REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'EDUCATION NATIONALE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT **GENIE ELECTRIQUE**
OPTION **AUTOMATIQUE**

PROJET DE FIN D'ETUDES

SUJET

ETUDE DE LA COMMANDE NEURO-LINGUISTIQUE APPLICATION SISO AUX SYSTEMES LINEAIRE ET NON LINEAIRE ET A UN SYSTEME MIMO : BRAS DE ROBOT

PROPOSE PAR
M. C. SOUAMI

ETUDIE PAR
A. MADANI
Y. OUSSAR

DIRIGE PAR
M. C. SOUAMI

PROMOTION
JUIN 1993

E.N.P 10, Avenue Hacen Badi El-Harrach . ALGER

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'EDUCATION NATIONALE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT **GENIE ELECTRIQUE**
OPTION **AUTOMATIQUE**

PROJET DE FIN D'ETUDES

SUJET

ETUDE DE LA COMMANDE NEURO-LINGUISTIQUE APPLICATION SISO AUX SYSTEMES LINEAIRE ET NON LINEAIRE ET A UN SYSTEME MIMO : BRAS DE ROBOT

PROPOSE PAR
M. C. SOUAMI

ETUDIE PAR
A. MADANI
Y. OUSSAR

DIRIGE PAR
M. C. SOUAMI

PROMOTION
JUIN 1993

D E D I C A C E S

A ma mère.

A mon père.

A mes frères et soeurs.

A ceux qui me sont chers.

Abou El-Kacem.

A ma mère

Pour son Amour.

A mon père

Pour ses Conseils.

A ma soeur

Pour son Assistance.

A mon oncle Abdenour

Pour ses Encouragements.

Yacine.

R E S U M E

Le présent projet consiste en l'étude d'une association de la commande linguistique et des réseaux de neurones multicouches pour la synthèse d'un contrôleur neuro-linguistique alliant les avantages de ces deux techniques.

L'application de ce contrôleur à la commande de deux systèmes monovariables l'un linéaire et le second non linéaire ainsi qu'à un système multivariable non linéaire constitué par un bras de robot manipulateur à trois degrés de liberté permet la mise en évidence de la puissance de commande du contrôleur neuro-linguistique.

REMERCIEMENTS

Nous tenons à remercier tout particulièrement notre promoteur **M. C. SOUAMI** qui n'a pas ménagé son temps pour nous apporter toute son aide. Ses judicieuses remarques et ses précieux conseils et directives nous ont permis de mener à bien ce travail.

Nous exprimons notre reconnaissance aux responsables de la société **CONDOR ENGINEERING** qui ont mis gracieusement à notre disposition leur puissant moyen de calcul sans lequel notre travail n'aurai pas pu être mené à bon terme.

Nous tenons à marquer notre gratitude envers tous nos enseignants qui nous ont dispensé non seulement leur savoir mais également aide et encouragements.

S O M M A I R E

Page

INTRODUCTION. 1

CHAPITRE I : Présentation des réseaux de neurones.

I.1. Introduction 5
I.2. Qu'est-ce qu'un réseau de neurones 6
I.3. Différents types de réseaux 8
I.4. Apprentissage d'un réseau 9
I.5. Classification des réseaux neuronaux suivant l'architecture et le type d'apprentissage. 11
I.6. Les niveaux de réseaux de neurones. 13

CHAPITRE II : Backpropagation.

II.1. Introduction 17
II.2. Modèle du neurone et du réseau 18
II.3. Présentation de l'algorithme Backpropagation 20
II.4. Problèmes posées par Backpropagation 23

CHAPITRE III : Backpropagation et la commande des systèmes.

III.1. Introduction 31
III.2. Une structure générale de commande 31
III.3. Apprentissage du correcteur feedforward 32
III.4. Avantages et limites du correcteur feedforward 34
III.5. Apprentissage du correcteur feedback 35
III.6. Conclusion. 36

CHAPITRE IV : Notion de contrôleur flou.

IV.1. Introduction 40
IV.2. Definitions 40
IV.3. Le contrôleur flou 42
IV.4. Mise en oeuvre du contrôleur flou 43
IV.5. Conclusion 46

CHAPITRE V : Contrôleur Neuro-Linguistique.

V.1. Motivation 49
V.2. Réalisation de la partie linguistique 49
V.3. Choix de la loi de commande 50
V.4. Architecture du réseau neuronal 54
V.5. Structure de réglage 56
V.6. Apprentissage des réseaux multicouches 56

CHAPITRE VI : Applications SISO.

VI.1. Introduction 64
VI.2. Application à la commande d'un système linéaire 64
VI.3. Application à un système non linéaire 70
VI.4. Conclusion 72

CHAPITRE VII : Commande d'un bras manipulateur.

VII.1. Introduction 84
VII.2. Modèlisation 84
VII.3. Simulation en boucle ouverte 89
VII.4. Commande en boucle fermée 90
VII.5. Test de robustesse 91
VII.6. Commentaire 92
VII.7. Synchronisation des trois articulations 92
VII.8. Commande du robot avec génération de trajectoires 92
VII.9. Test de robustesse sous génération de trajectoires 94
VII.10. Test de répetabilité 95
VII.11. Conclusion 96

CONCLUSION GENERALE 106

BIBLIOGRAPHIE

INTRODUCTION GENERALE

De nos jours, les automaticiens sont appelés à commander des systèmes de plus en plus complexes. Leurs modèles mathématiques, souvent trés compliqués, ne réflètent pas avec exactitude la dynamique de ces systèmes qui sont dans la plupart des cas non linéaires ou à paramètres variables.

Pour apporter une solution à ce problème, et grâce au developpement de l'intelligence artificielle et de l'electronique industrielle, la synthèse de commandes dites intelligentes et devenue possible.

L'objet de ce projet est double:

D'une part l'étude d'un contrôleur alliant les lois de commandes issues de la technique linguistique floue et le pouvoir d'apprentissage et de généralisation des réseaux neuronaux artificiels multicouches.

D'autre part, l'application du contrôleur ainsi obtenu à la commande de systèmes de type SISO et MIMO.

Le chapitre I fait une présentation des réseaux neuronaux artificiels.

Le chapitre II décrit l'algorithme d'apprentissage des réseaux neuronaux multicouches (Backpropagation) et les problèmes le plus souvent rencontrés lors de son utilisation. Quelques solutions sont proposées.

Le chapitre III présente les techniques et les structures d'apprentissage des réseaux multicouches utilisées pour leur intégration dans des chaines de réglage autant que contrôleurs.

Le chapitre IV introduit les principes de la commande

linguistique à base de logique floue.

Le chapitre V présente la loi de commande et la structure de différents contrôleurs neuro-linguistiques et des commentaires sur leur apprentissage.

Les chapitres VI et VII sont consacrés à la partie application. Le premier concerne l'application du contrôleur neuro-linguistique à deux systèmes SISO: l'un linéaire et le second non-linéaire. Le second chapitre présente la commande d'un robot manipulateur à trois degrés de liberté.

CHAPITRE I

PRESENTATION DES RESEAUX DE NEURONES

I.1. Introduction:

Depuis toujours l'homme a été fasciné par la compléxité et la puissance du cerveau humain.

De nos jours, il existe plusieurs techniques déstinées à imiter, voir simuler les fonctions du cerveau humain telles que: mémoriser, organiser, décider. Ces techniques sont rassemblées sous l'appelation d' "Intelligence Artificielle".

Parmi les techniques de l'intelligence artificielle, celle des "systèmes experts" [1] est sans doute la plus connue. Elle a été largement utilisée durant Ces vingts dernières années.
Cette technique est basée sur une approche dite "descendente".Les systèmes experts sont construits sur la base de l'acquisition de l'experience humaine.

Durant les années quarante, et grâce aux travaux de deux chercheurs Mc Culloch et Pitts [1],[2] est apparue une nouvelle approche de simulation du cerveau humain qui propose de modéliser ce dernier à partir de son constituant de base: le neurone. C'est de là qu'est né le concept de réseaux neuronaux artificiels.

Cependant, l'étude des réseaux neuronaux n'a pris son essor qu'à partir des années soixante. En effet, c'est à partir de cette époque qu'ont été écrits les premiers algorithmes pour la simulation de ces réseaux [2]. Malheureusement, et à cause des maigres résultats obtenus, l'engouement pour cette technique c'est rapidement calmé.

Aprés une éclipse qui a duré une vingtaine d'années, l'étude des réseaux de neurones fait l'objet d'un regain d'intérêt et

ceci depuis la seconde moitié des années quatre-vingts [1],[2].

Pour situer les réseaux neuronaux par rapport à l'intelligence artificielle, on peut dire que ces réseaux n'ont pas pour vocation de remplacer l'intelligence artificielle mais que c'est une technique complémentaire, et contrairement à cette dernière elle est basée sur une approche dite "ascendante", c'est à dire que les réseaux de neurones ont pour but de produire des phénomènes complexes à partir d'opérations simples [3].

Les réseaux neuronaux sont essentiellement pluridisciplinaires [3], c'est à dire qu'on les retrouve dans des domaines aussi variés que différents que sont la compression d'images, la reconnaissace de la parole, l'identification de modèles physiques ou la commande automatique des systèmes.

I.2. Qu'est-ce qu'un réseau de neurones ?

Un réseau de neurones artificiels est déterminé par trois caractéristiques:

-Le neurone.

-L'architecture du réseau.

-L'évolution du réseau.

I.2.1. Le neurone:

C'est une unité de calcul qui reçoit des informations, les traite et envoie le produit de son traitement à d'autres unités qui lui sont semblables.

Le modèle du neurone représenté sur la figure I.1 est appelé neurone de Mc Culloch et Pitts [4].
La structure du neurone artificiel est semblable à celle du neurone biologique. Le neurone biologique [3],[4] reçoit les informations venant d'autres neurones par l'intermediaire de connexions appelées synapses qu'il établit avec les dendrites émises par les axones d'autres neurones.

Dans le cas artificiel, le neurone reçoit des informations émises par d'autres neurones par l'intermediaire de synapses qui sont représentées par des coefficients appelés poids synaptiques et notés w (voir figure I.1). Il emet à son tour le produit de son traitement sur un axone qui le relie à d'autres neurones.
Le traitement de l'information par le neurone de Mc Culloch et Pitts s'éffectue en deux étapes:

1^{ere}étape: Le neurone éffectue une somme des informations qui lui parviennent ponderée chacune par le poids synaptique qui lui est associé.

2emeétape: Pour calculer sa sortie, le neurone applique à cette somme une fonction dite fonction d'activation qui peut être linéaire ou non linéaire.

La majorité des neurones utilisés dans les différents types de réseaux peuvent être considérés comme des neurones de Mc Culloch et Pitts. Une différence, que l'on rencontre souvent, réside dans le choix de la fonction d'activation. Du fait que cette fonction

peut être non linéaire découle l'originalité des réseaux neuronaux. Ceci les rend capables de modéliser des procédés non linéaires [3].

I.2.2. L'architecture du réseau:

L'architecture d'un réseau est déterminée par le nombre de neurones qu'il contient et la manière avec laquelle ils sont connéctés entre eux [1]. Il existe des réseaux dans lesquels les neurones sont entièrement connéctés et d'autres où ils le sont partiellement. Cette question est traitée plus en détail dans le paragraphe I.3.

I.2.3. L'évolution du réseau:

Cette notion d'évolution du réseau est trés importante. On entend par elle la manière avec laquelle les poids synaptiques entre neurones sont modifiés pour répondre à un besoin bien spécifique. Ces modifications ne peuvent se faire que grâce à des algorithmes qui éffectuent l'apprentissage d'un réseau.

I.3. Différents types de réseaux [2],[4]:

On peut distinguer principalement deux types de réseaux neuronaux:

-Les réseaux multi-couches.

-Les réseaux entièrement connéctés.

I.3.1. Les réseaux multicouches:

Ces réseaux, comme leur nom l'indique, sont constitués de couches de neurones. Ils possèdent une entrée et une sortie qui sont distinctes et l'information y circule dans un seul sens (de l'entrée vers la sortie). Au sein d'une même couche les neurones ne communiquent pas entre eux. Ils reçoivent des informations venant de la couche précédante, les traitent et les envoient à la couche d'ordre immédiatement supèrieur.

Il existe une variante de ce type qui diffère par le fait que la sortie du réseau est réinjectée dans son entrée. Dans ce cas on dit que le réseau est reccurent [2].

I.3.2. Les réseaux entièrement connectés:

Ces réseaux sont caractérisés par le fait que leurs neurones sont tous connectés entre eux. Dans cette configuration chaque neurone reçoit des informations de la part de tous les autres neurones, les traitent et envoit sa sortie à tous les autres neurones. Dans ce cas de figure il n'y a plus de distinction entre entrée et sortie du réseau.

I.4. Apprentissage d'un réseau:

L'interêt pour les réseaux de neurones s'explique par leur capacité à mémoriser, classer, organiser et plus généralement à apprendre.

L'apprentissage d'un réseau [2] consiste à le faire évoluer d'un état initial vers un état dans lequel chaque information en entrée du réseau produit la sortie désirée. Cette évolution est matérialisée par les variations que doivent subir les poids synaptiques pour arriver à l'état recherché. Ces variations sont determinées par des algorithmes d'apprentissage.
On peut distinguer deux types d'apprentissages différents:

I.4.1. L'apprentissage supervisé [2]:

Ce type d'apprentissage est basé sur la présentation au réseau d'exemples constitués chacun d'une entrée et de sa sortie correspondante. On parle d'apprentissage supervisé car le réseau n'est pas libre d'éffectuer son évolution librement à partir de la présentation d'entrée, mais ses poids synaptiques sont déterminés en fonction des sorties correspondantes aux entrées.

I.4.2. L'apprentissage non supervisé [2]:

Ce type d'apprentissage est bien plus intéressant que le premier car il se rapproche plus du modèle biologique.
Cet apprentissage consiste à présenter au réseau des informations groupées sous forme de classes à l'intèrieur desquelles ces données présentent des caractères communs.

I.5. Classification des réseaux neuronaux suivant l'architecture et le type d'apprentissage:

Aprés avoir présenté les différents types de réseaux et les différents types d'apprentissages, une classification des réseaux neuronaux avec leurs applications typiques a été proposée [5]:

	Non Recurrent	Reccurent
Supervisé	Type 1	Type 3
Non Supervisé	Type 2	Type 4

Type 1: Non reccurent-Supervisé.

Cette classe correspond aux réseaux multi-couches. Ces réseaux sont largements utilisés pour des diverses applications: Identification et commandes de processus, reconnaissances de formes et compression d'images. L'algorithme utilisé pour l'apprentissage est appelé Backpropagation [6]. Cet algorithme est l'objet du chapitre II de ce document.

Type 2: Non reccurent-Non supervisé.

Ce type de réseau est utilisé pour organiser et classer des informations. Ce modèle a été développé par un chercheur finlandais Teuvo Kohonen [7] dont les travaux a sont d'une grande rigueur mathématique [8].

Type 3: Reccurent-Supervisé.

C'est un type de réseau trés peu utilisé. Néanmoins, ce modèle existe et a été étudié par deux chercheurs Almeida en 1987 et Pineda en 1988. Ils ont montré que la reccurence n'affecte pas la stabilité du réseau et que celui-ci présente une bonne capacité d'apprentissage. Le réseau est caractérisé par un comportement dynamique lors de son exploitation.

Type 4: Non supervisé-reccurent.

Ceci correspond à un réseau de type totalement connécté. Le modèle utilisé pour l'apprentissage est le modèle de Hopfield [2],[23]. Il existe deux principales approches d'utilisation de ce modèle: la mémoire associative et la minimisation de formes quadratiques.

-Mémoire associative: dans ce cas la phase d'apprentissage est remplacé par une phase d'initialisation des poids synaptiques. On dit que l'apprentissage est statique. Par contre, au cours de son exploitation, le réseau à un comportement dynamique du fait de son caractère reccurent.

-Minimisation de formes quadratiques: Le réseau est alors un outil d'optimisation. Néanmoins, les performances du modèle de Hopfield utilisé en optimisation sont en général peu satisfaisantes. On préfere, dans ce cas, utiliser un modèle d'apprentissage basé sur une méthode statistique appelée la machine de Boltzmann. Ce modèle nous permet de remarquer que les réseaux neuronaux tirent leur modélisation de la structure du cerveau humain mais aussi des résultats de la physique

statistique [3].

A l'issue de cette classification on peut ajouter que quelque soit le type d'un réseau son fonctionnement reste caractérisé par trois principes communs à tous les types de réseaux de neurones:

-Le parallélisme des opérations de traitement.

-Le caractère collectif et distribué de l'activité des neurones.

-La capacité d'apprentissage à partir d'exemples.

I.6. Les niveaux de réseaux de neurones [3]:

Il existe trois moyens ou niveaux permettant l'apprentissage et/ou l'utilisation de réseaux neuronaux.

I.6.1. La simulation logicielle:

Actuellement, la majorité des recherches sur les réseaux de neurones se font à l'aide de simulations logicielles. Ce moyen présente plusieurs avantages tels que: faible coût et souplesse d'utilisation.

I.6.2. Les neuro-ordinateurs:

Ce ne sont pas réellement des réseaux neuronaux physiques comme leur nom l'indique mais des ordinateurs comportant des processeurs à architecture vectorielle dédiée au calcul neuronal. Malgré l'apparition de ces architectures parallèles, certaines tâches restent trés difficiles à éffectuer voir même impossibles à cause de la puissance de calcul demandée qui est souvent trés importante.

I.6.3. Les circuits neuronaux:

Ce sont des circuits électroniques où sont implantés des réseaux neuronaux. Il existe deux technologies utilisées pour réaliser de tels circuits:

-Technologie VLSI (Very Large Scale Integration).

-Technologie optique.

On retrouve aussi bien des circuits analogiques que des circuits numériques de réseaux neuronaux. La technologie optique a été introduite pour résoudre le problème de l'encombrement dû au grand nombre de connexions entre neurones. Ce phénomène de grande densité des connexions est caractéristique des réseaux de neurones.

A noter que ces technologies de réalisation des réseaux utilisent les résultats de la simulation logicielle pour connaitre l'architecture et les poids synaptiques du réseau à implanter.

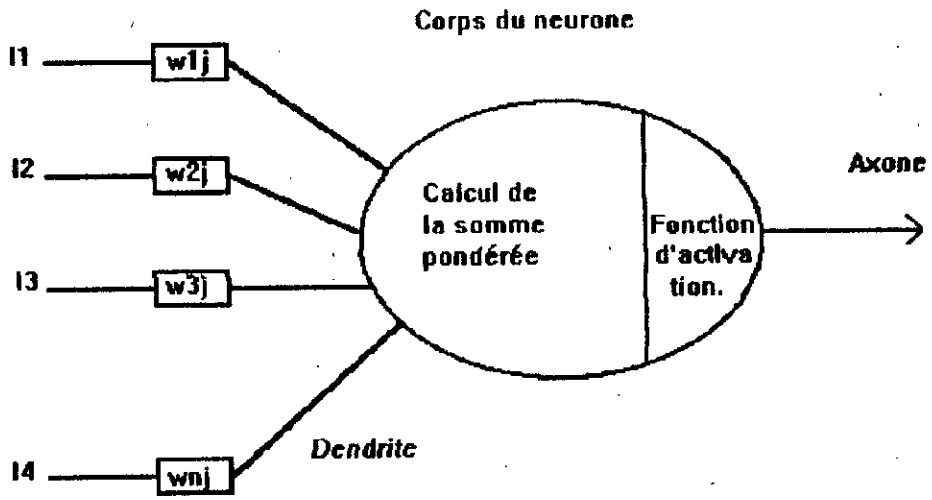

Figure I.1- Neurone de McCulloch et Pitts.

CHAPITRE II

BACKPROPAGATION

II.1. Introduction:

Minsky et Papert [2],[6] ont démontré en 1969 qu'il y avait de trés grandes restrictions dans la capacité d'apprentissage des réseaux de neurones monocouches. Et du fait qu'il n'existait pas à cette époque de technique pour l'entrainement des réseaux multi-couches, de nombreux chercheurs ont montré peu d'intêret pour le domaine des réseaux neuronaux et se sont intéréssés à d'autres domaines plus prometteurs.

En 1974, Werbos [9], de l'université de Harvard a développé dans sa thèse de doctorat une technique pour l'entrainement des réseaux multicouches basée sur la rétropropagation de l'erreur de sortie. Malheureusement, ce travail est resté presque inconnu de la communauté scientifique.

En 1982, Parker [2],[9] a retrouvé de nouveau la technique de la rétropropagation de l'erreur de sortie, et en avril 1985 ses travaux ont fait l'objet d'un rapport publié par le M.I.T. Quelques mois plus tard, Rumelhart, Hinton et Williams [6] ont présenté cette méthode d'une façon claire, précise et rigoureuse. Le résultat auquel ils sont arrivés était si important qu'il a eu un grand echo dans le domaine des réseaux neuronaux. Et on estime aujourd'hui que leur travail a relancé l'intêret pour ce domaine.

Cette technique (ou cet algorithme) est connue sous le nom de "Backpropagation". Cet algorithme repose sur une base mathématique et a pour principe la minimisation de l'erreur quadratique entre la sortie du réseau et celle désirée et ceci

en utilisant la méthode de descente du gradient (Steepest-Descent) [6].

II.2. Modèle du neurone et du réseau:

a. Le neurone:

En se basant sur le fonctionnement du neurone biologique et sur sa façon de recevoir, de traiter et de transmettre l'information, un modèle mathématique du neurone fut proposé [2],[4]. Ce modèle se base sur l'observation et le traitement d'informations venant d'autres neurones et transmises par les synapses. Afin de modéliser les synapses, des coefficients appelés "poids synaptiques" sont introduits. L'information transmise au neurone est alors une sommation des informations venant des autres neurones, pondérées par les poids synaptiques, traduisant l'inégale importance de ces informations. Elle est exprimée par:

$$NET=\sum_{i=1}^{n} W_i L_i \qquad (2.1)$$

où:

W_i sont les poids synaptiques.

L_i les informations en entrée du neurone.

n le nombre de neurones connéctés à celui-ci.

La deuxième partie du traitement de l'information par le neurone consiste en l'application à la somme pondérée NET d'une fonction d'activation choisie non linéaire pour permettre au réseau

d'identifier des processus non linéaires [3]. Cette fonction doit être nécéssairement dérivable, non décroissante et saturable de préference. Sa dérivabilité est rendue nécessaire pour les besoins de l'algorithme qui va être exposé plus loin dans ce chapitre.

La fonction généralement utilisée est la fonction sigmoide qu'on appelle aussi "The Logistic Function".

La sortie du neurone notée OUT est alors calculée de la manière suivante:

$$OUT = \frac{1}{1+e^{-NET}} \qquad (2.2)$$

Le graphe de cette fonction est donnée par la figure II.1.

Un montage du neurone peut être réalisé à l'aide de composants de l'electronique analogique, voir figures II.3 et II.4 [10].

b. Le réseau:

Un réseau de neurones multicouche est constitué d'un ensemble de neurones, eux même organisés en n groupes à l'intérieur desquels les différents neurones ne communiquent pas entre eux mais où chacun reçoit de l'information parvenant des neurones du groupe n-1 les traitent et les transmet aux neurones du groupe n+1. Ces groupes sont appelés couches de neurones.

Dans un réseau multicouches on peut distinguer trois types de couches:

-La couche d'entrée.

-Les couches cachées (une au minimum).

-La couche de sortie.

Au niveau de la couche d'entrée aucun traitement de l'information n'est éffectué. En fait cette couche ne comporte pas de neurone, elle est constituée de noeuds dont le rôle est de distribuer l'information pour les neurones de la première couche cachée.

Les couches cachées jouent un rôle trés important dans les réseaux multicouches. En effet, l'information venant de la couche d'entrée est codée en une "représentation interne" par les couches cachées pour générer les sorties désirées. Si le réseau contient plusieurs couches cachées, il peut coder toute entrée en une représentation interne de telle manière à produire la sortie désirée [6].

Il n'existe pas de règle générale [1] pour determiner le nombre de couches cachées ainsi que le nombre de neurones par couche cachée. Un exemple de réseau à 3 entrées, 3 sorties et une couche cachée est représenté sur la figure II.2.

II.3. Présentation de l'algorithme Backpropagation:

Le but de l'apprentissage est de déterminer les poids synaptiques qui définissent les connexions entre les neurones pour produire les sorties désirées.

Le réseau peut contenir un nombre quelconque de couches cachées.Néanmoins, un modèle à une couche cachée sera utilisé dans ce qui suit pour illustrer l'algorithme.

Avant de commencer le processus d'apprentissage, tous les

poids sont initialisés aléatoirement à des valeurs généralement comprises entre 0 et 1 [2].

Le déroulement de l'apprentissage nécéssite un ensemble d'entrées et de sorties organisées en paires: (entrée, sortie correspondante). Cet ensemble est appelé: Ensemble des couples d'apprentissage.

Les différentes étapes de l'algorithme sont :

Etape 1: Initialisation de tous les poids à de petites valeurs aléatoires.

Etape 2: Choisir un élement parmi l'ensemble des couples d'entrainement (une entrée et sa sortie correspondante) et le présenter au réseau.

Etape 3: Calculer la sortie du réseau.

Etape 4: Calculer l'erreur entre la sortie du réseau et la sortie désirée.

Etape 5: Corriger les poids du réseau de façon à minimiser l'erreur.

Etape 6: Refaire les étapes 2 à 5 pour chaque couple d'apprentissage tant que l'erreur n'est pas inférieure à un seuil qu'on se fixe.

Si on note par (X_k, D_k) le couple n°k de l'ensemble des couples d'apprentissage on définit l'erreur par:

$$E_k = Y_k - D_k = \frac{1}{2}\sum_{j=1}^{n} (Y_{jk} - D_{jk})^2 \qquad (2.3)$$

Où Y_k est la sortie du réseau correspondant à l'entrée X_k avec X_k appartenant à $\mathbf{R}^n$, D_k et Y_k appartenant à $\mathbf{R}^m$.
On définit l'erreur globale comme étant la somme des erreurs sur chaque couple d'apprentissage:

$$E=\sum_{k=1}^{p} E_k \qquad (2.4)$$

Où p represente le nombre de couples d'apprentissage.

La minimisation de l'erreur globale E revient à la minimisation de l'erreur E_k pour chaque couple d'apprentissage. Pour cela on utilise une descente du gradient [1],[2],[6].

Concernant les poids des connexions entre la couche cachée et la couche de sortie, on a la loi de variation suivante:

$$\Delta_k W_{ij}=-\eta \frac{\partial E_k}{\partial W_{ij}} \qquad (2.5)$$

où $\Delta_k W_{ij}$ est la variation du poids de la connexion entre le neurone i de la couche cachée et le neurone j de la couche de sortie et ceci pour le couple d'entrainement k. On peut écrire aussi:

$$\Delta_k W_{ij}=\eta \delta_{jk} OUT_{ik} \qquad (2.6)$$

OUT_{ik} étant la sortie du neurone i de la couche cachée, avec :

$$\delta_{jk}=(D_{jk}-Y_{jk})\,OUT_{jk}(1-out_{jk}) \qquad (2.7)$$

La modification du poids entre le neurone i et le noeud r de la couche d'entrée est donnée par:

$$\Delta_k W_{ri}=-\eta\frac{\partial E_k}{\partial W_{ri}}=\eta\delta_{ik}X_{rk} \qquad (2.8)$$

avec X_{ir} la composante n°r du k^{ème} vecteur d'entrée. Et dans ce cas:

$$\delta_{ik}=OUT_{ik}(1-OUT_{ik})\sum_{j=1}^{n}W_{ij}\delta_{jk} \qquad (2.9)$$

avec n le nombre de neurones de la couche cachée.

II.4. Problèmes posés par Backpropagation:

Backpropagation est un algorithme qui pose plusieurs problèmes et qui peuvent parfois compromettre l'apprentissage d'un réseau. Voici dans ce qui suit l'exposé des principales difficultés qu'on peut rencontrer avec cet algorithme:

a-La convergence: La convergence d'un réseau multicouche n'a pas pu être prouvée mathématiquement.
Néanmoins, on a observé expérimentalement que cet algorithme converge toujours vers une solution mais qui peut parfois être insatisfaisante.

Cependant, ces observations sont basées sur des cas particuliers [1], et ne peuvent constituer une règle générale quant à la certitude de la convergence de cet algorithme.

b-Le temps d'apprentissage: Le temps nécéssaire à l'apprentissage devient rapidement trés important lorsque l'application nécéssite un grand nombre de couples d'apprentissage. Le temps peut atteindre quelques jours voir plusieurs semaines et ceci même sur des machines puissantes tels que station de travail.
Ceci constitue le principal obstacle pour l'exploitation de façon concrète de cet algorithme.

c-Minimum local: Cet algorithme de minimisation étant basé sur le principe de la descente d'un gradient, il est confronté au problème des minimums locaux [1],[2].
En effet, la surface de l'erreur étant généralement non convexe elle possède plusieurs minimums de différentes valeurs. Quand le réseau se stabilise autour d'un minimum il ne peut plus en sortir car toutes les directions correspondent alors à une augmentation et non à une diminution de l'erreur.
En 1988, Wasserman a proposé une amélioration de l'algorithme par l'utilisation d'une méthode statistique dite "La machine de Cauchy". Elle consiste à rajouter un terme supplémentaire à la correction qu'on apporte aux poids à chaque itération.

L'originalité de cette technique est qu'elle permet au réseau de s'échapper d'un minimum local. Néanmoins, sa mise en oeuvre est assez délicate [2].

d-Le pas de correction des poids: Dans la description originale de l'algorithme [6], le pas de correction des poids est supposé infiniment petit. Or, ceci n'est pas applicable en pratique car l'apprentissage nécessiterait un temps quasiment infini. Un pas trop grand conduit le réseau à osciller ce qui compromet sa convergence [2].
Une solution adéquate consiste à choisir un pas variable. Celui ci sera initialisé à une grande valeur (comprise entre 0 et 1), qui sera diminué jusqu'a une valeur minimale positive fixée au préalable.

e-Saturation du réseau: Des cas où les poids prennent de grandes valeurs peuvent exister, impliquant des sorties de valeurs élevées, prochent, si ce n'est dans la zone de saturation de la fonction d'activation.
Une solution consisterait à choisir un petit pas de correction des poids. Or, ceci aurait pour conséquence d'augmenter considérablement la durée de l'apprentissage, constituant ainsi un dilemne.
Pour ne pas être confronté à cette situation, les poids sont initialisés à de trés petites valeurs diminuant ainsi le risque de saturation du réseau.

f-Dimensionnement du réseau: Pour une application donnée aucune information sur l'architecture du réseau n'existe à priori. Les questions qu'il faut résoudre sont les suivantes:

-Quel est le nombre de couches cachées qu'il faut donner au réseau?

-Combien de neurones doit on mettre dans chacune de ces couches cachées?

En fait, il n'existe aucune méthode ni aucune règle permettant de répondre à ces questions [1]. Un moyen permettant le choix d'une structure consiste à effectuer un apprentissage (ne serait-ce que partiel) de plusieurs types de réseaux et d'en déterminer celui qui présente les meilleures capacités d'apprentissage pour l'application considérée.

On peut remarquer rapidement en utilisant Backpropagation qu'augmenter le nombre de neurones d'une couche cachée ou qu'ajouter des couches cachées n'améliore pas systématiquement les performances du réseau.

Le problème de dimensionnement de réseaux multicouches fait actuellement l'objet d'intenses recherches.

De récents résultats ont montré qu'un réseau avec une ou deux couches cachées seulement permet d'approximer un grand nombre de fonctions continues [11].

Malgré toutes les difficultés présentées par Backpropagation, cet algorithme s'est révélé trés performant lors de son application dans plusieurs domaines aussi variés que:

reconnaissance de formes [1], compression d'images[2], étude de la stabilité de réseaux éléctriques [12] et aussi en identification et controle de processus [13].

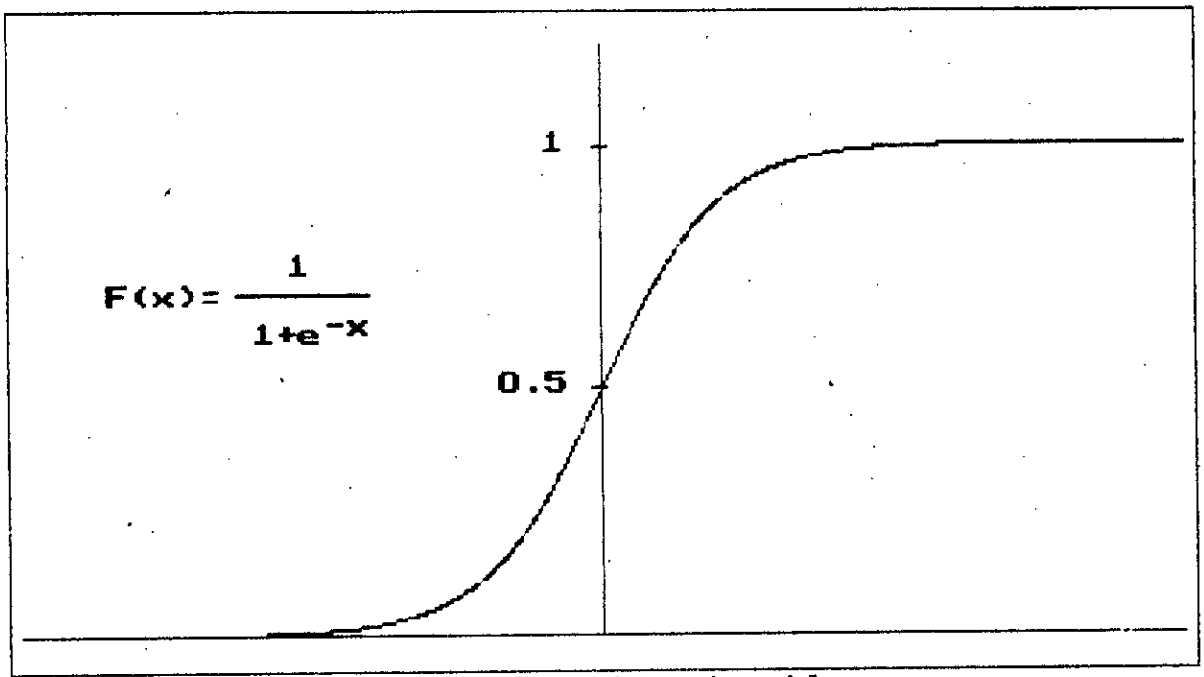

Figure II.1. Graphe de la fonction sigmoide.

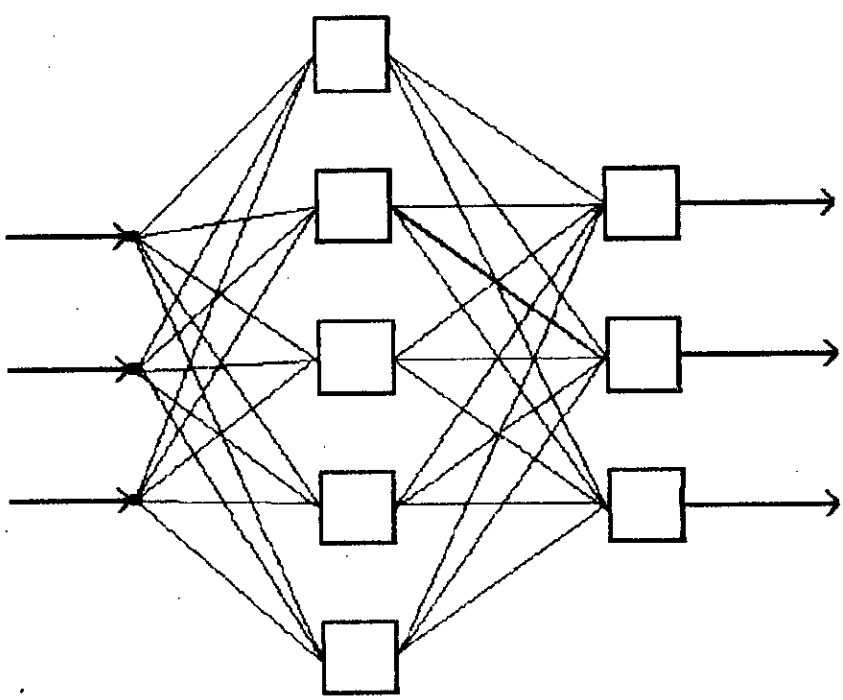

Figure II.2. Modèle du réseau.

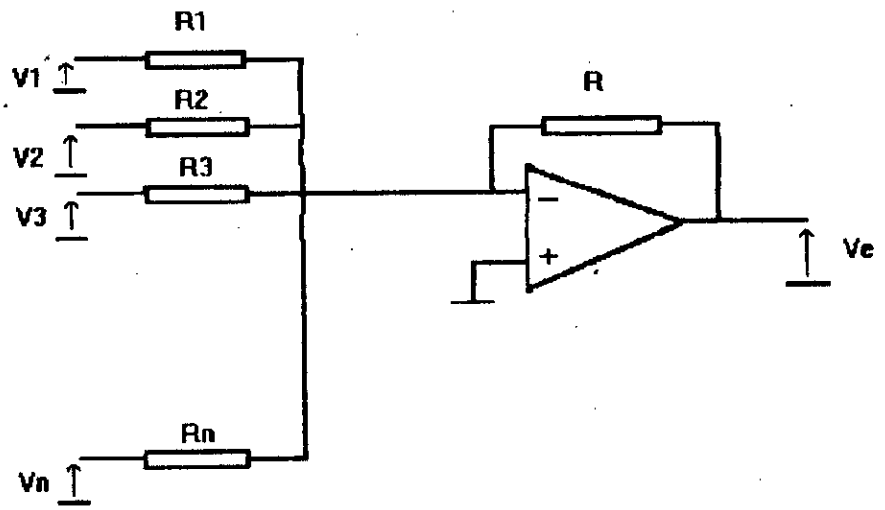

Figure II.3. Circuit de calcul de la somme podérée.

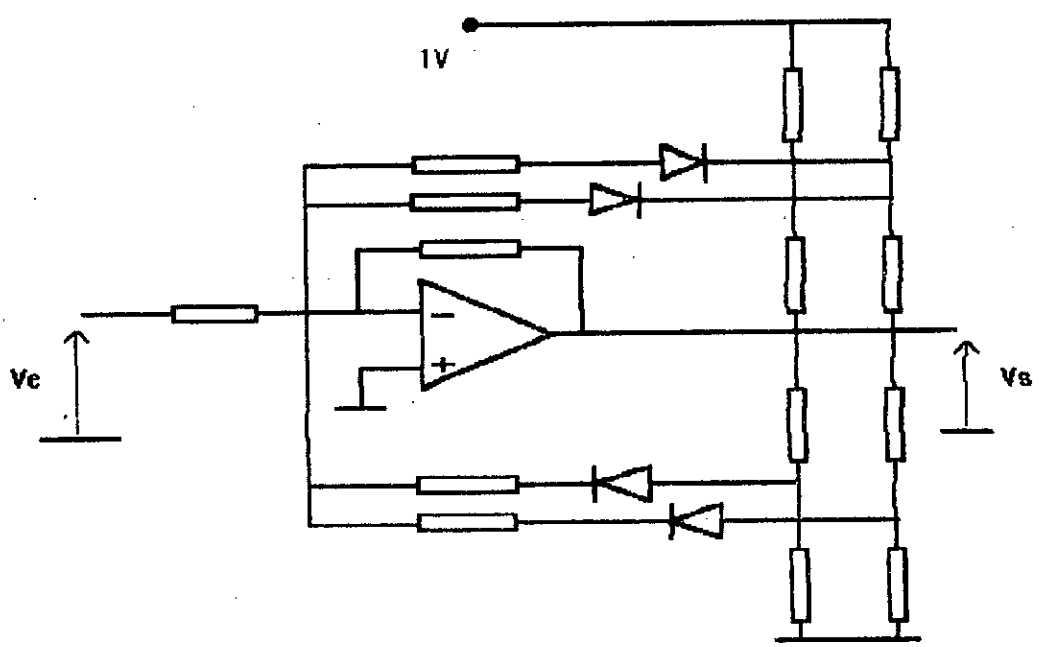

Figure II.4. Circuit de génération de la fonction d'activation.

CHAPITRE III

BACKPROPAGATION ET LA COMMANDE AUTOMATIQUE DES SYSTEMES

III.1. Introduction:

Depuis son apparition, l'algorithme de backpropagation a été appliqué dans plusieurs domaines, néamoins il a été conçu particulièrement en vue de la commande automatique des systèmes [3]. Son originalité lorsqu'il est appliqué à ce domaine est sa capacité à commander des systèmes non-lineaires dont les modèles sont mal ou peu connus ou dont les paramètres sont variables. De plus de telle réseaux présentent une bonne résistance au bruit en entrée [2].

Le problème qui se pose est la manière d'intégrer le réseau à une structure de réglage et aussi comment éffectuer son apprentissage en vue de sa préparation pour la fonction qui lui est assignée.
Pour cela plusieurs structures d'apprentissage des réseaux neuronaux éxistent et présentent chacunes des avantages et des inconvenients.

Le but de ce chapitre est de présenter ces différentes structures, les inconvenients qu'elles présentent et les techniques utilisées pour répondre à ces insuffisances.

III.2. Une structure génerale de commande:

Avant d'étudier les structures d'apprentissage, nous presentons une structure de réglage (voir la figure 3.1) qui contient deux types de correcteurs:

-Un correcteur qui fournit une commande à partir du signal d'erreur.(Feedback Controller).

-Un correcteur qui fournit une commande à partir du signal de consigne.(FeedForward Controller).

Le second correcteur trés peu utilisé, son action a pour effet d'accélerer la réponse du système à l'action du premier qui génere la commande proprement dite.

Un aspect fondamental sépare les deux correcteurs. Le premier a un comportement dynamique alors que le second, lui, a un comportement statique.

Ces deux types de controleurs peuvent être en fait des réseaux de neurones multicouches préalablement entrainés, par des vecteurs d'apprentissages différents.

Dans ce qui suit nous allons passer en revue les différentes possibilitées d'apprentissage du correcteur Feedforward puis nous nous interesserons aux problèmes posés par le correcteur Feedback et les différentes solutions possibles.

III.3. Apprentissage du correcteur feedforward:

III.3.1. Apprentissage généralisé (Off-Line):

Son architecture est representée sur la figure 3.2. Son but est d'identifier le modèle inverse du système, c'est à dire que le réseau neuronal fournira la commande qu'il faut appliquer au système par la donnée de la consigne.

Néanmoins, cette technique d'apprentissage [13],[15] présente un inconvénient certain: ne connaissant pas à priori les commandes correspondant aux sorties désirées, le succès de cet apprentissage depend alors de la capacité de généralisation du réseau choisi (nombre de couches cachées et nombre de neurones par couche).

Une solution est d'éffectuer l'apprentissage sur une grande plage de variation de la commande. Mais là aussi, la durée du temps de calcul est souvent trés importante.

III.3.2. Apprentissage Specialisé (On-Line):

Ce type d'apprentissage s'éffectue en On-line [13],[14],[15]. Son architecture est representée sur la figure 3.3.
Des consignes C sont appliquées au réseau qui délivre une commande U au système. les poids du réseau sont corrigés de telle sorte à minimiser l'erreur E=C-Y.

Ceci impose l'expression de l'erreur en sortie du réseau en terme de commande. En effet on a:

$$E=\frac{1}{2}\sum_{k=1}^{n}(Y_k-C_k)^2 \qquad (3.1)$$

n étant le nombre de vecteurs d'apprentissage.

et donc:

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial Y} \cdot \frac{\partial Y}{\partial U} \cdot \frac{\partial U}{\partial W} \qquad (3.2)$$

$$\frac{\partial E}{\partial W} = (Y-C) \cdot Y'(U) \cdot \frac{\partial U}{\partial W} \qquad (3.3)$$

Cette dernière égalité permet alors d'appliquer l'algorithme backpropagation.

La connaissance de la dérivée partielle de la sortie par rapport à la commande n'est pas toujours possible ce qui constitue un inconvénient pour la mise en oeuvre de cet apprentissage.

Un autre inconvénient de cet apprentissage est le fait qu'il soit éffectué en On-line et que les commandes en début d'apprentissage sont aléatoires. En effet les poids du réseau initialisés à des valeurs alèatoires, peuvent provoquer des dommages sur le système, voir même créer une instabilité.

III.4. Avantages et limites du correcteur feedforward:

Ces apprentissages permettent d'avoir des réseaux qui représentent des fonctions de transfert inverses.

Ces réseaux presentent l'avantage de pouvoir interpoler (traiter des cas non appris) [13]. Ils peuvent aussi être utilisés en temps réel si le modèle ainsi obtenu est implanté dans une puce [16].

A noter que cet aspect interesse particulièrement les domaines

où les systèmes sont non linéaires et à dynamique rapide (particulièrement en robotique) [14].

Un réseau entrainé suivant une des techniques précedantes constitue un modèle inverse statique du système. Il ne peut donc faire l'objet d'un correcteur dynamique qui puisse ramener le système d'un état à un autre suivant une dynamique préalablement choisie.

Cet aspet statique est intrinsèque au réseaux multicouches.

III.5. Apprentissage du correcteur feedback:

L'apprentissage d'un réseau neuronal, en vue de son utilisation en tant que correcteur dynamique, ne peut pas se faire directement et nécessite l'utilisation d'artifices et de techniques supplémentaires à cause du statisme du réseau.

Une solution a été apportée par T.Fukuda et al. [17] en ajoutant à la première couche du réseau des élements qui opérent un retard. La sortie du réseau dépend de son entrée aux instants t, t-1 et t-2. Dans le cas où les poids reliant les élements à retard à la couche suivante sont fixes le réseau est dit "Fixed Time Delay Neural Network (F.T.D.N.N)". Dans le cas où ces poids sont variables le réseau est dit "Active Time Delay Neural Network (A.T.D.N.N)" [24].

Une seconde solution appliquable dans le cas d'une poursuite de trajectoire consiste à informer le réseau sur la position, la

vitesse et l'accéleration du système ce qui donne un aspect dynamique au réseau neuronal.

Une autre approche à la résolution du problème, consiste en l'utilisation de la commande linguistique pour l'apprentissage d'un controleur neuronal.

III.6. Conclusion:

On a présenté dans ce chapitre les principales techniques utilisées pour l'apprentissage des réseaux multicouches en vue de leur application pour la commande de systèmes. L'apprentissage du controleur feedforward a été largement pratiqué ces dernières années [13],[14]. Par contre, les techniques d'apprentissage du controleur neuronal feedback font actuellement l'objet d'intenses recherches. Dans toute la suite de ce document nous allons étudier et appliquer une technique d'apprentissage basée sur la commande linguistique pour la conception d'un correcteur neuro-linguistique.

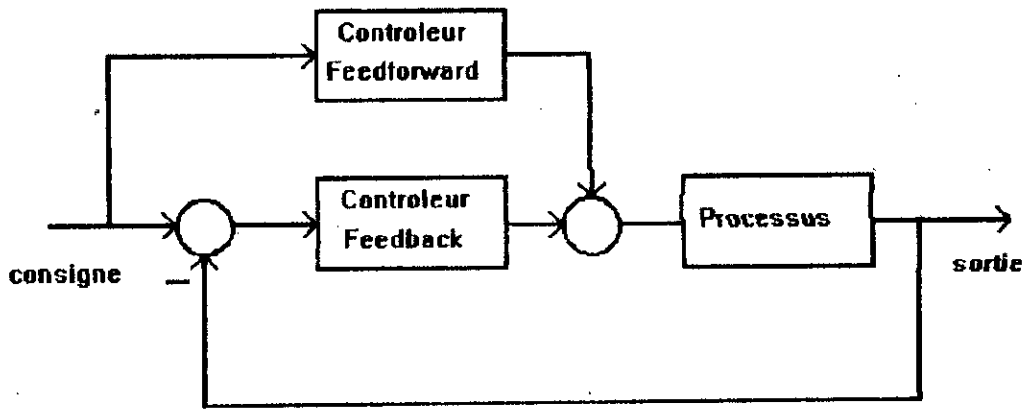

Figure III.1. Structure générale de commande.

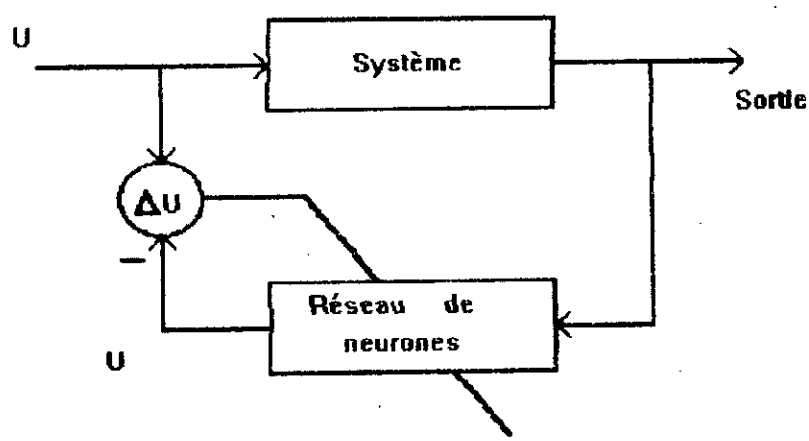

Figure III.2. Architecture de l'apprentissage généralisé.

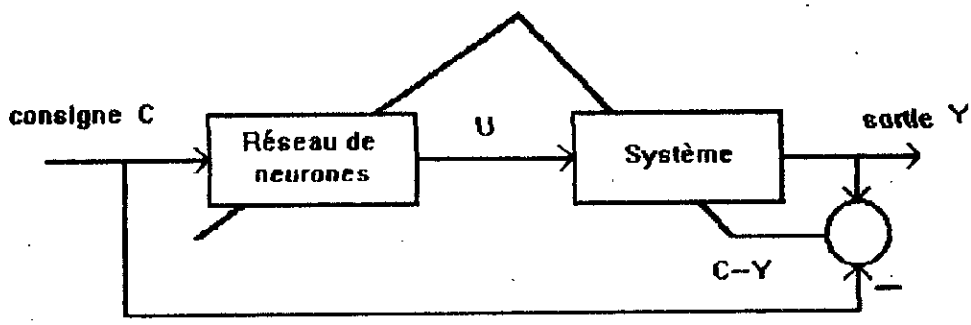

Figure III.3. Architecture de l'apprentissage spécialisé.

CHAPITRE IV

NOTION DE CONTROLEUR LINGUISTIQUE A BASE DE LOGIQUE FLOUE

IV.1. Introduction:

La commande linguistique, basée sur la logique floue, est une technique qui permet la manipulation de grandeurs dont les valeurs sont approximatives. Elle est trés proche du raisonnement humain et du language naturel [18].

IV.2. Definitions:

Ensemble flou:

Un ensemble flou F est caractérisé par une fonction d'appartenance μF (membership function) qui prend ses valeurs dans l'intervalle [0,1]. Pour tout élement x de l'intervalle d'intérêt U, on peut lui associer son degré d'appartenance à l'ensemble flou F en calculant μF(x).

l'ensemble flou sera défini par [18]:

$$F=\{(x,\mu F(x))/x\in U\} \qquad (4.1)$$

Variable linguistique:

Une variable linguistique est une grandeur dont l'ensemble des valeurs qu'elle peut prendre est divisé en plusieurs classes, à chacune de ces classes est associée une fonction d'appartenance.

Une variable linguistique est caractérisée par le couple (x,T(x))

où x est le nom de la variable et T(x) est l'ensemble des noms des valeurs lingiustiques que peut prendre x. Par exemple si x=vitesse, on peut avoir T(x)={ trés lent, lent, moyen, rapide, trés rapide }.

Fuzzification:

C'est l'opération qui consiste en la détermination des différents degrés d'appartenance d'une valeur x0 prise par une grandeur donnée aux différents sous-ensembles flous associés à cette grandeur.

Defuzzification:

C'est l'opération qui a pour effet de déterminer la valeur x0 prise par une grandeur donnée à partir de la connaissance de ses degrés d'appartenance à chacun des sous-ensembles flous associés. on a la relation dite du "Centre de Gravité":

$$x0=\frac{\sum_{i=1}^{n} \mu F_i(x0)\, T_i(x)}{\sum_{i=1}^{n} \mu F_i(x0)} \qquad (4.2)$$

où μFi(x0) est le degré d'appartenance de x0 au sous-ensemble flou Fi; Ti(x) est le représentant de la $i^{ème}$ classe et n le nombre total de classes.

IV.3. Le contrôleur flou:

Le contrôleur flou comporte essentiellement quatre parties [18]:

Une interface de fuzzification, une base de connaissances, une logique de décision et une interface de défuzzification.

-L'interface de fuzzification reçoit les valeurs des variables d'entrée, éffectue un changement d'échelle et convertie les grandeurs d'entrées en variables linguistiques auquelles on peut appliquer les régles d'inférences de la logique floue.

-La base de connaissances comporte des connaissances sur le comportement qualitatif du système et les objectifs que doit atteidre le controlleur.

-La logique de decision a la capacité de fournir les commandes floues à appliquer. La prise de décision par ce contrôleur se fait suivant le schéma suivant:

SI (une liste de conditions est satisfaite) ALORS (une liste de conséquences est déduite).

avec:

Liste de conditions= informations sur l'état du système.

Liste de conséquences= commandes à appliquer au système.

-L'interface de defuzzification éffectue l'opération de defuzzification et fait un changement d'echelle pour fournir la valeur physique de la commande à appliquer au système.

IV.4. Mise en oeuvre du contrôleur flou:

La mise en oeuvre du contrôleur flou ne nécessite pas la connaissance du modèle mathématique du système à commander mais uniquement son comportement qualitatif i.e. le sens de variations de ses sorties en fonction des variations de ses entrées [16]. Dans le cas multivariable, il est nécessaire de savoir si une entrée affecte toutes les sorties du système [19].
La mise en oeuvre du contrôleur passe par les étapes suivantes [18]:

1. Choix des variables qui serviront à la prise de décision:

Ceci consiste à choisir les grandeurs qui serviront de variables linguistiques pour le contrôleur. Ces grandeurs doivent refleter suffisamment l'état du système pour en déduire la commande à appliquer.

Le choix des variables linguistiques a un effet considérable sur les performances du système [16]. En général, on utilise comme variables linguistiques l'erreur entre la sortie et la consigne et la variation de cette erreur [20].

2. Normalisation:

Cette étape consiste à rapporter le domaine de variation de chacunes des variables linguistiques, qu'elles soient en entrée

ou en sortie, à un intervalle normalisé [-1,1]. Cette opération ne peut se faire indépendamment du processus à commander.

3. Partition des variables en classes:

Dans cette étape, on définie les valeurs linguistiques que peuvent prendre les variables linguistiques choisies en découpant l'intervalle d'interêt en un nombre fini de classes. Il n'existe pas de régles pour déterminer le nombre de classes, le choix est donc subjectif. Le nombre de classes doit être suffisamment grand pour avoir une bonne approximation, le choix des classes a une influence essentielle sur la finesse de la commande [18],[21],[22].

4. Complétude:

Le contrôleur flou doit être capable de déduire la commande à appliquer quelque soit l'état du système. On doit s'assurer que chaque élement de l'intervalle d'interêt appartient à au moins une des classes résultant du découpage de celui-ci. Ceci garantit que le contrôleur sera "Complet".

5. Choix de la fonction d'appartenance (membership function):

La fonction d'appartenance peut être exprimée sous deux

formes: numérique ou analytique.
Le choix de la fonction d'appartenance est trés subjectif et affecte sensiblement les performances du contrôleur. En particulier si les entrées du contrôleur sont bruitées, des fonctions d'appartenance choisies assez larges permettent de réduire sa sensibilité au bruit [18].
Il existe une grande variété de fonctions [21] suceptibles d'être utilisées comme fonctions d'appartenance. Certaines sont illustrées par la figure 4.1.

6-Source et derivation des régles floues:

Les régles floues pour la prise de décision peuvent être derivées en utilisant:

a-l'experience et la connaissance des systèmes:

A partir d'une bonne connaissance qualitative d'un système et utilisant son experience dans la commande des systèmes, un ingénieur est capable de constituer un ensemble de régles du type SI (...) ALORS (...) pour la commande d'un processus.
Une autre approche consiste à interroger des spécialistes experimentés dans la gestion d'un processus ou des opérateurs qui souvent à l'aide de manipulations basées sur l'experience arrivent à commander avec succés un processus dont ils ignorent trés souvent le modèle mathématique.

b-Un apprentissage:

Cette technique est différente de la première, elle a pour but d'utiliser une capacité d'apprentissage pour génerer une commande.

Procyk et Mamdani [19] ont décrit le premier controleur flou ayant la capacité de créer et de modifier les régles floues de commande en utilisant l'experience acquise (SOC).

Une autre possibilité consiste à utiliser la puissance de mémorisation et de généralisation des réseaux de neurones à partir d'exemples appris pour la géneration de la commande.

IV.5. Conclusion:

Contrairement aux méthodes de synthèse classiques, la logique floue permet d'élaborer une commande sans la connaissance du modèle mathèmatique du système à commander. Toutefois, le choix des régles floues, des variables linguistiques, des fonctions d'appartenance, des classes et de leur nombre est purement subjectif.

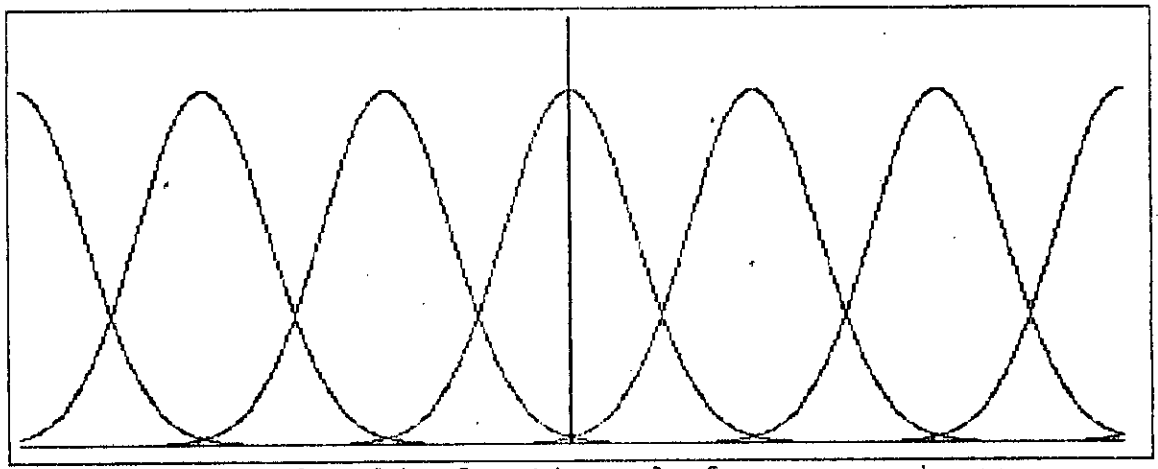

Figure 4.1. Membership functions de forme gaussienne.

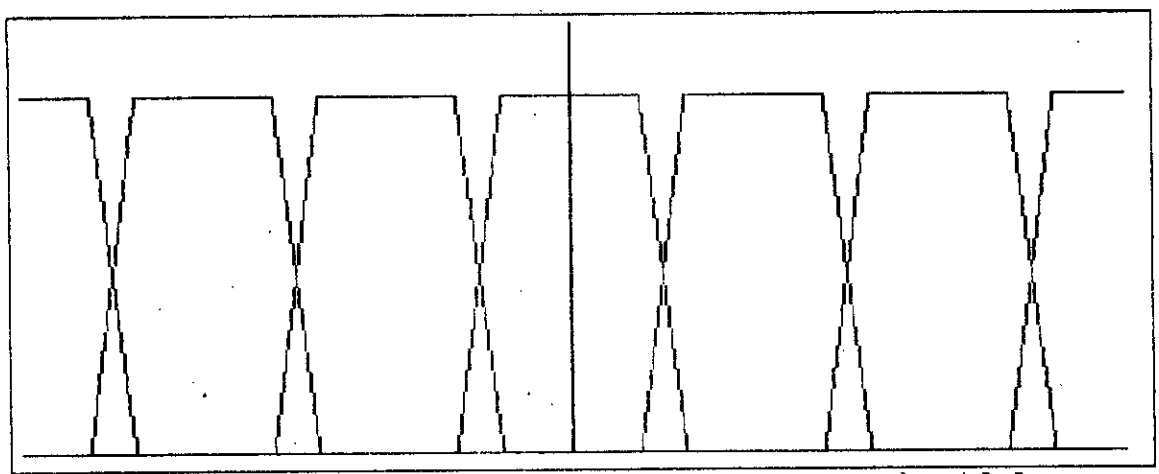

Figure 4.2. Membership functions de forme trapèzoidale.

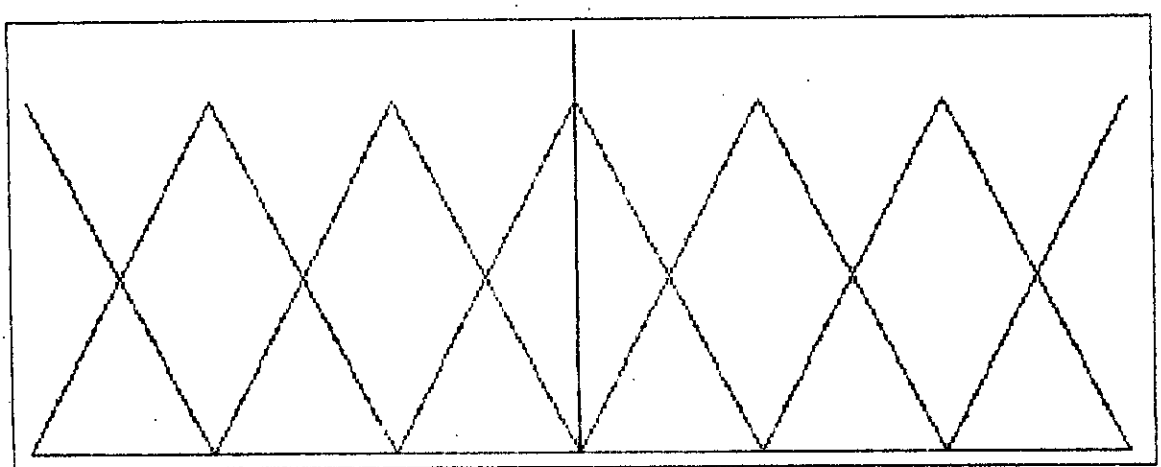

Figure 4.2. Membership functions de forme triangulaire.

CHAPITRE V

ASSOCIATION D'UN RESEAU DE NEURONES ET DE LA TECHNIQUE LINGUISTIQUE

CONTROLEUR NEURO-LINGUISTIQUE

V.1. Motivation:

La technique linguistique et la logique floue toutes deux associées constituent un moyen puissant pour la commande dynamique [18]. Néanmoins, elle nécessite une bonne connaissance qualitative du système à commander.

D'un autre coté, un réseau de neurones possède une grande capacité d'apprentissage et un pouvoir de généralisation à partir d'exemples appris. Il peut en principe approximer une quelconque loi analytique ou empiriqué [14].

De ceci apparait l'interêt de l'association de ces deux techniques aux caractéristiques complémentaires.

V.2. Réalisation de la partie linguistique:

Comme il a été exposé dans le chapitre IV de ce document, la premiere étape de la réalisation d'un contrôleur linguistique flou est le choix des variables linguistiques. Dans ce qui suit, nous avons choisi deux variables d'entrée et une variable de sortie. Dans le but de réaliser un contrôleur pouvant commander un grand nombre de systèmes, on a choisi pour variables d'entrée:

-L'erreur en sortie du système (par rapport à la consigne) et qu'on notera par "e".

-La dérivée de l'erreur et qu'on notera "Δe".

La variable de sortie est la commmande qu'on notera "u". Nous avons divisé l'erreur et sa dérivée en sept classes [21],[22] que nous avons nommées:

NB: Negative Big.

NM: Negative Medium.

NS: Negative Small.

ZR: Zero.

PS: Positive Small.

PM: Positive Medium.

PB: Positive Big.

Concernant la commande u, sa division en classes est plus délicate car dépendant de la division des variables d'entrée. En principe, pour la construction de règles de décision, si l'erreur et sa dérivée sont divisées en 2n+1 classes (avec $n \geq 1$) alors la commande peut l'être en 4n+1 [15] ou en 2n+1 classes [22]. Se trouvant devant ces deux possibilités nous avons décidé de considérer les deux cas, c'est à dire: u divisée en sept classes et en treize classes (étant donné que nous avons n=3). A noter que ceci engendre l'etude de deux contrôleurs neuro-linguistiques différents dont les performances seront déterminées par les essais de simulation.

V.3. Choix de la loi de commande:

La loi de commande consiste en une liste de décisions à appliquer dans tous les cas qui peuvent se présenter au contrôleur. Etant donné que e et Δe sont divisées en sept classes chacunes, alors quarante neuf cas sont à étudier. Les règles de décision sont de la forme:

Si e égale ... Et Δe égale ... Alors u à appliquer est ...

En réalité la plus part des cas qui se présenteront au correcteur ne sont pas présents dans la liste des règles de décision. C'est à ce niveau qu'apparait la puissance du réseau neuronal qui doit prendre une décision dans ces situations auxquelles il n'a pas été confrontées lors de son apprentissage.

En pratique on exprime la loi de commande sous forme d'une table [18],[21],[22] où chaque cas constitue une combinaison possible des valeurs que peuvent prendre e et Δe. Pour réaliser une telle table il est nécessaire d'avoir une connaissance qualitative du processus à commander. Dans notre cas, nous sommes partis du principe que si la commande augmente (l'energie en entrée du système augmente) alors il en est de même pour la sortie. En effet, un grand nombre de systèmes répondent à cette description qualitative (moteurs electriques, bras de robots, systèmes thermiques,...). Partant de ce principe et de l'existance de deux possibilités quant à la division de u en classes, on procède à la construction des tables de décision.

1^er^ cas: Division de u en sept classes:

La dénomination des classes est identique à celle donnée pour les classes de e et Δe. Voici la table de décision obtenu en utilisant la méthode de subdivision sur le plan de phase [22]:

Δe / e	NB	NM	NS	ZR	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	ZR
NM	NB	NB	NB	NM	NS	ZR	PS
NS	NB	NB	NM	NS	ZR	PS	PM
ZR	NB	NM	NS	ZR	PS	PM	PB
PS	NM	NS	ZR	PS	PM	PB	PB
PM	NS	ZR	PS	PM	PB	PB	PB
PB	ZR	PS	PM	PB	PB	PB	PB

Table A: Règles de Decision pour commande de sept classes

En observant la table on peut remarquer qu'elle se présente sous la forme d'une matrice antidiagonale et antisymétrique. A noter que pour la construction de cette table on a imposé que le système constitué par le contrôleur neuro-linguistique et le processus à commmander lorsqu'il est bouclé ait une réponse du type 1er ordre, ce qui justifie la forme antidiagonale de la table de décision. En effet, pour un système du premier ordre e et Δe sont liées par une loi linéaire [22].

2ème cas: Division de u en quinze classes.

Dans ce cas on est emmené à nommer de nouveaux représentants de classes. En effet, ceci revient à diviser chacune des classes précedemment utilisées en deux sous classes chacunes excepté la classe ZR qui voit son étendue diminuée.

Exemple: NB est divisée en NB1 et NB2.

NM est divisée en NM1 et NM2.

.
.

ZR reste telle quelle.

.
.
.

PB est divisée en PB1 et PB2.

La table qui découle de ce découpage est la suivante:

	NB	NM	NS	ZR	PS	PM	PB
NB	NB2	NB1	NM2	NM1	NS2	NS1	ZR
NM	NB1	NM2	NM1	NS2	NS1	ZR	PS1
NS	NM2	NM1	NS2	NS1	ZR	PS1	PS2
ZR	NM1	NS2	NS1	ZR	PS1	PS2	PM1
PS	NS2	NS1	ZR	PS1	PS2	PM1	PM2
PM	NS1	ZR	PS1	PS2	PM1	PM2	PB1
PB	ZR	PS1	PS2	PM1	PM2	PB1	PB2

Table B: Règles de décision pour commande de treize classes.

Cette table a pratiquement la même forme que la précédante. La seule différence est que pour deux cases adjacentes (deux situations trés semblables) la décision n'est jamais la même (ce qui n'est pas le cas pour la première table). Le critère de décision parait plus sélectif que le précedant. Seulement, et à ce stade, aucune conclusion sur les performances relatives de ces deux correcteurs ne peut être tirée.

V.4. Architecture du réseau neuronal:

Le réseau de neurones va subir un apprentissage basé sur la table de la loi de commande. Son architecture doit être choisie de telle manière à ce qu'elle soit adaptée à l'apprentissage du réseau et à son intégration dans une structure de réglage.

En entrée le réseau reçoit l'information sur l'erreur et sur sa dérivée. Etant donné que le réseau accepte des entrées de type continue, deux possibilités sont à étudier:

1- Utiliser e et Δe en entrée du réseau sous leur forme qualitative.

2-Codifier e et Δe sous forme d'une liste de sept valeurs où chacune representera le degré d'appartenance de la variable linguistique à chacune des classes. La codification peut être réalisée là aussi par un réseau de neurones.

En sortie, le réseau délivre le signal de commande. Ici, deux possibilités s'offrent à nous:

1-Laisser la commande telle quelle, sous forme d'une valeur

explicite.

2-Codifier la commande sous forme d'une liste de valeurs (de sept ou treize, selon le choix de division de u en classes). En effet, cette architecture originale permet pour chaque valeur de u de connaitre son degré d'appartenance à chaque classe. Cependant, un neurone supplémentaire en sortie est nécessaire pour calculer la valeur éffective de la commande qui sera appliquée au système. La loi permettant de calculer u est donnée par:

$$u = \frac{\sum_{i=1}^{n} B(i) * R(i)}{\sum_{i=1}^{n} B(i)} \qquad (5.1)$$

où: n est le nombre de classes de u.

B(i) est le degré d'appartenance de u à la $i^{ème}$ classe.

R(i) est le representant de la classe i.

En général, il faut fournir au contrôleur linguistique les fonctions d'appartenance (membership functions) pour fuzzifier les variables linguistiques. Dans notre cas, on ne fournira aucune information sur la fuzzification au réseau de neurones, celle-ci sera déterminée par le réseau lui même. Elle sera codifiée sous forme d'une représentation interne au niveau des couches cachées [6].

V.5. Structure de réglage:

La structure de réglage adoptée est illustrée sur la figure 5.1 [19],[22]. A noter les trois paramètres G_u, G_e, $G_{\Delta e}$ qui sont des gains variables. Leur influence sur les performances du contrôleur neuro-linguistique est considérable car ils apportent des corrections sur les erreurs commises dans l'estimation des intervalles de variations des variables linguistiques lors de leur normalisation. A ce jour, il n'existe pas de méthode établie qui puisse en faire la détermination [19]. L'expérimentation reste le seul moyen pour pouvoir leur attribuer des valeurs convenables.

V.6. Apprentissage des réseaux multicouches:

Etant donné les différentes approches pour le choix des entrées/sorties du réseau (citées dans le paragraphe IV), des compositions de ces configurations vont engendrer différents réseaux que nous allons étudier dans ce qui suit.

V.6.1. Réseaux délivrant une commande codifiée:

V.6.1.1. Réseau pour commande de sept classes à entrées non codifiées: Réseau A.

Les entrées du réseau sont de type continue et fournies par

la structure de réglage. L'apprentissage est effectué grace à la table A. Plusieurs tentatives d'apprentissage nous ont permis de déduire qu'une architecture comprenant une seule couche cachée est incapable d'approximer correctement la loi de commande, et qu'augmenter le nombre de couches cachées devenait nécessaire. La configuration finale choisie assure une erreur moyenne de 1.10^{-2} par composante en sortie.

V.6.1.2. Réseau pour commande de treize classes avec entrées non codifiés: Réseau B.

Dans ce cas aussi une structure comportant une seule couche cachée est insuffisante. La configuration adoptée assure comme pour le réseau précedent une erreur de 1.10^{-2} par composante en sortie.

La valeur de η utilisée est assez petite, elle présente les avantages suivants:

-Suppression des oscillations de l'erreur qui ce sont montrées assez fréquentes sur les réseaux pour des valeurs de η situées autour de 0.5 (valeur couramment utilisée).

-Un apprentissage éffectué avec un aussi petit pas de calcul assure une grande précision dans la détermination des poids synaptiques du réseau, et donc, lors de sa phase d'exploitation, le réseau interpolera les cas non appris avec une grande précision [6].

Cependant, les petites valeurs de η présentent un inconvénient certain. En effet, le temps nécessaire à l'apprentissage augmente d'une façon dramatique lorsque η diminue. Il y a là un compromis à trouver.

V.6.1.3. Réseau pour commande de sept classes avec entrées codifiées: Réseau C.

Ce réseau est entrainé grace à la table A. Son interface d'entrée est choisie de telle sorte à pouvoir recevoir les informations des deux autres réseaux dont le rôle est de codifier les valeurs prises par e et Δe. Ce réseau présente l'avantage d'être plus général que les précedents. Son architecture lui confère une plus grande souplesse d'utilisation.

V.6.1.4. Commentaires sur les moyens de calculs:

Nous nous sommes rapidement rendu compte qu'un ordinateur de type PC ne pouvait éffectuer l'apprentissage de l'un de ces réseaux, dans un délai raisonnable. L'utilisation d'un moyen de calcul beaucoup plus puissant s'est avéré indispensable. En ce qui nous concerne, nous avons effectué les calculs sur un ordinateur de type VAX 8650 qui a montré tout au long de son exploitation d'excellentes capacités quant au déroulage de l'algorithme de backpropagation, même quand les réseaux

devenaient volumineux (multitudes de couches et de neurones par couches).

En général, pour les réseaux entrainés avec backpropagation le problème du moyen de calcul se pose avec acuité. Cela est du au fait que cet algorithme est basé sur la méthode de descente du gradient (voir chapitre II), dont la convergence devient trés lente quand des centaines de poids synaptiques sont à déterminer.

V.6.2. Réseaux délivrant une commande non codifiée:

Ce type de réseau consiste à calculer directement la commande éffective à appliquer au système, sans aucune codification, en une valeur comprise dans l'intervalle [-1,1]. En effet la fonction d'activation devra prendre ses valeurs dans l'intervalle [-1,1] au lieu de [0,1]. Une modification de la fonction d'activation du neurone est nécessaire. La nouvelle fonction d'activation devra possèder les propriétés suivantes:

-Fonction définie sur $\mathbb{R}$ et prenant ses valeurs dans [-1,1].

-Fonction non linéaire (pour garder le caractère non linéaire du neurone).

-Fonction non decroissante.

-Fonction différentiable est saturable de préférence.

Parmi les fonctions qui répondent à cette description: la fonction tangente hyperbolique [9]. Son expression est donnée par:

$$F(x) = \frac{1-\exp(-\alpha x)}{1+\exp(-\alpha x)} \qquad (5.2)$$

Le paramètre peut être choisi aléatoirement dans l'intervalle [0,] et influe sur la pente de la fonction.
Deux réseaux basés sur cette approche sont discutés dans la suite.

V.6.2.1. Réseau pour commande à sept classes avec entrées non codifiées: Réseau D.

Une structure comprenant une seule couche cachée s'est révelée suffisante pour implémenter cette commande. L'élimination de la codification de la commande a simplifié la structure interne et le temps nécessaire à l'apprentissage(voir tableau C).

V.6.2.2. Réseau pour commande à treize classes avec entrées non codifiées: Réseau E.

De la même manière que ci-dessus, plusieurs essais nous ont permis de constater l'efficacité d'une couche cachée unique.

Il apparait donc que ces réseaux possèdent une architecture plus simple que ceux utilisant une codification de la commande. Cependant, ces réseaux ne peuvent que determiner le signal de commande et sont incapables de nous renseigner sur la fuzzification de la variable linguistique u établie par le

contrôleur neuro-linguistique lui-même.

V.6.2.3. Problèmes rencontrés lors de l'apprentissage:

Le choix d'une nouvelle fonction d'activation a causé l'apparition d'un phénomène de paralysie lors de l'apprentissage de ces deux réseaux. Il est caractérisé par la stabilisation du réseau dans un état qui ne correspond pas au minimum de l'erreur bloquant ainsi l'algorithme. Deux solutions sont envisagées pour surmonter ce phénomène:

1-Opérer une petite variation sur les poids synaptiques de telle sorte que l'algorithme puisse redémarrer à partir d'un état légèrement différent.

2-Intervenir sur le choix de l'architecture interne du réseau et sur le taux de variation des poids synaptiques η jusqu'a obtention de la combinaison donnant les meilleurs capacités d'apprentissage.

La seconde solution s'est révelée la plus efficace.

V.6.3. Commentaire:

Les différents contrôleurs neuro-linguistiques ne diffèrent à ce stade que par les architectures internes des réseaux, leurs

différentes interfaces d'entrées sorties (codifiées ou non codifiées), les apprentissages des réseaux basés sur différentes règles (sur la table A ou la table B).

Dans la suite du document nous allons nous intéresser à l'application de ces différents contrôleurs à la commande de quelques systèmes. Les simulations permettront alors de déterminer les performances relatives de ces neuro-contrôleurs.

	Réseau A	Réseau B	Réseau C	Réseau D	Réseau E
Itérations	3080	10.000	7000	1700	500

Table C: Nombre d'itérations pour chacun des réseaux entrainés.

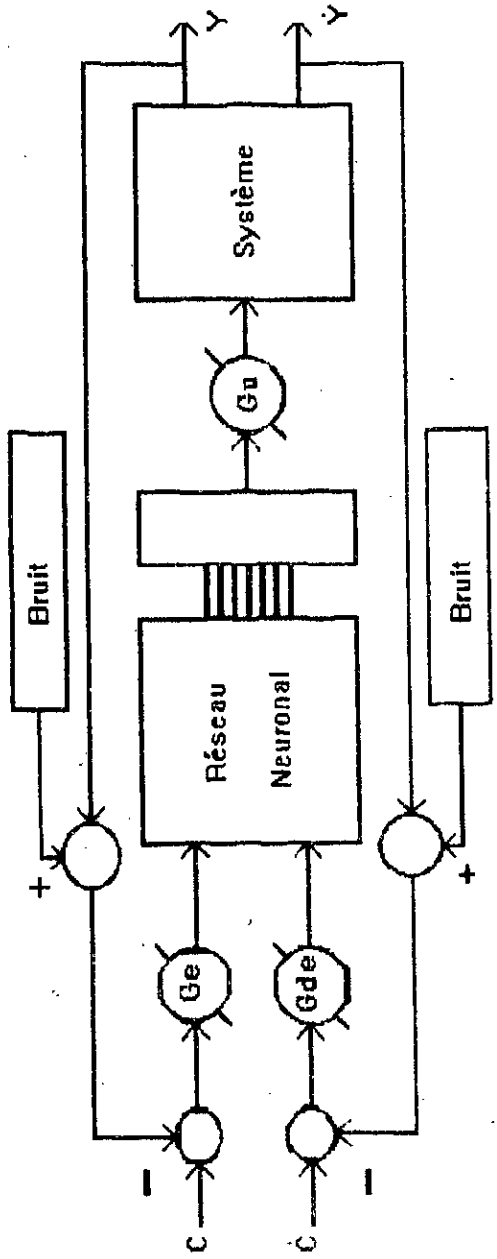

Figure V.1. Structure de réglage avec bruitage.

CHAPITRE VI

APPLICATIONS SISO DU CONTROLEUR NEURO-LINGUISTIQUE

VI.1. Introduction:

Aprés avoir présenté dans le chapitre V le contrôleur neuro-linguistique, nous allons chercher dans ce chapitre la mise en oeuvre de la commande de systèmes de type SISO (linéaires et non linéaires). A cet effet deux exemples ont été choisis: la commande en position d'un moteur à courant continu et la commande du système de Van Der Pol. La soumission des systèmes à des perturbations et des bruitages permettra de mettre en evidence les capacités de notre contrôleur à gérer de telles situations. La structure de réglage utilisée est la même que celle présentée dans le chapitre précédent (voir figure V.1).

VI.2. Application à la commande d'un système linéaire: Commande en position d'un moteur à Courant Continu.

L'objectif de cette partie est de montrer que cette technique de réglage permet de commander un système linéaire monovariable. Notre choix s'est porté sur le moteur à courant continu du fait qu'il constitue un actionneur classique largement utilisé dans un grand nombre de processus automatisés en industrie.

VI.2.1. Modelisation du moteur:

Le schéma synoptique du moteur à excitation séparée est donné sur la figure VI.0.
A flux constant, les équations du moteur sont:

$$V_m = R_a\, I_m + L_a \frac{dI_m}{dt} + E \qquad (6.1)$$

$$E = K\,\Omega \qquad (6.2)$$

$$\Gamma_m = J \frac{d\Omega}{dt} + K_f\,\Omega + \Gamma_c \qquad (6.3)$$

$$\Gamma_m = K\, I_m \qquad (6.4)$$

avec

R_a : *resistance des enroulements d'induit.*

L_a : *Inductance des enroulements d'induit.*

I_m : *Courant d'induit.*

V_m : *tension d'induit.*

E : *force contre electromotrice.*

J : *moment d'inertie des masses tournantes du moteur.*

K_f : *coefficient de frottement visqueu.*

Γ_m : *Couple moteur.*

Γ_c : *Couple resistant.*

K : *Constante dépendante du moteur.*

Si on néglige l'inductance de l'induit et le coefficient de frottement visqueux, on aura:

de (6.1) et (6.2)

$$I_m = \frac{(V_m - K \Omega)}{R_a} \qquad (6.5)$$

de (6.3) et (6.4)

$$\frac{d\Omega}{dt} = - \frac{K^2 \Omega}{J R_a} + \frac{K V_m}{J R_a} - \frac{\Gamma_c}{J} \qquad (6.6)$$

l'équation d'Etat du système s'ecrit:

$$\begin{pmatrix} \dot{\theta} \\ \dot{\Omega} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & - \frac{K^2}{J R_a} \end{pmatrix} \begin{pmatrix} \theta \\ \Omega \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{K}{J R_a} \end{pmatrix} V_m + \begin{pmatrix} 0 \\ - \frac{1}{J} \end{pmatrix} \Gamma_c . \qquad (6.7)$$

avec

$$\Omega = \frac{d\theta}{dt} . \qquad (6.8)$$

Les grandeurs numériques utilisées [25] pour les simulations sont:

K=1.2; R_a= 0.4 Ohm; J= 0.06 $K_g.m^2$; U_n= 110 Volts; I_n= 32 A;

Ω_n= 1500 tr/min; Γ_n= 38.4 N.m.

VI.2.2. Simulation du moteur en boucle fermée:

Les gains G_u, G_e et $G_{\Delta e}$ ont été fixés aux valeurs optimales qui permettent d'obtenir des réponses convenables, pour tous les essais de simulation.

VI.2.2.1. Simulation du moteur sans perturbation:

Les figures 6.1 et 6.2 illustrent respectivement la réponse indicielle du moteur et la commande correspondante. Le contrôleur permet d'obtenir une réponse indicielle en boucle fermée sans dépassement, un temps de réponse de l'ordre de 25 ms et une erreur statique en régime établie pratiquement nulle. Au démarrage, la commande est positive saturée pour que le moteur réponde le plus rapidement possible. Dès que l'écart entre la réponse et la consigne devient faible, le signal de commande est négatif saturé pour freiner la course du moteur. Quand la consigne est atteinte, le signal de commande décroit en oscillant jusqu'à s'annuler.

VI.2.2.2. Simulation du moteur avec perturbation:

La perturbation consiste en un couple resistant $\hat{O}_c$ dont l'expression analytique est la suivante:

$$\Gamma_c(t) = \Gamma_n \exp -\frac{(t-m)^2}{\sigma^2} \qquad (6.9)$$

avec

δ_n= 38.4 N.m ; m=0.23 s ; δ= 0.01 s.

Les figures 6.3 et 6.4 illustrent respectivement la réponse indicielle et la commande correspondante. La réponse indicielle est identique à celle du 2.2.1. Le contrôleur a donc rejeté totalement l'effet de la perturbation. Le signal de commande est aussi identique à celui du 2.2.1, sauf qu'au voisinage de t=0.23 où le couple résistant est à son maximum, il augmente et prend la forme de celui-ci pour en compenser l'effet.

VI.2.2.3. Test de robustesse au bruit:

On considère que les signaux délivrés par les capteurs sont bruités de façon aléatoire. Le but de cet essai est de tester la robustesse du contrôleur vis à vis des bruits introduits par les capteurs. La structure de réglage adoptée est donnée par la figure V.1.

A-Bruitage des sorties des capteurs avec un bruit alèatoire de 2 % :

La figure 6.5 illustre la réponse indicielle du moteur. L'effet du bruitage est pratiquement inexistant. Cependant, on peut

remarquer l'apparition d'une erreur statique de 0.17 % . Sur la figure 6.6, on peut voir que le signal de commande pendant le régime transitoire est identique au cas sans bruitage alors qu'en régime établie il ne décroit pas vers zéro mais oscille avec une amplitude alèatoire pour compenser les dérives introduites par le bruit.

B-Bruitage des sorties du capteur avec un bruit alèatoire de 5%:

Les figures 6.7 et 6.8 illustrent respectivement la réponse indicielle et la commande correspondante. On remarque que la dynamique du système est conservée. Cependant l'erreur statique a augmenté et atteint 0.208 % .

La commande possède la même allure par rapport au cas précedant alors que les amplitudes de ses oscillations sont plus importantes du fait que l'amplitude du bruit est plus grande.

VI.2.2.4. Simulation du moteur avec paramètre variable:

Pour mettre en évidence un autre aspect de la robustesse du contrôleur, on a simulé la réponse du moteur en boucle fermée avec un paramètre variable qu'est la résistance de l'induit. Pour cela, on a pris une résistance d'induit variable dans le temps suivant la loi:

$R_a(t)=0.4+5*t$ Ohm avec $t \in [0,0.1]$ et $R_a=0.9$ Ohm pour $t > 0.1$.

Sur la figure 6.9 on peut voir que la réponse indicielle du moteur à sensiblement changé: On a un dépassement de 2%, un temps de réponse de 24.5 ms et un écart en régime permanent de 0.74 ‰ qui va en diminuant.

Sur la figure 6.10, le signal de commande possède la même allure que celui d'un moteur sans paramètre variable (figure 6.2), alors que le temps de réglage est plus grand (la commande met plus de temps à s'annuler).

VI.3. Application à un système non linéaire: Système regit par l'équation de Van der Pol.

Cette équation décrit les oscillateurs à relaxation [26]. Elle représente un système du deuxième ordre avec amortissement non lineaire. elle a la forme suivante:

$$\frac{d^2x}{dt^2} - \alpha\,(1-\beta\,x^2)\,\frac{dx}{dt} + x = u(t)\,. \qquad (6.10)$$

L'équation d'état du système est:

$$\frac{dx_1}{dt} = x_2\,. \qquad (6.11)$$

$$\frac{dx_2}{dt} = \alpha\ (1-\beta\ x_1^2)\ x_2 - x_1 + u(t)\,. \qquad (6.12)$$

avec:

$$x_1 = x.$$
$$x_2 = \frac{dx}{dt}. \qquad (6.13)$$

VI.3.1. Simulation du système en boucle ouverte:

Pour la simulation, on a pris Ó=2 et ß=0.5.
La réponse indicielle du système est donnée par la figure 6.11. C'est un cycle limite d'amplitude 3.08 et de fréquence 0.11 Hz.

VI.3.2. Simulation du Système en boucle fermée:

L'ajustement des paramètres G_u, G_e et G_e permet d'obtenir la réponse indicielle illustrée par la figure 6.12. Le signal de commande correspondant est donné par la figure 6.13. Ces paramètres ne seront plus modifiés par la suite pendant tous les essais de simulation.
Le système bouclé est stable et présente un temps de réponse de 0.15 s sans depassement. Le signal de commande oscille avec une amplitude variable entre U_{max} et U_{min}.

VI.3.3. Test de robustesse au bruit:

A- Bruitage des sorties des capteurs avec un bruit de 2% :

Sur la figure 6.14, on peut voir la réponse indicielle du système.
Elle présente un écart permanant de 0.4 % . En revanche, le temps de réponse n'a pas changé. Le signal de commande correspondant est illustré par la figure 6.15. Il est pratiquement identique à celui de la figure 6.13. Du fait que la commande est oscillante, les changements introduits pour l'elimination du bruit ne sont pas visibles.

B- Bruitage des sorties des capteurs avec un bruit de 5% :

La réponse indicielle du système est illustrée par la figure 6.16. L'écart permanent est plus important et atteint 1.3 % du fait que l'amplitude du bruit est plus grande. Le temps de réponse reste inchangé. Le signal de commande correspondant (figure 6.17) semble inchangé par rapport à celui de la figure 6.15.

VI.4. Conclusions:

Les Simulations ont montré que le contrôleur neuro-linguistique est capable de commander un système instable en

boucle ouverte qu'il soit linéaire ou non-linéaire. Il a aussi la capacité de générer un signal de commande adapté qui permet un rejet total d'une perturbation, la suppression dans une certaine mesure d'un bruitage en entrée du correcteur et la compensation de dérives introduites par un paramètre variable.

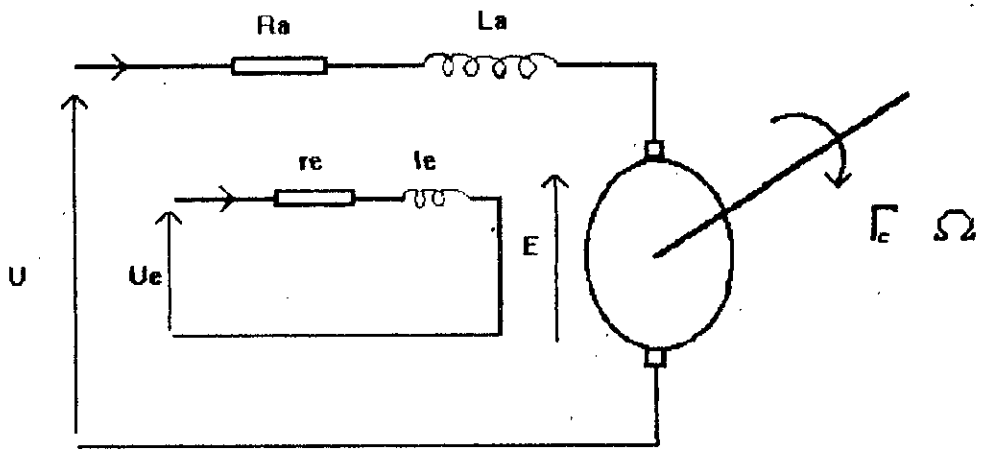

Figure VI. 0. Schéma synoptique du moteur.

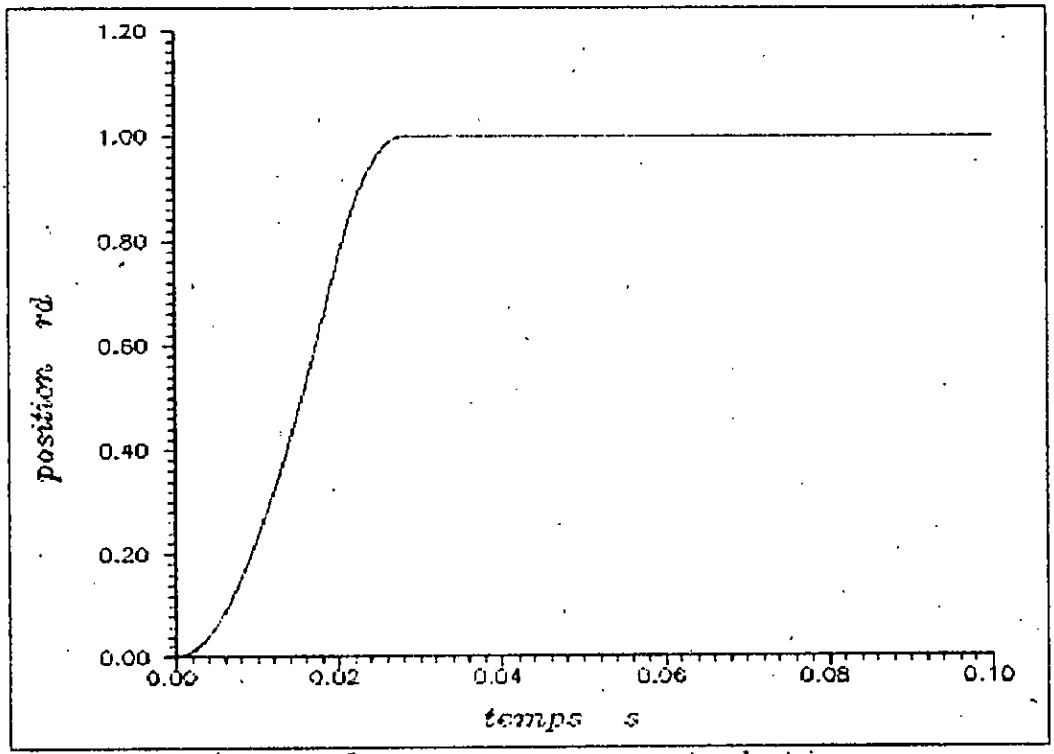

Figure1. Réponse du moteur sans perturbation.

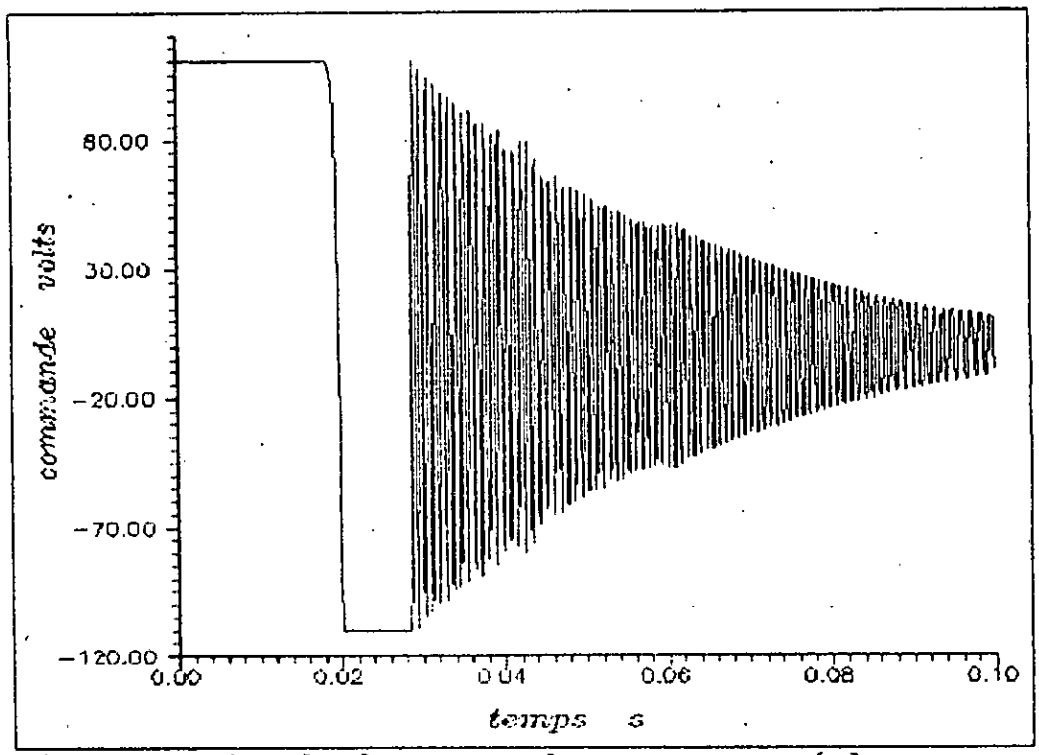

Figure2. Signal de commande pour un réglage sans perturbation.

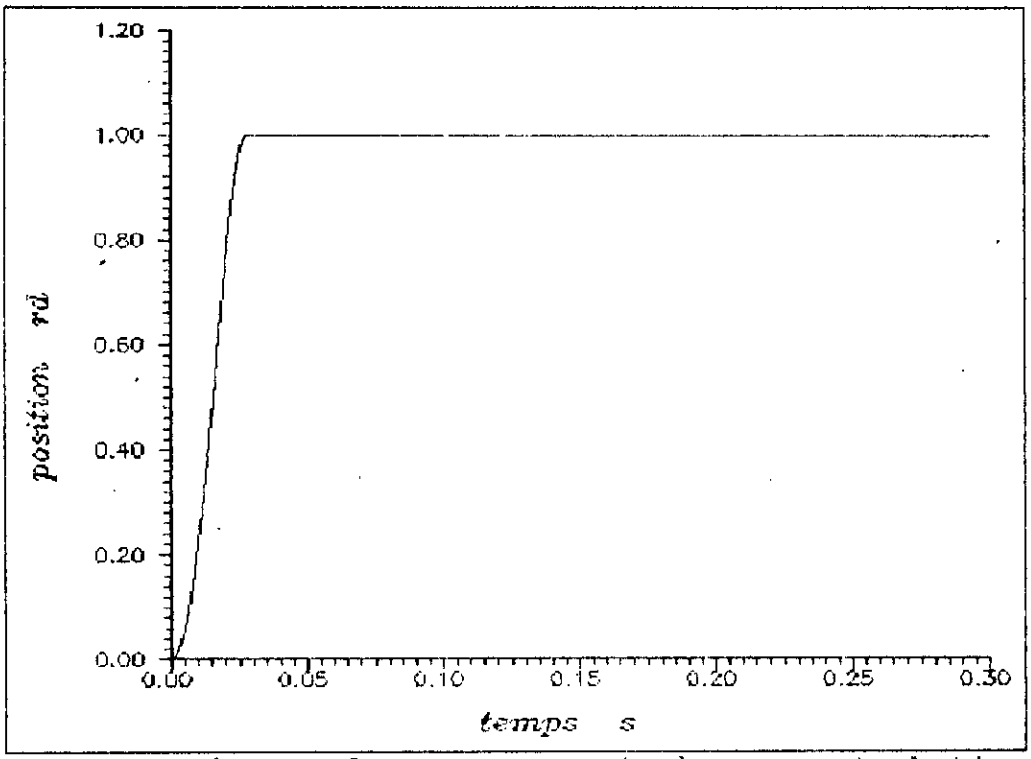

Figure3. Réponse du moteur soumis à une perturbation nominale.

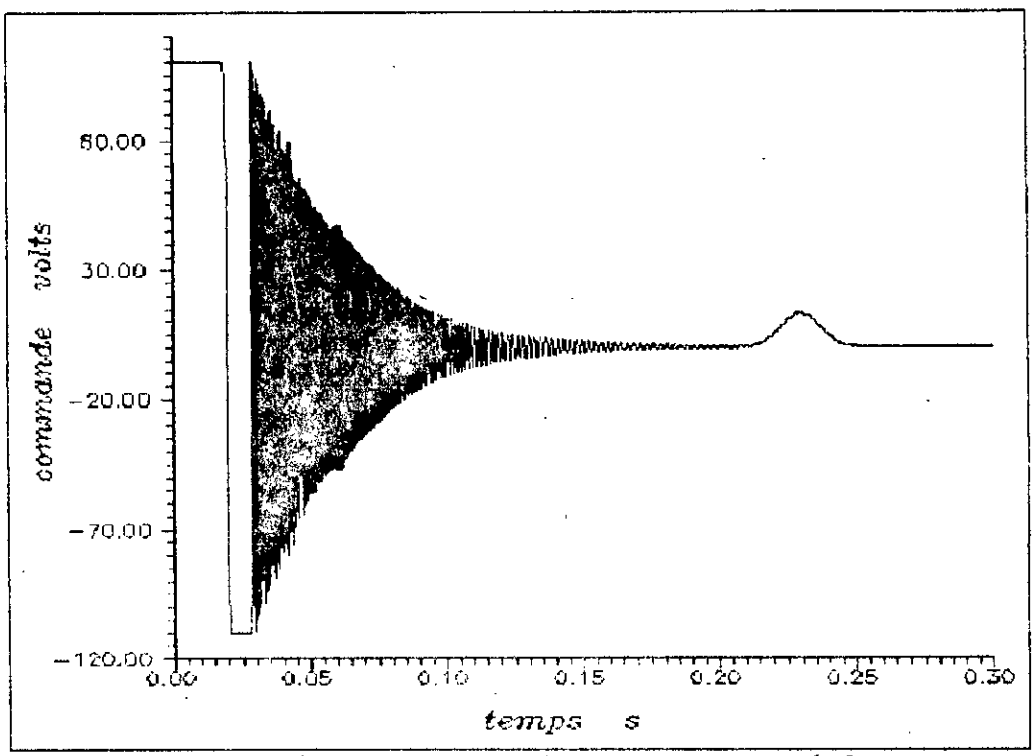

Figure4. Signal de commande pour un réglage avec perturbation.

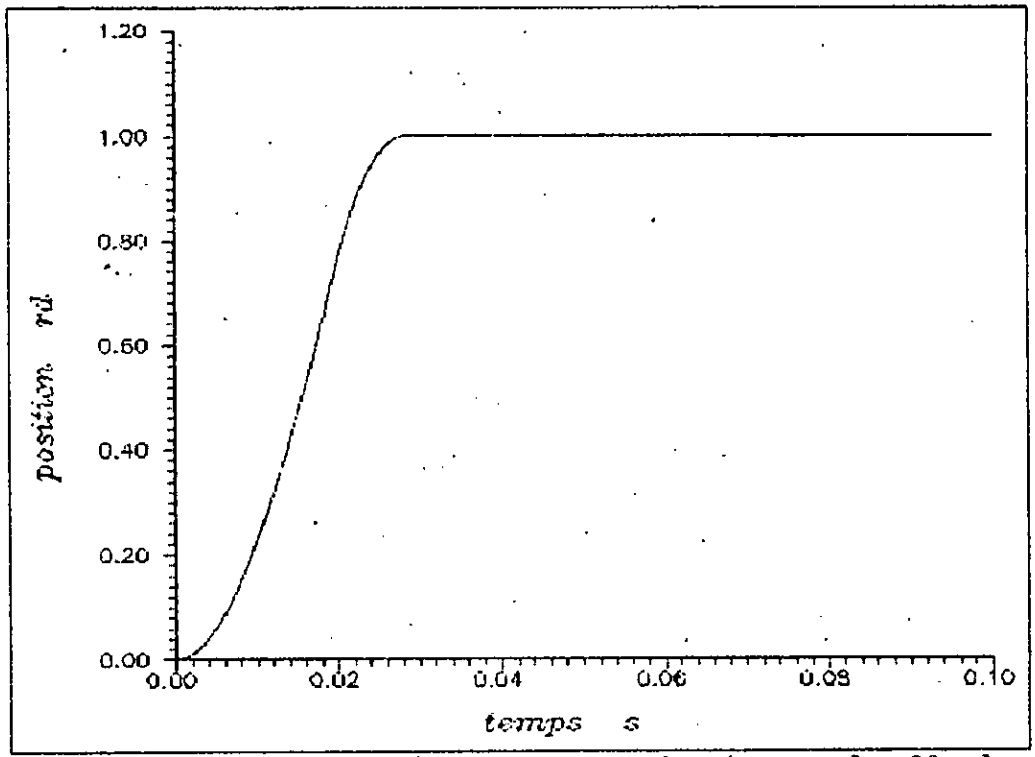

Figure5. Réponse du moteur avec bruitage de 2% des entrées du correcteur.

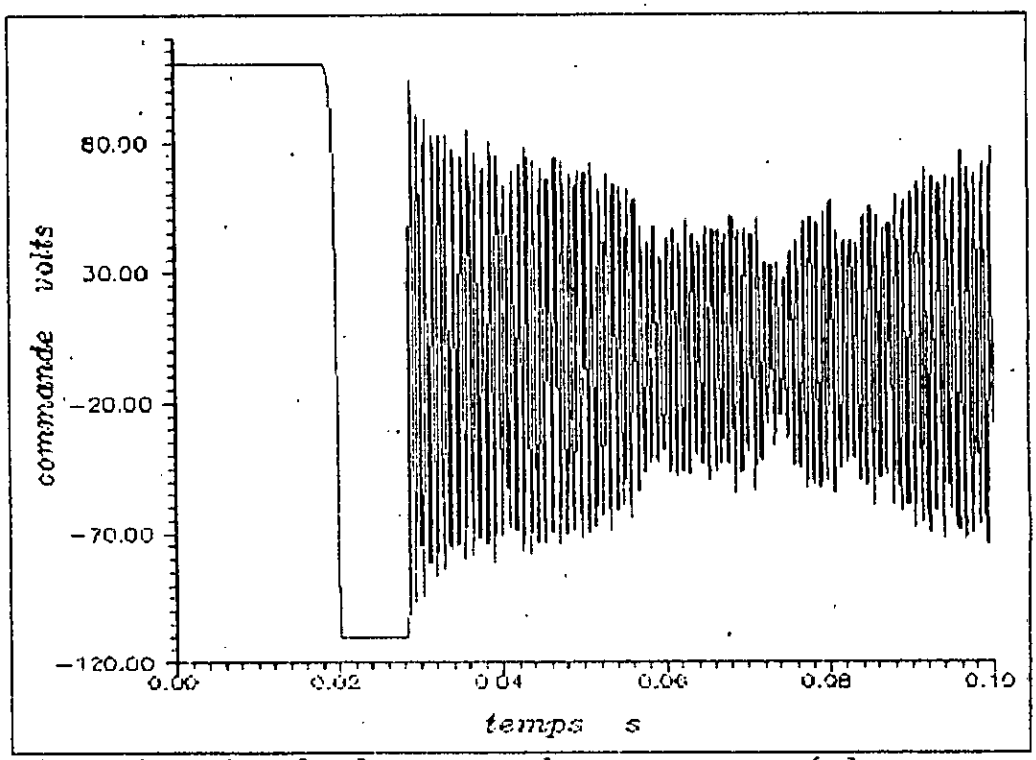

Figure6. Signal de commande pour un réglage avec bruitage de 2% des entrées du correcteur.

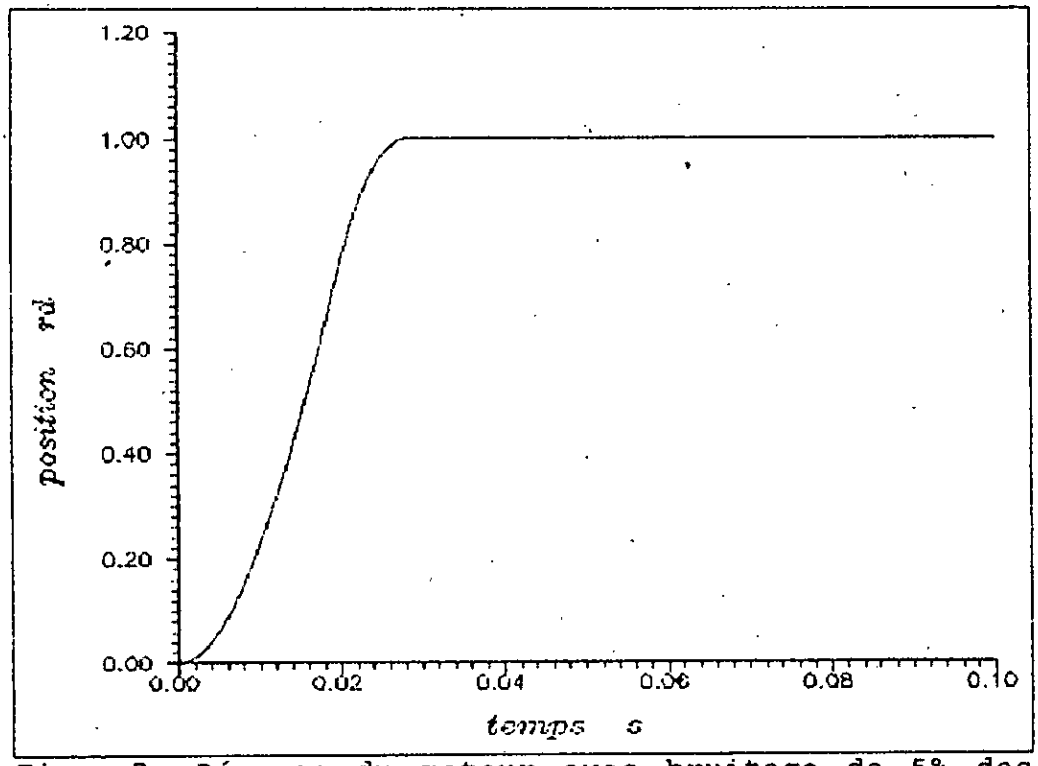

Figure7. Réponse du moteur avec bruitage de 5% des entrées du correcteur.

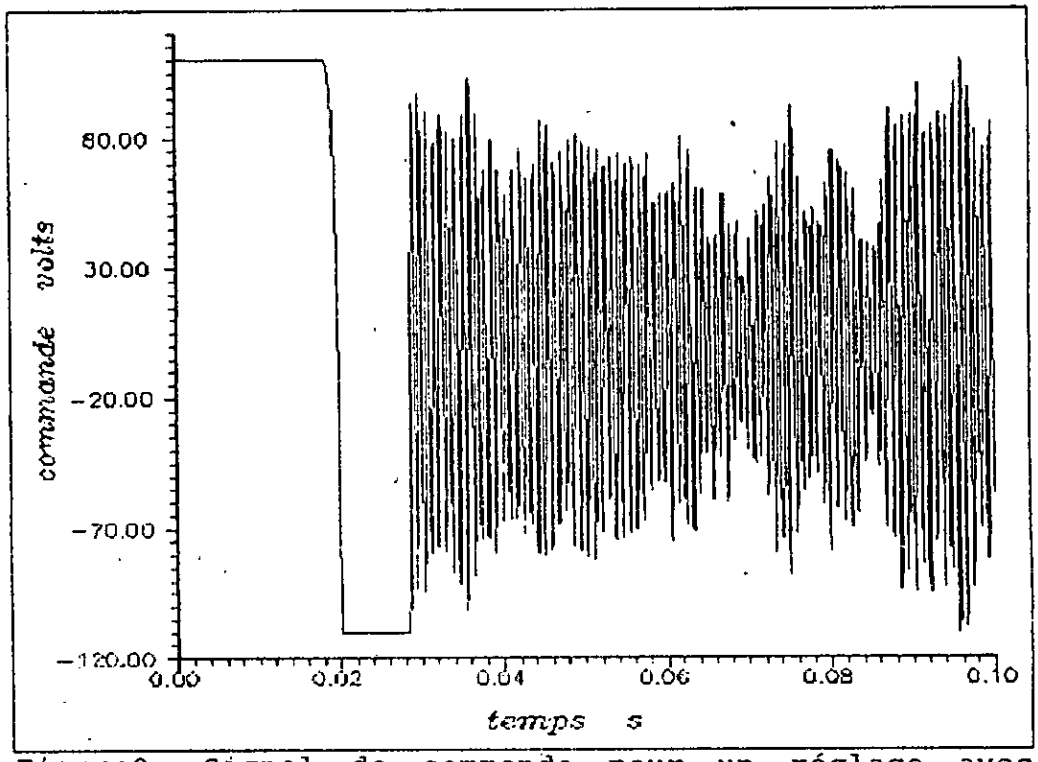

Figure8. Signal de commande pour un réglage avec bruitage de 5% des entrées du correcteur.

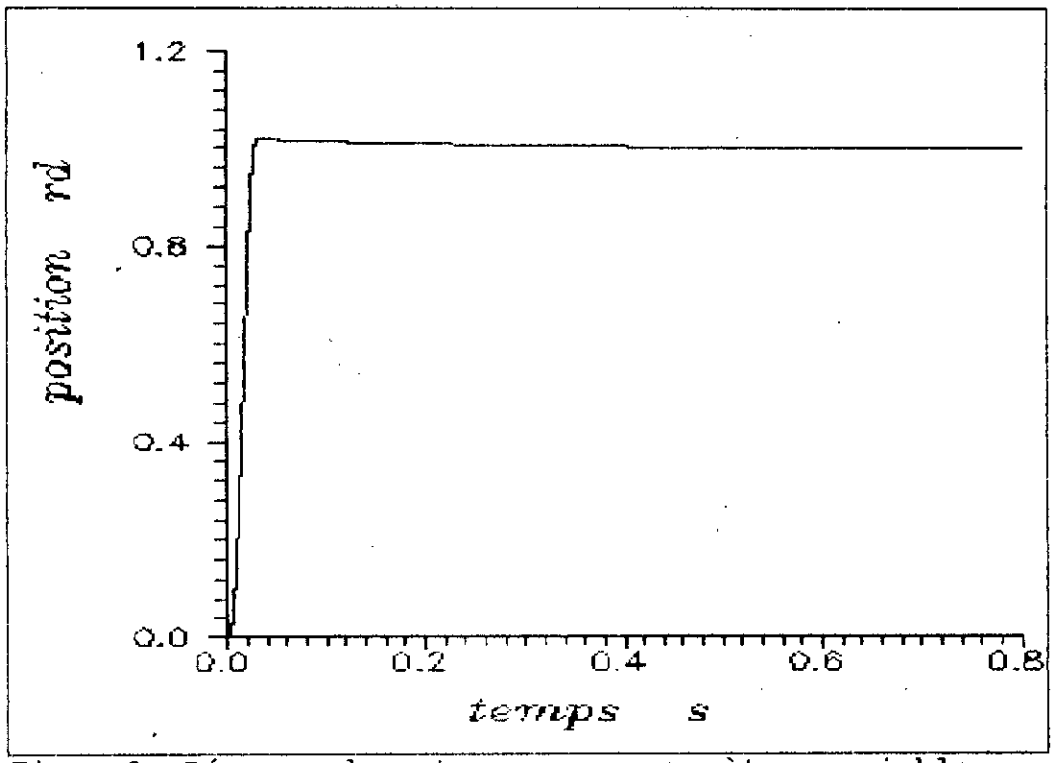

Figure9. Réponse du moteur avec paramètre variable.

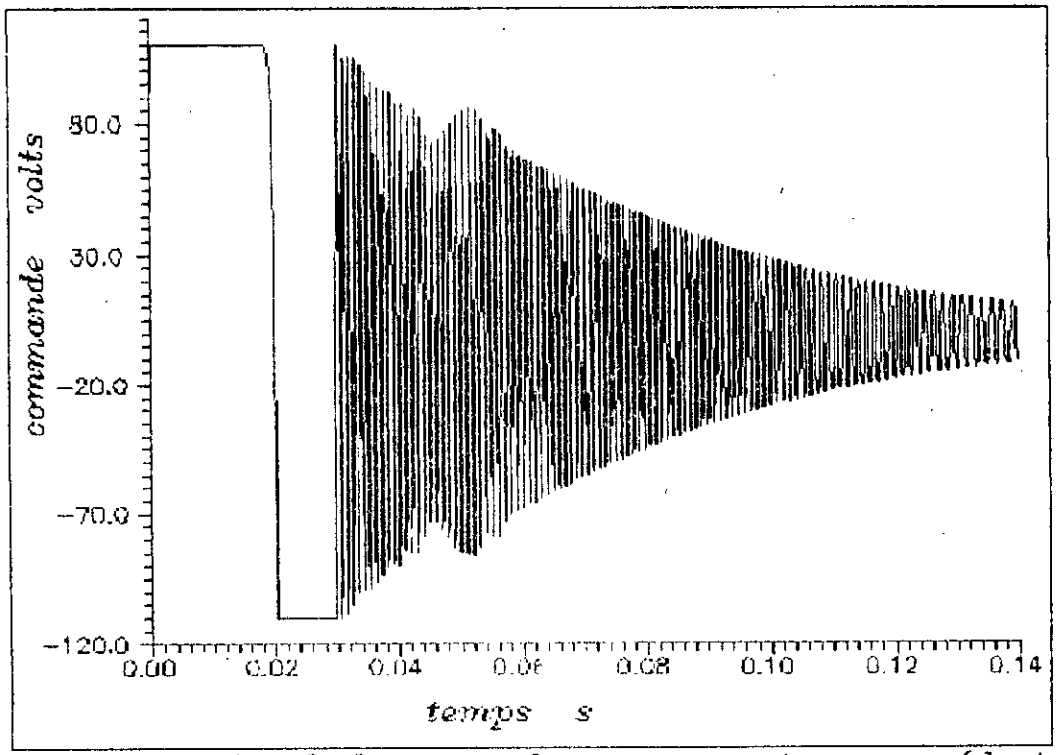

Figure10. Signal de commande pour un moteur possédant un paramètre variable.

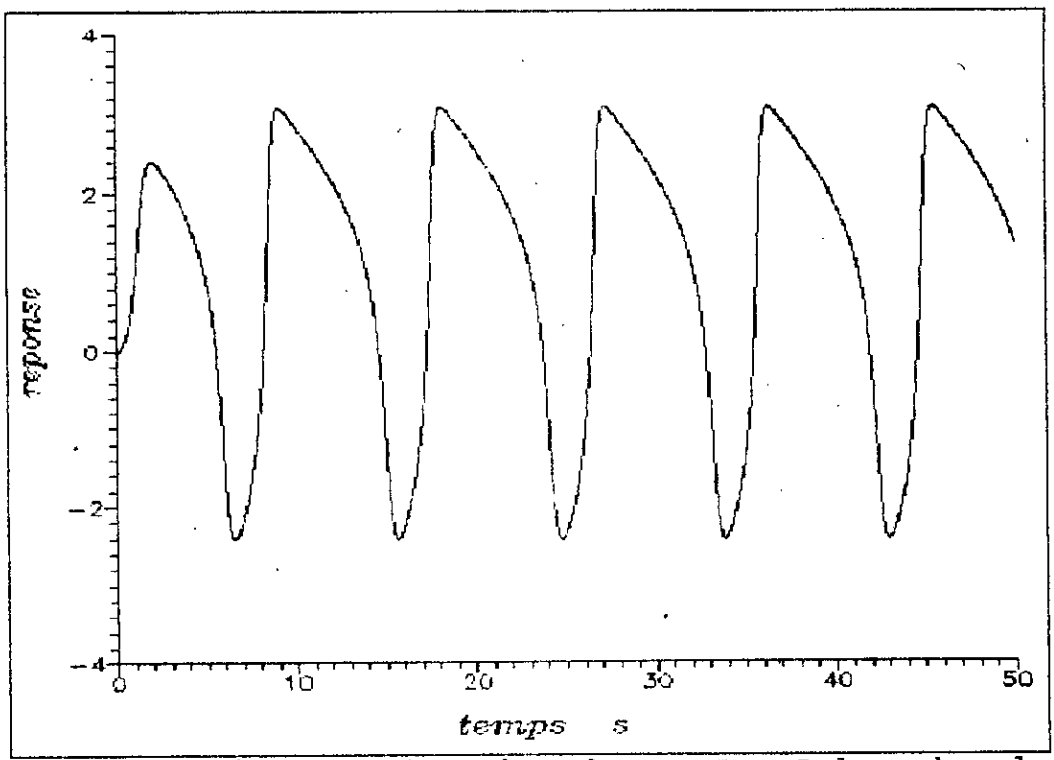

Figure11. Réponse du système de Van Der Pol en boucle ouverte.

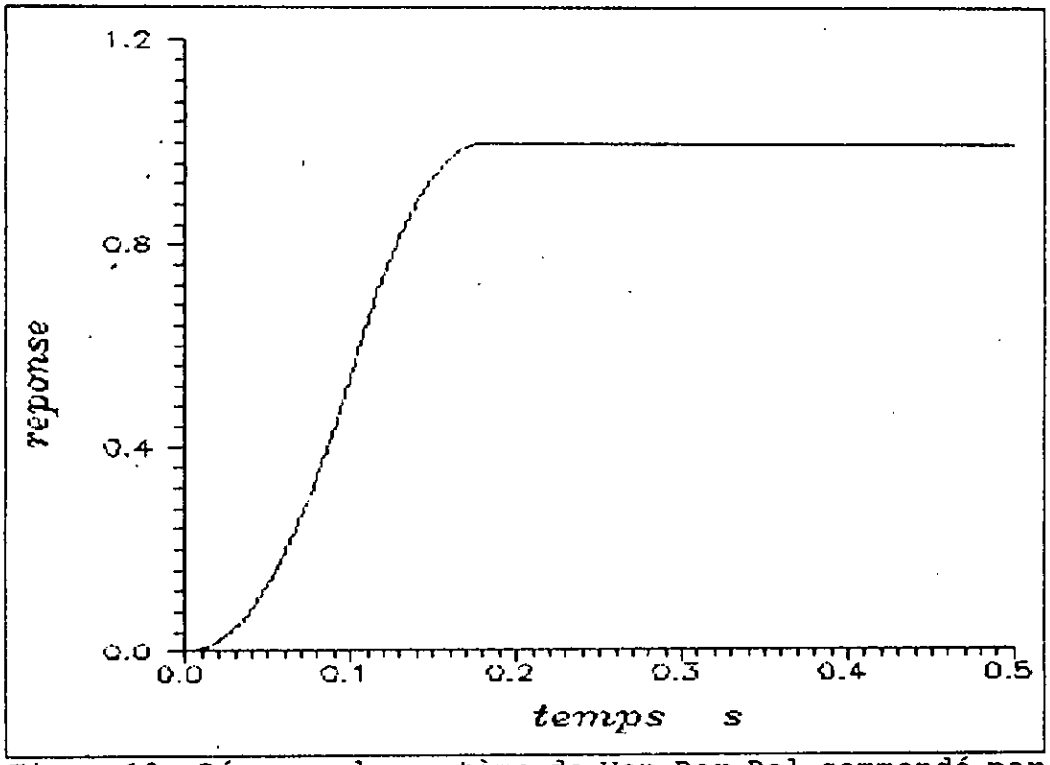

Figure12. Réponse du système de Van Der Pol commandé par le controleur neuro-linguistique.

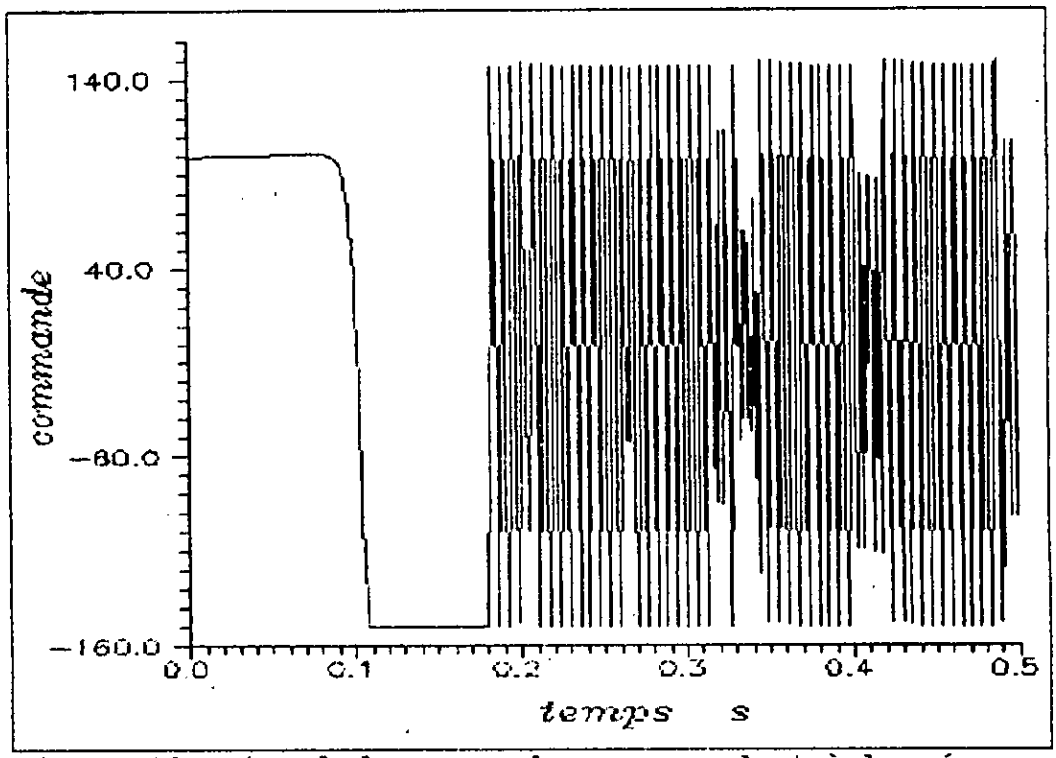

Figure 13. Signal de commande correspodant à la réponse ci-dessus.

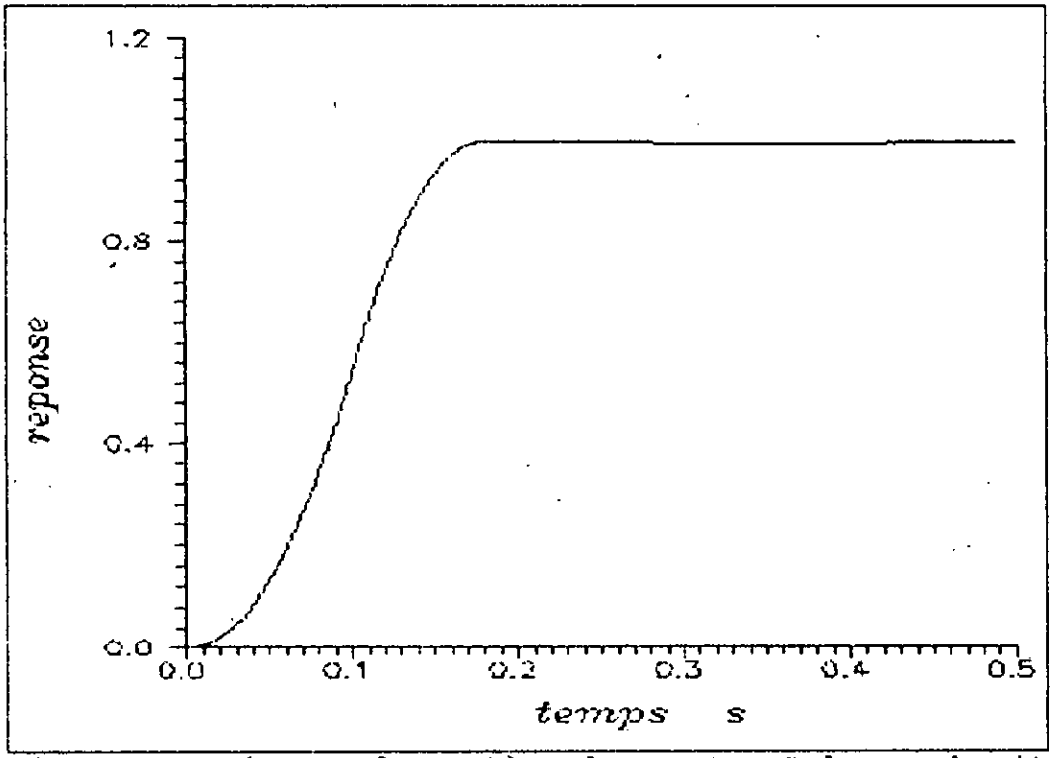

Figure 14. Réponse du système de Van Der Pol avec bruit de 2%.

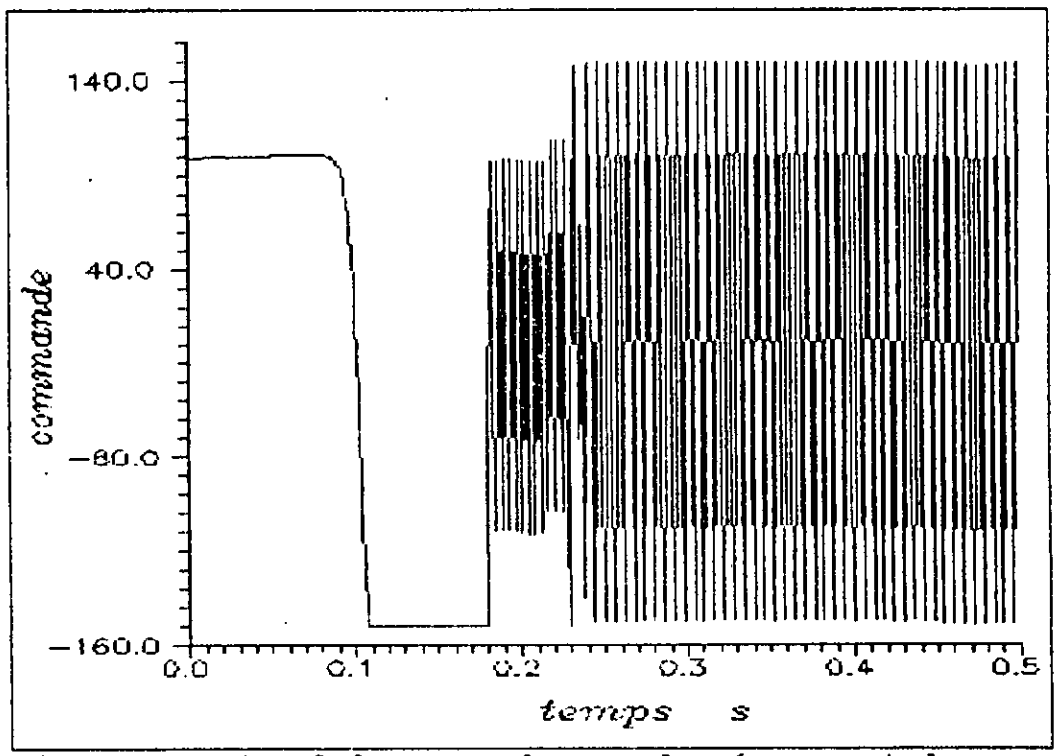

Figure 15. Signal de commande pour la réponse ci-dessus.

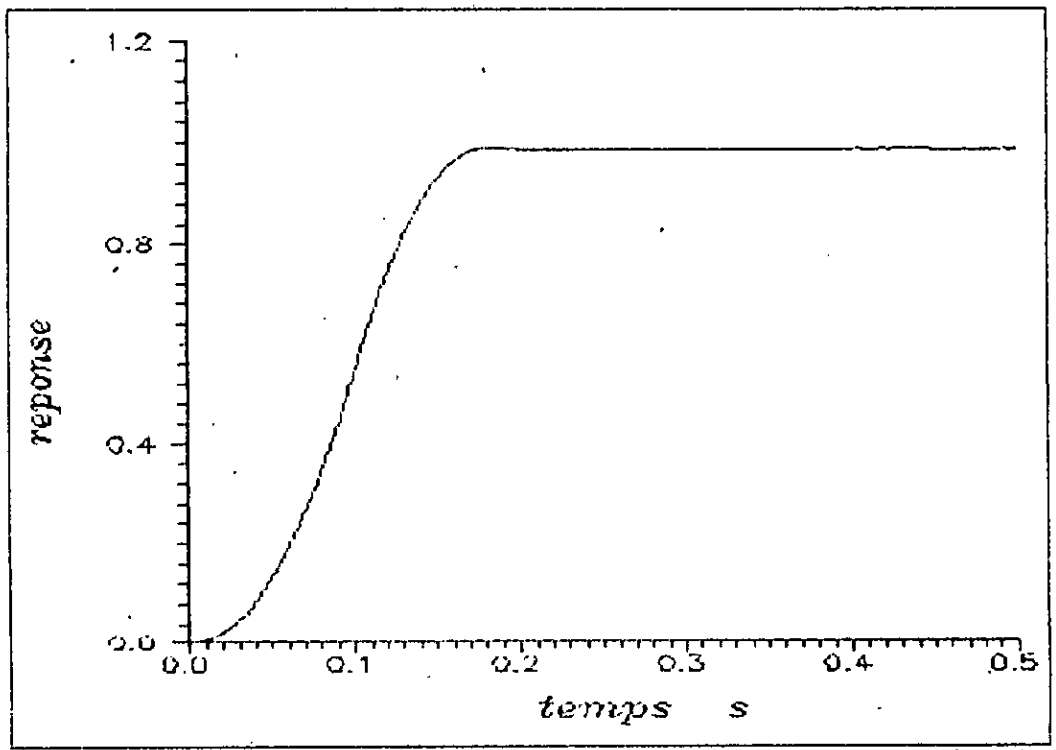

Figure 16. Réponse du système de Van Der Pol avec bruit de 5%.

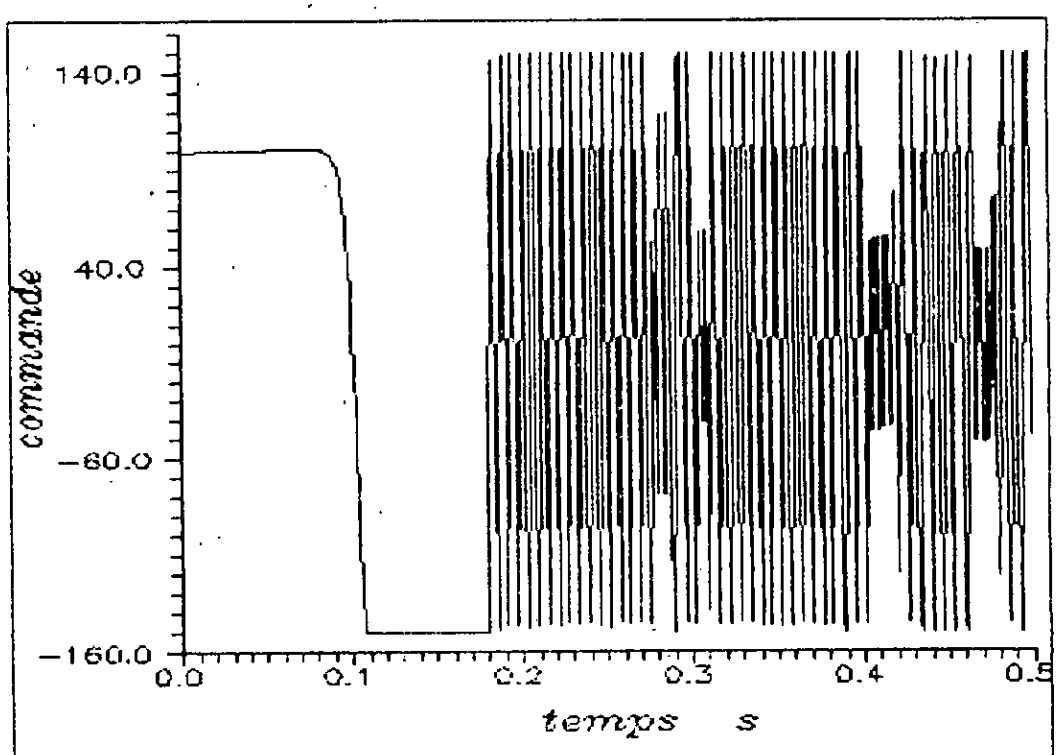

Figure 17. Signal de commande pour la réponse ci-dessus.

CHAPITRE VII

APPLICATION MIMO:

COMMANDE D'UN BRAS MANIPULATEUR

A TROIS DEGRES DE LIBERTE

VII.1. Introduction:

Le domaine de la robotique est en pleine expansion, et on ne compte plus ses applications en industrie (manipulation de produits dangereux, intervention en univers hostile, execution de taches nécessitant rapidité et précision, industrie automobile,...).

Une classification regroupant la majorité des différents types de robots manipulateurs les plus couramment utilisés a été établie et ensuite plus recemment complétée [27]. Cette classification comporte huits types de bras manipulateurs à trois degrés de liberté. Ils ont été classés suivant la nature du mouvement de chaque articulation (translation ou rotation).Nous avons choisi pour application MIMO de notre contrôleur neuro-linguistique la commande d'un robot manipulateur comportant trois articulations éffectuant toutes des mouvements de rotation. Ce choix est justifié par l'un important couplage existant entre les différentes articulations et qui va permettre la mise en évidence de la puissance et des limites d'un contrôleur neuro-linguistique à commander un tel système.

VII.2. Modélisation [27]:

La modélisation du robot manipulateur est basée sur l'application des lois de la mécanique. Pour établir les équations dynamiques du système, on s'appuie sur le formalisme de Lagrange. Le Lagrangien du processus robot est:

$$L=\sum_{i=1}^{3}(T_i-U_i) \qquad (7.1)$$

où: T_i représente l'énergie cinétique de la liaison i.

U_i représente l'énergie potentielle de la liaison i.

L'équation de Lagrange permet d'obtenir les équations différentielles du système:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}_i}-\frac{\partial L}{\partial x_i}=q_i \qquad (7.2)$$

où: q_i représente la force généralisée appliquée à l'articulation i. Elle regroupe la force appliquée par le moteur et une force de résistance (proportionnelle à la vitesse) due aux frottements.

x_i est la position angulaire de l'articulation i.

$$q_i=k_i u_i-c_i\dot{x}_i \qquad (7.3)$$

Les relations (7.1) et (7.2) permettent d'établir l'équation suivante:

$$M\ddot{X} + C\dot{X} + D\dot{X}_c + E\dot{X}^2 + F\,g = K\,u \qquad (7.4)$$

où:

$\ddot{X} = [\ddot{x}_1\ \ddot{x}_2\ \ddot{x}_3]^T$ *est le vecteur accéleration angulaire.*

$\dot{X} = [\dot{x}_1\ \dot{x}_2\ \dot{x}_3]^T$ *est le vecteur vitesse angulaire.*

$\dot{X}^2 = [\dot{x}_1^2\ \dot{x}_2^2\ \dot{x}_3^2]^T$ *est le vecteur acceleration centrifuge.*

$\dot{X}_c = [\dot{x}_1\dot{x}_2\ \dot{x}_1\dot{x}_3\ \dot{x}_2\dot{x}_3]^T$ *est le vecteur accéleration de coriolis.*

Les matrices M, C, D, E, F et K ont la forme suivante:

$$M = \begin{pmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ 0 & 0 & m_{33} \end{pmatrix}$$

avec:

$m_{11}=J_1+J_{x2}+2m_2L_1l_2\cos(x_2)$.

$m_{12}=m_{21}=J_{x2}+m_2L_1l_2\cos(x_2)$.

$m_{33}=J_1\sin^2(x_1)+J_{1p}\cos^2(x_1)+J_{x2}\sin^2(x_1+x_2)+J_{2p}\cos^2(x_1+x_2)$
$+J_3+2m_2L_1l_2\sin(x_1)\sin(x_1+x_2)$.

et

$J_1^*=J_{x1}+m_2L_1^2$.

$J_{xi}=J_i+m_il_i^2$.

$$C = \begin{pmatrix} c_1 & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_3 \end{pmatrix}$$

$$D = \begin{pmatrix} 0 & d_{12} & d_{13} \\ d_{21} & 0 & d_{23} \\ 0 & 0 & 0 \end{pmatrix}$$

avec:

$d_{12}=-m_2L_1l_2\sin(x_2)$.

$d_{13}=-(J_1-J_{1p})\sin(x_1)\cos(x_1)-(J_{x2}-J_{2p})\sin(x_1+x_2)\cos(x_1+x_2)-$
$m_2L_1l_2\sin(2x_1+x_2)$.

$d_{21}=m_2L_1l_2\sin(x_2)$.

$d_{23}=-(J_{x2}-J_{2p})\sin(x_1+x_2)\cos(x_1+x_2)-m_2L_1l_2\sin(x_1+x_2)\cos(x_1+x_2)$.

$$E = \begin{pmatrix} e_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & e_{32} & e_{33} \end{pmatrix}$$

avec:

$e_{11}=-2m_2L_1l_2\sin(x_2)$.

$e_{32}=2(J_1-J_{1p})\sin(x_1)\cos(x_1)+2(J_{x2}-J_{2p})\sin(x_1+x_2)\cos(x_1+x_2)$
$+2m_2L_1l_2\sin(2x_1+x_2)$.

$e_{33}=2(J_{x2}-J_{2p})\sin(x_1+x_2)\cos(x_1+x_2)+2m_2L_1l_2\sin(x_1)\cos(x_1+x_2)$.

$$F = \begin{pmatrix} f_1 \\ f_2 \\ 0 \end{pmatrix}$$

avec:

$f_1 = -(m_1 l_1 + m_2 L_1)\sin(x_1) - m_2 l_2 \sin(x_1 + x_2)$.

$f_2 = -(m_2 l_2 \sin(x_1 + x_2)$.

$$K = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{pmatrix}$$

Ces équations montrent la nature fortement non-linéaire du robot manipulateur. La dynamique de chaque articulation est determinée dans l'equation (7.4) par sa vitesse et son accéleration angulaire. Les sorties qui nous intéressent sont les positions de chaque articulation. Le vecteur sortie est alors:

$$Y = I X$$

où I représente la matrice unité.

A noter que dans toute la suite on utilisera ce modèle pour effectuer les essais de simulation.

Paramètres du modèle [28]:

L_1=1 m. Longueur de la liaison 1.

L_2=1 m. Longueur de la liaison 2.

l_1=0.5 m. Distance de l'extremité de la liaison 1 à son centre de masse.

l_2=0.5 m. Distance de l'extrémité de la liaison 2 à son centre de masse.

m_1=10 kg. Masse de la liaison 1.

m_2=10 kg. Masse de la liaison 2.

J_3=1 kgm². Moment d'inertie de la liaison 3.

J_{x1}=3 kgm². Moment d'inertie de la liaison 1.

J_{x2}=3 kgm². Moment d'inertie de la liaison 2.

J_{1p}=0.01 kgm². Moment d'inertie polaire de la liaison 1.

J_{2p}=0.01 kgm². Moment d'inertie polaire de la liaison 2.

c_1=75 Nms. Coefficient de frottement dans l'articulation 1.

c_2=10 Nms. Coefficient de frottement dans l'articulation 2.

c_3=1 Nms. Coefficient de frottement dans l'articulation 3.

k_1=40 NmV^{-1}. Gain statique de l'actionneur de l'articulation 1.

k_2=20 NmV^{-1}. Gain statique de l'actionneur de l'articulation 2.

k_3=30 NmV^{-1}. Gain statique de l'actionneur de l'articulation 3.

g=9.81 ms^{-2}. Accélération de la pesanteur.

VII.3. Simulation en boucle ouverte:

L'intégration de l'équation d'état du système a été éffectuée avec l'algorithme de Runge-Kutta d'ordre 4. Cette méthode

présente l'inconvénient d'être lente, mais, en contre-partie elle possède une trés bonne précision de calcul.
En boucle ouverte, et en réponse à des echelons unitaires de tension, déux articulations sont stables et la troisième réponse est divergente (Figures 1a, 1b et 1c).

VII.4. Commande en boucle fermée:

VII.4.1. Structure de réglage:

Etant en présence d'un système multivariable, nous devons introduire des modifications dans la structure de réglage précedemment présenté dans le chapitre V.

Le système possédant trois commandes, nous devons donc utiliser trois correcteurs neuro-linguistiques. Ces trois correcteurs sont identiques par leurs structures et leurs apprentissages. Ils interviennent chacun indépendemment l'un de l'autre sur l'une des trois commandes. De ce fait, chaque contrôleur considère les commandes générées par les deux autres correcteurs comme des perturbations qu'il doit rejeter.

VII.4.2. Premier essai de commande en boucle fermée:

Les figures 2a, 2b et 2c montrent les réponses des trois articulations du robot commandé par les trois contrôleurs neuro-linguistiques. L'obtention de ces réponses passe par un réglage des trois coefficients G_e, $G_{\Delta e}$ et G_u et ceci pour chaque

correcteur. Les trois bras ont des temps de réponses différents. La rotation la plus lente répond en 0.1s. On a aucun dépassement et l'erreur en régime établie est nulle.

VII.5. Test de robustesse:

Dans cette essai on va communiquer aux trois correcteurs des informations entachées d'erreurs sur la position et la vitesse de chaque articulation. Dans la pratique ceci correspond au bruit que l'on retrouve dans les circuits de mesure. Dans ce qui suit un bruit blanc sera injecté dans la ligne de retour (voir figure 5.1).

VII.5.1. Bruitage de 2%:

Les figures 3a, 3b et 3c montrent les réponses des trois articulations correspondant à cet essai. On ne remarque aucun changement dans la dynamique des trois articulations. Néanmoins, l'effet du bruitage se traduit par l'apparition de légères erreurs statiques particulièrement pour la troisième rotation. Ces erreurs sont infèrieures à 0.5% de la valeur de la consigne.

VII.5.2. Bruitage de 5%:

Les figures 4a, 4b et 4c montrent les réponses des trois rotations pour ce second essai de robustesse. Le fait d'avoir augmenté le bruit n'a affecté ni la stabilité, ni l'allure des réponses. On note cependant une augmentation des écarts statiques

pour les trois rotations néanmoins inférieures à 1% de la valeur de la consigne.

VII.6. Commentaires:

Grâce aux essais de simulation précedants, on remarque que les controleurs neuro-linguistiques sont capables de commander un système multivariable fortement non-linéaire. Mais comme il s'agit de la commande d'un bras manipulateur un problème se pose. Les mouvements des trois articulations ne sont pas synchronisés.

VII.7. Synchronisation des trois articulations:

Lorsqu'on commande un bras manipulateur, on recherche souvent la synchronisation de toutes les articulations du robot. On entend par synchronisation des mouvements le fait d'imposer que toutes les articulation atteignent leur régime établie au même instant.
Un essai d'adaptation des paramètres des correcteurs s'est avéré incapable de réaliser la synchronisation.
L'utilisation d'une technique appropriée est rendue donc nécessaire.

VII.8. Commande du robot avec géneration de trajectoire:

Une technique de synchronisation des mouvements présentée dans [29] consiste en la géneration de la trajectoire que doit éffectuée le bras manipulateur et ceci non pas en lui donnant

directement la consigne finale à atteindre mais en lui donnant des consignes intermédiaires. Ces consignes sont différentes vu que les dynamiques des articulations sont différentes. Ces consignes sont déterminées comme suit [29]:

$$D_i(t) = Cf_i(t) - Y_i(t) \qquad (7.5)$$

$$\Delta D_i(t) = D_i(t) \frac{T_c}{Tr_{max}} \frac{Min(D_i(t))}{Max(D_i(t))} \qquad (7.6)$$

$$C_i(t) = C_i(t-1) + \Delta D_i(t). \qquad (7.7)$$

où: $Cf_i(t)$ est la consigne finale à atteindre.

$Y_i(t)$ est la position de l'articulation i.

$D_i(t)$ est l'ecart entre la position actuelle et la position finale à atteindre.

Tr_{max} est le temps de réponse correspondant à l'articulation la plus lente.

T_c est la periode de commande.

Cette technique permet de synchroniser le mouvement des articulations du bras manipulateur par rapport à l'articulation la plus lente.

VII.8.1. Réponses indicielles des articulations:

Les figures 5a, 5b et 5c montrent les reponses des trois

rotations avec synchronisation par géneration de trajectoires. Non seulement les trois articulations atteignent leur régime permanent au même instant mais leurs réponses sont pratiquement confondues pendant la phase de montée.

VII.8.2. Visualisation des signaux de commande:

Les figures 6a, 6b et 6c montrent les signaux de commandes génerés par les trois correcteurs pour les réponses des figures 5a, 5b et 5c et ce pendant le régime transitoire (phase de montée). Pendant le régime permanent les commandes oscillent alternativement entre U_{max} et U_{min} et entre deux valeurs autour de $0.5U_{min}$ et $0.5U_{max}$. Globalement les trois signaux sont différents pour les trois états.

VII.9. Test de robustesse sous géneration de trajectoires:

Cet essai a pour but (comme dans VII.5) de tester la robustesse des contrôleurs vis-à-vis du bruitage. Cette fois nous disposons d'un outil supplémentaire: la géneration de trajectoire. On va pouvoir à la suite comparer avec l'essai du VII.5.

VII.9.1. Bruitage de 2%:

Les figures 7a, 7b et 7c montrent les réponses des trois rotations en présence d'un bruit de 2% de la valeur de la consigne. On ne remarque aucun changement par rapport à la

dynamique sans bruitage. Dans ce cas l'erreur statique introduite par le bruitage est pratiquement nulle.

VII.9.2. Bruitage de 5%:

Les figures 8a, 8b et 8c montrent les résultats de cet essai de simulation. On remarque une légère déterioration dans la réponse de la troisième articulation. Par contre les erreurs statiques demeurent pratiquement nulles.

VII.9.3. Commentaire:

Ces deux essais nous permettent de conclure que l'introduction de la géneration de trajectoires a permis non seulement la synchronisation des mouvements des trois articulations mais d'améliorer la robustesse au bruitage.

VII.10. Test de répétabilité:

Ce test consiste à donner aux trois articulations un signal de consigne carré periodique variant entre deux valeurs différentes (On a choisi 0 et 1). Le but de cet essai est de quantifier l'accumulation des erreurs statiques de chaque articulation lorsqu'elle effectue un mouvement de va et vient entre deux positions différentes.

Les figures 9a, 9b et 9c montrent le comportement des trois articulations. On constate des oscillations au voisinage de zéro particulièrement pour les articulations 2 et 3. Par contre on ne

note aucune accumulation d'erreurs statiques et ceci pour les trois articulations.

VII.11. Conclusion:

Ces essais ont montré que le controleur neuro-linguistique est capable de commander un système MIMO non linéaire avec un bon rejet du bruitage. La technique de génération de trajectoires nous a permis non seulement de réaliser la synchronisation des trois articulations mais nous avons aussi remarqué qu'elle permet une élimination totale de l'erreur statique en présence en d'un bruit. Cette technique permet également l'obtention de trés bon résultats pour les trois articulations en test de répétabilité.

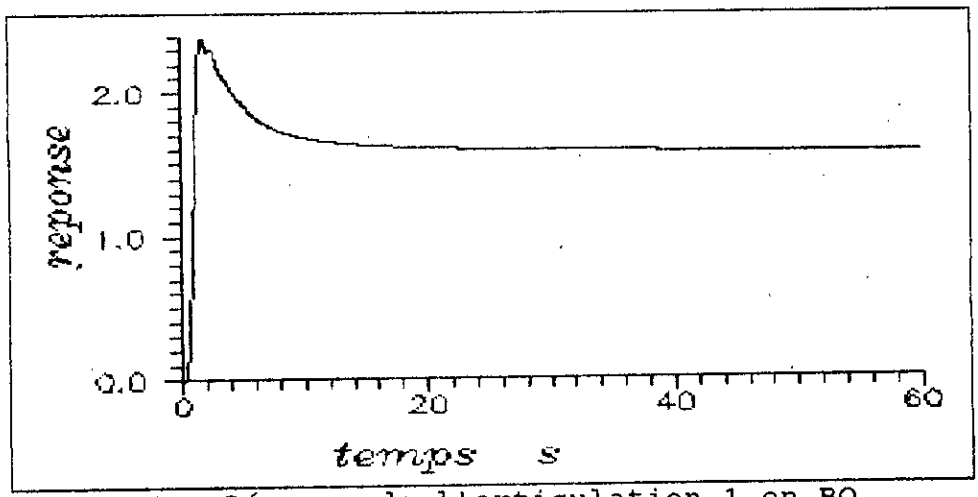

Figure 1a. Réponse de l'articulation 1 en BO.

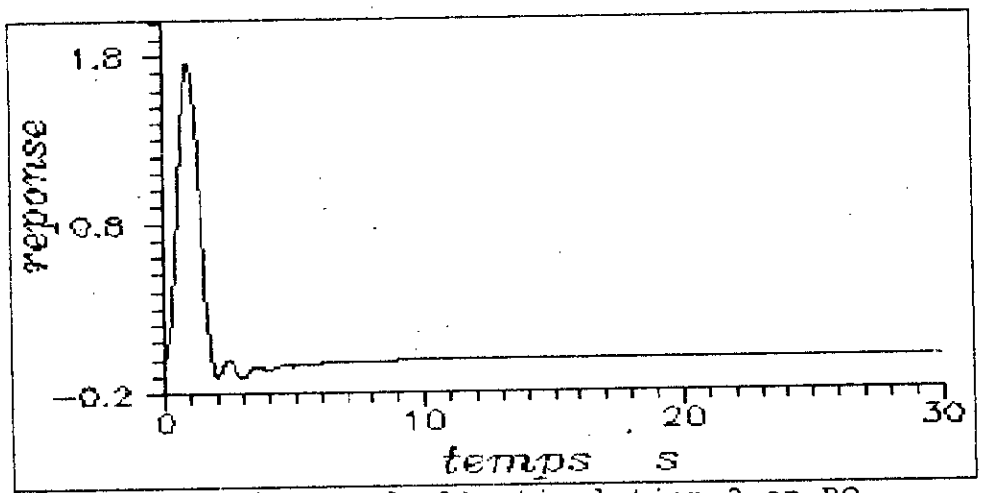

Figure 1b. Réponse de l'articulation 2 en BO.

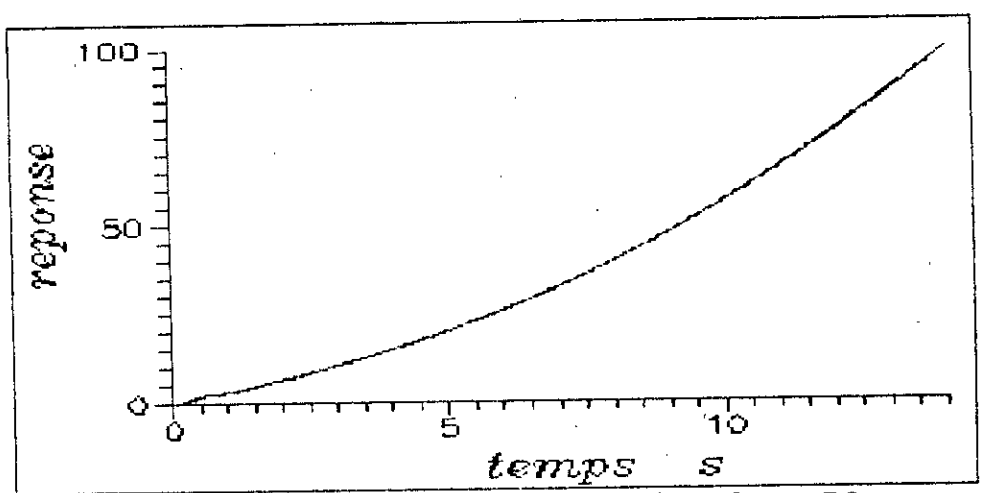

Figure 1c. Réponse de l'articulation 3 en BO.

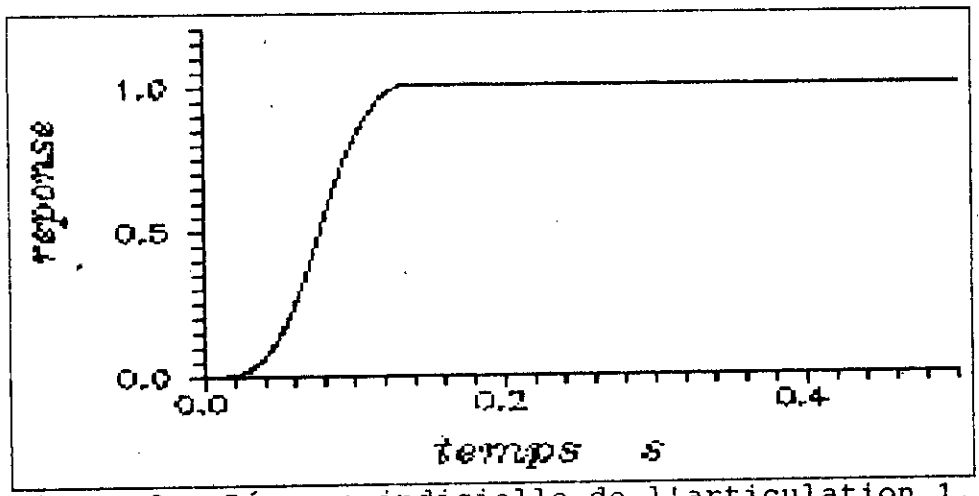

Figure 2a. Réponse indicielle de l'articulation 1.

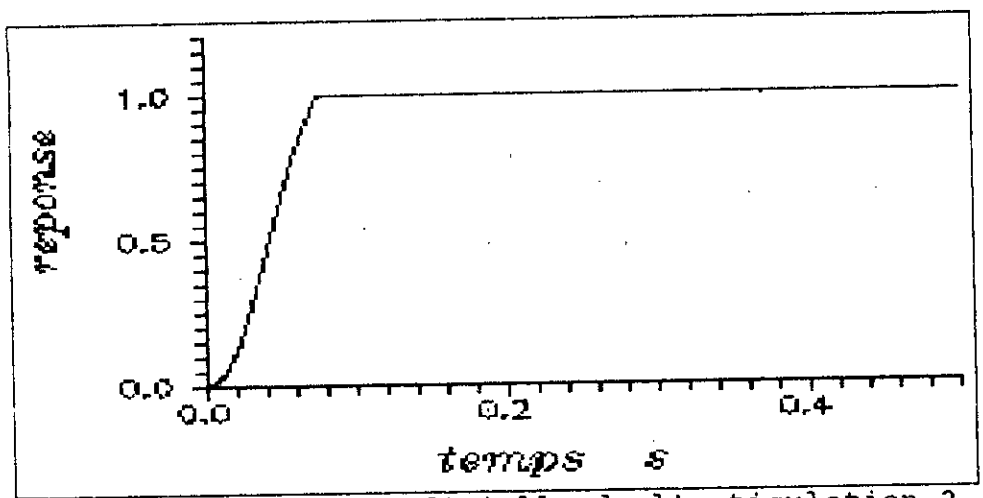

Figure 2b. Réponse indicielle de l'articulation 2.

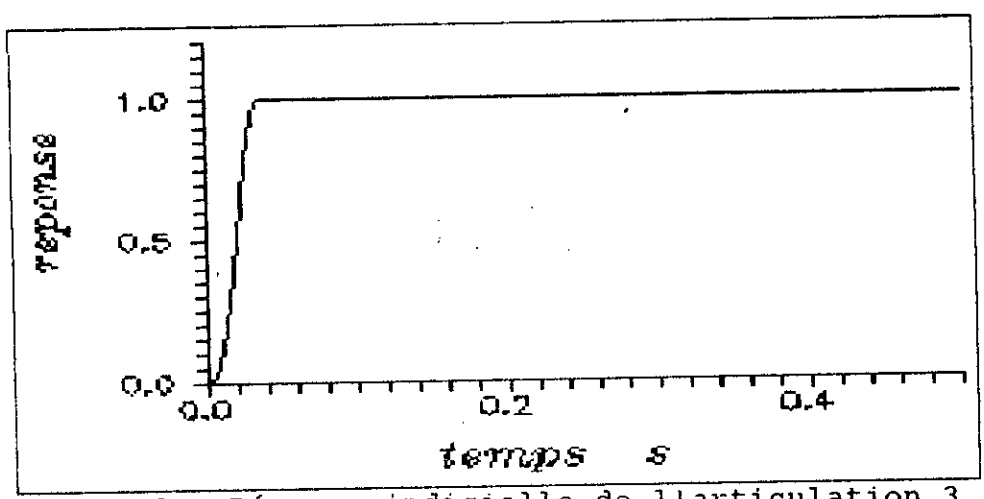

Figure 2c. Réponse indicielle de l'articulation 3.

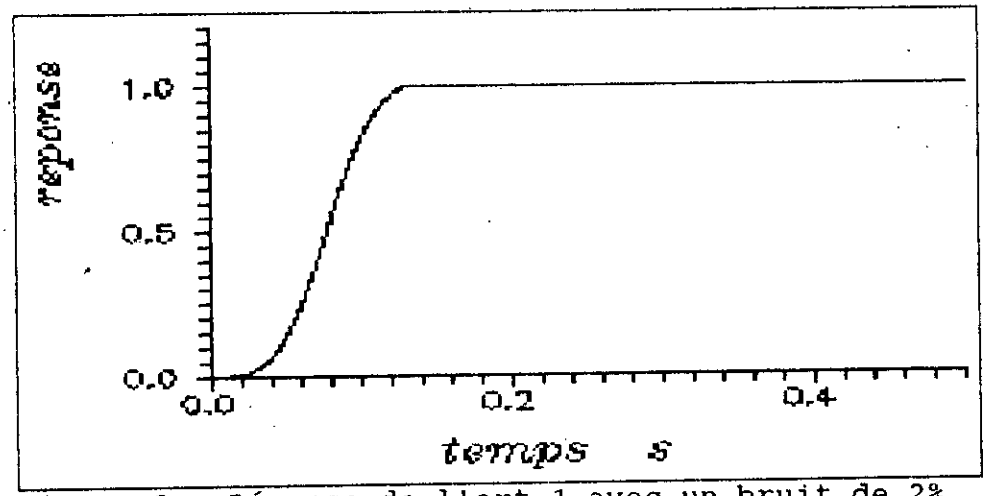

Figure 3a. Réponse de l'art.1 avec un bruit de 2% .

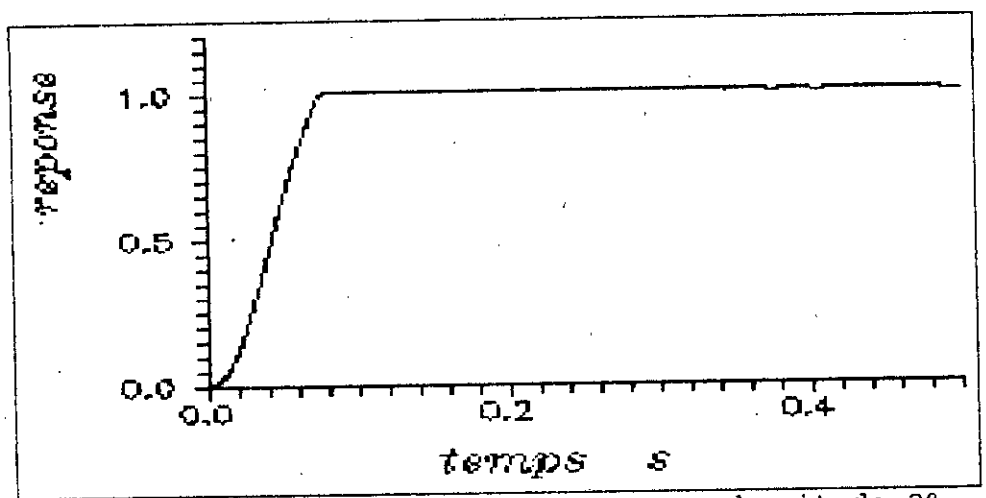

Figure 3b. Réponse de l'art.2 avec un bruit de 2% .

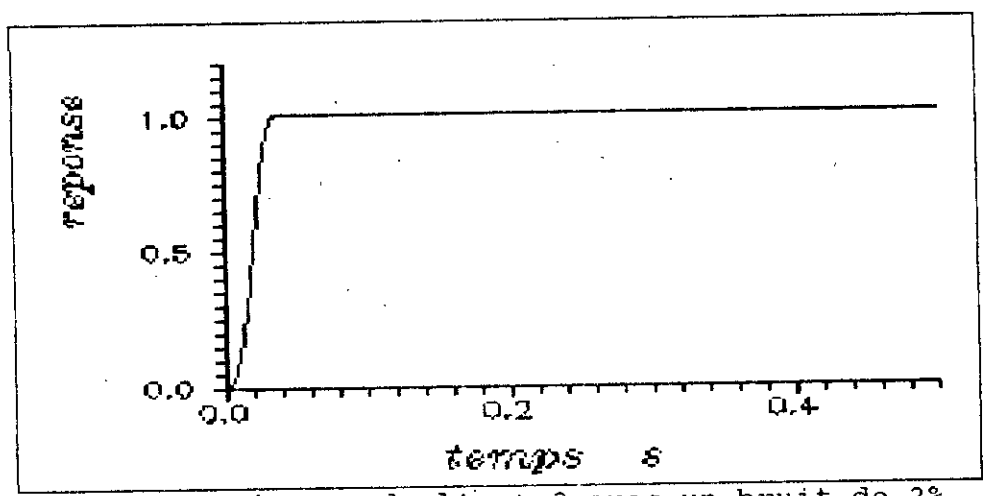

Figure 3c. Réponse de l'art.3 avec un bruit de 2% .

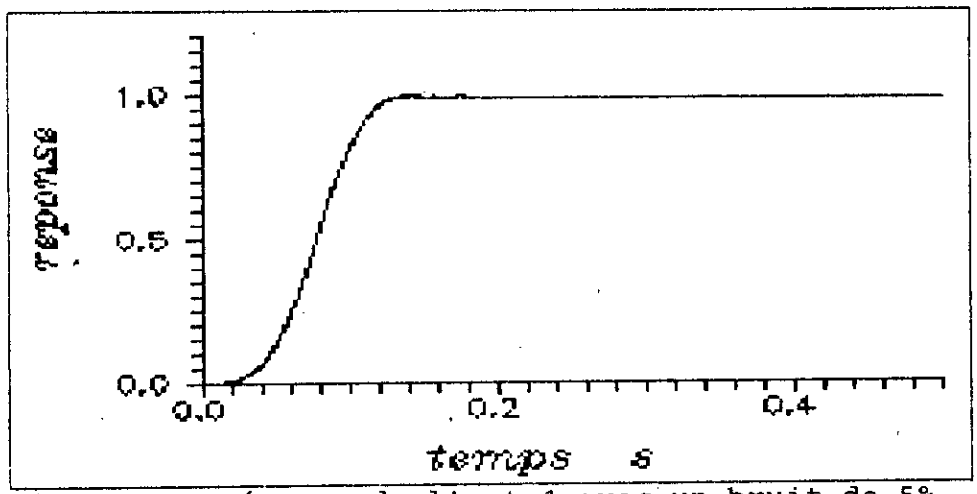

Figure 4a. Réponse de l'art.1 avec un bruit de 5% .

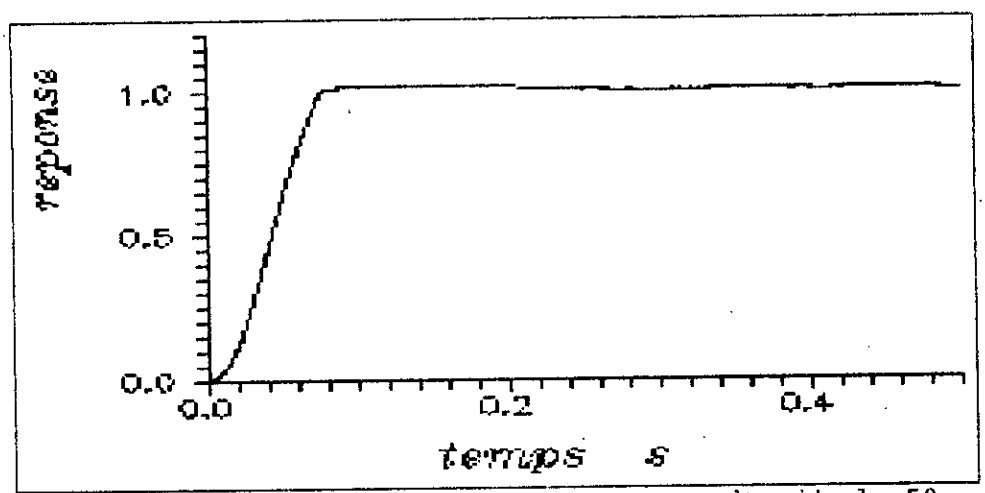

Figure 4b. Réponse de l'art.2 avec un bruit de 5% .

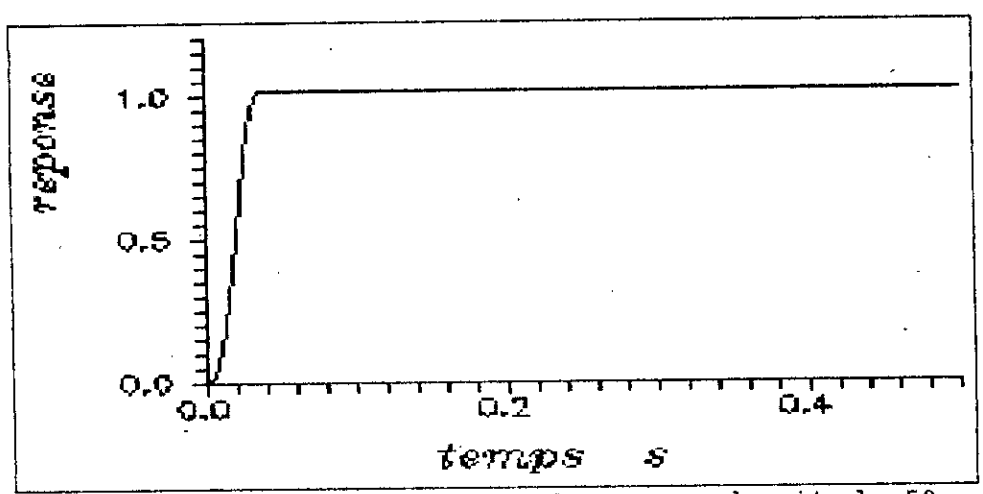

Figure 4c. Réponse de l'art.3 avec un bruit de 5% .

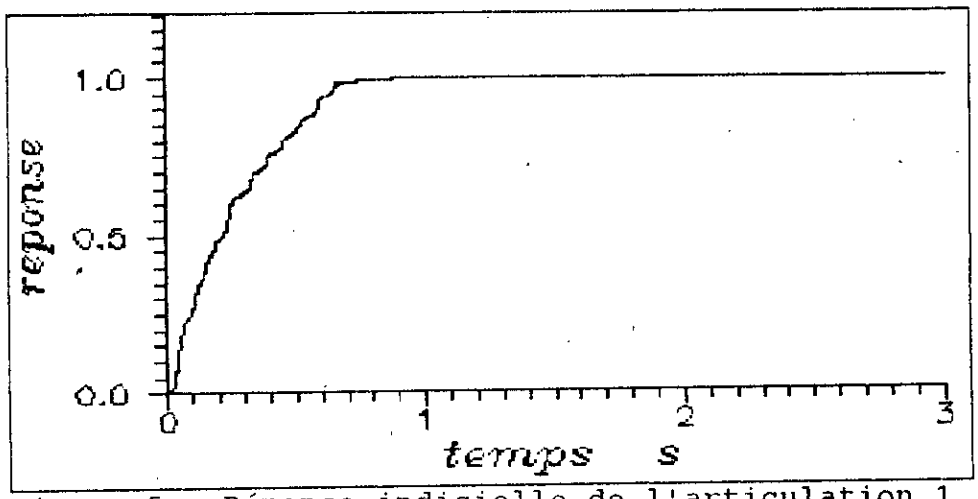

Figure 5a. Réponse indicielle de l'articulation 1.

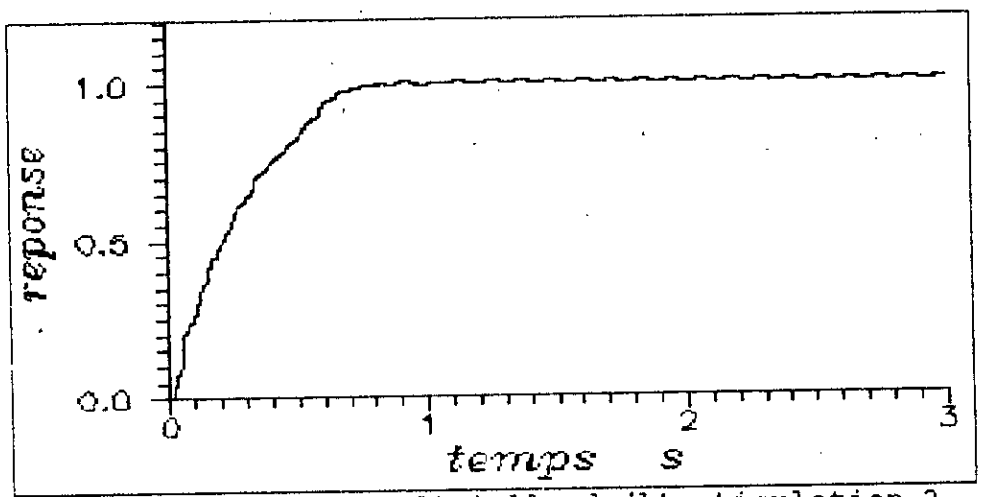

Figure 5b. Réponse indicielle de l'articulation 2.

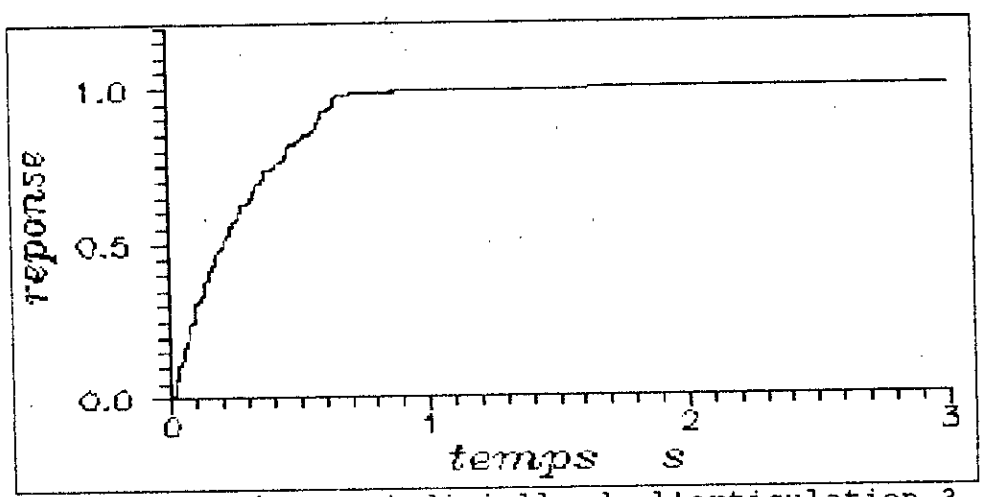

Figure 5c. Réponse indicielle de l'articulation 3.

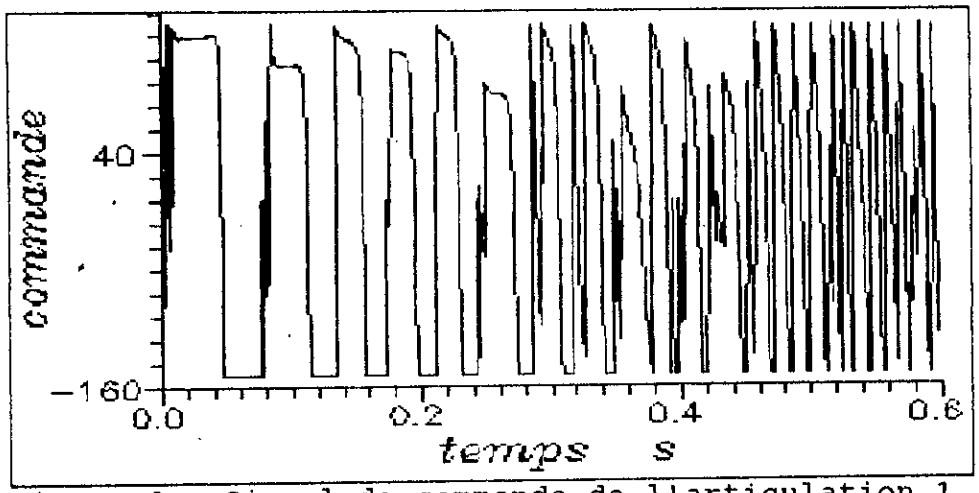

Figure 6a. Signal de commande de l'articulation 1.

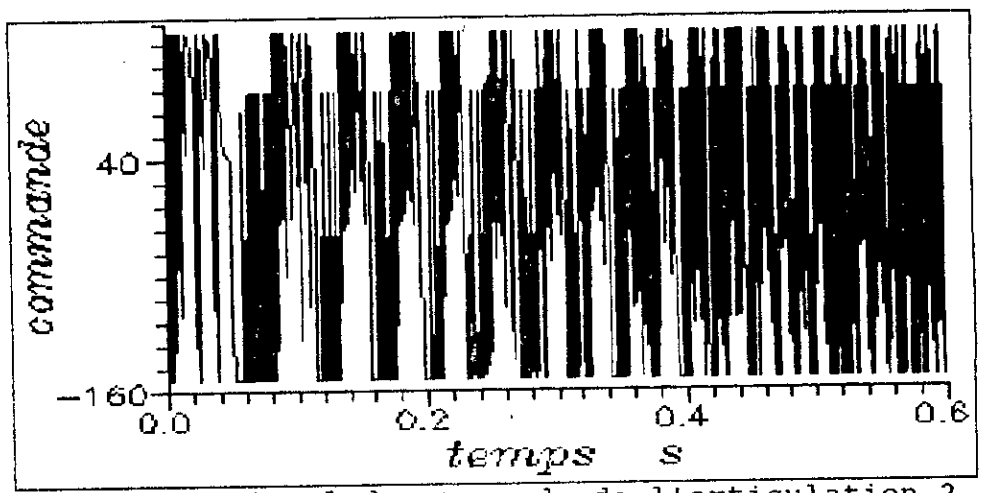

Figure 6b. Signal de commande de l'articulation 2.

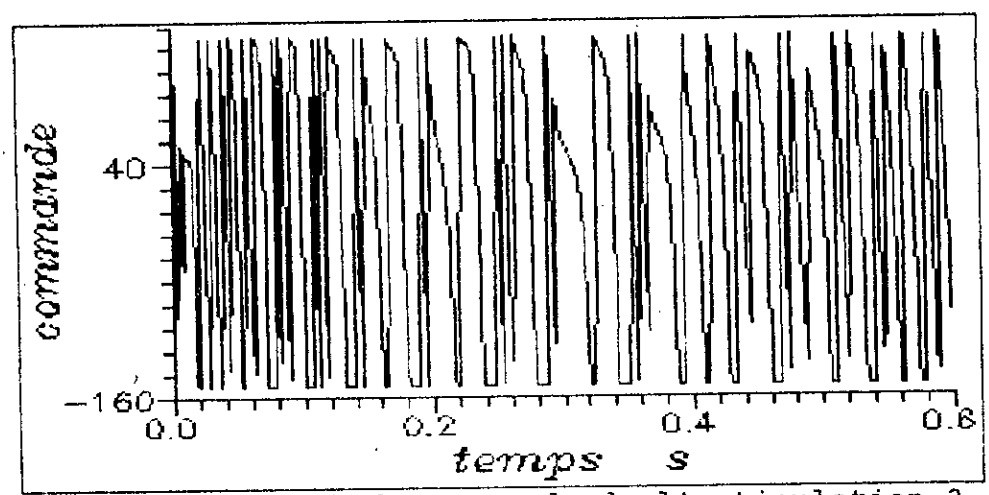

Figure 6c. Signal de commande de l'articulation 3.

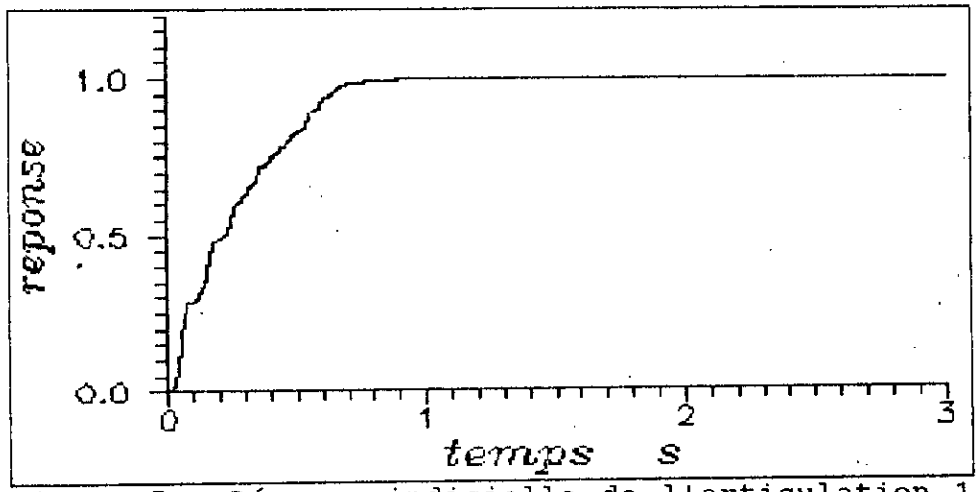

Figure 7a. Réponse indicielle de l'articulation 1 avec bruit de 2%.

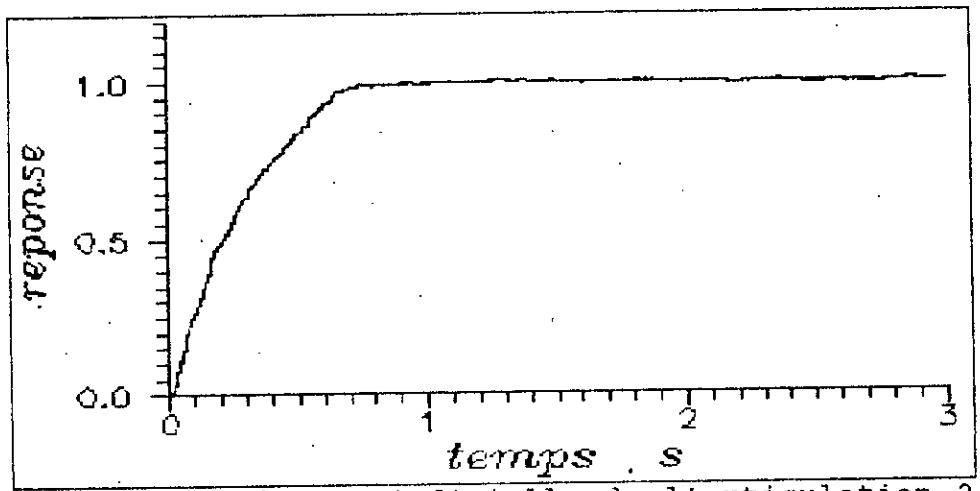

Figure 7b. Réponse indicielle de l'articulation 2 avec bruit de 2%.

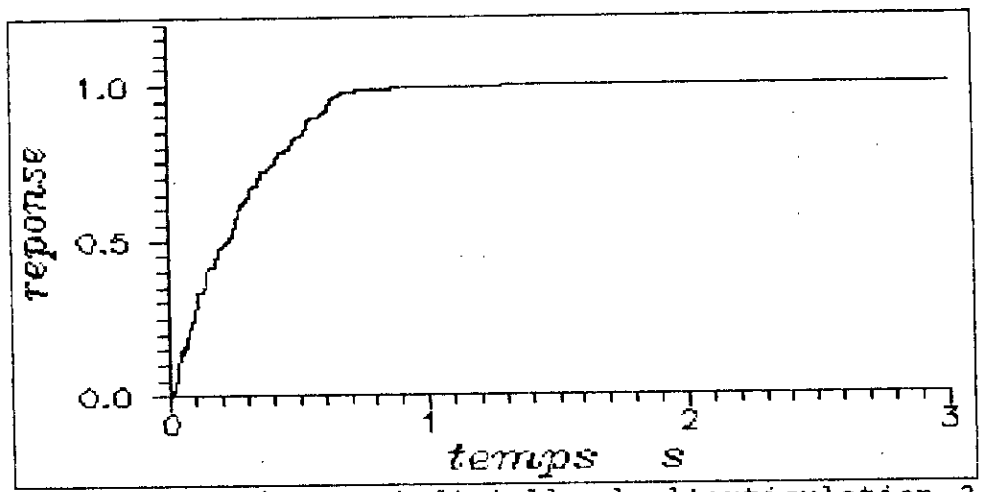

Figure 7c. Réponse indicielle de l'articulation 3 avec bruit de 2%.

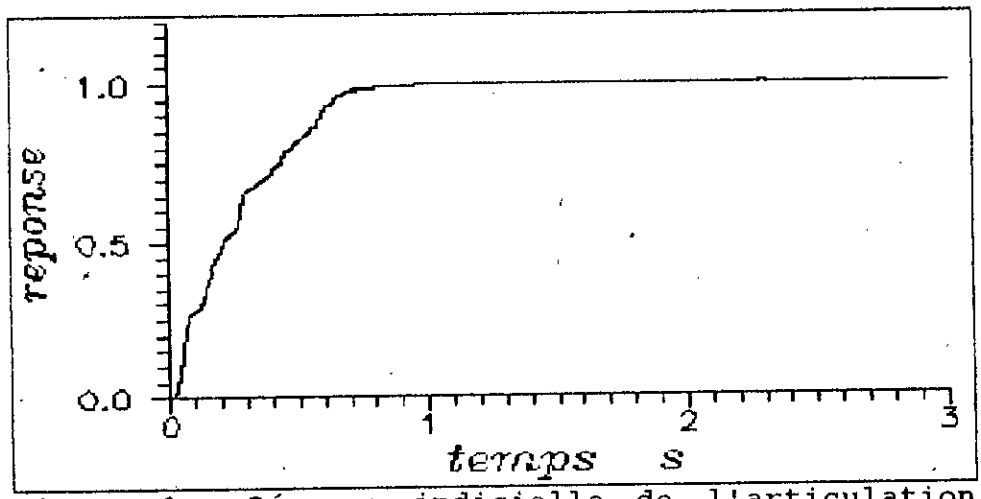

Figure 8a. Réponse indicielle de l'articulation avec bruit de 5%.

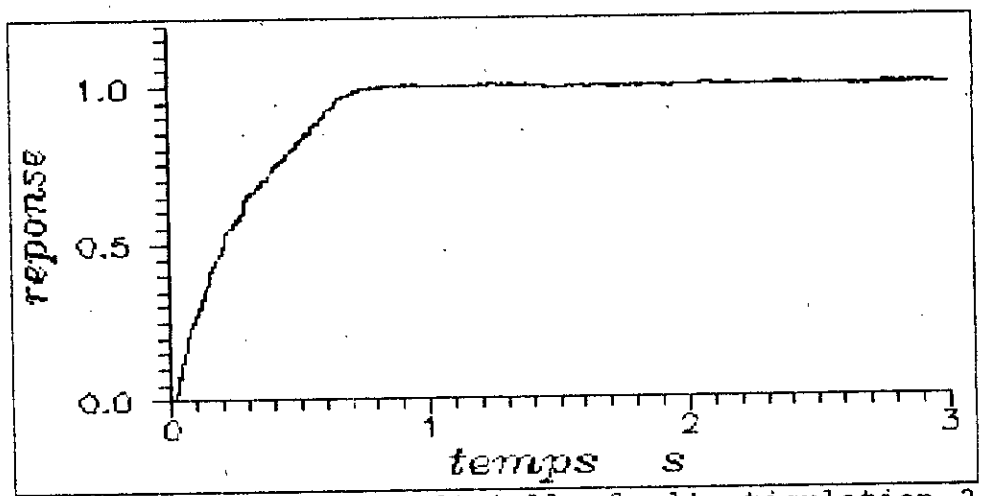

Figure 8b. Réponse indicielle de l'articulation 2 avec bruit de 5%.

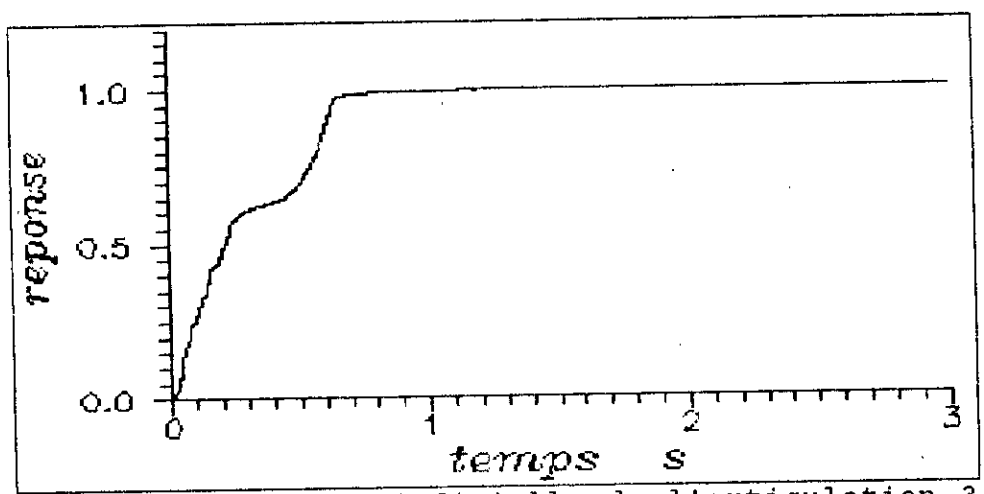

Figure 8c. Réponse indicielle de l'articulation 3 avec bruit de 5%.

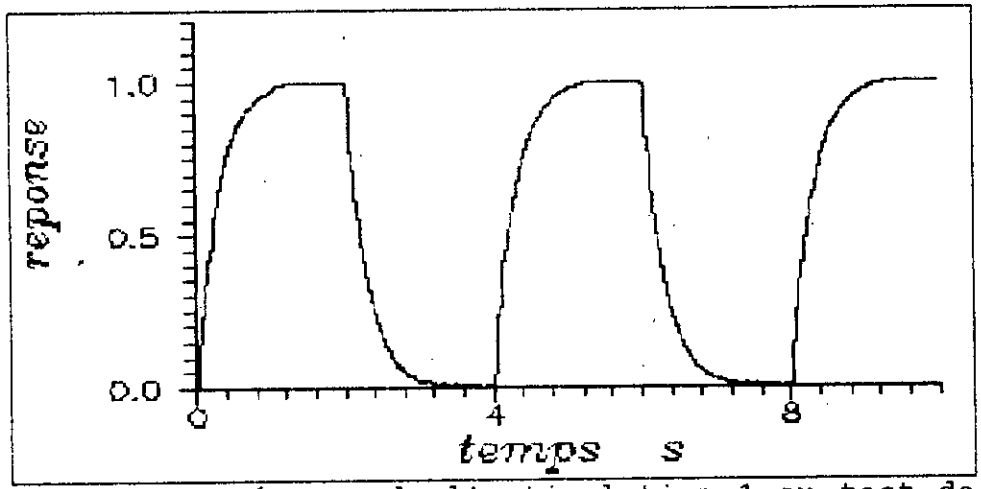

Figure 9a. Réponse de l'articulation 1 au test de répétabilité.

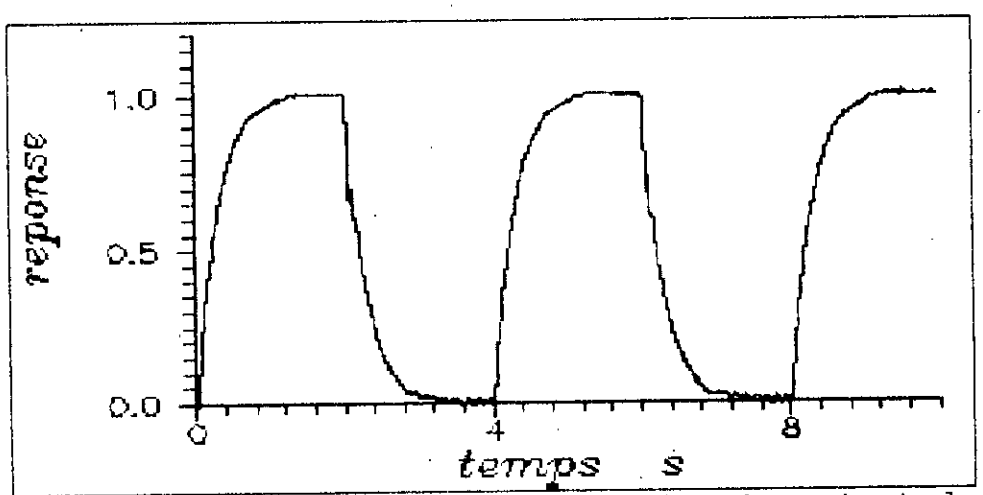

Figure 9b. Réponse de l'articulation 2 au test de répétabilité.

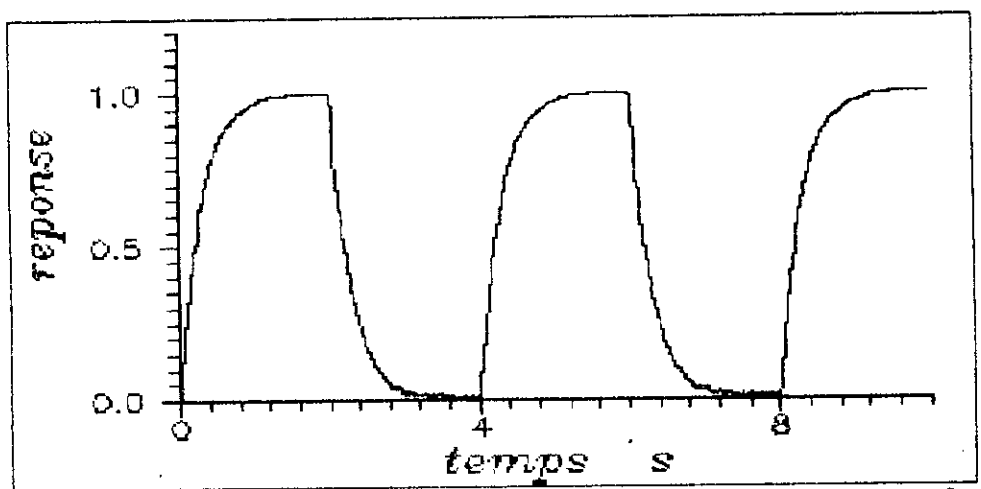

Figure 9c. Réponse de l'articulation 3 au test de répétabilité.

CONCLUSION GENERALE

Notre travail a consisté à étudier les réseaux neuronaux artificiels et la commande linguistique floue. Leur association nous a permis la conception d'un contrôleur possédant les avantages de ces deux techniques.

L'apprentissage des réseaux neuronaux multicouches a nécessité l'utilisation d'un puissant moyen de calcul, et la recherche de l'architecture possédant la meilleure capacité d'apprentissage a demandé de nombreux essais.

Le contrôleur neuro-linguistique ainsi obtenu a montré une bonne capacité à commander des systèmes SISO linéaire et non linéaire et MIMO non linéaire.

Les essais de simulation de la commande de systèmes SISO ont montré que:

-Le contrôleur neuro-linguistique peut commander des systèmes SISO instables aussi bien linéaires que non linéaires avec un bon rejet des perturbations et génère des signaux de commande capables de supprimer dans une certaine mesure un bruitage provenant des circuits de mesure.

-Le contrôleur présente également une bonne robustesse à la variation d'un paramètre du système.

D'autre part, l'application à la commande d'un bras manipulateur a montré que:

-Le contrôleur neuro-linguistique est capable de commander un système multivariable non linéaire dont les états sont fortement couplés.

-L'introduction de bruit de mesure dans une certaine limite

n'affecte pas les performances du contrôleur.

Le contrôleur neuro-liguistique ne permettant pas de réaliser la synchronisation des mouvements des trois articulations du robot, l'utilisation d'une technique de synchronisation procédant par génération de trajectoires a permis:

-La synchronisation des mouvements des trois articulations.

-L'amélioration des performances du contrôleur en présence d'un bruitage.

-L'obtention de bons résultats en test de répétabilité pour les trois articulations.

Le contrôleur neuro-linguistique de part cette puissance de commande se caractérise par une grande facilité de mise en oeuvre.

Son application à d'autres systèmes réels tels que: suspension active, commande de processus complexes et larges tels que raffinerie et réseau électrique est à envisager.

B I B L I O G R A P H I E

[1] "Les réseaux neuronaux Artificiels". Article Paru dans la revue Langages & Systèmes N° 71. Par H. Lemberg.

[2] Philip: D. Wasserman. "Neural Computing: Theory an Practice". Ed. Van Nostrand Reinhold, New York, 1989.

[3] "Réseaux de Neurones: 30 ans aprés, les applications". Extrait de la revue Micro-Systèmes. Par Claire Rémy. Juin 91.

[4] R. P. Lippmann. "An Introduction to Computing with Neural Nets". IEEE ASSP Magazine. April 1987.

[5] H. Mori et al. "An Artificial Neural-Net Based Technique for Power System Dynamic Stability with the Kohonen Model". IEEE Transactions On Power Systems, Vol. 7, N° 2, May 1992.

[6] D. E. Rumelhart, G. E. Hinton, R. J. Williams. "Learning Internal Representations by Error Propagation". In D. E. Rumelhart & J. L. McClelland (Eds), "Parallel Distributed Processing ". MIT Press 1986.

[7] Teuvo Kohonen. "The Self Organizing Map". Proceedings of the IEEE, Vol. 78, N°9, September 1990.

[8] H.Ritter & K.Shulten. "Convergence Proprieties of Kohonen's Topology Conserving Maps: Fluctuations, Stability and Dimension Selection". Biological Cybernetics, N°60, 1988.

[9] B. Windrow, M. A. Lehr. "30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation". Proccedings Of The IEEE, Vol. 78, N°9, September 1990.

[10] H. Buhler. "Conception des systèmes automatiques". Presses Polythechniques Romandes, 1988.

[11] S. Thiria, F. Badran, C. Mejia, M. Crepon. "Fonction de Transfert et Classifieur Neuronal". Rapport de Recherche N°658 du Laboratoire de Recherche En Informatique (L.R.I). Université de Paris-Sud (Orsay). Avril 1991.

[12] M. Aggoune et al. "Preliminary results Of Using Artificial Neural Networks for Security Assessment". IEEE Proceedings of 1989 PICA. Seattle, WA. May 1989.

[13] Psaltis et al. "A Multilayered Neural Network Controller". IEEE Control Systems Magazine, April 1988.

[14] Fukuda et al. "Theory and Applications Of neural Networks For Industrial Electronic". IEEE Transactions On Industrial Electronics, Vol. 39, N°6, December 1992.

[15] J. M. Martinez et al. "La retro-Propagation sous l'angle de la théorie du contrôle". Proceedings de neuro-Nimes 1991.

[16] P. Y. Glorennec. "Association d'un Réseau Neuronal et de Règles Floues pour le Contrôle d'un Système Dynamique". Proceedings de neuro-Nimes 1990.

[17] Fukuda et al. "Neuromorphic Control: Adaptation and Learning". IEEE Transactions On Industrial Electronics, Vol. 39, N°6, December 1992.

[18] C. C. Lee. "Fuzzy Logic In control systems: Fuzzy Logic Controller-Part 1". IEEE Transactions On Systems, Man and Cybernetics, Vol. 20, N°2, 1990.

[19] J. J. Procyk and E. H. Mamdani. "A Linguistic Self-Organizing Process Controller". Automatica, Vol. 15, N°1, 1979.

[20] H. Maaref, M. Brunet et W. Nouibat. "Commande de Processus Linéaires Par Un Controleur Flou". Proceedings Of the ICEA, Tizi-Ouzou, Algeria, May 1992.

[21] M. C..Souami. "Real Time Transputer Based Intelligent Robot Control System Design". PHD Theses, Leeds University 1991.

[22] M. C. Souami & K. F. Gill. "Multilevel Self-Organizing Controller". To Be Published.

[23] E. Davalo & P. Naim. "Des Réseaux de neurones". Editions EYROLLES, 1990.

[24] T.Fukuda et al. "Neural Network Application for Robotic Motion control". Proceedings of Int. Joint Conf. Neural Networks, IJCNN'90, San Diego, CA, 1990.

[25] D. Trabzi. "Commande par retour d'etat avec observateur-Etude comparative". Projet de fin d'etude, E.N.P. Département Genie-Electrique, Juin 1992.

[26] J.C. Gilles, P. Decaulne, M. Pélegrin. " Systèmes asservis non linéaires". Ed. DUNOD, 1988.

[27] D. P. Stoten. "Generalized Manipulator Dynamics, With Regard To Model Reference Adaptive Control". International Journal Of Control, Vol. 50, N°6, 1989.

[28] D. P. Stoten & Benchoubène. "Empirical Studies Of An MRAC Algorithm With Minimal Controller Syntheses". Int. Journal of Control, Vol. 51, N°4, 1990.

[29] M. C. Souami. "A Simple Trajectory Generation For Real Time On-Line Control Of A Robot". Internal Report, Leeds University, 1988.