MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

----------o----------

UNIVERSITÉ FERHAT ABBAS

SÉTIF

MÉMOIRE

Présenté par

TITEL FAOUZI

Pour obtenir le titre de **Magister**

de l'Institut d'Électronique

Option

CONTROLE

CONCEPTION DE CONTRÔLEUR À LOGIQUE FLOUE PAR LES ALGORITHMES GÉNÉTIQUES

Date de soutenance : 28/06/1998

Devant le jury composé de :

Président :	Mr. K. BENMAHAMMED	Prof. Université de Sétif
Rapporteur :	Mr. K. BELARBI	M.C. Université de Constantine
Examinateurs :	Mr. A. BENNIA	M.C. Université de Constantine
	Mr. D. SLIMANI	M.C. Université de Sétif
	Mr. A. BARTIL	C.C. Université de Sétif

Année 1997/98

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

----------o----------

UNIVERSITÉ FERHAT ABBAS

SÉTIF

MÉMOIRE

Présenté par

TITEL FAOUZI

Pour obtenir le titre de **Magister**

de l'Institut d'Électronique

Option

CONTROLE

CONCEPTION DE CONTRÔLEUR À LOGIQUE FLOUE PAR LES ALGORITHMES GÉNÉTIQUES

Date de soutenance : 28/06/1998

Devant le jury composé de :

Président :	Mr. K. BENMAHAMMED	Prof. Université de Sétif
Rapporteur :	Mr. K. BELARBI	M.C. Université de Constantine
Examinateurs :	Mr. A. BENNIA	M.C. Université de Constantine
	Mr. D. SLIMANI	M.C. Université de Sétif
	Mr. A. BARTIL	C.C. Université de Sétif

Année 1997/98

Dédicaces

Je dédie ce modeste travail à mes très chers parents qui m'ont encouragé tout le long de mon cycle d'étude et m'ont guidé vers la réussite.

Je le dédie également à mes frères et soeurs .

Ainsi qu'à Raouf , Sandra , Rima , Marwan et Marwa .

Remerciement

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

Je remercie Mr. K.Benmahammed ,professeur à l'université de Setif de l'honneur qu'il m'a fait en présidant le jury de cette thèse .

Je tiens à remercier Mrs: A.Bennia, maître de conférences à l'université de Constantine, D. Slimani, maître de conférences à l'université de Sétif et A. Bartil, chargé de cours à l'université de Sétif, pour l'intérêt qu'ils ont accordé à mon travail en acceptant d'examiner cette thèse .

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

Résumé

Une nouvelle méthodologie de conception des contrôleurs à logique floue été développée.
Le contrôleur est projeté sur un réseau interconnecté multicouche, les poids et les entrées du réseau sont numériques. Il s'agit alors de concevoir simultanément les paramètres des fonctions d'appartenance triangulaires symétriques et l'ensemble des règles d'inférence. La solution utilise les algorithmes génétiques qui sont des procédures de recherche basées sur les mécanismes de la génétique naturelle pour trouver le contrôleur qui minimise une certaine fonction coût.
Pour étudier ses performances, cette nouvelle méthode a été appliquée pour la conception d'un contrôleur flou pour un pendule inversé .

Abstract

A new designing methodology for fuzzy logic controller is developed. The controller is implemented through a multi-layer network with crisp inputs and real valued weights. The problem is then to find simultaneously the optimal parameters of symmetric triangular membership functions and the rule set. The solution procedure uses genetic algorithms ,a recent search and optimisation procedures based on the mechanics of natural genetics, to find the controller that minimises a certain cost function.
This performance of the new design procedure is demonstrated on an application to the design of a fuzzy controller for an inverted pendulum system.

SOMMAIRE

INTRODUCTION 1

CHAPITRE II : ***LES ALGORITHMES GENETIQUES***

II.1 Introduction 5
II.2 Formulation du problème d'optimisation 5
II.3 Principe de fonctionnement des AG 6
II.4 Les mécanismes d'un AG simple 7
II.4.1 La reproduction 7
II.4.2 Le croisement 10
II.4.3 La mutation 11
II.4.4 La notion de schéma 11
II.4.4.1 Effet de la reproduction 12
II.4.4.2 Effet du croisement 13
II.4.4.3 Effet de la mutation 14
II.5 La sélection des individus d'une nouvelle génération 14
II.5.1 Sélection par descendance 15
II.5.2 Sélection par compétition 15
II.5.3 Steady state selection 15
II.5.4 Selective breeding selection 16
II.6 Description du codage 16
II.7 Description du décodage 16
II.8 Organigramme de l'AG 17
II.9 La convergence de l'AG 18
II.9.1 Critère de convergence 18
II.9.2 Taux de convergence 18
II.10 Conclusion 19

CHAPITRE III : ***LES APPROCHES DE CONCEPTION D'UN FLC***

III.1 La logique floue 21

III.1.1 Introduction 21
III.1.2 Contrôleur à logique floue 21
III.1.2.1 La fuzzification 21
III.1.2.2 Bloc de contrôle 22
III.1.2.3 La défuzzification 24
III.1.2.4 Paramètres de conception d'un FLC 24
III.2 Les approches de conception d'un FLC 25
III.2.1 L'approche connexioniste 25
III.2.1.1 Les réseaux de neurones 25
a) Définition 25
b) Architecture des réseaux de neurones 26
c) Entraînement des réseaux de neurones 27
d) L'algorithme de la rétropropagation 27
III.2.1.2 Les réseaux de neurones flous 28
a) Définition 28
b) Les différentes approches des FNN 28
c) Les architectures des FNN 29
III.2.2 L'approche directe 37
III.2.2.1 L'application des AG pour les contrôleurs flous 37
a) Définition du problème :paramètres à optimiser 38
b) Codage des paramètres 39
III.2.2.2 L'apprentissage des systèmes flous par les AG.....41
III.3 Conclusion 42

CHAPITRE IV : ***CONCEPTION D'UN CONTROLEUR FLOU PAR LES AG***

IV.1 Introduction 45
IV.2 Implémentation du contrôleur flou 45
IV.3 Description de l'algorithme d'apprentissage 49
IV.3.1 Codage des paramètres 50
IV.3.2 Mécanisme de l'AG 50
a) Reproduction 50
b) Croisement 51

c) Mutation 51
d) La sélection des individus d'une nouvelle génération 52
IV.3.3 Décodage des paramètres 52
IV.4 Interaction du module contrôleur avec l'AG 52
IV.5 Conclusion 54

CHAPITRE V : ***RESULTATS DE SIMULATION***

V.1 Introduction 57
V.2 Le système du pendule inversé 57
V.3 Le contrôle du pendule inversé 58
V.3.1 Structure de simulation de la commande 58
V.3.2 Conception du contrôleur 59
V.3.3 Contrôleur 333 60
V.3.3.1 Résultats de simulation 62
V.3.3.2 Robustesse du contrôleur 333 64
V.3.4 Contrôleur 555 74
V.3.4.1 Résultats de simulation 74
V.3.4.2 Robustesse du contrôleur 555 75
V.4 Conclusion 77

CONCLUSION 86
BIBLIOGRAPHIE 88

Liste des figures

Figures :

Fig.2.1 : Croisement à un site (k=3) 10

Fig.2.2 : Principe de la mutation 11

Fig.2.3 : Organigramme de l'AG 17

Fig.3.1 : Structure d'un contrôleur à logique floue 21

Fig.3.2 : Formes des fonctions d'appartenance 23

Fig.3.3 : Univers de discours TP 23

Fig.3.4 : Univers de discours TPE avec 5 ensembles flous 23

Fig.3.5 : Réseau de Hopfield 26

Fig.3.6 : Structure d'un réseau statique 27

Fig.3.7 : Configuration d'un FNN régulier 30

Fig.3.8 : Configuration d'un FNN avec des entrées ,poids et sorties flous 31

Fig.3.9 : processeur logique 33

Fig.3.10 : Architecture d'un FLP 34

Fig.3.11 : Structure d'un FNN comme contrôleur flou 35

Fig.3.12 : Structure d'un FNN proposé dans [36] 37

Fig.3.13 : Fonction d'appartenance triangulaire de centre C et de largeur W 39

Fig.3.14 : Codage typique d'une base de règle floue 41

Fig.3.15 : Mécanisme d'apprentissage off-line 42

Fig.4.1 : Architecture du réseau contrôleur 48

Fig.4.2 : Univers de discours TPE de largeur L 49

Fig.4.3 : Principe du croisement conditionné 51

Fig.4.4 : Principe de la mutation conditionnée 52

Fig.4.5 : Diagramme des interactions de l'AG ,FLC et le modèle de simulation 53

Fig.4.6 : Organigramme de l'opération du contrôleur 54

Fig.5.1 : Le système du pendule inversé 58

Fig.5.2 : Structure de commande du pendule inversé 59

Fig.5.3 : Les fonctions d'appartenance du contrôleur 333 64

Fig.5.5.a : Variations de l'angle à partir du point (20deg , 0deg/s) 66
Fig.5.5.b : Variations de la vitesse angulaire à partir du point (20deg , 0deg/s) 66
Fig.5.5.c : plan de phase à partir de (20deg , 0deg/s) 67
Fig.5.5.d : Variations de la force 67
Fig.5.6.a : Variations de l'angle pour des conditions initiales différentes 68
Fig.5.6.b : Variations de la vitesse angulaire pour des conditions initiales différentes 68
Fig.5.6.c : plan de phase pour des conditions initiales différentes 69
Fig.5.6.d : Variations de la force pour des conditions initiales différentes 69
Fig.5.7.a : Variations de l'angle pour les conditions initiales :
(10,20) ,(30,-20) ,(50,10) ,(70,-10) 70
Fig.5.7.b : Variations de la vitesse angulaire pour les conditions initiales :
(10,20) ,(30,-20) ,(50,10) ,(70,-10) 70
Fig.5.7.c : Plan de phase à partir des points :(10,20) ,(30,-20) ,(50,10) ,(70,-10) 71
Fig.5.7.d : Variations de la force à partir des points : (10,20) ,(30,-20) ,(50,10) ,(70,-10) 71
Fig.5.8.a : Variations de l'angle pour des pendules de longueur variable 72
Fig.5.8.b : Variations de la vitesse angulaire pour des pendules de longueur variable 72
Fig.5.8.c : Plan de phase pour des pendules de longueur variable 73
Fig.5.8.d : Variations de la force pour des pendules de longueur variable 73
Fig.5.10 : Les fonctions d'appartenance du contrôleur 555 76
Fig.5.11.a : Variations de l'angle à partir du point (20deg , 0deg/s) 78
Fig.5.11.b : Variations de la vitesse angulaire à partir du point (20deg , 0deg/s) 78
Fig.5.11.c : plan de phase à partir de (20deg , 0deg/s) 79
Fig.5.11.d : Variations de la force 79
Fig.5.12.a : Variations de l'angle pour des conditions initiales différentes 80
Fig.5.12.b : Variations de la vitesse angulaire pour des conditions initiales différentes 80
Fig.5.12.c : plan de phase pour des conditions initiales différentes 81
Fig.5.12.d : Variations de la force pour des conditions initiales différentes 81
Fig.5.13.a : Variations de l'angle pour les conditions initiales :
(10,20) ,(30,-20) ,(50,10) ,(70,-10) 82
Fig.5.13.b : Variations de la vitesse angulaire pour les conditions initiales :
(10,20) ,(30,-20) ,(50,10) ,(70,-10) 82
Fig.5.13.c : Plan de phase à partir des points :(10,20) ,(30,-20) ,(50,10) ,(70,-10) 83

Fig.5.13.d : Variations de la force à partir des points : (10,20) ,(30,-20) ,(50,10) ,(70,-10) ...83
Fig.5.14.a : Variations de l'angle pour des pendules de longueur variable 84
Fig.5.14.b : Variations de la vitesse angulaire pour des pendules de longueur variable 84
Fig.5.14.c : Plan de phase pour des pendules de longueur variable 85
Fig.5.14.d : Variations de la force pour des pendules de longueur variable 85

Tableaux :

Tab.3.1 : Règles de contrôle floues d'un procédé industriel 35
Tab.5.1 : Intervalles de recherche des paramètres pour le contrôleur 333.....61
Tab.5.2 : Les valeurs des paramètres génétiques 61
Tab.5.3 : Base de règles floues du contrôleur 333 63
Tab.5.4 : Intervalles de recherche des paramètres pour le contrôleur 555 74
Tab.5.5 : Base de règles floues du contrôleur 555 75

INTRODUCTION

Introduction

Depuis la première application du formalisme de la logique floue à la commande des systèmes proposée par Mamdani, plusieurs études ont montré que le contrôle à logique floue est une méthode adéquate pour la commande des procédés mal définis ou complètement inconnus qui ne peuvent pas être modélisé facilement d'une manière mathématique.

L'approche traditionnelle pour la conception floue est basée sur les connaissances acquises par des opérateurs experts, bien que cette approche ait prouvé son efficacité dans plusieurs applications, il se peut cependant que les opérateurs ne peuvent pas transcrire leur connaissance et expérience sous forme de contrôleur à logique floue. De plus, il arrive parfois que le domaine d'expertise ne soit pas disponible.

Pour ces raisons, plusieurs chercheurs se sont intéressé à l'élaboration des méthodes optimales et systématiques pour la conception des contrôleurs flous. Deux directions de recherche sont apparues :l'approche connexioniste et l'approche directe.

L'approche connexioniste consiste à combiner la théorie des réseaux de neurones artificiels (ANN, Artificial Neural Networks) et les systèmes flous pour construire ce qu'on appelle un réseau de neurones flous (FNN, Fuzzy Neural Network). Ce réseau constitué de plusieurs couches contenant des neurones qui réalisent les opérations floues de base telles que la fuzzification, l'opération AND, l'opération OR et la défuzzification est utilisé pour implanter le mécanisme d'inférence flou.

L'apprentissage du réseau permet d'ajuster les paramètres de conception et les inférences sont extraites après convergence.

Dans l'approche directe, le problème de conception du contrôleur flou est traité comme un problème d'optimisation qui consiste à trouver un ensemble optimal des fonctions d'appartenance ou/et des règles d'inférence qui minimisent une certaine fonction coût. Du à la complexité de l'espace de recherche, ce problème d'optimisation est résolu en utilisant les algorithmes génétiques (AG) qui sont des procédures de recherche basées sur les mécanismes de la génétique naturelle.

L'objectif de ce travail s'inscrit dans la même direction et concerne la conception des contrôleurs à logique floue par la combinaison des deux paradigmes :les réseaux de neurones et les algorithmes génétiques.

Pour cela le contrôleur à logique floue est projeté sur un réseau interconnecté multicouche et l'algorithme génétique est utilisé pour optimiser les paramètres de ce réseau. Il s'agit alors de concevoir simultanément les paramètres des fonctions d'appartenance triangulaires symétriques et l'ensemble des règles d'inférence. La solution utilise les algorithmes génétiques qui sont des procédures de recherche basées sur les mécanisme de la génétique naturelle pour trouver le contrôleur qui minimise une certaine fonction coût.

Pour étudier ses performances, cette nouvelle méthode a été appliquée pour la conception d'un contrôleur flou pour un pendule inversé .

Ce travail est organisé en cinq chapitres :

- Dans le deuxième chapitre, il y a une description des outils nécessaires à la résolution des problèmes d'optimisation posés par la mise en œuvre des algorithmes génétiques.
- Le troisième chapitre présente un aperçu sur les contrôleurs à logique floue et traite les différentes approches de conception de systèmes flous à savoir l'approche directe et l'approche connexioniste.
- Le chapitre quatre comprend la nouvelle méthodologie de conception qu'on a proposé, c'est une approche mixte et concerne la combinaison des AG et les réseaux de neurones afin d'assurer une conception optimale du contrôleur flou.
- Le cinquième chapitre est consacré aux résultats des différents tests effectués pour la commande du système du pendule inversé.
- Et enfin une conclusion et perspectives font l'objet du dernier chapitre.

Les algorithmes génétiques

II.1 Introduction.

II.2 Formulation du problème d'optimisation.

II.3 Principe de fonctionnement des algorithmes génétiques.

II.4 Les mécanismes d'un algorithme génétique simple.

II.4.1 La reproduction.

II.4.2 Le croisement.

II.4.3 La mutation.

II.4.4 Notion de schéma.

II.4.4.1 Effet de la reproduction.

II.4.4.2 Effet du croisement.

II.4.4.3 Effet de la mutation.

II.5 La sélection des individus d'une nouvelle génération.

II.6 Description du codage.

II.7 Description du décodage.

II.8 Organigramme de l'algorithme génétique.

II.9 La convergence des algorithmes génétiques.

II.9.1 Critères de convergence .

II.9.2 Taux de convergence .

II.10Conclusion.

II.1 Introduction

Dans la pratique, un grand nombre de fonctions à optimiser ne sont pas dérivables et ne sont souvent même pas continues. Le monde réel à explorer est envahi de discontinuités, ce qui le rend bien moins adapté au calcul. Il est donc évident que les méthodes soumises aux contraintes de discontinuité et de dérivabilité ne sont adaptées qu'à une classe de problèmes très limitée. En plus elles ne sont pas suffisamment robustes. Ce type de méthodes dites classiques sont susceptibles d'être piégées dans des optimums locaux [38].

Récemment, une classe de méthodes est apparue, employant les principes d'évolution et d'hérédité de la nature et présentant une probabilité importante de convergence vers un optimum global de la fonction à optimiser. Ce sont des méthodes pseudo-aléatoires appelées "LES ALGORITHMES GENETIQUES".

Les algorithmes génétiques (AG) développés par John Holland sont des techniques d'optimisation stochastiques qui tentent d'imiter les processus d'évolution naturelle des espèces et de la génétique. Ils agissent sur une population d'individus assujettis à une sélection darwinienne :les individus(ou parents)les mieux adaptés à leur environnement survivent et peuvent se reproduire. Ils sont alors soumis à des mécanismes de recombinaisons analogues à ceux de la génétique. Des échanges de gènes entre parents résulte la création de nouveaux individus(ou enfants), qui permettent de tester d'autres configurations de l'espace de recherche.

Dans leurs recherche du maximum ces algorithmes ne nécessitent que la connaissance de la valeur de la fonction qu'on souhaite optimiser dite fonction coût(fonction qui doit être toujours positive).

II.2 Formulation du problème d'optimisation

En raison de l'analogie avec la théorie de l'évolution (survie des individus les mieux adaptés à leur environnement), l'algorithme génétique est naturellement formulé en terme de maximisation. Etant donnée une fonction f réelle à une ou plusieurs variables, le problème d'optimisation sur l'espace de recherche E s'écrit de la manière suivante :

$$\max_{x \in E} f(x) \qquad (2.1)$$

De plus, la fonction à optimiser par un algorithme génétique doit avoir des valeurs positives sur l'ensemble du domaine *E.* Dans le cas contraire, il convient d'ajouter aux valeurs de *f* une constante positive *Fmin* conformément à l'équivalence de (2.1) et (2.2).

$$\max_{x \in E} f(x)+Fmin \tag{2.2}$$

Dans beaucoup de problème, l'objectif est exprimé sous la forme de minimisation d'une fonction coût g ,

$$\min_{x \in E} g(x) \tag{2.3}$$

Le passage du problème de minimisation à un problème de maximisation est obtenu par transformation de la fonction g. La transformation souvent utilisée est :

$$\max_{x \in E} h(x) \tag{2.4}$$

avec :

$$h(x) = \begin{cases} G\max - g(x) & si \quad G\max \geq g(x) \\ 0 & \sin on \end{cases} \tag{2.5}$$

de la fonction h n'est pas unique. En effet, toute composition de la fonction g par une fonction quelconque Notons que le choix décroissante et monotone sur le domaine E, conduirait à un problème de maximisation équivalent à (2.3). on rencontre notamment dans la littérature la fonction de transformation h suivante [29].

$$h(x) = \frac{1}{1+g(x)} \tag{2.6}$$

II.3 Principe de fonctionnement des AG

Les AG effectuent le procédé d'optimisation en agissant sur une population de créatures artificielles (chaînes de caractères) analogues aux chromosomes en nature. Chaque créature ou individu représente un point de recherche dans l'espace des solutions à qui on associe une valeur de fonction coût, dont on veut obtenir la valeur maximum.

La population est aléatoirement crée, ensuite elle va tendre vers les meilleurs régions de l'espace de recherche en réalisant des opérations génétiques.

Les AG diffèrent fondamentalement des autres méthodes dans la recherche de l'optimum[29]:

1. Les AG utilisent un codage des paramètres d'origine du problème d'optimisation et non les paramètres eux même.
2. Les AG travaillent sur une population de points, au lieu d'un point unique.
3. Les AG n'utilisent que les valeurs de la fonction à optimiser, pas sa dérivée ou une autre connaissance auxiliaire.

4.Les AG utilisent des règles de transition probabilistes et non déterministes.

II.4 Les mécanismes d'un AG simple

A partir d'une population initiale d'individus crée aléatoirement, les AG génèrent de nouveaux individus plus performant que leurs prédécesseurs en effectuant des opérations génétiques.

Un AG simple est composé de trois opérateurs :la reproduction, le croisement et la mutation.

La reproduction est une version artificielle de la sélection naturelle, c'est un processus dans lequel chaque individu est copié en fonction des valeurs de la fonction coût. Le croisement est l'opérateur le plus dominant dans un AG, il permet à deux chaînes d'échanger des portions de leurs structures produisant ainsi de nouvelles chaînes. La mutation est un opérateur local qui est appliqué avec une très faible probabilité.

II.4.1 La reproduction (la selection des parents)

La reproduction ou la sélection des parents est un mécanisme qui consiste à former une nouvelle génération par une sélection aléatoire des chaines d'une population existante en fonction de leur performance(fonction coût).

Lors de cette phase, les individus les plus forts sont généralement dupliqués et forment les parents de la génération en cours, alors que les faibles disparaissent sans avoir la possibilité de se reproduire.

II.4.1.1 Les méthodes de sélection

Elles sont caractérisées par la probabilité P_s (a^t_j) qu'un individu a^t_j de la population (t) soit retenu pour participer à la recombinaison génétique.

a. La sélection proportionnelle(roue de loterie biaisée)

Ce mode de sélection des parents consiste à dupliquer chaque individu de la population proportionnellement à son adaptation dans son milieu.

Soit :

f_j la valeur de la fonction coût associée au j^{eme} individu.

f_s la somme des valeurs de cette fonction.

La sélection pouvant être faite en utilisant le rapport (f_j/f_s) pour réaliser une roulette pondérée ou chaque individu occupe une surface proportionnelle au rapport précédent. Des tirages aléatoires sur cette roulette donneront les chaînes qui participeront à la prochaine population. De cette façon, les chaînes bien adaptées ont un plus grand nombre de descendants dans les générations suivantes.

b. La sélection à reste stochastique

Dans ce mode de sélection, le nombre de copies $n(a^t_j)$ d'un individu a^t_j est directement fixé par le rapport (f_j/f_{moy}) ou f_{moy} est la valeur moyenne de la fonction coût. Dans un premier temps, on reproduit chaque individu (partie entière de (f_j/f_{moy})) fois :

$$n(a^t_j)=\text{partie entière}[f_j/f_{moy}] \tag{2.7}$$

Puis la population est complétée par tirages au sort en associant à chaque individu a^t_j une probabilité $P_s(a^t_j)$:

$$P_s(a^t_j) = f_j/f_{moy} - \text{partie entière}[f_j/f_{moy}] \tag{2.8}$$

c. La sélection par tournoi stochastique

dans cette méthode de sélection qui a été proposée par Bridle, les probabilités de sélection sont calculées normalement et des paires successives d'individus sont tirées au sort grâce à la sélection par roue de loterie. Après avoir tirer une paire de chaînes, la chaîne ayant l'adaptation la plus élevée est déclarée vainqueur, elle est ajoutée à la nouvelle population, et une nouvelle paire est tirée. Ce processus continu jusqu'à ce que la population soit remplie [29]. D'autres méthodes de sélection sont présentées dans [40].

II.4.1.2 Mécanismes de changement d'échelle

Bien que la reproduction ne fait que copier les individus en fonction de leur fonction coût, il est fréquent d'avoir dans les premières générations des "supers-individus"qui vont tendre à dominer les procédés de sélection et d'accouplement, ils peuvent donc se reproduire entre eux et entraîneront une convergence prématuré.

Pour éviter ce problème et de préserver la diversité des individus, il est recommandé de réaliser un réajustement de la fonction coût avant d'effectuer la reproduction.

Les mécanismes de changement d'échelle les plus utilisés sont :

1. le changement d'échelle linéaire.
2. la troncature en sigma (σ).
3. le changement d'échelle en puissance.

a) Le changement d'échelle linéaire

C'est la technique de réajustement la plus répandue, elle consiste à transformer l'adaptation brute f en une adaptation transformée f' en utilisant une relation linéaire de la forme :

$$f' = af+b \tag{2.9}$$

Les coefficients a et b sont choisis de façon à ce que l'adaptation brute et transformée ont la même moyenne ($f'_{moy}=f_{moy}$) et à ce que la valeur maximale de l'adaptation transformée est un multiple (en général le double) de cette moyenne :

$$f'_{max}=\beta f_{moy} \quad , \quad 1 \leq \beta \leq 2 \tag{2.10}$$

De cette manière, on peut s'assurer que les individus de valeur moyenne sont copiés une fois en moyenne et les meilleurs sont copiés un nombre de fois égal au multiple choisi.

remarque : Dans ce mécanisme, il faut faire attention d'éviter des valeurs d'adaptation transformées négatives.

b) La troncature en sigma

Dans cette procédure, on utilise l'écart type de la population, ce qui permet d'éliminer les individus trop faibles.

La nouvelle fonction coût s'écrit sous la forme :

$$f'=f-(f_{moy}-c\sigma) \tag{2.11}$$

où

c :coefficient de l'écart type choisi arbitrairement $1 \leq c \leq 3$.

σ :l'écart type de l'adaptation calculé sur la population.

c) Le changement d'échelle en puissance

Dans ce cas l'adaptation transformée f' prend la valeur d'une certaine puissance de l'adaptation brute f :

$$f' = f^k \tag{2.12}$$

où la constante k peut être adaptée au cours des générations pour accentuer la sélection des individus situés à proximité des sommets.

II.4.2 Le croisement

Après avoir sélectionner les chaînes les mieux adaptées dans le processus de reproduction, elles vont subir maintenant à l'opération du croisement qui consiste à échanger des matériels génétiques entre deux chaînes reproductrices (parents) pour produire deux nouvelles chaînes (enfants).

Le croisement est un processus aléatoire de probabilité P_c appliquée à un couple de parents arbitrairement choisis dans la population.

un entier k représentant une position sur la chaîne est choisi aléatoirement entre 1 et la longueur de la chaîne moins 1 ($k \in [1\ L-1]$), cet entier représentera donc la position ou se produira le croisement, et enfin deux nouvelles chaînes sont crées en échangeant le matériel génétique (caractères) compris entre les positions k+1 et L inclues des chaînes reproductrices.

Ce processus est illustré dans la figure (Fig. 2.1) ou P1 et P2 sont les parents et O1 et O2 sont les enfants résultats de l'opération du croisement au point k=3.

P1 010|10011

P2 001|01001

O1 010 01001

O2 001 10011

Fig. 2.1 : Croisement à un site (k=3)

II.4.3 La mutation

La mutation est la modification aléatoire occasionnelle (de faible probabilité) d'un gène d'un individu. l'opérateur de mutation consiste à complémenter la valeur d'un bit du chromosome avec une probabilité P_m. Le processus est exécuté bit par bit (voir Fig.2.2).

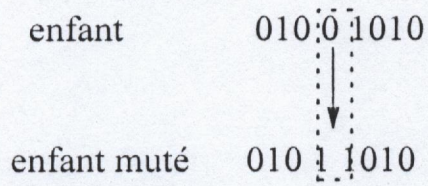


Fig. 2.2 : Principe de la mutation

Pour ne pas détériorer les performances de l'AG, Goldberg conseille une fréquence de mutation tous les 1000 bits [29]. Certains préconisent plutôt la valeur suivante pour P_m [40] ,

$$P_m^* = \frac{1}{L} \tag{2.13}$$

ou L est la longueur du chromosome.

D'autres études ont conduit à une formule empirique qui exprime le taux optimal de mutation en fonction de la longueur du chromosome L et de la taille de la population N :

$$P_m^* = \frac{1}{N\sqrt{L}} \tag{2.14}$$

Les effets de la reproduction, du croisement et de la mutation sur l'évolution d'une population donnée s'expliquent en utilisant la notion de schéma.

II.4.4 Notion de schéma

Un schéma est un motif de similarité décrivant un sous ensemble de chaînes avec des similarités à des positions définies. Un alphabet étendu est utilisé pour présenter cette notion {0,1,*} ou le symbole * pouvant prendre indifféremment la valeur 0 ou 1.

Par exemple, considérons les chaînes et les schémas de longueur 5, le schéma H=*101* décrit l'ensemble de quatre éléments :{01010,01011,11010,11011}.

Il y a deux caractéristiques propres au schéma :son ordre et sa longueur utile.

L'ordre d'un schéma H, noté o(H) est le nombre de positions instanciées (nombre de 1 et de 0 pour un alphabet binaire) dans le motif considéré.

Par exemple : o(*101*) = 3 .

La longueur utile d'un schéma H, notée δ(H) désigne la distance entre la première et la dernière position instanciée dans le motif considéré.

Par exemple : δ(*101*) = 4-2 = 2.

Les schémas sont des outils très intéressants qui permettent d'analyser l'effet de la reproduction et des autres opérateurs génétiques. On va considérer par la suite, les effets isolés puis combinés de la reproduction, du croisement et de la mutation sur les schémas contenus dans la population de chaînes.

II.4.4.1 Effet de la reproduction

L'importance du rôle de la sélection peut être mesurée par son effet sur la propagation d'un schéma H dans une population de N individus entre deux générations. on suppose que la sélection est de type proportionnelle et que le nombre d'exemplaires du schéma H est égal à $m(H,t)$ à la génération t. A la génération t+1, le nombre attendu de représentants de H est estimé comme suit [29] :

$$m(H,t+1) = m(H,t)\frac{f(H)}{f_{moy}} \qquad (2.15)$$

ou $f(H)$ et f_{moy} désignent respectivement l'adaptation moyenne des individus représentant le schéma H à la génération t et l'adaptation moyenne de la population.

L'équation (2.15) montre que les schémas ayant des coûts moyens supérieurs à celui de la population seront davantage copiés à la génération suivante, ceux qui ont un coût moyen inférieur à la moyenne le seront moins.

En supposant qu'un schéma quelconque H reste au dessus de la moyenne d'une quantité égale à $c.f_{moy}$ ou c est une constante ($f(H) = f_{moy} + c.f_{moy}$).

L'équation (2.15) qui traduit l'évolution des schémas peut s'écrire comme suit :

$$m(H,t+1) = (1+c).m(H,t) \qquad (2.16)$$

en supposant une valeur stationnaire de c, on obtient l'équation :

$$m(H,t+1) = (1+c)^{t}.m(H,0) \tag{2.17}$$

Ainsi, la reproduction alloue une place exponentiellement croissante (décroissante) aux schémas qui ont des performances au dessus (au dessous) de la moyenne.

II.4.4.2 Effet du croisement

La reproduction en elle même ne fait que copier les anciens individus et ne fait rien de promouvoir de nouveaux individus de l'espace de recherche. Le croisement permet d'introduire une nouvelle configuration en effectuant un échange d'information entre les chaînes.

L'effet du croisement peut être perçu à travers la propagation des schémas comme lors de la sélection des parents. Il convient alors de déterminer le taux de destruction (ou de survie) d'un schéma quelconque sous l'opérateur considéré.

Goldberg a indiqué qu'un schéma survit quand le point de croisement tombe à l'extérieur de sa longueur utile [29].

La probabilité de destruction d'un schéma H de longueur utile δ(H) lors du croisement simple est donnée par :

$$P_d(H) = \frac{\delta(H)}{L-1} \tag{2.18}$$

ou L désigne la longueur du chromosome.

Si le croisement a lieu avec une probabilité P_c, la probabilité de survie est donc :

$$P_s(H) \geq 1 - P_c.\frac{\delta(H)}{L-1} \tag{2.19}$$

En considérant que la reproduction et le croisement sont des opérations indépendantes, l'effet combiné de ces deux opérations s'obtient en multipliant le nombre attendu de schémas lors de la reproduction seule par la probabilité P_s de survie lors d'un croisement :

$$m(H,t+1) \geq m(H,t).\frac{f(H)}{f_{moy}}\left[1 - P_c\frac{\delta(H)}{L-1}\right] \tag{2.20}$$

Cette équation montre que les schémas de longueurs δ(H) faibles et de coût moyen *f*(H)

supérieur à la moyenne f_{moy} verront leur nombres augmenter à la génération t+1 par rapport à t.

II.4.4.3 Effet de la mutation

Comme il a été déjà dit, la mutation est une modification aléatoire d'une position quelconque de la chaîne moyennant une probabilité de mutation P_m.

Pour qu'un schéma H survive à l'opération de mutation, toutes ses positions instanciées doivent survivre. comme la probabilité de survie d'une position quelconque est $(1-P_m)$ et que le nombre de positions fixes dans le schéma est o(H), la probabilité de survie d'un schéma h à la mutation est :

$$P_s(H) = (1 - P_m)^{o(H)} \tag{2.21}$$

La probabilité de mutation étant petite ($P_m << 1$), l'expression précédente peut être approximée par :

$$P_s(H) = 1 - o(H).P_m \tag{2.22}$$

Nous pouvons donc conclure qu'on peut s'attendre à ce qu'un schéma H quelconque reçoive, du fait de la reproduction, du croisement et de la mutation, un nombre de copies donné par l'équation suivante :

$$m(H,t+1) \geq m(H,t).\frac{f(H)}{f_{moy}}\left[1 - P_c \frac{\delta(H)}{L-1}\right]\left[1 - o(H).P_m\right] \tag{2.23}$$

Cette équation nous permet d'en conclure que les schémas courts, d'ordre faible et de performance au-dessus de la moyenne font l'objet d'un nombre de tests exponentiellement croissants dans les générations suivantes, cette conclusion est appelée le théorème des schémas ou le théorème fondamental des AG.

II.5 La sélection des individus d'une nouvelle génération

A la suite de la recombinaison génétique (croisement et mutation), la population compte 2N individus (N parents et N enfants) ;N étant la taille de la population. il faut donc éliminer N individus pour constituer la génération suivante. C'est le rôle de la sélection finale

S_f qui agit sur les populations de parents ($P^{'t}$) et d'enfants ($P^{''t}$) d'une génération pour créer la nouvelle génération (P^{t+1}).

$$P^{t+1} = S_f(P^{'t}, P^{''t}) \tag{2.24}$$

Pour effectuer cette sélection entre parents et enfants, plusieurs stratégies sont possibles.

II.5.1 La sélection par descendance

Il n'y a aucune compétition entre parents et enfants. La population de la nouvelle génération est obtenue par descendance, les enfants remplaçant automatiquement leurs parents quel que soit leur adaptation.

$$P^{t+1} = S_f(P^{'t}, P^{''t}) = P^{''t} \tag{2.25}$$

L'inconvénient de ce mode de sélection est que l'on risque de voir disparaître les caractéristiques génétiques des parents les mieux adaptés si elles n'ont pas été totalement transmises lors de la recombinaison génétique.

II.5.2 La sélection par compétition

Une compétition a lieu entre parents et enfants pour déterminer les "survivants" de la génération. Ainsi, les enfants peuvent être insérés dans la population si et seulement si leur performance est supérieure à celle de leurs parents à rang équivalent.

$$a_i^{t+1} = \begin{cases} a_i^{''t} & si \quad f(a_i^{''t}) > f(a_i^{'t}) \\ a_i^{'t} & \text{sinon} \end{cases} \tag{2.26}$$

Avec a_i^{t+1} est le i^{eme} individu de la génération (t+1).

II.5.3 Steady state selection

Cette technique consiste à effectuer une compétition après chaque recombinaison génétique entre parents et enfants en retenant les deux meilleurs individus parmi les quatre.

II.5.4 Selective breeding selection

Dans cette méthode, on garde les N meilleurs individus parmi la population intermédiaire de parents et d'enfants pour former la nouvelle génération. Ceci garantie que les meilleurs individus de la population (qu'ils soient enfants ou parents) sont nécessairement conservés pour la génération suivante.

II.6 Description du codage

Chaque chaîne de la population est codée sur un nombre fini de bit. La longueur de la chaîne est fixée par l'utilisateur selon :

- Le domaine de variation de la chaîne.
- La précision demandée.

Pour l'optimisation d'une fonction à plusieurs variables $f(x_1,x_2,...,x_n)$, on utilise un codage binaire concaténé qui consiste à :

- Coder chaque variable selon le choix de la longueur.
- Construire les chaînes en concaténant les différents codes.

Soit chaîne = (code1)(code2)...(coden)

avec codei :le code du i^{eme} paramètre dans l'espace de recherche de dimension n.

II.7 Description du décodage

Chaque chaîne doit être décodée pour pouvoir calculer la valeur de la fonction coût qui lui est associée. parmi les types de décodage possibles, le décodage binaire est souvent le plus utilisé.

Dans le cas ou chaque paramètre x_i est situé dans un intervalle $[x_{imin}\ x_{imax}]$, et à qui est associé une chaîne binaire $b_0b_1...b_{Lxi-1}$ définie sur Lxi bits. A cette chaîne correspond une valeur entière naturelle,

$$N(x_i) = \sum_{i=0}^{Lxi-1} 2^{Lxi-i-1}.b_i \qquad (2.27)$$

Le paramètre réel x_i de l'espace de recherche relatif à $N(x_i)$ est obtenu par interpolation linéaire :

$$x_i = x_{i\min} + \frac{x_{i\max} - x_{i\min}}{2^{Lxi} - 1}.N(x_i) \qquad (2.28)$$

la précision des paramètres par ce type de décodage est :

$$\varepsilon_i = \frac{x_{i\max} - x_{i\min}}{2^{Lxi} - 1} \qquad (2.29)$$

II.8 Organigramme de l'AG

Un AG procède selon l'organigramme suivant :

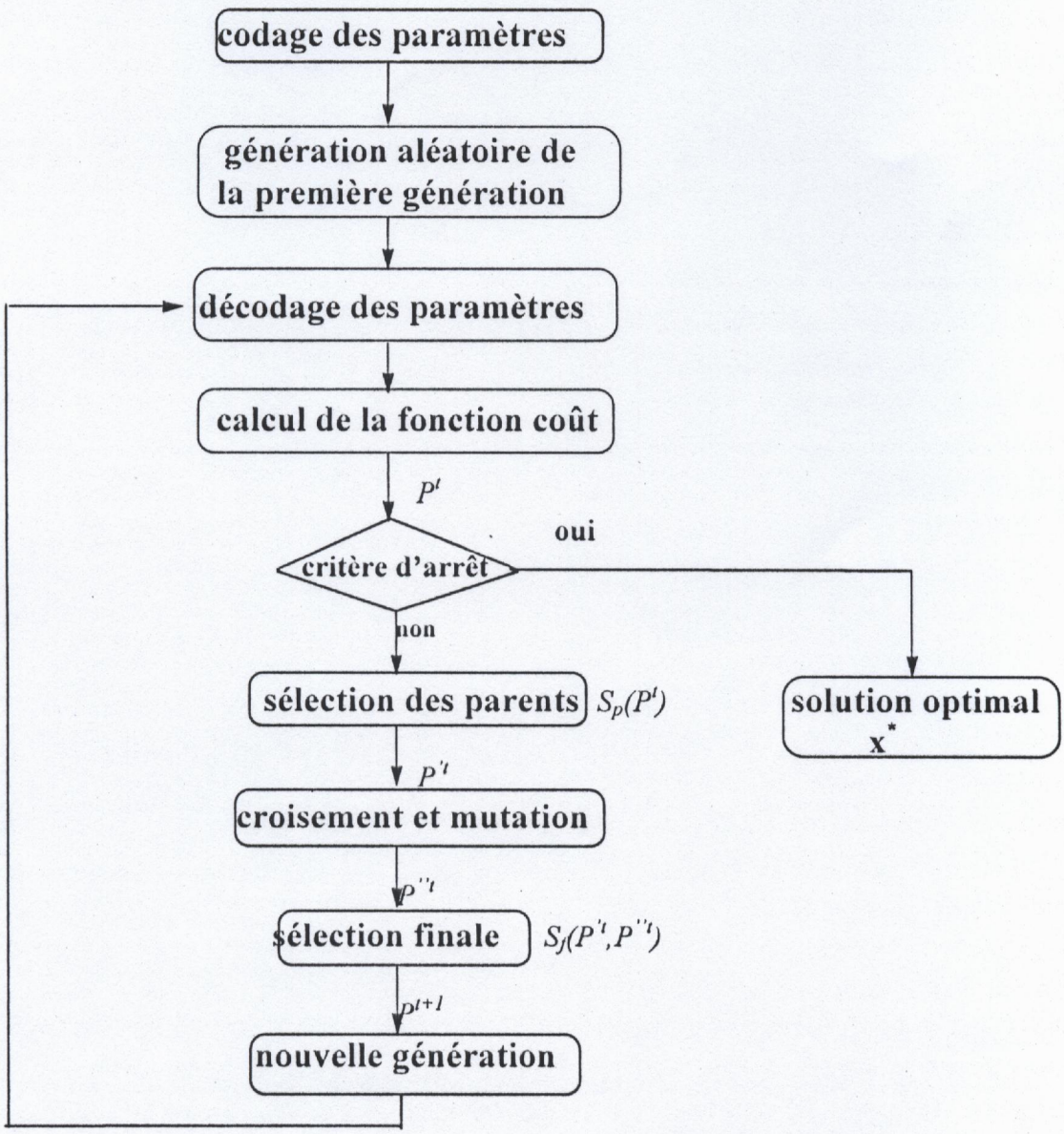


Fig. 2.3 : Organigramme de l'AG

II.9 La convergence des AG

II.9.1 Critères de convergence

Le problème de convergence est généralement éludé en imposant un nombre maximal de générations t_{max} et en arrêtant la recherche lorsque $t = t_{max}$. On estime alors que l'algorithme a convergé et que l'individu de plus forte performance dans la population P^{tmax} correspond à la solution recherchée.

Une méthode plus rigoureuse consiste à supposer que l'algorithme converge vers l'optimum lorsque l'adaptation d'une partie (ou de l'ensemble) de la population se rapproche de celle du meilleur individu. On peut considérer que cet événement se produit à la génération t pour laquelle :

$$1 - \frac{f_{moy}}{f_{max}} \leq \varepsilon \qquad (2.30)$$

ou : ε est la précision requise sur la convergence.

f_{max} est la performance du meilleur individu de la population à la génération t.

f_{moy} est la moyenne de l'adaptation calculée sur l'ensemble de la population ou sur une partie correspondant à un pourcentage des représentants les plus performants.

Nous pouvons aussi supposer que l'algorithme a convergé lorsque le meilleur individu de la population n'évolue plus.

Ces critères ne sont pourtant absolument pas fiables dans mesure ou l'algorithme peut converger et se stabiliser autour d'une solution dans l'attente d'une mutation qui le dirigera vers une autre région plus intéressante. Le choix d'un critère d'arrêt idéal reste donc sans réponse.

II-9-2 Taux de convergence

Du fait des problèmes de convergence et du caractère stochastique de l'exploration génétique, il est habituel d'exécuter plusieurs fois le même algorithme sur le même problème. Un taux de convergence (ou taux de réussite) rend compte de son efficacité. On peut par exemple définir le taux de convergence comme le rapport du nombre de fois ou l'algorithme a convergé vers la solution optimale sur le nombre total d'exécution.

II-10 Conclusion

Les AG sont des outils d'optimisation performants qui permettent de réaliser une exploration globale de l'espace. Contrairement aux méthodes déterministes classiques, ils ne nécessitent aucun calcul de dérivées et peuvent être appliquées aussi bien à des fonctions continues d'une seule variable, qu'à des fonctions discontinues dépendant d'un grand nombre de paramètres.

Les AG sont capables d'obtenir des solutions quasi-optimales pour différents types de problèmes sans connaissance explicite du domaine de travail en manipulant simplement des chaînes de bits et en utilisant des opérateurs simples qui ne mettent en jeu que des procédures aussi peu complexes que la génération de nombres aléatoires, la copie de chaînes et les échanges de morceaux de chaînes.

Les approches de conception d'un FLC

III.1 La logique floue .

III.1.1 Introduction .

III.1.2 Contrôleur à logique floue .

III.1.2.1 La fuzzification .

III.1.2.2 Bloc de contrôle .

III.1.2.3 La défuzzification .

III.1.2.4 Paramètres de conception d'un FLC .

III.2 Les approches de conception d'un FLC .

III.2.1 L'approche connexioniste .

III.2.1.1 Les réseaux de neurones .

a) Définition .

b) Architecture des réseaux de neurones .

c) Entraînement des réseaux de neurones .

d) L'algorithme de la rétropropagation .

III.2.1.2 Les réseaux de neurones flous .

a) Définition .

b) Les différentes approches des FNN .

c) Les architectures des FNN .

III.2.2 L'approche directe .

III.2.2.1 L'application des AG pour les contrôleurs flous .

a) Définition du problème : paramètres à optimiser

b) codage de l'AG .

III.2.2.2 L'apprentissage des systèmes flous par les AG

III.3 Conclusion .

III.1 La logique floue

III.1.1 Introduction

Les bases théoriques de logique floue ont été établies en 1965 par le professeur Lotfi A. Zadeh. Le développement de cette théorie vient de l'incapacité de décrire certains phénomènes physiques avec des modèles mathématiques exacts .

La logique floue possède certaines caractéristiques qui la distingue des techniques conventionnelles tel que la manipulation des variables linguistiques au lieu des variables numériques et les relations entre ces variables sont exprimées par des citations conditionnelles floues. C'est Mamdani qui a été le premier à appliquer cette nouvelle théorie à la commande des systèmes en 1974, par la suite une variété d'applications a été faites [2] .

III.1.2 Contrôleur à logique floue

Le contrôleur à logique floue (FLC ; fuzzy logic controller) est décrit par un ensemble de règles floues du type SI {conditions} ALORS {actions}, le FLC fournit donc un algorithme capable de convertir la stratégie de contrôle linguistique basée sur la connaissance experte en une stratégie de contrôle automatique .

La structure d'un FLC classique est montrée dans la figure (3.1) ,il est composé de trois parties :

- bloc de fuzzification.
- bloc de contrôle (base de règles floues et procédure d'inférence).
- bloc de defuzzification .

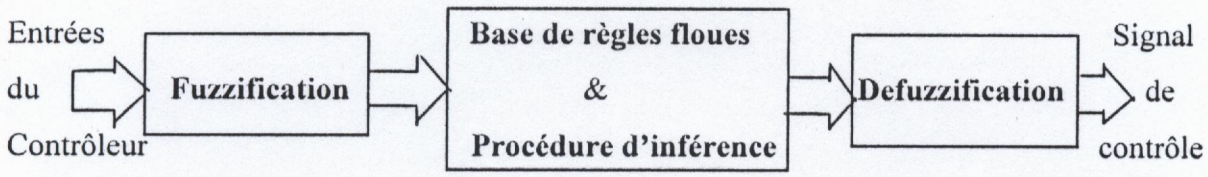


Fig. 3.1 : Structure d'un contrôleur à logique floue

III.1.2.1 La fuzzification

La fuzzification consiste à convertir les valeurs numériques des variables d'entrée en variables linguistiques (floues), pour cette fin on utilise des fonctions d'appartenance qui servent à subdiviser l'espace d'entrée, univers de discours, en ensembles floues.

Les formes les plus utilisées de ces fonctions sont la forme triangulaire ou trapézoïdal selon la relation (3.1) et la forme gaussienne selon la relation (3.2) (Fig. 3.2) .

$$\mu_A(x) = \begin{cases} \dfrac{x - L_A}{c_A - L_A} & si \quad L_A < x \leq c_A \\ \dfrac{x - r_A}{c_A - r_A} & si \quad c_A < x < r_A \\ 0 & ailleurs \end{cases} \tag{3.1}$$

$$\mu(x) = \exp\left(-\frac{(x-a)^2}{b}\right) \quad , \qquad -\infty < x < +\infty \tag{3.2}$$

Dans la plus part des applications, l'univers de discours est partitionné avec entre trois et neuf ensembles flous, la décomposition triangulaire de l'univers de discours comme indiquée dans la figure (3.3) est appelée système TP (Triangular Partition System) ,celle de la figure (3.4) est appelée système TPE (Triangular Partition with Evenly spaced midpoints System), dans cette décomposition on utilise une séquence de triangles isocèles avec des bases de la même largeur. En général, il est désirable d'avoir un certain chevauchement des ensembles flous afin d'obtenir des transitions lisses de la surface de contrôle.

III.1.2.2 Bloc de contrôle

Ce bloc est composé de deux parties :

- La base de règles floues : elle est formée d'un ensemble de règles de contrôles floues représentées par ensemble de relations linguistiques liant les variables d'entrée (les prémises) et les variables de sortie (conséquences) ,ces règles sont de la forme :

SI {l'ensemble des conditions sont satisfaites }

ALORS {un ensemble de conséquences peuvent être inférées }

- La procédure d'inférence ;elle consiste à dériver les actions de contrôle flou utilisant l'implication floue et les règles d'inférence déjà mentionnées .

Ces règles constituent un moteur d'inférence du contrôleur dans lequel on distingue deux types d'opérateurs ,l'opérateur de conjonction ET qui lie les différentes variables de la règle et l'opérateur de disjonction OU qui lie l'ensemble des règles formant la base de connaissance .

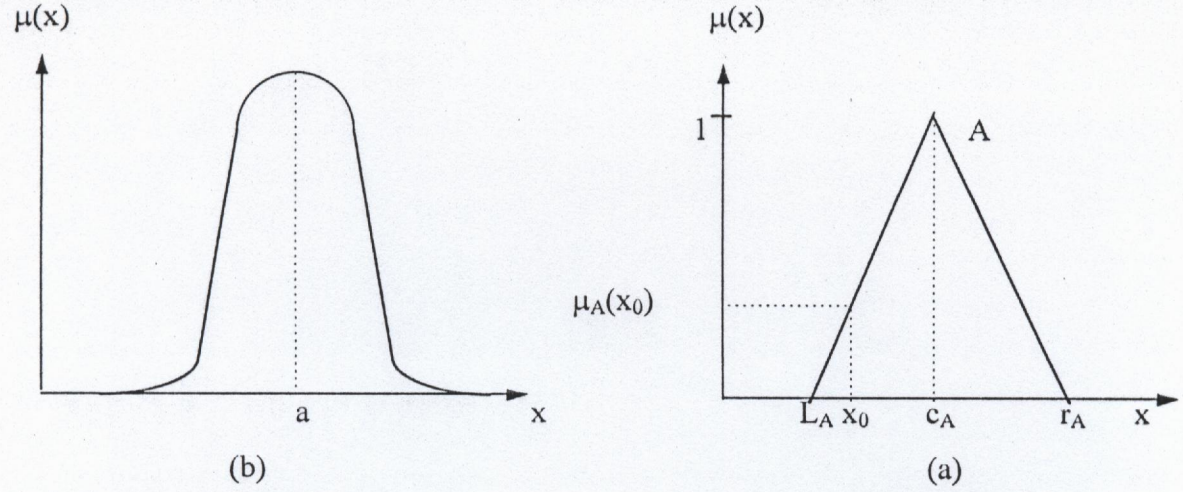


Fig. 3.2 : Formes des fonctions d'appartenance

(a) triangulaire
(b) gaussienne

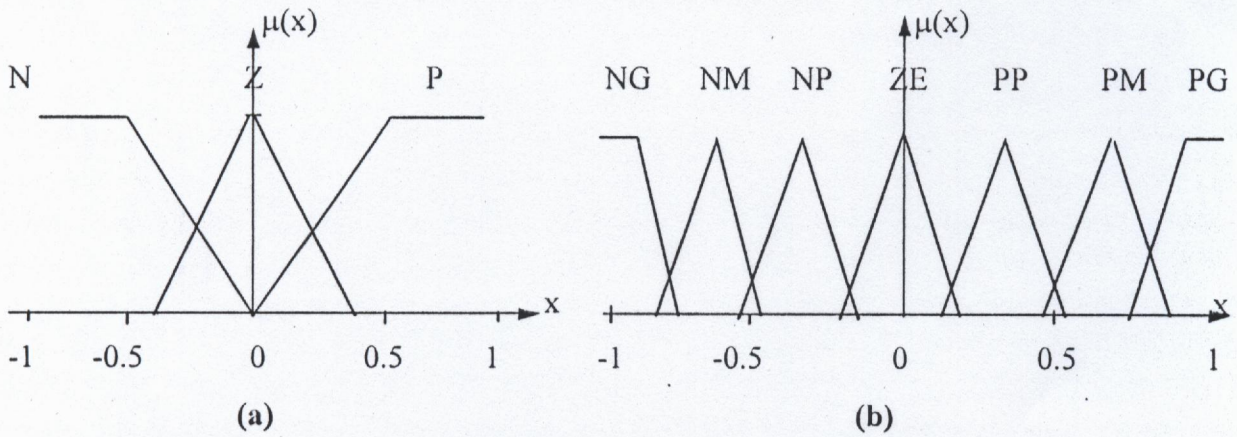


Fig. 3.3 : Univers de discours (TP) avec : (a) trois ensembles flous
(b) sept ensembles flous

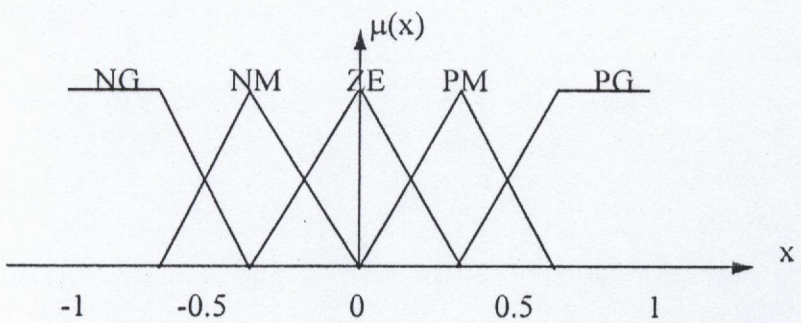


Fig. 3.4 : Univers de discours (TPE) avec 5 ensembles flous

Ces deux opérateurs peuvent être réalisés par différentes manières ,en effet l'opérateur ET peut être réalisé par la norme-t tel que le produit algébrique ou l'intersection floue (Min) et l'opérateur OU par la conorme-t tel que la somme algébrique ou l'union floue (Max) [2]. De ce fait plusieurs mécanismes d'inférence se présentent :

- Mécanisme d'inférence Max - Min .
- Mécanisme d'inférence Max - Prod .
- Mécanisme d'inférence Som - Prod .

III.1.2.3 La défuzzification

La défuzzification consiste à convertir l'action de contrôle floue inférée en une grandeur numérique. La stratégie de défuzzification la plus utilisée est celle du centre de gravité . Considérons un ensemble de règles floues de la forme :

R^i : SI x1 est A_1^i **ET x2 est** A_2^i **ALORS** y est B^i (3.3)

la sortie numérique y est la moyenne pondérée des sorties de toutes les règles :

$$y = \frac{\sum_{i=1}^{m} y_i . w_i}{\sum_{i=1}^{m} w_i} \tag{3.4}$$

avec m représente le nombre de règles floues .

y_i est le centre de la fonction d'appartenance caractérisant la valeur linguistique B_i.

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

$$w_i = \mu_{A_1^i}(x1).\mu_{A_2^i}(x2) \tag{3.5}$$

III.1.2.4 Paramètres de conception d'un FLC :

Ainsi la conception d'un contrôleur a logique floue implique le choix des paramètres suivants :

- la largeur de l'univers de discours .

-Le nombre de sous ensemble flous (variables linguistiques), la partition de l'univers de discours .

- Le type de fonctions d'appartenance : triangulaire ou gaussienne .
- Les règles d'inférences .
- Le choix de la stratégie de défuzzification .

Historiquement ce choix était empirique et était basé sur l'expérience d'opérateurs experts du procédé à commander. Cependant depuis quelques années, des méthodes systématiques de conception de contrôleur à logique floue ont été proposées. Dans ce qui suit nous passons en revue les principales approches systématiques de conception des FLC.

III.2 Les approches de conception d'un FLC

Bien que les contrôleurs à logique floue ont été appliqués avec succès sur plusieurs procédés industriels complexes, leur conception reste cependant une tache très difficile.
L'approche traditionnelle pour la conception floue qui est basée sur les connaissances acquises par des opérateurs experts est laborieuse, consomme beaucoup de temps et dans la plus part des cas spécifique pour chaque application ,en plus cette approche présente d'autres difficultés tel que :

- Les opérateurs ne peuvent pas facilement transformer leurs connaissances et expériences en une forme algorithmique ou base de règle nécessaire pour la conversion en une stratégie de contrôle automatique .
- Le domaine d'expertise n'est pas toujours disponible .
- Disponibilité des techniques d'acquisition des connaissances .

Suite à ces difficultés, des recherches intensives ont été effectuées dans le but de construire des méthodes systématiques et optimales pour la conception des FLC, ces recherches ont conduit au développement de deux nouvelles approches :

1. Approche connexioniste : elle consiste à combiner les réseaux de neurones artificiels (ANN; Artificial Neural Networks) et les systèmes flous pour construire ce qu'on appelle les réseaux de neurones flous (FNN; Fuzzy Neural Networks) [25] .
2. Approche directe : elle consiste à appliquer un algorithme d'optimisation pour la conception d'un système flou. Du à la complexité de l'espace de recherche, l'algorithme génétique est utilisé dans la plupart des cas.

III.2.1 L'approche connexioniste

III.2.1.1 Les réseaux de neurones

a) Définition

Un réseau de neurones traite l'information qu'il reçoit d'une manière analogue aux neurones du cerveau. Il est constitué de plusieurs éléments processeurs dits neurones

artificiels reliés les uns aux autres par un réseau complexe. Chaque neurone effectue une somme pondérée des signaux d'entrée modulés par une fonction dite d'activation (une fonction non linéaire) et génère une sortie qui sera appliquée aux autres neurones via des connexions pondérées. Avec cette simple structure et en choisissant un nombre approprié de neurones ,ces réseaux sont capables d'approximer n'importe quelle fonction continue avec une certaine précision.

b) Architecture des réseaux de neurones

Les réseaux de neurones peuvent être classés en deux principales catégories suivant la structure des connexions :les réseaux récurrents (dynamiques) et les réseaux non récurrents (statiques) .

Dans les réseaux récurrents, plusieurs neurones sont interconnectés pour organiser le réseau. Le réseau de Hopfield qui est indiqué dans (Fig. 3.5) appartient à ce type de structure.

Les réseaux non récurrents ont une structure hiérarchique qui consiste en plusieurs couches; une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Dans ce type de réseau (Fig. 3.6), il n'y a pas d'interconnexions entre les neurones de la même couche, et les signaux circulent de la couche d'entrée à la couche de sortie dans une seule direction .

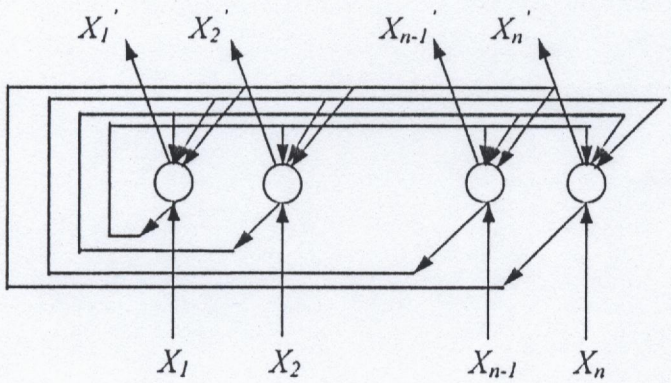


Fig. 3.5 : Réseau de Hopfield

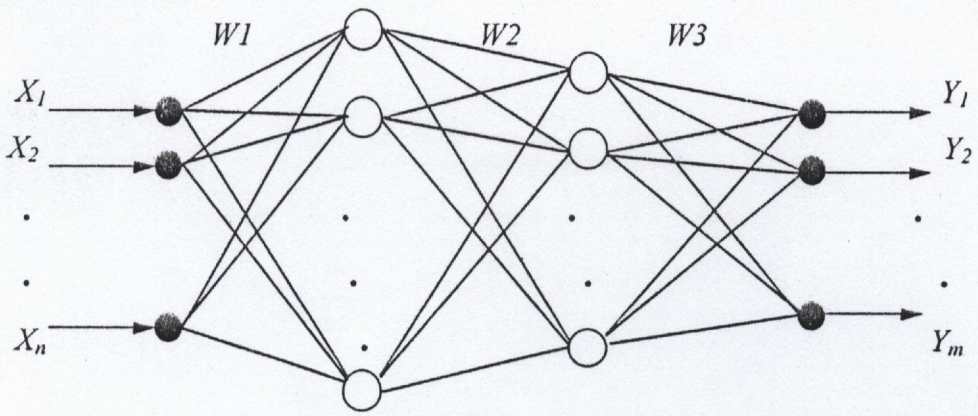


Fig. 3.6 : Structure d'un réseau statique

c) Entraînement des réseaux de neurones

L'application des réseaux de neurones comprend typiquement deux phases : une phase d'apprentissage et une phase d'opération .

L'apprentissage est un processus à travers lequel les paramètres (les poids des connexions) du réseau sont ajustés pour refléter l'information contenue dans la structure du réseau .

Une fois le réseau est entraîné, il représente une base de connaissance statique qui peut être appelée durant la phase d'opération .

Il existe trois types d'apprentissage: apprentissage supervisé ,apprentissage non supervisé et apprentissage par renforcement [39] .

Dans l'apprentissage supervisé, le réseau est alimenté par les valeurs d'entrée et les valeurs de sortie désirées, le réseau ajuste alors ses poids en se basant sur l'erreur de la sortie calculée .

Dans l'apprentissage non supervisé, le réseau est seulement alimenté par les valeurs d'entrée et le réseau ajuste les poids en se basant uniquement sur les valeurs d'entrée et la sortie courante du réseau.

Dans l'apprentissage par renforcement, le réseau reçoit des évaluations scalaires implicites des entrées précédentes.

d) L'algorithme de la retropropagation

La rétropropagation est la méthode la plus utilisée pour l'entraînement des réseaux de neurones, c'est un algorithme d'apprentissage supervisé qui consiste à ajuster les poids des connexions du réseau en utilisant simplement des données d'entrée/sortie. Cet algorithme

utilise la technique de recherche du gradient pour minimiser une fonction coût J qui est égale à la moyenne de l'erreur quadratique entre la sortie désirée et la sortie actuelle du réseau.
L'algorithme de la retropropagation est donné par :

$$W(k+1) = W(k) - \eta.\frac{dJ}{dW(k)} \tag{3.6}$$

avec : W est le vecteur des poids

η est le taux d'apprentissage

J est la fonction coût définie par :

$$J = \frac{1}{2}\sum_{j}\left(d_j - y_j\right)^2 \tag{3.7}$$

où d_j est la sortie désirée

y_j est la sortie du réseau

Bien que cet algorithme soit le plus utilisé, il présente deux problèmes :

- convergence lente .
- l'estimation des paramètres risque de tomber dans un minimum local du critère d'optimisation.

III.2.1.2 Les réseaux de neurones flous

a) Définition

Les systèmes neuronaux flous combinent la théorie des réseaux de neurones artificiels (ANN) et les systèmes flous. La méthode d'apprentissage utilisée pour l'entraînement des réseaux de neurones permet à ces systèmes d'apprendre à partir de la présentation d'un ensemble de données (entrées/sorties) d'entraînement. Cet apprentissage consiste à ajuster les poids des connexions reliant les différentes couches et permet un ajustement très précis des paramètres des fonctions d'appartenance des ensembles flous incorporés dans le réseau neuronal. D'autre part, la théorie des ensembles flous permet aux systèmes neuronaux flous de présenter l'information apprise sous une forme plus compréhensible à l'être humain.

b) Les différentes approches des FNN

Plusieurs méthodes de combiner les systèmes flous et les réseaux de neurones sont reportées dans la littérature [22,23,26,27,36,41,42].

Différentes stratégies d'apprentissage ont été utilisées dans ces applications tel que apprentissage supervisé [36] et apprentissage par renforcement [16,43].

Ces méthodes peuvent être classé suivant la relation recherchée entre le système d'inférence flou et le réseau neuronal en deux approches distinctives :

- Approche fonctionnelle : dans cette approche le système neuronal-flou est considéré comme étant un réseau neuronal ordinaire qui est désigné pour approximer l'algorithme de contrôle flou [44].
- Approche structurelle : elle consiste à réaliser le processus de raisonnement et d'inférence flou à travers la structure d'un réseau connexioniste [45].

Récemment, une nouvelle approche a été développée et consiste à construire des réseaux connexionistes contenant des neurones particuliers dits neurones-flous capables d'implémenter les opérations de base rencontrées dans la théorie des ensembles flous tel que ET et OU [25,46].

c) Les architectures des FNN

- **Les FNN réguliers**

Dans cette classe, les neurones réalisent simplement des opérations arithmétiques tel que addition et multiplication. L'approche la plus directe consiste à implémenter le système d'inférence dans un réseau neuronal multicouche [41]. la figure (3.7) montre la configuration d'un FNN qui implémente un système flou décrit par des règles d'inférence du type :

R^i : **SI** x_1 *est* A_1^i **ET** ... **ET** x_n *est* A_n^i **ALORS** y_1 *est* w_1^i **ET** ... **ET** y_m *est* w_m^i

avec :

A_p^i : les variables linguistiques d'entée.

W_q^i : nobres réels de sortie.

Le réseau possède en total quatre couches ;une couche d'entrée, la deuxième couche contient des noeuds qui représentent les fonctions d'appartenance ($\mu_{A_p^i}$) associées aux variables linguistiques (A_p^i), dans la troisième couche, chaque noeud i calcule la valeur de vérité (μ_i) de la $i^{éme}$ règle par :

$$\mu_i = \prod_{p=1}^{n} \mu_{A_p^i}(x_p) \qquad (3.8)$$

Ce noeud représente une règle floue, et enfin une couche de sortie. Les connexions entre la troisième et quatrième couche sont pondérées avec les valeurs w_q^i.

Les sorties du réseau sont données par :

$$y_q = \frac{\sum_i \mu_i . w_q^i}{\sum_i \mu_i} \quad , \qquad q=1 \ldots m \tag{3.9}$$

L'apprentissage du réseau permet d'ajuster les paramètres des fonctions d'appartenance ainsi que les poids (w_q^i). L'algorithme d'apprentissage utilisé est celui de la rétropropagation.
D'autres approches considèrent les FNN comme étant des réseaux de neurones ordinaires avec des signaux d'entrée flous (des ensembles flous) et des poids flous [25]. Leur architecture est similaire à celle de la figure (3.8) où les entrées $\overline{X_1}, \overline{X_2}$, les poids w_{ik}, v_k et la sortie $\overline{Y}$ sont tous flous.
l'entrée du neurone caché j est donnée par :

$$\overline{I_j} = \overline{X_1}.\overline{W_{1j}} + \overline{X_2}.\overline{W_{2j}} \quad , \qquad 1 \le j \le k \tag{3.10}$$

La barre indique qu'il s'agit de nombres flous.

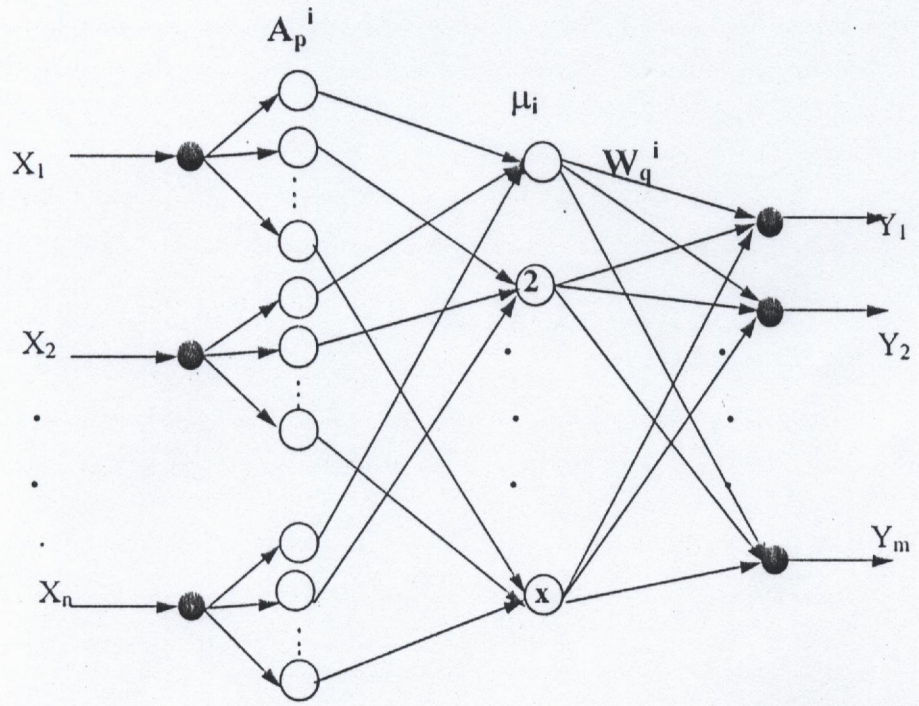


Fig. 3.7 : Configuration d'un FNN régulier

La sortie du j^{ème} neurone caché est donnée par :

$$\overline{Z_j} = f(\overline{I_j}) \quad , \qquad 1 \leq j \leq k \tag{3.11}$$

où f est la fonction d'activation qui est en général une fonction sigmoïdale.

L'entrée du neurone de sortie est alors :

$$\overline{I_0} = \overline{Z_1}.\overline{V_1} + ... + \overline{Z_k}.\overline{V_k} \tag{3.12}$$

et la sortie finale du réseau sera :

$$\overline{Y} = f(\overline{I_0}) \tag{3.13}$$

Une variante de cette approche a été introduite par Tanaka et ses collègues [47], ils ont proposé une architecture d'un réseau neuronal flou avec des poids et des biais flous avec des fonctions d'appartenance triangulaires (ou trapézoïdales). Les entrées et sorties du réseau sont fuzzifiées utilisant des nombres flous représentés par 'α-level sets'.

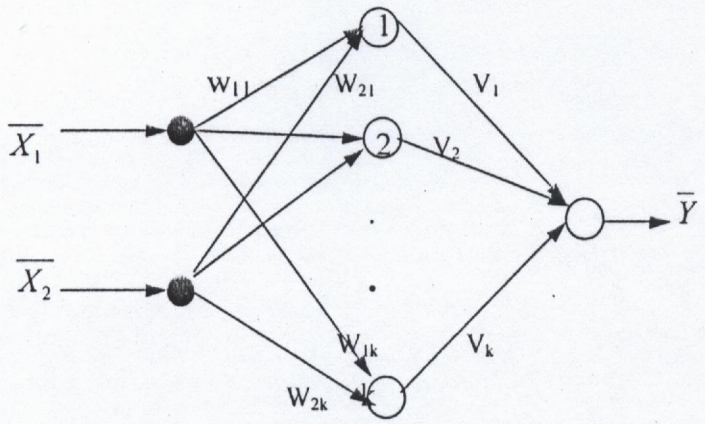


Fig. 3.8 : Configuration d'un FNN avec des entrées, poids et sorties flous

Pour l'entraînement du réseau, ils ont développé l'algorithme de la rétropropagation standard basé sur 'α-cut'(α-cut based backpropagation) [25] pour qu'il puisse s'adapter aux poids et biais flous, mais cette méthode n'est pas applicable si d'autres formes des fonctions d'appartenance autre que triangulaires (ou trapézoïdales) sont utilisées.

- **Les FNN hybrides**

Cette approche est basée sur l'utilisation des neurones-flous capables d'accomplir les opérations floues fondamentales tel que ET et OU. Les entrées du neurone-flou sont des ensembles flous et sont connectés à travers des poids réels ou flous. L'opérateur de conjonction (ET) et de disjonction (OU) sont réalisés utilisant la norme-t, la conorme-t ou la norme-s. Le réseau neuronal-flou est organisé en cinq couches : une couche d'entrée, une couche de fuzzification, une couche de neurones ET, une couche de neurones OU et une couche de defuzzification.

Puisqu'il y a plusieurs possibilités de réaliser les opérateurs logiques ET et OU, différentes structures neuronales-floues ont été développées, on va présenter par la suite deux structures :

Dans la première structure, on va considérer l'implémentation des opérateurs ET et OU par les normes (s-t) [46], dans la seconde ces opérateurs sont réalisés par les opérations Min et Max respectivement [36].

■ **Première structure**

Dans cette structure dite FLP (Fuzzy Logic Processor), l'implémentation structurelle d'un système d'inférence flou dans un réseau neuronal multicouche est basée sur les présentations par Pedrycz [46].

Les opérateurs ET et OU utilisés par ce réseau sont réalisés par les normes t-s comme suit :

La fonction OU : $\hat{u} = OU(X,W)$

est définie en utilisant les normes t et s :

$$\hat{u} = \underset{i=1}{\overset{I}{S}}\left[x_i t w_i\right] \tag{3.14}$$

où $X=[x_i]$, $W=[w_i]$;i=1,2,..,I sont la fonction d'entrée et les vecteurs des poids réels respectivement, et $\hat{u}$ est la fonction scalaire de sortie (Fig. 3.9).

En choisissant la norme-t comme le produit (atb = a.b) et la norme-s comme la somme algébrique (asb = a+b-a.b), alors la fonction OU sera donnée par :

$$\hat{u} = 1 - \prod_{i=1}^{I}\left(1 - x_i.w_i\right) \tag{3.15}$$

La fonction ET : $\hat{u} = ET(X,W)$

est définie de la même façon en utilisant les normes t-s mais dans l'ordre inverse :

$$\hat{u} = \overset{I}{\underset{i=1}{T}}\left[x_i sw_i\right] \tag{3.16}$$

Les normes t et s étant le produit et la somme algébrique respectivement, la fonction ET peut s'écrire comme :

$$\hat{u} = \prod_{i=1}^{I}\left(x_i + w_i - x_i w_i\right) \tag{3.17}$$

Un processeur logique peut être construit en utilisant ces deux neurones comme le montre la figure (3.9), où les $\hat{Z}_i^{L-1}$ sont les sorties des neurones de la couche (L-1) et l'élément non linéaire peut être une fonction sigmoïdale f modifiée par les paramètres m et b :

$$f(\hat{u}) = \frac{1}{1+e^{-(\hat{u}-m)b}} \tag{3.18}$$

et $\hat{Z}_n^L$ représente la sortie du neurone n de la couche L

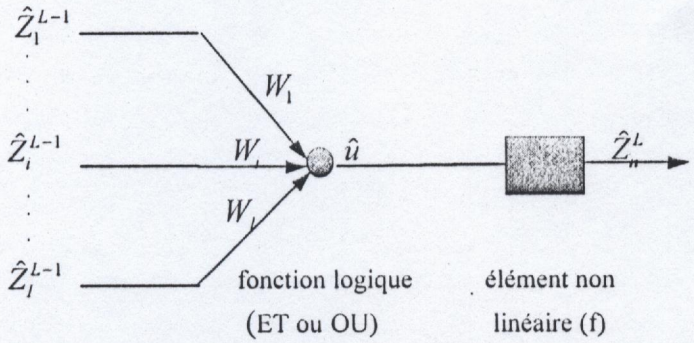


Fig. 3.9 : Processeur logique

Un réseau est formé donc de processeurs logiques interconnectés comme l'indique la figure (3.10), où la structure est composée d'une couche d'entrée (signaux d'entrée réels), d'une couche de fuzzification qui utilise des fonctions d'appartenance de forme triangulaire, d'une couche cachée de neurones ET suivit par une couche cachée de neurones OU et une couche de defuzzification qui sert à convertir les sorties en valeurs déterminées utilisant la méthode du centre de gravité. Le signal d'erreur utilisé pour l'entraînement du réseau est obtenu à partir de la différence entre la sortie désirée et la sortie du réseau. La méthode de la rétropropagation est utilisée pour ajuster les poids des neurones ET et OU, il n y a pas d'ajustement des ensembles flous d'E/S. Une fois le réseau converge, on peut extraire les règles d'inférence [27].

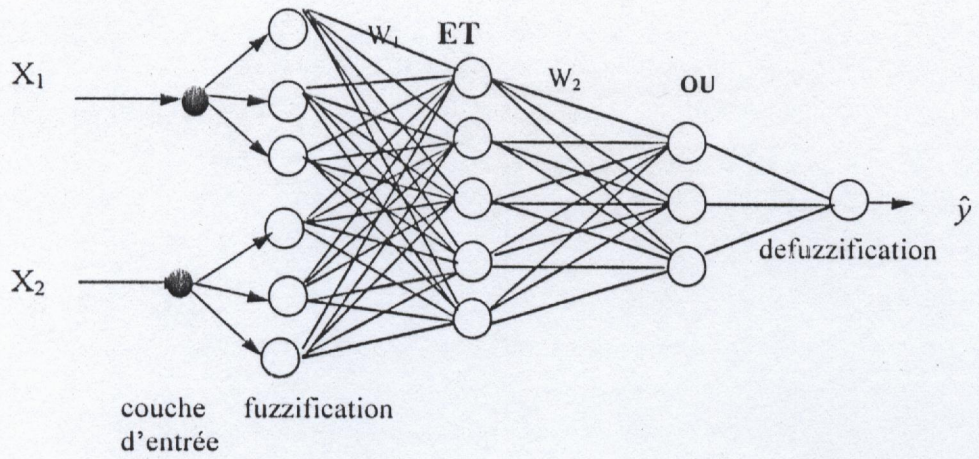


Fig. 3.10 : Architecture d'un FLP avec

- **2 entrées/1 sortie**
- **3 ensembles flous pour les E/S**
- **5 neurones ET et 3 neurones OU**

■ **deuxième structure**

Dans cette structure, un réseau connexioniste à cinq couches est considéré pour la réalisation du système d'inférence flou, ce réseau contient des neurones implémentant les opérations Min (ET) et Max (OU).

Hayashi et al.[48] ont utilisé un FNN avec des entrées/sorties réelles, des poids flous et des fonctions d'appartenance triangulaires pour modéliser un contrôleur flou spécifié par la table des règles donnée par le tableau (3.1) Les A_i, B_i et C_i étant les ensembles flous triangulaires associés à l'erreur (e), la variation de l'erreur (Δe) et l'action y du contrôleur respectivement.

Ce contrôleur flou a comme entrées (e) et (Δe), étant données les valeurs de ces dernières, les neuf règles sont évaluées comme suit :

$\Delta 1 = \text{Min}\ (B_1(e),\ A_2(\Delta e))$

$\Delta 2 = \text{Min}\ (B_2(e),\ A_2(\Delta e))$

.

.

$\Delta 9 = \text{Min}\ (B_5(e),\ A_4(\Delta e))$

e \ Δe	A_1	A_2	A_3	A_4	A_5
B_1		(1) C_1			
B_2		(2) C_1		(3) C_2	
B_3	(4) C_2		(5) C_3		(6) C_4
B_4		(7) C_4		(8) C_5	
B_5				(9) C_5	

Tab.3.1 : Règles de contrôle floues d'un procédé industriel (9règles floues)

Et puisqu'on a neuf règles et seulement cinq actions de contrôle, on maximise la Δi correspondante à la même action C_k comme suit :

ε_1=Max (Δ1, Δ2), ε_2=Max (Δ3, Δ4), ε_3= Δ5, ε_4=Max (Δ6, Δ7), ε_5=Max (Δ8, Δ9),

Alors, chaque ε_k est assigné à son C_k, $1 \leq k \leq 5$

Pour defuzzifier le résultat, on calcule d'abord :

$$C = \text{Max} (\varepsilon_k . C_k) \ , \quad 1 \leq k \leq 5$$

Ensuite trouver δ qui est égal au centre de gravité de C, et alors δ est la sortie defuzzifiée du contrôleur.

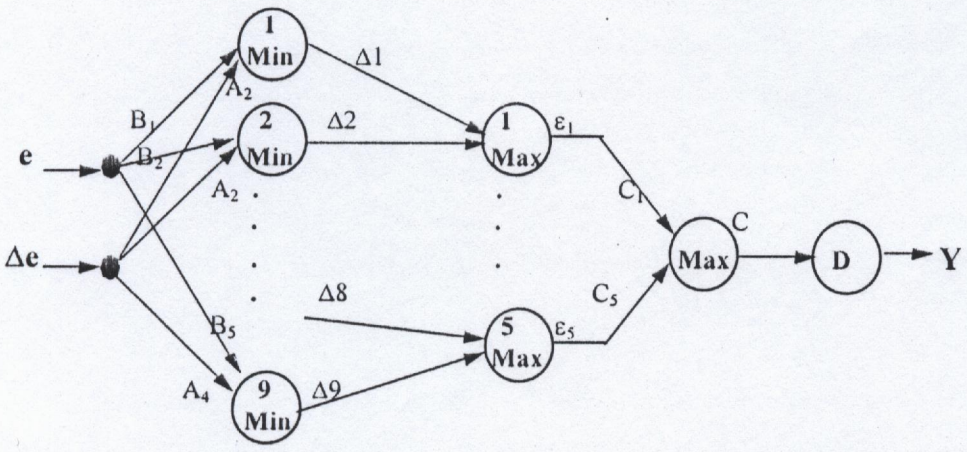


Fig. 3.11 : Structure d'un FNN comme contrôleur flou

Le contrôleur flou modélisé par un FNN est indiqué à la figure (3.11), les noeuds de la couche ET (Min) réalisent le Min de leurs entrées, l'interaction des signaux e et Δe et les

poids flous (B_i et A_i) sert à évaluer B_i en e et A_i en Δe ($A_i(\Delta e)$ et $B_i(e)$). Les neurones de la couche ET (Max) réalisent le Max de leurs entrées, les poids de ces neurones sont tous égaux à un, le neurone suivant prend le Max de ses entrées floues (ε_i. C_i, $1 \leq i \leq 5$). Le dernier neurone a un poids égal à un et assure l'opération de defuzzification.

Ce FNN peut être utilisé à la place d'un contrôleur flou ou bien pour apprendre les règles de contrôle floues (les poids A_i, B_i et C_i) à partir des données d'entraînement. A cause de l'utilisation des opérations Min et Max et les poids flous, ce réseau nécessite un algorithme d'apprentissage spécial basé sur une fonction coût prenant des valeurs floues [48].

Dans [36], Lin et Lu ont proposé une architecture d'un FNN (Fig. 3.12) capable de traiter des informations (signaux d'entrée) réelles ou floues. Le réseau proposé comprend en total cinq couches. L'information floue que peut posséder le réseau est décrite par des nombres flous représentés par 'α-level sets', et peuvent être d'une forme quelconque. La couche d'entrée reçoit des signaux flous ou réels, chaque noeud de cette couche correspond à une seule variable linguistique d'entrée, la cinquième couche consiste en noeuds de sortie dont les sorties sont des nombres flous ou réels, chaque noeud de cette couche correspond à une seule variable linguistique de sortie, les noeuds de la deuxième et quatrième couche définissent les fonctions d'appartenance représentant les termes flous de la variable linguistique respective, chaque noeud de la quatrième couche réalise l'opération Max pour intégrer les règles 'allumées' qui ont la même conséquence. Chaque noeud de la troisième couche représente une règle floue, d'où tous les noeuds de cette couche forment la base de règle floue.

Les noeuds de la deuxième et cinquième couche ont des poids flous, les autres sont numériques, les connexions entre la deuxième et troisième couche définissent les prémises des règles floues et ceux entre la troisième et quatrième couche définissent les conséquences.

Pour l'entraînement du réseau, ils ont développé un algorithme d'apprentissage supervisé qui permet au système proposé d'apprendre les règles de contrôle floues.

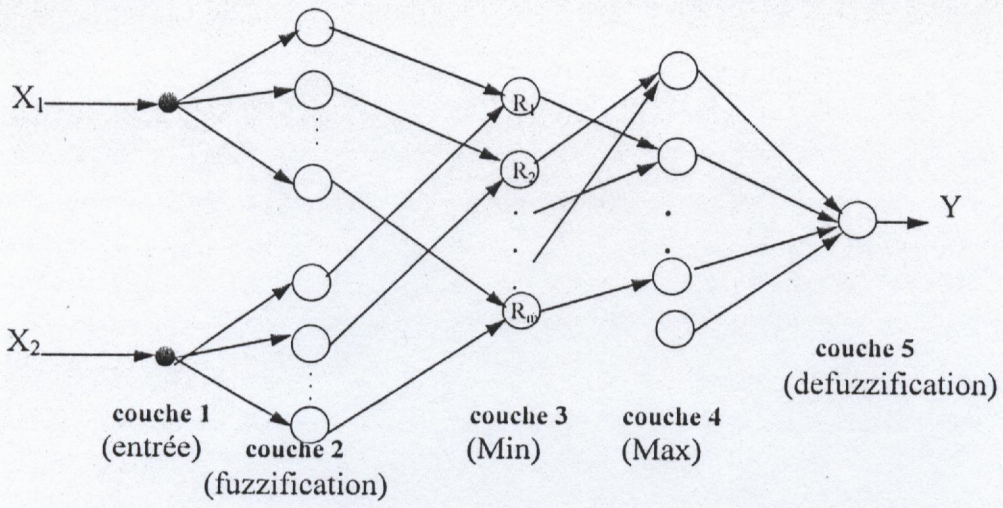


Fig. 3.12 : Structure du FNN proposé dans [36]

III.2.2 L'approche directe

Comme on vient de voir dans les paragraphes précédents, les problèmes à résoudre dans la conception des contrôleurs flous concernent la détermination de l'univers de discours linguistique, la définition des fonctions d'appartenance pour chaque terme linguistique et la dérivation des règles de contrôle [33].

Certains de ces problèmes peuvent être résolus par l'application des algorithmes d'apprentissage, le choix d'un algorithme dépend de la nature du domaine de travail et de l'information disponible. Du à la complexité de l'espace de recherche pour le problème de conception floue, due entre autres à la non dérivabilité et la discontinuité des fonction impliquées, la seule alternative pour résoudre ce type de problème est l'utilisation de méta heuristiques. Parmi celles-ci, les algorithmes génétiques ont été appliqués avec le plus de succès.

L'incorporation de l'apprentissage génétique dans un processus de conception floue ajoute une dimension intelligente au FLC qui lui permet de créer et de modifier ses règles. Les AG offrent la possibilité d'ajuster les fonctions d'appartenance et de générer les règles qui utilisent ces fonctions. L'application des algorithmes génétiques est seulement limitée par la dimension du problème (nombre de chromosomes d'une chaîne).

III.2.2.1 L'application des AG pour les contrôleurs flous :

La difficulté commune dans les systèmes flous est que leurs paramètres doivent être spécifiés par un opérateur (concepteur) humain .

Vu leur application avec succès sur une variété de problèmes d'apprentissage et d'optimisation, les AG ont été proposés comme une méthode d'apprentissage qui permet une génération automatique de paramètres optimaux pour les contrôleurs flous. Pour cela la première étape concerne la définition du problème. Celle ci consiste principalement à définir le ou les paramètres à optimiser, le type de codage des paramètres et la fonction coût.

a) Définition du problème : paramètres à optimiser

Il y a trois approches d'application des AG pour la conception des FLC. Dans un premier cas, les règles linguistiques d'un contrôleur flou conventionnel sont fixées et leurs fonctions d'appartenance sont optimisées par l'AG [31,50]. Dans le second cas, les fonctions d'appartenance de valeurs linguistiques spécifiées sont fixées et l'AG est utilisé pour déterminer un ensemble optimal de règles [49]. Dans la troisième approche les règles et les fonctions d'appartenance sont ajustées simultanément [30,32,33] afin d'accomplir de meilleures performances.

Thrift [49] a examiné la possibilité d'utiliser les AG pour trouver les règles floues. Dans cet article, la synthèse du contrôleur flou est faite sous forme d'une table de vérité.

Karr [50] a examiné l'utilisation d'un AG pour trouver des fonctions d'appartenance de haute performance pour un contrôleur flou, le procédé étudié est celui du 'pole-cart system'.

Dans [31] Karr a utilisé un AG pour produire un FLC adaptatif pour le contrôle de pH en modifiant les fonctions d'appartenance en ligne.

De tels travaux ont montré l'efficacité des AG de créer avec succès les parties individuelles d'un contrôleur flou (l'ensemble des règles et les fonctions d'appartenance), mais l'interdépendance entre ces deux parties suggère qu'une procédure de conception simultanée doit être une méthodologie plus appropriée.

Homaifar et McCormick [32] ont utilisé un AG pour la conception simultanée des fonctions d'appartenance et l'ensemble des règles floues pour un FLC. Ainsi, l'ensemble de règles et les fonctions d'appartenance sont incorporés dans une seule chaîne que l'AG cherche à optimiser

Dans cet article, seulement les longueurs des bases des ensembles flous triangulaires de l'espace d'entrée sont ajustés (ceux de l'espace de sortie sont fixés afin de faciliter le calcul i.e minimisation de la taille de la chaîne),il n'y a pas de localisation des centres.

Linkens et [33] proposent l'optimisation des règles d'inférence et des fonctions d'appartenance triangulaires. La base et le centre de ces fonctions sont variables. Une alternative en ligne est proposée [34].

Zeigler [30] a développé une nouvelle technique, dite HDGA (Hierarchical distributed genetic algorithms) qui est basée sur une stratégie de recherche 'multilevel resolution search strategy' pour la conception d'un FLC optimal en termes de fonctions d'appartenance et de règles d'inférences. Les fonctions d'appartenance peuvent être soit traingulaire soit gaussienne. Le HDGA examine plusieurs alternatives de conception du FLC aussi bien que l'optimisation des ensembles de paramètre correspondant. Cette méthode a été appliquée pour la conception d'un contrôleur flou pour un pendule inversé.

b) Codage des paramètres

Le type de codage utilisé est celui du codage binaire ou entier. où les codes segment de tous les paramètres à optimiser sont combinés pour former une seule chaîne.

- Codage binaire :

Pour illustrer ce type de codage nous présentons le formalisme de [33].

Soit le code du $i^{ème}$ paramètre dans l'espace de recherche R^n de dimension n est désigné par σ^i, si l'alphabet binaire usuel est utilisé, alors σ^i est un ensemble de 0 et de 1 d'une longueur L_i : $\sigma^i \leftrightarrow \{0,1\}^{Li}$.

La structure de chaque chaîne représentant une solution possible du problème d'une population P sera donnée par :

$$A \in P = [\sigma^1\ \sigma^2\ \sigma^3 \ldots\ \sigma^n] \qquad (3.19)$$

La structure d'une règle floue est construite des codes des ensembles flous des antécédents (prémises) et des conséquences linguistiques de la règle. En outre, chaque variable linguistique est définie par sa fonction d'appartenance qui, dans le cas des fonctions d'appartenance triangulaires, est définie par son centre C et sa largeur W (Fig. 3.13).

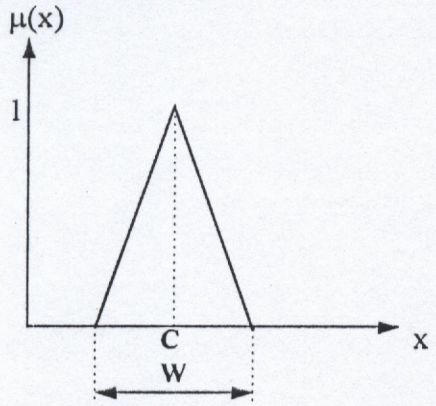


Fig. 3.13 : Fonction d'appartenance triangulaire de centre C et de largeur W

Typiquement, chaque variable linguistique dans une règle floue peut être représentée par trois codes segments : un pour son centre C, un pour sa largeur W et un autre pour sa valeur linguistique (ou nom) L tel que Medium, High, Low, ...

$$\sigma_F = [\ \sigma_L\ \sigma_C\ \sigma_W\ ] \tag{3.20}$$

En considérant une règle floue avec n conditions et m actions, le codage composé d'une règle d'essai $A \in P$ est donnée par :

$$A = [\sigma_{Fc}^{1} \ldots . \sigma_{Fc}^{n} \sigma_{Fa}^{1} \ldots . \sigma_{Fa}^{m}] \tag{3.21}$$

avec σ_{Fc}^{i} ou σ_{Fa}^{i} sont les codes des ensembles flous de la i^{ème} condition ou action.

La figure (3.14) illustre un format de codage typique pour une population de bases de règle floues contenant deux conditions E et CE et une action U. C'est la structure de règles adoptée dans plusieurs applications du contrôle flou, où E représente l'erreur, CE représente la variation de l'erreur (CE ; change in error) et U est l'entrée de contrôle du procédé.

Si l'espace d'entrée est partitionné en sept ensembles flous linguistiques {Negative Big NB, Negative Medium NM, Negative Small NS, Zero ZE, Positive Small PS, Positive Medium PM, Positive Big PB} pour chacune des antécédents de la règle floue, alors la base de règle consiste en 49 règles au maximum.

En représentant le codage total pour la stratégie de contrôle flou sous la forme usuelle d'une table de règles à deux dimensions (Fig. 3.14), le codage des antécédants de la règle E et CE peut être fait implicitement par les positions des règles dans la table. par exemple, toutes les lignes sont associées à une seule valeur de E et toutes les colonnes sont associées à une seule valeur de CE. Donc, des codes explicites sont seulement nécessaire pour les centres et les largeurs des variables floues.

Pour obtenir des attributs quasi-optimaux d'une règle, ses paramètres (les valeurs linguistiques, les centres et les largeurs des ensembles flous) peuvent être modifiés.

Le codage du j^{ième} paramètre dans une chaîne de codes représente une valeur λ_j entre une valeur minimale λ_{min} et maximale λ_{max} prédéterminées donnée par :

$$\lambda_j = \lambda_{min} + (\lambda_{max} - \lambda_{min}) \frac{N(\lambda_j)}{2^{L_j} - 1} \tag{3.22}$$

où Lj est longueur de la chaîne binaire associée à λ_j.

ce / e	NB	NM	NS	ZE	PS	PM	PB
NB	R_1	R_2	...			...	R_7
. . .							. . .
PB							R_{49}

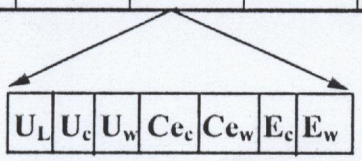


Fig. 3.14 : Codage typique d'une base de règle floue

- Codage entier

Pour coder les paramètres, Homaifar et McCormick [32] ont utilisé un codage spécial qui consiste à utiliser des chaînes à nombres entiers (des nombres compris entre 1 et le nombre d'ensembles flous de l'espace de sortie) au lieu de chaînes binaire, ceci nécessite une nouvelle forme de décodage. La longueur de la chaîne dépend de la taille de l'ensemble de règles et le nombre des ensembles flous utilisés pour partitioner les espaces des variables d'E/S. Ce paradigme a été appliqué sur deux problèmes :'cart-centering system' et 'truck-backing system'.

III.2.2.2 L'apprentissage des systèmes flous par les AG :

Dans l'apprentissage des paramètres du contrôleur flou, l'AG commence par une génération aléatoire d'une population de N chaînes où chaque chaîne complète de code représente une solution possible du problème et comprend un ensemble de règles floues et leurs fonctions d'appartenance.

Chacune de ces chaînes est décodée pour obtenir les paramètres actuels du FLC, ces paramètres sont envoyés au modèle du système à commander, évalués avec une certaine fonction coût tel que l'erreur cumulée à travers le temps, et assignés une valeur d'adaptation 'fitness value'. Cette valeur reflète la qualité de la solution représentée par la chaîne, et peut

être vue comme une mesure qui indique la performance du FLC pour le contrôle du système utilisant les fonctions d'appartenance et les règles associées.

Ces valeurs d'adaptation sont des valeurs sur lesquelles est basée l'application des trois opérateurs (reproduction, croisement et mutation) qui produisent une nouvelle population de chaînes (une nouvelle génération) ;cette nouvelle génération contiendra des paramètres plus performants. les nouvelles chaînes sont décodées, évaluées et transformées en utilisant les opérateurs de base. Ce processus continu jusqu'à ce que la convergence soit accomplie ou à ce qu'une solution optimale soit trouvée.

La configuration d'un tel mécanisme d'apprentissage est indiquée dans la figure (3.15).

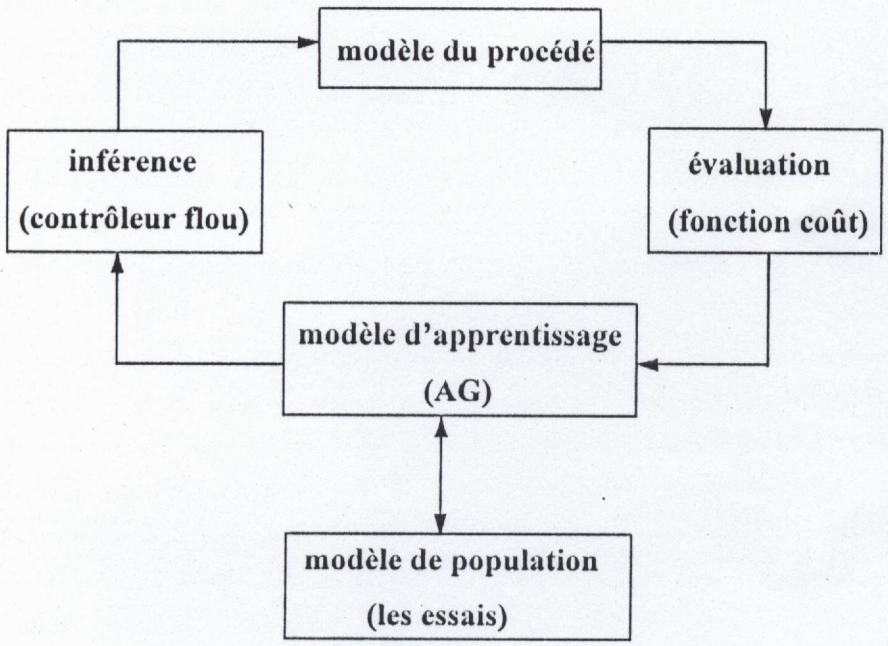


Fig. 3.15 : Mécanisme d'apprentissage off-line

III.3 Conclusion

Les contrôleurs flous sont des candidats optimaux pour la commande des systèmes complexes non identifiés ou partiellement identifiés, qui ne peuvent pas être modélisé facilement d'une manière mathématique.

L'approche traditionnelle pour la conception floue est basée sur les connaissances acquises par des opérateurs experts, bien que cette approche ait prouvé son efficacité dans plusieurs applications, il peut cependant que les opérateurs ne peuvent pas transcrire leurs

connaissances et expériences sous forme de contrôleur à logique floue, en plus le domaine d'expertise n'est pas toujours disponible.

De ce fait, des recherches intensives ont été concentrées vers l'élaboration des méthodes plus systématiques et optimales pour la conception des contrôleurs flous, et ont conduit au développement de deux nouvelles approches d'apprentissage que nous avons étudié dans ce chapitre :

- L'approche connexioniste ;dans ce cas un réseau neuronal flou est utilisé pour implémenter le mécanisme d'inférence flou, l'algorithme de la rétropropagation est alors utilisé pour ajuster les paramètres du réseau et les règles de contrôle sont extraites après convergence.
- L'approche directe ;elle consiste à utiliser un algorithme d'apprentissage pour la conception d'un système flou qui consiste à trouver les fonctions d'appartenance et les règles d'inférence tout en minimisant une certaine fonction coût. Du à la complexité de l'espace de recherche, la méthode d'optimisation requise est celle des algorithmes génétiques.

Conception d'un contrôleur flou par les AG

IV.1 Introduction.

IV.2 Implémentation du contrôleur flou.

IV.3 Description de l'algorithme d'apprentissage.

IV.3.1 Codage des paramètres.

IV.3.2 Mécanisme de l'AG.

a) Reproduction.

b) Croisement.

c) Mutation.

d) La sélection des individus d'une nouvelle génération.

IV.3.3 Décodage des paramètres.

IV.4 Interaction du module contrôleur avec l'AG.

IV.5 Conclusion.

IV.1 Introduction

Dans le chapitre précédent, nous avons présenté plusieurs techniques systématiques de conception des contrôleurs à logique floue. Ces techniques peuvent être classées en deux approches :approche connexioniste et approche directe,.

Dans l'approche connexioniste, l'apprentissage du contrôleur flou est réalisé à l'aide d'un réseau neuronal. L'approche directe utilise généralement, les algorithmes génétiques, une technique d'optimisation qui fonctionne en manipulant les codes des paramètres du système sans connaissance explicite de l'espace de recherche. L'utilisation de ces techniques implique le codage des paramètres de conception, nécessitant ainsi un décodage explicite lors de l'application de la loi de commande. De plus la longueur de la chaîne (string) codant les paramètres peut être inhibitive en terme de temps de calcul et d'espace mémoire, amenant certains auteurs à limiter les paramètres de conception ou utiliser des codages spéciaux.

Dans ce chapitre, nous présentons une nouvelle approche de conception (apprentissage) du contrôleur flou, c'est une approche mixte qui réside dans la combinaison des deux paradigmes: les réseaux de neurones et les AG afin d'assurer une conception optimale d'un FLC. Dans cette approche, on a considéré l'implémentation structurelle d'un système d'inférence flou de type TPE dans un réseau multicouche, les grandeurs ajustables de ce réseau sont les largeurs des différents univers de discours et les poids des connexions. Ainsi l'apprentissage du contrôleur consiste à déterminer les valeurs optimales de ces grandeurs qui minimisent une certaine fonction coût sous certaines contraintes de connectivité du réseau. Les caractéristiques de ce problème font que les méthodes d'optimisation classiques ne peuvent pas être utilisées, pour cela on utilise les algorithmes génétiques.

IV.2 Implémentation du contrôleur flou

Dans cette section, on va considérer l'implémentation structurelle d'un contrôleur flou dans un réseau multicouche.

Le réseau qu'on a adopté consiste en cinq couches (Fig. 4.1), une couche d'entrée, une couche de fuzzification, une couche de neurones AND, une couche de neurones OR et une couche de défuzzification. Chaque neurone de ces couches réalise une fonction particulière sur ses signaux d'entrée utilisant un ensemble de paramètres spécifique à ce neurone.

Le choix de cette fonction dépend de la fonction totale que le réseau est sensé réaliser.

L'architecture du réseau (Fig. 4.1) comporte les couches suivantes :

- **Couche d'entrée (input layer)**, le nombre de neurones de cette couche correspond au nombre de variables (réels) d'entrée du contrôleur. Aucune opération n'est effectuée au niveau de cette couche.
- **Couche de fuzzification (fuzzification layer)**, cette couche permet la décomposition de l'univers de discours pour chaque variable d'entrée en un ensemble fixe de sous ensembles flous selon la stratégie TPE.

  Chaque neurone i de cette couche reçoit une seule entrée soit x à partir de la couche d'entrée et génère une sortie donnée par :

$$Out_i^F = \mu_{Ai}(x) \tag{4.1}$$

  où Ai est la valeur linguistique (LOW,MEDIUM,HIGH,...) associée à ce neurone, en d'autre terme, Out_i^F est la fonction d'appartenance de Ai qui indique à quel degré une entrée x donnée satisfait Ai. Les poids de cette couche sont tous égaux à un.
- **Couche AND (AND layer)**, chaque neurone de cette couche représente une règle floue, donc cette couche comprend un nombre de neurones AND égal au nombre de règles.

  Chaque neurone reçoit à partir de la couche de fuzzification un nombre d'entrée égal au nombre de conditions (antécédents) de la règle, ces entrées représentent les fonctions d'appartenance.

  L'opération logique AND réalisée par ce neurone est implémentée par l'opérateur norme-T défini par :

$$T(x,y) = x.y \tag{4.2}$$

  ainsi la sortie du $i^{ème}$ neurone est donnée par :

$$Out_i^{AND} = \mathop{T}_{j=1}^{n} Out_j^F \quad , \ i = 1,...,Na \tag{4.3.a}$$

ou encore :

$$Out_i^{AND} = \prod_{j=1}^{n} Out_j^F \quad , \ i = 1,...,Na \tag{4.3.b}$$

avec : Na : le nombre de neurones AND.

n : le nombre des conditions de la règle.

Out_j^F : les sorties floues de la couche de fuzzification qui représentent les degrés d'appartenance des antécédents de la $i^{ème}$ règle.

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

- **Couche OR (OR layer)**, chaque neurone de cette couche reçoit toutes les sorties des neurones de le couche AND et réalise l'opération logique OR en utilisant les opérateurs norme-T et norme-S définis par :

$$T(x,y) = x.y \tag{4.4}$$

$$S(x,y) = x + y - x.y \tag{4.5}$$

La sortie du $j^{ème}$ neurone est donnée par :

$$Out_j^{OR} = \overset{Na}{\underset{i=1}{S}}\left(T(Out_i^{AND}, Wij)\right) \quad , \quad j=1,...,No \tag{4.6}$$

d'où :

$$Out_j^{OR} = 1 - \prod_{i=1}^{Na}\left[1 - Out_i^{AND}.Wij\right] \quad , \quad j=1,...,No \tag{4.7}$$

avec : Wij : les poids des connexions entre le $i^{ème}$ neurone AND et le $j^{ème}$ neurone OR.

No : le nombre de neurones OR.

Le nombre de neurones de la couche OR est égal au nombre de sous ensembles flous utilisés pour partitioner l'univers de discours de l'espace de sortie.

Les poids de cette couche sont ajustables et ne peuvent prendre que deux valeurs possibles soit zéro ou bien un et ils sont assujettis à la contrainte suivante :

$$\sum_{j=1}^{N0} Wij = 1 \,, \forall\ i = 1,..,Na \tag{4.8}$$

Cela signifie que parmi les poids connectant le $i^{ème}$ neurone AND aux différents neurones OR, un seul poids est égal à un et les autres sont égaux à zéro.

Physiquement cette contrainte signifie que pour des valeurs linguistiques données des antécédents d'une règle, une seule valeur linguistique est associée à l'action (conséquence) de cette règle.

- **Couche de défuzzification (defuzzification layer)**, le neurone de cette couche réalise l'opération de défuzzification selon la règle du centre de gravité et la méthode TPE à savoir :

$$\hat{y} = \frac{\sum_{i=1}^{No} C_i . Out_i^{OR}}{\sum_{i=1}^{No} Out_i^{OR}} \tag{4.9}$$

où $\hat{y}$ est valeur numérique de sortie, Out_i^{OR} sont les sorties des neurones de la couche OR (les valeurs d'appartenance de l'interface de sortie)et C_i sont les centres des fonctions d'appartenance symétriques de la sortie.

Les centres C_i peuvent être considérés comme des poids des connexions entre la couche OR et la couche de défuzzification.

La figure (4.1) illustre un exemple d'un tel réseau avec des univers de discours divisés en trois sous ensembles flous LOW (L), MEDIUM (M) et HIGH (H) pour toutes les variables d'E/S. Le réseau possède deux entrées et une sortie, avec neuf neurones AND (9 règles floues i.e toutes les combinaisons possibles des variables linguistiques des entrées) et trois neurones OR.

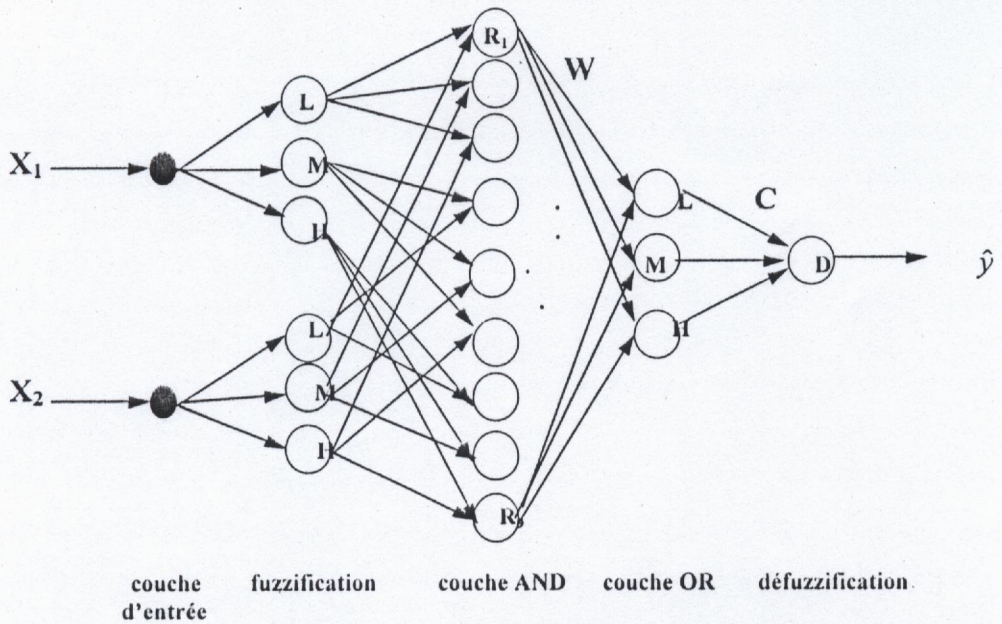


Fig. 4.1 : Architecture du réseau contrôleur

Les règles que ce réseau réalise sont de la forme :

$$R^i : Si\ (x_1\ est\ \{L,M\ ou\ H\})\ et\ (x_2\ est\ \{L,M\ ou\ H\})\ Alors\ (y\ est\ \{L,M\ ou\ H\})$$

exemple : La règle R^3 est donnée par :

$$R^3 : Si\ (x_1\ est\ L\ et\ x_2\ est\ H)\ Alors\ (y\ est\ \{L,M\ ou\ H\})$$

en prenant par exemple $W3=[w_{31}\ w_{32}\ w_{33}]=[0\ 1\ 0]$, alors la règle R^3 deviendra :

$$R^3 : Si\ (x_1\ est\ L\ et\ x_2\ est\ H)\ Alors\ (y\ est\ M)$$

IV.3 Description de l'algorithme d'apprentissage

L'idée de base de notre travail consiste à assurer une conception simultanée des fonctions d'appartenance et l'ensemble de règles pour un contrôleur flou que nous avons projeté sur un réseau multicouche.

Pour cela, les grandeurs ajustables sont les suivantes :

- Les largeurs (L) des différents univers de discours (Fig. 4.2) des variables d'E/S, cet ajustement permet d'altérer les fonctions d'appartenance associées.
- Les poids (W) du réseau connectant la couche AND et la couche OR, l'ajustement de ces poids permet de modifier les règles de contrôle floues.

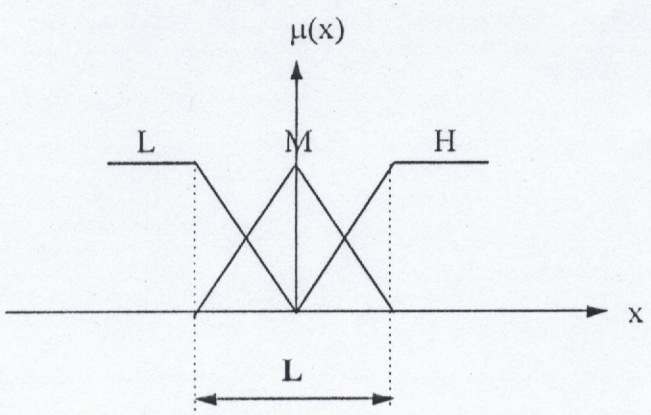


Fig. 4.2 : Univers de discours (TPE) de largeur L

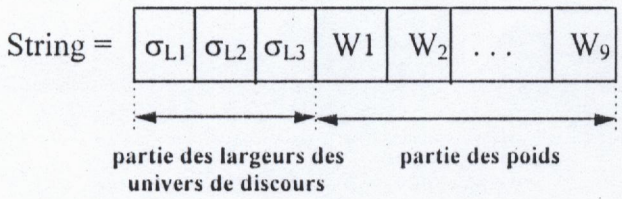
Ainsi l'apprentissage du contrôleur consiste à déterminer les largeurs des univers de discours et les poids qui minimisent une certaine fonction coût et tel que les contraintes (4.8) soient satisfaites.

Les caractéristiques de ce problème font que les méthodes d'optimisation classiques ne peuvent pas être utilisées, pour cela on utilise l'algorithme génétique.

IV.3.1Codage des paramètres

Des chaînes binaires d'une longueur donnée sont utilisées pour coder les largeurs des univers de discours. On n'a pas besoin de coder les poids puisqu'ils prennent des valeurs binaires (0 ou 1). Donc chaque chaîne complète de la population contient les valeurs des poids et les codes des différentes largeurs des univers de discours.

Ainsi pour le contrôleur flou représenté par le réseau de la figure (4.1), la chaîne aura la forme suivante :



où $\sigma_{L_1}, \sigma_{L_2}, \sigma_{L3}$: les codes des largeurs des univers de discours (L_1,L_2,L_3) des entrées X_1 et X_2 et de la sortie $\exists$ respectivement.

W_i :vecteur des poids connectant le $i^{ème}$ neurone AND aux différents neurones OR :

$W_i = [w_{i1}\ w_{i2}\ w_{i3}]$, $i = 1 .. 9$

IV.3.2 Mécanisme de l'AG

L'AG commence par une génération aléatoire d'une population de N chaînes où chaque chaîne doit satisfaire la contrainte (4.8) (initialisation conditionnée). De nouvelles générations sont crées en utilisant les opérations génétiques :reproduction, croisement et mutation.

En vue de respecter la contrainte citée ci-dessus, on a introduit un certain nombre d'opérations supplémentaires lors du croisement et de la mutation. Ces opérations consistent en des perturbations qui n'affectent pas leur aspect aléatoire.

a) Reproduction

Dans notre travail deux méthodes de sélection des individus de la prochaine génération ont été considérées : la sélection par roue de loterie (sélection proportionnelle) et

la sélection à reste stochastique.

b) Croisement

Un croisement ordinaire entre deux chaînes parents aura lieu si la position ou se produira le croisement se trouve à l'intérieur de la partie de la chaîne représentant les largeurs des différents univers de discours.

Dans le cas ou le point du croisement se trouve à l'intérieur de la partie de la chaîne qui représente les poids, le croisement consistera à échanger l'ensemble des poids (le vecteur W_i) correspondant à ce point, tout en satisfaisant la contrainte cela signifie qu'une seule règle sera changée.

En considérant le réseau contrôleur de la figure (4.1), le processus du croisement conditionné est illustré dans la figure (4.3).

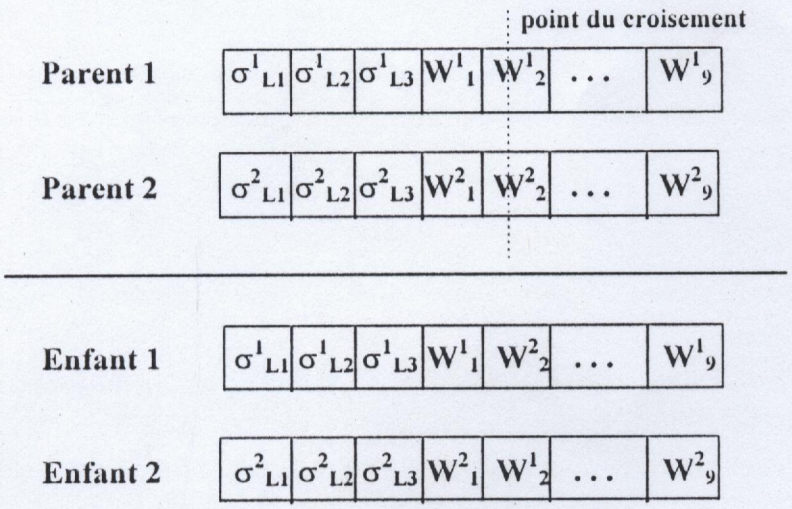


Fig. 4.3 : Principe du croisement conditionné

c) Mutation

De la même manière, si une mutation a lieu et la position du bit à muter se trouve dans la partie des largeurs des univers de discours de la chaîne, alors la valeur de ce bit sera complémenté (mutation ordinaire).

Dans le cas ou la position du bit à muter se trouve dans la partie des poids, alors l'ensemble des poids à qui appartient ce bit sera transformé à un autre ensemble de poids valide (vérifiant la contrainte).

exemple : Considérons toujours le contrôleur de la figure (4.1), le processus de la mutation conditionnée est indiqué dans la figure (4.4) où le vecteur $W_2 = [0\ 1\ 0]$ sera transformé à l'ensemble [0 0 1] ou [1 0 0].

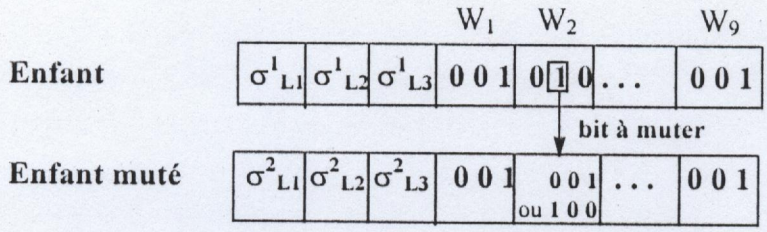


Fig. 4.4 : Principe de la mutation conditionné

d) La sélection des individus d'une nouvelle génération

Dans notre travail on a considéré deux stratégies de sélection finale :

- la sélection par compétition.
- steady state selection.

IV.3.3 Décodage des paramètres

Chaque chaîne doit être décodée afin d'obtenir les paramètres actuels du contrôleur et de pouvoir calculer la valeur de la fonction coût qui lui est associée.

Le décodage binaire est utilisé et concerne seulement les largeurs des différents univers de discours, les valeurs des poids sont prises telles quelles.

IV.4 Interaction du module contrôleur avec l'AG

Le problème qui apparaît dans l'optimisation est celui de l'utilisation des modèles mathématiques pour évaluer le coût d'une chaîne donnée.

L'un des plus fort points des contrôleurs flous est le fait qu'ils n'ont pas besoin de tels modèles. Cependant, pour obtenir un coût d'un contrôleur donné, l'AG doit avoir un modèle du système à commander pour évaluer la performance du contrôleur. Ce modèle peut être un modèle mathématique linéaire ou non linéaire, un modèle neuronal ou un modèle 'flou', l'essentiel est qu'il permette de simuler le comportement du système.

La figure (4.5) illustre les interactions de l'AG du module du réseau contrôleur et le modèle de simulation du procédé.

La figure (4.6) montre un organigramme détaillé des opérations; au début de la simulation, l'AG génère une population initiale d'individus où chaque individus représente les paramètres du réseau contrôleur. Ces individus sont envoyés au module contrôleur pour initialiser un certain ensemble de fonctions d'appartenance (largeurs des univers de discours) et de règles floues (les poids), le contrôleur envoie donc des commandes opérationnelles au modèle de simulation et la trajectoire du procédé est échantillonnée pour calculer la performance du contrôleur qui considère la minimisation des composants suivants :l'erreur totale entre la trajectoire désirée et la trajectoire actuelle et l'énergie utilisée pour contrôler le procédé.

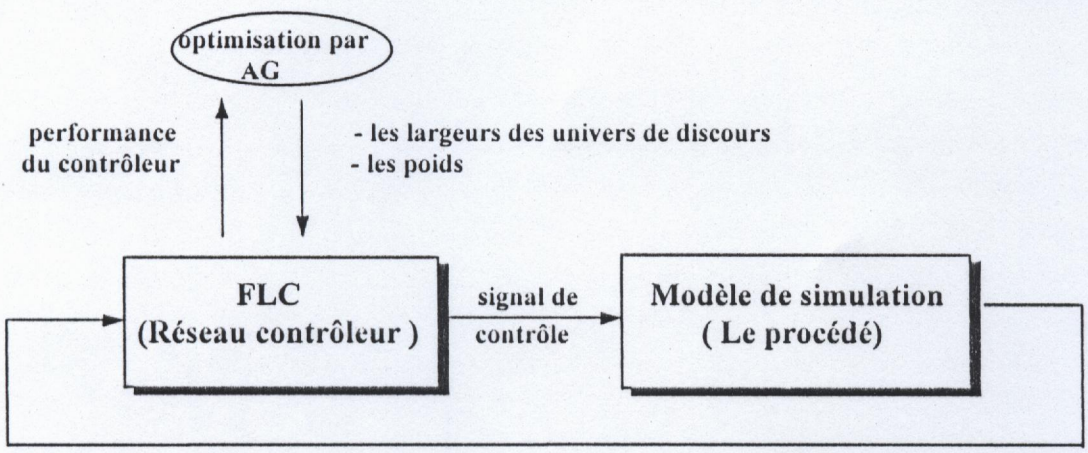


Fig. 4.5 : Diagramme des interactions de l'AG, FLC et le modèle de simulation

Après l'évaluation de tous les individus de cette population, une nouvelle génération sera crée par l'AG en utilisant les opérateurs génétiques, de même les chaînes de cette nouvelle population seront décodées puis évaluées, ce processus continu jusqu'à ce que la convergence soit accomplie. le critère de convergence qu'on a choisi est fixé par le nombre maximale de générations.

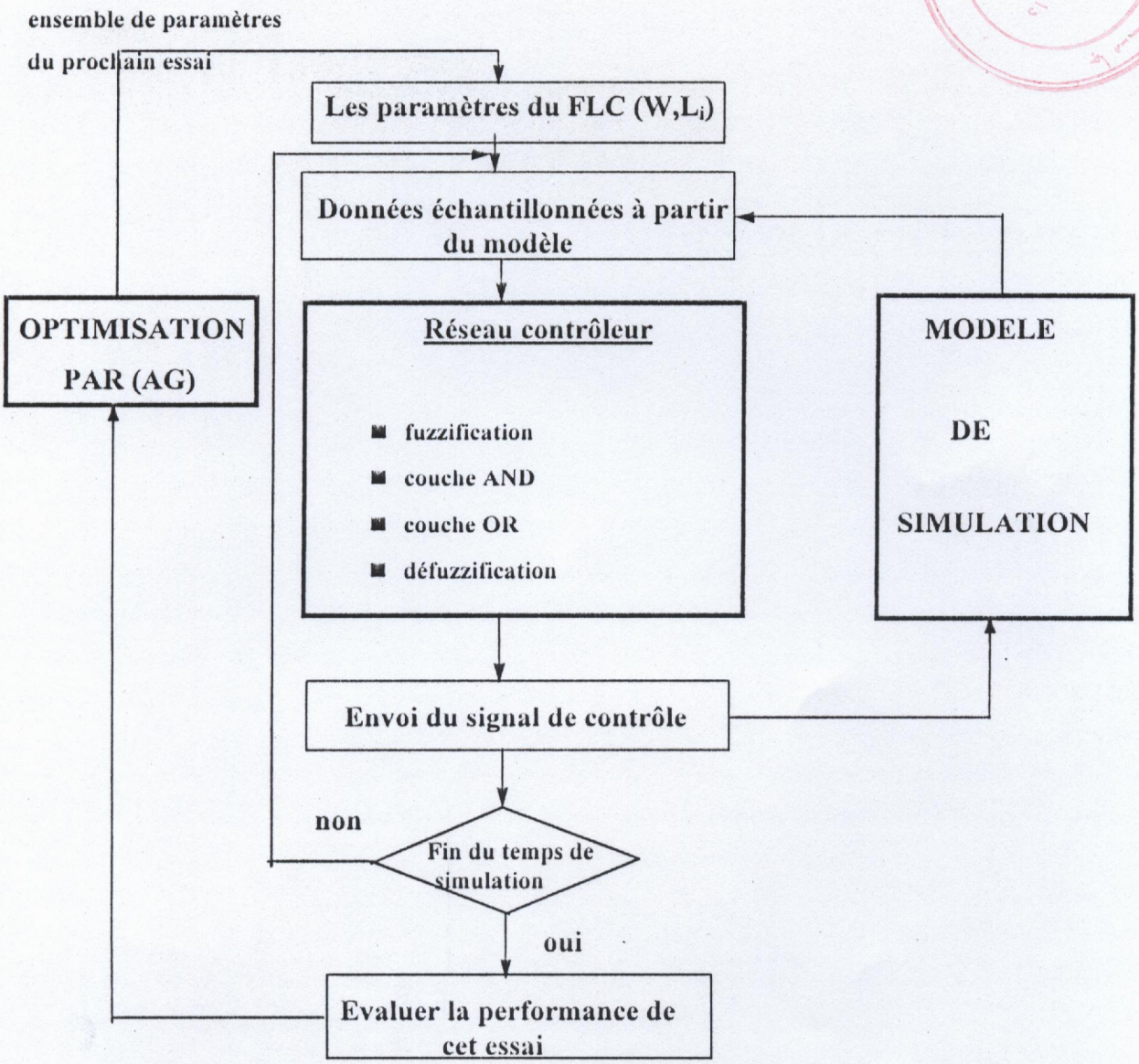


Fig. 4.6 : Organigramme de l'opération du contrôleur

IV.5 Conclusion

Dans ce chapitre nous avons proposé une méthodologie de conception des contrôleurs à logique floue à l'aide des algorithmes génétiques.

Cette méthodologie permet une conception complète des deux composants majeurs des contrôleurs flous, l'ensemble des règles et les fonctions d'appartenance donnant des contrôleurs performants complètement conçus par calculateurs.

Pour cela nous avons considéré l'implémentation structurelle d'un contrôleur flou du type TPE dans un réseau connexioniste multicouche.

Les grandeurs ajustables de ce réseau sont les largeurs des univers de discours des espaces d'E/S et les poids des connexions, ces derniers sont assujettis à une certaine contrainte.

Ainsi l'apprentissage du réseau consiste à obtenir des valeurs optimales pour ces grandeurs qui minimisent une certaine fonction coût et tel que la contrainte cité ci-dessus soit satisfaite.

Vu les caractéristiques de ce problème, on a utilisé un algorithme simple avec trois opérations :reproduction, croisement et mutation pour l'optimisation de ces grandeurs.

En vu de respecter la contrainte imposée sur les poids, on a introduit un certain nombre d'opérations supplémentaires lors du croisement et de la mutation.

Résultats de simulation

V.1 Introduction.

V.2 Le système du pendule inversé.

V.3 Le contrôle du pendule inversé.

V.3.1 Structure de simulation de la commande .

V.3.2 Conception du contrôleur.

V.3.3 Contrôleur 333.

V.3.3.1 Résultats de simulation.

V.3.3.2 Robustesse du contrôleur 333.

V.3.4 Contrôleur 555.

V.3.4.1 Résultats de simulation

V.3.4.2 Robustesse du contrôleur 555.

V.4 Conclusion.

V.1 Introduction

L'approche de conception des contrôleurs à logique floue que nous avons développé au chapitre précédent est tout à fait générale et elle peut être appliquée à une variété de problèmes de contrôle. Dans ce chapitre, on va démontrer l'efficacité de cette méthode en l'appliquant à la conception d'un contrôleur flou pour un problème de référence 'benchmark problem' dans le contrôle intelligent - le système du pendule inversé.

Le pendule étant à un angle et vitesse angulaire non nuls, l'objectif est de déterminer un contrôleur flou capable de ramener le pendule à sa position d'équilibre (verticale) en minimisant une certaine fonction coût. Deux types de contrôleurs ont été conçus pour ce problème en divisant les espaces d'entrée et de sortie en deux formes de partition.

L'AG utilisé pour l'apprentissage du contrôleur a été testé pour différentes méthodes de reproduction et de sélection finale.

V.2 Le système du pendule inversé

La figure (5.1) montre le diagramme schématique du système du pendule inversé.

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

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

- θ :l'angle du segment par rapport à l'axe vertical.
- $\dot{\theta}$:la vitesse angulaire du segment.

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

$$\ddot{\theta} = \frac{g.\sin\theta + \cos\theta.\left(\dfrac{-F - m.l.\dot{\theta}^2.\sin\theta}{m_c + m}\right)}{l.\left(\dfrac{4}{3} - \dfrac{m.\cos^2\theta}{m_c + m}\right)} \qquad (5.1)$$

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

m_c, masse du chariot = 1 Kg.

m, masse du segment = 0.1 Kg.

l, demi longueur du segment = 0.5 m.

F, la force appliquée en Newton (N).

Si on pose :

$$\begin{cases} x_1(t) = \theta(t) \\ x_2(t) = \dot{\theta}(t) \end{cases}$$

Le système décrit par l'équation (5.1) sera défini par les équations différentielles suivantes :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dfrac{g.\sin(x_1) + \cos(x_1).\left(\dfrac{-F - m.l.x_2^2.\sin(x_1)}{m_c + m}\right)}{l.\left(\dfrac{4}{3} - \dfrac{m.\cos^2(x_1)}{m_c + m}\right)} \end{cases} \tag{5.2}$$

Ce système est simulé par la méthode d'Euler avec une période d'échantillonnage égale à 10 ms.

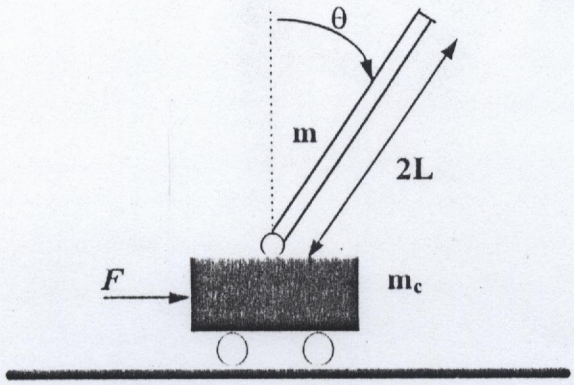


Fig. 5.1 : Le système du pendule inversé

V.3 Le contrôle du pendule inversé

V.3.1 Structure de simulation de la commande

Comme on a vu au chapitre précédent, chaque fois qu'un contrôleur est conçu par l'algorithme génétique, il est testé en simulation sur le modèle du système à commander pour évaluation. La structure de simulation de la commande dans le cas du pendule inversé est illustrée par la figure (5.2), elle est constituée de deux blocs :

- **Modèle du système :** Comme mentionné précédemment, il y a plusieurs manières

de modéliser le système à commander à savoir fonction de transfert, des équations d'état, un modèle neuronal, modèle 'flou'..., cela dépend de notre connaissance du système.
Dans notre cas, le procédé est un système dynamique non linéaire déterministe avec des équations différentielles définies avec précision. Ainsi, on peut juste employer une approximation linéaire pour obtenir les équations aux différence suivantes :

$$\begin{cases} x_1(t+h) = h.\dot{x}_1(t) + x_1(t) \\ x_2(t+h) = h.\dot{x}_2(t) + x_2(t) \end{cases} \tag{5.3}$$

où :

$x_1(.) = \theta(.)$

$x_2(.) = \dot{\theta}(.)$

h est la période d'échantillonnage

- **Bloc du contrôleur :** On suppose qu'on ne dispose pas de contrôleur obtenu par expertise. Si c'est le cas, ce contrôleur sera pris comme contrôleur initial au cours de l'optimisation. Le contrôleur flou dans la figure (5.2) est réalisé par un réseau interconnecté multicouche qui a été présenté au chapitre IV, avec :

- Les signaux d'entrée du contrôleur : l'angle du segment (θ en degré) et la vitesse angulaire du segment ($\dot{\theta}$ en deg./s), soit le vecteur d'entrée X=(θ,$\dot{\theta}$).
- Le signal de sortie du contrôleur : la force (F en Newton).

Le contrôleur a pour objectif de générer une force F qui va stabiliser le pendule dans sa position d'équilibre (verticale) qui correspond à un angle et vitesse angulaire nuls. Pendant l'optimisation (l'apprentissage), l'algorithme génétique délivre à chaque génération un contrôleur qui est évalué et le processus de simulation est réalisé pendant un temps T=5s.
Une fois le contrôleur optimal est trouvé, il est appliqué en simulation sur le même modèle pour tester ses performances.

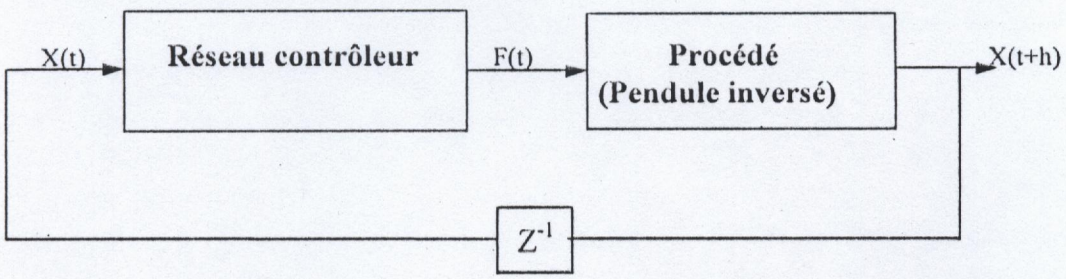


Fig. 5.2 : Structure de commande du pendule inversé

V.3.2 Conception du contrôleur

L'approche utilisée pour la conception du contrôleur flou pour le pendule inversé est celle développée dans le chapitre IV. L'AG a pour objectif de trouver des valeurs optimales pour les différentes largeurs des univers de discours (un ensemble optimal de fonctions d'appartenance) et pour les poids (ensemble optimal de règles floues) en minimisant le critère suivant :

$$J = \sum_{k=1}^{500} p_1 . |\theta(k)| + p_2 . |F(k-1)| \qquad (5.4)$$

avec p_1 et p_2 sont des facteurs de pondération que nous avons fixé empiriquement :

$p_1 = 1$ et $p_2 = 10^{-2}$ pour $t \leq 2$ s

$p_1 = 50$ et $p_2 = 10^{-3}$ pour $2 \text{ s} < t \leq 5$ s

Comme les AG sont des procédures de recherche du maximum, le problème de minimisation du critère J est transformé en problème de maximisation d'une fonction d'adaptation f définie par :

$$f = \left(\frac{C_1}{J}\right)^{C_2} \qquad (5.5)$$

Avec C_1 et C_2 sont des paramètres choisis de tel manière à avoir une variance de la fonction d'adaptation (les différences dans la fonction d'adaptation relative entre les individus) grande dans la population pour assurer une meilleur recherche. Ces paramètres sont choisis empiriquement par :

$$C_1 = 10^4$$

$$C_2 = 2$$

Les conditions physiques du pendule inversé dans notre simulation sont :

- angle initial : $\theta(0) = 20^0$ et vitesse angulaire initiale : $\dot{\theta}(0) = 0^0 / s$

Deux contrôleurs flous ont été développés pour le problème du pendule inversé, ils sont référés par le nombre d'ensembles flous utilisés pour la partition de l'univers de discours de l'angle, de la vitesse angulaire et de la force. Par exemple le contrôleur qui a un angle, une vitesse angulaire et une force tous divisés en trois ensembles flous est appelé contrôleur 333.

V.3.3 Contrôleur 333

Dans ce type de contrôleur, l'angle, la vitesse angulaire et la force sont tous divisés en trois ensembles flous :NEGATIVE (NE), ZERO (ZE) et POSITIVE (PO). Neuf règles floues

de contrôle sont utilisées ceci en considérant toutes les combinaisons possibles des variables floues d'entrée. L'architecture du réseau contrôleur est spécifiée par :

- Nombre de neurones de la couche d'entrée : 2
- Nombre de neurones de la couche de fuzzification : 6
- Nombre de neurones AND : 9
- Nombre de neurones OR : 3
- Nombre de neurones de la couche de défuzzification : 1

L'AG est utilisé pour trouver les valeurs optimales des différentes largeurs des univers de discours soient: $L_\theta, L_{\dot{\theta}}$ et L_F associées à l'angle, la vitesse angulaire et la force respectivement et les poids W (27 poids) en maximisant le critère décrit par (5.5).

La recherche des paramètres est effectuée selon le tableau ci dessus :

Paramètres	Intervalle de recherche
L_θ	[2 80]
$L_{\dot{\theta}}$	[2 100]
L_F	[2 200]

Tab.5.1 : Intervalles de recherche des paramètres pour le contrôleur 333

Le tableau (5.2) illustre les valeurs des différents paramètres de L'AG.

Un codage sur 8 bits est utilisé pour coder chacune des largeurs des différents univers de discours d'où chaque chaîne complète de la population contiendra 51 bits (24 bits pour les différentes largeurs et 27 bits pour les poids).

Paramètres génétiques	Valeur
Probabilité de croisement (P_c)	0.9
Probabilité de mutation (P_m)	0.02
Taille de la population (Nombre d'individus)	100
Nombre de génération	100

Tab.5.2 : Les valeurs des paramètres génétiques

V.3.3.1 Résultats de simulation

L'AG utilisé pour optimiser les paramètres du contrôleur 333 a été testé en employant différentes stratégies de reproduction et de sélection finale (sélection des individus d'une nouvelle génération), les cas d'études que nous avons considéré sont les suivants :

Cas1 : La reproduction est réalisé par la méthode de sélection à reste stochastique et la sélection finale est réalisée par la méthode de sélection "steady state ".

Cas2 : La reproduction est réalisée par la même méthode considérée dans le premier cas et la sélection finale est réalisée par la méthode de sélection par compétition.

Cas3 : Dans ce cas, nous avons utilisé la méthode de sélection par roue de loterie pour la reproduction et la méthode de sélection "steady state " pour la sélection finale.

Cas4 : La reproduction est réalisée par la même méthode utilisée dans le troisième cas c'est à dire par roue de loterie et la méthode de sélection par compétition est considérée pour la sélection finale.

Les caractéristiques du processus d'apprentissage sont analysées par la fonction d'adaptation moyenne 'mean fitness', MF, de la population et la fonction d'adaptation du meilleur individus 'best fitness', BF, dans la population.

Les figures (Fig. 5.4.a) et (Fig. 5.4.b) montrent les allures des fonctions d'adaptations BF et MF, pour chaque génération des quatre cas considérés ci-dessus.

On constate que la reproduction par la méthode de sélection à reste stochastique est plus performante que celle par roue de loterie. Ceci peut s'expliquer par le fait que la sélection à reste stochastique est une méthode élitiste qui garantit la survie du meilleur individus de la population et le conserve pour la génération suivante par contre dans la sélection par roue de loterie il est tout à fait possible que le meilleur individu de la population ne soit pas retenu dans la prochaine génération.

D'un autre côté, la méthode de sélection "steady state " pour la sélection finale retient les deux meilleurs individus parmi les quatre (enfants et parents) après chaque recombinaison génétique ce qui permet d'accélérer le processus d'apprentissage (convergence rapide), la sélection finale par compétition qui consiste à effectuer une compétition entre parents et enfants pour déterminer les survivants de la génération garantit une bonne convergence mais relativement lente par rapport à la "steady state selection " .

On peut conclure donc que la reproduction par la méthode de sélection à reste stochastique combinée avec la méthode de la "steady state selection " donne la meilleure convergence.

Les résultats du processus d'apprentissage obtenues par l'AG employant ces deux méthodes sont :

- valeur maximale de la fonction d'adaptation : 53.5467.
- $L_\theta = 15.4588$.
- $L_{\dot{\theta}} = 48.5019$.
- $L_F = 200$.
- $W_1 = [1\ 0\ 0]$, $W_2 = [1\ 0\ 0]$, $W_3 = [0\ 1\ 0]$.
- $W_4 = [1\ 0\ 0]$, $W_5 = [0\ 1\ 0]$, $W_6 = [0\ 0\ 1]$.
- $W_7 = [0\ 1\ 0]$, $W_8 = [0\ 0\ 1]$, $W_9 = [0\ 0\ 1]$.

La figure (5.3) montre les différentes fonctions d'appartenance de l'angle, la vitesse angulaire et de la force et le tableau (5.3) représente la base de règles de contrôle floues pour la commande du pendule inversé.

Les figures (Fig. 5.5.a-d) représentent (a) l'angle en degré, (b) la vitesse angulaire en degré par seconde et (d) les actions de contrôle (N) à partir de t=0 à t=5s, (c) est l'espace d'état qui montre comment la trajectoire approche l'origine à partir du point initial (20,0).

On remarque que le contrôleur arrive à stabiliser le pendule sans oscillations autour de la position d'équilibre pendant un temps moins de 2s, le retour à l'état d'équilibre se fait pendant deux phases; une phase d'accélération et une phase de décélération.

$\dot{\theta}$ \ θ	NE	ZE	PO
NE	**NE**	**NE**	**ZE**
ZE	**NE**	**ZE**	**PO**
PO	**ZE**	**PO**	**PO**

Tab.5.3 : Base de règles floues du contrôleur 333

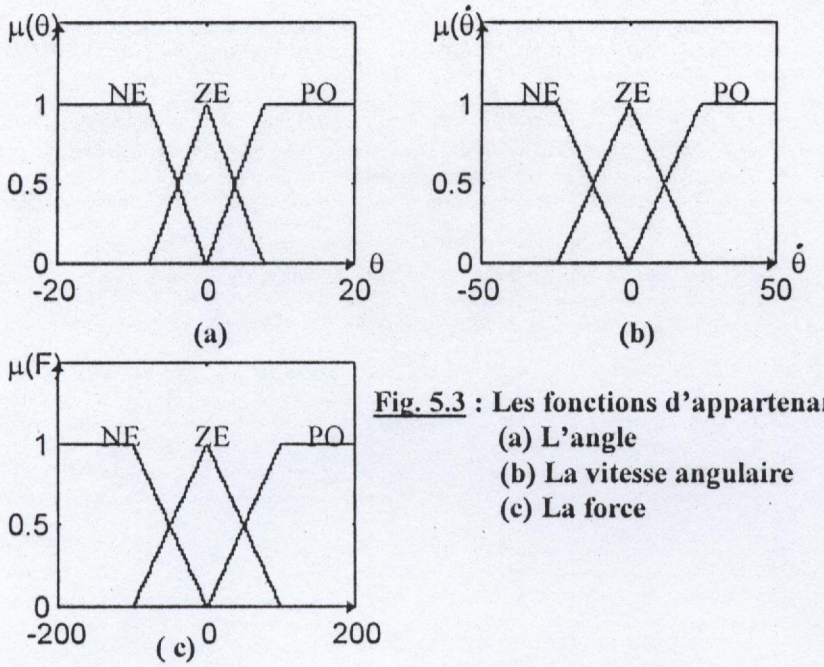


Fig. 5.3 : Les fonctions d'appartenance
(a) L'angle
(b) La vitesse angulaire
(c) La force

V.3.3.2 Robustesse du contrôleur 333

Le contrôleur flou obtenu est capable de ramener le pendule à sa position d'équilibre à partir du point initial (20,0). Cependant un bon contrôleur doit être robuste, pour cela le contrôleur 333 a été testé pour plusieurs conditions initiales. Les figures (Fig. 5.6.a-d) et (Fig. 5.7.a-d) montrent la performance du contrôleur flou optimisé pour le contrôle du pendule inversé avec des conditions initiales différentes. On constate que le contrôleur arrive toujours à stabiliser le pendule, et prend plus de 4s pour balancer le pendule avec un angle initial de 70° mais échoue pour un angle initial supérieur à 80°.

Pour voir comment le contrôleur va réagir aux changements des paramètres du procédé, on l'a testé sur des segments de différentes longueurs (0.25, 0.5, 1 et 1.5 m), les résultats sont donnés dans les figures (Fig. 5.8.a-d). Il est remarquable de noter que le contrôleur peut gouverner le pendule court plus facilement.

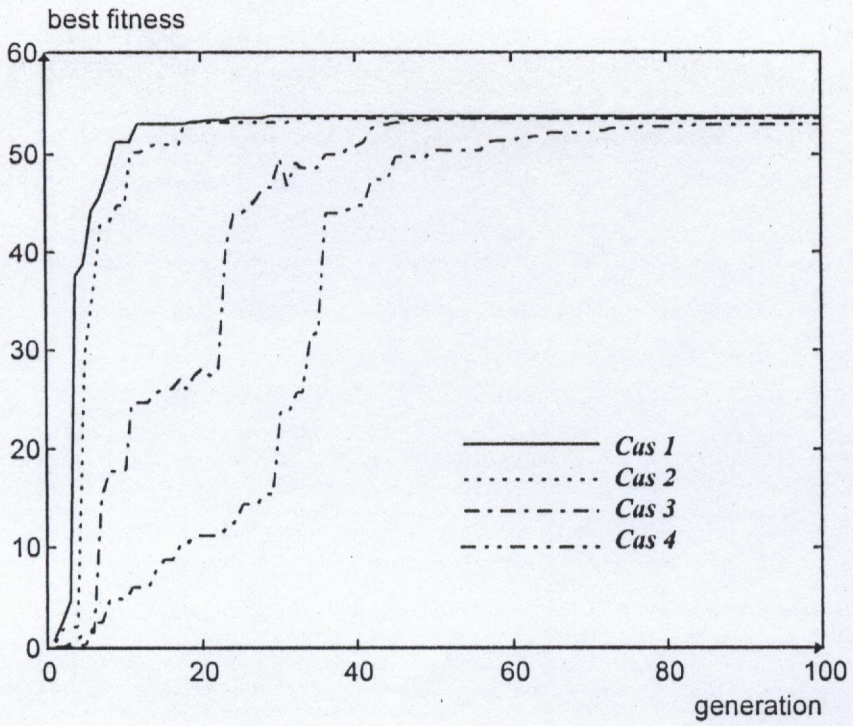


Fig. 5.4.a

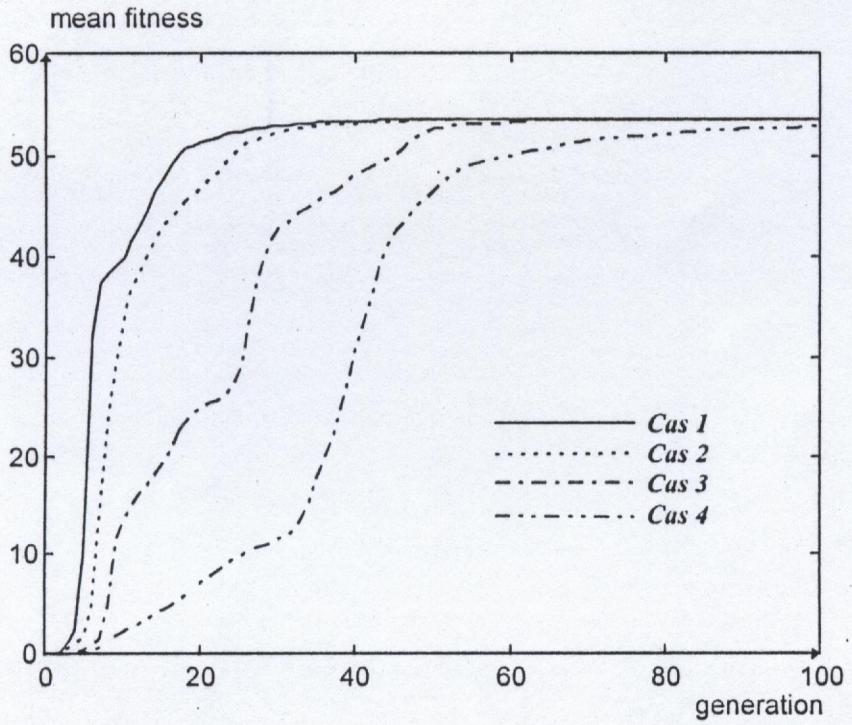


Fig. 5.4.b

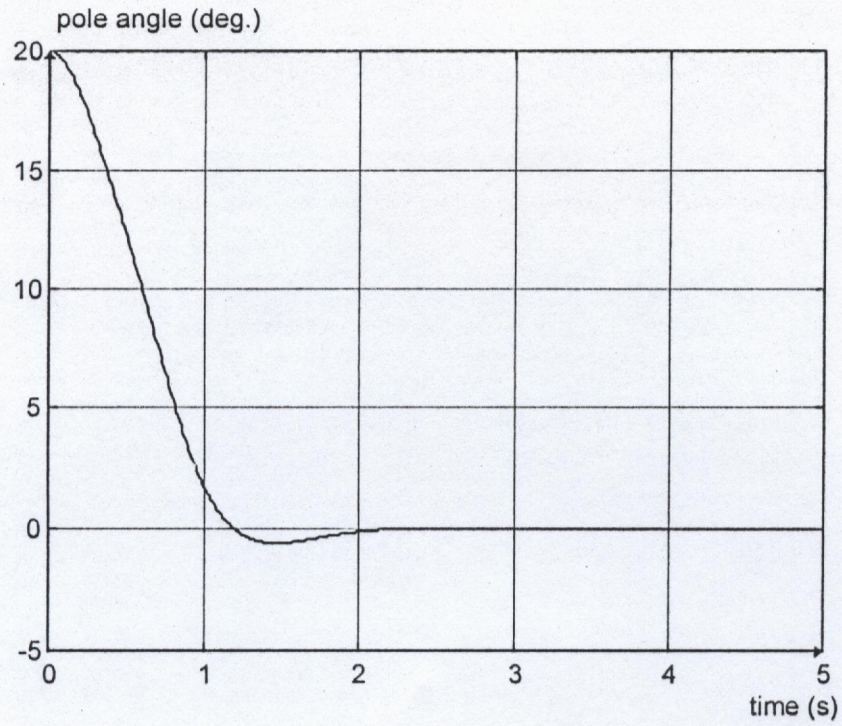


Fig. 5.5.a : Variations de l'angle à partir du point (20°,0°/s)

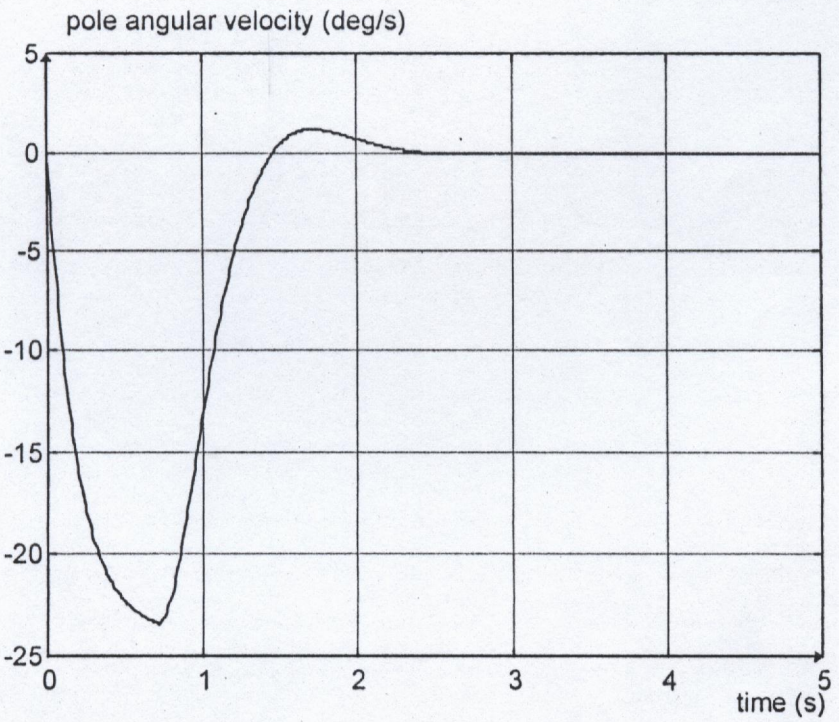


Fig. 5.5.b : Variations de la vitesse angulaire à partir du point (20°,0°/s)

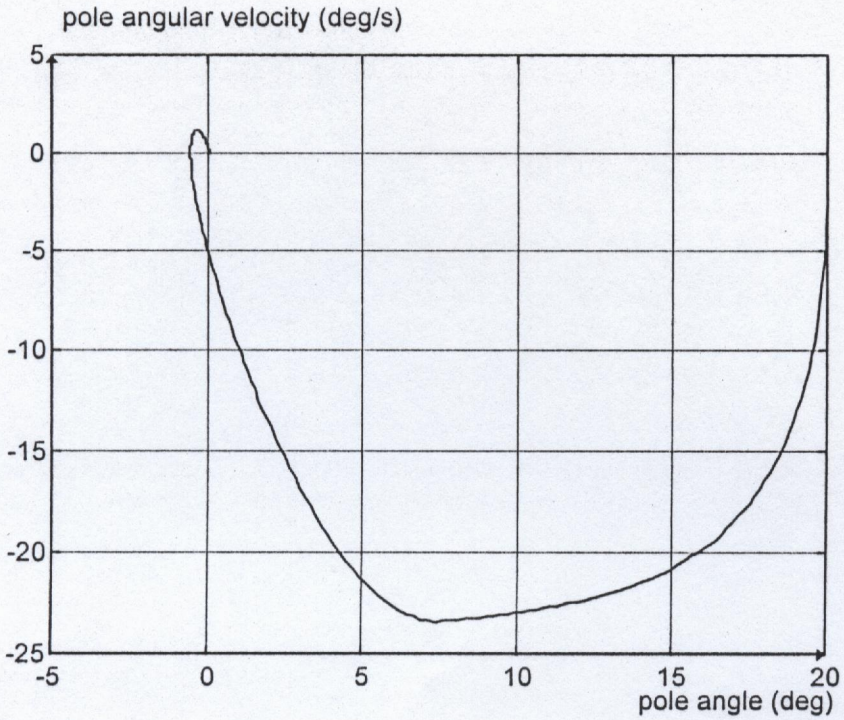


Fig. 5.5.c : plan de phase à partir de (20°,0°/s)

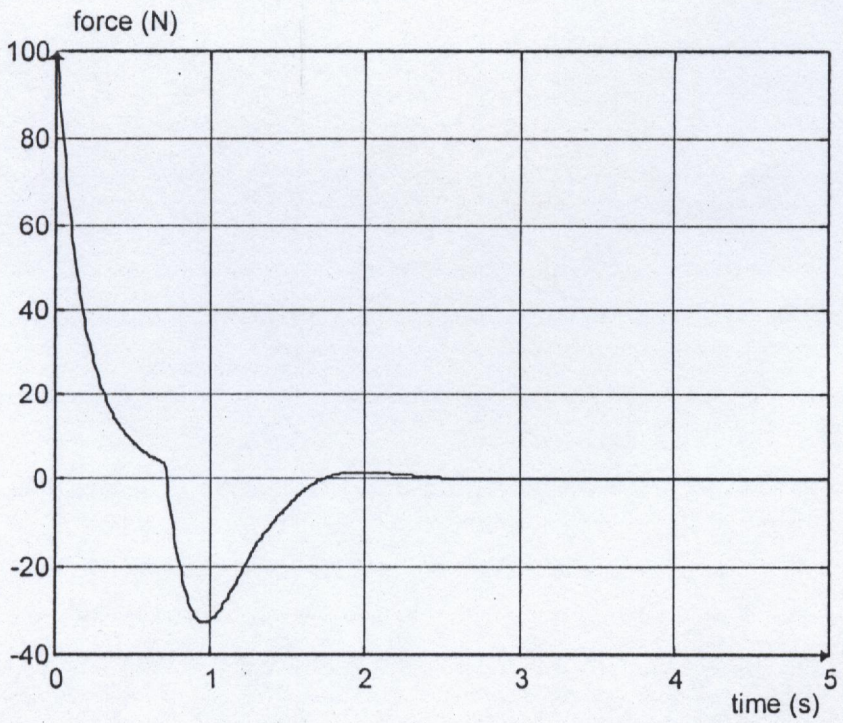


Fig. 5.5.d : Variations de la force

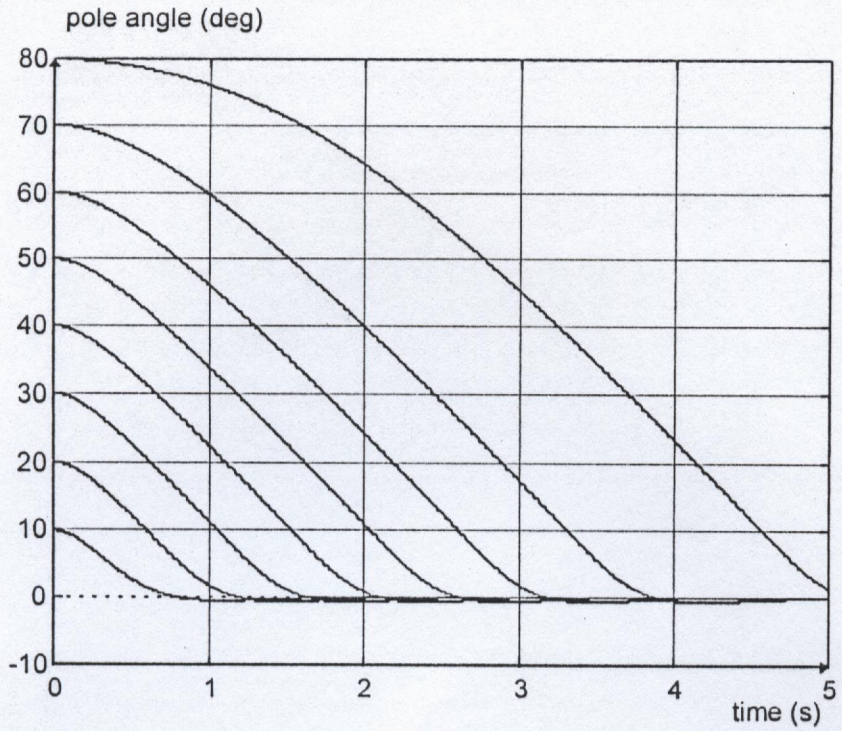


Fig. 5.6.a : Variations de l'angle pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

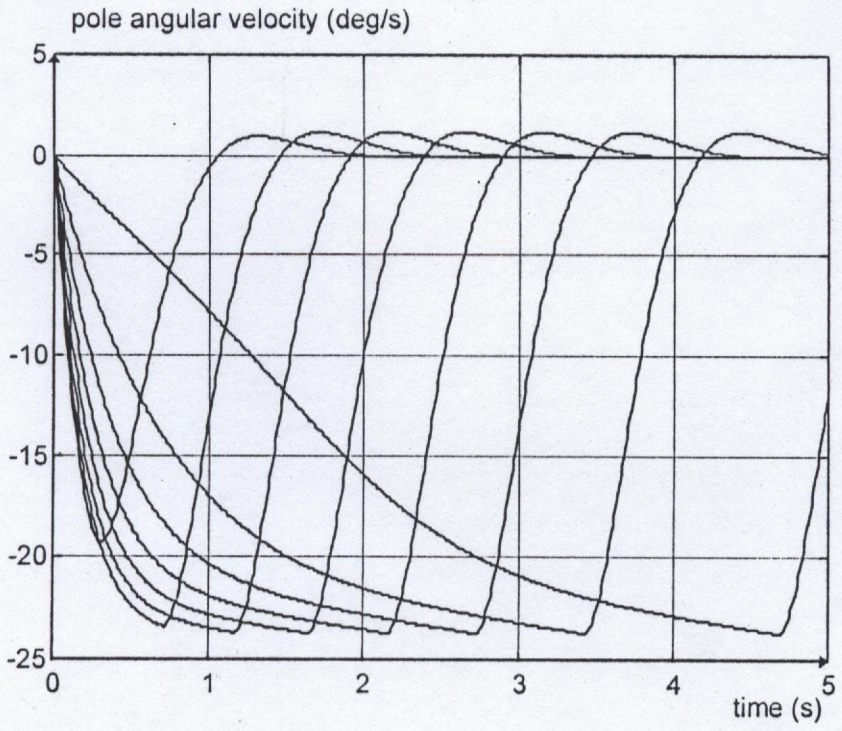


Fig. 5.6.b : Variations de la vitesse angulaire pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

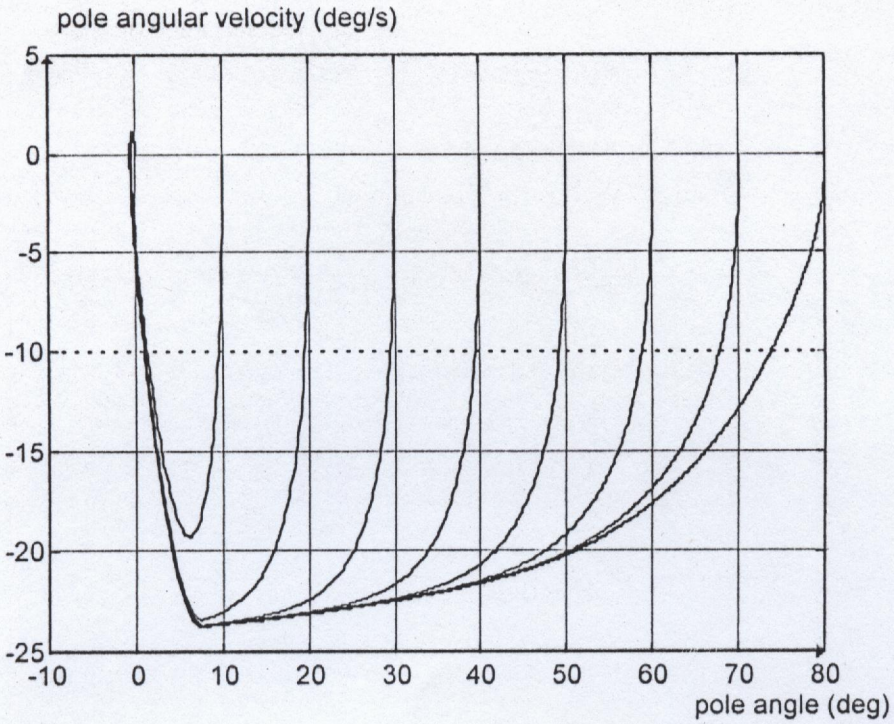


Fig. 5.6.c : Plan de phase pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

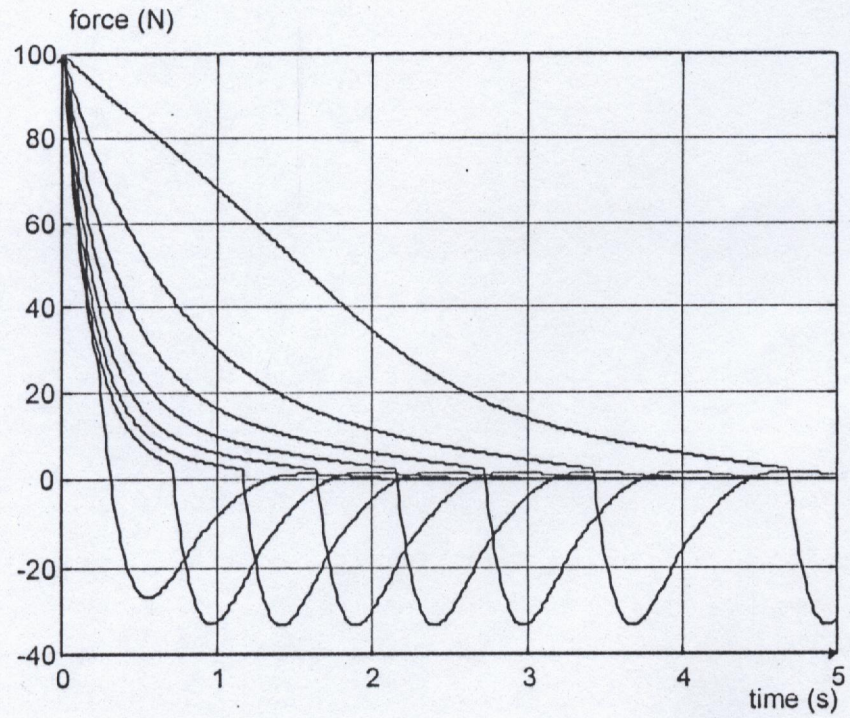


Fig. 5.6.d : Variations de la force pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

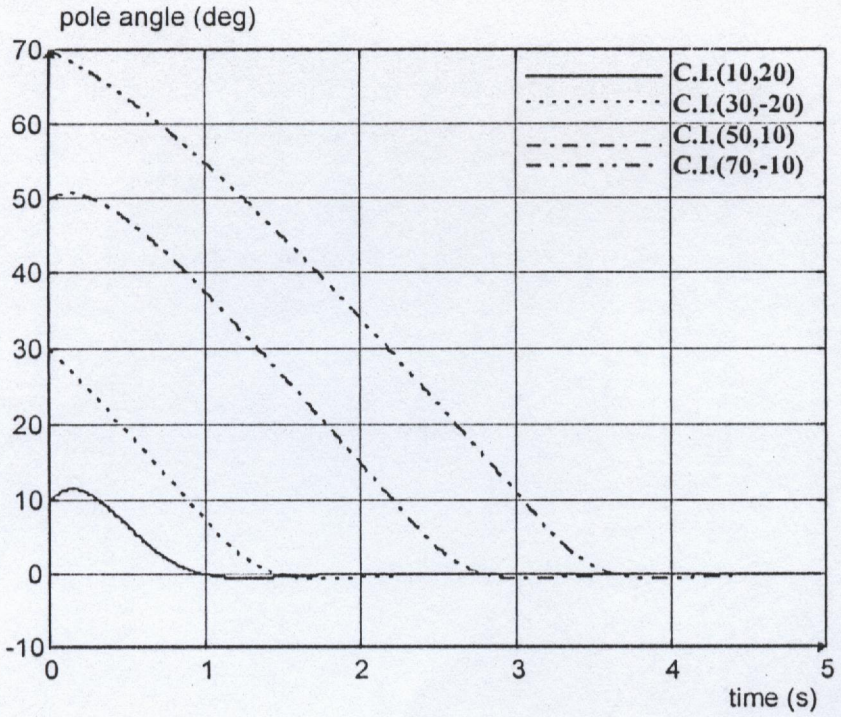


Fig. 5.7.a : Variations de l'angle pour les conditions initiales :
(10 °,20 °/s),(30 °,-20 °/s),(50 °, 10 °/s),(70 °,-10 °/s)

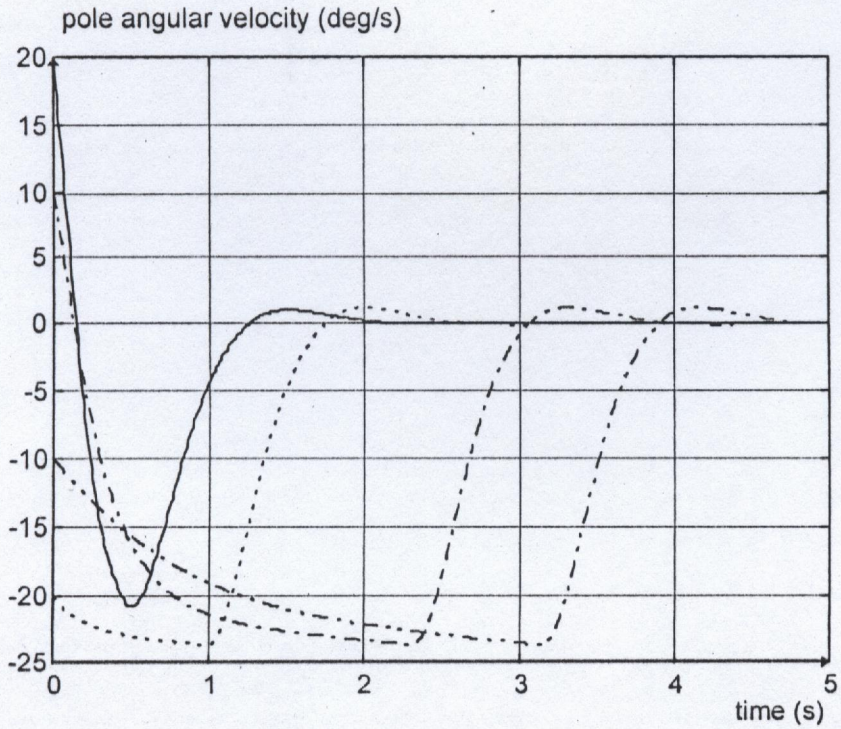


Fig. 5.7.b : Variations de la vitesse angulaire pour les conditions initiales :
(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

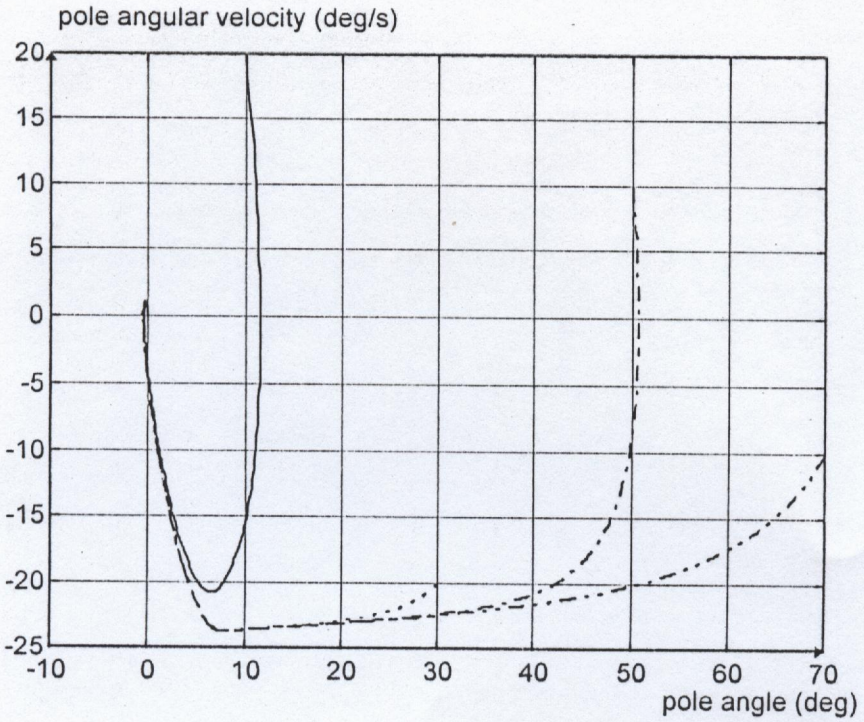


Fig. 5.7.c : Plan de phase à partir des point s :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

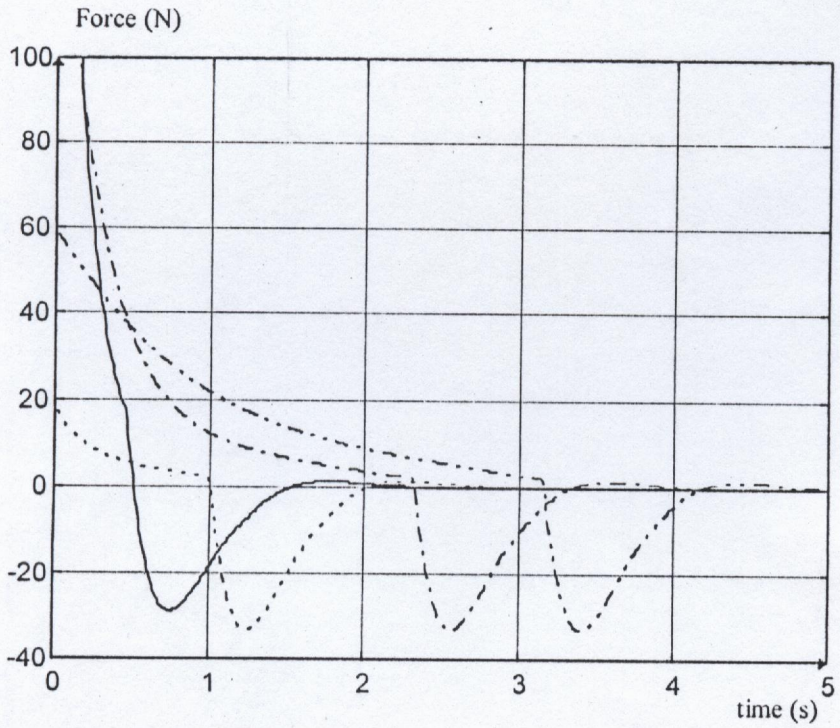


Fig. 5.7.d : Variations de la force à partir des conditions initiales :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

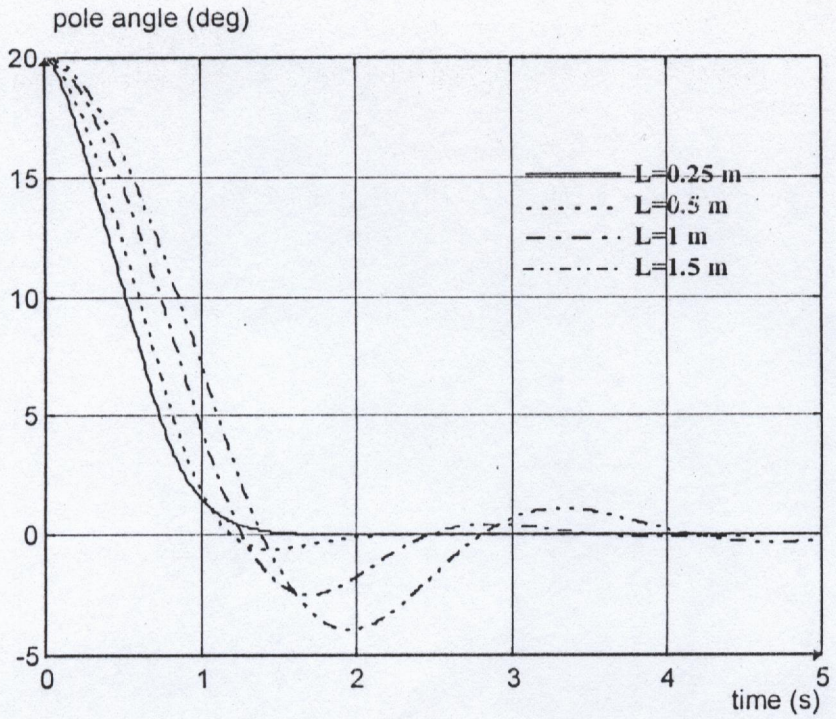


Fig. 5.8.a : Variations de l'angle pour des pendules de longueur variable (0.25,0.5,1 et 1.5m)

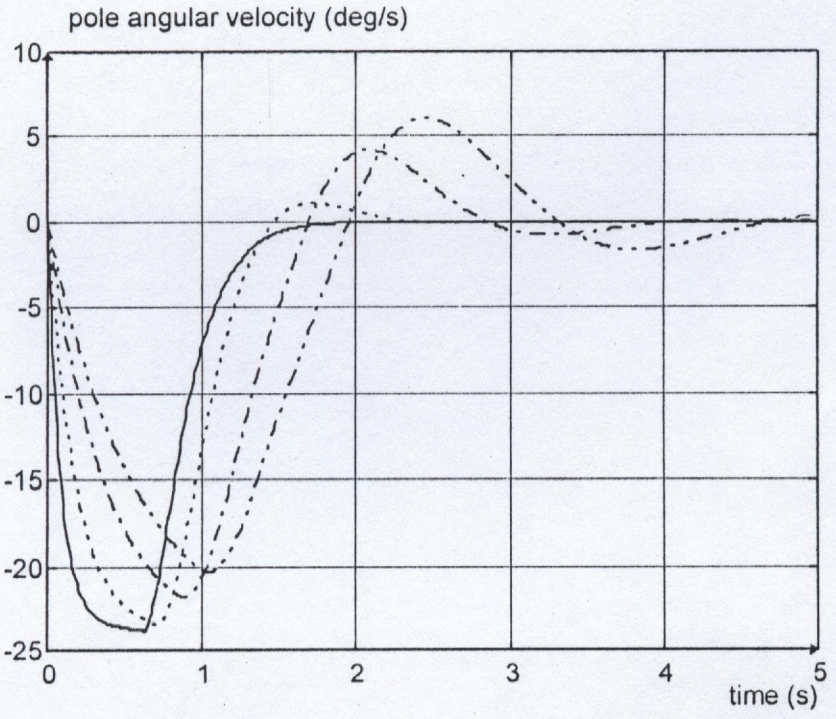


Fig. 5.8.b : Variations de la vitesse angulaire pour des pendules de longueur variable (0.25,0.5,1 et 1.5m)

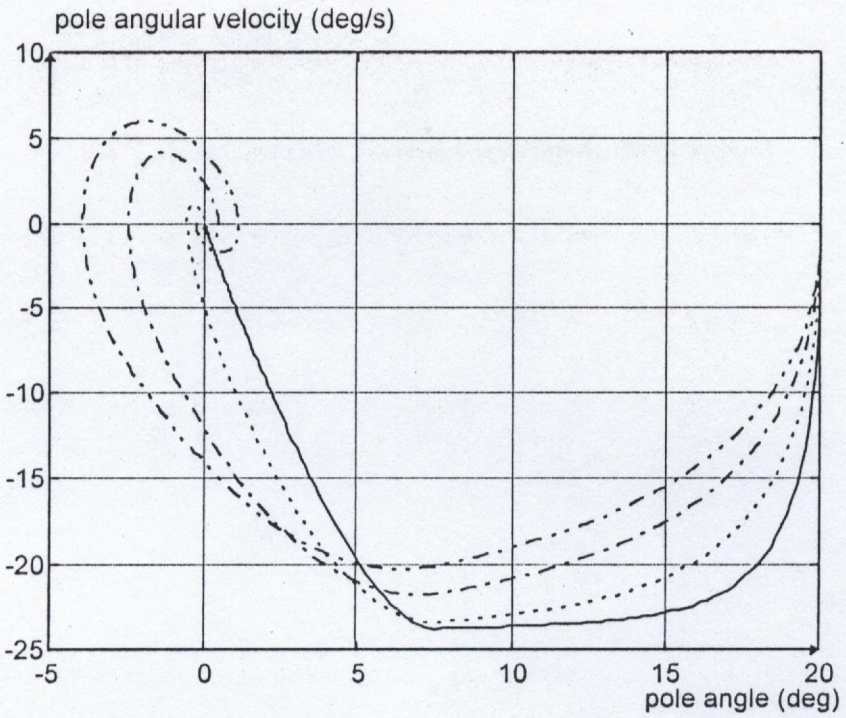


Fig. 5.8.c : Plan de phase pour des pendules de longueur variable

(0.25,0.5,1 et 1.5m)

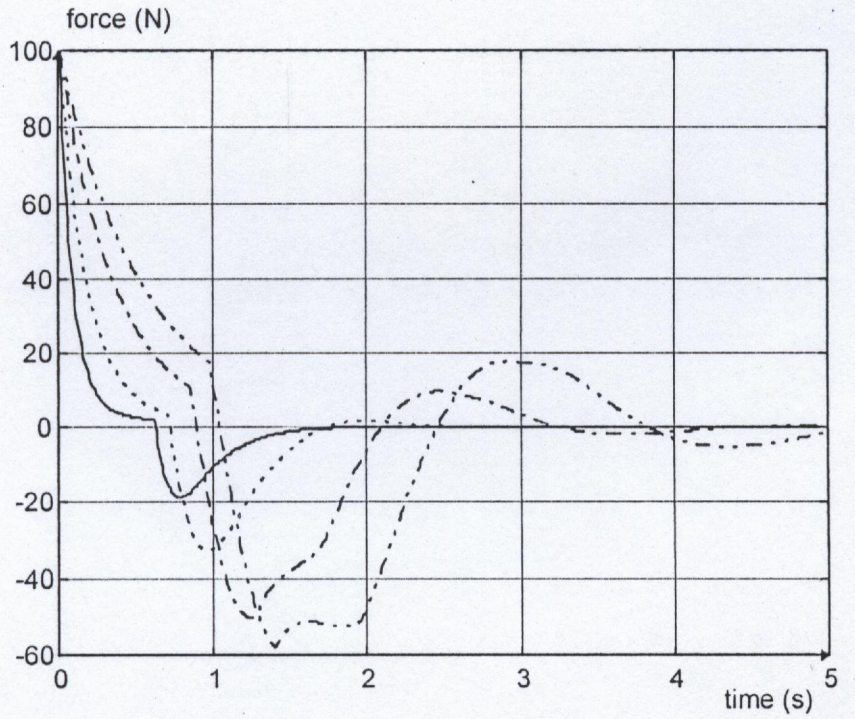


Fig. 5.8.d : Variations de la force pour des pendules de longueur variable

(0.25,0.5,1 et 1.5m)

V.3.4 Contrôleur 555

Dans ce cas, l'angle, la vitesse angulaire et la force sont tous divisés en cinq ensembles flous :NEGATIVE (NE), NEGATIVE MEDIUM (NM), ZERO (ZE), POSITIVE MEDIUM (PM) et POSITIVE (PO), ce contrôleur utilise 25 règles floues.

L'architecture du réseau contrôleur est spécifiée par :

- Nombre de neurones de la couche d'entrée : 2
- Nombre de neurones de la couche de fuzzification : 10
- Nombre de neurones AND : 25
- Nombre de neurones OR : 5
- Nombre de neurones de la couche de défuzzification : 1

Le nombre de poids à optimiser consiste en 125 poids et la recherche des paramètres : $L_\theta, L_{\dot{\theta}}$ et L_F est effectuée selon le tableau (5.4).

Paramètres	Intervalle de recherche
L_θ	[4 120]
$L_{\dot{\theta}}$	[4 130]
L_F	[4 240]

Tab.5.4 : Intervalles de recherche des paramètres pour le contrôleur 555

Les valeurs des paramètres génétiques sont identiques à celles utilisées pour l'optimisation du contrôleur 333, elles sont indiquées dans le tableau (5.2).

Huit bits sont utilisés pour coder chacune des largeurs des différents univers de discours, d'où chaque chaîne complète de la population contiendra 149 bits (24 bits pour les différentes largeurs et 125 bits pour les poids).

V.3.4.1 Résultats de simulation

Pour l'optimisation du contrôleur 555 on a considéré un AG employant pour la reproduction la méthode de sélection à reste stochastique et pour la sélection finale la méthode "steady state selection", ce choix est du aux meilleures résultats obtenues par ces deux méthodes lors de l'optimisation du contrôleur 333.

Les caractéristiques du processus d'apprentissage sont illustrées dans la figure (5.9) qui montre la meilleure fonction d'adaptation et la fonction d'adaptation moyenne pour chaque génération. La valeur maximale de la fonction d'adaptation obtenue est égale à 120.7498 qui est environ deux fois plus grande que celle obtenue par le contrôleur 333.
Le réseau contrôleur résultant est spécifié par les fonctions d'appartenance indiquées dans la figure (5.10) avec des univers de discours de largeur :

- $L_\theta = 10.8235$.
- $L_{\dot{\theta}} = 66.7529$.
- $L_F = 240$.

et par la base des règles floues représentée dans le tableau (5.5).
Les figures (Fig. 5.11.a-d) montrent les variations de l'angle (a), la vitesse angulaire (b)et (d) de la force de t=0 à t=5s. La figure(c) est l'espace d'état indiquant la trajectoire d'approche de l'origine à partir du point initial (20,0).

$\dot{\theta}$ \ θ	NE	NM	ZE	PM	PO
NE	**NE**	**NE**	**NE**	**NE**	**ZE**
NM	**PO**	**NE**	**NE**	**NE**	**PO**
ZE	**NE**	**NE**	**ZE**	**PO**	**PO**
PM	**PM**	**PO**	**PO**	**PO**	**PO**
PO	**ZE**	**NE**	**PO**	**ZE**	**PO**

Tab.5.5 : Base de règles floues du contrôleur 555

Ce contrôleur permet de ramener le pendule à sa position d'équilibre plus rapidement que le contrôleur 333 en un temps inférieur à 1.5s sans oscillations.

V.3.4.2 Robustesse du contrôleur 555

Les même tests de robustesse effectués pour le contrôleur 333 sont considérés pour le contrôleur 555, en effet les figures (Fig. 5.12.a-d) et (Fig. 5.13.a-d) montre la performance du contrôleur optimisé pour des conditions initiales différentes.

On remarque que le contrôleur arrive toujours à stabiliser le pendule et prend moins de 4s pour balancer le pendule avec un angle initial de 80°, mais échoue pour un angle initial supérieur à 85°.

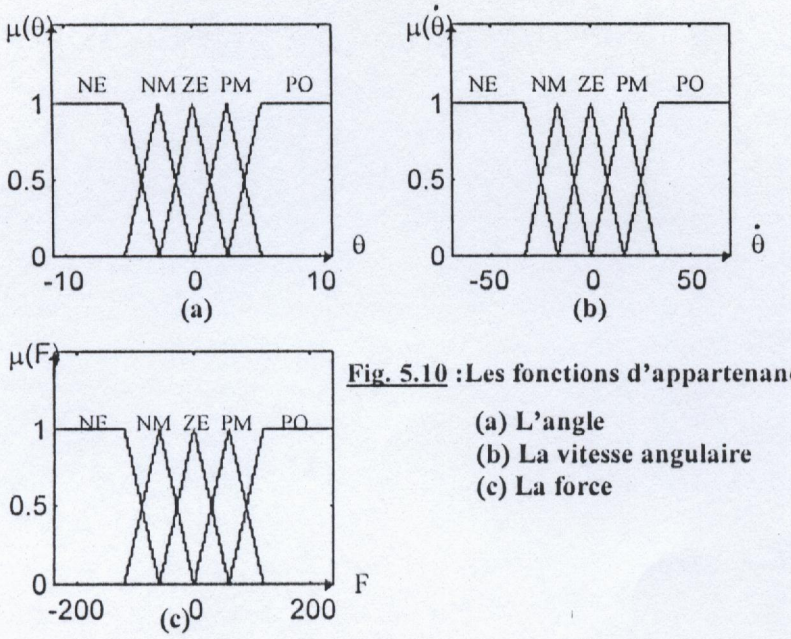


Fig. 5.10 :Les fonctions d'appartenance

(a) L'angle
(b) La vitesse angulaire
(c) La force

Les figures (Fig. 5.14.a-d) montrent la robustesse du contrôleur 555 en considérant des pendules de différentes longueurs (0.25, 0.5, 1 et 1.5m).

D'après les courbes des figures (Fig. 5.12.a-d), (Fig. 5.13.a-d) et (Fig. 5.14.a-d), on constate que la robustesse, mesurée ici par la capacité du contrôleur à ramener le pendule à la verticale pour des conditions initiales variables et pour des pendules de différentes longueurs, s'est nettement améliorée.

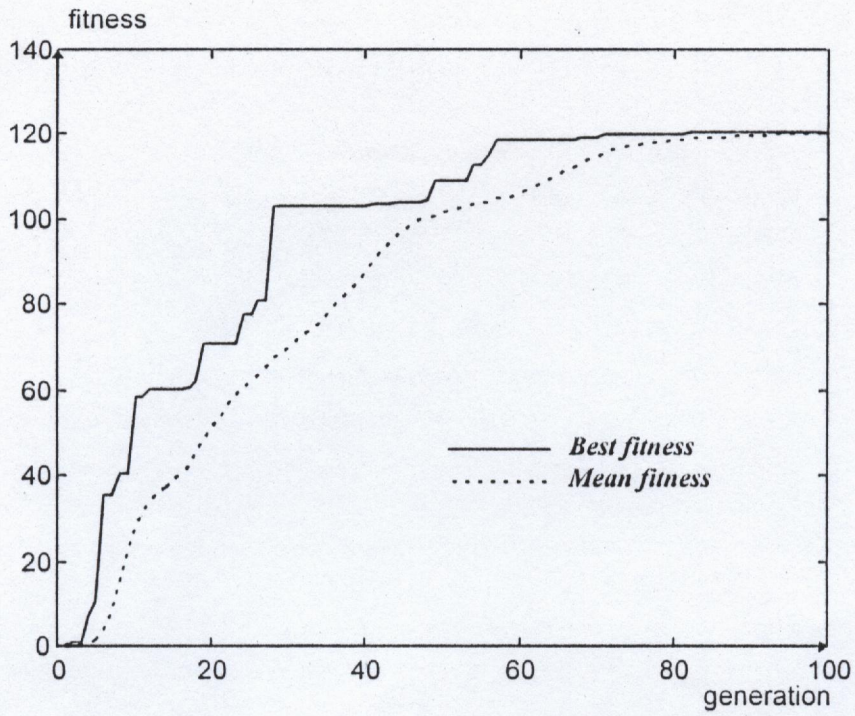


Fig. 5.9

V.4 Conclusion

Dans ce chapitre, l'approche de contrôle que nous avons développé a été testé pour la commande du pendule inversé.

Dans un premier cas, nous avons construit par apprentissage un contrôleur flou à trois variables linguistiques : NE, ZE et PO pour toutes les variables d'E/S. Bien que composé d'un nombre restreint de règles (9 règles floues), ce contrôleur a donné de très bons résultats.

Ensuite nous avons construit un contrôleur à cinq variables linguistiques : NE, NM, ZE, PM et PO qui utilise 25 règles floues de contrôle, la robustesse s'est améliorée mesurée ici par le taux de réussite du contrôleur exprimé par la capacité de ramener le pendule à la verticale pour des conditions initiales variables et pour différentes valeur des paramètres du pendule inversé (longueur du segment).

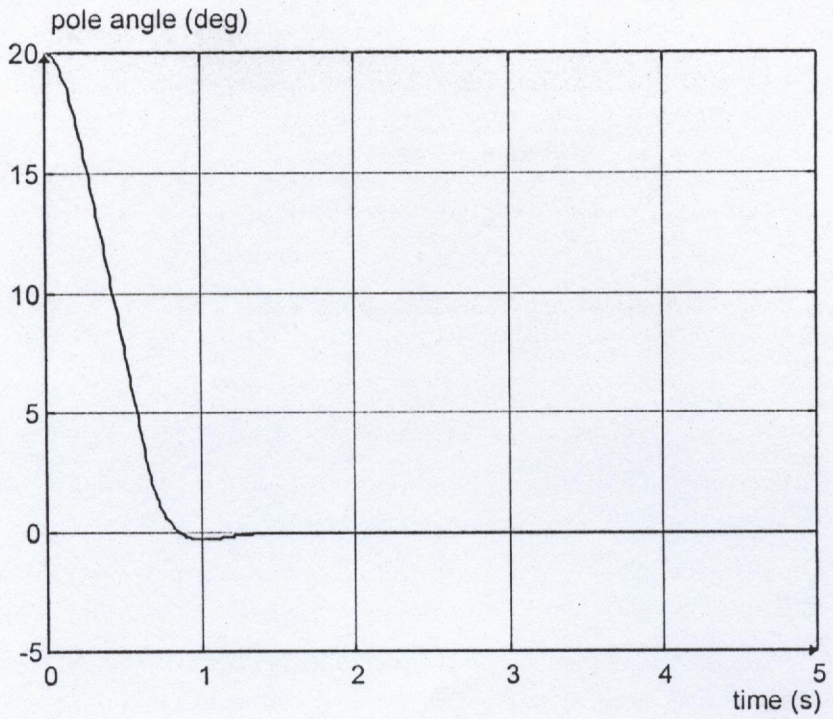


Fig. 5.11.a : Variations de l'angle à partir du point (20°,0°/s)

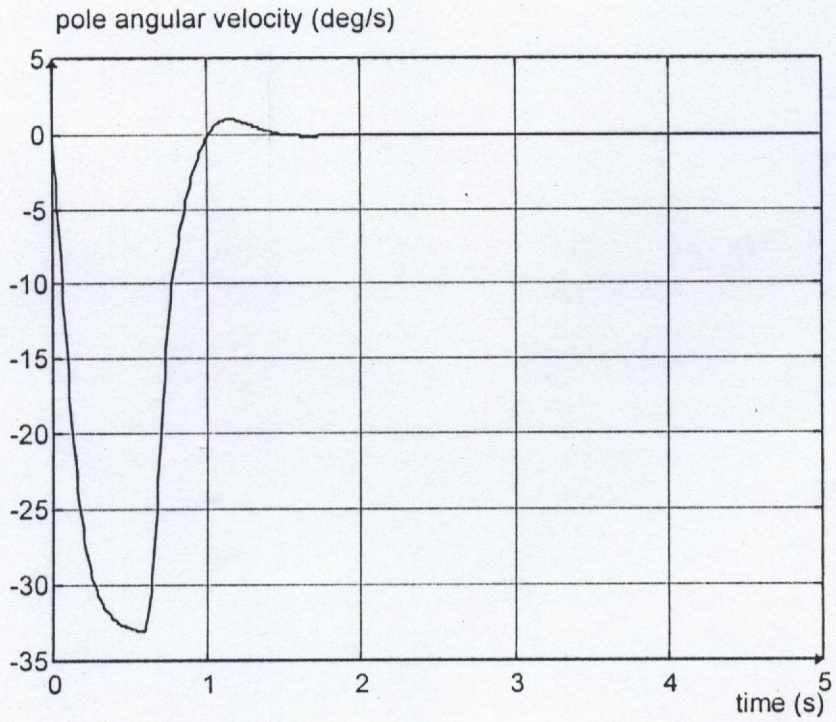


Fig. 5.11.b : Variations de la vitesse angulaire à partir du point (20°,0°/s)

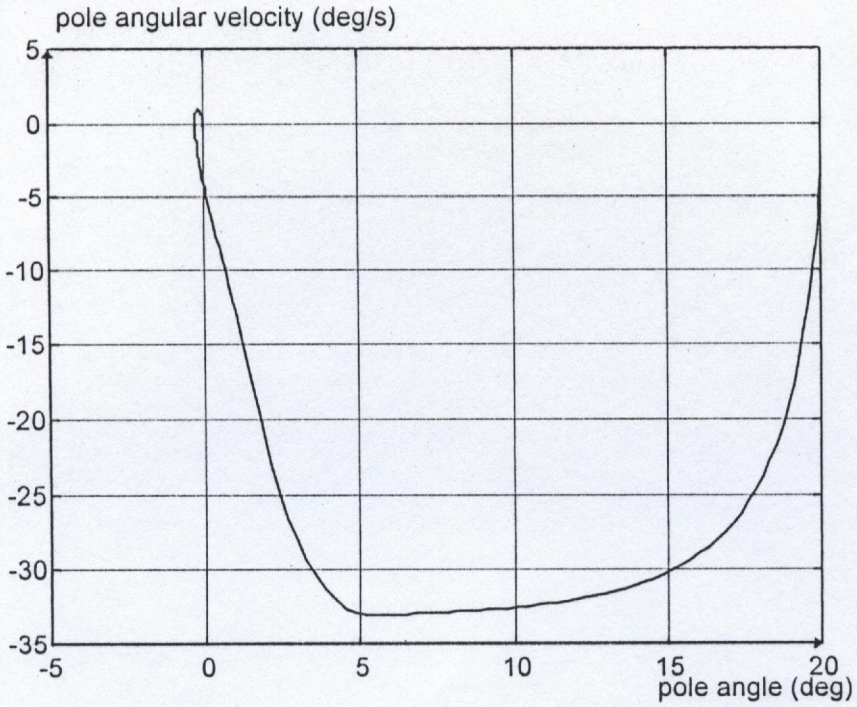


Fig. 5.11.c : plan de phase à partir de (20°,0°/s)

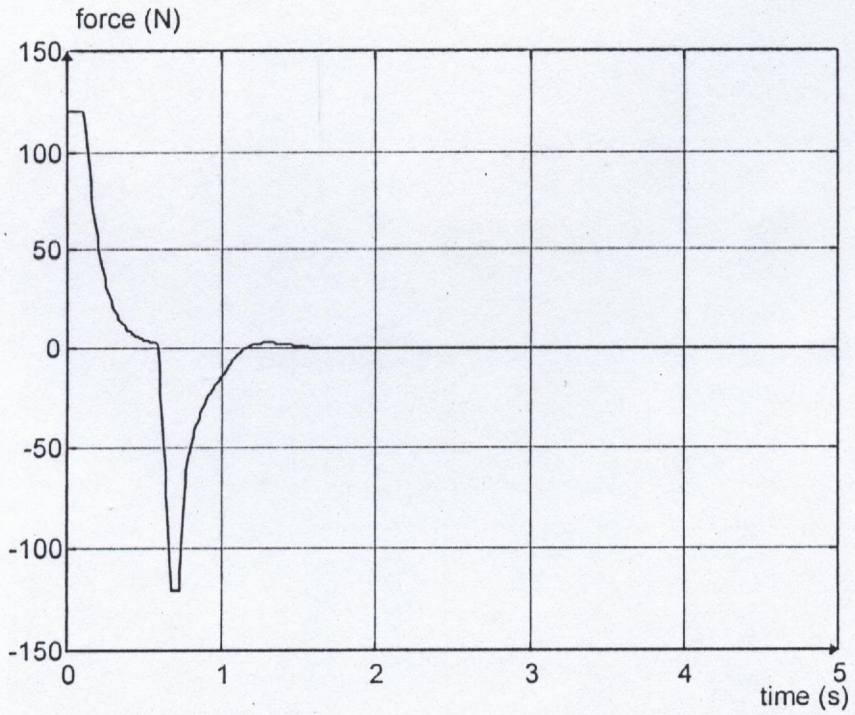


Fig. 5.11.d : Variations de la force

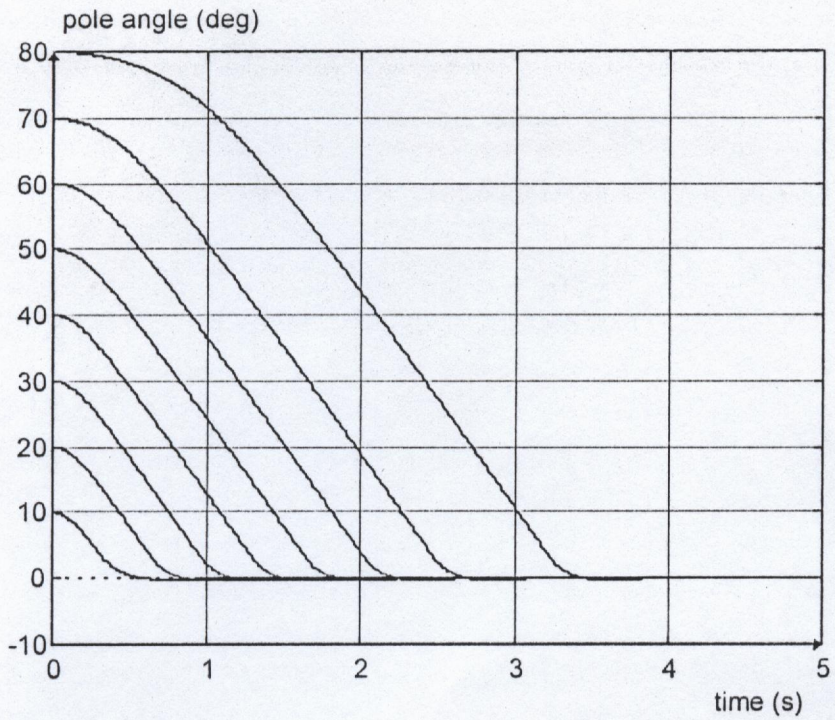


Fig. 5.12.a : Variations de l'angle pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

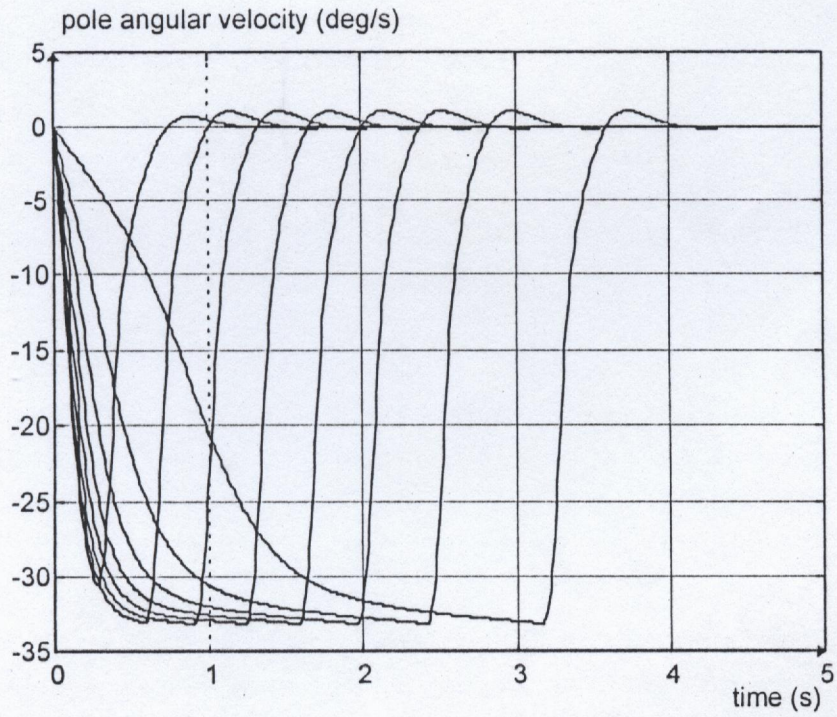


Fig. 5.12.b : Variations de la vitesse angulaire pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

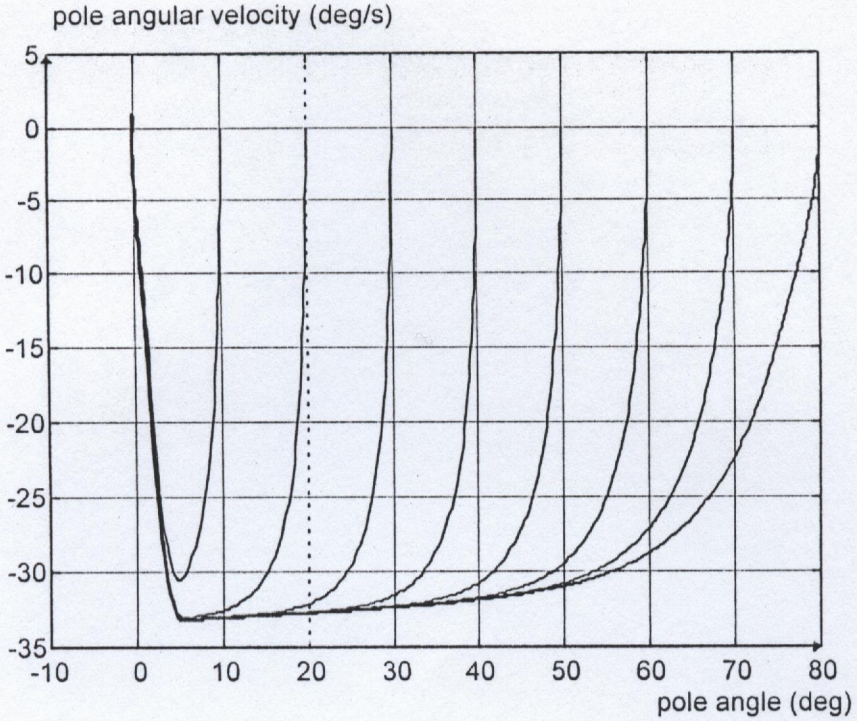


Fig. 5.12.c : Plan de phase pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

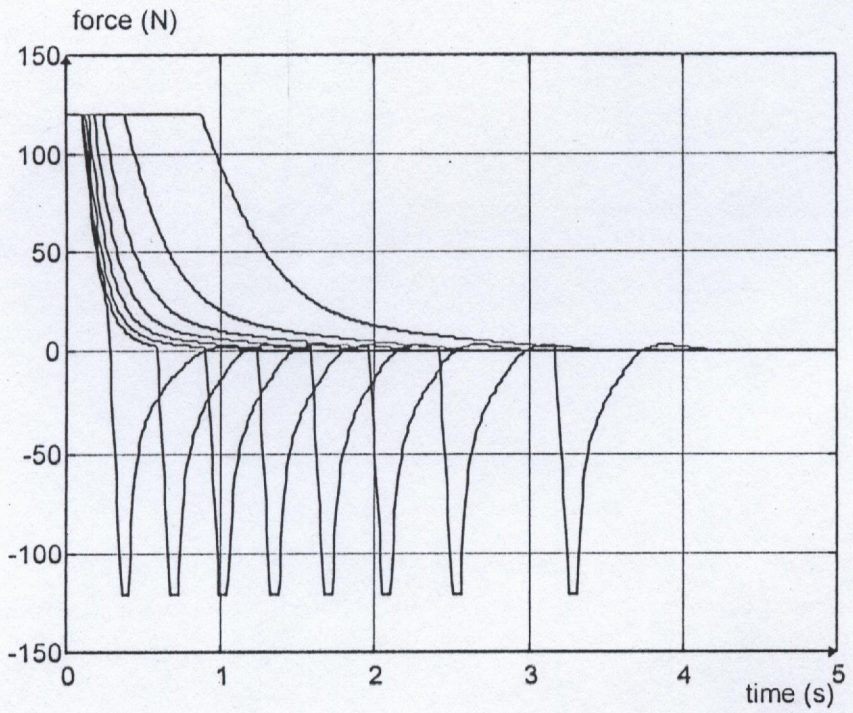


Fig. 5.12.d : Variations de la force pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

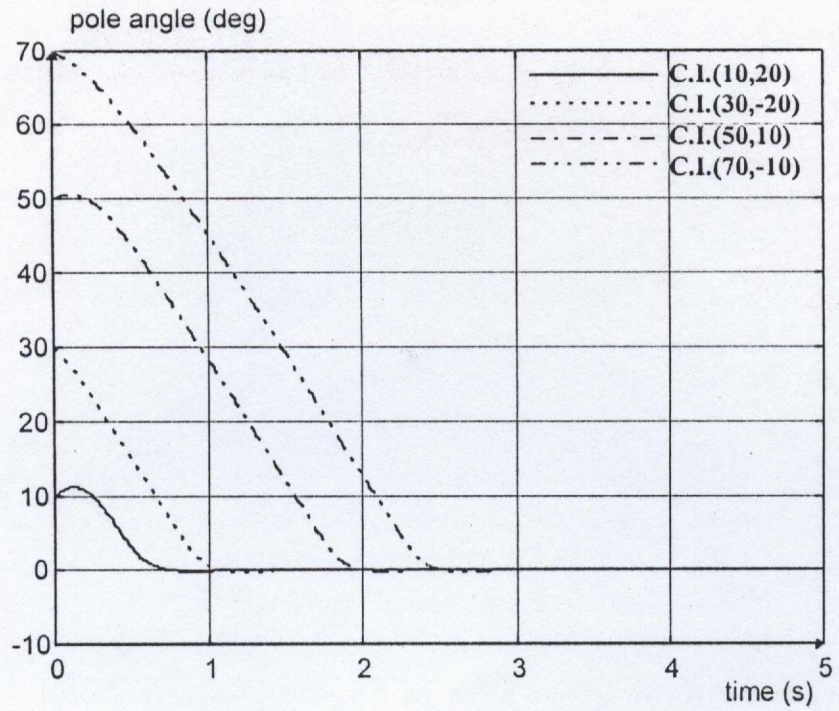


Fig. 5.13.a : Variations de l'angle pour les conditions initiales :

(10 °,20 °/s),(30 °,-20 °/s),(50 °, 10 °/s),(70 °,-10 °/s)

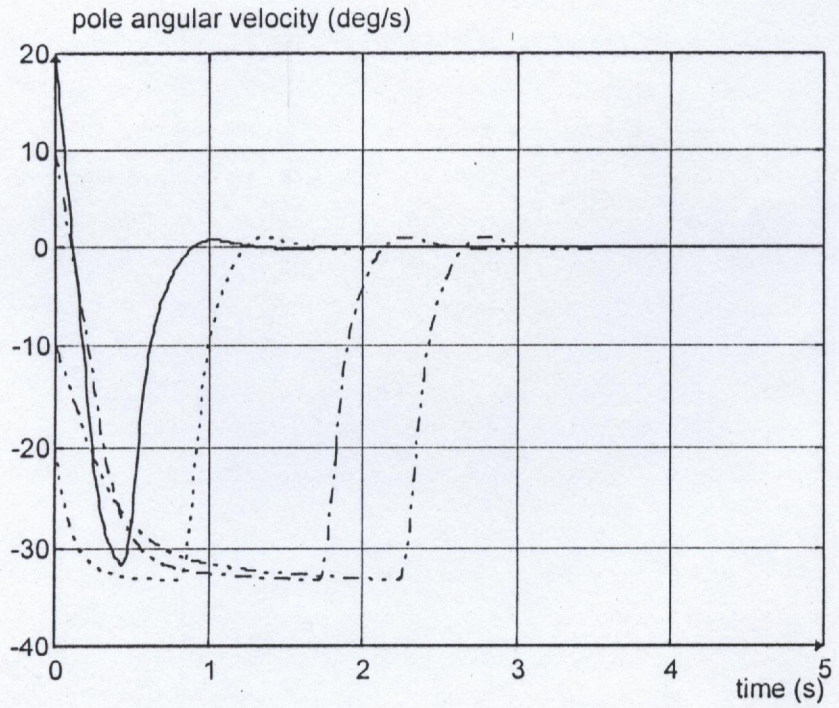


Fig. 5.13.b : Variations de la vitesse angulaire pour les conditions initiales :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

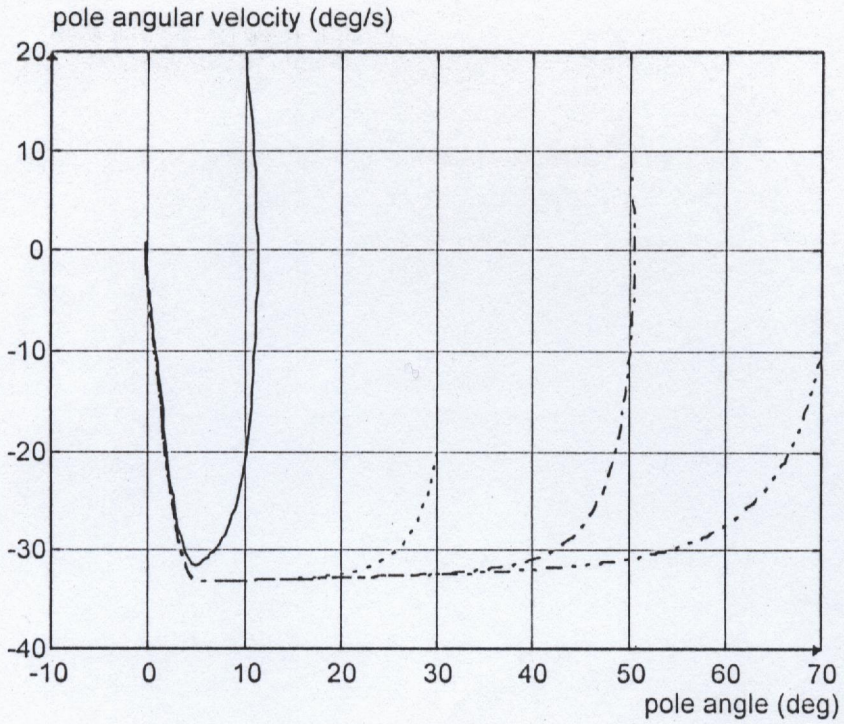


Fig. 5.13.c : Plan de phase à partir des point s :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

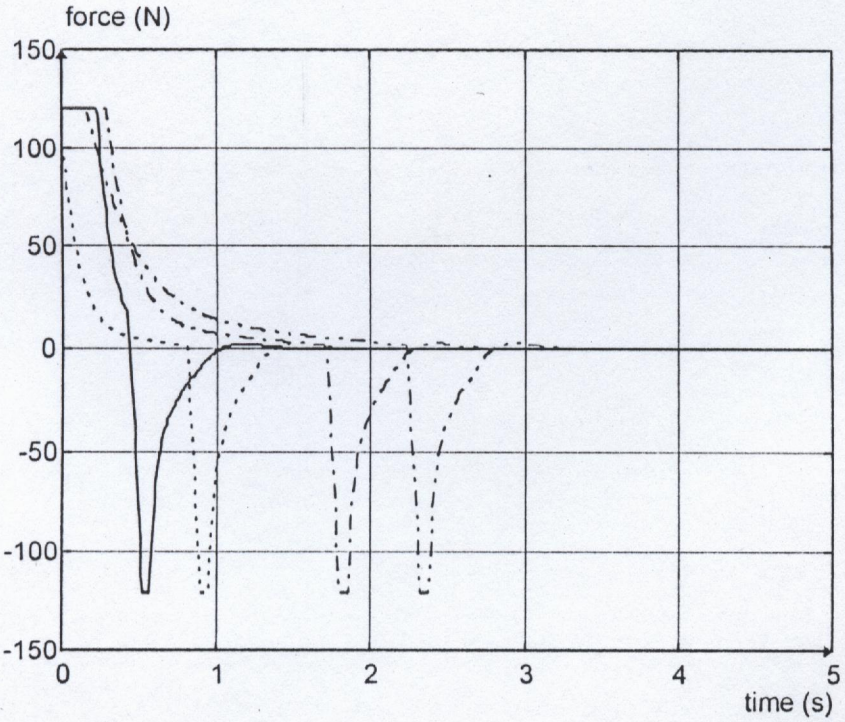


Fig. 5.13.d : Variations de la force à partir des conditions initiales :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

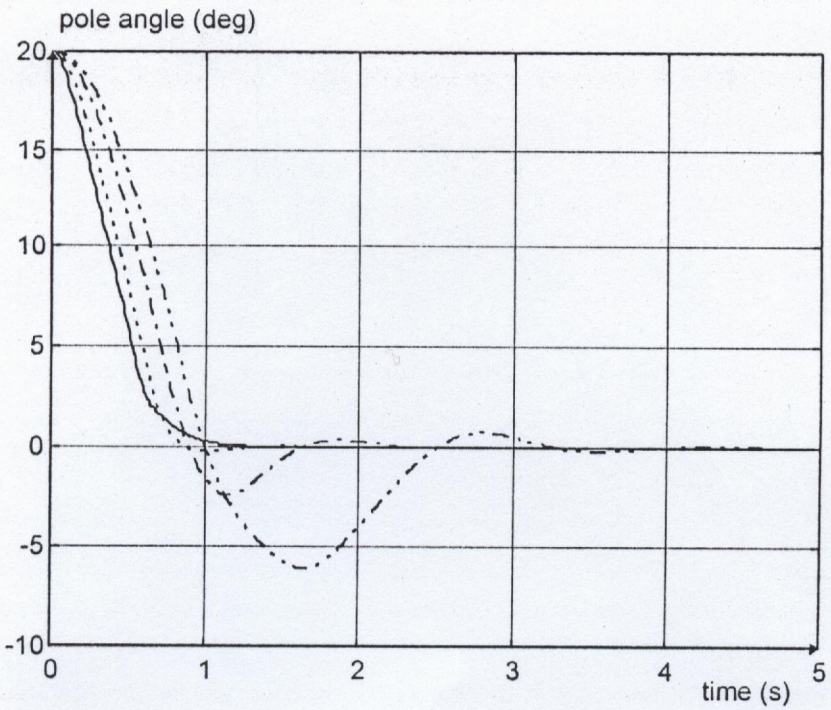


Fig. 5.14.a : Variations de l'angle pour des pendules de longueur variable (0.25,0.5,1 et 1.5m)

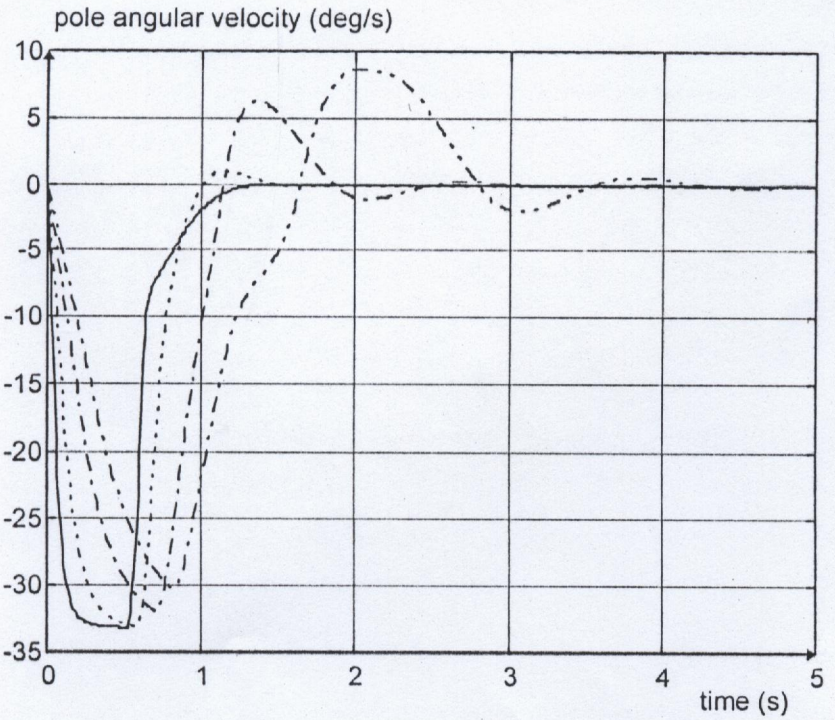


Fig. 5.14.b : Variations de la vitesse angulaire pour des pendules de longueur variable (0.25,0.5,1 et 1.5m)

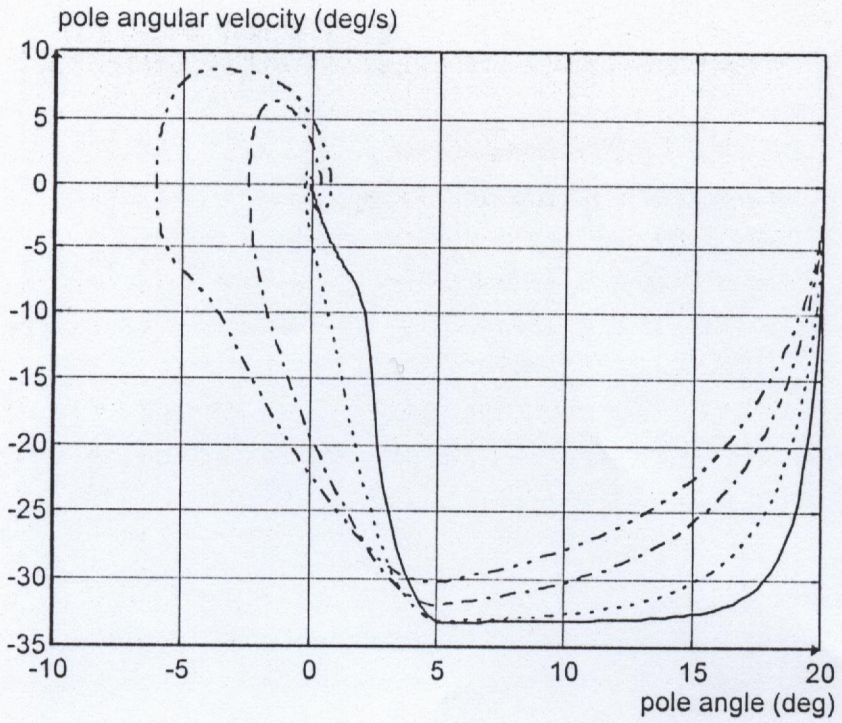


Fig. 5.14.c : Plan de phase pour des pendules de longueur variable
(0.25,0.5,1 et 1.5m)

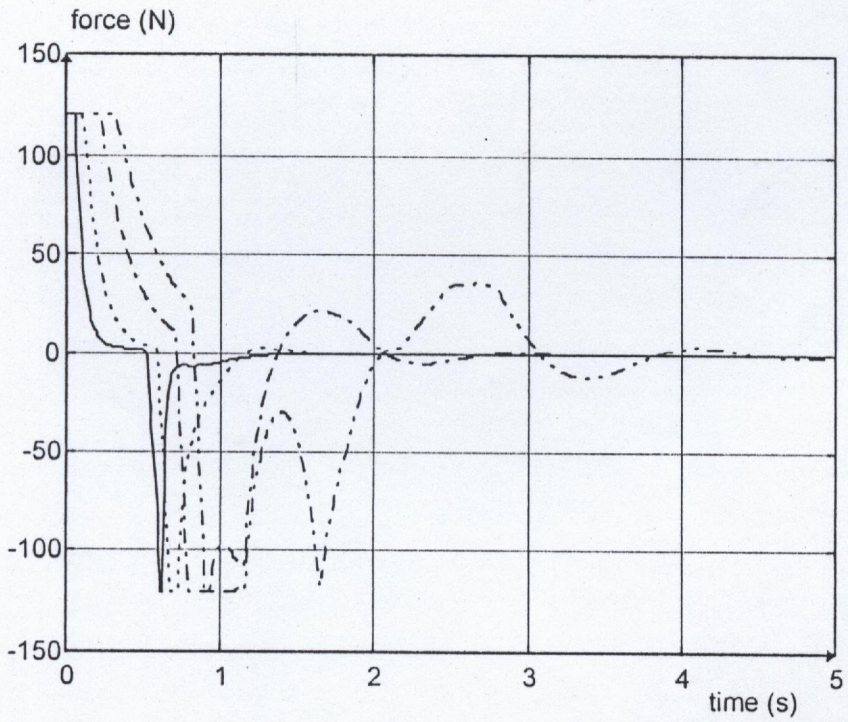


Fig. 5.14.d : Variations de la force pour des pendules de longueur variable
(0.25,0.5,1 et 1.5m)

CONCLUSION

Conclusion

Une nouvelle approche de conception des contrôleurs flous a été développée. C'est une approche mixte qui réside dans la combinaison des deux paradigmes :les réseaux de neurones et les algorithmes génétiques

Dans cette approche on a considéré l'implémentation structurelle d'un système d'inférence flou de type TPE dans un réseau multicouche. Les grandeurs ajustables de ce réseau sont les largeurs des univers de discours et les poids des connexions qui sont assujettis à une certaine contrainte. L'optimisation de ces grandeurs est réalisée par un AG en minimisant une certaine fonction coût. En vu de respecter la contrainte imposée sur les poids les opérations de croisement et de mutation ont été perturbées.

L'algorithme développé a été testé pour la commande du pendule inversé. Dans un premier cas nous avons construit par apprentissage un contrôleur flou à trois variables linguistiques pour toutes les variables d'E/S. Bien que composé d'un nombre restreint de règles (9 règles) ce contrôleur a donné de très bons résultats. Ensuite nous avons construit un contrôleur à cinq variables linguistiques (25 règles),la robustesse s'est améliorée mesurée ici par la capacité de ramener le pendule à la verticale pour des conditions initiales variables et pour différentes valeur des paramètres du pendule. La simulation a été effectuée sous Borland Pascal 7.0.

En vu d'améliorer cette nouvelle méthodologie basée sur un apprentissage hors ligne on propose une direction de recherche intéressante qui consiste à l'ajustement en ligne des paramètres du contrôleur flou. Bien que les AG soient moins adaptés au calcul en temps réel on propose de travailler sur un horizon d'erreur fini et sur quelques générations ce qui permettra de réduire considérablement le temps mis par les AG à converger vers la bonne solution en utilisant aussi d'autres techniques d'aide à la convergence.

Aussi la combinaison de ces deux approches (apprentissage en ligne et hors ligne) présente une technique attractive pour la conception des contrôleurs flous adaptatifs ainsi la méthode hors ligne est utilisée pour générer une base de règles initiale qui peut être modifiée en ligne.

Bibliographie

[1] C.C.Lee , " Fuzzy logic in control systems : Fuzzy logic controller -Part I ", IEEE Trans. Syst. Man Cybern. . Vol. 20 , N°. 2 , Mar. / Apr. 1990 , PP. 404 - 418 .

[2] C.C.Lee , " Fuzzy logic in control systems : Fuzzy logic controller -Part II ", IEEE Trans. Syst. Man Cybern. . Vol. 20 , N°. 2 , Mar. / Apr. 1990 , PP. 419 - 435 .

[3] J.Lu ,W.Jasper and G.K.Lee , " A multivariable self-learning Fuzzy control algorithm for deyeing processes ", Proceedings of the American control conference .Baltimore , Maryland .June 1994 , PP. 983 - 987 .

[4] M.M.Gupta ,J.B.Kiszka and J.M.Trojan , " Multivariable structure of Fuzzy control systems ", IEEE Trans. Syst. Man Cybern. .Vol. SMC-16 ,N° 5 ,Sept. / Oct. 1986 , PP. 638 - 655 .

[5] A.Belmehdi et A. Kara Mostefa ," Conception d'un régulateur auto-ajustable à logique floue pour un bioréacteur anaerobie ", Projet de fin d'étude (Ing) ,Institut d'électronique Constantine ,Juin 1994 .

[6] M.J.Willis ,C.Di Massimo ,G.A.Montague ,M.T.Tham and A.J.Morris ," Artificial neural networks in process engineering " , IEE Proceedings-D ,Vol. 138 ,N° 3 ,May 1991 , PP. 256 - 266 .

[7] T.Low ,T.Lee and H.lim , " A methodology for Neural Network training for control of drives with non linearities " , IEEE Trans. on industrial electronics ,vol.39, N°2 , Apr 1993 , PP. 243 - 249.

[8] J.Tanomaru and S. omatu , " Process control by on-line trained Neural controllers " , IEEE Trans. on industrial electronics, Vol.39 , N°6 , Dec. 1992 , PP.511-521.

[9] T.Fukuda , T.Shibata , M.Tokita and T.Mitsuoka , "Neuromorphic control : adaptation and learning " , IEEE Trans. on industrial electronics , Vol.39 , N°6 , Dec 1992 , PP. 497-503.

[10] H.M.Tai, J.Wang, and K.Ashenayi , " A neural network-based tracking control system" IEEE Trans. on industrial electronics , Vol.39, N°6 , Dec. 1992 , PP. 504-510.

[11] T. Fukuda and T.Shibata , "Theory and applications of neural networks for industrial control systems", IEEE Trans. on industrial electronics , Vol.39 , N°6 , Dec. 1992 , PP. 472-489.

[12] Richard P.Lippmann , "An introduction to computing with neural nets ", IEEE ASSP Magazine Vol.4 , April 1987 , PP. 4-22.

[13] P. Melsa, "Neural networks : a conceptual overview , "Technical report T.R.C.89-08.Tellabs Research center ,Mishawaka ,Aug. 1989 .

[14] S. K.Pal and S.Mitra , "Multilayer perceptron, Fuzzy sets, and classification ", IEEE Trans. on neural networks , Vol.3 , N°5 , Sept 1992 , PP. 683-697.

[15] J. R.Jang , "Self learning Fuzzy controllers based on temporal back propagation",IEEE Trans. on neural networks ,Vol.3 ,N°5, Sept 1992, PP. 714-723

[16] H. R.Berenji and P.Khedkar, "Learning and Tuning Fuzzy logic controllers through reinforcements",IEEE Trans. on neural networks, Vol.3, N°5, Sept1992, PP.724-740.

[17] A.Bachtarzi , " Commande des systèmes à structures variables (applications à la poursuite de trajectoires) ", Thèse de Magister ,institut d'électronique ,Constantine, 1995 .

[18] A.U.Levin and K.S.Narendra , "Control of Non linear dynamical systems using neural networks : controllability and stabilization ", IEEE Trans. on Neural Networks , Vol. 4 , N° 2 March 1993 , PP. 192 - 206 .

[19] A.U.Levin and K.S.Narendra , "Control of Non linear dynamical systems using neural networks - Part II : observability ,identification and control ", IEEE Trans. on Neural Networks , Vol. 7 , N° 1 January 1996 , PP. 30 -42 .

[20] B.Mendil , "Contrôle neuronal flou ", Thèse de Magister ,institut d'électronique , Setif , 1994 .

[21] C.Wang ,W.Wang , T.Lee and P.S Tseng,"Fuzzy B-spline membership function (BMF) and its applications in Fuzzy-neural control ", IEEE Trans. on Systems ,Man,and Cybernetics ,Vol. 25 ,N° 5 , may 1995 ,PP. 841 -851 .

[22] Y.Jin ,J.Jiang and J.Zhu ,"Neural network based Fuzzy identification and its application to modeling and control of complex systems ", IEEE Trans. on Systems ,Man,and Cybernetics ,Vol. 25 ,N° 6 ,June 1995 ,PP. 990 -997 .

[23] J.Zhang and A.Julian Morris,"Process fault diagnosis using Fuzzy neural networks ", Proceedings of the American Control Conference,Baltimore,Maryland ,June 1994 PP. 971-975 .

[24] Syed I.Ashon ,"Petri net models of Fuzzy neural networks ", IEEE Trans. on Systems , Man,and Cybernetics ,Vol. 25 ,N° 6 ,June 1995 ,PP. 926 -932 .

[25] J.J.Buckley and Y.Hayashi ,"Fuzzy neural networks : a survey ",Fuzzy Sets and Systems , 66 ,1994 ,PP. 1-13 .

[26] S.K.Halgamuge ,M.Glesner ,"Neural networks in designing Fuzzy systems for real world applications ", Fuzzy Sets and Systems, 65 ,1994 ,PP. 1-12 .

[27] E.Ikonen and K.Najim ,"Fuzzy neural networks and application to the FBC process " , to appear in IEE Proceedings on control theory and applications.

[28] W.Pedrycz ,"Genetic algorithms for learning in Fuzzy relational structures ", Fuzzy Sets and Systems, 69 ,1995 ,PP. 37-52 .

[29] David E.Goldberg ,"Algorithmes génétiques exploration ,optimisation et apprentissage automatique ",Edition Addison-Wesley 1994 .

[30] J.Kim and B.P.Zeigler ,"Designing Fuzzy logic controllers using a multiresolutional search paradigm ",IEEE Trans. on Fuzzy Systems , Vol. 4 ,No 3 , august 1996 , PP. 213-226 .

[31] C.L.Karr and E.J.Gentry ,"Fuzzy control of pH using genetic algorithms ",IEEE Trans. on Fuzzy Systems ,Vol. 1 ,N° 1 ,February 1993 , PP. 46-53 .

[32] A.Homaifar and Ed McCormick ,"Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms ",IEEE Trans. on Fuzzy Systems, Vol. 3 ,N° 2 ,May 1995 , PP. 129-139 .

[33] D.A.Linkens and H.O.Nyongesa ,"Genetic Algorithms for Fuzzy control-Part 1 :Off-line system development and application ",IEE Proceedings-Control Theory and applications Vol. 142 ,N° 3 ,May 1995 , PP. 161-176 .

[34] D.A.Linkens and H.O..Nyongesa ,"Genetic Algorithms for Fuzzy control-Part 2 :Online system development and application ",IEE Proceedings-Control Theory and applications Vol. 142 ,N° 3 ,May 1995 , PP. 177-185 .

[35] Y.H.Joo ,H.S.Hwang ,K.B.Kim and K.B.Woo ,"Fuzzy system modeling by Fuzzy partition and GA hybrid schemes ",Fuzzy Sets and Systems ,86,1997, PP. 279-288 .

[36] C.T.Lin and Y.C.Lu ,"A neural Fuzzy system with Fuzzy supervised learning ",IEEE Trans. on Syst. Man ,and Cybernetics-Part B :cybernetics.Vol. 26 ,N° 5 ,October 1996 , PP. 744-763 .

[37] F.R.Rubio, M. Berenguel and E.F.Camacho ,"Fuzzy logic control of a solar power plant ",IEEE Trans. on Fuzzy Systems ,Vol. 3 ,N° 4 ,November 1995 , PP. 459-468 .

[38] K.Mesghouni ,"Application des algorithmes génétiques à la commande des MAS - investigation dans le domaine de la commande adaptative ",Rapport de DEA de génie électrique ,CEGELY Lyon ,France .

[39] D.A.Linkens and H.O..Nyongesa ,"Learning systems in intelligent control : an appraisal of Fuzzy ,neural and genetic algorithm control applications ",IEE Proceedings-Control Theory and applications ,Vol. 143 ,N° 4 ,July 1996 , PP. 367-386 .

[40] B.Sareni ,"Algorithmes génétiques standards ",Rapport interne R-97-0.1 ,Septembre 1997 ,CEGELY Lyon , France .

[41] S.Horikawa ,T.Furuhashi and Y.Uchikawa ,"On Fuzzy modeling using Fuzzy neural networks with the back-propagation algorithm ",IEEE Trans. Neural Networks , Vol. 3, N° 5, Sept. 1992 .

[42] C.T.Lin and C.S.G.Lee ,"Real-time supervised structure/parameter learning for Fuzzy neural network ",IEEE International Conference on Fuzzy Systems ,San Diego ,USA, 1992 , PP. 1283-1291 .

[43] C.T.Lin and C.S.G.Lee ,"reinforcement structure/parameter learning for neural network based Fuzzy logic control systems ",Proceedings of IEEE international conference on Fuzzy systems ,San Francisco ,CA,USA,1993 , PP. 88-93 .

[44] L.X.Wang and J.M.Mendel ,"Fuzzy basis functions ,universal approximation and orthogonal least squares ",IEEE Trans. on Neural Networks,1992 , PP. 807-814 .

[45] S.Horikawa,T.Furuhashi,S.Ouma and Y.Uchikawa ,"A Fuzzy controller using a neural network and its capability to learn expert's control rules ",Proceedings of international conference on Fuzzy logic and neural networks ,1990 ,Vol. 1, PP. 103-106 .

[46] W.Pedrycz ,"Fuzzy neural networks and neurocomputations ",Fuzzy Sets and Systems, 1993 , 56,PP. 1-28 .

[47] H.Ishibushi,H.Okada and H.Tanaka ,"Fuzzy neural networks with Fuzzy weights and Fuzzy biases ",Proceedings of int. Joint.conf. on Neural networks ,San Francisco CA, 1993 , PP. 1650 - 1655 .

[48] Y.Hayashi,J.J.Buckley and E.Czogala ,"Systems engineering applications of Fuzzy neural networks ",Proc. of internat.joint conf. on Neural Networks ,Baltimore ,MD,1992 Vol. 2 ,PP. 412-418 .

[49] P.Thrift ,"Fuzzy logic synthesis with genetic algorithms ",Proc. of the fourth internat. conf. On Genetic Algorithms ,1991 , PP. 509-513 .

[50] C.L.Karr ,"Design of an adaptive Fuzzy logic controller using a genetic algorithm", Proc. of the fourth Int. conf. on Genetic Algorithms ,1991 ,PP. 450-457 .

[51] H.Nomura,I.Hayashi and N.Wakami ,"A self tuning method of Fuzzy reasoning by genetic algorithm ",Proc. Int. Fuzzy syst.intell.contr.conf. ,1992 ,PP. 236-245 .

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE L

A RECHERCHE SCIENTIFIQUE

----------o----------

UNIVERSITÉ FERHAT ABBAS

SÉTIF

MÉMOIRE

Présenté par

TITEL FAOUZI

Pour obtenir le titre de **Magister**

de l'Institut d'Électronique

Option

CONTROLE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

----------o----------

UNIVERSITÉ FERHAT ABBAS

SÉTIF

MÉMOIRE

Présenté par

TITEL FAOUZI

Pour obtenir le titre de **Magister**

de l'Institut d'Électronique

Option

CONTROLE

CONCEPTION DE CONTRÔLEUR À LOGIQUE FLOUE PAR LES ALGORITHMES GÉNÉTIQUES

Date de soutenance : 28/06/1998

Devant le jury composé de :

Président :	Mr. K. BENMAHAMMED	Prof. Université de Sétif
Rapporteur :	Mr. K. BELARBI	M.C. Université de Constantine
Examinateurs :	Mr. A. BENNIA	M.C. Université de Constantine
	Mr. D. SLIMANI	M.C. Université de Sétif
	Mr. A. BARTIL	C.C. Université de Sétif

Année 1997/98

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

----------o----------

UNIVERSITÉ FERHAT ABBAS

SÉTIF

MÉMOIRE

Présenté par

TITEL FAOUZI

Pour obtenir le titre de **Magister**

de l'Institut d'Électronique

Option

CONTROLE

CONCEPTION DE CONTRÔLEUR À LOGIQUE FLOUE PAR LES ALGORITHMES GÉNÉTIQUES

Date de soutenance : 28/06/1998

Devant le jury composé de :

Président :	Mr. K. BENMAHAMMED	Prof. Université de Sétif
Rapporteur :	Mr. K. BELARBI	M.C. Université de Constantine
Examinateurs :	Mr. A. BENNIA	M.C. Université de Constantine
	Mr. D. SLIMANI	M.C. Université de Sétif
	Mr. A. BARTIL	C.C. Université de Sétif

Année 1997/98

Dédicaces

Je dédie ce modeste travail à mes très chers parents qui m'ont encouragé tout le long de mon cycle d'étude et m'ont guidé vers la réussite.

Je le dédie également à mes frères et soeurs .

Ainsi qu'à Raouf , Sandra , Rima , Marwan et Marwa .

Remerciement

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

Je remercie Mr. K.Benmahammed ,professeur à l'université de Setif de l'honneur qu'il m'a fait en présidant le jury de cette thèse .

Je tiens à remercier Mrs: A.Bennia, maître de conférences à l'université de Constantine, D. Slimani, maître de conférences à l'université de Sétif et A. Bartil, chargé de cours à l'université de Sétif, pour l'intérêt qu'ils ont accordé à mon travail en acceptant d'examiner cette thèse .

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

Résumé

Une nouvelle méthodologie de conception des contrôleurs à logique floue été développée.
Le contrôleur est projeté sur un réseau interconnecté multicouche, les poids et les entrées du réseau sont numériques. Il s'agit alors de concevoir simultanément les paramètres des fonctions d'appartenance triangulaires symétriques et l'ensemble des règles d'inférence. La solution utilise les algorithmes génétiques qui sont des procédures de recherche basées sur les mécanismes de la génétique naturelle pour trouver le contrôleur qui minimise une certaine fonction coût.
Pour étudier ses performances, cette nouvelle méthode a été appliquée pour la conception d'un contrôleur flou pour un pendule inversé .

Abstract

A new designing methodology for fuzzy logic controller is developed. The controller is implemented through a multi-layer network with crisp inputs and real valued weights. The problem is then to find simultaneously the optimal parameters of symmetric triangular membership functions and the rule set. The solution procedure uses genetic algorithms ,a recent search and optimisation procedures based on the mechanics of natural genetics, to find the controller that minimises a certain cost function.
This performance of the new design procedure is demonstrated on an application to the design of a fuzzy controller for an inverted pendulum system.

SOMMAIRE

INTRODUCTION 1

CHAPITRE II : ***LES ALGORITHMES GENETIQUES***

II.1 Introduction 5
II.2 Formulation du problème d'optimisation 5
II.3 Principe de fonctionnement des AG 6
II.4 Les mécanismes d'un AG simple 7
II.4.1 La reproduction 7
II.4.2 Le croisement 10
II.4.3 La mutation 11
II.4.4 La notion de schéma 11
II.4.4.1 Effet de la reproduction 12
II.4.4.2 Effet du croisement 13
II.4.4.3 Effet de la mutation 14
II.5 La sélection des individus d'une nouvelle génération 14
II.5.1 Sélection par descendance 15
II.5.2 Sélection par compétition 15
II.5.3 Steady state selection 15
II.5.4 Selective breeding selection 16
II.6 Description du codage 16
II.7 Description du décodage 16
II.8 Organigramme de l'AG 17
II.9 La convergence de l'AG 18
II.9.1 Critère de convergence 18
II.9.2 Taux de convergence 18
II.10 Conclusion 19

CHAPITRE III : ***LES APPROCHES DE CONCEPTION D'UN FLC***

III.1 La logique floue 21

III.1.1 Introduction 21
III.1.2 Contrôleur à logique floue 21
III.1.2.1 La fuzzification 21
III.1.2.2 Bloc de contrôle 22
III.1.2.3 La défuzzification 24
III.1.2.4 Paramètres de conception d'un FLC 24
III.2 Les approches de conception d'un FLC 25
III.2.1 L'approche connexioniste 25
III.2.1.1 Les réseaux de neurones 25
a) Définition 25
b) Architecture des réseaux de neurones 26
c) Entraînement des réseaux de neurones 27
d) L'algorithme de la rétropropagation 27
III.2.1.2 Les réseaux de neurones flous 28
a) Définition 28
b) Les différentes approches des FNN 28
c) Les architectures des FNN 29
III.2.2 L'approche directe 37
III.2.2.1 L'application des AG pour les contrôleurs flous 37
a) Définition du problème :paramètres à optimiser 38
b) Codage des paramètres 39
III.2.2.2 L'apprentissage des systèmes flous par les AG.....41
III.3 Conclusion 42

CHAPITRE IV : ***CONCEPTION D'UN CONTROLEUR FLOU PAR LES AG***

IV.1 Introduction 45
IV.2 Implémentation du contrôleur flou 45
IV.3 Description de l'algorithme d'apprentissage 49
IV.3.1 Codage des paramètres 50
IV.3.2 Mécanisme de l'AG 50
a) Reproduction 50
b) Croisement 51

c) Mutation 51
d) La sélection des individus d'une nouvelle génération 52
IV.3.3 Décodage des paramètres 52
IV.4 Interaction du module contrôleur avec l'AG 52
IV.5 Conclusion 54

CHAPITRE V : ***RESULTATS DE SIMULATION***

V.1 Introduction 57
V.2 Le système du pendule inversé 57
V.3 Le contrôle du pendule inversé 58
V.3.1 Structure de simulation de la commande 58
V.3.2 Conception du contrôleur 59
V.3.3 Contrôleur 333 60
V.3.3.1 Résultats de simulation 62
V.3.3.2 Robustesse du contrôleur 333 64
V.3.4 Contrôleur 555 74
V.3.4.1 Résultats de simulation 74
V.3.4.2 Robustesse du contrôleur 555 75
V.4 Conclusion 77

CONCLUSION 86
BIBLIOGRAPHIE 88

Liste des figures

Figures :

Fig.2.1 : Croisement à un site (k=3) 10

Fig.2.2 : Principe de la mutation 11

Fig.2.3 : Organigramme de l'AG 17

Fig.3.1 : Structure d'un contrôleur à logique floue 21

Fig.3.2 : Formes des fonctions d'appartenance 23

Fig.3.3 : Univers de discours TP 23

Fig.3.4 : Univers de discours TPE avec 5 ensembles flous 23

Fig.3.5 : Réseau de Hopfield 26

Fig.3.6 : Structure d'un réseau statique 27

Fig.3.7 : Configuration d'un FNN régulier 30

Fig.3.8 : Configuration d'un FNN avec des entrées ,poids et sorties flous 31

Fig.3.9 : processeur logique 33

Fig.3.10 : Architecture d'un FLP 34

Fig.3.11 : Structure d'un FNN comme contrôleur flou 35

Fig.3.12 : Structure d'un FNN proposé dans [36] 37

Fig.3.13 : Fonction d'appartenance triangulaire de centre C et de largeur W 39

Fig.3.14 : Codage typique d'une base de règle floue 41

Fig.3.15 : Mécanisme d'apprentissage off-line 42

Fig.4.1 : Architecture du réseau contrôleur 48

Fig.4.2 : Univers de discours TPE de largeur L 49

Fig.4.3 : Principe du croisement conditionné 51

Fig.4.4 : Principe de la mutation conditionnée 52

Fig.4.5 : Diagramme des interactions de l'AG ,FLC et le modèle de simulation 53

Fig.4.6 : Organigramme de l'opération du contrôleur 54

Fig.5.1 : Le système du pendule inversé 58

Fig.5.2 : Structure de commande du pendule inversé 59

Fig.5.3 : Les fonctions d'appartenance du contrôleur 333 64

Fig.5.5.a : Variations de l'angle à partir du point (20deg , 0deg/s) 66
Fig.5.5.b : Variations de la vitesse angulaire à partir du point (20deg , 0deg/s) 66
Fig.5.5.c : plan de phase à partir de (20deg , 0deg/s) 67
Fig.5.5.d : Variations de la force 67
Fig.5.6.a : Variations de l'angle pour des conditions initiales différentes 68
Fig.5.6.b : Variations de la vitesse angulaire pour des conditions initiales différentes 68
Fig.5.6.c : plan de phase pour des conditions initiales différentes 69
Fig.5.6.d : Variations de la force pour des conditions initiales différentes 69
Fig.5.7.a : Variations de l'angle pour les conditions initiales :
(10,20) ,(30,-20) ,(50,10) ,(70,-10) 70
Fig.5.7.b : Variations de la vitesse angulaire pour les conditions initiales :
(10,20) ,(30,-20) ,(50,10) ,(70,-10) 70
Fig.5.7.c : Plan de phase à partir des points :(10,20) ,(30,-20) ,(50,10) ,(70,-10) 71
Fig.5.7.d : Variations de la force à partir des points : (10,20) ,(30,-20) ,(50,10) ,(70,-10) 71
Fig.5.8.a : Variations de l'angle pour des pendules de longueur variable 72
Fig.5.8.b : Variations de la vitesse angulaire pour des pendules de longueur variable 72
Fig.5.8.c : Plan de phase pour des pendules de longueur variable 73
Fig.5.8.d : Variations de la force pour des pendules de longueur variable 73
Fig.5.10 : Les fonctions d'appartenance du contrôleur 555 76
Fig.5.11.a : Variations de l'angle à partir du point (20deg , 0deg/s) 78
Fig.5.11.b : Variations de la vitesse angulaire à partir du point (20deg , 0deg/s) 78
Fig.5.11.c : plan de phase à partir de (20deg , 0deg/s) 79
Fig.5.11.d : Variations de la force 79
Fig.5.12.a : Variations de l'angle pour des conditions initiales différentes 80
Fig.5.12.b : Variations de la vitesse angulaire pour des conditions initiales différentes 80
Fig.5.12.c : plan de phase pour des conditions initiales différentes 81
Fig.5.12.d : Variations de la force pour des conditions initiales différentes 81
Fig.5.13.a : Variations de l'angle pour les conditions initiales :
(10,20) ,(30,-20) ,(50,10) ,(70,-10) 82
Fig.5.13.b : Variations de la vitesse angulaire pour les conditions initiales :
(10,20) ,(30,-20) ,(50,10) ,(70,-10) 82
Fig.5.13.c : Plan de phase à partir des points :(10,20) ,(30,-20) ,(50,10) ,(70,-10) 83

Fig.5.13.d : Variations de la force à partir des points : (10,20) ,(30,-20) ,(50,10) ,(70,-10) ...83
Fig.5.14.a : Variations de l'angle pour des pendules de longueur variable 84
Fig.5.14.b : Variations de la vitesse angulaire pour des pendules de longueur variable 84
Fig.5.14.c : Plan de phase pour des pendules de longueur variable 85
Fig.5.14.d : Variations de la force pour des pendules de longueur variable 85

Tableaux :

Tab.3.1 : Règles de contrôle floues d'un procédé industriel 35
Tab.5.1 : Intervalles de recherche des paramètres pour le contrôleur 333.....61
Tab.5.2 : Les valeurs des paramètres génétiques 61
Tab.5.3 : Base de règles floues du contrôleur 333 63
Tab.5.4 : Intervalles de recherche des paramètres pour le contrôleur 555 74
Tab.5.5 : Base de règles floues du contrôleur 555 75

INTRODUCTION

Introduction

Depuis la première application du formalisme de la logique floue à la commande des systèmes proposée par Mamdani, plusieurs études ont montré que le contrôle à logique floue est une méthode adéquate pour la commande des procédés mal définis ou complètement inconnus qui ne peuvent pas être modélisé facilement d'une manière mathématique.

L'approche traditionnelle pour la conception floue est basée sur les connaissances acquises par des opérateurs experts, bien que cette approche ait prouvé son efficacité dans plusieurs applications, il se peut cependant que les opérateurs ne peuvent pas transcrire leur connaissance et expérience sous forme de contrôleur à logique floue. De plus, il arrive parfois que le domaine d'expertise ne soit pas disponible.

Pour ces raisons, plusieurs chercheurs se sont intéressé à l'élaboration des méthodes optimales et systématiques pour la conception des contrôleurs flous. Deux directions de recherche sont apparues :l'approche connexioniste et l'approche directe.

L'approche connexioniste consiste à combiner la théorie des réseaux de neurones artificiels (ANN, Artificial Neural Networks) et les systèmes flous pour construire ce qu'on appelle un réseau de neurones flous (FNN, Fuzzy Neural Network). Ce réseau constitué de plusieurs couches contenant des neurones qui réalisent les opérations floues de base telles que la fuzzification, l'opération AND, l'opération OR et la défuzzification est utilisé pour implanter le mécanisme d'inférence flou.

L'apprentissage du réseau permet d'ajuster les paramètres de conception et les inférences sont extraites après convergence.

Dans l'approche directe, le problème de conception du contrôleur flou est traité comme un problème d'optimisation qui consiste à trouver un ensemble optimal des fonctions d'appartenance ou/et des règles d'inférence qui minimisent une certaine fonction coût. Du à la complexité de l'espace de recherche, ce problème d'optimisation est résolu en utilisant les algorithmes génétiques (AG) qui sont des procédures de recherche basées sur les mécanismes de la génétique naturelle.

L'objectif de ce travail s'inscrit dans la même direction et concerne la conception des contrôleurs à logique floue par la combinaison des deux paradigmes :les réseaux de neurones et les algorithmes génétiques.

Pour cela le contrôleur à logique floue est projeté sur un réseau interconnecté multicouche et l'algorithme génétique est utilisé pour optimiser les paramètres de ce réseau. Il s'agit alors de concevoir simultanément les paramètres des fonctions d'appartenance triangulaires symétriques et l'ensemble des règles d'inférence. La solution utilise les algorithmes génétiques qui sont des procédures de recherche basées sur les mécanisme de la génétique naturelle pour trouver le contrôleur qui minimise une certaine fonction coût.

Pour étudier ses performances, cette nouvelle méthode a été appliquée pour la conception d'un contrôleur flou pour un pendule inversé .

Ce travail est organisé en cinq chapitres :

- Dans le deuxième chapitre, il y a une description des outils nécessaires à la résolution des problèmes d'optimisation posés par la mise en œuvre des algorithmes génétiques.
- Le troisième chapitre présente un aperçu sur les contrôleurs à logique floue et traite les différentes approches de conception de systèmes flous à savoir l'approche directe et l'approche connexioniste.
- Le chapitre quatre comprend la nouvelle méthodologie de conception qu'on a proposé, c'est une approche mixte et concerne la combinaison des AG et les réseaux de neurones afin d'assurer une conception optimale du contrôleur flou.
- Le cinquième chapitre est consacré aux résultats des différents tests effectués pour la commande du système du pendule inversé.
- Et enfin une conclusion et perspectives font l'objet du dernier chapitre.

Les algorithmes génétiques

II.1 Introduction.

II.2 Formulation du problème d'optimisation.

II.3 Principe de fonctionnement des algorithmes génétiques.

II.4 Les mécanismes d'un algorithme génétique simple.

II.4.1 La reproduction.

II.4.2 Le croisement.

II.4.3 La mutation.

II.4.4 Notion de schéma.

II.4.4.1 Effet de la reproduction.

II.4.4.2 Effet du croisement.

II.4.4.3 Effet de la mutation.

II.5 La sélection des individus d'une nouvelle génération.

II.6 Description du codage.

II.7 Description du décodage.

II.8 Organigramme de l'algorithme génétique.

II.9 La convergence des algorithmes génétiques.

II.9.1 Critères de convergence .

II.9.2 Taux de convergence .

II.10Conclusion.

II.1 Introduction

Dans la pratique, un grand nombre de fonctions à optimiser ne sont pas dérivables et ne sont souvent même pas continues. Le monde réel à explorer est envahi de discontinuités, ce qui le rend bien moins adapté au calcul. Il est donc évident que les méthodes soumises aux contraintes de discontinuité et de dérivabilité ne sont adaptées qu'à une classe de problèmes très limitée. En plus elles ne sont pas suffisamment robustes. Ce type de méthodes dites classiques sont susceptibles d'être piégées dans des optimums locaux [38].

Récemment, une classe de méthodes est apparue, employant les principes d'évolution et d'hérédité de la nature et présentant une probabilité importante de convergence vers un optimum global de la fonction à optimiser. Ce sont des méthodes pseudo-aléatoires appelées "LES ALGORITHMES GENETIQUES".

Les algorithmes génétiques (AG) développés par John Holland sont des techniques d'optimisation stochastiques qui tentent d'imiter les processus d'évolution naturelle des espèces et de la génétique. Ils agissent sur une population d'individus assujettis à une sélection darwinienne :les individus(ou parents)les mieux adaptés à leur environnement survivent et peuvent se reproduire. Ils sont alors soumis à des mécanismes de recombinaisons analogues à ceux de la génétique. Des échanges de gènes entre parents résulte la création de nouveaux individus(ou enfants), qui permettent de tester d'autres configurations de l'espace de recherche.

Dans leurs recherche du maximum ces algorithmes ne nécessitent que la connaissance de la valeur de la fonction qu'on souhaite optimiser dite fonction coût(fonction qui doit être toujours positive).

II.2 Formulation du problème d'optimisation

En raison de l'analogie avec la théorie de l'évolution (survie des individus les mieux adaptés à leur environnement), l'algorithme génétique est naturellement formulé en terme de maximisation. Etant donnée une fonction f réelle à une ou plusieurs variables, le problème d'optimisation sur l'espace de recherche E s'écrit de la manière suivante :

$$\max_{x \in E} f(x) \qquad (2.1)$$

De plus, la fonction à optimiser par un algorithme génétique doit avoir des valeurs positives sur l'ensemble du domaine *E.* Dans le cas contraire, il convient d'ajouter aux valeurs de *f* une constante positive *Fmin* conformément à l'équivalence de (2.1) et (2.2).

$$\max_{x \in E} f(x)+Fmin \tag{2.2}$$

Dans beaucoup de problème, l'objectif est exprimé sous la forme de minimisation d'une fonction coût g ,

$$\min_{x \in E} g(x) \tag{2.3}$$

Le passage du problème de minimisation à un problème de maximisation est obtenu par transformation de la fonction g. La transformation souvent utilisée est :

$$\max_{x \in E} h(x) \tag{2.4}$$

avec :

$$h(x) = \begin{cases} G\max - g(x) & si \quad G\max \geq g(x) \\ 0 & \sin on \end{cases} \tag{2.5}$$

de la fonction h n'est pas unique. En effet, toute composition de la fonction g par une fonction quelconque Notons que le choix décroissante et monotone sur le domaine E, conduirait à un problème de maximisation équivalent à (2.3). on rencontre notamment dans la littérature la fonction de transformation h suivante [29].

$$h(x) = \frac{1}{1+g(x)} \tag{2.6}$$

II.3 Principe de fonctionnement des AG

Les AG effectuent le procédé d'optimisation en agissant sur une population de créatures artificielles (chaînes de caractères) analogues aux chromosomes en nature. Chaque créature ou individu représente un point de recherche dans l'espace des solutions à qui on associe une valeur de fonction coût, dont on veut obtenir la valeur maximum.

La population est aléatoirement crée, ensuite elle va tendre vers les meilleurs régions de l'espace de recherche en réalisant des opérations génétiques.

Les AG diffèrent fondamentalement des autres méthodes dans la recherche de l'optimum[29]:

1. Les AG utilisent un codage des paramètres d'origine du problème d'optimisation et non les paramètres eux même.
2. Les AG travaillent sur une population de points, au lieu d'un point unique.
3. Les AG n'utilisent que les valeurs de la fonction à optimiser, pas sa dérivée ou une autre connaissance auxiliaire.

4.Les AG utilisent des règles de transition probabilistes et non déterministes.

II.4 Les mécanismes d'un AG simple

A partir d'une population initiale d'individus crée aléatoirement, les AG génèrent de nouveaux individus plus performant que leurs prédécesseurs en effectuant des opérations génétiques.

Un AG simple est composé de trois opérateurs :la reproduction, le croisement et la mutation.

La reproduction est une version artificielle de la sélection naturelle, c'est un processus dans lequel chaque individu est copié en fonction des valeurs de la fonction coût. Le croisement est l'opérateur le plus dominant dans un AG, il permet à deux chaînes d'échanger des portions de leurs structures produisant ainsi de nouvelles chaînes. La mutation est un opérateur local qui est appliqué avec une très faible probabilité.

II.4.1 La reproduction (la selection des parents)

La reproduction ou la sélection des parents est un mécanisme qui consiste à former une nouvelle génération par une sélection aléatoire des chaines d'une population existante en fonction de leur performance(fonction coût).

Lors de cette phase, les individus les plus forts sont généralement dupliqués et forment les parents de la génération en cours, alors que les faibles disparaissent sans avoir la possibilité de se reproduire.

II.4.1.1 Les méthodes de sélection

Elles sont caractérisées par la probabilité P_s (a^t_j) qu'un individu a^t_j de la population (t) soit retenu pour participer à la recombinaison génétique.

a. La sélection proportionnelle(roue de loterie biaisée)

Ce mode de sélection des parents consiste à dupliquer chaque individu de la population proportionnellement à son adaptation dans son milieu.

Soit :

f_j la valeur de la fonction coût associée au j^{eme} individu.

f_s la somme des valeurs de cette fonction.

La sélection pouvant être faite en utilisant le rapport (f_j/f_s) pour réaliser une roulette pondérée ou chaque individu occupe une surface proportionnelle au rapport précédent. Des tirages aléatoires sur cette roulette donneront les chaînes qui participeront à la prochaine population. De cette façon, les chaînes bien adaptées ont un plus grand nombre de descendants dans les générations suivantes.

b. La sélection à reste stochastique

Dans ce mode de sélection, le nombre de copies $n(a^t_j)$ d'un individu a^t_j est directement fixé par le rapport (f_j/f_{moy}) ou f_{moy} est la valeur moyenne de la fonction coût. Dans un premier temps, on reproduit chaque individu (partie entière de (f_j/f_{moy})) fois :

$$n(a^t_j)=\text{partie entière}[f_j/f_{moy}] \tag{2.7}$$

Puis la population est complétée par tirages au sort en associant à chaque individu a^t_j une probabilité $P_s(a^t_j)$:

$$P_s(a^t_j) = f_j/f_{moy} - \text{partie entière}[f_j/f_{moy}] \tag{2.8}$$

c. La sélection par tournoi stochastique

dans cette méthode de sélection qui a été proposée par Bridle, les probabilités de sélection sont calculées normalement et des paires successives d'individus sont tirées au sort grâce à la sélection par roue de loterie. Après avoir tirer une paire de chaînes, la chaîne ayant l'adaptation la plus élevée est déclarée vainqueur, elle est ajoutée à la nouvelle population, et une nouvelle paire est tirée. Ce processus continu jusqu'à ce que la population soit remplie [29]. D'autres méthodes de sélection sont présentées dans [40].

II.4.1.2 Mécanismes de changement d'échelle

Bien que la reproduction ne fait que copier les individus en fonction de leur fonction coût, il est fréquent d'avoir dans les premières générations des "supers-individus"qui vont tendre à dominer les procédés de sélection et d'accouplement, ils peuvent donc se reproduire entre eux et entraîneront une convergence prématuré.

Pour éviter ce problème et de préserver la diversité des individus, il est recommandé de réaliser un réajustement de la fonction coût avant d'effectuer la reproduction.

Les mécanismes de changement d'échelle les plus utilisés sont :

1. le changement d'échelle linéaire.
2. la troncature en sigma (σ).
3. le changement d'échelle en puissance.

a) Le changement d'échelle linéaire

C'est la technique de réajustement la plus répandue, elle consiste à transformer l'adaptation brute f en une adaptation transformée f' en utilisant une relation linéaire de la forme :

$$f' = af+b \tag{2.9}$$

Les coefficients a et b sont choisis de façon à ce que l'adaptation brute et transformée ont la même moyenne ($f'_{moy}=f_{moy}$) et à ce que la valeur maximale de l'adaptation transformée est un multiple (en général le double) de cette moyenne :

$$f'_{max}=\beta f_{moy} \quad , \quad 1 \leq \beta \leq 2 \tag{2.10}$$

De cette manière, on peut s'assurer que les individus de valeur moyenne sont copiés une fois en moyenne et les meilleurs sont copiés un nombre de fois égal au multiple choisi.

remarque : Dans ce mécanisme, il faut faire attention d'éviter des valeurs d'adaptation transformées négatives.

b) La troncature en sigma

Dans cette procédure, on utilise l'écart type de la population, ce qui permet d'éliminer les individus trop faibles.

La nouvelle fonction coût s'écrit sous la forme :

$$f'=f-(f_{moy}-c\sigma) \tag{2.11}$$

où

c :coefficient de l'écart type choisi arbitrairement $1 \leq c \leq 3$.

σ :l'écart type de l'adaptation calculé sur la population.

c) Le changement d'échelle en puissance

Dans ce cas l'adaptation transformée f' prend la valeur d'une certaine puissance de l'adaptation brute f :

$$f' = f^k \tag{2.12}$$

où la constante k peut être adaptée au cours des générations pour accentuer la sélection des individus situés à proximité des sommets.

II.4.2 Le croisement

Après avoir sélectionner les chaînes les mieux adaptées dans le processus de reproduction, elles vont subir maintenant à l'opération du croisement qui consiste à échanger des matériels génétiques entre deux chaînes reproductrices (parents) pour produire deux nouvelles chaînes (enfants).

Le croisement est un processus aléatoire de probabilité P_c appliquée à un couple de parents arbitrairement choisis dans la population.

un entier k représentant une position sur la chaîne est choisi aléatoirement entre 1 et la longueur de la chaîne moins 1 ($k \in [1\ L-1]$), cet entier représentera donc la position ou se produira le croisement, et enfin deux nouvelles chaînes sont crées en échangeant le matériel génétique (caractères) compris entre les positions k+1 et L inclues des chaînes reproductrices.

Ce processus est illustré dans la figure (Fig. 2.1) ou P1 et P2 sont les parents et O1 et O2 sont les enfants résultats de l'opération du croisement au point k=3.

P1 010|10011

P2 001|01001

O1 010 01001

O2 001 10011

Fig. 2.1 : Croisement à un site (k=3)

II.4.3 La mutation

La mutation est la modification aléatoire occasionnelle (de faible probabilité) d'un gène d'un individu. l'opérateur de mutation consiste à complémenter la valeur d'un bit du chromosome avec une probabilité P_m. Le processus est exécuté bit par bit (voir Fig.2.2).

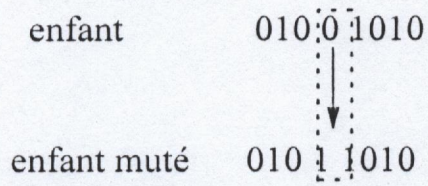


Fig. 2.2 : Principe de la mutation

Pour ne pas détériorer les performances de l'AG, Goldberg conseille une fréquence de mutation tous les 1000 bits [29]. Certains préconisent plutôt la valeur suivante pour P_m [40] ,

$$P_m^* = \frac{1}{L} \tag{2.13}$$

ou L est la longueur du chromosome.

D'autres études ont conduit à une formule empirique qui exprime le taux optimal de mutation en fonction de la longueur du chromosome L et de la taille de la population N :

$$P_m^* = \frac{1}{N\sqrt{L}} \tag{2.14}$$

Les effets de la reproduction, du croisement et de la mutation sur l'évolution d'une population donnée s'expliquent en utilisant la notion de schéma.

II.4.4 Notion de schéma

Un schéma est un motif de similarité décrivant un sous ensemble de chaînes avec des similarités à des positions définies. Un alphabet étendu est utilisé pour présenter cette notion {0,1,*} ou le symbole * pouvant prendre indifféremment la valeur 0 ou 1.

Par exemple, considérons les chaînes et les schémas de longueur 5, le schéma H=*101* décrit l'ensemble de quatre éléments :{01010,01011,11010,11011}.

Il y a deux caractéristiques propres au schéma :son ordre et sa longueur utile.

L'ordre d'un schéma H, noté o(H) est le nombre de positions instanciées (nombre de 1 et de 0 pour un alphabet binaire) dans le motif considéré.

Par exemple : o(*101*) = 3 .

La longueur utile d'un schéma H, notée δ(H) désigne la distance entre la première et la dernière position instanciée dans le motif considéré.

Par exemple : δ(*101*) = 4-2 = 2.

Les schémas sont des outils très intéressants qui permettent d'analyser l'effet de la reproduction et des autres opérateurs génétiques. On va considérer par la suite, les effets isolés puis combinés de la reproduction, du croisement et de la mutation sur les schémas contenus dans la population de chaînes.

II.4.4.1 Effet de la reproduction

L'importance du rôle de la sélection peut être mesurée par son effet sur la propagation d'un schéma H dans une population de N individus entre deux générations. on suppose que la sélection est de type proportionnelle et que le nombre d'exemplaires du schéma H est égal à $m(H,t)$ à la génération t. A la génération t+1, le nombre attendu de représentants de H est estimé comme suit [29] :

$$m(H,t+1) = m(H,t)\frac{f(H)}{f_{moy}} \qquad (2.15)$$

ou $f(H)$ et f_{moy} désignent respectivement l'adaptation moyenne des individus représentant le schéma H à la génération t et l'adaptation moyenne de la population.

L'équation (2.15) montre que les schémas ayant des coûts moyens supérieurs à celui de la population seront davantage copiés à la génération suivante, ceux qui ont un coût moyen inférieur à la moyenne le seront moins.

En supposant qu'un schéma quelconque H reste au dessus de la moyenne d'une quantité égale à $c.f_{moy}$ ou c est une constante ($f(H) = f_{moy} + c.f_{moy}$).

L'équation (2.15) qui traduit l'évolution des schémas peut s'écrire comme suit :

$$m(H,t+1) = (1+c).m(H,t) \qquad (2.16)$$

en supposant une valeur stationnaire de c, on obtient l'équation :

$$m(H,t+1) = (1+c)^{t}.m(H,0) \tag{2.17}$$

Ainsi, la reproduction alloue une place exponentiellement croissante (décroissante) aux schémas qui ont des performances au dessus (au dessous) de la moyenne.

II.4.4.2 Effet du croisement

La reproduction en elle même ne fait que copier les anciens individus et ne fait rien de promouvoir de nouveaux individus de l'espace de recherche. Le croisement permet d'introduire une nouvelle configuration en effectuant un échange d'information entre les chaînes.

L'effet du croisement peut être perçu à travers la propagation des schémas comme lors de la sélection des parents. Il convient alors de déterminer le taux de destruction (ou de survie) d'un schéma quelconque sous l'opérateur considéré.

Goldberg a indiqué qu'un schéma survit quand le point de croisement tombe à l'extérieur de sa longueur utile [29].

La probabilité de destruction d'un schéma H de longueur utile δ(H) lors du croisement simple est donnée par :

$$P_d(H) = \frac{\delta(H)}{L-1} \tag{2.18}$$

ou L désigne la longueur du chromosome.

Si le croisement a lieu avec une probabilité P_c, la probabilité de survie est donc :

$$P_s(H) \geq 1 - P_c.\frac{\delta(H)}{L-1} \tag{2.19}$$

En considérant que la reproduction et le croisement sont des opérations indépendantes, l'effet combiné de ces deux opérations s'obtient en multipliant le nombre attendu de schémas lors de la reproduction seule par la probabilité P_s de survie lors d'un croisement :

$$m(H,t+1) \geq m(H,t).\frac{f(H)}{f_{moy}}\left[1 - P_c\frac{\delta(H)}{L-1}\right] \tag{2.20}$$

Cette équation montre que les schémas de longueurs δ(H) faibles et de coût moyen *f*(H)

supérieur à la moyenne f_{moy} verront leur nombres augmenter à la génération t+1 par rapport à t.

II.4.4.3 Effet de la mutation

Comme il a été déjà dit, la mutation est une modification aléatoire d'une position quelconque de la chaîne moyennant une probabilité de mutation P_m.

Pour qu'un schéma H survive à l'opération de mutation, toutes ses positions instanciées doivent survivre. comme la probabilité de survie d'une position quelconque est $(1-P_m)$ et que le nombre de positions fixes dans le schéma est o(H), la probabilité de survie d'un schéma h à la mutation est :

$$P_s(H) = (1 - P_m)^{o(H)} \tag{2.21}$$

La probabilité de mutation étant petite ($P_m << 1$), l'expression précédente peut être approximée par :

$$P_s(H) = 1 - o(H).P_m \tag{2.22}$$

Nous pouvons donc conclure qu'on peut s'attendre à ce qu'un schéma H quelconque reçoive, du fait de la reproduction, du croisement et de la mutation, un nombre de copies donné par l'équation suivante :

$$m(H,t+1) \geq m(H,t).\frac{f(H)}{f_{moy}}\left[1 - P_c \frac{\delta(H)}{L-1}\right]\left[1 - o(H).P_m\right] \tag{2.23}$$

Cette équation nous permet d'en conclure que les schémas courts, d'ordre faible et de performance au-dessus de la moyenne font l'objet d'un nombre de tests exponentiellement croissants dans les générations suivantes, cette conclusion est appelée le théorème des schémas ou le théorème fondamental des AG.

II.5 La sélection des individus d'une nouvelle génération

A la suite de la recombinaison génétique (croisement et mutation), la population compte 2N individus (N parents et N enfants) ;N étant la taille de la population. il faut donc éliminer N individus pour constituer la génération suivante. C'est le rôle de la sélection finale

S_f qui agit sur les populations de parents ($P^{'t}$) et d'enfants ($P^{''t}$) d'une génération pour créer la nouvelle génération (P^{t+1}).

$$P^{t+1} = S_f(P^{'t}, P^{''t}) \tag{2.24}$$

Pour effectuer cette sélection entre parents et enfants, plusieurs stratégies sont possibles.

II.5.1 La sélection par descendance

Il n'y a aucune compétition entre parents et enfants. La population de la nouvelle génération est obtenue par descendance, les enfants remplaçant automatiquement leurs parents quel que soit leur adaptation.

$$P^{t+1} = S_f(P^{'t}, P^{''t}) = P^{''t} \tag{2.25}$$

L'inconvénient de ce mode de sélection est que l'on risque de voir disparaître les caractéristiques génétiques des parents les mieux adaptés si elles n'ont pas été totalement transmises lors de la recombinaison génétique.

II.5.2 La sélection par compétition

Une compétition a lieu entre parents et enfants pour déterminer les "survivants" de la génération. Ainsi, les enfants peuvent être insérés dans la population si et seulement si leur performance est supérieure à celle de leurs parents à rang équivalent.

$$a_i^{t+1} = \begin{cases} a_i^{''t} & si \quad f(a_i^{''t}) > f(a_i^{'t}) \\ a_i^{'t} & \text{sinon} \end{cases} \tag{2.26}$$

Avec a_i^{t+1} est le i^{eme} individu de la génération (t+1).

II.5.3 Steady state selection

Cette technique consiste à effectuer une compétition après chaque recombinaison génétique entre parents et enfants en retenant les deux meilleurs individus parmi les quatre.

II.5.4 Selective breeding selection

Dans cette méthode, on garde les N meilleurs individus parmi la population intermédiaire de parents et d'enfants pour former la nouvelle génération. Ceci garantie que les meilleurs individus de la population (qu'ils soient enfants ou parents) sont nécessairement conservés pour la génération suivante.

II.6 Description du codage

Chaque chaîne de la population est codée sur un nombre fini de bit. La longueur de la chaîne est fixée par l'utilisateur selon :

- Le domaine de variation de la chaîne.
- La précision demandée.

Pour l'optimisation d'une fonction à plusieurs variables $f(x_1,x_2,...,x_n)$, on utilise un codage binaire concaténé qui consiste à :

- Coder chaque variable selon le choix de la longueur.
- Construire les chaînes en concaténant les différents codes.

Soit chaîne = (code1)(code2)...(coden)

avec codei :le code du i^{eme} paramètre dans l'espace de recherche de dimension n.

II.7 Description du décodage

Chaque chaîne doit être décodée pour pouvoir calculer la valeur de la fonction coût qui lui est associée. parmi les types de décodage possibles, le décodage binaire est souvent le plus utilisé.

Dans le cas ou chaque paramètre x_i est situé dans un intervalle $[x_{imin}\ x_{imax}]$, et à qui est associé une chaîne binaire $b_0b_1...b_{Lxi-1}$ définie sur Lxi bits. A cette chaîne correspond une valeur entière naturelle,

$$N(x_i) = \sum_{i=0}^{Lxi-1} 2^{Lxi-i-1}.b_i \qquad (2.27)$$

Le paramètre réel x_i de l'espace de recherche relatif à $N(x_i)$ est obtenu par interpolation linéaire :

$$x_i = x_{i\min} + \frac{x_{i\max} - x_{i\min}}{2^{Lxi} - 1}.N(x_i) \qquad (2.28)$$

la précision des paramètres par ce type de décodage est :

$$\varepsilon_i = \frac{x_{i\max} - x_{i\min}}{2^{Lxi} - 1} \qquad (2.29)$$

II.8 Organigramme de l'AG

Un AG procède selon l'organigramme suivant :

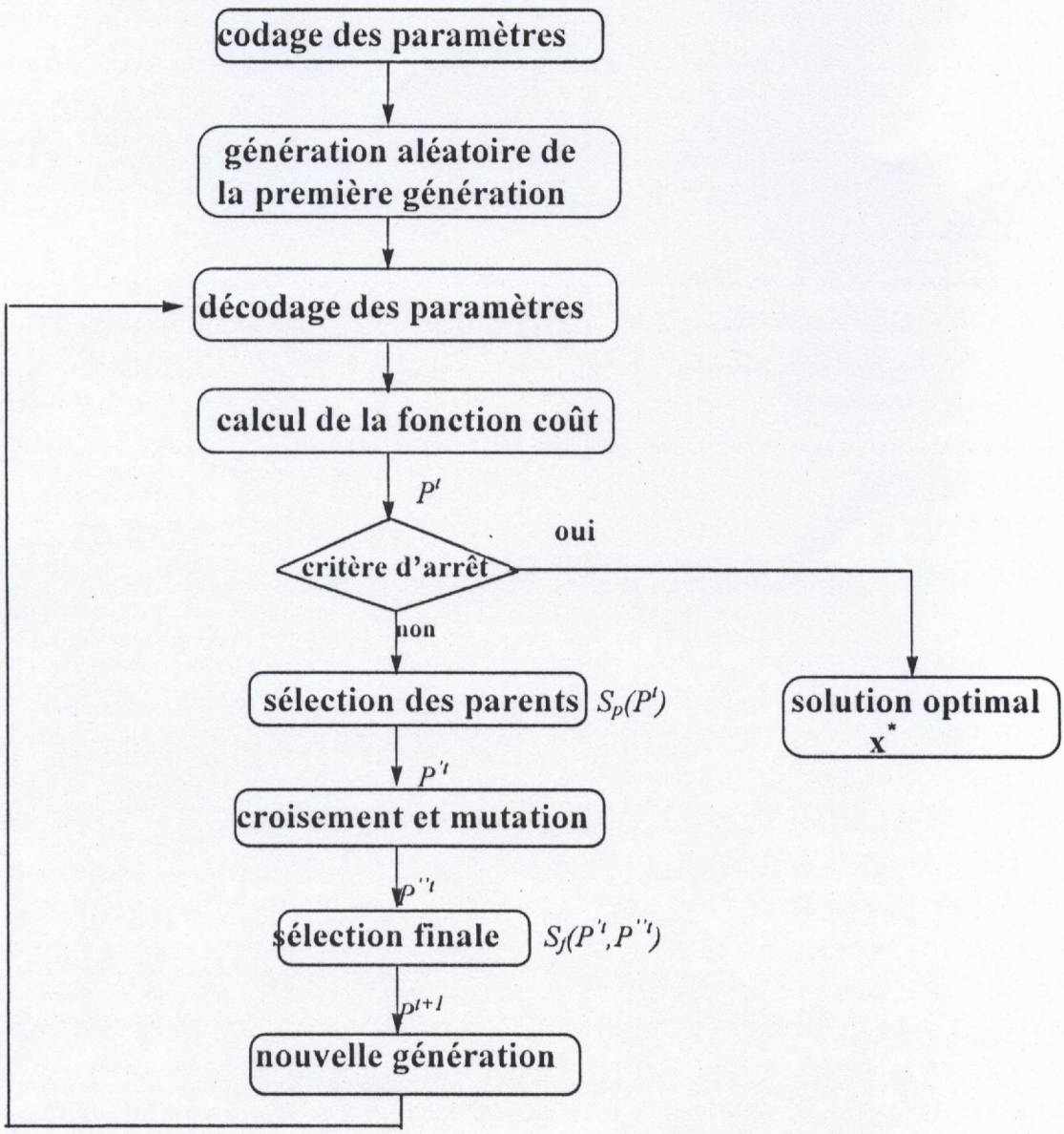


Fig. 2.3 : Organigramme de l'AG

II.9 La convergence des AG

II.9.1 Critères de convergence

Le problème de convergence est généralement éludé en imposant un nombre maximal de générations t_{max} et en arrêtant la recherche lorsque $t = t_{max}$. On estime alors que l'algorithme a convergé et que l'individu de plus forte performance dans la population P^{tmax} correspond à la solution recherchée.

Une méthode plus rigoureuse consiste à supposer que l'algorithme converge vers l'optimum lorsque l'adaptation d'une partie (ou de l'ensemble) de la population se rapproche de celle du meilleur individu. On peut considérer que cet événement se produit à la génération t pour laquelle :

$$1 - \frac{f_{moy}}{f_{max}} \leq \varepsilon \qquad (2.30)$$

ou : ε est la précision requise sur la convergence.

f_{max} est la performance du meilleur individu de la population à la génération t.

f_{moy} est la moyenne de l'adaptation calculée sur l'ensemble de la population ou sur une partie correspondant à un pourcentage des représentants les plus performants.

Nous pouvons aussi supposer que l'algorithme a convergé lorsque le meilleur individu de la population n'évolue plus.

Ces critères ne sont pourtant absolument pas fiables dans mesure ou l'algorithme peut converger et se stabiliser autour d'une solution dans l'attente d'une mutation qui le dirigera vers une autre région plus intéressante. Le choix d'un critère d'arrêt idéal reste donc sans réponse.

II-9-2 Taux de convergence

Du fait des problèmes de convergence et du caractère stochastique de l'exploration génétique, il est habituel d'exécuter plusieurs fois le même algorithme sur le même problème. Un taux de convergence (ou taux de réussite) rend compte de son efficacité. On peut par exemple définir le taux de convergence comme le rapport du nombre de fois ou l'algorithme a convergé vers la solution optimale sur le nombre total d'exécution.

II-10 Conclusion

Les AG sont des outils d'optimisation performants qui permettent de réaliser une exploration globale de l'espace. Contrairement aux méthodes déterministes classiques, ils ne nécessitent aucun calcul de dérivées et peuvent être appliquées aussi bien à des fonctions continues d'une seule variable, qu'à des fonctions discontinues dépendant d'un grand nombre de paramètres.

Les AG sont capables d'obtenir des solutions quasi-optimales pour différents types de problèmes sans connaissance explicite du domaine de travail en manipulant simplement des chaînes de bits et en utilisant des opérateurs simples qui ne mettent en jeu que des procédures aussi peu complexes que la génération de nombres aléatoires, la copie de chaînes et les échanges de morceaux de chaînes.

Les approches de conception d'un FLC

III.1 La logique floue .

III.1.1 Introduction .

III.1.2 Contrôleur à logique floue .

III.1.2.1 La fuzzification .

III.1.2.2 Bloc de contrôle .

III.1.2.3 La défuzzification .

III.1.2.4 Paramètres de conception d'un FLC .

III.2 Les approches de conception d'un FLC .

III.2.1 L'approche connexioniste .

III.2.1.1 Les réseaux de neurones .

a) Définition .

b) Architecture des réseaux de neurones .

c) Entraînement des réseaux de neurones .

d) L'algorithme de la rétropropagation .

III.2.1.2 Les réseaux de neurones flous .

a) Définition .

b) Les différentes approches des FNN .

c) Les architectures des FNN .

III.2.2 L'approche directe .

III.2.2.1 L'application des AG pour les contrôleurs flous .

a) Définition du problème : paramètres à optimiser

b) codage de l'AG .

III.2.2.2 L'apprentissage des systèmes flous par les AG

III.3 Conclusion .

III.1 La logique floue

III.1.1 Introduction

Les bases théoriques de logique floue ont été établies en 1965 par le professeur Lotfi A. Zadeh. Le développement de cette théorie vient de l'incapacité de décrire certains phénomènes physiques avec des modèles mathématiques exacts .

La logique floue possède certaines caractéristiques qui la distingue des techniques conventionnelles tel que la manipulation des variables linguistiques au lieu des variables numériques et les relations entre ces variables sont exprimées par des citations conditionnelles floues. C'est Mamdani qui a été le premier à appliquer cette nouvelle théorie à la commande des systèmes en 1974, par la suite une variété d'applications a été faites [2] .

III.1.2 Contrôleur à logique floue

Le contrôleur à logique floue (FLC ; fuzzy logic controller) est décrit par un ensemble de règles floues du type SI {conditions} ALORS {actions}, le FLC fournit donc un algorithme capable de convertir la stratégie de contrôle linguistique basée sur la connaissance experte en une stratégie de contrôle automatique .

La structure d'un FLC classique est montrée dans la figure (3.1) ,il est composé de trois parties :

- bloc de fuzzification.
- bloc de contrôle (base de règles floues et procédure d'inférence).
- bloc de defuzzification .

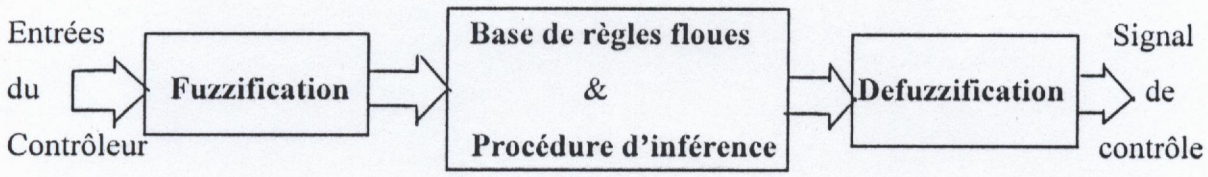


Fig. 3.1 : Structure d'un contrôleur à logique floue

III.1.2.1 La fuzzification

La fuzzification consiste à convertir les valeurs numériques des variables d'entrée en variables linguistiques (floues), pour cette fin on utilise des fonctions d'appartenance qui servent à subdiviser l'espace d'entrée, univers de discours, en ensembles floues.

Les formes les plus utilisées de ces fonctions sont la forme triangulaire ou trapézoïdal selon la relation (3.1) et la forme gaussienne selon la relation (3.2) (Fig. 3.2) .

$$\mu_A(x) = \begin{cases} \dfrac{x - L_A}{c_A - L_A} & si \quad L_A < x \leq c_A \\ \dfrac{x - r_A}{c_A - r_A} & si \quad c_A < x < r_A \\ 0 & ailleurs \end{cases} \tag{3.1}$$

$$\mu(x) = \exp\left(-\frac{(x-a)^2}{b}\right) \quad , \qquad -\infty < x < +\infty \tag{3.2}$$

Dans la plus part des applications, l'univers de discours est partitionné avec entre trois et neuf ensembles flous, la décomposition triangulaire de l'univers de discours comme indiquée dans la figure (3.3) est appelée système TP (Triangular Partition System) ,celle de la figure (3.4) est appelée système TPE (Triangular Partition with Evenly spaced midpoints System), dans cette décomposition on utilise une séquence de triangles isocèles avec des bases de la même largeur. En général, il est désirable d'avoir un certain chevauchement des ensembles flous afin d'obtenir des transitions lisses de la surface de contrôle.

III.1.2.2 Bloc de contrôle

Ce bloc est composé de deux parties :

- La base de règles floues : elle est formée d'un ensemble de règles de contrôles floues représentées par ensemble de relations linguistiques liant les variables d'entrée (les prémises) et les variables de sortie (conséquences) ,ces règles sont de la forme :

SI {l'ensemble des conditions sont satisfaites }

ALORS {un ensemble de conséquences peuvent être inférées }

- La procédure d'inférence ;elle consiste à dériver les actions de contrôle flou utilisant l'implication floue et les règles d'inférence déjà mentionnées .

Ces règles constituent un moteur d'inférence du contrôleur dans lequel on distingue deux types d'opérateurs ,l'opérateur de conjonction ET qui lie les différentes variables de la règle et l'opérateur de disjonction OU qui lie l'ensemble des règles formant la base de connaissance .

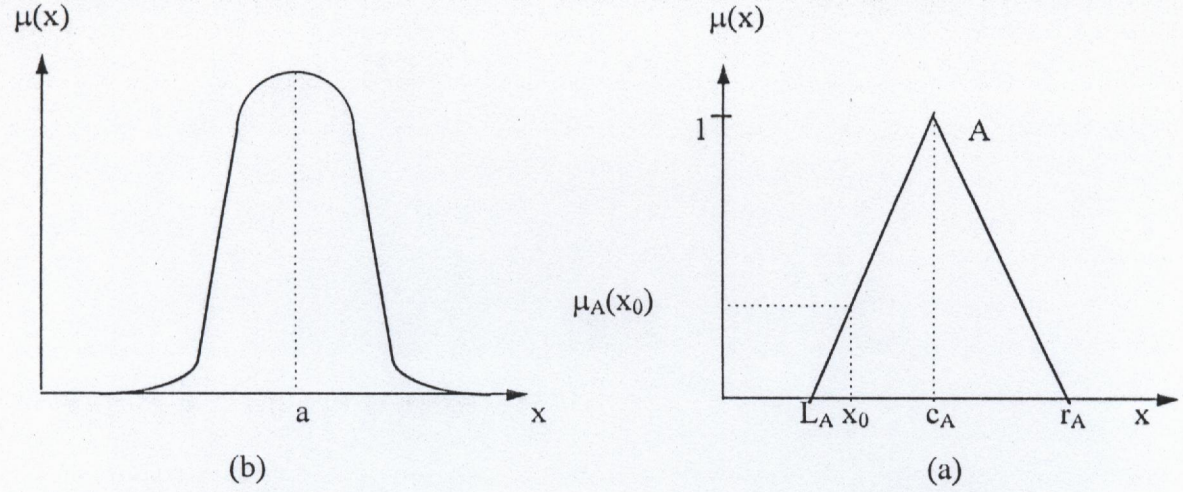


Fig. 3.2 : Formes des fonctions d'appartenance

(a) triangulaire
(b) gaussienne

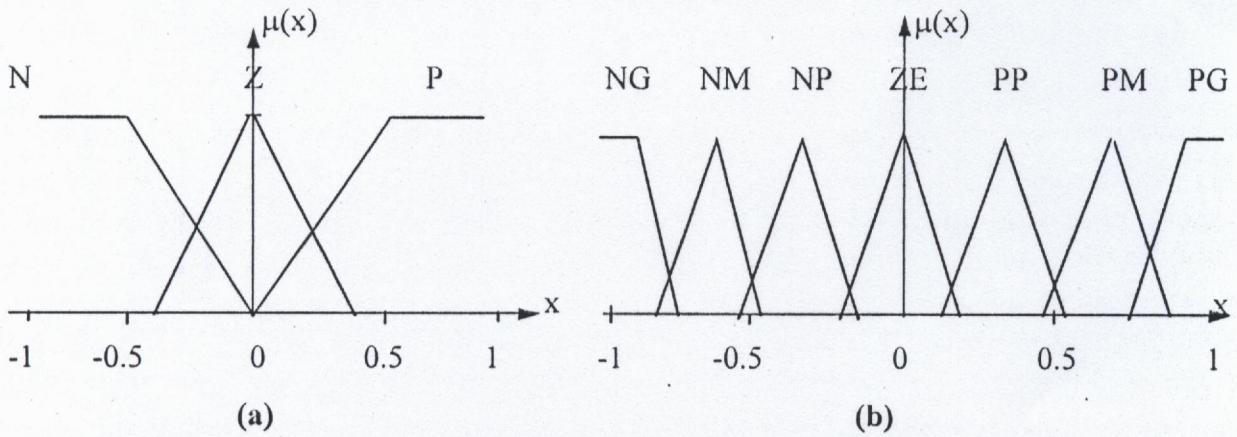


Fig. 3.3 : Univers de discours (TP) avec : (a) trois ensembles flous
(b) sept ensembles flous

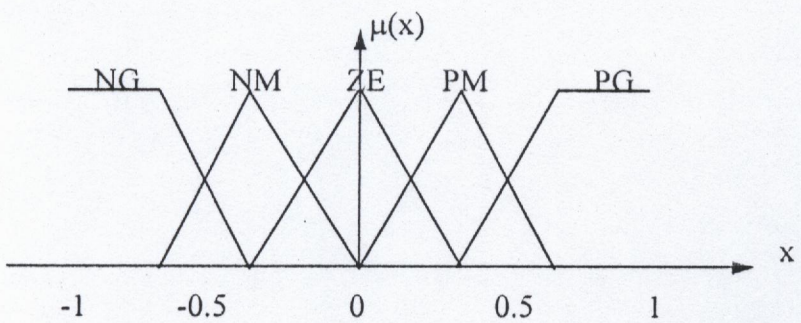


Fig. 3.4 : Univers de discours (TPE) avec 5 ensembles flous

Ces deux opérateurs peuvent être réalisés par différentes manières ,en effet l'opérateur ET peut être réalisé par la norme-t tel que le produit algébrique ou l'intersection floue (Min) et l'opérateur OU par la conorme-t tel que la somme algébrique ou l'union floue (Max) [2]. De ce fait plusieurs mécanismes d'inférence se présentent :

- Mécanisme d'inférence Max - Min .
- Mécanisme d'inférence Max - Prod .
- Mécanisme d'inférence Som - Prod .

III.1.2.3 La défuzzification

La défuzzification consiste à convertir l'action de contrôle floue inférée en une grandeur numérique. La stratégie de défuzzification la plus utilisée est celle du centre de gravité . Considérons un ensemble de règles floues de la forme :

R^i : SI x1 est A_1^i **ET x2 est** A_2^i **ALORS** y est B^i (3.3)

la sortie numérique y est la moyenne pondérée des sorties de toutes les règles :

$$y = \frac{\sum_{i=1}^{m} y_i . w_i}{\sum_{i=1}^{m} w_i} \tag{3.4}$$

avec m représente le nombre de règles floues .

y_i est le centre de la fonction d'appartenance caractérisant la valeur linguistique B_i.

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

$$w_i = \mu_{A_1^i}(x1).\mu_{A_2^i}(x2) \tag{3.5}$$

III.1.2.4 Paramètres de conception d'un FLC :

Ainsi la conception d'un contrôleur a logique floue implique le choix des paramètres suivants :

- la largeur de l'univers de discours .

-Le nombre de sous ensemble flous (variables linguistiques), la partition de l'univers de discours .

- Le type de fonctions d'appartenance : triangulaire ou gaussienne .
- Les règles d'inférences .
- Le choix de la stratégie de défuzzification .

Historiquement ce choix était empirique et était basé sur l'expérience d'opérateurs experts du procédé à commander. Cependant depuis quelques années, des méthodes systématiques de conception de contrôleur à logique floue ont été proposées. Dans ce qui suit nous passons en revue les principales approches systématiques de conception des FLC.

III.2 Les approches de conception d'un FLC

Bien que les contrôleurs à logique floue ont été appliqués avec succès sur plusieurs procédés industriels complexes, leur conception reste cependant une tache très difficile.
L'approche traditionnelle pour la conception floue qui est basée sur les connaissances acquises par des opérateurs experts est laborieuse, consomme beaucoup de temps et dans la plus part des cas spécifique pour chaque application ,en plus cette approche présente d'autres difficultés tel que :

- Les opérateurs ne peuvent pas facilement transformer leurs connaissances et expériences en une forme algorithmique ou base de règle nécessaire pour la conversion en une stratégie de contrôle automatique .
- Le domaine d'expertise n'est pas toujours disponible .
- Disponibilité des techniques d'acquisition des connaissances .

Suite à ces difficultés, des recherches intensives ont été effectuées dans le but de construire des méthodes systématiques et optimales pour la conception des FLC, ces recherches ont conduit au développement de deux nouvelles approches :

1. Approche connexioniste : elle consiste à combiner les réseaux de neurones artificiels (ANN; Artificial Neural Networks) et les systèmes flous pour construire ce qu'on appelle les réseaux de neurones flous (FNN; Fuzzy Neural Networks) [25] .
2. Approche directe : elle consiste à appliquer un algorithme d'optimisation pour la conception d'un système flou. Du à la complexité de l'espace de recherche, l'algorithme génétique est utilisé dans la plupart des cas.

III.2.1 L'approche connexioniste

III.2.1.1 Les réseaux de neurones

a) Définition

Un réseau de neurones traite l'information qu'il reçoit d'une manière analogue aux neurones du cerveau. Il est constitué de plusieurs éléments processeurs dits neurones

artificiels reliés les uns aux autres par un réseau complexe. Chaque neurone effectue une somme pondérée des signaux d'entrée modulés par une fonction dite d'activation (une fonction non linéaire) et génère une sortie qui sera appliquée aux autres neurones via des connexions pondérées. Avec cette simple structure et en choisissant un nombre approprié de neurones ,ces réseaux sont capables d'approximer n'importe quelle fonction continue avec une certaine précision.

b) Architecture des réseaux de neurones

Les réseaux de neurones peuvent être classés en deux principales catégories suivant la structure des connexions :les réseaux récurrents (dynamiques) et les réseaux non récurrents (statiques) .

Dans les réseaux récurrents, plusieurs neurones sont interconnectés pour organiser le réseau. Le réseau de Hopfield qui est indiqué dans (Fig. 3.5) appartient à ce type de structure.

Les réseaux non récurrents ont une structure hiérarchique qui consiste en plusieurs couches; une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Dans ce type de réseau (Fig. 3.6), il n'y a pas d'interconnexions entre les neurones de la même couche, et les signaux circulent de la couche d'entrée à la couche de sortie dans une seule direction .

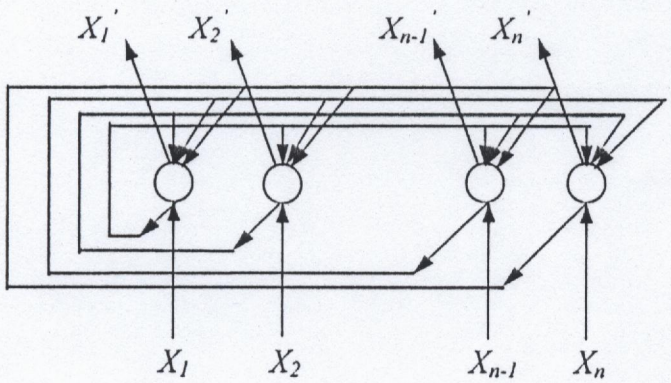


Fig. 3.5 : Réseau de Hopfield

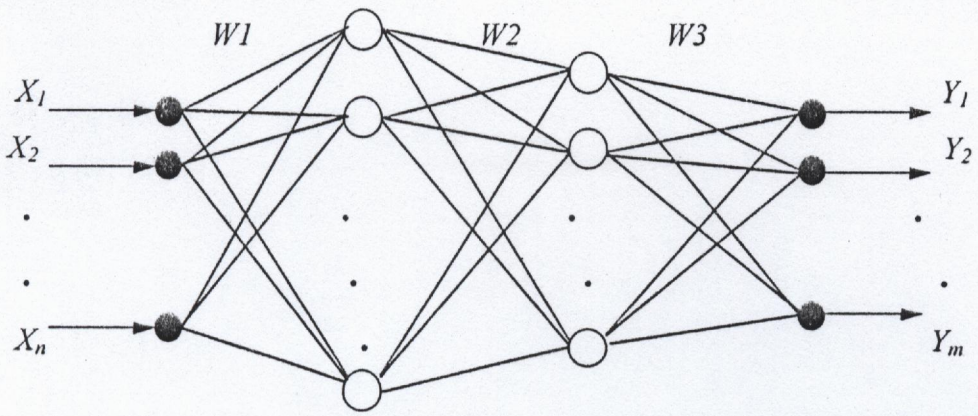


Fig. 3.6 : Structure d'un réseau statique

c) Entraînement des réseaux de neurones

L'application des réseaux de neurones comprend typiquement deux phases : une phase d'apprentissage et une phase d'opération .

L'apprentissage est un processus à travers lequel les paramètres (les poids des connexions) du réseau sont ajustés pour refléter l'information contenue dans la structure du réseau .

Une fois le réseau est entraîné, il représente une base de connaissance statique qui peut être appelée durant la phase d'opération .

Il existe trois types d'apprentissage: apprentissage supervisé ,apprentissage non supervisé et apprentissage par renforcement [39] .

Dans l'apprentissage supervisé, le réseau est alimenté par les valeurs d'entrée et les valeurs de sortie désirées, le réseau ajuste alors ses poids en se basant sur l'erreur de la sortie calculée .

Dans l'apprentissage non supervisé, le réseau est seulement alimenté par les valeurs d'entrée et le réseau ajuste les poids en se basant uniquement sur les valeurs d'entrée et la sortie courante du réseau.

Dans l'apprentissage par renforcement, le réseau reçoit des évaluations scalaires implicites des entrées précédentes.

d) L'algorithme de la retropropagation

La rétropropagation est la méthode la plus utilisée pour l'entraînement des réseaux de neurones, c'est un algorithme d'apprentissage supervisé qui consiste à ajuster les poids des connexions du réseau en utilisant simplement des données d'entrée/sortie. Cet algorithme

utilise la technique de recherche du gradient pour minimiser une fonction coût J qui est égale à la moyenne de l'erreur quadratique entre la sortie désirée et la sortie actuelle du réseau.
L'algorithme de la retropropagation est donné par :

$$W(k+1) = W(k) - \eta.\frac{dJ}{dW(k)} \tag{3.6}$$

avec : W est le vecteur des poids

η est le taux d'apprentissage

J est la fonction coût définie par :

$$J = \frac{1}{2}\sum_{j}\left(d_j - y_j\right)^2 \tag{3.7}$$

où d_j est la sortie désirée

y_j est la sortie du réseau

Bien que cet algorithme soit le plus utilisé, il présente deux problèmes :

- convergence lente .
- l'estimation des paramètres risque de tomber dans un minimum local du critère d'optimisation.

III.2.1.2 Les réseaux de neurones flous

a) Définition

Les systèmes neuronaux flous combinent la théorie des réseaux de neurones artificiels (ANN) et les systèmes flous. La méthode d'apprentissage utilisée pour l'entraînement des réseaux de neurones permet à ces systèmes d'apprendre à partir de la présentation d'un ensemble de données (entrées/sorties) d'entraînement. Cet apprentissage consiste à ajuster les poids des connexions reliant les différentes couches et permet un ajustement très précis des paramètres des fonctions d'appartenance des ensembles flous incorporés dans le réseau neuronal. D'autre part, la théorie des ensembles flous permet aux systèmes neuronaux flous de présenter l'information apprise sous une forme plus compréhensible à l'être humain.

b) Les différentes approches des FNN

Plusieurs méthodes de combiner les systèmes flous et les réseaux de neurones sont reportées dans la littérature [22,23,26,27,36,41,42].

Différentes stratégies d'apprentissage ont été utilisées dans ces applications tel que apprentissage supervisé [36] et apprentissage par renforcement [16,43].

Ces méthodes peuvent être classé suivant la relation recherchée entre le système d'inférence flou et le réseau neuronal en deux approches distinctives :

- Approche fonctionnelle : dans cette approche le système neuronal-flou est considéré comme étant un réseau neuronal ordinaire qui est désigné pour approximer l'algorithme de contrôle flou [44].
- Approche structurelle : elle consiste à réaliser le processus de raisonnement et d'inférence flou à travers la structure d'un réseau connexioniste [45].

Récemment, une nouvelle approche a été développée et consiste à construire des réseaux connexionistes contenant des neurones particuliers dits neurones-flous capables d'implémenter les opérations de base rencontrées dans la théorie des ensembles flous tel que ET et OU [25,46].

c) Les architectures des FNN

- **Les FNN réguliers**

Dans cette classe, les neurones réalisent simplement des opérations arithmétiques tel que addition et multiplication. L'approche la plus directe consiste à implémenter le système d'inférence dans un réseau neuronal multicouche [41]. la figure (3.7) montre la configuration d'un FNN qui implémente un système flou décrit par des règles d'inférence du type :

R^i : **SI** x_1 *est* A_1^i **ET** ... **ET** x_n *est* A_n^i **ALORS** y_1 *est* w_1^i **ET** ... **ET** y_m *est* w_m^i

avec :

A_p^i : les variables linguistiques d'entée.

W_q^i : nobres réels de sortie.

Le réseau possède en total quatre couches ;une couche d'entrée, la deuxième couche contient des noeuds qui représentent les fonctions d'appartenance ($\mu_{A_p^i}$) associées aux variables linguistiques (A_p^i), dans la troisième couche, chaque noeud i calcule la valeur de vérité (μ_i) de la $i^{éme}$ règle par :

$$\mu_i = \prod_{p=1}^{n} \mu_{A_p^i}(x_p) \qquad (3.8)$$

Ce noeud représente une règle floue, et enfin une couche de sortie. Les connexions entre la troisième et quatrième couche sont pondérées avec les valeurs w_q^i.

Les sorties du réseau sont données par :

$$y_q = \frac{\sum_i \mu_i . w_q^i}{\sum_i \mu_i} \quad , \qquad q=1 \ldots m \tag{3.9}$$

L'apprentissage du réseau permet d'ajuster les paramètres des fonctions d'appartenance ainsi que les poids (w_q^i). L'algorithme d'apprentissage utilisé est celui de la rétropropagation.
D'autres approches considèrent les FNN comme étant des réseaux de neurones ordinaires avec des signaux d'entrée flous (des ensembles flous) et des poids flous [25]. Leur architecture est similaire à celle de la figure (3.8) où les entrées $\overline{X_1}, \overline{X_2}$, les poids w_{ik}, v_k et la sortie $\overline{Y}$ sont tous flous.
l'entrée du neurone caché j est donnée par :

$$\overline{I_j} = \overline{X_1}.\overline{W_{1j}} + \overline{X_2}.\overline{W_{2j}} \quad , \qquad 1 \le j \le k \tag{3.10}$$

La barre indique qu'il s'agit de nombres flous.

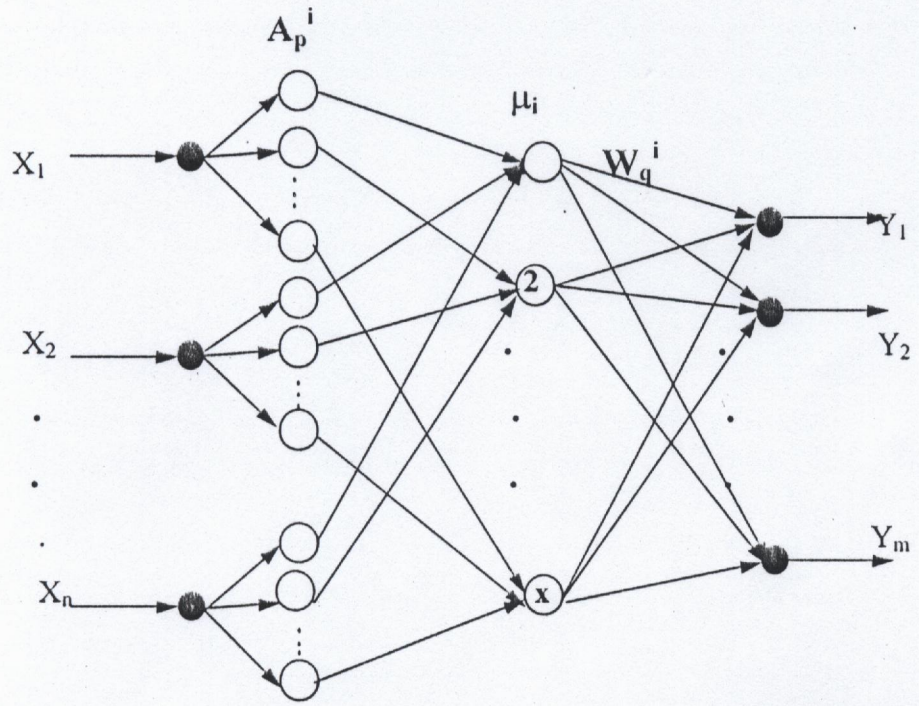


Fig. 3.7 : Configuration d'un FNN régulier

La sortie du j^{ème} neurone caché est donnée par :

$$\overline{Z_j} = f(\overline{I_j}) \quad , \qquad 1 \leq j \leq k \tag{3.11}$$

où f est la fonction d'activation qui est en général une fonction sigmoïdale.

L'entrée du neurone de sortie est alors :

$$\overline{I_0} = \overline{Z_1}.\overline{V_1} + ... + \overline{Z_k}.\overline{V_k} \tag{3.12}$$

et la sortie finale du réseau sera :

$$\overline{Y} = f(\overline{I_0}) \tag{3.13}$$

Une variante de cette approche a été introduite par Tanaka et ses collègues [47], ils ont proposé une architecture d'un réseau neuronal flou avec des poids et des biais flous avec des fonctions d'appartenance triangulaires (ou trapézoïdales). Les entrées et sorties du réseau sont fuzzifiées utilisant des nombres flous représentés par 'α-level sets'.

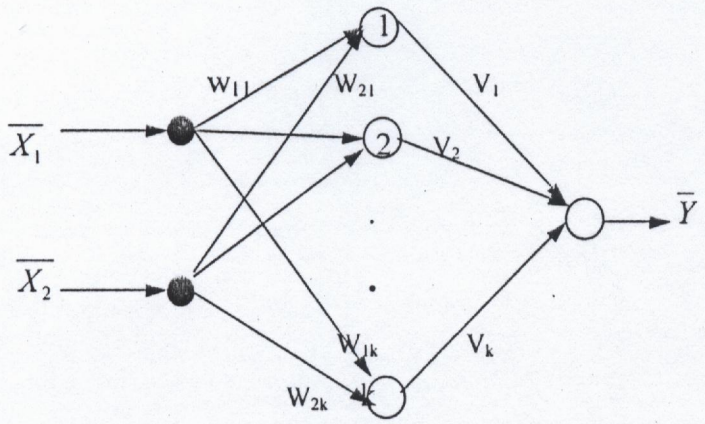


Fig. 3.8 : Configuration d'un FNN avec des entrées, poids et sorties flous

Pour l'entraînement du réseau, ils ont développé l'algorithme de la rétropropagation standard basé sur 'α-cut'(α-cut based backpropagation) [25] pour qu'il puisse s'adapter aux poids et biais flous, mais cette méthode n'est pas applicable si d'autres formes des fonctions d'appartenance autre que triangulaires (ou trapézoïdales) sont utilisées.

- **Les FNN hybrides**

Cette approche est basée sur l'utilisation des neurones-flous capables d'accomplir les opérations floues fondamentales tel que ET et OU. Les entrées du neurone-flou sont des ensembles flous et sont connectés à travers des poids réels ou flous. L'opérateur de conjonction (ET) et de disjonction (OU) sont réalisés utilisant la norme-t, la conorme-t ou la norme-s. Le réseau neuronal-flou est organisé en cinq couches : une couche d'entrée, une couche de fuzzification, une couche de neurones ET, une couche de neurones OU et une couche de defuzzification.

Puisqu'il y a plusieurs possibilités de réaliser les opérateurs logiques ET et OU, différentes structures neuronales-floues ont été développées, on va présenter par la suite deux structures :

Dans la première structure, on va considérer l'implémentation des opérateurs ET et OU par les normes (s-t) [46], dans la seconde ces opérateurs sont réalisés par les opérations Min et Max respectivement [36].

■ **Première structure**

Dans cette structure dite FLP (Fuzzy Logic Processor), l'implémentation structurelle d'un système d'inférence flou dans un réseau neuronal multicouche est basée sur les présentations par Pedrycz [46].

Les opérateurs ET et OU utilisés par ce réseau sont réalisés par les normes t-s comme suit :

La fonction OU : $\hat{u} = OU(X,W)$

est définie en utilisant les normes t et s :

$$\hat{u} = \underset{i=1}{\overset{I}{S}}\left[x_i t w_i\right] \tag{3.14}$$

où $X=[x_i]$, $W=[w_i]$;i=1,2,..,I sont la fonction d'entrée et les vecteurs des poids réels respectivement, et $\hat{u}$ est la fonction scalaire de sortie (Fig. 3.9).

En choisissant la norme-t comme le produit (atb = a.b) et la norme-s comme la somme algébrique (asb = a+b-a.b), alors la fonction OU sera donnée par :

$$\hat{u} = 1 - \prod_{i=1}^{I}\left(1 - x_i.w_i\right) \tag{3.15}$$

La fonction ET : $\hat{u} = ET(X,W)$

est définie de la même façon en utilisant les normes t-s mais dans l'ordre inverse :

$$\hat{u} = \overset{I}{\underset{i=1}{T}}\left[x_i sw_i\right] \tag{3.16}$$

Les normes t et s étant le produit et la somme algébrique respectivement, la fonction ET peut s'écrire comme :

$$\hat{u} = \prod_{i=1}^{I}\left(x_i + w_i - x_i w_i\right) \tag{3.17}$$

Un processeur logique peut être construit en utilisant ces deux neurones comme le montre la figure (3.9), où les $\hat{Z}_i^{L-1}$ sont les sorties des neurones de la couche (L-1) et l'élément non linéaire peut être une fonction sigmoïdale f modifiée par les paramètres m et b :

$$f(\hat{u}) = \frac{1}{1+e^{-(\hat{u}-m)b}} \tag{3.18}$$

et $\hat{Z}_n^L$ représente la sortie du neurone n de la couche L

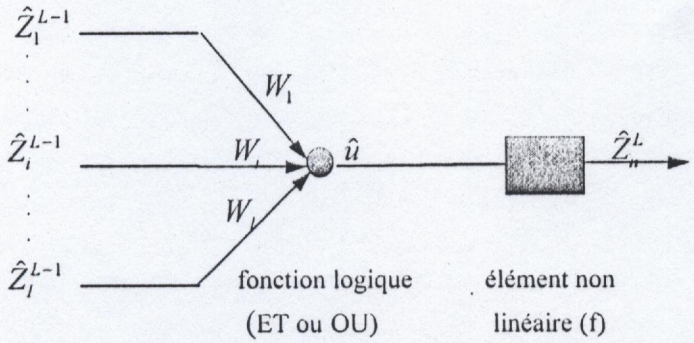


Fig. 3.9 : Processeur logique

Un réseau est formé donc de processeurs logiques interconnectés comme l'indique la figure (3.10), où la structure est composée d'une couche d'entrée (signaux d'entrée réels), d'une couche de fuzzification qui utilise des fonctions d'appartenance de forme triangulaire, d'une couche cachée de neurones ET suivit par une couche cachée de neurones OU et une couche de defuzzification qui sert à convertir les sorties en valeurs déterminées utilisant la méthode du centre de gravité. Le signal d'erreur utilisé pour l'entraînement du réseau est obtenu à partir de la différence entre la sortie désirée et la sortie du réseau. La méthode de la rétropropagation est utilisée pour ajuster les poids des neurones ET et OU, il n y a pas d'ajustement des ensembles flous d'E/S. Une fois le réseau converge, on peut extraire les règles d'inférence [27].

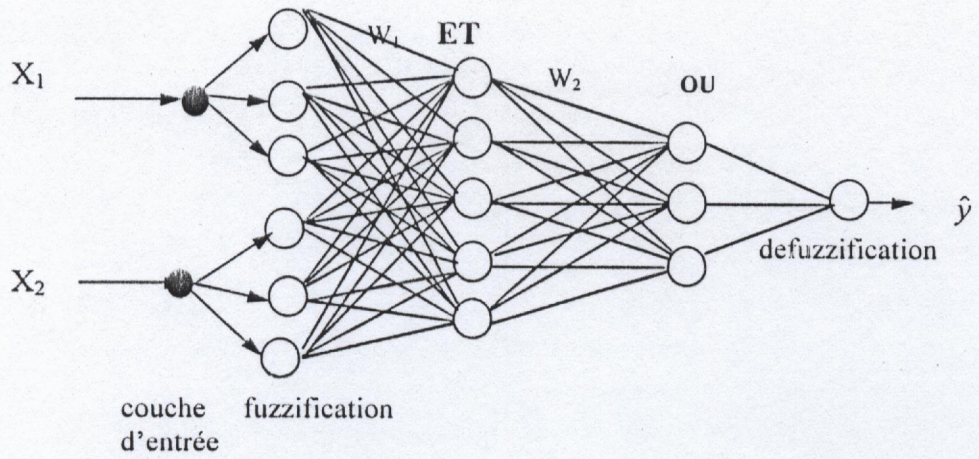


Fig. 3.10 : Architecture d'un FLP avec

- **2 entrées/1 sortie**
- **3 ensembles flous pour les E/S**
- **5 neurones ET et 3 neurones OU**

■ **deuxième structure**

Dans cette structure, un réseau connexioniste à cinq couches est considéré pour la réalisation du système d'inférence flou, ce réseau contient des neurones implémentant les opérations Min (ET) et Max (OU).

Hayashi et al.[48] ont utilisé un FNN avec des entrées/sorties réelles, des poids flous et des fonctions d'appartenance triangulaires pour modéliser un contrôleur flou spécifié par la table des règles donnée par le tableau (3.1) Les A_i, B_i et C_i étant les ensembles flous triangulaires associés à l'erreur (e), la variation de l'erreur (Δe) et l'action y du contrôleur respectivement.

Ce contrôleur flou a comme entrées (e) et (Δe), étant données les valeurs de ces dernières, les neuf règles sont évaluées comme suit :

$\Delta 1 = \text{Min}\ (B_1(e),\ A_2(\Delta e))$

$\Delta 2 = \text{Min}\ (B_2(e),\ A_2(\Delta e))$

.

.

$\Delta 9 = \text{Min}\ (B_5(e),\ A_4(\Delta e))$

e \ Δe	A_1	A_2	A_3	A_4	A_5
B_1		(1) C_1			
B_2		(2) C_1		(3) C_2	
B_3	(4) C_2		(5) C_3		(6) C_4
B_4		(7) C_4		(8) C_5	
B_5				(9) C_5	

Tab.3.1 : Règles de contrôle floues d'un procédé industriel (9règles floues)

Et puisqu'on a neuf règles et seulement cinq actions de contrôle, on maximise la Δi correspondante à la même action C_k comme suit :

ε_1=Max (Δ1, Δ2), ε_2=Max (Δ3, Δ4), ε_3= Δ5, ε_4=Max (Δ6, Δ7), ε_5=Max (Δ8, Δ9),

Alors, chaque ε_k est assigné à son C_k, $1 \leq k \leq 5$

Pour defuzzifier le résultat, on calcule d'abord :

$$C = \text{Max} (\varepsilon_k . C_k) \ , \quad 1 \leq k \leq 5$$

Ensuite trouver δ qui est égal au centre de gravité de C, et alors δ est la sortie defuzzifiée du contrôleur.

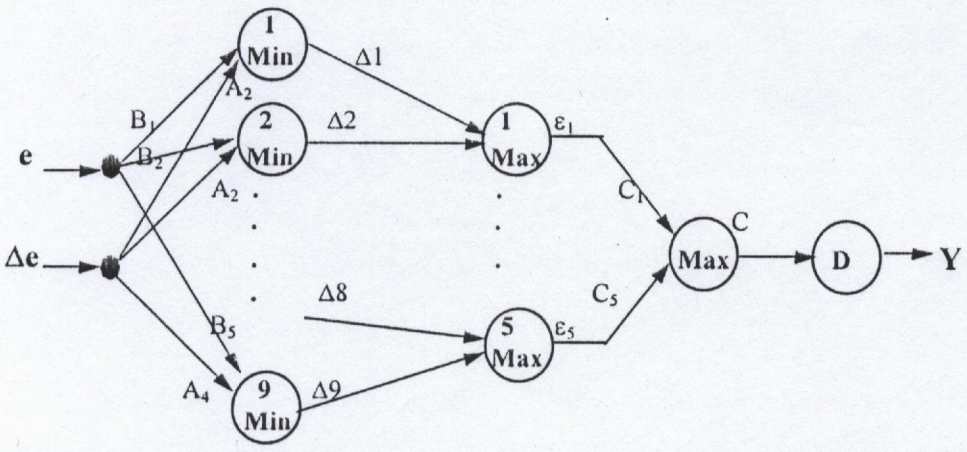


Fig. 3.11 : Structure d'un FNN comme contrôleur flou

Le contrôleur flou modélisé par un FNN est indiqué à la figure (3.11), les noeuds de la couche ET (Min) réalisent le Min de leurs entrées, l'interaction des signaux e et Δe et les

poids flous (B_i et A_i) sert à évaluer B_i en e et A_i en Δe ($A_i(\Delta e)$ et $B_i(e)$). Les neurones de la couche ET (Max) réalisent le Max de leurs entrées, les poids de ces neurones sont tous égaux à un, le neurone suivant prend le Max de ses entrées floues (ε_i. C_i, $1 \leq i \leq 5$). Le dernier neurone a un poids égal à un et assure l'opération de defuzzification.

Ce FNN peut être utilisé à la place d'un contrôleur flou ou bien pour apprendre les règles de contrôle floues (les poids A_i, B_i et C_i) à partir des données d'entraînement. A cause de l'utilisation des opérations Min et Max et les poids flous, ce réseau nécessite un algorithme d'apprentissage spécial basé sur une fonction coût prenant des valeurs floues [48].

Dans [36], Lin et Lu ont proposé une architecture d'un FNN (Fig. 3.12) capable de traiter des informations (signaux d'entrée) réelles ou floues. Le réseau proposé comprend en total cinq couches. L'information floue que peut posséder le réseau est décrite par des nombres flous représentés par 'α-level sets', et peuvent être d'une forme quelconque. La couche d'entrée reçoit des signaux flous ou réels, chaque noeud de cette couche correspond à une seule variable linguistique d'entrée, la cinquième couche consiste en noeuds de sortie dont les sorties sont des nombres flous ou réels, chaque noeud de cette couche correspond à une seule variable linguistique de sortie, les noeuds de la deuxième et quatrième couche définissent les fonctions d'appartenance représentant les termes flous de la variable linguistique respective, chaque noeud de la quatrième couche réalise l'opération Max pour intégrer les règles 'allumées' qui ont la même conséquence. Chaque noeud de la troisième couche représente une règle floue, d'où tous les noeuds de cette couche forment la base de règle floue.

Les noeuds de la deuxième et cinquième couche ont des poids flous, les autres sont numériques, les connexions entre la deuxième et troisième couche définissent les prémises des règles floues et ceux entre la troisième et quatrième couche définissent les conséquences.

Pour l'entraînement du réseau, ils ont développé un algorithme d'apprentissage supervisé qui permet au système proposé d'apprendre les règles de contrôle floues.

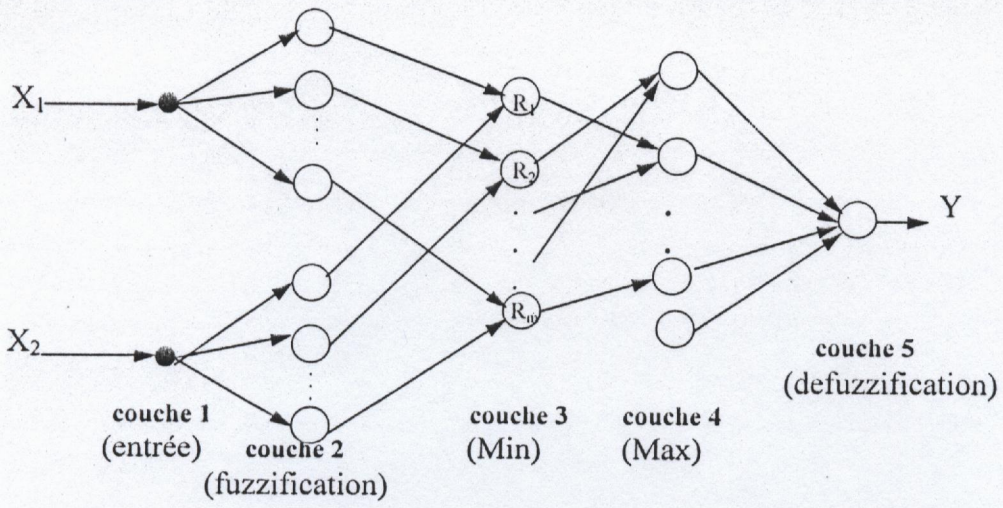


Fig. 3.12 : Structure du FNN proposé dans [36]

III.2.2 L'approche directe

Comme on vient de voir dans les paragraphes précédents, les problèmes à résoudre dans la conception des contrôleurs flous concernent la détermination de l'univers de discours linguistique, la définition des fonctions d'appartenance pour chaque terme linguistique et la dérivation des règles de contrôle [33].

Certains de ces problèmes peuvent être résolus par l'application des algorithmes d'apprentissage, le choix d'un algorithme dépend de la nature du domaine de travail et de l'information disponible. Du à la complexité de l'espace de recherche pour le problème de conception floue, due entre autres à la non dérivabilité et la discontinuité des fonction impliquées, la seule alternative pour résoudre ce type de problème est l'utilisation de méta heuristiques. Parmi celles-ci, les algorithmes génétiques ont été appliqués avec le plus de succès.

L'incorporation de l'apprentissage génétique dans un processus de conception floue ajoute une dimension intelligente au FLC qui lui permet de créer et de modifier ses règles. Les AG offrent la possibilité d'ajuster les fonctions d'appartenance et de générer les règles qui utilisent ces fonctions. L'application des algorithmes génétiques est seulement limitée par la dimension du problème (nombre de chromosomes d'une chaîne).

III.2.2.1 L'application des AG pour les contrôleurs flous :

La difficulté commune dans les systèmes flous est que leurs paramètres doivent être spécifiés par un opérateur (concepteur) humain .

Vu leur application avec succès sur une variété de problèmes d'apprentissage et d'optimisation, les AG ont été proposés comme une méthode d'apprentissage qui permet une génération automatique de paramètres optimaux pour les contrôleurs flous. Pour cela la première étape concerne la définition du problème. Celle ci consiste principalement à définir le ou les paramètres à optimiser, le type de codage des paramètres et la fonction coût.

a) Définition du problème : paramètres à optimiser

Il y a trois approches d'application des AG pour la conception des FLC. Dans un premier cas, les règles linguistiques d'un contrôleur flou conventionnel sont fixées et leurs fonctions d'appartenance sont optimisées par l'AG [31,50]. Dans le second cas, les fonctions d'appartenance de valeurs linguistiques spécifiées sont fixées et l'AG est utilisé pour déterminer un ensemble optimal de règles [49]. Dans la troisième approche les règles et les fonctions d'appartenance sont ajustées simultanément [30,32,33] afin d'accomplir de meilleures performances.

Thrift [49] a examiné la possibilité d'utiliser les AG pour trouver les règles floues. Dans cet article, la synthèse du contrôleur flou est faite sous forme d'une table de vérité.

Karr [50] a examiné l'utilisation d'un AG pour trouver des fonctions d'appartenance de haute performance pour un contrôleur flou, le procédé étudié est celui du 'pole-cart system'.

Dans [31] Karr a utilisé un AG pour produire un FLC adaptatif pour le contrôle de pH en modifiant les fonctions d'appartenance en ligne.

De tels travaux ont montré l'efficacité des AG de créer avec succès les parties individuelles d'un contrôleur flou (l'ensemble des règles et les fonctions d'appartenance), mais l'interdépendance entre ces deux parties suggère qu'une procédure de conception simultanée doit être une méthodologie plus appropriée.

Homaifar et McCormick [32] ont utilisé un AG pour la conception simultanée des fonctions d'appartenance et l'ensemble des règles floues pour un FLC. Ainsi, l'ensemble de règles et les fonctions d'appartenance sont incorporés dans une seule chaîne que l'AG cherche à optimiser

Dans cet article, seulement les longueurs des bases des ensembles flous triangulaires de l'espace d'entrée sont ajustés (ceux de l'espace de sortie sont fixés afin de faciliter le calcul i.e minimisation de la taille de la chaîne),il n'y a pas de localisation des centres.

Linkens et [33] proposent l'optimisation des règles d'inférence et des fonctions d'appartenance triangulaires. La base et le centre de ces fonctions sont variables. Une alternative en ligne est proposée [34].

Zeigler [30] a développé une nouvelle technique, dite HDGA (Hierarchical distributed genetic algorithms) qui est basée sur une stratégie de recherche 'multilevel resolution search strategy' pour la conception d'un FLC optimal en termes de fonctions d'appartenance et de règles d'inférences. Les fonctions d'appartenance peuvent être soit traingulaire soit gaussienne. Le HDGA examine plusieurs alternatives de conception du FLC aussi bien que l'optimisation des ensembles de paramètre correspondant. Cette méthode a été appliquée pour la conception d'un contrôleur flou pour un pendule inversé.

b) Codage des paramètres

Le type de codage utilisé est celui du codage binaire ou entier. où les codes segment de tous les paramètres à optimiser sont combinés pour former une seule chaîne.

- Codage binaire :

Pour illustrer ce type de codage nous présentons le formalisme de [33].

Soit le code du $i^{ème}$ paramètre dans l'espace de recherche R^n de dimension n est désigné par σ^i, si l'alphabet binaire usuel est utilisé, alors σ^i est un ensemble de 0 et de 1 d'une longueur L_i : $\sigma^i \leftrightarrow \{0,1\}^{Li}$.

La structure de chaque chaîne représentant une solution possible du problème d'une population P sera donnée par :

$$A \in P = [\sigma^1\ \sigma^2\ \sigma^3 \ldots\ \sigma^n] \qquad (3.19)$$

La structure d'une règle floue est construite des codes des ensembles flous des antécédents (prémises) et des conséquences linguistiques de la règle. En outre, chaque variable linguistique est définie par sa fonction d'appartenance qui, dans le cas des fonctions d'appartenance triangulaires, est définie par son centre C et sa largeur W (Fig. 3.13).

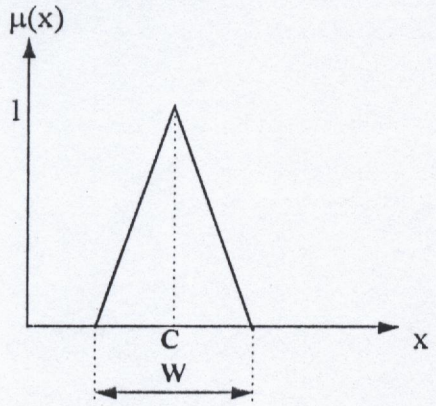


Fig. 3.13 : Fonction d'appartenance triangulaire de centre C et de largeur W

Typiquement, chaque variable linguistique dans une règle floue peut être représentée par trois codes segments : un pour son centre C, un pour sa largeur W et un autre pour sa valeur linguistique (ou nom) L tel que Medium, High, Low, ...

$$\sigma_F = [\ \sigma_L\ \sigma_C\ \sigma_W\ ] \tag{3.20}$$

En considérant une règle floue avec n conditions et m actions, le codage composé d'une règle d'essai $A \in P$ est donnée par :

$$A = [\sigma_{Fc}^{1} \ldots . \sigma_{Fc}^{n} \sigma_{Fa}^{1} \ldots . \sigma_{Fa}^{m}] \tag{3.21}$$

avec σ_{Fc}^{i} ou σ_{Fa}^{i} sont les codes des ensembles flous de la i^{ème} condition ou action.

La figure (3.14) illustre un format de codage typique pour une population de bases de règle floues contenant deux conditions E et CE et une action U. C'est la structure de règles adoptée dans plusieurs applications du contrôle flou, où E représente l'erreur, CE représente la variation de l'erreur (CE ; change in error) et U est l'entrée de contrôle du procédé.

Si l'espace d'entrée est partitionné en sept ensembles flous linguistiques {Negative Big NB, Negative Medium NM, Negative Small NS, Zero ZE, Positive Small PS, Positive Medium PM, Positive Big PB} pour chacune des antécédents de la règle floue, alors la base de règle consiste en 49 règles au maximum.

En représentant le codage total pour la stratégie de contrôle flou sous la forme usuelle d'une table de règles à deux dimensions (Fig. 3.14), le codage des antécédants de la règle E et CE peut être fait implicitement par les positions des règles dans la table. par exemple, toutes les lignes sont associées à une seule valeur de E et toutes les colonnes sont associées à une seule valeur de CE. Donc, des codes explicites sont seulement nécessaire pour les centres et les largeurs des variables floues.

Pour obtenir des attributs quasi-optimaux d'une règle, ses paramètres (les valeurs linguistiques, les centres et les largeurs des ensembles flous) peuvent être modifiés.

Le codage du j^{ième} paramètre dans une chaîne de codes représente une valeur λ_j entre une valeur minimale λ_{min} et maximale λ_{max} prédéterminées donnée par :

$$\lambda_j = \lambda_{min} + (\lambda_{max} - \lambda_{min}) \frac{N(\lambda_j)}{2^{L_j} - 1} \tag{3.22}$$

où Lj est longueur de la chaîne binaire associée à λ_j.

ce / e	NB	NM	NS	ZE	PS	PM	PB
NB	R_1	R_2	...			...	R_7
. . .							. . .
PB							R_{49}

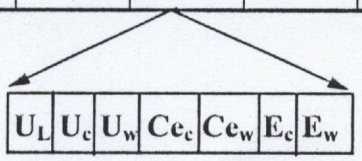


Fig. 3.14 : Codage typique d'une base de règle floue

- Codage entier

Pour coder les paramètres, Homaifar et McCormick [32] ont utilisé un codage spécial qui consiste à utiliser des chaînes à nombres entiers (des nombres compris entre 1 et le nombre d'ensembles flous de l'espace de sortie) au lieu de chaînes binaire, ceci nécessite une nouvelle forme de décodage. La longueur de la chaîne dépend de la taille de l'ensemble de règles et le nombre des ensembles flous utilisés pour partitioner les espaces des variables d'E/S. Ce paradigme a été appliqué sur deux problèmes :'cart-centering system' et 'truck-backing system'.

III.2.2.2 L'apprentissage des systèmes flous par les AG :

Dans l'apprentissage des paramètres du contrôleur flou, l'AG commence par une génération aléatoire d'une population de N chaînes où chaque chaîne complète de code représente une solution possible du problème et comprend un ensemble de règles floues et leurs fonctions d'appartenance.

Chacune de ces chaînes est décodée pour obtenir les paramètres actuels du FLC, ces paramètres sont envoyés au modèle du système à commander, évalués avec une certaine fonction coût tel que l'erreur cumulée à travers le temps, et assignés une valeur d'adaptation 'fitness value'. Cette valeur reflète la qualité de la solution représentée par la chaîne, et peut

être vue comme une mesure qui indique la performance du FLC pour le contrôle du système utilisant les fonctions d'appartenance et les règles associées.

Ces valeurs d'adaptation sont des valeurs sur lesquelles est basée l'application des trois opérateurs (reproduction, croisement et mutation) qui produisent une nouvelle population de chaînes (une nouvelle génération) ;cette nouvelle génération contiendra des paramètres plus performants. les nouvelles chaînes sont décodées, évaluées et transformées en utilisant les opérateurs de base. Ce processus continu jusqu'à ce que la convergence soit accomplie ou à ce qu'une solution optimale soit trouvée.

La configuration d'un tel mécanisme d'apprentissage est indiquée dans la figure (3.15).

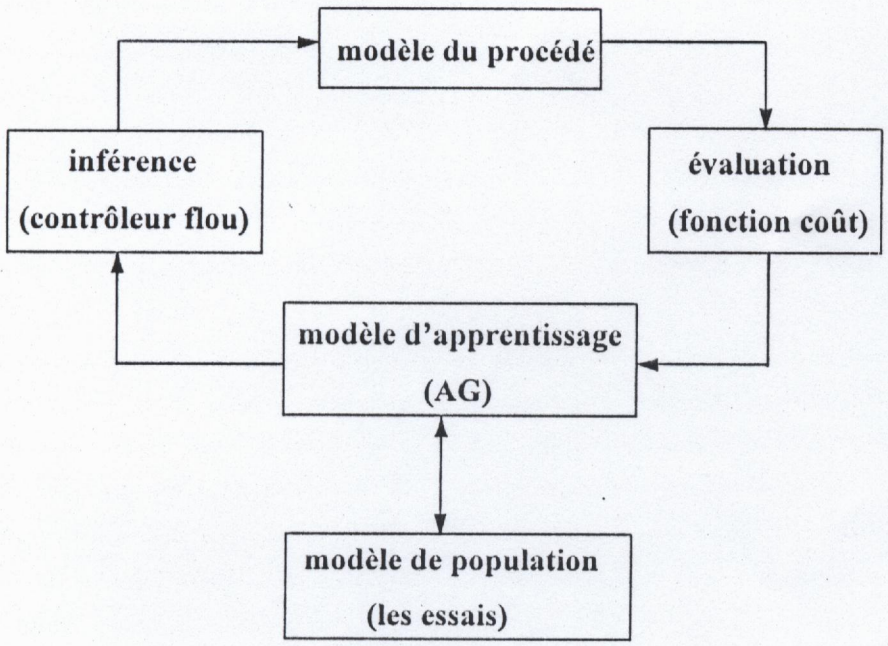


Fig. 3.15 : Mécanisme d'apprentissage off-line

III.3 Conclusion

Les contrôleurs flous sont des candidats optimaux pour la commande des systèmes complexes non identifiés ou partiellement identifiés, qui ne peuvent pas être modélisé facilement d'une manière mathématique.

L'approche traditionnelle pour la conception floue est basée sur les connaissances acquises par des opérateurs experts, bien que cette approche ait prouvé son efficacité dans plusieurs applications, il peut cependant que les opérateurs ne peuvent pas transcrire leurs

connaissances et expériences sous forme de contrôleur à logique floue, en plus le domaine d'expertise n'est pas toujours disponible.

De ce fait, des recherches intensives ont été concentrées vers l'élaboration des méthodes plus systématiques et optimales pour la conception des contrôleurs flous, et ont conduit au développement de deux nouvelles approches d'apprentissage que nous avons étudié dans ce chapitre :

- L'approche connexioniste ;dans ce cas un réseau neuronal flou est utilisé pour implémenter le mécanisme d'inférence flou, l'algorithme de la rétropropagation est alors utilisé pour ajuster les paramètres du réseau et les règles de contrôle sont extraites après convergence.
- L'approche directe ;elle consiste à utiliser un algorithme d'apprentissage pour la conception d'un système flou qui consiste à trouver les fonctions d'appartenance et les règles d'inférence tout en minimisant une certaine fonction coût. Du à la complexité de l'espace de recherche, la méthode d'optimisation requise est celle des algorithmes génétiques.

Conception d'un contrôleur flou par les AG

IV.1 Introduction.

IV.2 Implémentation du contrôleur flou.

IV.3 Description de l'algorithme d'apprentissage.

IV.3.1 Codage des paramètres.

IV.3.2 Mécanisme de l'AG.

a) Reproduction.

b) Croisement.

c) Mutation.

d) La sélection des individus d'une nouvelle génération.

IV.3.3 Décodage des paramètres.

IV.4 Interaction du module contrôleur avec l'AG.

IV.5 Conclusion.

IV.1 Introduction

Dans le chapitre précédent, nous avons présenté plusieurs techniques systématiques de conception des contrôleurs à logique floue. Ces techniques peuvent être classées en deux approches :approche connexioniste et approche directe,.

Dans l'approche connexioniste, l'apprentissage du contrôleur flou est réalisé à l'aide d'un réseau neuronal. L'approche directe utilise généralement, les algorithmes génétiques, une technique d'optimisation qui fonctionne en manipulant les codes des paramètres du système sans connaissance explicite de l'espace de recherche. L'utilisation de ces techniques implique le codage des paramètres de conception, nécessitant ainsi un décodage explicite lors de l'application de la loi de commande. De plus la longueur de la chaîne (string) codant les paramètres peut être inhibitive en terme de temps de calcul et d'espace mémoire, amenant certains auteurs à limiter les paramètres de conception ou utiliser des codages spéciaux.

Dans ce chapitre, nous présentons une nouvelle approche de conception (apprentissage) du contrôleur flou, c'est une approche mixte qui réside dans la combinaison des deux paradigmes: les réseaux de neurones et les AG afin d'assurer une conception optimale d'un FLC. Dans cette approche, on a considéré l'implémentation structurelle d'un système d'inférence flou de type TPE dans un réseau multicouche, les grandeurs ajustables de ce réseau sont les largeurs des différents univers de discours et les poids des connexions. Ainsi l'apprentissage du contrôleur consiste à déterminer les valeurs optimales de ces grandeurs qui minimisent une certaine fonction coût sous certaines contraintes de connectivité du réseau. Les caractéristiques de ce problème font que les méthodes d'optimisation classiques ne peuvent pas être utilisées, pour cela on utilise les algorithmes génétiques.

IV.2 Implémentation du contrôleur flou

Dans cette section, on va considérer l'implémentation structurelle d'un contrôleur flou dans un réseau multicouche.

Le réseau qu'on a adopté consiste en cinq couches (Fig. 4.1), une couche d'entrée, une couche de fuzzification, une couche de neurones AND, une couche de neurones OR et une couche de défuzzification. Chaque neurone de ces couches réalise une fonction particulière sur ses signaux d'entrée utilisant un ensemble de paramètres spécifique à ce neurone.

Le choix de cette fonction dépend de la fonction totale que le réseau est sensé réaliser.

L'architecture du réseau (Fig. 4.1) comporte les couches suivantes :

- **Couche d'entrée (input layer)**, le nombre de neurones de cette couche correspond au nombre de variables (réels) d'entrée du contrôleur. Aucune opération n'est effectuée au niveau de cette couche.
- **Couche de fuzzification (fuzzification layer)**, cette couche permet la décomposition de l'univers de discours pour chaque variable d'entrée en un ensemble fixe de sous ensembles flous selon la stratégie TPE.

  Chaque neurone i de cette couche reçoit une seule entrée soit x à partir de la couche d'entrée et génère une sortie donnée par :

$$Out_i^F = \mu_{Ai}(x) \tag{4.1}$$

  où Ai est la valeur linguistique (LOW,MEDIUM,HIGH,...) associée à ce neurone, en d'autre terme, Out_i^F est la fonction d'appartenance de Ai qui indique à quel degré une entrée x donnée satisfait Ai. Les poids de cette couche sont tous égaux à un.
- **Couche AND (AND layer)**, chaque neurone de cette couche représente une règle floue, donc cette couche comprend un nombre de neurones AND égal au nombre de règles.

  Chaque neurone reçoit à partir de la couche de fuzzification un nombre d'entrée égal au nombre de conditions (antécédents) de la règle, ces entrées représentent les fonctions d'appartenance.

  L'opération logique AND réalisée par ce neurone est implémentée par l'opérateur norme-T défini par :

$$T(x,y) = x.y \tag{4.2}$$

  ainsi la sortie du $i^{ème}$ neurone est donnée par :

$$Out_i^{AND} = \mathop{T}_{j=1}^{n} Out_j^F \quad , \ i = 1,...,Na \tag{4.3.a}$$

ou encore :

$$Out_i^{AND} = \prod_{j=1}^{n} Out_j^F \quad , \ i = 1,...,Na \tag{4.3.b}$$

avec : Na : le nombre de neurones AND.

n : le nombre des conditions de la règle.

Out_j^F : les sorties floues de la couche de fuzzification qui représentent les degrés d'appartenance des antécédents de la $i^{ème}$ règle.

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

- **Couche OR (OR layer)**, chaque neurone de cette couche reçoit toutes les sorties des neurones de le couche AND et réalise l'opération logique OR en utilisant les opérateurs norme-T et norme-S définis par :

$$T(x,y) = x.y \tag{4.4}$$

$$S(x,y) = x + y - x.y \tag{4.5}$$

La sortie du $j^{ème}$ neurone est donnée par :

$$Out_j^{OR} = \overset{Na}{\underset{i=1}{S}}\left(T(Out_i^{AND}, Wij)\right) \quad , \quad j=1,...,No \tag{4.6}$$

d'où :

$$Out_j^{OR} = 1 - \prod_{i=1}^{Na}\left[1 - Out_i^{AND}.Wij\right] \quad , \quad j=1,...,No \tag{4.7}$$

avec : Wij : les poids des connexions entre le $i^{ème}$ neurone AND et le $j^{ème}$ neurone OR.

No : le nombre de neurones OR.

Le nombre de neurones de la couche OR est égal au nombre de sous ensembles flous utilisés pour partitioner l'univers de discours de l'espace de sortie.

Les poids de cette couche sont ajustables et ne peuvent prendre que deux valeurs possibles soit zéro ou bien un et ils sont assujettis à la contrainte suivante :

$$\sum_{j=1}^{N0} Wij = 1 \,, \forall\ i = 1,..,Na \tag{4.8}$$

Cela signifie que parmi les poids connectant le $i^{ème}$ neurone AND aux différents neurones OR, un seul poids est égal à un et les autres sont égaux à zéro.

Physiquement cette contrainte signifie que pour des valeurs linguistiques données des antécédents d'une règle, une seule valeur linguistique est associée à l'action (conséquence) de cette règle.

- **Couche de défuzzification (defuzzification layer)**, le neurone de cette couche réalise l'opération de défuzzification selon la règle du centre de gravité et la méthode TPE à savoir :

$$\hat{y} = \frac{\sum_{i=1}^{No} C_i . Out_i^{OR}}{\sum_{i=1}^{No} Out_i^{OR}} \tag{4.9}$$

où $\hat{y}$ est valeur numérique de sortie, Out_i^{OR} sont les sorties des neurones de la couche OR (les valeurs d'appartenance de l'interface de sortie)et C_i sont les centres des fonctions d'appartenance symétriques de la sortie.

Les centres C_i peuvent être considérés comme des poids des connexions entre la couche OR et la couche de défuzzification.

La figure (4.1) illustre un exemple d'un tel réseau avec des univers de discours divisés en trois sous ensembles flous LOW (L), MEDIUM (M) et HIGH (H) pour toutes les variables d'E/S. Le réseau possède deux entrées et une sortie, avec neuf neurones AND (9 règles floues i.e toutes les combinaisons possibles des variables linguistiques des entrées) et trois neurones OR.

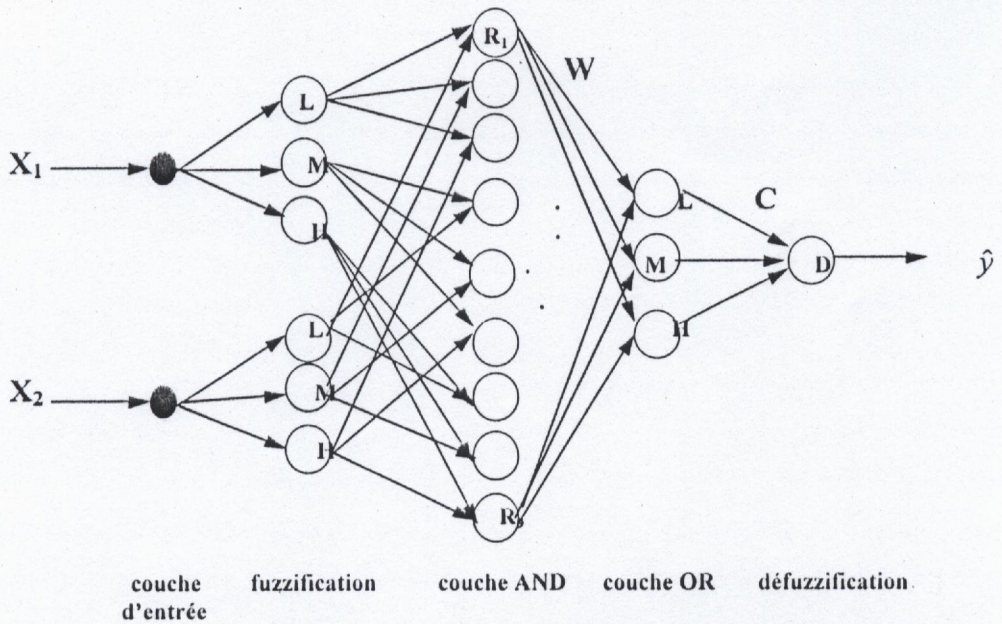


Fig. 4.1 : Architecture du réseau contrôleur

Les règles que ce réseau réalise sont de la forme :

$$R^i : Si\ (x_1\ est\ \{L,M\ ou\ H\})\ et\ (x_2\ est\ \{L,M\ ou\ H\})\ Alors\ (y\ est\ \{L,M\ ou\ H\})$$

exemple : La règle R^3 est donnée par :

$$R^3 : Si\ (x_1\ est\ L\ et\ x_2\ est\ H)\ Alors\ (y\ est\ \{L,M\ ou\ H\})$$

en prenant par exemple $W3=[w_{31}\ w_{32}\ w_{33}]=[0\ 1\ 0]$, alors la règle R^3 deviendra :

$$R^3 : Si\ (x_1\ est\ L\ et\ x_2\ est\ H)\ Alors\ (y\ est\ M)$$

IV.3 Description de l'algorithme d'apprentissage

L'idée de base de notre travail consiste à assurer une conception simultanée des fonctions d'appartenance et l'ensemble de règles pour un contrôleur flou que nous avons projeté sur un réseau multicouche.

Pour cela, les grandeurs ajustables sont les suivantes :

- Les largeurs (L) des différents univers de discours (Fig. 4.2) des variables d'E/S, cet ajustement permet d'altérer les fonctions d'appartenance associées.
- Les poids (W) du réseau connectant la couche AND et la couche OR, l'ajustement de ces poids permet de modifier les règles de contrôle floues.

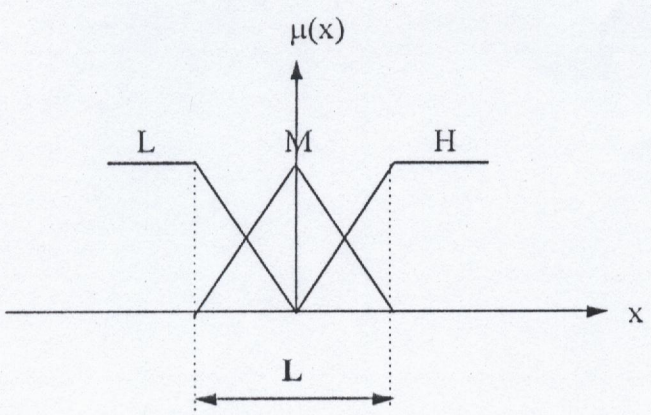


Fig. 4.2 : Univers de discours (TPE) de largeur L

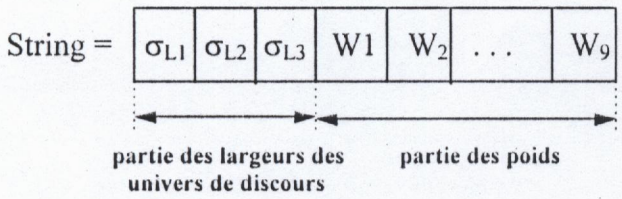
Ainsi l'apprentissage du contrôleur consiste à déterminer les largeurs des univers de discours et les poids qui minimisent une certaine fonction coût et tel que les contraintes (4.8) soient satisfaites.

Les caractéristiques de ce problème font que les méthodes d'optimisation classiques ne peuvent pas être utilisées, pour cela on utilise l'algorithme génétique.

IV.3.1Codage des paramètres

Des chaînes binaires d'une longueur donnée sont utilisées pour coder les largeurs des univers de discours. On n'a pas besoin de coder les poids puisqu'ils prennent des valeurs binaires (0 ou 1). Donc chaque chaîne complète de la population contient les valeurs des poids et les codes des différentes largeurs des univers de discours.

Ainsi pour le contrôleur flou représenté par le réseau de la figure (4.1), la chaîne aura la forme suivante :



où $\sigma_{L_1}, \sigma_{L_2}, \sigma_{L3}$: les codes des largeurs des univers de discours (L_1,L_2,L_3) des entrées X_1 et X_2 et de la sortie $\exists$ respectivement.

W_i :vecteur des poids connectant le $i^{ème}$ neurone AND aux différents neurones OR :

$W_i = [w_{i1}\ w_{i2}\ w_{i3}]$, $i = 1 .. 9$

IV.3.2 Mécanisme de l'AG

L'AG commence par une génération aléatoire d'une population de N chaînes où chaque chaîne doit satisfaire la contrainte (4.8) (initialisation conditionnée). De nouvelles générations sont crées en utilisant les opérations génétiques :reproduction, croisement et mutation.

En vue de respecter la contrainte citée ci-dessus, on a introduit un certain nombre d'opérations supplémentaires lors du croisement et de la mutation. Ces opérations consistent en des perturbations qui n'affectent pas leur aspect aléatoire.

a) Reproduction

Dans notre travail deux méthodes de sélection des individus de la prochaine génération ont été considérées : la sélection par roue de loterie (sélection proportionnelle) et

la sélection à reste stochastique.

b) Croisement

Un croisement ordinaire entre deux chaînes parents aura lieu si la position ou se produira le croisement se trouve à l'intérieur de la partie de la chaîne représentant les largeurs des différents univers de discours.

Dans le cas ou le point du croisement se trouve à l'intérieur de la partie de la chaîne qui représente les poids, le croisement consistera à échanger l'ensemble des poids (le vecteur W_i) correspondant à ce point, tout en satisfaisant la contrainte cela signifie qu'une seule règle sera changée.

En considérant le réseau contrôleur de la figure (4.1), le processus du croisement conditionné est illustré dans la figure (4.3).

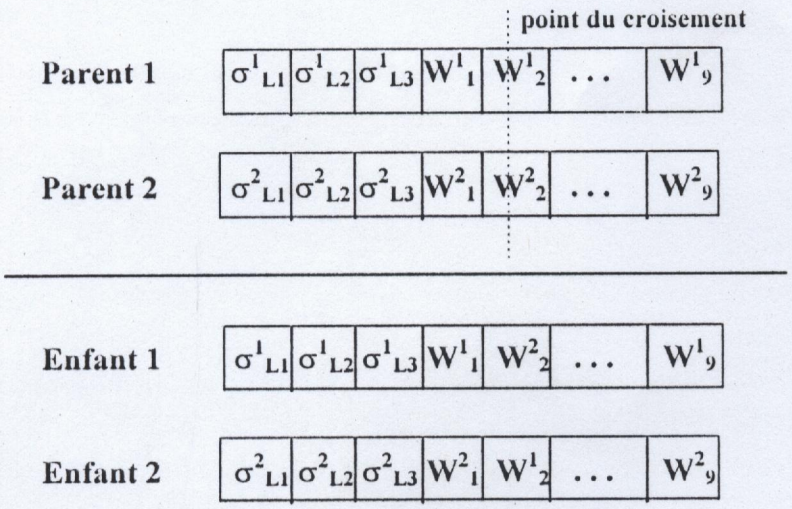


Fig. 4.3 : Principe du croisement conditionné

c) Mutation

De la même manière, si une mutation a lieu et la position du bit à muter se trouve dans la partie des largeurs des univers de discours de la chaîne, alors la valeur de ce bit sera complémenté (mutation ordinaire).

Dans le cas ou la position du bit à muter se trouve dans la partie des poids, alors l'ensemble des poids à qui appartient ce bit sera transformé à un autre ensemble de poids valide (vérifiant la contrainte).

exemple : Considérons toujours le contrôleur de la figure (4.1), le processus de la mutation conditionnée est indiqué dans la figure (4.4) où le vecteur $W_2 = [0\ 1\ 0]$ sera transformé à l'ensemble [0 0 1] ou [1 0 0].

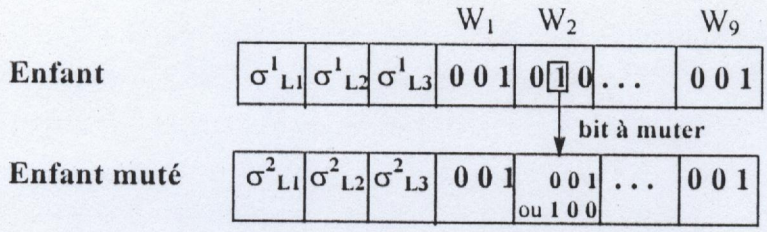


Fig. 4.4 : Principe de la mutation conditionné

d) La sélection des individus d'une nouvelle génération

Dans notre travail on a considéré deux stratégies de sélection finale :

- la sélection par compétition.
- steady state selection.

IV.3.3 Décodage des paramètres

Chaque chaîne doit être décodée afin d'obtenir les paramètres actuels du contrôleur et de pouvoir calculer la valeur de la fonction coût qui lui est associée.

Le décodage binaire est utilisé et concerne seulement les largeurs des différents univers de discours, les valeurs des poids sont prises telles quelles.

IV.4 Interaction du module contrôleur avec l'AG

Le problème qui apparaît dans l'optimisation est celui de l'utilisation des modèles mathématiques pour évaluer le coût d'une chaîne donnée.

L'un des plus fort points des contrôleurs flous est le fait qu'ils n'ont pas besoin de tels modèles. Cependant, pour obtenir un coût d'un contrôleur donné, l'AG doit avoir un modèle du système à commander pour évaluer la performance du contrôleur. Ce modèle peut être un modèle mathématique linéaire ou non linéaire, un modèle neuronal ou un modèle 'flou', l'essentiel est qu'il permette de simuler le comportement du système.

La figure (4.5) illustre les interactions de l'AG du module du réseau contrôleur et le modèle de simulation du procédé.

La figure (4.6) montre un organigramme détaillé des opérations; au début de la simulation, l'AG génère une population initiale d'individus où chaque individus représente les paramètres du réseau contrôleur. Ces individus sont envoyés au module contrôleur pour initialiser un certain ensemble de fonctions d'appartenance (largeurs des univers de discours) et de règles floues (les poids), le contrôleur envoie donc des commandes opérationnelles au modèle de simulation et la trajectoire du procédé est échantillonnée pour calculer la performance du contrôleur qui considère la minimisation des composants suivants :l'erreur totale entre la trajectoire désirée et la trajectoire actuelle et l'énergie utilisée pour contrôler le procédé.

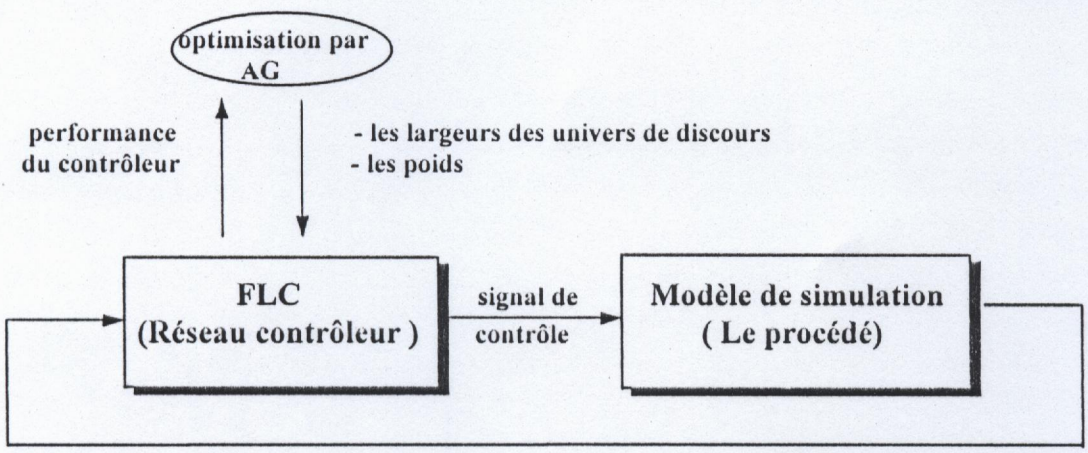


Fig. 4.5 : Diagramme des interactions de l'AG, FLC et le modèle de simulation

Après l'évaluation de tous les individus de cette population, une nouvelle génération sera crée par l'AG en utilisant les opérateurs génétiques, de même les chaînes de cette nouvelle population seront décodées puis évaluées, ce processus continu jusqu'à ce que la convergence soit accomplie. le critère de convergence qu'on a choisi est fixé par le nombre maximale de générations.

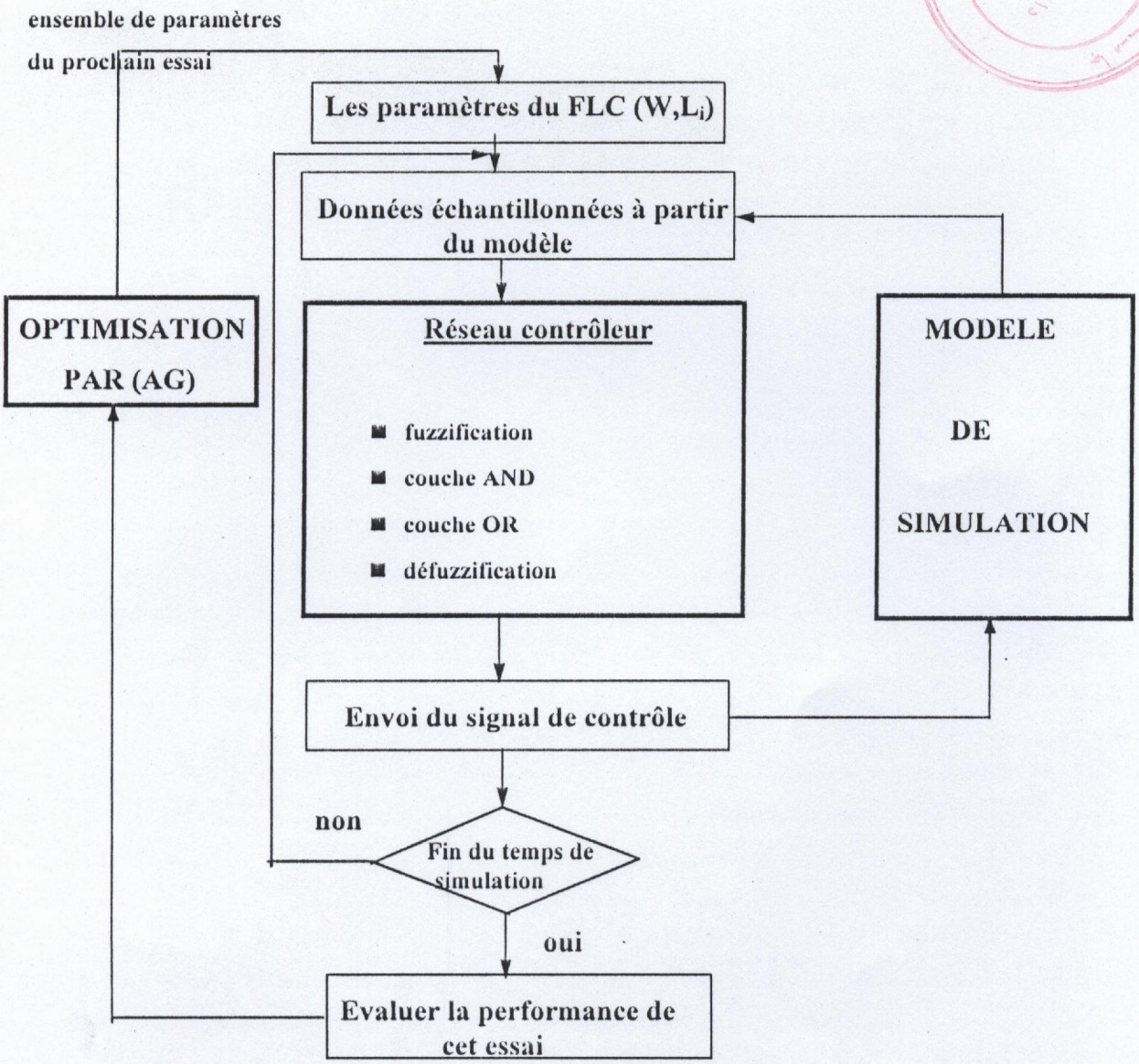


Fig. 4.6 : Organigramme de l'opération du contrôleur

IV.5 Conclusion

Dans ce chapitre nous avons proposé une méthodologie de conception des contrôleurs à logique floue à l'aide des algorithmes génétiques.

Cette méthodologie permet une conception complète des deux composants majeurs des contrôleurs flous, l'ensemble des règles et les fonctions d'appartenance donnant des contrôleurs performants complètement conçus par calculateurs.

Pour cela nous avons considéré l'implémentation structurelle d'un contrôleur flou du type TPE dans un réseau connexioniste multicouche.

Les grandeurs ajustables de ce réseau sont les largeurs des univers de discours des espaces d'E/S et les poids des connexions, ces derniers sont assujettis à une certaine contrainte.

Ainsi l'apprentissage du réseau consiste à obtenir des valeurs optimales pour ces grandeurs qui minimisent une certaine fonction coût et tel que la contrainte cité ci-dessus soit satisfaite.

Vu les caractéristiques de ce problème, on a utilisé un algorithme simple avec trois opérations :reproduction, croisement et mutation pour l'optimisation de ces grandeurs.

En vu de respecter la contrainte imposée sur les poids, on a introduit un certain nombre d'opérations supplémentaires lors du croisement et de la mutation.

Résultats de simulation

V.1 Introduction.

V.2 Le système du pendule inversé.

V.3 Le contrôle du pendule inversé.

V.3.1 Structure de simulation de la commande .

V.3.2 Conception du contrôleur.

V.3.3 Contrôleur 333.

V.3.3.1 Résultats de simulation.

V.3.3.2 Robustesse du contrôleur 333.

V.3.4 Contrôleur 555.

V.3.4.1 Résultats de simulation

V.3.4.2 Robustesse du contrôleur 555.

V.4 Conclusion.

V.1 Introduction

L'approche de conception des contrôleurs à logique floue que nous avons développé au chapitre précédent est tout à fait générale et elle peut être appliquée à une variété de problèmes de contrôle. Dans ce chapitre, on va démontrer l'efficacité de cette méthode en l'appliquant à la conception d'un contrôleur flou pour un problème de référence 'benchmark problem' dans le contrôle intelligent - le système du pendule inversé.

Le pendule étant à un angle et vitesse angulaire non nuls, l'objectif est de déterminer un contrôleur flou capable de ramener le pendule à sa position d'équilibre (verticale) en minimisant une certaine fonction coût. Deux types de contrôleurs ont été conçus pour ce problème en divisant les espaces d'entrée et de sortie en deux formes de partition.

L'AG utilisé pour l'apprentissage du contrôleur a été testé pour différentes méthodes de reproduction et de sélection finale.

V.2 Le système du pendule inversé

La figure (5.1) montre le diagramme schématique du système du pendule inversé.

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

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

- θ :l'angle du segment par rapport à l'axe vertical.
- $\dot{\theta}$:la vitesse angulaire du segment.

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

$$\ddot{\theta} = \frac{g.\sin\theta + \cos\theta.\left(\dfrac{-F - m.l.\dot{\theta}^2.\sin\theta}{m_c + m}\right)}{l.\left(\dfrac{4}{3} - \dfrac{m.\cos^2\theta}{m_c + m}\right)} \qquad (5.1)$$

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

m_c, masse du chariot = 1 Kg.

m, masse du segment = 0.1 Kg.

l, demi longueur du segment = 0.5 m.

F, la force appliquée en Newton (N).

Si on pose :

$$\begin{cases} x_1(t) = \theta(t) \\ x_2(t) = \dot{\theta}(t) \end{cases}$$

Le système décrit par l'équation (5.1) sera défini par les équations différentielles suivantes :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dfrac{g.\sin(x_1) + \cos(x_1).\left(\dfrac{-F - m.l.x_2^2.\sin(x_1)}{m_c + m}\right)}{l.\left(\dfrac{4}{3} - \dfrac{m.\cos^2(x_1)}{m_c + m}\right)} \end{cases} \tag{5.2}$$

Ce système est simulé par la méthode d'Euler avec une période d'échantillonnage égale à 10 ms.

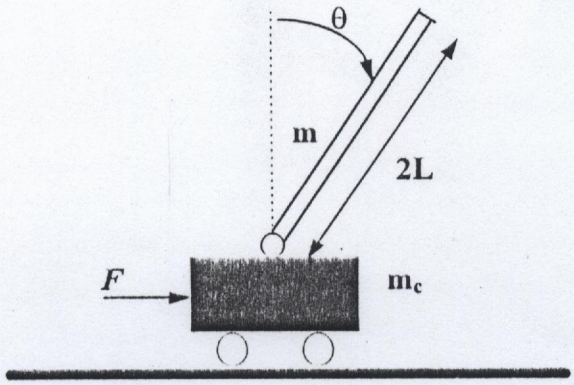


Fig. 5.1 : Le système du pendule inversé

V.3 Le contrôle du pendule inversé

V.3.1 Structure de simulation de la commande

Comme on a vu au chapitre précédent, chaque fois qu'un contrôleur est conçu par l'algorithme génétique, il est testé en simulation sur le modèle du système à commander pour évaluation. La structure de simulation de la commande dans le cas du pendule inversé est illustrée par la figure (5.2), elle est constituée de deux blocs :

- **Modèle du système :** Comme mentionné précédemment, il y a plusieurs manières

de modéliser le système à commander à savoir fonction de transfert, des équations d'état, un modèle neuronal, modèle 'flou'..., cela dépend de notre connaissance du système.
Dans notre cas, le procédé est un système dynamique non linéaire déterministe avec des équations différentielles définies avec précision. Ainsi, on peut juste employer une approximation linéaire pour obtenir les équations aux différence suivantes :

$$\begin{cases} x_1(t+h) = h.\dot{x}_1(t) + x_1(t) \\ x_2(t+h) = h.\dot{x}_2(t) + x_2(t) \end{cases} \tag{5.3}$$

où :

$x_1(.) = \theta(.)$

$x_2(.) = \dot{\theta}(.)$

h est la période d'échantillonnage

- **Bloc du contrôleur :** On suppose qu'on ne dispose pas de contrôleur obtenu par expertise. Si c'est le cas, ce contrôleur sera pris comme contrôleur initial au cours de l'optimisation. Le contrôleur flou dans la figure (5.2) est réalisé par un réseau interconnecté multicouche qui a été présenté au chapitre IV, avec :

- Les signaux d'entrée du contrôleur : l'angle du segment (θ en degré) et la vitesse angulaire du segment ($\dot{\theta}$ en deg./s), soit le vecteur d'entrée X=(θ,$\dot{\theta}$).
- Le signal de sortie du contrôleur : la force (F en Newton).

Le contrôleur a pour objectif de générer une force F qui va stabiliser le pendule dans sa position d'équilibre (verticale) qui correspond à un angle et vitesse angulaire nuls. Pendant l'optimisation (l'apprentissage), l'algorithme génétique délivre à chaque génération un contrôleur qui est évalué et le processus de simulation est réalisé pendant un temps T=5s.
Une fois le contrôleur optimal est trouvé, il est appliqué en simulation sur le même modèle pour tester ses performances.

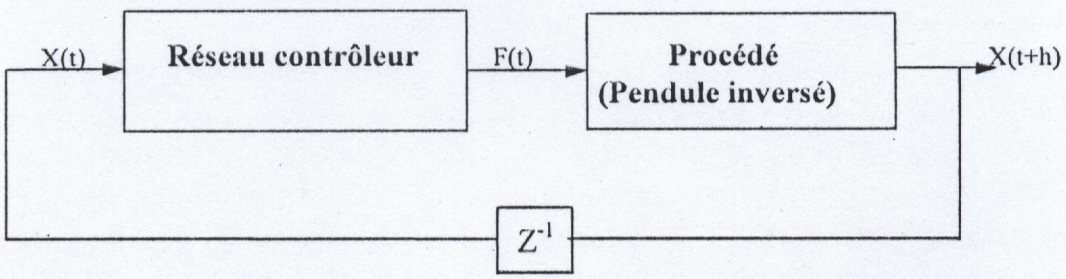


Fig. 5.2 : Structure de commande du pendule inversé

V.3.2 Conception du contrôleur

L'approche utilisée pour la conception du contrôleur flou pour le pendule inversé est celle développée dans le chapitre IV. L'AG a pour objectif de trouver des valeurs optimales pour les différentes largeurs des univers de discours (un ensemble optimal de fonctions d'appartenance) et pour les poids (ensemble optimal de règles floues) en minimisant le critère suivant :

$$J = \sum_{k=1}^{500} p_1 . |\theta(k)| + p_2 . |F(k-1)| \qquad (5.4)$$

avec p_1 et p_2 sont des facteurs de pondération que nous avons fixé empiriquement :

$p_1 = 1$ et $p_2 = 10^{-2}$ pour $t \leq 2$ s

$p_1 = 50$ et $p_2 = 10^{-3}$ pour $2 \text{ s} < t \leq 5$ s

Comme les AG sont des procédures de recherche du maximum, le problème de minimisation du critère J est transformé en problème de maximisation d'une fonction d'adaptation f définie par :

$$f = \left(\frac{C_1}{J}\right)^{C_2} \qquad (5.5)$$

Avec C_1 et C_2 sont des paramètres choisis de tel manière à avoir une variance de la fonction d'adaptation (les différences dans la fonction d'adaptation relative entre les individus) grande dans la population pour assurer une meilleur recherche. Ces paramètres sont choisis empiriquement par :

$$C_1 = 10^4$$

$$C_2 = 2$$

Les conditions physiques du pendule inversé dans notre simulation sont :

- angle initial : $\theta(0) = 20^0$ et vitesse angulaire initiale : $\dot{\theta}(0) = 0^0 / s$

Deux contrôleurs flous ont été développés pour le problème du pendule inversé, ils sont référés par le nombre d'ensembles flous utilisés pour la partition de l'univers de discours de l'angle, de la vitesse angulaire et de la force. Par exemple le contrôleur qui a un angle, une vitesse angulaire et une force tous divisés en trois ensembles flous est appelé contrôleur 333.

V.3.3 Contrôleur 333

Dans ce type de contrôleur, l'angle, la vitesse angulaire et la force sont tous divisés en trois ensembles flous :NEGATIVE (NE), ZERO (ZE) et POSITIVE (PO). Neuf règles floues

de contrôle sont utilisées ceci en considérant toutes les combinaisons possibles des variables floues d'entrée. L'architecture du réseau contrôleur est spécifiée par :

- Nombre de neurones de la couche d'entrée : 2
- Nombre de neurones de la couche de fuzzification : 6
- Nombre de neurones AND : 9
- Nombre de neurones OR : 3
- Nombre de neurones de la couche de défuzzification : 1

L'AG est utilisé pour trouver les valeurs optimales des différentes largeurs des univers de discours soient: $L_\theta, L_{\dot{\theta}}$ et L_F associées à l'angle, la vitesse angulaire et la force respectivement et les poids W (27 poids) en maximisant le critère décrit par (5.5).

La recherche des paramètres est effectuée selon le tableau ci dessus :

Paramètres	Intervalle de recherche
L_θ	[2 80]
$L_{\dot{\theta}}$	[2 100]
L_F	[2 200]

Tab.5.1 : Intervalles de recherche des paramètres pour le contrôleur 333

Le tableau (5.2) illustre les valeurs des différents paramètres de L'AG.

Un codage sur 8 bits est utilisé pour coder chacune des largeurs des différents univers de discours d'où chaque chaîne complète de la population contiendra 51 bits (24 bits pour les différentes largeurs et 27 bits pour les poids).

Paramètres génétiques	Valeur
Probabilité de croisement (P_c)	0.9
Probabilité de mutation (P_m)	0.02
Taille de la population (Nombre d'individus)	100
Nombre de génération	100

Tab.5.2 : Les valeurs des paramètres génétiques

V.3.3.1 Résultats de simulation

L'AG utilisé pour optimiser les paramètres du contrôleur 333 a été testé en employant différentes stratégies de reproduction et de sélection finale (sélection des individus d'une nouvelle génération), les cas d'études que nous avons considéré sont les suivants :

Cas1 : La reproduction est réalisé par la méthode de sélection à reste stochastique et la sélection finale est réalisée par la méthode de sélection "steady state ".

Cas2 : La reproduction est réalisée par la même méthode considérée dans le premier cas et la sélection finale est réalisée par la méthode de sélection par compétition.

Cas3 : Dans ce cas, nous avons utilisé la méthode de sélection par roue de loterie pour la reproduction et la méthode de sélection "steady state " pour la sélection finale.

Cas4 : La reproduction est réalisée par la même méthode utilisée dans le troisième cas c'est à dire par roue de loterie et la méthode de sélection par compétition est considérée pour la sélection finale.

Les caractéristiques du processus d'apprentissage sont analysées par la fonction d'adaptation moyenne 'mean fitness', MF, de la population et la fonction d'adaptation du meilleur individus 'best fitness', BF, dans la population.

Les figures (Fig. 5.4.a) et (Fig. 5.4.b) montrent les allures des fonctions d'adaptations BF et MF, pour chaque génération des quatre cas considérés ci-dessus.

On constate que la reproduction par la méthode de sélection à reste stochastique est plus performante que celle par roue de loterie. Ceci peut s'expliquer par le fait que la sélection à reste stochastique est une méthode élitiste qui garantit la survie du meilleur individus de la population et le conserve pour la génération suivante par contre dans la sélection par roue de loterie il est tout à fait possible que le meilleur individu de la population ne soit pas retenu dans la prochaine génération.

D'un autre côté, la méthode de sélection "steady state " pour la sélection finale retient les deux meilleurs individus parmi les quatre (enfants et parents) après chaque recombinaison génétique ce qui permet d'accélérer le processus d'apprentissage (convergence rapide), la sélection finale par compétition qui consiste à effectuer une compétition entre parents et enfants pour déterminer les survivants de la génération garantit une bonne convergence mais relativement lente par rapport à la "steady state selection " .

On peut conclure donc que la reproduction par la méthode de sélection à reste stochastique combinée avec la méthode de la "steady state selection " donne la meilleure convergence.

Les résultats du processus d'apprentissage obtenues par l'AG employant ces deux méthodes sont :

- valeur maximale de la fonction d'adaptation : 53.5467.
- $L_\theta = 15.4588$.
- $L_{\dot{\theta}} = 48.5019$.
- $L_F = 200$.
- $W_1 = [1\ 0\ 0]$, $W_2 = [1\ 0\ 0]$, $W_3 = [0\ 1\ 0]$.
- $W_4 = [1\ 0\ 0]$, $W_5 = [0\ 1\ 0]$, $W_6 = [0\ 0\ 1]$.
- $W_7 = [0\ 1\ 0]$, $W_8 = [0\ 0\ 1]$, $W_9 = [0\ 0\ 1]$.

La figure (5.3) montre les différentes fonctions d'appartenance de l'angle, la vitesse angulaire et de la force et le tableau (5.3) représente la base de règles de contrôle floues pour la commande du pendule inversé.

Les figures (Fig. 5.5.a-d) représentent (a) l'angle en degré, (b) la vitesse angulaire en degré par seconde et (d) les actions de contrôle (N) à partir de t=0 à t=5s, (c) est l'espace d'état qui montre comment la trajectoire approche l'origine à partir du point initial (20,0).

On remarque que le contrôleur arrive à stabiliser le pendule sans oscillations autour de la position d'équilibre pendant un temps moins de 2s, le retour à l'état d'équilibre se fait pendant deux phases; une phase d'accélération et une phase de décélération.

$\dot{\theta}$ \ θ	NE	ZE	PO
NE	**NE**	**NE**	**ZE**
ZE	**NE**	**ZE**	**PO**
PO	**ZE**	**PO**	**PO**

Tab.5.3 : Base de règles floues du contrôleur 333

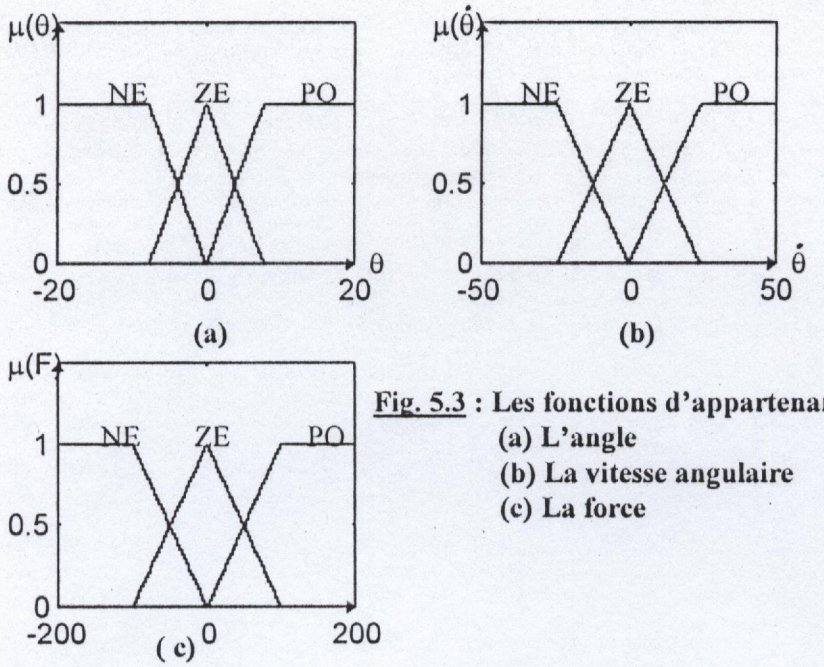


Fig. 5.3 : Les fonctions d'appartenance
(a) L'angle
(b) La vitesse angulaire
(c) La force

V.3.3.2 Robustesse du contrôleur 333

Le contrôleur flou obtenu est capable de ramener le pendule à sa position d'équilibre à partir du point initial (20,0). Cependant un bon contrôleur doit être robuste, pour cela le contrôleur 333 a été testé pour plusieurs conditions initiales. Les figures (Fig. 5.6.a-d) et (Fig. 5.7.a-d) montrent la performance du contrôleur flou optimisé pour le contrôle du pendule inversé avec des conditions initiales différentes. On constate que le contrôleur arrive toujours à stabiliser le pendule, et prend plus de 4s pour balancer le pendule avec un angle initial de 70° mais échoue pour un angle initial supérieur à 80°.

Pour voir comment le contrôleur va réagir aux changements des paramètres du procédé, on l'a testé sur des segments de différentes longueurs (0.25, 0.5, 1 et 1.5 m), les résultats sont donnés dans les figures (Fig. 5.8.a-d). Il est remarquable de noter que le contrôleur peut gouverner le pendule court plus facilement.

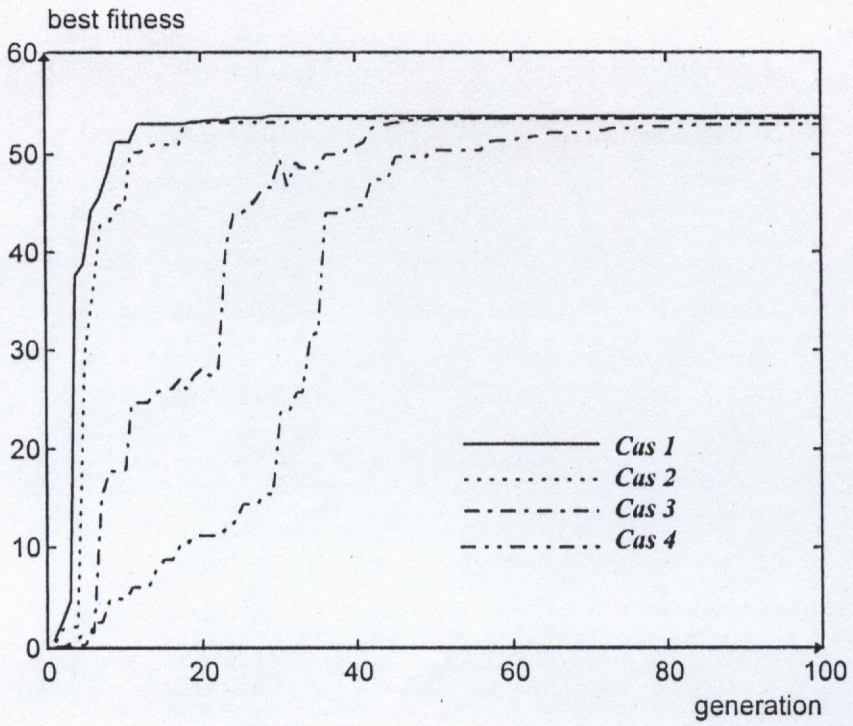


Fig. 5.4.a

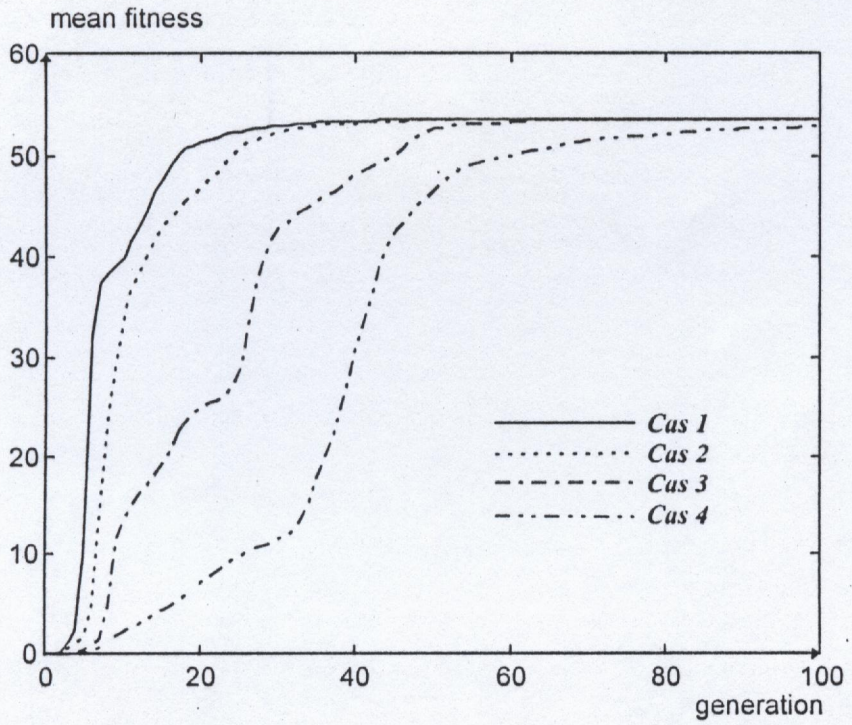


Fig. 5.4.b

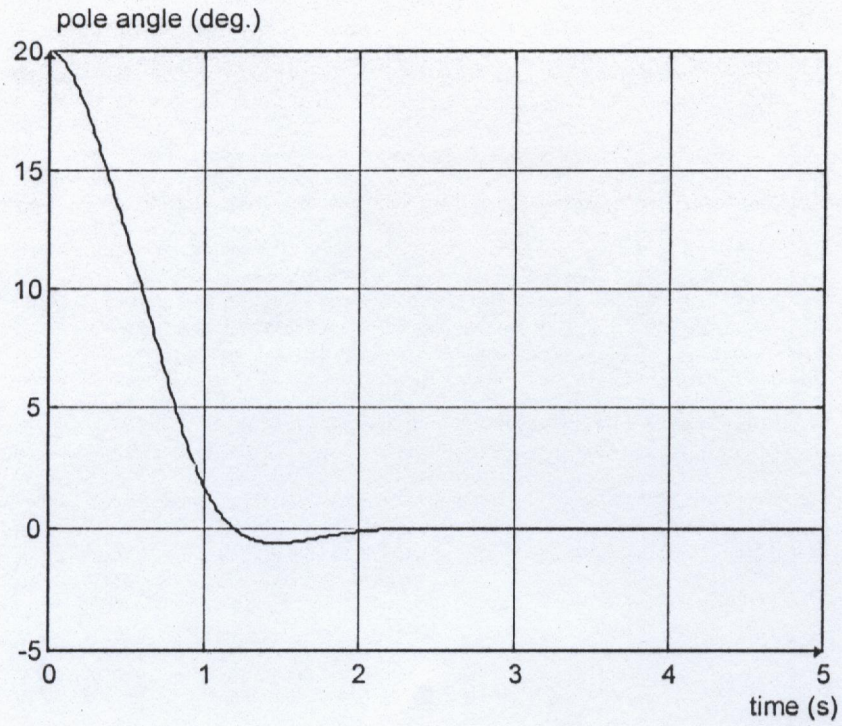


Fig. 5.5.a : Variations de l'angle à partir du point (20°,0°/s)

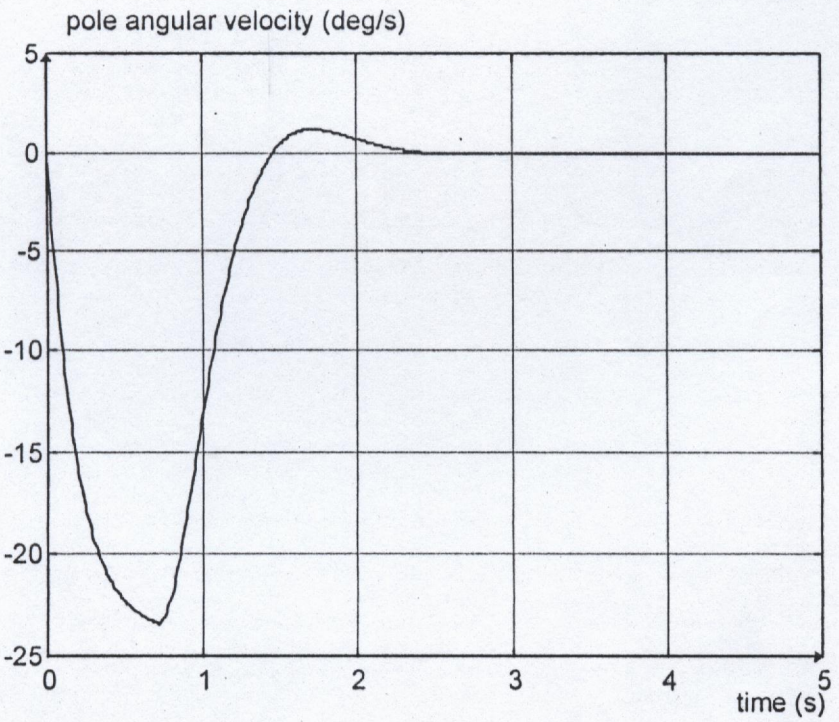


Fig. 5.5.b : Variations de la vitesse angulaire à partir du point (20°,0°/s)

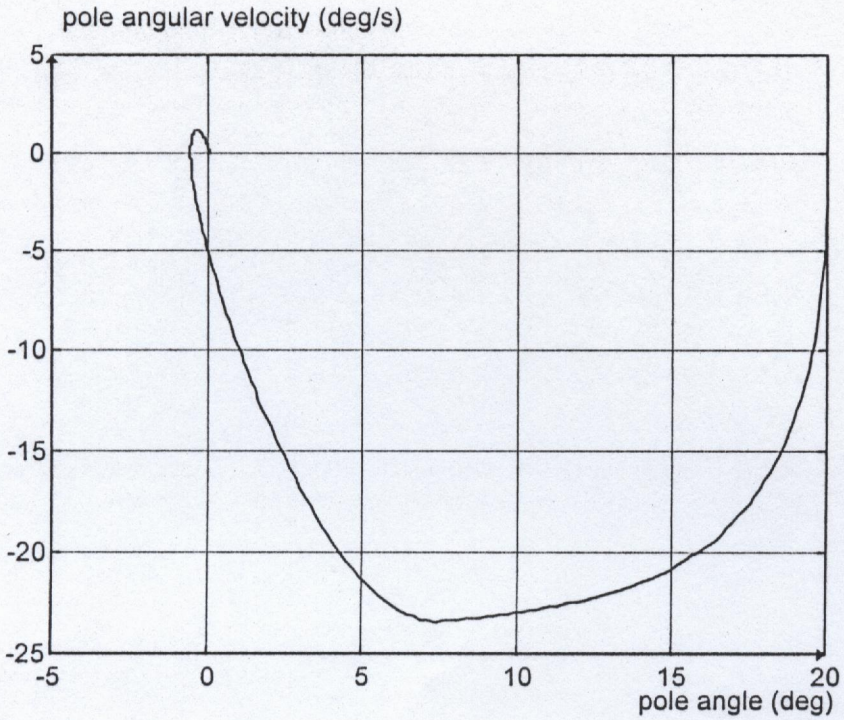


Fig. 5.5.c : plan de phase à partir de (20°,0°/s)

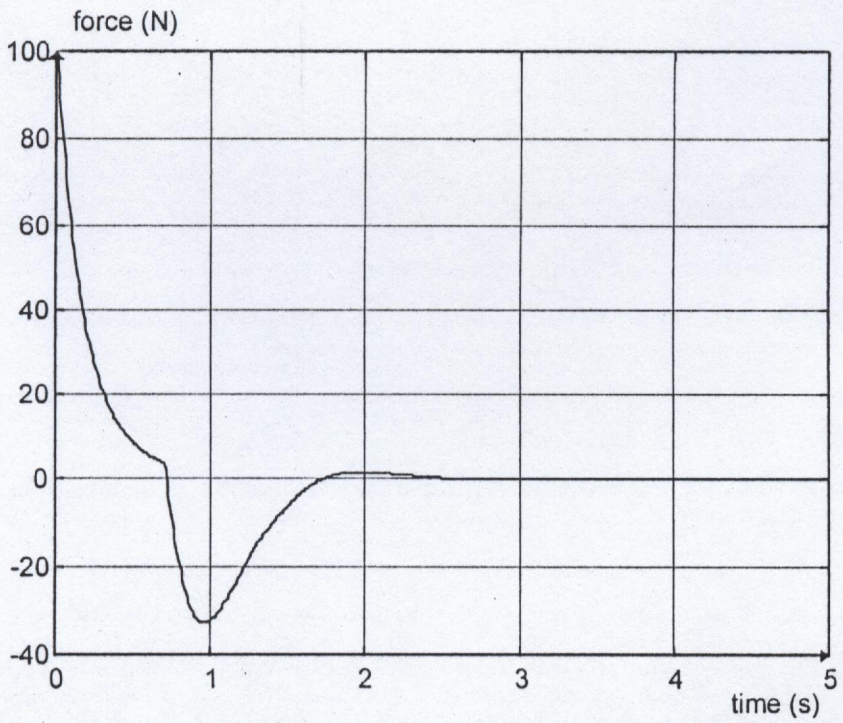


Fig. 5.5.d : Variations de la force

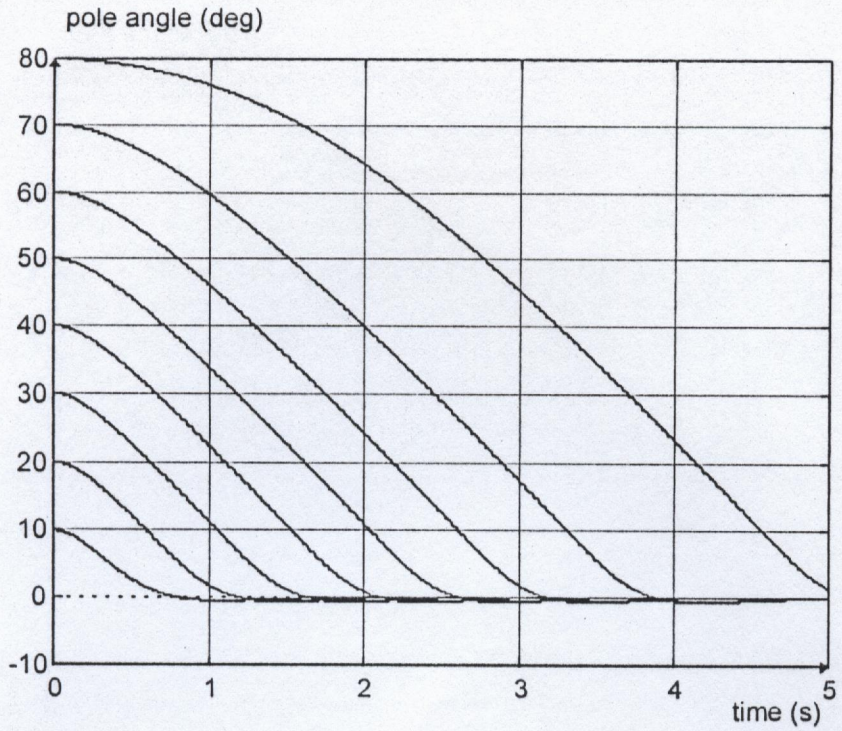


Fig. 5.6.a : Variations de l'angle pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

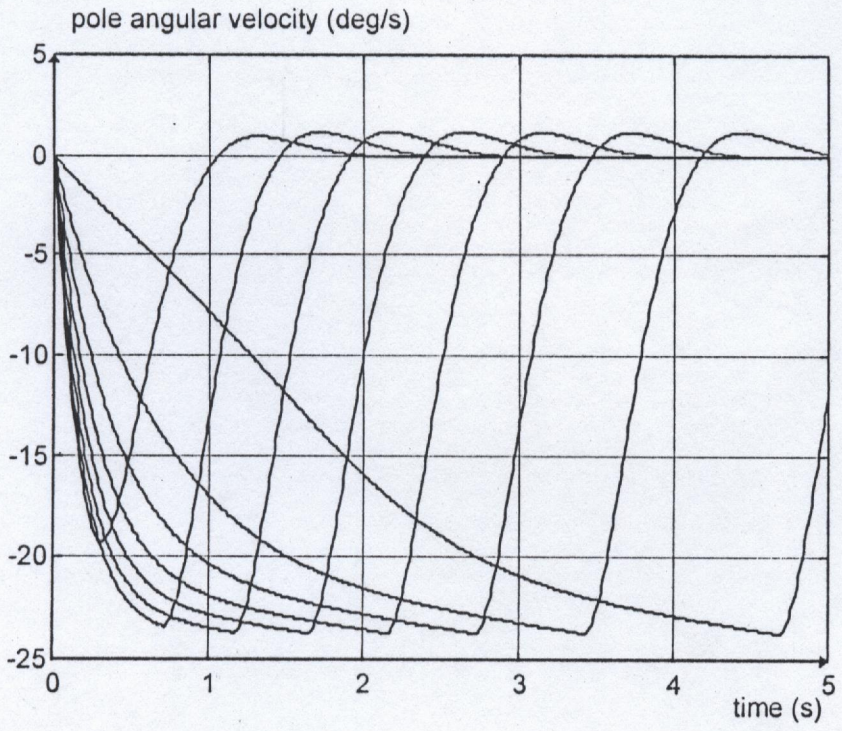


Fig. 5.6.b : Variations de la vitesse angulaire pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

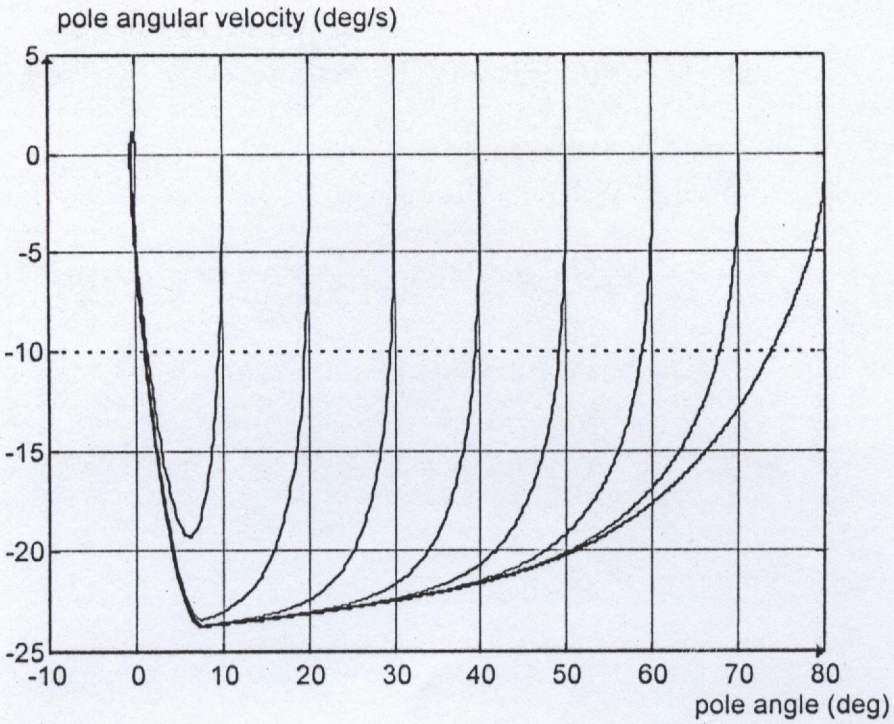


Fig. 5.6.c : Plan de phase pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

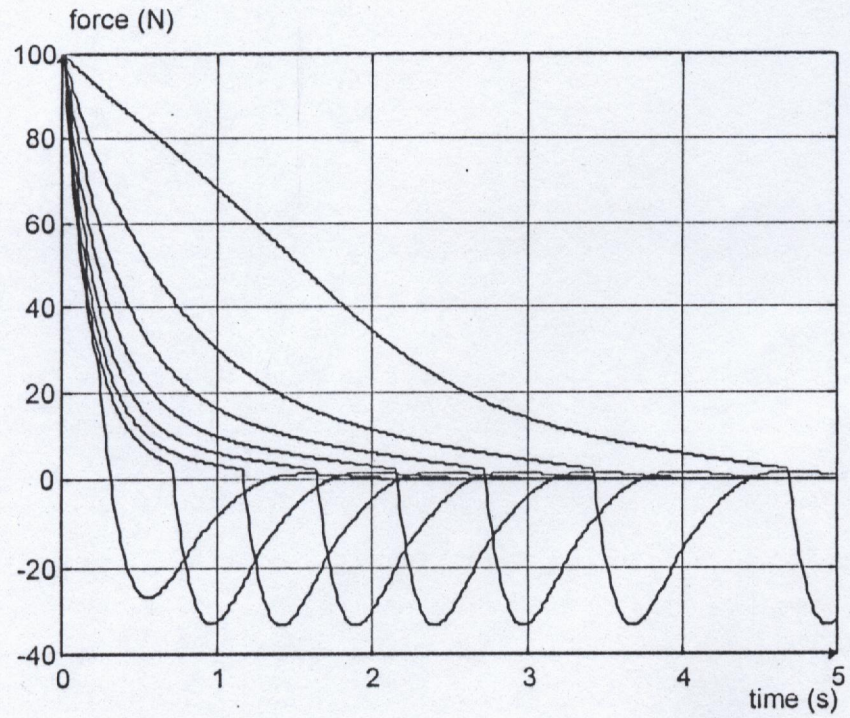


Fig. 5.6.d : Variations de la force pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

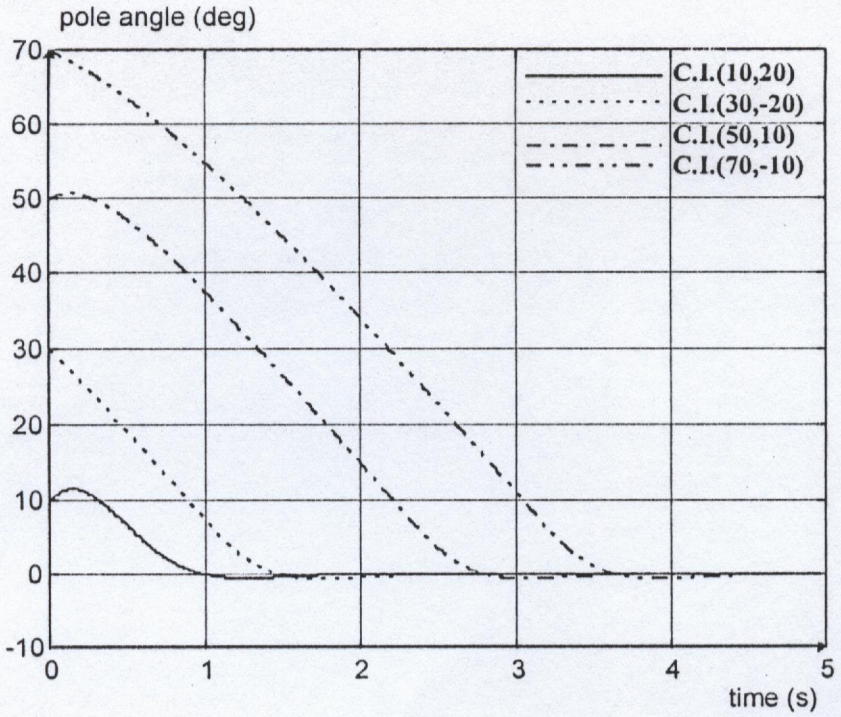


Fig. 5.7.a : Variations de l'angle pour les conditions initiales :
(10 °,20 °/s),(30 °,-20 °/s),(50 °, 10 °/s),(70 °,-10 °/s)

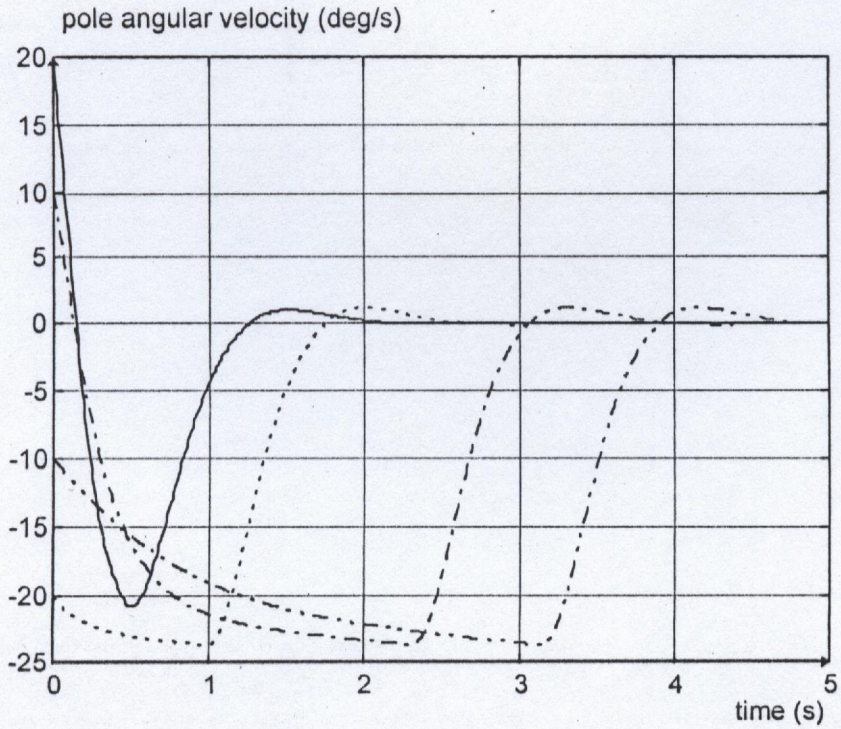


Fig. 5.7.b : Variations de la vitesse angulaire pour les conditions initiales :
(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

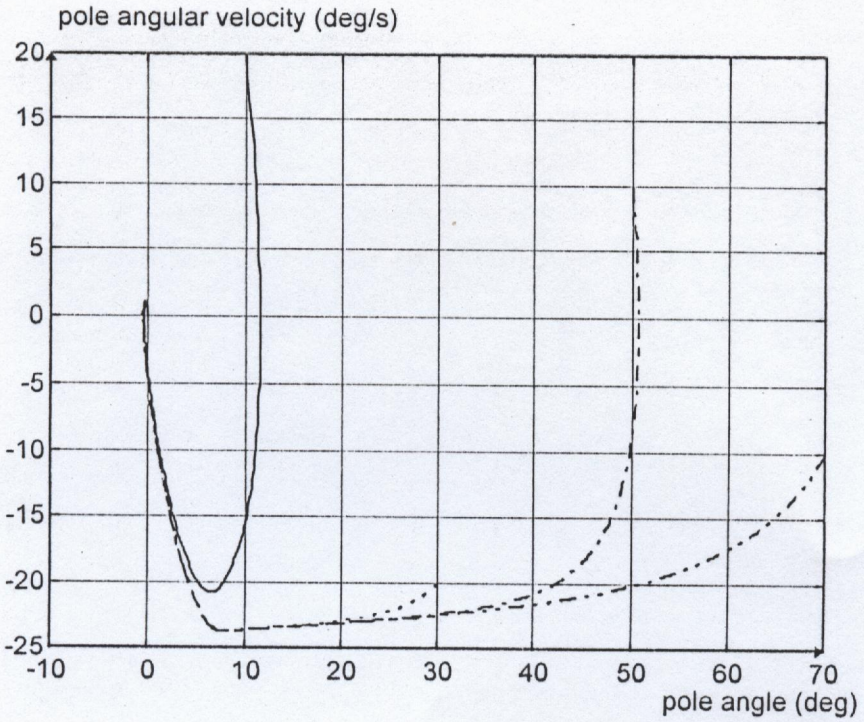


Fig. 5.7.c : Plan de phase à partir des point s :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

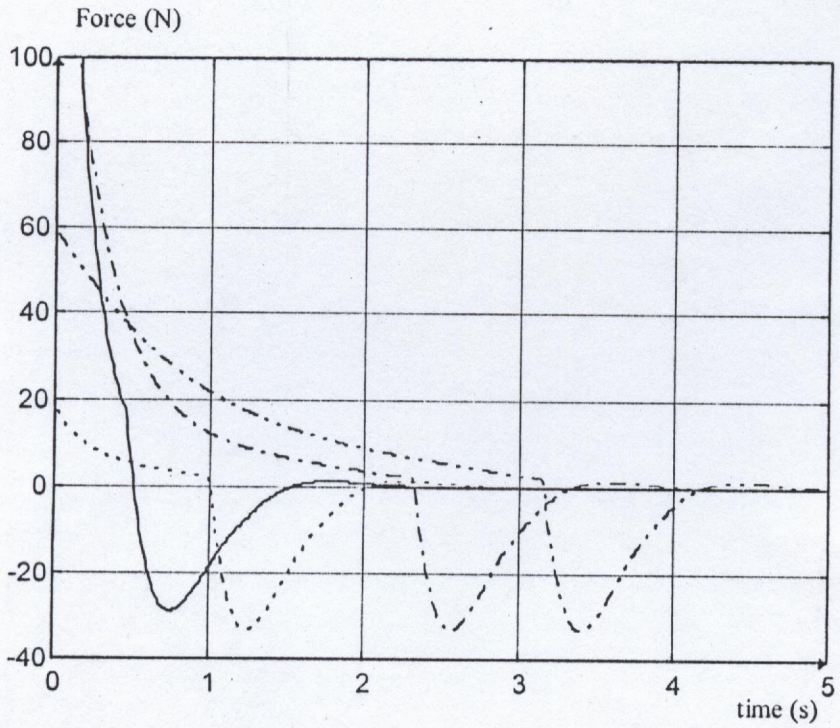


Fig. 5.7.d : Variations de la force à partir des conditions initiales :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

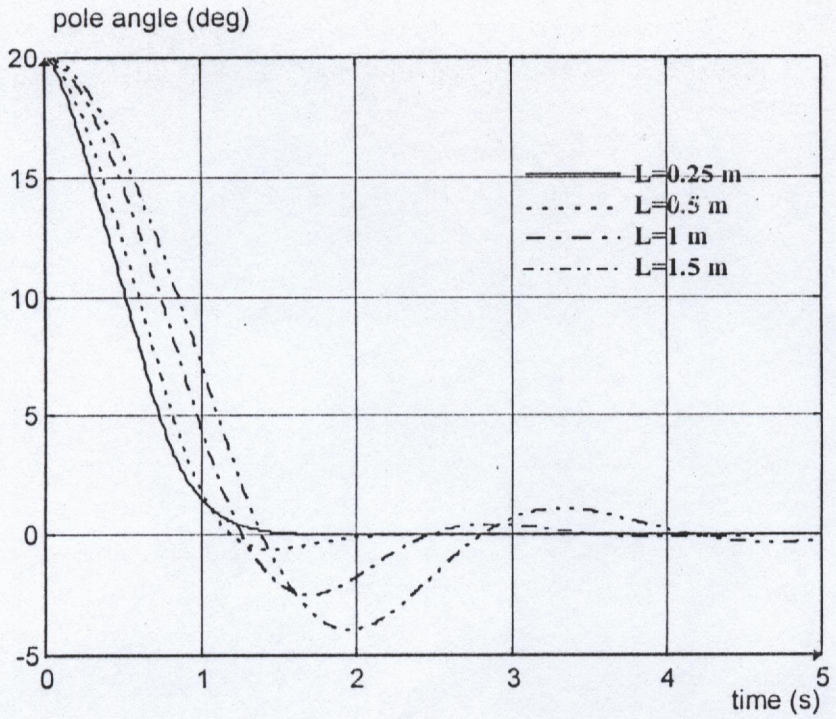


Fig. 5.8.a : Variations de l'angle pour des pendules de longueur variable (0.25,0.5,1 et 1.5m)

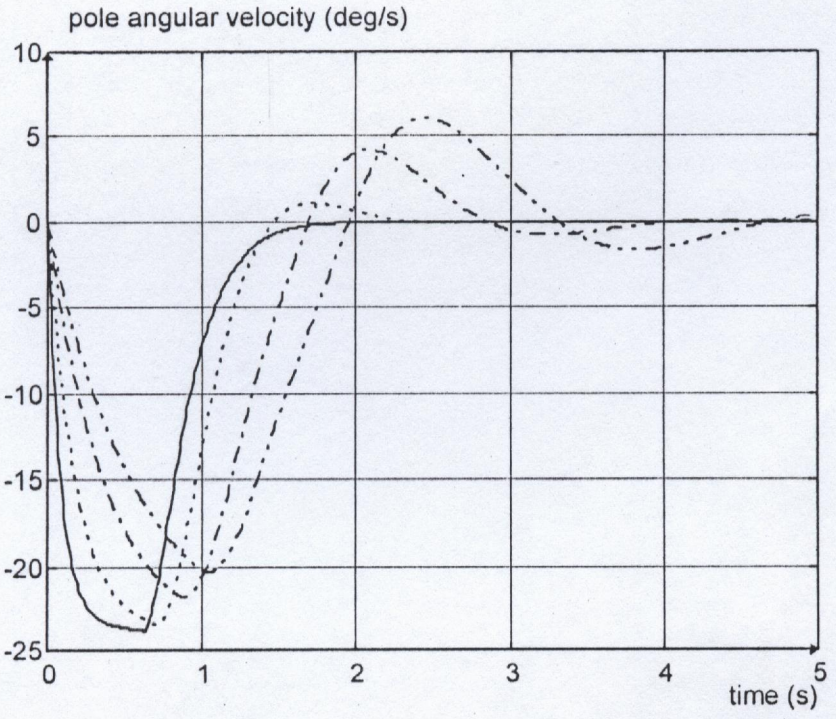


Fig. 5.8.b : Variations de la vitesse angulaire pour des pendules de longueur variable (0.25,0.5,1 et 1.5m)

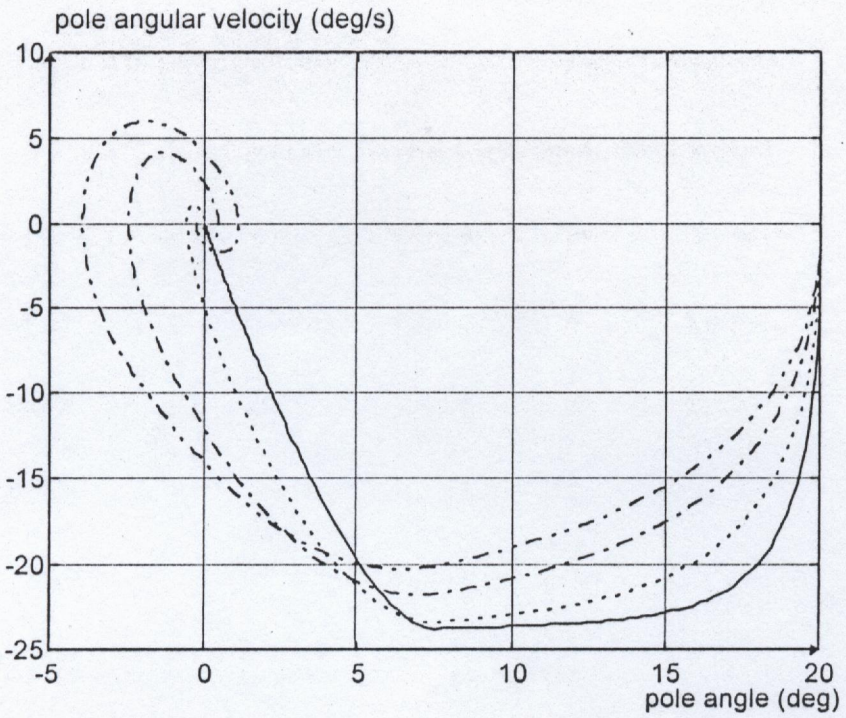


Fig. 5.8.c : Plan de phase pour des pendules de longueur variable

(0.25,0.5,1 et 1.5m)

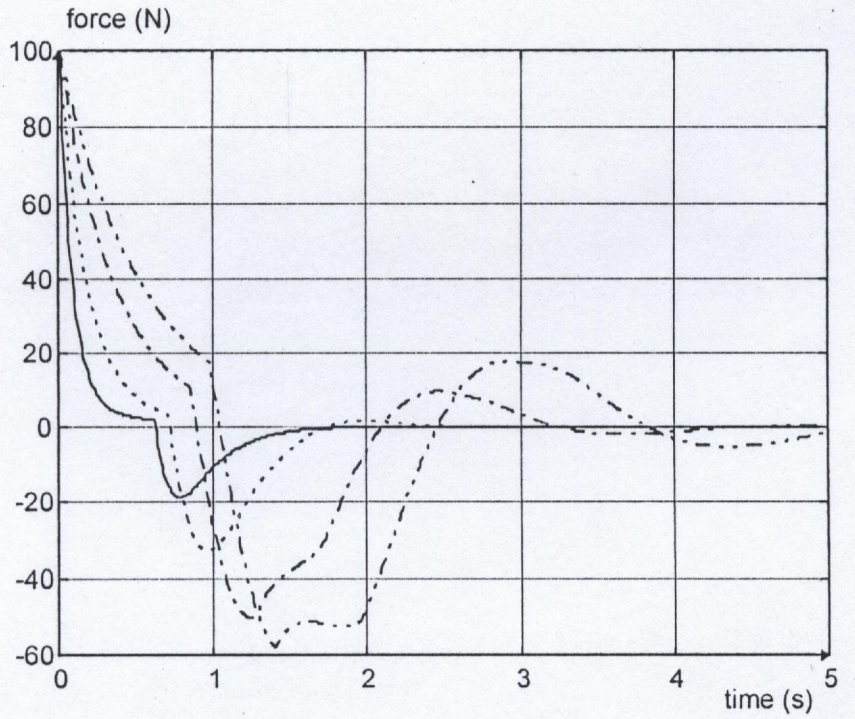


Fig. 5.8.d : Variations de la force pour des pendules de longueur variable

(0.25,0.5,1 et 1.5m)

V.3.4 Contrôleur 555

Dans ce cas, l'angle, la vitesse angulaire et la force sont tous divisés en cinq ensembles flous :NEGATIVE (NE), NEGATIVE MEDIUM (NM), ZERO (ZE), POSITIVE MEDIUM (PM) et POSITIVE (PO), ce contrôleur utilise 25 règles floues.

L'architecture du réseau contrôleur est spécifiée par :

- Nombre de neurones de la couche d'entrée : 2
- Nombre de neurones de la couche de fuzzification : 10
- Nombre de neurones AND : 25
- Nombre de neurones OR : 5
- Nombre de neurones de la couche de défuzzification : 1

Le nombre de poids à optimiser consiste en 125 poids et la recherche des paramètres : $L_\theta, L_{\dot{\theta}}$ et L_F est effectuée selon le tableau (5.4).

Paramètres	Intervalle de recherche
L_θ	[4 120]
$L_{\dot{\theta}}$	[4 130]
L_F	[4 240]

Tab.5.4 : Intervalles de recherche des paramètres pour le contrôleur 555

Les valeurs des paramètres génétiques sont identiques à celles utilisées pour l'optimisation du contrôleur 333, elles sont indiquées dans le tableau (5.2).

Huit bits sont utilisés pour coder chacune des largeurs des différents univers de discours, d'où chaque chaîne complète de la population contiendra 149 bits (24 bits pour les différentes largeurs et 125 bits pour les poids).

V.3.4.1 Résultats de simulation

Pour l'optimisation du contrôleur 555 on a considéré un AG employant pour la reproduction la méthode de sélection à reste stochastique et pour la sélection finale la méthode "steady state selection", ce choix est du aux meilleures résultats obtenues par ces deux méthodes lors de l'optimisation du contrôleur 333.

Les caractéristiques du processus d'apprentissage sont illustrées dans la figure (5.9) qui montre la meilleure fonction d'adaptation et la fonction d'adaptation moyenne pour chaque génération. La valeur maximale de la fonction d'adaptation obtenue est égale à 120.7498 qui est environ deux fois plus grande que celle obtenue par le contrôleur 333.
Le réseau contrôleur résultant est spécifié par les fonctions d'appartenance indiquées dans la figure (5.10) avec des univers de discours de largeur :

- $L_\theta = 10.8235$.
- $L_{\dot{\theta}} = 66.7529$.
- $L_F = 240$.

et par la base des règles floues représentée dans le tableau (5.5).
Les figures (Fig. 5.11.a-d) montrent les variations de l'angle (a), la vitesse angulaire (b)et (d) de la force de t=0 à t=5s. La figure(c) est l'espace d'état indiquant la trajectoire d'approche de l'origine à partir du point initial (20,0).

$\dot{\theta}$ \ θ	NE	NM	ZE	PM	PO
NE	**NE**	**NE**	**NE**	**NE**	**ZE**
NM	**PO**	**NE**	**NE**	**NE**	**PO**
ZE	**NE**	**NE**	**ZE**	**PO**	**PO**
PM	**PM**	**PO**	**PO**	**PO**	**PO**
PO	**ZE**	**NE**	**PO**	**ZE**	**PO**

Tab.5.5 : Base de règles floues du contrôleur 555

Ce contrôleur permet de ramener le pendule à sa position d'équilibre plus rapidement que le contrôleur 333 en un temps inférieur à 1.5s sans oscillations.

V.3.4.2 Robustesse du contrôleur 555

Les même tests de robustesse effectués pour le contrôleur 333 sont considérés pour le contrôleur 555, en effet les figures (Fig. 5.12.a-d) et (Fig. 5.13.a-d) montre la performance du contrôleur optimisé pour des conditions initiales différentes.

On remarque que le contrôleur arrive toujours à stabiliser le pendule et prend moins de 4s pour balancer le pendule avec un angle initial de 80°, mais échoue pour un angle initial supérieur à 85°.

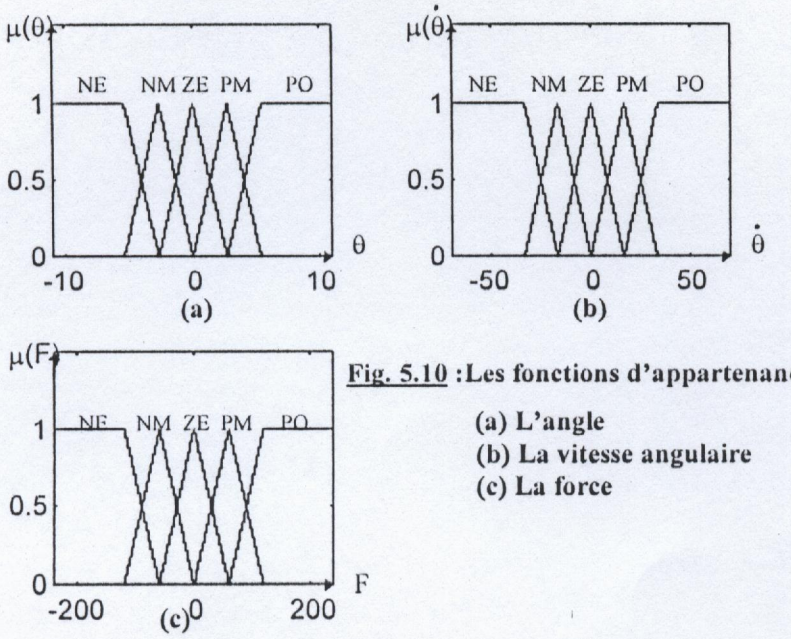


Fig. 5.10 :Les fonctions d'appartenance

(a) L'angle
(b) La vitesse angulaire
(c) La force

Les figures (Fig. 5.14.a-d) montrent la robustesse du contrôleur 555 en considérant des pendules de différentes longueurs (0.25, 0.5, 1 et 1.5m).

D'après les courbes des figures (Fig. 5.12.a-d), (Fig. 5.13.a-d) et (Fig. 5.14.a-d), on constate que la robustesse, mesurée ici par la capacité du contrôleur à ramener le pendule à la verticale pour des conditions initiales variables et pour des pendules de différentes longueurs, s'est nettement améliorée.

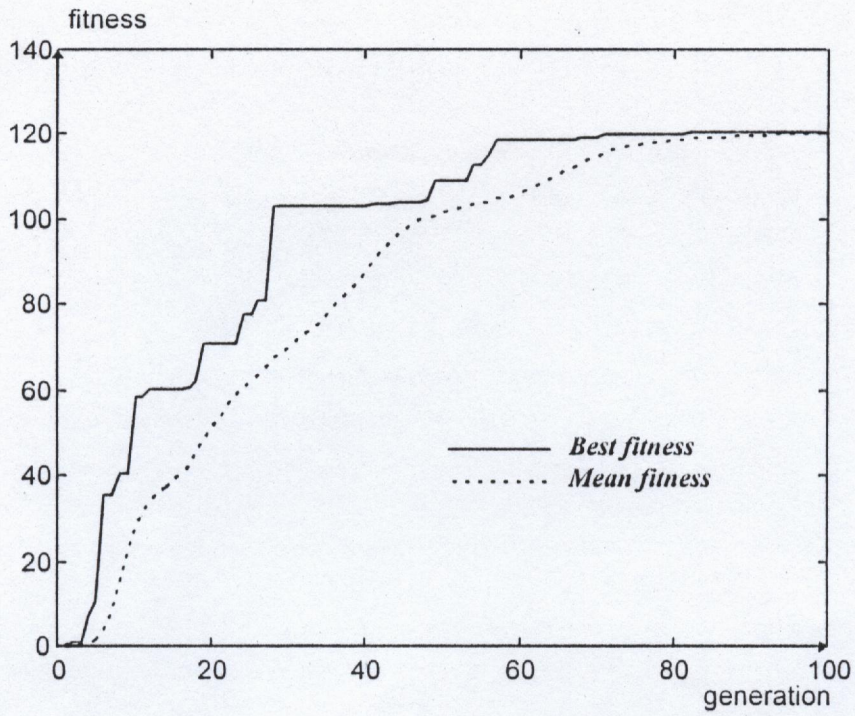


Fig. 5.9

V.4 Conclusion

Dans ce chapitre, l'approche de contrôle que nous avons développé a été testé pour la commande du pendule inversé.

Dans un premier cas, nous avons construit par apprentissage un contrôleur flou à trois variables linguistiques : NE, ZE et PO pour toutes les variables d'E/S. Bien que composé d'un nombre restreint de règles (9 règles floues), ce contrôleur a donné de très bons résultats.

Ensuite nous avons construit un contrôleur à cinq variables linguistiques : NE, NM, ZE, PM et PO qui utilise 25 règles floues de contrôle, la robustesse s'est améliorée mesurée ici par le taux de réussite du contrôleur exprimé par la capacité de ramener le pendule à la verticale pour des conditions initiales variables et pour différentes valeur des paramètres du pendule inversé (longueur du segment).

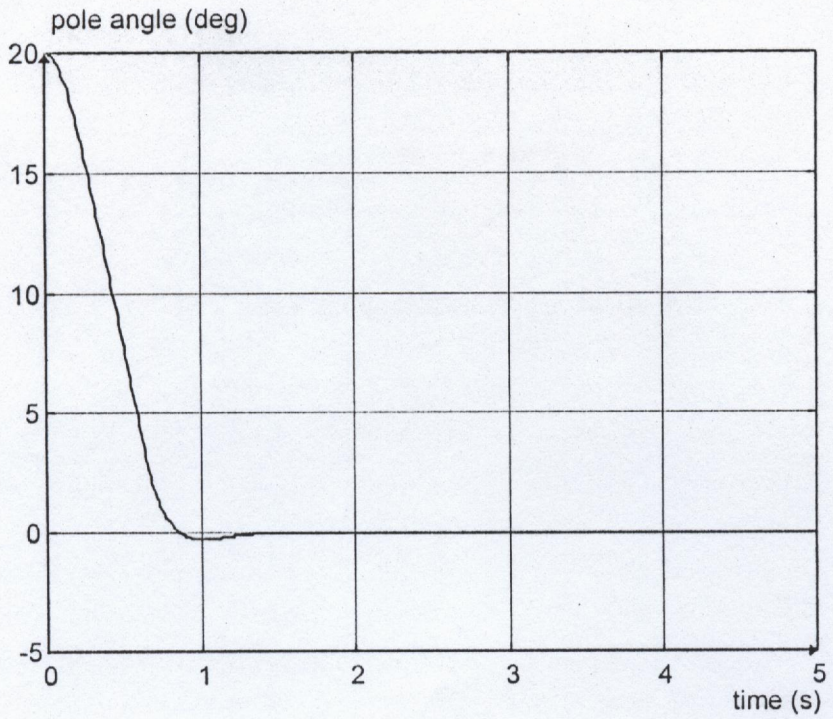


Fig. 5.11.a : Variations de l'angle à partir du point (20°,0°/s)

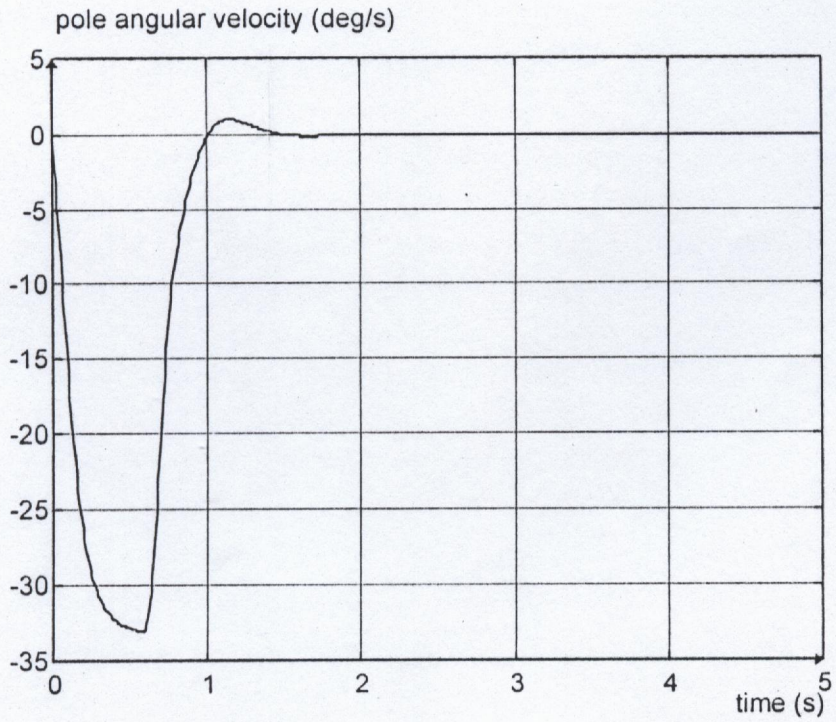


Fig. 5.11.b : Variations de la vitesse angulaire à partir du point (20°,0°/s)

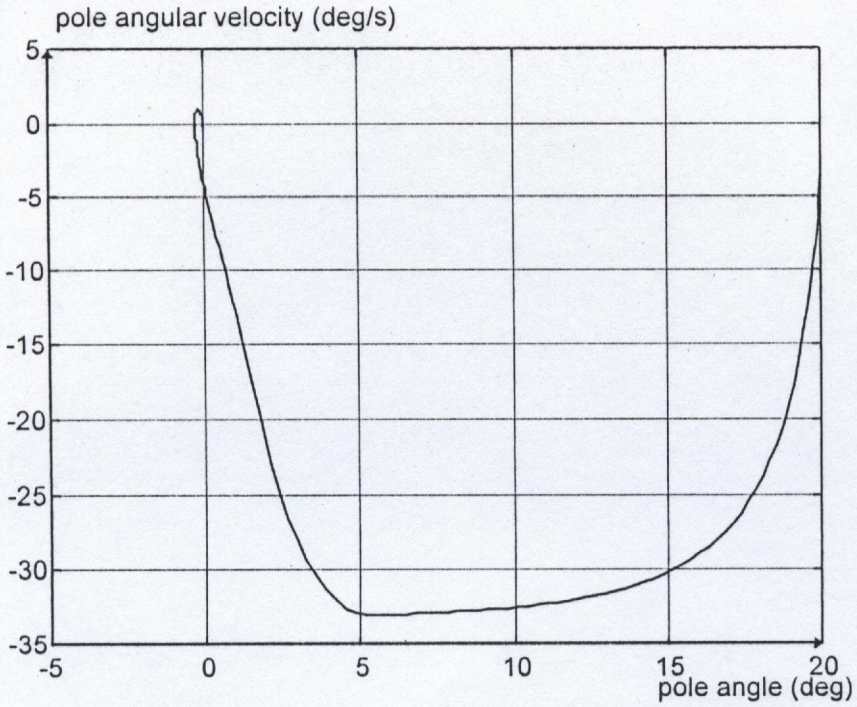


Fig. 5.11.c : plan de phase à partir de (20°,0°/s)

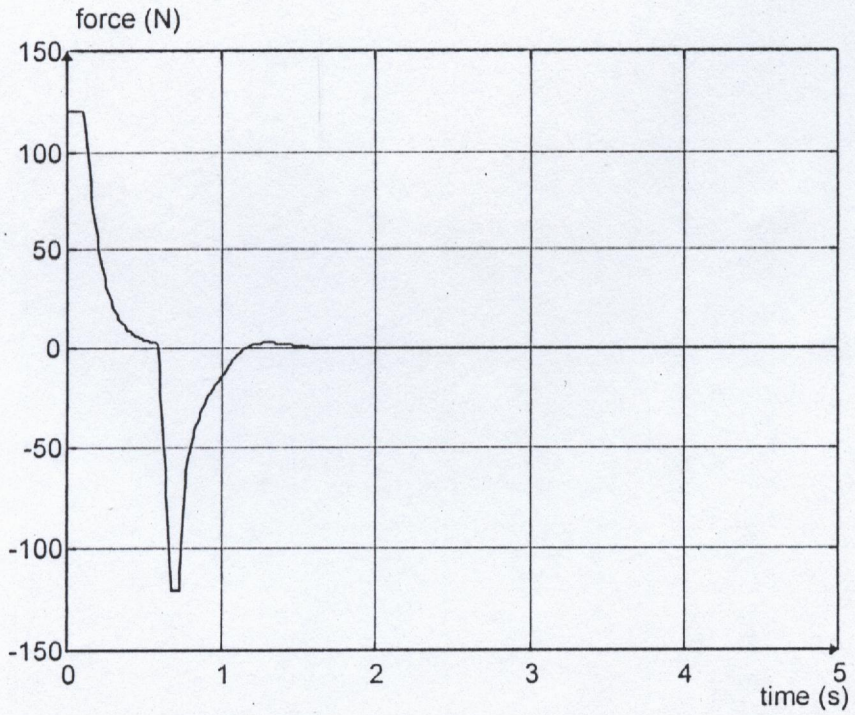


Fig. 5.11.d : Variations de la force

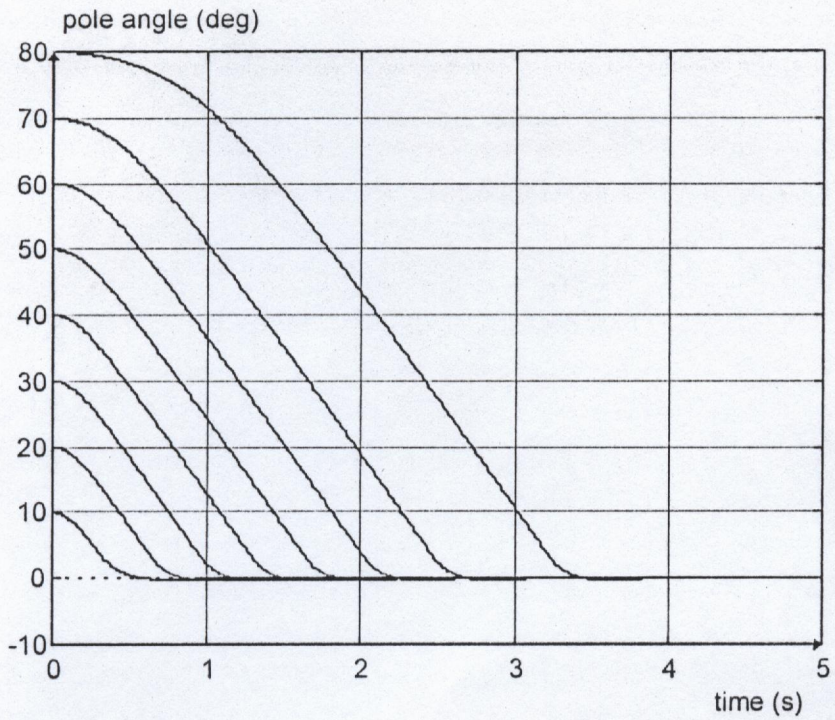


Fig. 5.12.a : Variations de l'angle pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

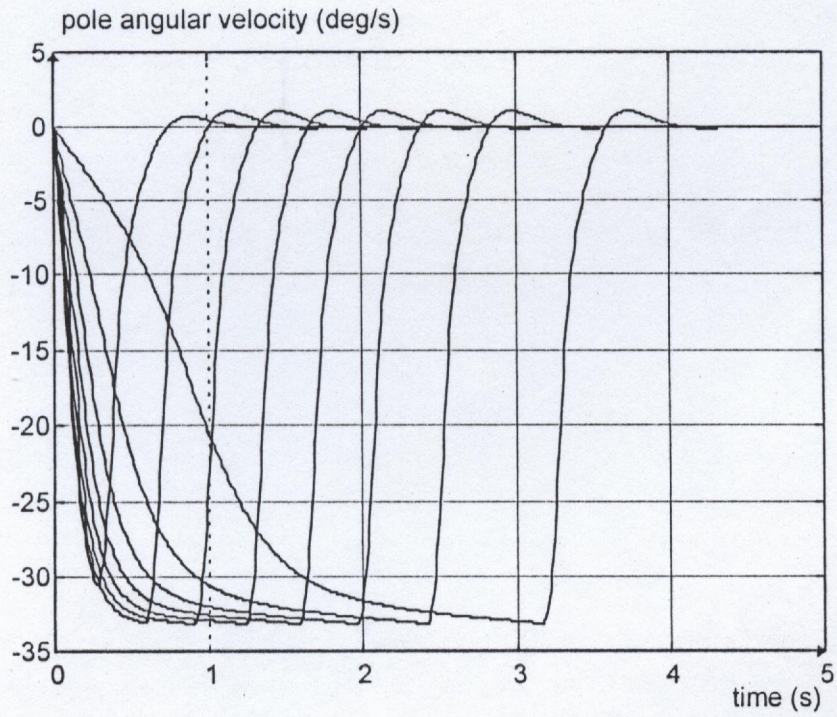


Fig. 5.12.b : Variations de la vitesse angulaire pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

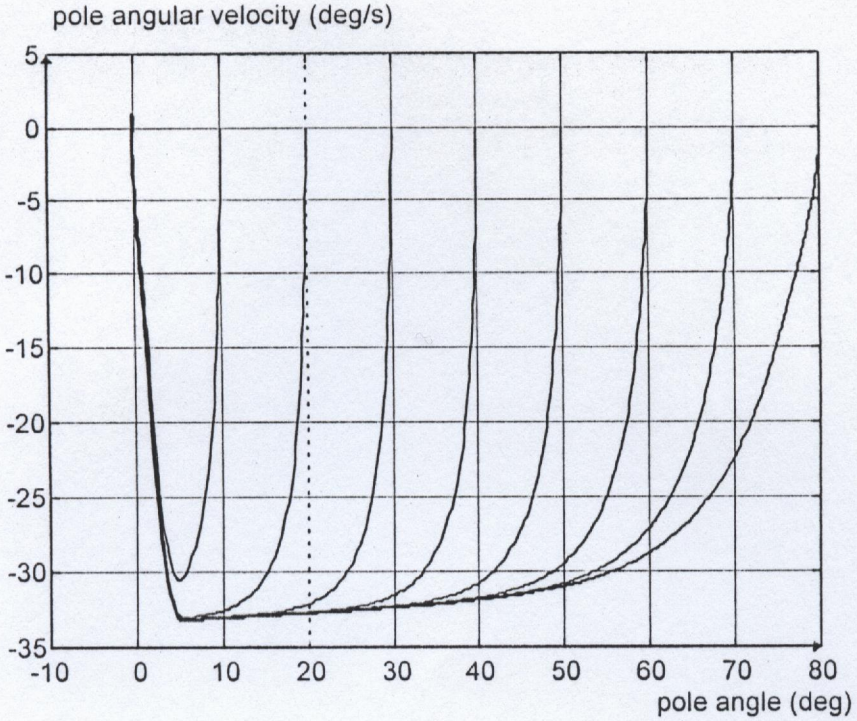


Fig. 5.12.c : Plan de phase pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

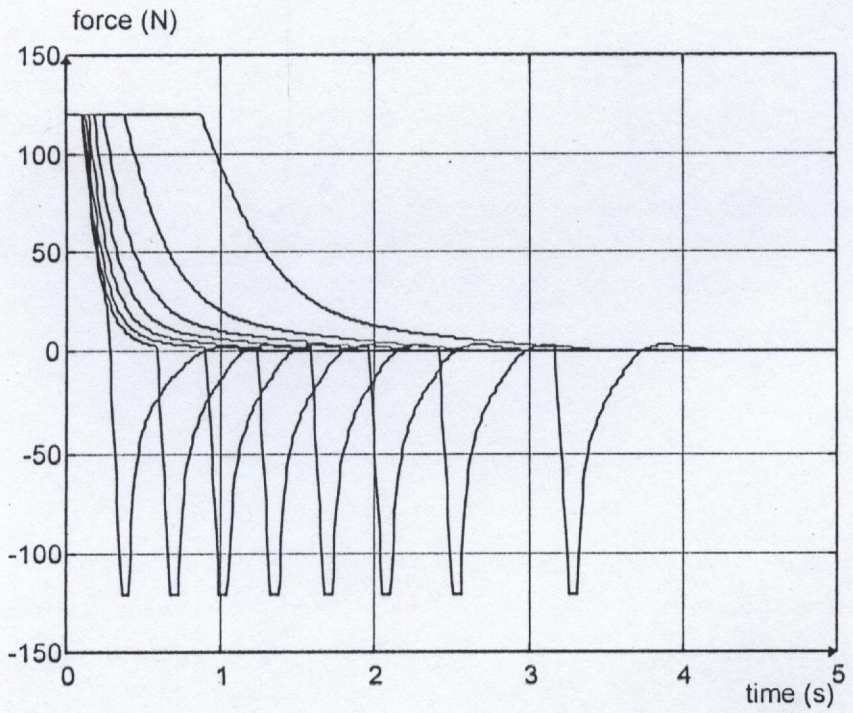


Fig. 5.12.d : Variations de la force pour des conditions initiales différentes

(10,20,30,40,50,60,70 et 80 deg, 0deg/s)

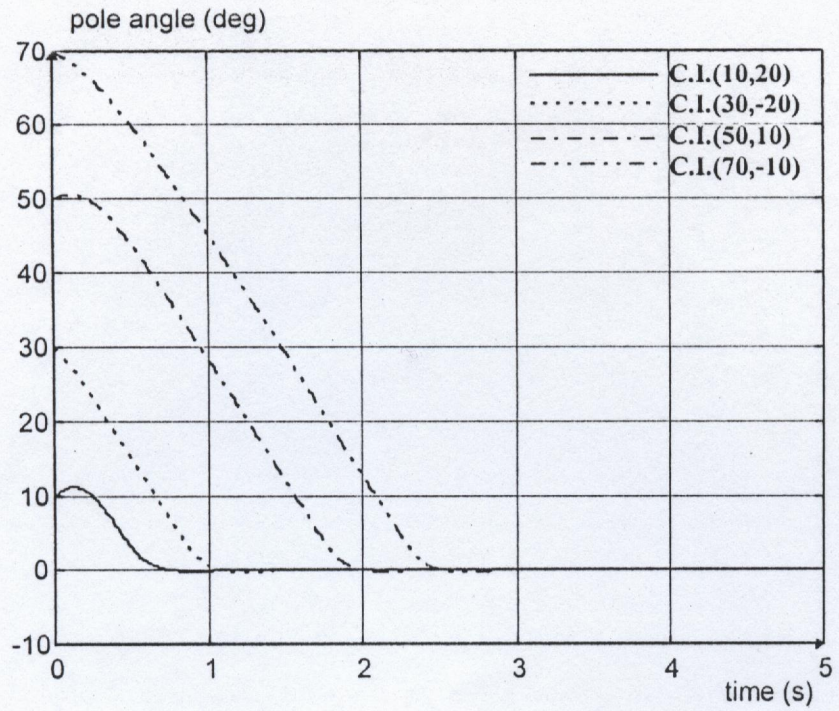


Fig. 5.13.a : Variations de l'angle pour les conditions initiales :

(10 °,20 °/s),(30 °,-20 °/s),(50 °, 10 °/s),(70 °,-10 °/s)

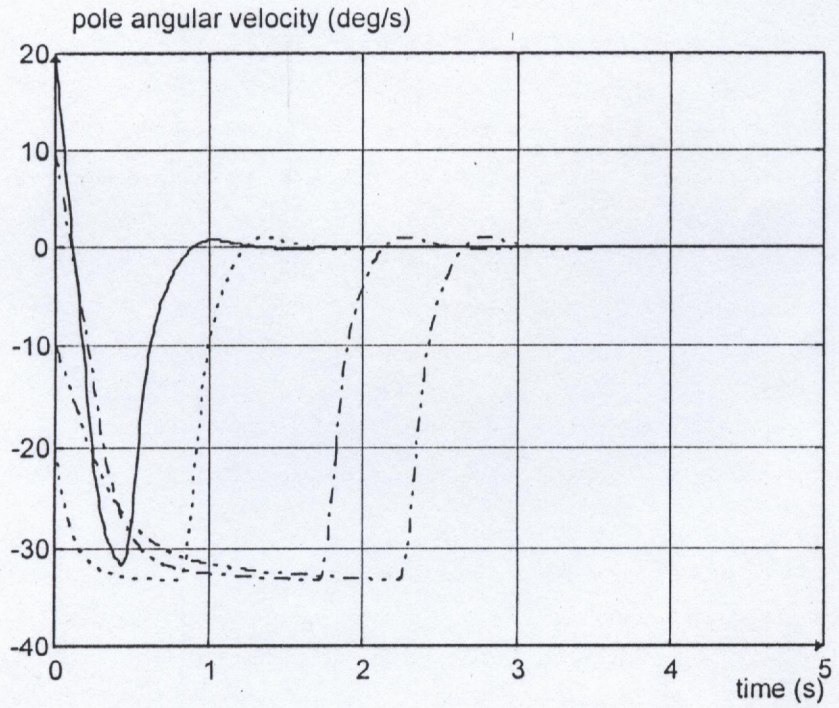


Fig. 5.13.b : Variations de la vitesse angulaire pour les conditions initiales :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

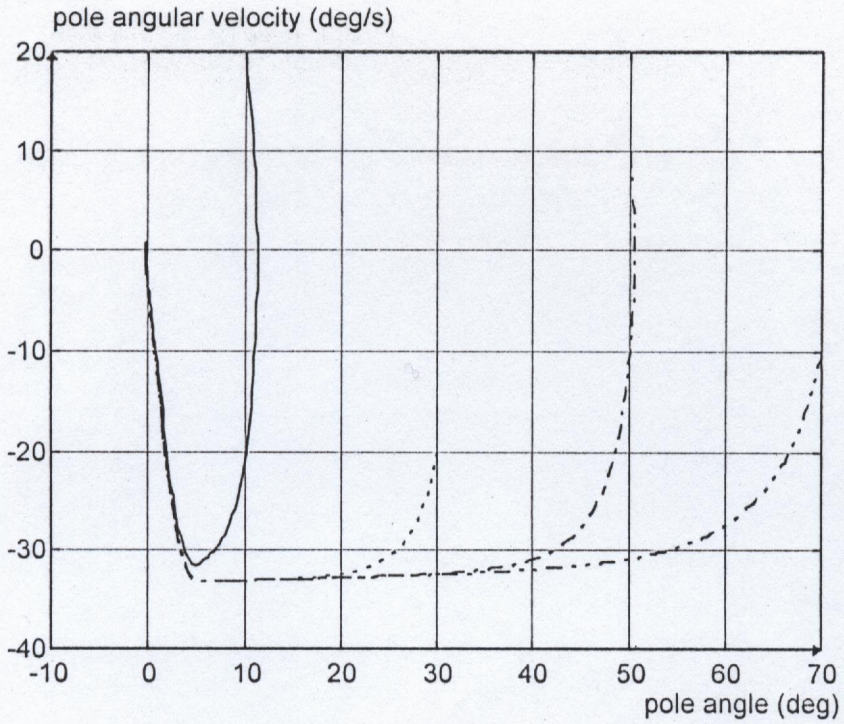


Fig. 5.13.c : Plan de phase à partir des point s :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

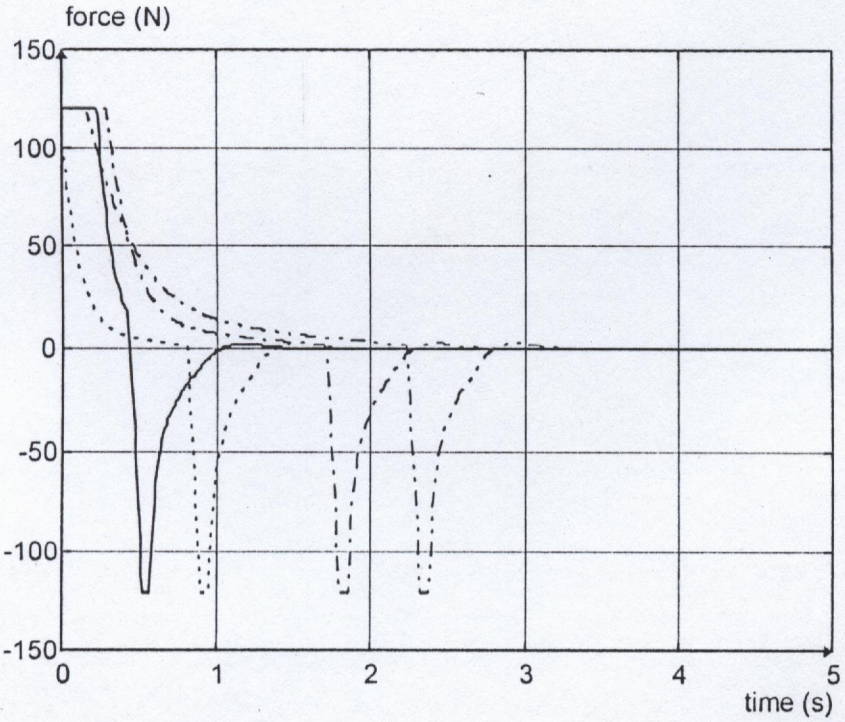


Fig. 5.13.d : Variations de la force à partir des conditions initiales :

(10 °,20 °/s),(30 °,-20 °/s),(50 °,10 °/s),(70 °,-10 °/s)

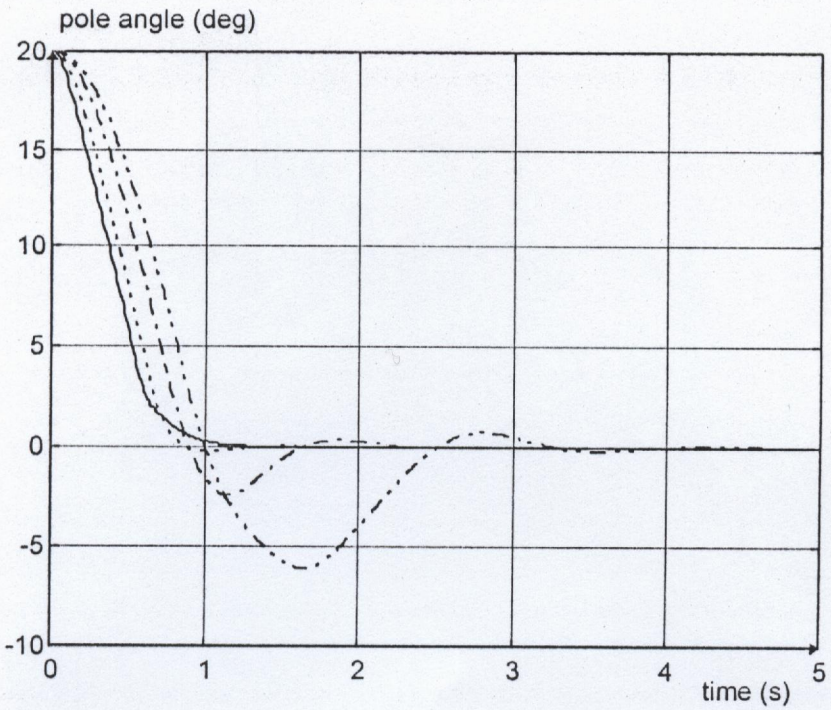


Fig. 5.14.a : Variations de l'angle pour des pendules de longueur variable (0.25,0.5,1 et 1.5m)

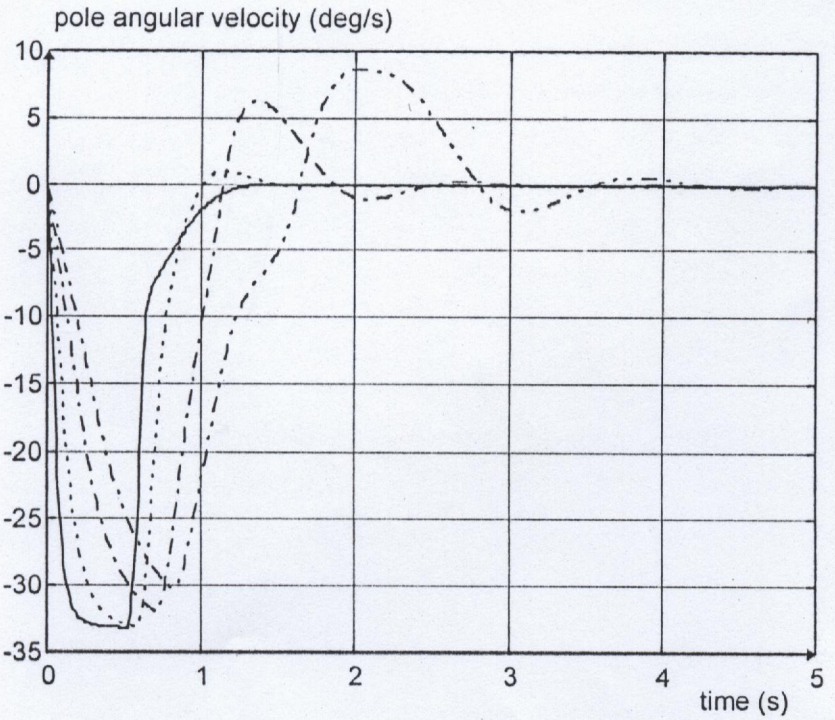


Fig. 5.14.b : Variations de la vitesse angulaire pour des pendules de longueur variable (0.25,0.5,1 et 1.5m)

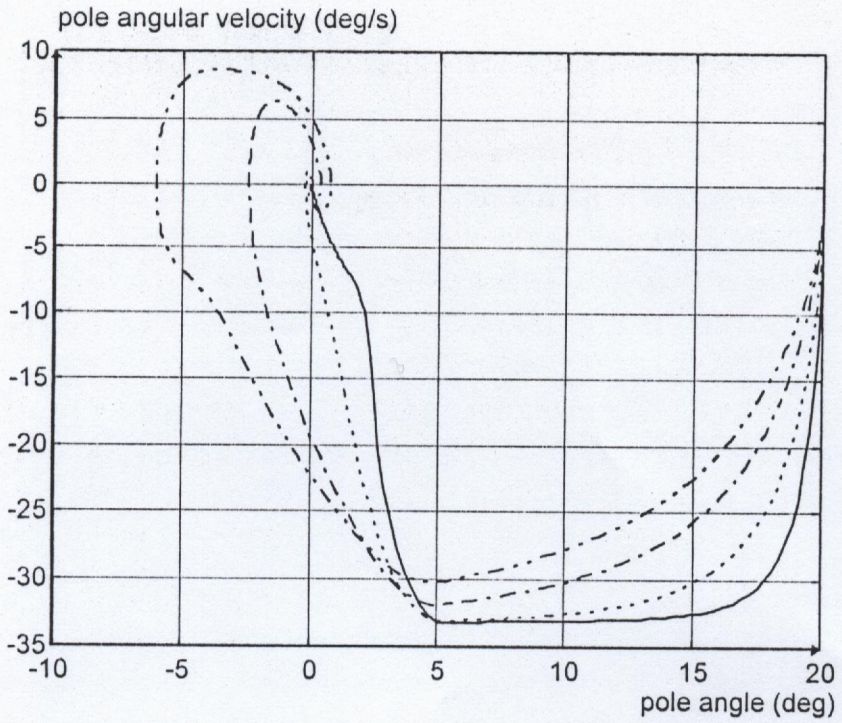


Fig. 5.14.c : Plan de phase pour des pendules de longueur variable
(0.25,0.5,1 et 1.5m)

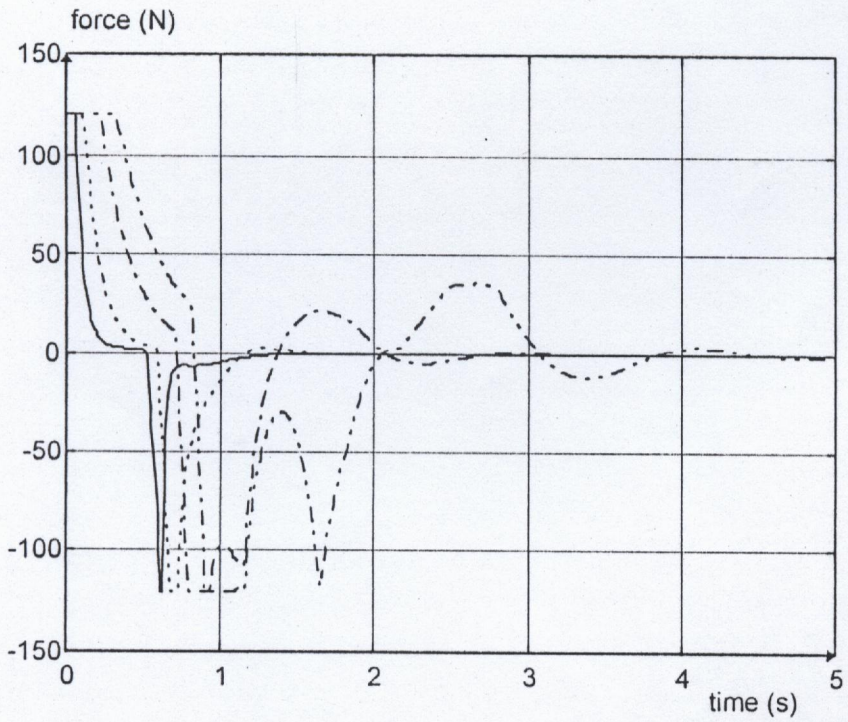


Fig. 5.14.d : Variations de la force pour des pendules de longueur variable
(0.25,0.5,1 et 1.5m)

CONCLUSION

Conclusion

Une nouvelle approche de conception des contrôleurs flous a été développée. C'est une approche mixte qui réside dans la combinaison des deux paradigmes :les réseaux de neurones et les algorithmes génétiques

Dans cette approche on a considéré l'implémentation structurelle d'un système d'inférence flou de type TPE dans un réseau multicouche. Les grandeurs ajustables de ce réseau sont les largeurs des univers de discours et les poids des connexions qui sont assujettis à une certaine contrainte. L'optimisation de ces grandeurs est réalisée par un AG en minimisant une certaine fonction coût. En vu de respecter la contrainte imposée sur les poids les opérations de croisement et de mutation ont été perturbées.

L'algorithme développé a été testé pour la commande du pendule inversé. Dans un premier cas nous avons construit par apprentissage un contrôleur flou à trois variables linguistiques pour toutes les variables d'E/S. Bien que composé d'un nombre restreint de règles (9 règles) ce contrôleur a donné de très bons résultats. Ensuite nous avons construit un contrôleur à cinq variables linguistiques (25 règles),la robustesse s'est améliorée mesurée ici par la capacité de ramener le pendule à la verticale pour des conditions initiales variables et pour différentes valeur des paramètres du pendule. La simulation a été effectuée sous Borland Pascal 7.0.

En vu d'améliorer cette nouvelle méthodologie basée sur un apprentissage hors ligne on propose une direction de recherche intéressante qui consiste à l'ajustement en ligne des paramètres du contrôleur flou. Bien que les AG soient moins adaptés au calcul en temps réel on propose de travailler sur un horizon d'erreur fini et sur quelques générations ce qui permettra de réduire considérablement le temps mis par les AG à converger vers la bonne solution en utilisant aussi d'autres techniques d'aide à la convergence.

Aussi la combinaison de ces deux approches (apprentissage en ligne et hors ligne) présente une technique attractive pour la conception des contrôleurs flous adaptatifs ainsi la méthode hors ligne est utilisée pour générer une base de règles initiale qui peut être modifiée en ligne.

Bibliographie

[1] C.C.Lee , " Fuzzy logic in control systems : Fuzzy logic controller -Part I ", IEEE Trans. Syst. Man Cybern. . Vol. 20 , N°. 2 , Mar. / Apr. 1990 , PP. 404 - 418 .

[2] C.C.Lee , " Fuzzy logic in control systems : Fuzzy logic controller -Part II ", IEEE Trans. Syst. Man Cybern. . Vol. 20 , N°. 2 , Mar. / Apr. 1990 , PP. 419 - 435 .

[3] J.Lu ,W.Jasper and G.K.Lee , " A multivariable self-learning Fuzzy control algorithm for deyeing processes ", Proceedings of the American control conference .Baltimore , Maryland .June 1994 , PP. 983 - 987 .

[4] M.M.Gupta ,J.B.Kiszka and J.M.Trojan , " Multivariable structure of Fuzzy control systems ", IEEE Trans. Syst. Man Cybern. .Vol. SMC-16 ,N° 5 ,Sept. / Oct. 1986 , PP. 638 - 655 .

[5] A.Belmehdi et A. Kara Mostefa ," Conception d'un régulateur auto-ajustable à logique floue pour un bioréacteur anaerobie ", Projet de fin d'étude (Ing) ,Institut d'électronique Constantine ,Juin 1994 .

[6] M.J.Willis ,C.Di Massimo ,G.A.Montague ,M.T.Tham and A.J.Morris ," Artificial neural networks in process engineering " , IEE Proceedings-D ,Vol. 138 ,N° 3 ,May 1991 , PP. 256 - 266 .

[7] T.Low ,T.Lee and H.lim , " A methodology for Neural Network training for control of drives with non linearities " , IEEE Trans. on industrial electronics ,vol.39, N°2 , Apr 1993 , PP. 243 - 249.

[8] J.Tanomaru and S. omatu , " Process control by on-line trained Neural controllers " , IEEE Trans. on industrial electronics, Vol.39 , N°6 , Dec. 1992 , PP.511-521.

[9] T.Fukuda , T.Shibata , M.Tokita and T.Mitsuoka , "Neuromorphic control : adaptation and learning " , IEEE Trans. on industrial electronics , Vol.39 , N°6 , Dec 1992 , PP. 497-503.

[10] H.M.Tai, J.Wang, and K.Ashenayi , " A neural network-based tracking control system" IEEE Trans. on industrial electronics , Vol.39, N°6 , Dec. 1992 , PP. 504-510.

[11] T. Fukuda and T.Shibata , "Theory and applications of neural networks for industrial control systems", IEEE Trans. on industrial electronics , Vol.39 , N°6 , Dec. 1992 , PP. 472-489.

[12] Richard P.Lippmann , "An introduction to computing with neural nets ", IEEE ASSP Magazine Vol.4 , April 1987 , PP. 4-22.

[13] P. Melsa, "Neural networks : a conceptual overview , "Technical report T.R.C.89-08.Tellabs Research center ,Mishawaka ,Aug. 1989 .

[14] S. K.Pal and S.Mitra , "Multilayer perceptron, Fuzzy sets, and classification ", IEEE Trans. on neural networks , Vol.3 , N°5 , Sept 1992 , PP. 683-697.

[15] J. R.Jang , "Self learning Fuzzy controllers based on temporal back propagation",IEEE Trans. on neural networks ,Vol.3 ,N°5, Sept 1992, PP. 714-723

[16] H. R.Berenji and P.Khedkar, "Learning and Tuning Fuzzy logic controllers through reinforcements",IEEE Trans. on neural networks, Vol.3, N°5, Sept1992, PP.724-740.

[17] A.Bachtarzi , " Commande des systèmes à structures variables (applications à la poursuite de trajectoires) ", Thèse de Magister ,institut d'électronique ,Constantine, 1995 .

[18] A.U.Levin and K.S.Narendra , "Control of Non linear dynamical systems using neural networks : controllability and stabilization ", IEEE Trans. on Neural Networks , Vol. 4 , N° 2 March 1993 , PP. 192 - 206 .

[19] A.U.Levin and K.S.Narendra , "Control of Non linear dynamical systems using neural networks - Part II : observability ,identification and control ", IEEE Trans. on Neural Networks , Vol. 7 , N° 1 January 1996 , PP. 30 -42 .

[20] B.Mendil , "Contrôle neuronal flou ", Thèse de Magister ,institut d'électronique , Setif , 1994 .

[21] C.Wang ,W.Wang , T.Lee and P.S Tseng,"Fuzzy B-spline membership function (BMF) and its applications in Fuzzy-neural control ", IEEE Trans. on Systems ,Man,and Cybernetics ,Vol. 25 ,N° 5 , may 1995 ,PP. 841 -851 .

[22] Y.Jin ,J.Jiang and J.Zhu ,"Neural network based Fuzzy identification and its application to modeling and control of complex systems ", IEEE Trans. on Systems ,Man,and Cybernetics ,Vol. 25 ,N° 6 ,June 1995 ,PP. 990 -997 .

[23] J.Zhang and A.Julian Morris,"Process fault diagnosis using Fuzzy neural networks ", Proceedings of the American Control Conference,Baltimore,Maryland ,June 1994 PP. 971-975 .

[24] Syed I.Ashon ,"Petri net models of Fuzzy neural networks ", IEEE Trans. on Systems , Man,and Cybernetics ,Vol. 25 ,N° 6 ,June 1995 ,PP. 926 -932 .

[25] J.J.Buckley and Y.Hayashi ,"Fuzzy neural networks : a survey ",Fuzzy Sets and Systems , 66 ,1994 ,PP. 1-13 .

[26] S.K.Halgamuge ,M.Glesner ,"Neural networks in designing Fuzzy systems for real world applications ", Fuzzy Sets and Systems, 65 ,1994 ,PP. 1-12 .

[27] E.Ikonen and K.Najim ,"Fuzzy neural networks and application to the FBC process " , to appear in IEE Proceedings on control theory and applications.

[28] W.Pedrycz ,"Genetic algorithms for learning in Fuzzy relational structures ", Fuzzy Sets and Systems, 69 ,1995 ,PP. 37-52 .

[29] David E.Goldberg ,"Algorithmes génétiques exploration ,optimisation et apprentissage automatique ",Edition Addison-Wesley 1994 .

[30] J.Kim and B.P.Zeigler ,"Designing Fuzzy logic controllers using a multiresolutional search paradigm ",IEEE Trans. on Fuzzy Systems , Vol. 4 ,No 3 , august 1996 , PP. 213-226 .

[31] C.L.Karr and E.J.Gentry ,"Fuzzy control of pH using genetic algorithms ",IEEE Trans. on Fuzzy Systems ,Vol. 1 ,N° 1 ,February 1993 , PP. 46-53 .

[32] A.Homaifar and Ed McCormick ,"Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms ",IEEE Trans. on Fuzzy Systems, Vol. 3 ,N° 2 ,May 1995 , PP. 129-139 .

[33] D.A.Linkens and H.O.Nyongesa ,"Genetic Algorithms for Fuzzy control-Part 1 :Off-line system development and application ",IEE Proceedings-Control Theory and applications Vol. 142 ,N° 3 ,May 1995 , PP. 161-176 .

[34] D.A.Linkens and H.O..Nyongesa ,"Genetic Algorithms for Fuzzy control-Part 2 :Online system development and application ",IEE Proceedings-Control Theory and applications Vol. 142 ,N° 3 ,May 1995 , PP. 177-185 .

[35] Y.H.Joo ,H.S.Hwang ,K.B.Kim and K.B.Woo ,"Fuzzy system modeling by Fuzzy partition and GA hybrid schemes ",Fuzzy Sets and Systems ,86,1997, PP. 279-288 .

[36] C.T.Lin and Y.C.Lu ,"A neural Fuzzy system with Fuzzy supervised learning ",IEEE Trans. on Syst. Man ,and Cybernetics-Part B :cybernetics.Vol. 26 ,N° 5 ,October 1996 , PP. 744-763 .

[37] F.R.Rubio, M. Berenguel and E.F.Camacho ,"Fuzzy logic control of a solar power plant ",IEEE Trans. on Fuzzy Systems ,Vol. 3 ,N° 4 ,November 1995 , PP. 459-468 .

[38] K.Mesghouni ,"Application des algorithmes génétiques à la commande des MAS - investigation dans le domaine de la commande adaptative ",Rapport de DEA de génie électrique ,CEGELY Lyon ,France .

[39] D.A.Linkens and H.O..Nyongesa ,"Learning systems in intelligent control : an appraisal of Fuzzy ,neural and genetic algorithm control applications ",IEE Proceedings-Control Theory and applications ,Vol. 143 ,N° 4 ,July 1996 , PP. 367-386 .

[40] B.Sareni ,"Algorithmes génétiques standards ",Rapport interne R-97-0.1 ,Septembre 1997 ,CEGELY Lyon , France .

[41] S.Horikawa ,T.Furuhashi and Y.Uchikawa ,"On Fuzzy modeling using Fuzzy neural networks with the back-propagation algorithm ",IEEE Trans. Neural Networks , Vol. 3, N° 5, Sept. 1992 .

[42] C.T.Lin and C.S.G.Lee ,"Real-time supervised structure/parameter learning for Fuzzy neural network ",IEEE International Conference on Fuzzy Systems ,San Diego ,USA, 1992 , PP. 1283-1291 .

[43] C.T.Lin and C.S.G.Lee ,"reinforcement structure/parameter learning for neural network based Fuzzy logic control systems ",Proceedings of IEEE international conference on Fuzzy systems ,San Francisco ,CA,USA,1993 , PP. 88-93 .

[44] L.X.Wang and J.M.Mendel ,"Fuzzy basis functions ,universal approximation and orthogonal least squares ",IEEE Trans. on Neural Networks,1992 , PP. 807-814 .

[45] S.Horikawa,T.Furuhashi,S.Ouma and Y.Uchikawa ,"A Fuzzy controller using a neural network and its capability to learn expert's control rules ",Proceedings of international conference on Fuzzy logic and neural networks ,1990 ,Vol. 1, PP. 103-106 .

[46] W.Pedrycz ,"Fuzzy neural networks and neurocomputations ",Fuzzy Sets and Systems, 1993 , 56,PP. 1-28 .

[47] H.Ishibushi,H.Okada and H.Tanaka ,"Fuzzy neural networks with Fuzzy weights and Fuzzy biases ",Proceedings of int. Joint.conf. on Neural networks ,San Francisco CA, 1993 , PP. 1650 - 1655 .

[48] Y.Hayashi,J.J.Buckley and E.Czogala ,"Systems engineering applications of Fuzzy neural networks ",Proc. of internat.joint conf. on Neural Networks ,Baltimore ,MD,1992 Vol. 2 ,PP. 412-418 .

[49] P.Thrift ,"Fuzzy logic synthesis with genetic algorithms ",Proc. of the fourth internat. conf. On Genetic Algorithms ,1991 , PP. 509-513 .

[50] C.L.Karr ,"Design of an adaptive Fuzzy logic controller using a genetic algorithm", Proc. of the fourth Int. conf. on Genetic Algorithms ,1991 ,PP. 450-457 .

[51] H.Nomura,I.Hayashi and N.Wakami ,"A self tuning method of Fuzzy reasoning by genetic algorithm ",Proc. Int. Fuzzy syst.intell.contr.conf. ,1992 ,PP. 236-245 .

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE L

A RECHERCHE SCIENTIFIQUE

----------o----------

UNIVERSITÉ FERHAT ABBAS

SÉTIF

MÉMOIRE

Présenté par

TITEL FAOUZI

Pour obtenir le titre de **Magister**

de l'Institut d'Électronique

Option

CONTROLE