TIMMIS, 2003) : « Un système immunitaire artificiel est un système informatique basé sur les métaphores du système immunitaire naturel ».
- ***Définition* 2** : (DASGUPTA, 1999) a défini le système immunitaire artificiel comme suit : « Le système immunitaire artificiel est la composition de méthodologies intelligentes inspirées par le système immunitaire naturel afin de résoudre des problèmes du monde réel ».
- ***Définition* 3** Tandis que (TIMMIS, 2003) ont donné la définition suivante (DE CASTRO, 2002) : « Les systèmes immunitaires artificiels sont des systèmes adaptatifs inspirés par des théories immunologiques et des observations de fonctions immunitaires, des principes et des modèles, qui seront appliqués à la résolution des problèmes ».

5.2 Synthèse sur les systèmes immunitaires artificiels (SIA)

Nous avons essayé de réaliser une synthèse non exhaustive de quelques unes de ces applications dans la Table 4.1

Référence	**Model ou technique de description**	**Théorie**	**Application**
(Neal, 2003)	Meta-stable memory immune system for multivariate data analysis.	Réseaux immunitaires	Analyse de données
(Rouchen , 2003)	Un algorithme de sélection clonal pour l'optimisation. multi- tâche	Sélection clonale	Optimisation
(Zuo, 2003)	A Chaos Artificial Immune Algorithm (CAIF) by integrating of chaotic search and CLONALG	Sélection clonale	Optimisation
(Nasraoui, 2003)	Techno – streams model for detecting an unknown number of evolving clusters in a noisy data stream	Réseaux immunitaires	Classification
(Secker, 2003)	Un système immunitaire artificiel pour la classification des E-mail	Réseaux immunitaires	classification
(Garrett, 2004)	Un algorithme clonal adaptatif (ACS) une amélioration de CLONALG	Sélection clonale	Optimisation
(Gonzalez, 2004)	Un algorithme de sélection négative pour la détection des anomalies	Sélection clonal	Détection des anomalies
(Yu and Hou, 2004)	Un algorithme de sélection clonal amélioré de CLONALG	Sélection clonale	Machine Learning
(Luh, 2004)	Un réseau immunisé réactif (RIN) pour le robot mobile apprenant des stratégies de navigation dans les environnements inconnus	Réseaux immunitaires	Contrôle des systèmes robotisés
(Panimadai, 2007)	Optimal Control of Class of Non-linear Plants Using Artificial Immune Systems: application of the Clonal Selection Algorithm	Sélection clonale	Optimisation et contrôle

(H. Wang,2008)	Artificial immune system based image pattern recognition in Energy efficient wireless multimedia sensor networks	Réseaux immunitaires	Traitement d'image
(Sahraoui, 2014)	An artificial immune optimization approach in tuning nonlinear PID controllers	Sélection clonale	Optimisation des paramètres
(Leonardo ,2014)	Artificial Immune Systems applied to the reconfiguration of electrical power distribution networks for energy loss minimization	Réseaux immunitaires	distribution d'énergie électrique

Table 4.1 Application concernant le système immunitaire artificiel

Le SIA a été largement utilisé dans différents domaines, Ce qui implique la précision et la robustesse des commande basée sur ce type de système adaptatif, soit pour estimer les paramètres incertains du modèle, soit pour la reconnaissance des forme, soit pour la commandes des systèmes non linéaires soit pour estimer la partie corrective de la commande, ainsi que pour l'optimisation etc.

5.3 Processus de conception d'un système immunitaire artificiel

5.3.1 Modélisation des systèmes immunitaires artificiels

Le modèle commun connu sous le nom du Framework des systèmes immunitaires artificiels, définit les règles qui doivent respecter un AIS ainsi que les processus à suivre pour l'élaboration de nouvelles approches. Les conditions nécessaires sont :

- La représentation des composants systèmes (modèles abstraits des cellules immunitaires).
- L'utilisation des mesures d'affinité pour évaluer l'affinité entre les composants systèmes.
- Un ensemble d'algorithmes pour contrôler l'évolution et la dynamique d'AIS.

Les trois conditions citées ci-dessus sont indispensables pour l'élaboration d'un Framework pour définir un système immunitaire artificiel.

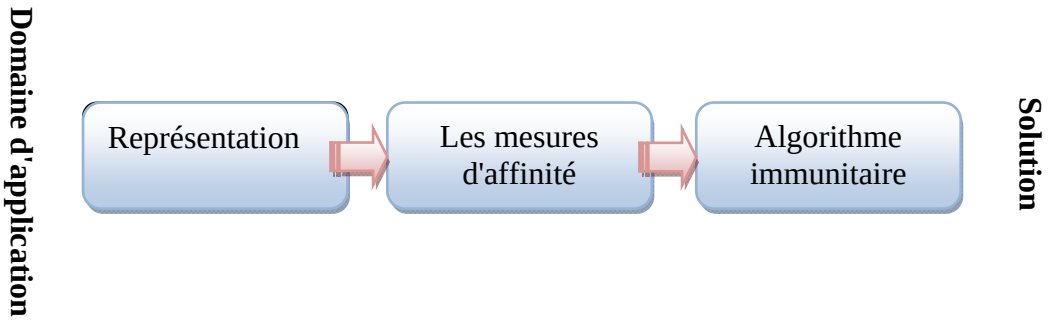


Figure 4.12 : Structure de conception d'un AIS

Ce schéma est adopté par (TIMMIS, 2003)

5.3.2 Mesures d'affinités

L'affinité entre un anticorps et un antigène est relative à leur distance (JONES, 1998) Elle peut être estimée via n'importe quelle mesure de distance entre deux chaînes (ou vecteurs) par exemple par l'utilisation de la distance *Euclidienne*, la distance de *Manhattan* ou la distance de *Hamming* (JONES, 1998). Si on considère un anticorps Ab = <Ab1, Ab2, …AbL> et un antigène Ag =<Ag1, Ag2, …AgL>, alors la distance D peut être calculée selon l'une des distances précédentes qui seront présentées respectivement comme suit :

- La distance Euclidienne : calcul de la distance entre les deux points Ab et Ag dans chacun des espaces : (DEREK, 1997)

$$D = \sqrt{\sum_{i=1}^{L} (Ab_i - Ag_i)^2} \tag{4.7}$$

- La distance de Manhattan : calcul de la distance entre les deux points Ab et Ag en ne suivant que les axes de coordonnées de l'espace : (LEANDRO, 2002)

$$D = \sum_{i=1}^{L} \left| Ab_i - Ag_i \right| \tag{4.8}$$

-La distance de Hamming : représente les anticorps et les antigènes sous forme de symboles et calcule la différence entre ces symboles.

$$D = \sum_{i=1}^{L} \delta_i \quad avec\ \delta_i = \begin{cases} 1 & si\ Ab_i \neq Ag_i \\ \delta_i = 0 & \text{si } non \end{cases} \tag{4.9}$$

5.4 Mécanismes du système immunitaire artificiel

Les mécanismes (algorithmes) des SIA peuvent être divisés en trois grandes parties, chacune d'elles s'inspirant d'un comportement ou d'une théorie du système immunitaire biologique : la sélection clonale, la sélection négative et positive, les réseaux immunitaires (ou idiotypiques). (STEVEN, 2000) (DEREK, 1997), (LEANDRO, 2002).

5.4.1 Sélection négative/positive

Le système immunitaire humain utilise la sélection négative pour éliminer les cellules immunitaires immatures. L'idée sur laquelle se base cette théorie selon (FORREST, 1996) est que seules les cellules T qui ne s'attaquent pas aux cellules du soi sont autorisées à quitter le thymus et auront pour tâche de reconnaitre les cellules du non soi.

L'algorithme de ce mécanisme se déroule comme le montre la figure 4.13:

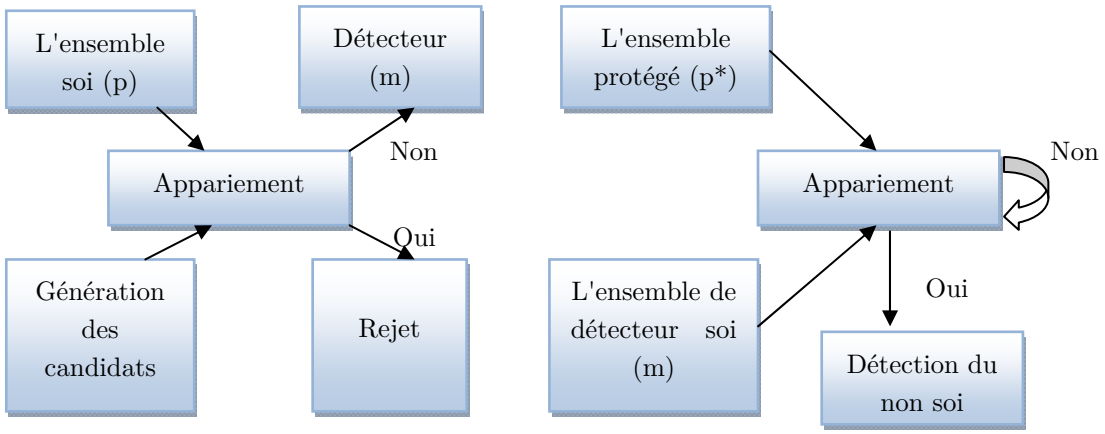


Figure 4.13 : La structure générale de l'algorithme de la sélection négative.

L'algorithme de la sélection positive est une alternative de l'algorithme de la sélection négative. La différence principale est la génération des détecteurs qui détectent des éléments de soi au lieu de ceux qui détectent des éléments de non soi. Selon cet algorithme, un élément de non soi suspect doit être comparé avec tout l'ensemble des détecteurs de soi ; s'il n'est pas détecté alors il est considéré comme un élément de non soi.

5.4.2 Sélection clonale

(DE CASTRO, 2002) ont proposé l'algorithme de la sélection clonale nommé CLONALG qui accompli les tâches de base impliquées dans le processus

de la sélection clonale dans le système immunitaire humain. Les étapes de base de l'algorithme CLONALG sont résumées dans l'enchainement suivant :

1. Soit la population (M) de solution candidate.
2. Déterminer l'affinité de chaque solution candidate avec un ensemble d'antigènes.
3. Choisir les n_1 meilleurs éléments de (M) dont ils ont les plus hautes affinités et produire des copies de ces individus proportionnellement à leur affinité avec l'antigène : l'élément qui possède la plus haute affinité aura le plus haut nombre de clones et réciproquement.
4. Muter toutes ces copies avec un taux inversement proportionnel à leur affinité avec l'antigène : l'élément qui possède la plus haute affinité aura un taux de mutation faible et réciproquement.
5. Sélectionner n_2 éléments à partir des clones mutés dont ils ont la plus haute affinité pour remplacer les n1 anticorps à l'origine choisis par ces mutants.
6. Remplacer les cellules de faible affinité par des nouvelles cellules aléatoires.
7. Répéter les pas 2 à 5 tant que certains critères sont vérifiés.

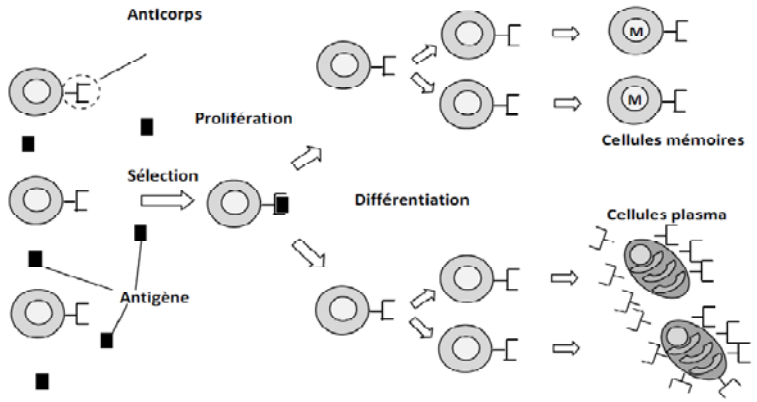


Figure 4.14 : La structure générale de l'algorithme de la sélection clonale

5.5 Appariement entre système immunitaire et algorithme immunitaire artificiel

L'appariement entre le système immunitaire et le problème d'optimisation se fait comme suit : la réponse immunitaire représente les solutions et les antigènes représentent le problème à résoudre. Plus précisément, les cellules B sont considérés comme agents artificiels qui errent autour et à explorer un environnement. Le mécanisme de sélection positif et négative est utilisé pour éliminer les inutiles ou de mauvaises solutions.

Systèmes immunitaires SIA	Algorithmes immunitaires AIA
Antigènes	Problème à résoudre
Anticorps	Valeur des meilleures solutions
Reconnaissance antigènes	Identification du problème
Production d'anticorps à partir des cellules mémoires	Chargement des meilleures solutions préalablement trouvées
Suppression des cellules T	Elimination du surplus des solutions potentielles
Prolifération d'anticorps	Utilisation d'un processus pour la création de copies exactes de la solution

Table 4.2 : Appariement entre le système immunitaire et l'algorithme immunitaire

6. Etape de l'algorithme immunitaire artificiel (AIA)

6.1 Initialisation des paramètres de l'AIA

L'initialisation des paramètres d'AIA sont nécessaires. Ces paramètres comprennent la taille de la population qui indique le nombre d'individus, nombre de générations nécessaires pour le critère de terminaison, le nombre d'anticorps à cloner, facteur de multiplication, le nombre de variables de conception et les gammes respectives pour les variables de conception.

6.2 Génération de la population

La deuxième étape consacrée à la génération d'une population aléatoirement égale à la taille de la population spécifiée (KOZA, 1994). Cette valeur de la variable de conception est générée aléatoirement entre la plage de variable de conception spécifié. Dans AIA, population signifie le groupe des anticorps qui représente là un ensemble de solutions.

6.3 Calcul des valeurs de la fonction objective

Le calcul est effectué pour obtenir les valeurs de la fonction objective pour tous les membres de la population. La valeur de la fonction objective ainsi obtenu indique l'affinité de l'anticorps. L'affinité du l'anticorps est décidée par sa valeur de la fonction objective selon (4.7).

6.4 Sélection clonale

Sélectionnez les n anticorps les plus élevées en affinité de la population qui comprend un nouvel ensemble d'anticorps de haute affinité. Ces anticorps sont clonés en fonction de ses affinités. Cela génère un groupe de clones. Plus l'affinité est élevé plus le nombre de clones générés pour chacun des n anticorps sélectionnés. Il est calculé comme :

$$N_c = \sum_{i=1}^{n} round\left(\frac{\beta N}{i}\right) \tag{4.10}$$

Où est le facteur de multiplication qui contrôle le nombre de clones et N est le nombre total d'anticorps. Les n anticorps clonés sont sélectionnés de façon indépendante et proportionnelle à leurs affinités.

6.5 Maturation

Le groupe de clones subit par le processus de maturation qui est inversement proportionnel à l'affinité (ie: l'affinité est élevée, le taux de mutation est faible) tel que :

$$x_{i,m} = x_i + A(rand\left[-1,1\right])(x_{\max} - x_{\min}) \tag{4.11}$$

Où, A est un facteur dépendant de l'affinité il diminue au fur et à mesure que l'augmentation d'affinité. Un nouvel ensemble de solutions est généré constitué de clones mûri, a partir de cet ensemble de clones matures, sélectionner à nouveau des solutions de la plus haute affinité. Si l'affinité antigénique de cette solution est meilleure que la solution de l'itération précédente, alors remplacer la population par la nouvelle.

6.6 Modification

Remplacer les anticorps ayant une affinité plus basse de la population par de nouveaux anticorps en utilisant (4.12)

$$x_i = x_{\min} + rand(0,1)(x_{\max} - x_{\min}) \tag{4.12}$$

6.7 Régénération

Répétez les étapes (de l'étape 3) jusqu'à ce que le nombre spécifié de générations ou critère de rupture soient atteint.

L'algorithme AIA peut être structuré comme dans l'organigramme suivant :

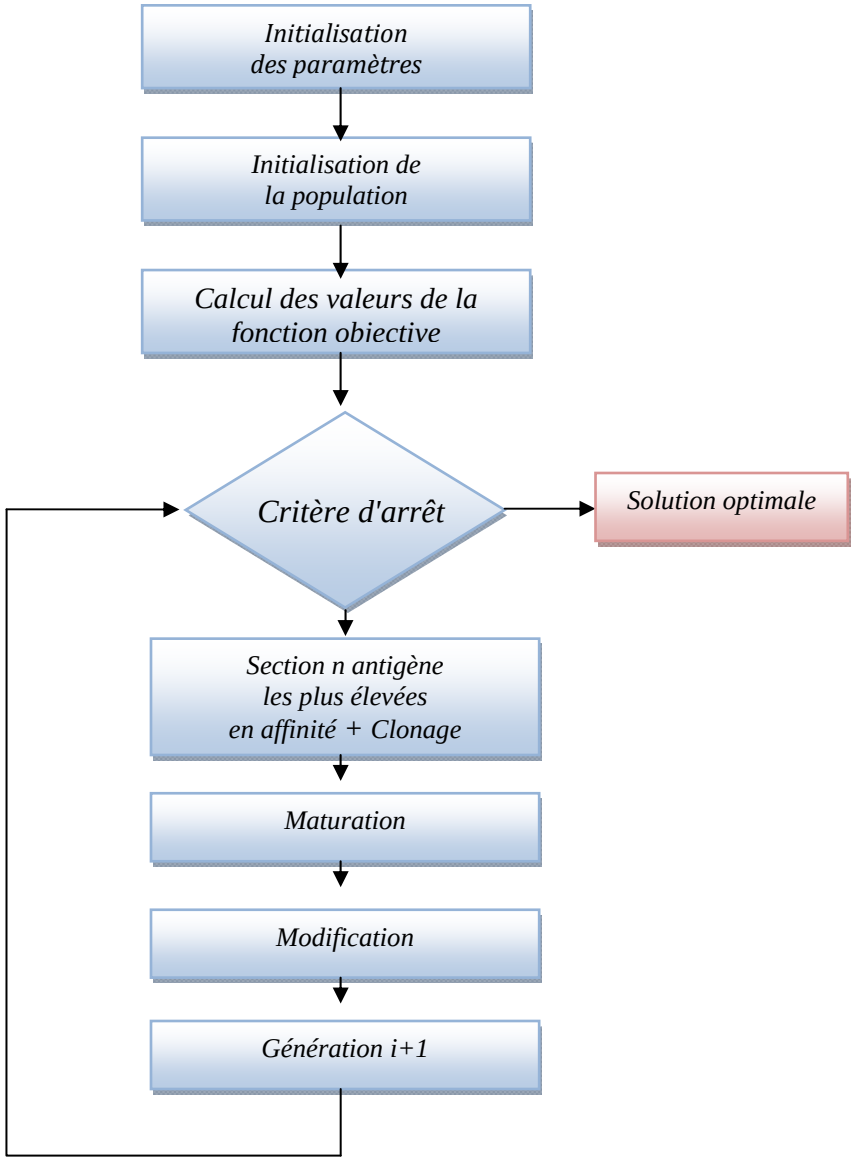


Figure 4.15 : Organigramme général de l'algorithme AIA

7. Optimisation des gains du contrôleur PID par l'algorithme immunitaire artificiel (AIA)

L'optimisation est l'une des branches les plus importantes des mathématiques appliquées, elle a été le sujet de nombreuses recherches à la fois pratiques et théoriques. Une étude comparative de quelques approches méta-heuristiques pour le problème de l'optimisation du PID est donnée dans

(NAGARAJ, 2011). Parmi les problèmes les plus connus dans le domaine de l'automatique, nous citons le problème de trouver la combinaison optimale des gains d'un contrôleur PID. Le réglage des paramètres d'un contrôleur PID peut être considéré comme un problème d'optimisation, d'où il s'agit de trouver la solution optimale des gains du contrôleur dans un espace de recherche prédéfini pour permettre au système de suivre une référence désirée. (SAHRAOUI, 2015).

Nous avons essayé de dresser un résumé des récentes approches naturelles d'optimisation pour le réglage du contrôleur PID. La Table 4.2 représente pour chaque algorithme, le système utilisé ainsi que l'algorithme comparé. La plupart des approche sont comparées aux algorithmes génétiques et avec la méthode conventionnelle Ziegler Nichols.

Algo	**Approche**	**Système**	**Comparaison**
AG	(Kim, 2008) (Yin, 2004) (Zain, 2009)	Système d'osmose inversé Système sous-amorti Bras manipulateur flexible	Ziegler-Nichols
DE	(Ruijun, 2009) (Youxin, 2010)	Système de premier ordre Servo-moteur	PSO
ACO	(Soundarrajan , 2010) (Berbaoui, 2011)	Moteur à courant continu Filtre actif	Ziegler Nichol
PSO	GirirajKumar, 2010) (Solihin, 2011) (Lahoty , 2013	Perceuse Moteur DC Système FOPDT	Ziegler Nichol AG
ABC	(Gozde , 2010) (Sundareswaran, 2008).	AVR Moteur à courant continu	AG
BBO	(Salem, 2012)	Pendule inversé	AG

Table 4.3 : Méta-heuristiques pour l'optimisation d'un PID

7.1 Architecture d'optimisation adoptée

L'optimisation basée sur l'algorithme immunitaire artificiel (AIA) est parmi les algorithmes d'optimisation les plus récents (SIMON, 2008). La section courante est consacrée à la présentation et l'application de l'AIA, dont l'objectif est trouver la combinaison optimale des paramètres du contrôleur PID (K_p, K_i, K_d) afin de contrôler un système non linéaire. L'espace de

recherche défini dans l'équation (4.13) est limité par les contraintes physiques supportées par le système étudié.

$$\begin{aligned} K_p &\in \left[K_{p\min}, K_{p\max} \right] \\ K_i &\in \left[K_{i\min}, K_{i\max} \right] \\ K_d &\in \left[K_{d\min}, K_{d\max} \right] \end{aligned} \tag{4.13}$$

Lors de la première étape, la population initiale va être créée aléatoirement avec n antigènes contenant 3 variables de décision : $K_p;K_i;K_d$.

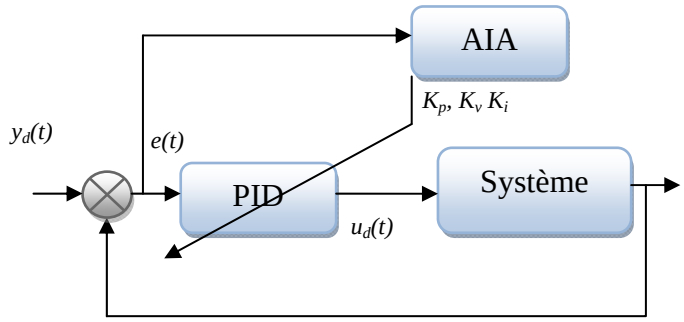


Figure 4.16 : Architecture d'optimisation adoptée

7.2 Application de l'algorithme immunitaire artificiel avec le système masse ressort amortisseur

Pour évaluer l'algorithme proposé, nous allons l'appliquer pour ajuster les paramètres d'un contrôleur PID pour commander un système non linéaire système masse-ressort-Amortisseur décrit dans l'annexe avec un frottement donnée par (EMMA, 2008).

Nous avons utilisé deux fonctions objectives différentes (4.14).

$$\begin{cases} f = \int e^2 dt \\ h = (1-\exp^{-\beta}) * (over + s_e) + \exp^{-\beta}(t_s - t_r) \end{cases} \tag{4.14}$$

Paramètres	***Signification***
t_s	temps de stabilisation
t_r	temps de montée
S_e	erreur statique
$Over$	Dépassement
β	constante à définir

Table 4.4 Paramètres indiqué dans les fonctions objectives

7.2.1 Résultats des simulations en utilisant la fonction objective *f*

Afin de commander le système Masse-Ressort-Amortisseur, nous avons appliqué l'algorithme AIA pour optimiser les paramètres du contrôleur PID.

La référence que le système doit atteindre est $x_d = 2$m. Avant de procéder l'approche proposée, les paramètres nécessaires de l'algorithme AIA à l'accomplissement de l'apprentissage sont initialisés dans le Table 4.5. Lors de la première étape, la population initiale va être crée aléatoirement avec n antigènes contenant 3 variables de décision : $K_p;K_i;K_d$.

Taille de population Nombre de génération	100 100
Nombre des anticorps a clonés (nc)	40
Facteur multiplicatif (F)	0.02
Nombre des individus	3
K_p,K_i,K_d	[0 50], [0 50], [0 50]

Table 4.5 : Initialisation des paramètres d'AIA

L'évolution des minima de la fonction f (4.13) au long des 100 générations est donnée dans (Figure 4.20). Le minimum atteint la valeur 0.587 autour de troisième génération. Les gains correspondants aux meilleures solutions sont représentés dans la Figure 4.19 où nous pouvons constater que la combinaison retenue est K_p=*20*,K_i=*50 et* K_d =0.

Les positions désirées et réelle sont montrées dans (Figure 4.17) et l'erreur correspondante dans (Figure 4.18).

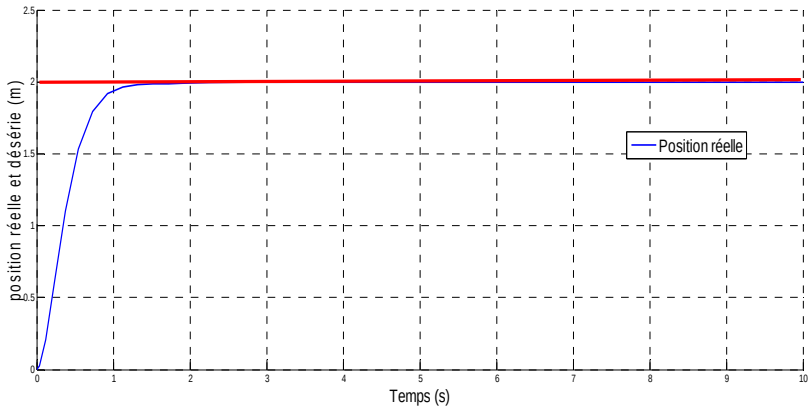


Figure 4.17 la position désirée à l'aide de l'AIA (avec la fonction objective *f)*

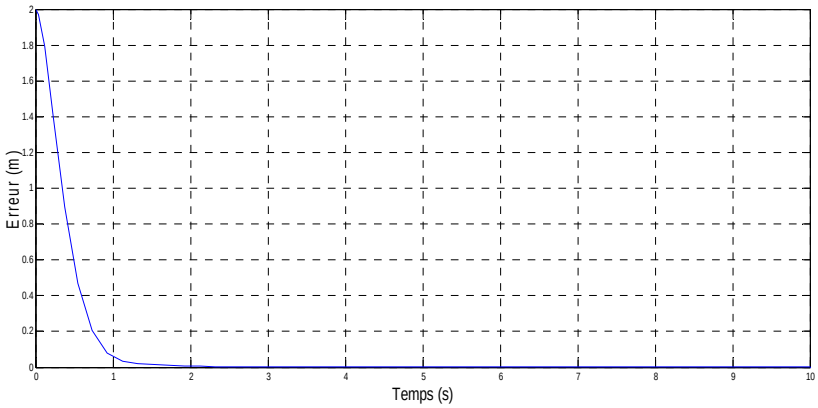


Figure 4.18 : Erreur de position (avec la fonction objective *f*)

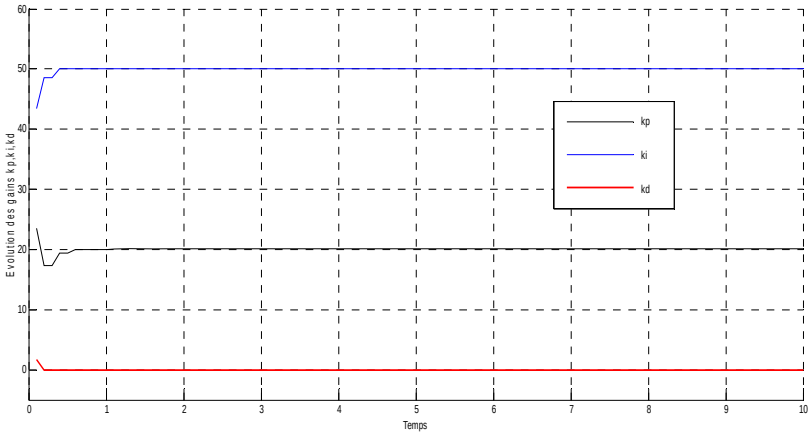


Figure 4.19 : Évolution des gains du PID par AIA (avec la fonction objective *f)*

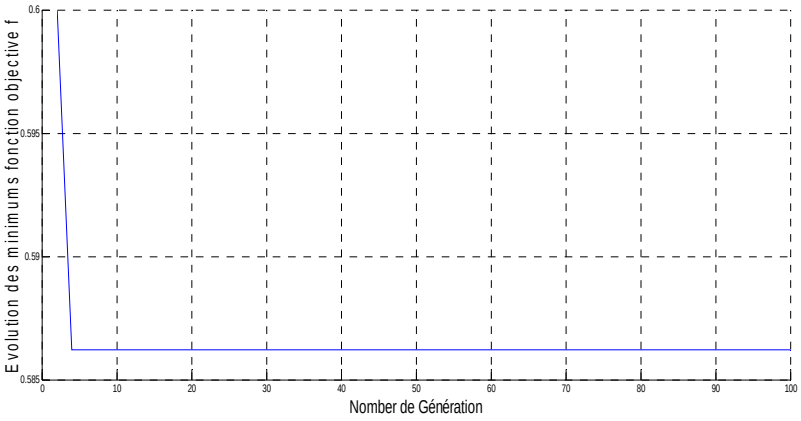


Figure 4.20 : Evolution des meilleurs minimums de la fonction Objectif *f*

7.2.2 Résultats des simulations en utilisant la fonction objective *h*

Dans les même conditions, la deuxième application de l'algorithme proposé pour optimiser les paramètres du contrôleur PID est basée sur la deuxième fonction objective h comme dans (4.14) afin de commander le système masse ressort amortisseur.

L'évolution des minima de la fonction h le long des 100 générations est donnée dans (Figure 4.23) où nous pouvons remarquer que l'algorithme atteint son minimum après 10 générations. Les gains correspondants au minimum obtenu sont dans (Figure 4.25). L'erreur de régulation est présentée dans (Figure 4.22).

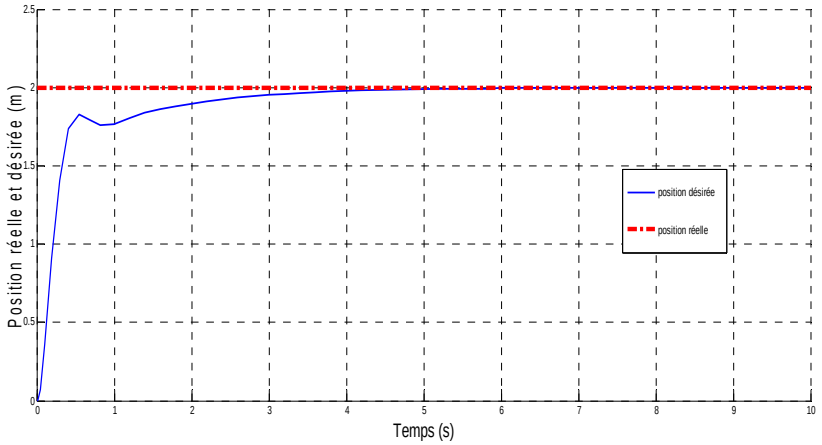


Figure 4.21 la position désirée à l'aide de l'AIA (avec la fonction objective *h)*

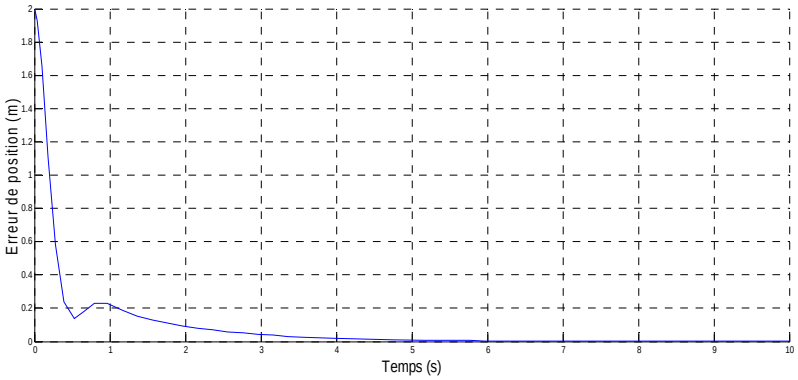


Figure 4.22 : Erreur de position (avec la fonction objective *h)*

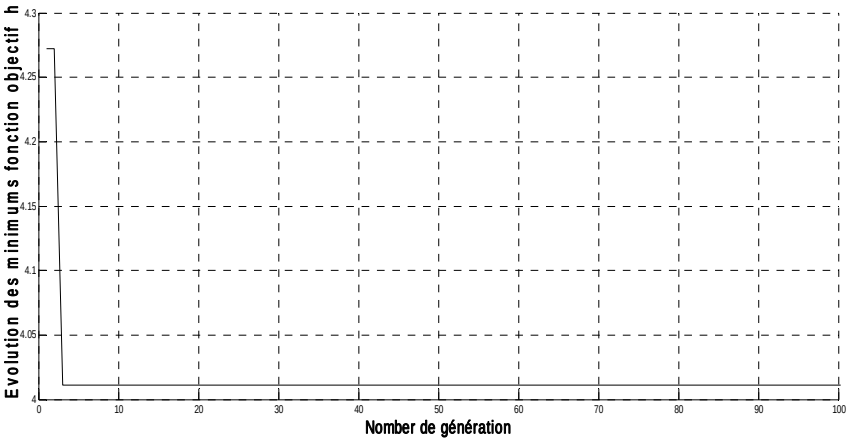


Figure 4.23 : évolution des minimums de la fonction objective h

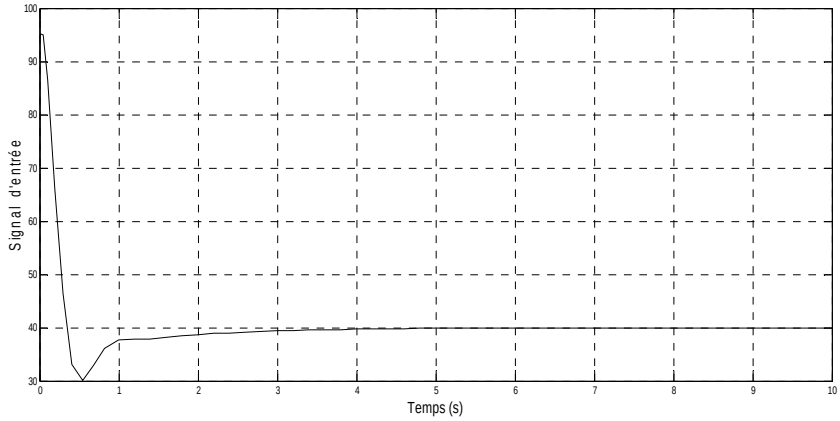


Figure 4.24 : Evolution de signal du contrôleur PID

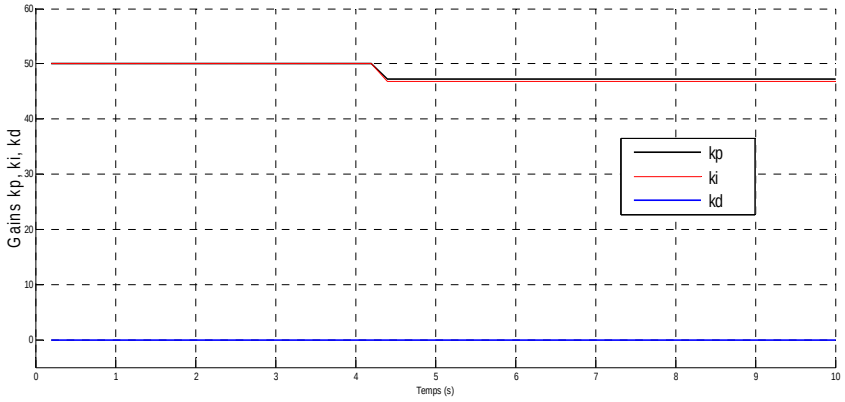


Figure 4.25 : Évolution des gains du PID par AIA (avec la fonction objective *h)*

7.3 Etude Comparative de l'algorithme AIA avec les Algorithmes génétiques

Les performances de l'algorithme AIA ont été comparées avec celle des algorithmes génétiques (AG), La comparaison a été réalisée dans les mêmes conditions (Nombre d'évaluations, Taille de population, Population initiale). Le

système masse ressort amortisseur a été utilisé pour comparer les algorithmes. La position réelle et la position désirée sont représentées en (Figure 4.26).

Il est clair que le minimum de la fonction objectif h avec l'algorithme AIA est meilleur que celui des AG avec une stabilité dans les valeurs des minimums depuis 10 générations pour AIA comme il est indiqué dans (Figure 4.27).

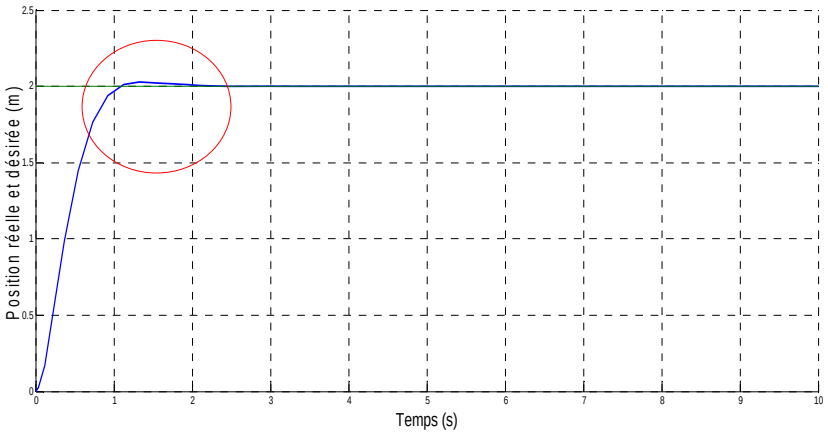


Figure 4.26 : Position réelle et désirée avec les algorithmes génétiques (AG)

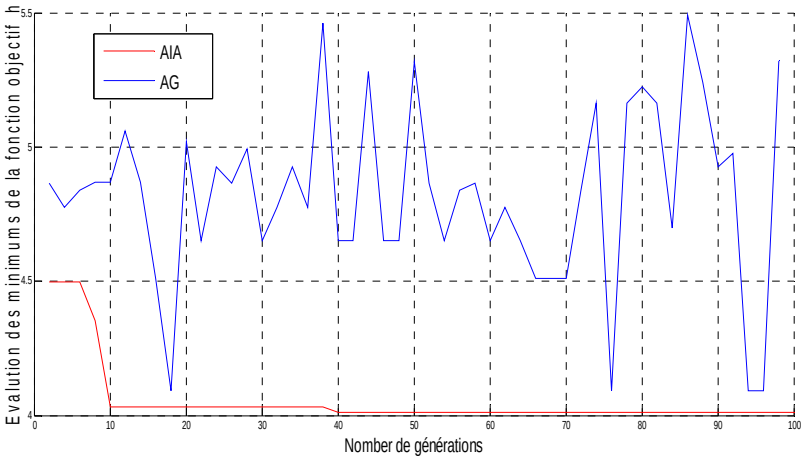


Figure 4.27 : Evolution des minimums de la fonction objectif h (AG and AIA)

7.4 Test de robustesse

Afin de finaliser cette partie, nous avons constaté que l'optimisation des paramètres du contrôleur PID (K_p, K_i, K_d) pour avoir un maximum de performance des systèmes non linéaire repose sur l'utilisation et le bon choix des paramètres de l'algorithme AIA. Cela nous a permis d'avoir une étude sur la robustesse empirique des paramètres de l'algorithme immunitaire artificiel.

7.4.1 L'impact du paramètre facteur multiplicateur de clonage (F) sur les performances du système

Avec les mêmes conditions dans la Table 4.6 et des différentes valeurs du facteur multiplicateur de clonage (F), on peut voir le tracé de l'erreur de position après plusieurs exécutions dans (Figure 4.28), et le test de la moyenne l'erreur quadratique (RMS:*root mean square*) des l'erreur de position selon les valeurs de F dans la Table 4.7.

Taille de population Nombre de génération	100 100
Nombre des anticorps a clonés	40
Nombre des individus	3

Table 4.6 : Valeur des paramètres de l'AIA

	Facteur multiplicateur de clonage				
	0.03	0.04	0.05	0.06	0.07
Erreurs	0.0895	0.0875	0.0853	0.0867	0.0873

Table 4.7 : Effet du paramètre facteur de clonage

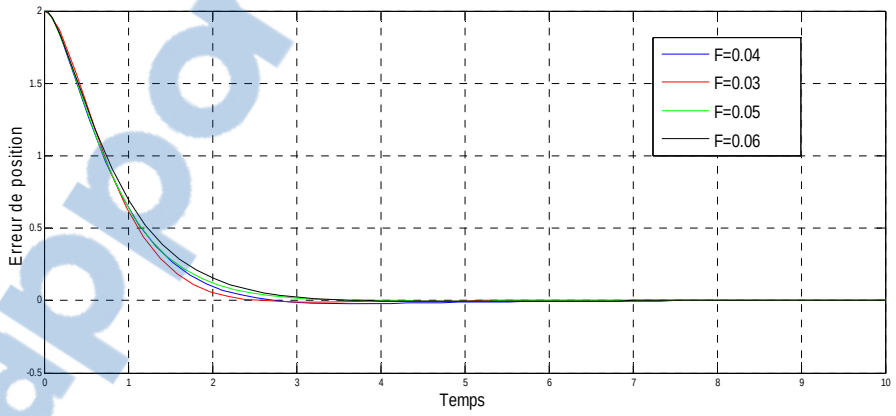


Figure 4.28 : Effet du paramètre facteur de clonage (F) sur l'erreur de position

De ces résultats on peut tirer l'impact du facteur multiplicateur contrôlant le nombre de clones pour obtenir des meilleures valeurs optimales de K_p, K_i, K_d qui amène à un état robuste et performant du système commandé.

7.4.2 L'impact du paramètre nombre des anticorps à cloner sur les performances du système

Après avoir fixé les paramètres selon la Table 4.8 et donné plusieurs valeurs au paramètre *nc* (nombre de clone), les résultats d'erreur de positions obtenues présentées dans (Figure 4.29) explique clairement l'impact important du choix des paramètres pour une situation robuste du système. (Figure 4.30) montre l'instabilité des valeurs de K_p, K_i, K_d .

Taille de population Nombre de génération	100 100
Facteur multiplicateur de clonage	0.05
Nombre des individus	3

Table 4.8 : Valeur des paramètres de l'AIA pour des valeurs de *nc* différentes

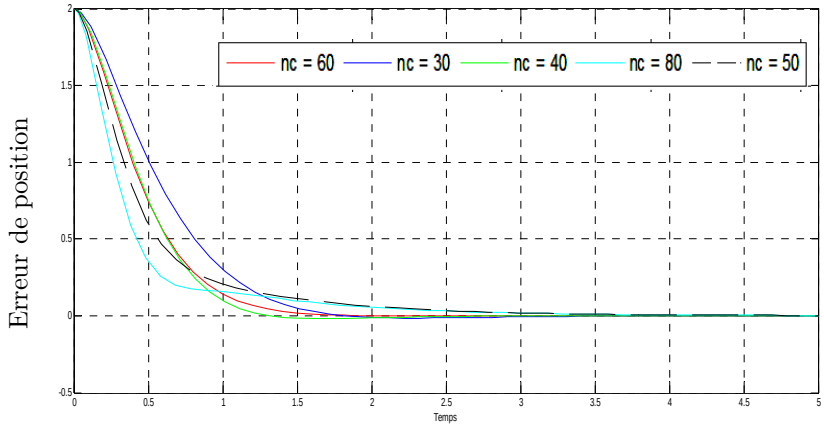


Figure 4.29 : Effet du paramètre nombre de clone (*nc*) sur l'erreur de position

Nombre de clone *nc*	***K_p***		***K_i***		***K_d***	
	$K_{P\ max}$	$K_{P\ min}$	$K_{i\ max}$	$K_{i\ min}$	$K_{d\ max}$	$K_{d\ min}$
30	17.0205	23.8324	39.6394	44.4836	0.3273	1.1925
40	17.8502	24.9164	33.7741	46.8685	0.8479	3.4968
50	21.0709	36.6443	43.0656	49.3616	0.0740	6.4892
60	22.3287	25.5894	35.4592	48.1391	0.9115	3.3826
80	14.1096	31.3713	36.5674	27.1839	0	2.7794

Table 4.9 : Effet du paramètre nombre de clone (*nc*) sur K_p,K_i,K_d

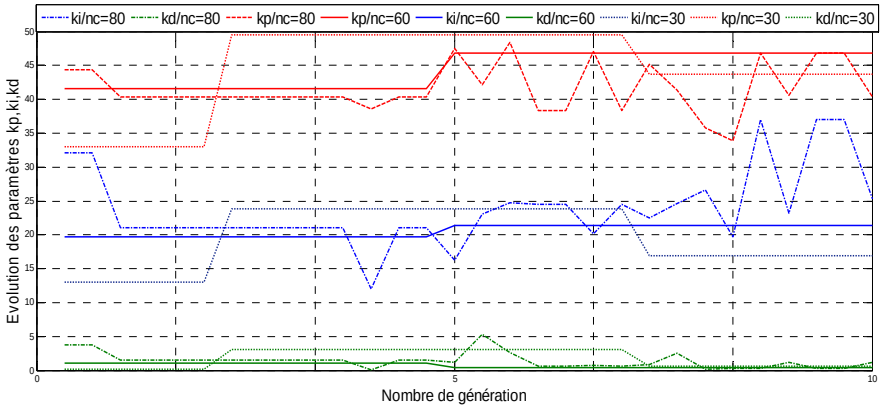


Figure 4.30 : Évolution des gains du PID selon le nombre de clone

Les résultats obtenus expliquent clairement l'impact du facteur multiplicateur contrôlant le nombre de clones ainsi nombre d'anticorps à cloner (nc) pour obtenir des meilleures valeurs optimales de K_p, K_i, K_d qui amène à un état robuste et performant du système commandé.

De cette étude nous somme devant une contrainte qui implique l'optimisation des paramètres de l'algorithme proposé d'une manière automatique afin de compléter l'optimisation des paramètres du PID dont le but d'avoir une loi de commande robuste qui donne un maximum de performance du système.

8. Conclusion

Dans ce chapitre, nous avons présenté et implémenté le principe de l'algorithme immunitaire artificiel (AIA) dans le but d'exploiter le mécanisme d'optimisation pour optimiser les gains d'un contrôleur PID sous la base de deux fonctions objectives afin de commander un système non linéaire. Cet algorithme se distingue par rapport autres modèles bio-inspirés par :

- *La structure* : Les AIA peuvent être utilisés sous forme de réseau d'éléments interconnectés ou sous forme d'un ensemble d'éléments travaillant ensemble sans notion de communication, contrairement aux réseaux de neurones qui ne fonctionnent qu'en étant interconnectés et les AG en simple individus.

- *L'adaptation* : Les AIA réagissent face à des situations qu'ils n'ont jamais rencontré jusque là et ceci grâce à la mutation de leurs éléments ce qu'on n'est pas capable de réaliser avec des réseaux de neurones.

L'algorithme immunitaire artificiel (AIA) est présenté et implémenté comme un algorithme évolutionnaire pour l'optimisation des gains d'un contrôleur PID sous la base de deux fonctions objectives afin de commander un système non linéaire.

L'approche a été appliquée sur un système masse ressort amortisseur, en donnant des résultats acceptables. Une comparaison de l'algorithme AIA a été réalisée avec les algorithmes génétiques (AG). Les résultats de cette comparaison ont confirmé l'amélioration des performances d'optimisation de l'algorithme proposé.

Nous avons constaté que le choix des paramètres de l'algorithme AIA joue un rôle très important dans la robustesse du système, ce qui implique l'optimisation automatique de ces paramètres afin d'obtenir une loi de commande robuste basée sur le contrôleur PID. Cette conclusion nous permet d'ouvrir un axe de recherche qui consiste à améliorer ce type d'algorithmes évolutionnaires.

Les résultats obtenus afférant aux travaux réalisés ont été validés par des instances scientifiques internationales, en l'occurrence :

- WCCS14 (The 2nd World Conference on Complex Systems) pour l'application de l'algorithme de l'optimisation basé sur système immunitaire artificiel pour optimiser les gains du contrôleur PID.
- IJAAC15 (Int. J. Automation and Control) ce dernier travail est l'extension améliorée du travail de WCCS 14.

Conclusion générale

Les recherches que nous avons effectuées pour établir ce travail ont montré les difficultés dominées dans la théorie de la commande des systèmes dynamiques non linéaires. Or la commande passe par l'élaboration d'un modèle mathématique du système, ce qui suppose une bonne connaissance de la dynamique du système et ses propriétés, mais en réalité elle n'est pas toujours possible à cause de la complexité et les incertitudes. La commande adaptative et la commande robuste sont des commandes largement utilisées dans la littérature pour contrôler les systèmes non linéaires.

La présente thèse a été dédiée à la résolution des limites des lois de commande classique en absence des paramètres du modèle. Dont le recours aux méthodes de commandes basées sur les techniques de l'intelligence artificielle est devenu une nécessité.

Ces techniques comportent les réseaux de neurones, la logique floue et les algorithmes évolutionnaires parmi tant d'autres.

En effet, l'adaptabilité de la logique floue, la capacité d'apprentissage et de la généralisation des réseaux de neurones sont combinées afin d'obtenir un réseau neuro-flou qui représente un outil intelligent puissant dans le domaine de la commande des systèmes non linéaires.

En premier lieu, nous avons essayé de développer un algorithme d'apprentissage en ligne en utilisant le réseau de neurone STFIS qui est caractérisé par la qualité d'adaptation et d'approximation dont le but est de surpasser les algorithmes d'apprentissage hors ligne et réduire le nombre des couches se qui réduisent le temps de calcul pour la commande des systèmes non linéaires.

Notant que les réseaux de neurones RBF par leurs qualités d'adaptation et d'approximation sont un outil performant pour estimer les paramètres du modèle du système commandé. Ces performances dépendent essentiellement de la qualité de l'algorithme d'apprentissage utilisé.

Dans la deuxième partie nous avons fait appel à un algorithme d'apprentissage en ligne. Dans ce contexte, nous avons présenté une architecture intelligente adaptative et séquentielle SAFIS basée sur

l'algorithme GAP-EKF qui est un algorithme en ligne fondé sur deux stratégies, une stratégie GAP (Growing And Pruning) et un filtre de Kalman étendu. D'où l'apprentissage du réseau commence avec 0 neurones et il ne nécessite pas la disponibilité de la totalité de la base d'apprentissage. A chaque couple entrée/sortie présenté, l'algorithme décide d'ajouter un nouveau neurone selon un critère de distance, si ce critère n'est pas vérifié, les paramètres du réseau sont modifiés par un filtre de Kalman étendu. Alors l'architecture SAFIS a été utilisée pour concevoir une approche de commande intelligente computed torque dont le but est :

- d'estimer en ligne les paramètres du modèle dynamique et compenser les incertitudes et les variations du modèle,
- d'injecter les paramètres estimés dans la commande computed torque commande pour la commande du robot manipulateur SCARA 3 ddl.

Les résultats de simulation ont montré un comportement satisfaisant pour la loi de commande couple calculée en présence de bruits et incertitudes pour commander un robot manipulateur. D'un autre coté, nous avons constaté que les paramètres de l'algorithme SAFIS jouent un rôle important pour la robustesse de la loi de commande. Après une étude, nous avons conclu que le choix de ces paramètres doit être optimisé d'une manière automatique ce qui fait appel à des techniques de l'intelligence artificielle.

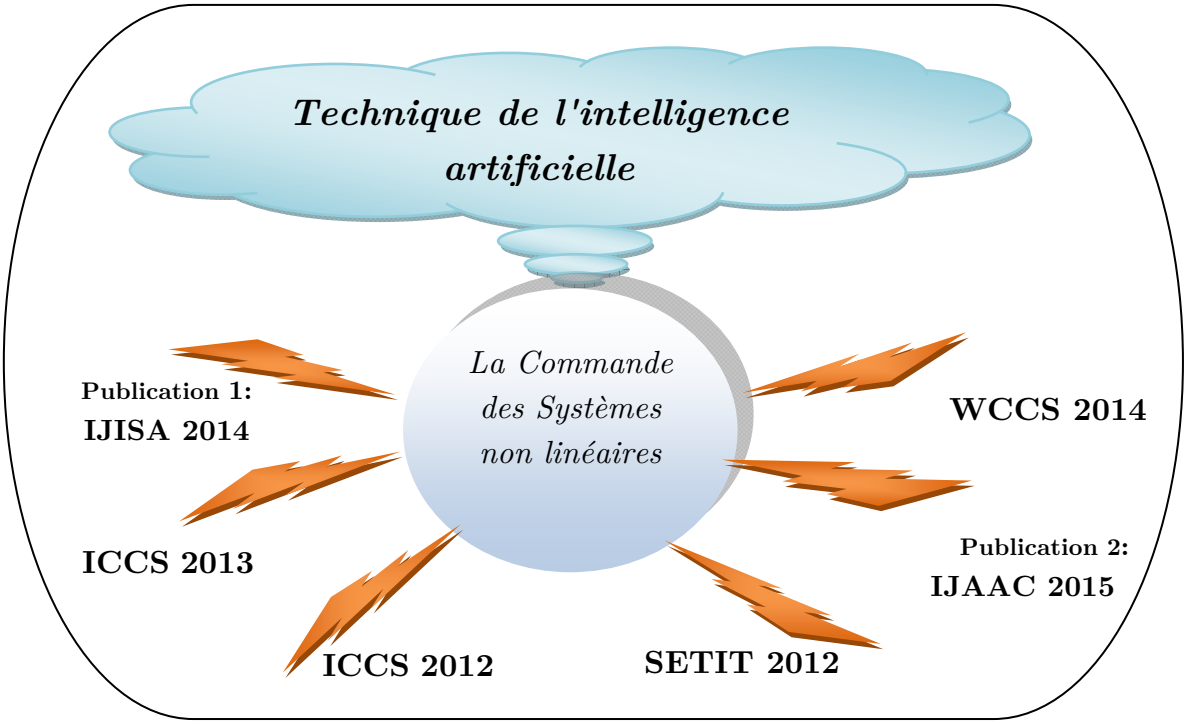
En effet, les contrôleurs PID représentent une solution facile et efficace devant certains types de problèmes de contrôle des systèmes non linéaires. Les trois principales composantes du contrôleur PID impliquées "Proportionnelle, Intégrale et Dérivée" sont empiriquement ajustées. La procédure de réglage de paramètres PID est considérée comme un problème d'optimisation.

Cette thèse a présenté un algorithme évolutionnaire inspiré des mécanismes naturels. L'algorithme immunitaire artificiel est un outil puissant d'optimisation pour optimiser les paramètres du contrôleur PID afin de commander des systèmes non linéaire. Pour valider cette approche intelligente deux fonctions objectives ont été utilisées afin d'optimiser les paramètres du PID via l'algorithme immunitaire artificiel. Les résultats obtenus on été comparés avec un algorithme évolutionnaire de base (les algorithmes génétiques) afin de prouver les performances de l'AIA.

On peut conclure après ce travail, qui a été basé sur le développement et l'application des techniques de l'intelligence artificielle afin de montrer que :

- les réseaux neuro-flous comme STFIS présente un outil puissant dans la commande des systèmes non linéaires.
- L'architecture SAFIS basée sur l'algorithme GAP-EKF est utile pour les applications nécessitant des estimations en temps réel. Il suffit juste de fixer ses paramètres
- L'algorithme évolutionnaire AIA est consacré à l'optimisation des paramètres du PID et qui représente une solution intelligente précise.
- Le bon choix des paramètres des algorithmes SAFIS, AIA font l'objet de la robustesse des lois basées sur derniers.

Nous pouvons résumer les résultats obtenus dans cette thèse par les productions suivantes :



Bibliographie

ABRAHAM, A. (2001). *Neuro fuzzy systems: State of the art modeling techniques.* Master's thesis, School of Computing & Information Technology, University Monash, Australia

Ak, A. G. et CANSEVER, G. (2006). *Fuzzy sliding mode controlller with RBF networks for robotic manipulator trajectory tracking.* Lecture notes in control and information sciences (Intelligent Control and Automation), 144 :527_532.

BARALDI ANDREA, PALMA BLONDA, ALFREDO PETROSINO, (1998), *Fuzzy Neural Networks for Pattern Recognition*, pp 35-83

ANGELES.J, (2002), Fundamentals of Robotic Mechanical Systems: Theory, Methods, And Algorithms, 2ième Ed, Springer

MAGÀLY DE PAULA CANUTO, (2001).*Combining Neural Networks and Fuzzy Logic for Applications in Character Recognition.* A thesis submitted to the University of Kent at Canterbury for the degree of Doctor Philosophy in the subject of Electronic Engineering.

ALEXENDER TARAKANOV et DIPANKAR DASGUPTA. (2000), *A formal model for an artificial immune system.* BioSystems, 55(1-3) :151.158.

ASTRÖM.K.J, ET B. WITTENMARK, (1989), *Adaptive Control*, Addisson Wesley.

BERENJI.H.R, P. KHEDKAR, (1992), *Learning and Tuning Fuzzy Logic Controllers through Reinforcements*, IEEE Transaction on Neural Networks.

BRUNET.M.L, (1996), *Identification et Contrôle de processus par Réseaux Neuro-Flous*, thèse de doctorat, Evry,

BERBAOUI, B., BENACHAIBA, C.RAHLI,M AND TEDJINI.H, (2011). An efficient algorithm to tuning PI-controller parameters for shunt active power filter using ant colony optimization. Przeglad elektrotechniczny (Electrical Review), 6 pp:140-145.

CHELOUAH.R, (2000), *Adaptation aux problèmes à variables continues de plusieurs métaheuristiques d'optimisation combinatoire : la méthode tabou, les algorithmes génétiques et les méthodes hybrides. Application en contrôle non destructif*, thèse, Université de Cergy Pontoise.

CHIHA.I, N. LIOUANE AND P.BORNE (2012), *Tuning PID Controller Using multiobjectives, Ant colony Optimization* , Applied Computational intelligence and Soft Computing Journal, Volume 2

CHO.K.B, B.H.WANG, (1996), *Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction,* Fuzzy Sets and Systems 83 325–339.

COIFFET P, (1992), *La robotique : principes et applications*, 3e édition revue et complétée", *Edition Hermès*, Paris

CRAIG J.J, (1985), *Adaptive control of mechanical manipulators*, Adisson-Wesley Publishing Company, Inc, New York

CREPINSEK, M., LIU, S.-H., et MERNIK, M. (2013). *Exploration and exploitation in evolutionary algorithms : A survey.* ACM Computing Surveys, 45(3) :35 :1_33.

CYBENKO G, (1989), *Approximation by superposition of sigmoidale functions*, Mathematics of of Control Signal and Systems, vol.2, pp: 303-314

DASGUPTA.D, (2000),*artificial immune systems and their application,* pringer-verglas.

DE CASTRO.L.N, J. TIMMIS, *(2002), In Artificial Neural Networks in Pattern Recognition Artificial Immune Systems: A Novel Paradigm to Pattern Recognition, University* of Paisley, UK, pp. 67-84.

DE CASTRO.L.N., F. J. VON ZUBEN, (2003), *The Construction of a Boolean Competitive Neural Network Using Ideas from Immunology,* Neurocomputing, 50C, pp. 51-8.

DASGUPTA.D, (1999), *Artificial Immune Systems and Their Application.* Springer-Verglas.

DISHA SHARMA, (2011) *Designing and Modeling Fuzzy Control Systems* International Journal of Computer Applications (0975–8887) Volume 16 – No.1, pp :46-49

DEREK J. SMITH, STEPHANIE FORREST, RON R. HIGHTOWER, et ALAN S. PERELSON. (1997) *Deriving shape space parameters from immunological data for a model of cross-reactive memory.* Working Papers 97-03-017, Santa Fe Institute

DOMBRE E. et W. KHALIL, (1988), *Modélisation et commande des robots*, *Edition Hermès*, Paris

DREYFUS. G, (2002), *Réseaux de neurones : méthodologies et applications*, Ed Eyrolles.

DREYFUS.G,J.M. (2008),ARTINEZ, M. SAMUELIDES, M. GORDON, *Apprentissage statique : réseaux de neurones, cartes topologiques, machine a vecteur supports*, Eyrolles, Paris,.

EIBEN A.E, AND SMITH J.E. (2003), *Introduction to Evolutionary Computing*, *Springer*.

EMMA, J. TIMMIS, (2008). *Investigating the evolution and stability of a resource limited artificial immune system.* In A.S.Wu, editor, Special Workshop on Artificial Immune Systems, Gentic and Evolutionay Computtion Conference (GECCO), Workshop Program, pp: 40.41, Las Vegas, Nevada, U.S.A.

FORREST.D & DASGUPTA.S, (1996), *Novelty Detection in Time Series Data Using Ideas From Immunology*, Proc. of the ISCA'96.

FUNAHASHI K, (1989), *On the approximate realization of continuous mappings by neural networks*, Neural Networks, vol. 2, pp: 183-192.

GIRIRAJKUMAR, S., JAYARAJ, D., AND KISHAN, A, (2010). *PSO based tuning of a PID controller for a high performance drilling machine.* International Journal of Computer Applications, 1(19) :pp 12-18.

GOZDE, H., TAPLAMACIOGLU, M. C., AND KOCAARSLAN, I. (2010). *Applications of artificial bees colony algorithm in an automatic voltage regulator (AVR) system.* International Journal on Technical and Physical Problems of Engineering (IJTPE), 2(3) :pp 88-92.

GLORENNEC.P.Y, (1999), *Algorithmes d'Apprentissage pour Systèmes d'inférence Floue*, Edition Hermès.

GARRETT.S, (2004). *Parameter-Free Adaptive Clonal Selection.* Congress on Evolutionary Computation (CEC), 19-23 June, Volume: 1, pp: 1052- 1058.

GONZALEZ .L et Cannady.J, (2004). A Self-Adaptive Negative Selection Approach for Anomaly Detection. Congress on Evolutionary Computation (CEC), volume 2, Issue , 19-23 June, Page(s): 1561 - 1568.

HAGAN, T. M., DEMUTH, H, et BEALE, M. (1995), *Neural network design*. Ed PWS publishing company.

HAI-JUN RONG, N. SUNDARARAJAN, GUANG-BIN HUANG, P. SARATCHANDRAN. (2006), *Sequential Adaptive Fuzzy Inference System (SAFIS) for nonlinear system identification and prediction*, IEEE Trans, Nanyang Technological University, Singapore

HAYKIN.S, Neuronal Networks, (1999), *A comprehensive foundation*, Prentice Hall International, Upper Saddle River, New Jersey, USA,

HORNIK, K. (1994). *Approximation capabilities of multilayer feedforward networks*. Neural Networks, vol 4, pp: 251-257.

HUNT K, SBARNARO D, (1991), *Neural networks for non-linear internal model control*, IEEE Proceedings-D, vol.138 n°5, pp: 431-438.

HUANG.G.-B., P. SARATCHANDRAN, N. SUNDARARAJAN, (2004) ,*An efficient sequential learning algorithm for growing and pruning RBF (GAP–RBF) networks*, IEEE Trans. Systems Man Cybernet.—Part B: Cybernet. 34 (6) 2284–2292.

HUANG G.-B, P. (2005) Saratchandran, N. Sundararajan, *A generalized growing and pruning RBF (GGAP–RBF) neural network for function approximation*, IEEE Trans. Neural Networks 16 (1) 57–67.

H. Wang (2008), *Artificial immune system based image pattern recognition in Energy efficient wireless multimedia sensor networks*. Military Communications Conference, MILCOM. IEEE 978-1-4244-2676-8

IRWIN.G, K. WARWICH, K. HUNT, (1995), *Neuronal Networks applications in control*, The Institution of Electrical, London,

ISE ,M.ZELMAT. (1992) *Commande adaptative*. OP, 2001, Edition : 20.47.4440, I.S.B.N :9961.0.0510.4

JACQUES.J, et E. SLOTINE, (1991), *Applied Nonlinear Control*, Prentice Hall. ISE M.ZELMAT. COMMANDE ADAPTATIVE. OP, 2001, Edition: 20.47.4440, I.S.B.N :9961.0.0510.4

JANG.R, (1992), *Neuro-fuzzy modeling :architecture, analyses and applications*, PhD thesis ,Dep. Of electrical Engineering and computer Science, University of California, Berkeley.

JANG J.-S.R, C.-T. SUN, E. MIZUTANI, (1997), *Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence*, Prentice-Hall, NewYork

JERNE.N, (1974), *towards a network theory of the immune system.* Annals of Immunology, vol. 125, pp.373–389

JOHNSON, M. A. et MORADI, M. H. (2005). *PID Controllers New Identification and Design Methods.* Springer

JONES.A et J. RABELO. (1998), Survey of job shop scheduling techniques,

NEAL MARK et JON TIMMIS. (2000), *Investigating the evolution and stability of a resource limited artificial immune system.* In A.S.Wu, editor, Special Workshop on Artificial Immune Systems, Gentic and Evolutionay Computtion Conference (GECCO) 200*0*, Workshop Program, pages 40.41, Las Vegas, Nevada, U.S.A.

JORDAN.M.I, D. RUMELHART (1991), *Internal word models and supervised learning* , *Proceeding* of the 8th Inernational Workshop on Machine learning, pp 70- 74.

JUANG.C, C.-T. LIN, (2002), *An on-line self-constructing neural fuzzy inference network and its applications*, IEEE Trans. Fuzzy Systems 10 (2) 144–154

JOG, P., SUH, J., et GUCHT, G. (1991). *Parallel genetic algorithms applied to the traveling salesman problem.* SIAM Journal on Optimization, 1(4) :515_529.

KARTALOPOULOS, S. V, (2004), *Understanding Neural Networks and Fuzzy Logic Basic Concepts and Applications.* IEEE Press - PHI.

KHALIL W, DOMBRE E, (2002), *Modelisation, identification and control of robots*, Hermes Penton Science, London, ISBN 1-90399-613-9, pp 480 .

KHALIL.S, (2003). *Inverted pendulum analysis, design and implementation.* Technical report, The Instrumentation and Control Lab at the Institute of Industrial Electronics Engineering, Pakistan.

KIM, J. S. (2008). *Auto tuning PID controller based on improved genetic algorithm for reverse osmosis plant.* In Proceedings of the World Academy of Science and Engineering

KLIR, G.J. et B. YUAN (1994). *Fuzzy sets and fuzzy logic: theory and applications.* Prentice-Hall. Inc. Upper Saddle River, NJ, USA.

KOSKO. B, (1993), *Fuzzy Systems as Universal Approximators*, *IEEE Transactions on Computers*.

KOZA, J.R, BENNETT, F.H., ANDRE.D, et KEANE, M.A. (1999), G*enetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann.

KOZA.J. R.. (1994); *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Massachussetts.

LANDAU. I. D, R. LOZANO ET M. M'SAAD, (1998), *Adaptive Control*, Springer.

LEANDRO N. DE CASTRO et JONATHAN TIMMIS. (2002), *Artificial Immune Systems : A New Computational Intelligence Approach*. Springer-Verlag.

LEONARDO W. de Oliveira, (2014). Artificial *Immune Systems applied to the reconfiguration of electrical power distribution networks for energy loss minimization.* International Journal of Electrical Power & Energy Systems, Volume 56, March 2014, Pages 64–74. doi:10.1016/j.ijepes.2013.11.008

LEE.C.W, Y.C. Shin, (2003), *Construction of fuzzy systems using least-squares method and genetic algorithm*, Fuzzy Sets and Systems 137 297–323

LEE C.W, McGinnity, G. Prasad, (2005)*, An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network,* Fuzzy Sets and Systems 150 (2), 211–243

LENG.G, T.M. MCGINNITY, G. PRASAD, (2005) , *An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network*, Fuzzy Sets and Systems 150 (2) 211–243.

LEWIS F.L, C.T. ABDELLAH et D. M. DAWSON, (1993), *Control of robot manipulators*, New York, Macmillan.

LIN,C.-T et LEE, G (1996), *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems* .Ed. Prentice Hall.

LUH.G et Liu.W, (2004). *Reactive Immune Network Based Mobile Robot Navigation.* Proceeding of the Third Conference ICARIS, pages 119-132, Springer.

MAARAF. H,(2002), *Notion de base de la théorie de flou*, Cours de la théorie de flou, université d'Evry Val d'Essonne

MAHABIR, C., F. HICKS et F.R. FAYEK (2006). *Neuro-Fuzzy river ice breakup forecasting System.* Cold régions science and technology, 46, 100-112.

MAMDANI, E. H. AND ASSILIAN, S. (1975). *An experiment in linguistic synthesis with a fuzzy logic controller.* International Journal of Man-machine Studies, pp 7:113.

MCCULLOCH, W. S. et PITTS, W. (1943). *A logical calculus of the ideas immanent in nervous activity.* Bulletin of Mathematical Biophysics, 5 :115_133

MICHAIL ZAK. (2000), *Physical model of immune inspired computing.* Inf. Sci. Inf. Comput. Sci., 129(1-4) :61.79.

MICHALEWICZ.Z. (1996), *Genetic Algorithms + Data Structures = Evolution Programs.* Springer Verlag, New-York,1st-3rd edition.

MITCHELL, M. (1998). An Introduction to Genetic Algorithms. Prentice-Hall.

HOSSEINI SHEIKH, MARYAM ZEKRI. (2011), *Review of Medical Image Classification using the Adaptive Neuro-Fuzzy Inference System.* Journal of Medical Signals & Sensors.

MOODY, J. AND DARKEN, C, (1995). *Fast learning in network of locally-tuned processing units.* Neural Computation, 1:606_623.

NAGARAJ, B. et VIJAYAKUMAR, P. (2011), *A comparative study of pid controller tuning using ga, ep, pso and aco.* Journal of Automation, Mobile Robotics & Intelligent Systems, 5(2) :42-48.

NARENDRA.S, PARTHASARATHY. K, (1990), Identification and control of dynamical system using neural netorks, IEEE Transactions on neural networks, vol 1, no 1,pp 4-27

NASRAOUI O, C. Cardona-Uribe, and C. Rojas-Coronel, *Tecno-streams: Tracking evolving clusters in noisy data streams with a scalable immune system learning model.* In IEEE International Conference on Data Mining, Melbourne, Florida,Nov. 2003.

OPEYEMI OBANIJESU, EMUOYIBOFARHE O. JUSTICE, (2012), *Development of Neuro-fuzzy System for Early Prediction of Heart Attack* . I.J. Information Technology and Computer Science, 9, pp 22-28

PATNAIK SRIKANTA, YEON-MO YANG , (2012), *Soft Computing Techniques in Vision Science.* Studies in Computational Intelligence 395, Springer, ISBN 978-3- 642-25506-9

PASCHOS, V. T. (2005). *Optimisation combinatoire : concepts fondamentaux.* Hermes science publication.

PANIMADAI Ramaswamy. (2007),*Optimal Control of Class of Non-linear Plants Using Artificial Immune Systems: Application of the Clonal Selection Algorithm.* Institute of Electrical and Electronics Engineers (IEEE) 10.1109/ISIC.2007.4450893

PERELSON et G. OSTER. (1979), *Theoretical studies of clonal selection : minimal antibody repertoire size and reliability of self-non-self discrimination.* J. Theor. Biol, 81 :645.670.

POGGIO T. GIROSI F., (1990), *Networks for Approximation and Learning* , Proceedings of the IEEE, vol. 78, n° 9, p. 1481-1497,.

MATZINGER POLLY. *The danger model in its historical context.* Scandinavian journal of immunology, 54 :4.9, 2001.

RACOCEANU, D. (2006). *Contribution à la surveillance des Systèmes de Production en utilisant les Techniques de l'Intelligence Artificielle*, Habilitation à Diriger des Recherches, Université de Franche Comté, Besançon.. 84, 85, 86.

RUIJUN, D. (2009). *Differential evolution versus particle swarm optimization for PID controller design.* In Fifth International Conference on Natural Computation, ICNC '09.

SAHRAOUI.M et KHELFI.M.F et SALEM.M,(2014) *Sequential Adaptive Fuzzy Inference System Based Intelligent Control of Robot Manipulators,* International Journal of Intelligent Systems and Applications(IJISA) ISSN online: 2074-9058 Volume 6, Number 11, October 2014

SAHRAOUI.M et KHELFI.M.F et SALEM.M,(2015) *Application of artificial immune algorithm-based optimisation in tuning a PID controller for nonlinear systems,* Int. J. Automation and Control, Vol. 9, No. 3, 2015

SAHRAOUI.M et KHELFI.M.F et SALEM.M,(2014) Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on Year: 2012 Pages: 912 - 916, DOI: 10.1109/SETIT.2012.6482036 IEEE Conference Publications

SALEM.M (2014), *Approches de l'intelligence artificielle pour la commande robuste des systèmes non linéaires,* thèse PHD en science, Département d'Informatique, université d'oran1.

SALEM et M.F KHELFI (2012), *Application of Biogeography based optimization in tuning a PID controller for nonlinear systems.* In Proc *IEEE* International Conference on Complex Systems (ICCS), Nov 05-06 2012, Agadir Morocco.

SECKER Andrew, Alex Freitas et Jon Timmis. (2003),*Un système immunitaire artificiel pour la classification des E-mail.* In A Scime, editor, Web Mining: applications and techniques, pages 145-168.

SIMON, D. (2008). *Biogeography-based optimization.* IEEE Transactions on volutionary Computation, 12(6) :702-713

SOLIHIN.G ; L.F. TACK AND M.L. KEAN (2011), '*Tuning of PID Controller Using Particle Swarm Optimization (PSO)*'. Proceeding of the International Conference on Advanced Science, Engineering and Information Technology, 2011 ,Malaysia.

SLOTINE, J. J. E. AND LI, W. (1991). *Applied Nonlinear Control.* Prentice-Hall

SOLIHIN, TACK.L, AND KEAN, M. L. (2011). *Tuning of PID controller using particle swarm optimization (PSO).* In Proceeding of the International Conference on Advanced Science, Engineering and Information Technology 2011, Bangi-Putrajaya, Malaysia.

STEVEN A. HOFMEYR et AND STEPHANIE FORREST. (2000), *Architecture for an artificial immune system. Evolutionary Computation*, 8(4) :443.473.

SUNDARESWARAN, K. (2008). D.C. *motor speed controller design through a colony of honey bees.* In TENCON (IEEE Region) conference, pages 1-6.

TAKAGI, T. AND SUGENO, M. (1985). *Fuzzy identification of systems and its applications to modeling and control.* IEEE Trans. Systems, 15 :116_132.

TAKAGI.T, M. SUGENO, (1983), *Derivation of fuzzy control rules from human operator's control actions*, in: Proceedings of the IFAC Symposium On Fuzzy Information, Knowledge Representation and Decision Analysis, , pp. 55–60.

TENG.T.K, J. S. SHIEH AND C. S. CHEN (2003),' *Genetic algorithms applied in online auto tuning PID parameters of a liquid-level control system*'. Journal of

Transaction of the Institute of Measurement and control vol 25, 5 (2003), pp.433~450

TIMMIS & T. KNIGHT & L.N. DE CASTRO & E.HART, (2003).«*An overview of Artificial immune Systems*», Natural computation series, pages51-86, Springer.

VERHOEVEN, M. et AARTS, E. (1995). *Parallel local search.* Journal of Heuristics, 1(1) pp:43-65.

WANG.L.X., (1994), *Adaptive Fuzzy Systems and Control: Design and Stability Analysis,* Prentice-Hall, Englewood Clifs, NJ.

WANG.M, N. K. LIU JAMES (2008), *Fuzzy Logic based Real-time Robot Navigation in Unknown Environment with Dead Ends*, Robotics and Autonomous Systems, vol. 56, pp.625-643,.

WANG.L, J. YEN, (1999), *Extracting fuzzy rules for system modelling using a hybrid of genetic algorithm and Kalman filter*, Fuzzy Sets and Systems 101 353–362.

WEISE Thomas.Global, (2009), *Optimization Algorithms, theory and Application*– An electronic book available for download at http://www.itweise. de/projects/book.pdf. 2nd edition . self-published: Germany.

WIDROW, B. et LEHR, M. A. (1990). *30 years of adaptive neural networks : Perceptrons, madeline and backpropagation.* Proceedings of the IEEE, 78(9) pp:1415-1442.

YINGWEI.L, N. SUNDARARAJAN, AND P. SARATCHANDRAN, (1998), *Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm,*" IEEE Trans. Neural Networks, vol. 9, pp. 308–318.

YIN.F, WANG.J, AND GUO.C, (2004). *Design of PID controllers using genetic algorithms approach for low damping slow response plants.* Advances in Neural Networks, 3174 :219_220.

YOUXIN. L, AND XIAOYI. C, (2010). *Tuning PID control parameters on hydraulic servo control system based on differential evolution algorithm.* In 2nd International Conference on Advanced Computer Control (ICACC).

YU Y. et Hou.C, (2004). *A Clonal Selection Algorithm By Using Learning Operator.* Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29 Aug.

ZADEH L.A (1972), *A relational for fuzzy control*, journal of systems, Measurement and control, pp. 3-4

ZADEH, L. (1965). *Information and control*. Fuzzy Sets, 8 :338_353

ZAIN B. A. M, TOKHI.M, AND TOHA.S, (2009). *PID based control of a single-link flexible manipulator in vertical motion with genetic optimisation*. In Third UK Sim European Symposium on Computer Modeling and Simulation. Athens, Greece.

ZEMALACHE.K, H. MAAREF, 2008), *Intelligent control for a drone by self-tunable fuzzy inference system*, 6th international multi-conference on systems, signals and devises, Djerba, Tunisia, 23-26 March.

ZUO X. and Li.S, 2003. *The Chaos Artificial Immune Algorithm and Its Application to RBF Neuro-Fuzzy Controller Design*. IEEE International Conference on Systems, Man and Cybernetics, Volume 3, Issue,5-8 Oct, Page(s): 2809 - 2814.

Annexes

Systèmes non linéaires utilisés dans ce travail :

1. Le système masse ressort amortisseur

Le système masse ressort amortisseur est un système non linéaire décrit par l'équation différentielle de deuxième ordre (A.1).(KHALIL, 2013)

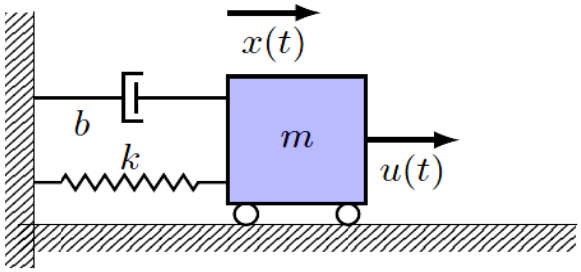


Figure A.1 : Système masse ressort amortisseur

$$\ddot{x} = \frac{b}{m}\dot{x} + \frac{k}{m}x + \frac{1}{m}x^3 - \frac{1}{m}u \tag{A.1}$$

où m est la masse, $u(t)$ est la force exercé, k est la constante de rigidité du ressort et b : le coefficient d'amortisseur. Avec $m = 1kg$, $b = 10\ Ns/m$, $k = 20\ N/m$.

2. Pendule inversé

Le pendule inversé est un système formé d'une tige montée en inverse sur un chariot mobile ; il est un système non linéaire de second ordre dont le modèle est donnée par l'équation (A.2) avec $x(t) = [\theta, \dot{\theta}]$ vecteur d'état, $d(t)$ est le vecteur des perturbations externes et $u(t)$ le vecteur de commande (KHALIL, 2003).

Le système pendule inversé à modéliser est représenté par (Figure A.2).

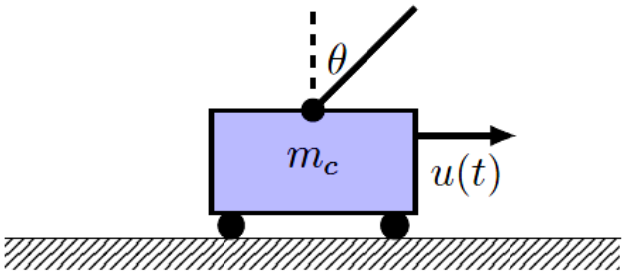


Figure A.2 pendule inversé simple

$$\dot{\theta} = f(x) + g(x)u(t) + d(t) \tag{A.2}$$

où

$$\dot{\theta} = \frac{g\sin\dot{\theta} - ml\dot{\theta}^2\cos\theta/(m_c+m)}{l(4/3 - m\cos^2\theta/(m_c+m))} + \frac{\cos x_1/(m_c+m)}{l(4/3 - m\cos^2\theta/(m_c+m))}u(t) + d(t) \tag{A.3}$$

$\dot{\theta}$ et θ sont la positions et vitesse angulaires de la tige respectivement, g : la gravité, m la masse de la tige et m_c la masse du chariot ; l la moitié de la longueur de la tige et $d(t)$ représente le bruit (KHALIL, 2003) avec $m = 0.1Kg$, $m_c = 1Kg$, $g = 9.8m/s^2$ *et* $l = 0.5m$. (LEWIS, 1993)

3. Robots manipulateurs

Un robot est un système mécanique multi-articulé mû par des actionneurs et destiné à effectuer une grande variété de tâches (Figure A.3).

Le modèle dynamique d'un robot manipulateur de n degrés de liberté est donné par l'équation (A.4) (LEWIS, 1993) :

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) + F(\dot{q}) = \Gamma \tag{A.4}$$

où $q, \dot{q}, \ddot{q}$ sont les vecteurs des positions, des vitesses et accélérations angulaires respectives, $\boldsymbol{\Gamma}$ est le vecteur des couples. $M(q)$ est la matrice d'inertie, elle est symétrique et définie positive, $C(q,\dot{q})$ est la matrice de Coriolis/Centrifuges et $G(q)$ est le vecteur de gravité. F($\dot{q}$) est le vecteur de frottement.

Dans le cas d'un robot SCARA à 3 ddl, les matrices du modèle sont données par l'équation (A.5) (Ak, 2006) :

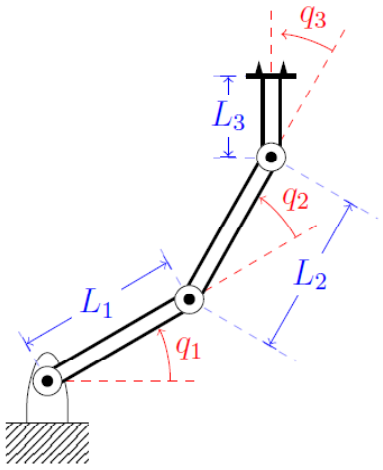


Figure **A.3** robot manipulateur

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) + F(\dot{q}) = \Gamma \tag{A.5}$$

où :

$$M(q) = \begin{cases} M_{11} = 2.1240 + 1.44\cos(q_2) \\ M_{12} = -0.4907 - 0.72\cos(q_2) \\ M_{13} = 0 \\ M_{21} = -0.4907 - 0.72\cos(q_2) \\ M_{22} = 0.4907, M_{23} = 0 \\ M_{31} = M_{32} = M_{33} = 0 \end{cases} \tag{A.6}$$

$$C(q,\dot{q}) = \begin{cases} C_{11} = 1.44.\dot{q}_2.\sin(q_2) \\ C_{12} = -0.72.\dot{q}_2\sin(q_2) \\ C_{22} = -0.72.\dot{q}_2\sin(q_2) \\ C_{13} = C_{21} = C_{23} = 0 \\ C_{31} = C_{32} = C_{33} = 0 \\ M_{31} = M_{32} = M_{33} = 0 \end{cases} \tag{A.7}$$

$$G(q) = \begin{cases} G_1 = 0 \\ G_2 = 0 \\ G_3 = -4.9 \end{cases} \tag{A.8}$$

$$F(\dot{q}) = \begin{cases} F_1 = 12\dot{q}_1 + 0.02sign(\dot{q}_1) + 3\sin(3t) \\ F_2 = 12\dot{q}_2 + 0.02sign(\dot{q}_2) + 3\sin(3t) \\ F_3 = 12\dot{q}_3 + 0.02sign(\dot{q}_3) + 3\sin(3t) \end{cases} \tag{A.9}$$

Les bruits externes sont données par :

$$\Gamma_d = \begin{cases} 5\sin(2t) \\ 5\sin(2t) \\ 5\sin(2t) \end{cases} \tag{A.10}$$