REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

D.E.R de Génie Electrique et Informatique
Spécialité : **AUTOMATIQUE**

Projet de Fin d'Etudes

en vue de l'obtention du diplôme d'Ingénieur d'Etat
en Automatique

Sujet

COMMANDE DECENTRALISEE SUPERVISEE PAR RESEAUX DE NEURONES ARTIFICIELS D'UN BRAS DE ROBOT MANIPULATEUR

Proposé et dirigé par :
D.BOUKHETALA
F.BOUDJEMA

Etudié par :
DJEBIRI Mustapha

Juillet 1997

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

D.E.R de Génie Electrique et Informatique
Spécialité : **AUTOMATIQUE**

Projet de Fin d'Etudes

en vue de l'obtention du diplôme d'Ingénieur d'Etat
en Automatique

Sujet

COMMANDE DECENTRALISEE SUPERVISEE PAR RESEAUX DE NEURONES ARTIFICIELS D'UN BRAS DE ROBOT MANIPULATEUR

Proposé et dirigé par :
D.BOUKHETALA
F.BOUDJEMA

Etudié par :
DJEBIRI Mustapha

Juillet 1997

Dédicace

Je dédie ce travail

à mes parents
à mes frères
à mes soeurs
à toute ma famille
à tous mes amis

AVANT-PROPOS

Ce travail a été effectué au laboratoire de commande des processus de l'Ecole Nationale Polytechnique

Je tiens à exprimer ma profonde reconnaissance à mes promoteurs Mr. D. Boukhetala et Mr. F. Boudjema qui ont dirigé ce travail et qui m'ont aidé à surmonter les difficultés rencontrées.

Je remercie aussi Salim et Omar pour les précieux conseils qu'ils m'ont prodigué.

Enfin, à tous ceux qui ont contribué de prés ou de loin à l'aboutissement de ce travail, j'exprime tous mes remerciements.

ملخص:

هذا العمل يستعرض مبدأ التحكم اللامركزي المراقب بواسطة الشبكات العصبية الإصطناعية. في هذا الإطار، قمنا بتعليم قانوني تحكم للشبكات العصبية الإصطناعية (تحكم لامركزي تلاؤمي، تحكم العزم المحسوب). نتائج تطبيق هذه التقنية في التحكم على ذراع آلي قد عرضت و كذا اختبارات الضلاعة على المنظم العصبي.

كلمات مفتاحية : تحكم لامركزي، شبكات عصبية إصطناعية، تحكم مراقب، تحكم العزم المحسوب.

Abstract :

This work exposes the principle of the decentralized supervised control using neural networks. In this context, two control laws are supervised (decentralized adaptive control, computed torque control). Results of application of this control technique to a robot manipulator are presented as well as robustness tests of the neural controller.

Key words : Decentralized Control, Artificial Neural Networks, Supervised Control, Computed Torque Control.

Résumé :

Ce travail expose le principe de la commande décentralisée supervisée par réseaux de neurones artificiels. Dans ce cadre, deux lois de commande sont supervisées (commande décentralisée adaptative, commande par couple calculé). Les résultats de l'application de cette technique de commande à un bras de robot manipulateur sont présentés ainsi que des tests de robustesse du contrôleur neuronal.

Mots clés : Commande Décentralisée, Réseaux de Neurones Artificiels, Commande Supervisée, Commande par Couple Calculé.

SOMMAIRE

Chapitre III

Commande décentralisée supervisée par Réseaux de Neurones Artificiels

Chapitre IV

Application de la commande décentralisée supervisée par RNA à un bras de robot manipulateur

Conclusion générale

Références bibliographiques

Introduction Générale

Introduction générale

La grande diversité des systèmes non linéaires peut être considérée comme la première raison qui a empêché le développement d'une théorie systématique et générale dans le domaine de l'Automatique. Bien qu'il existe des techniques traditionnelles pour l'analyse et la synthèse des systèmes non linéaires (méthode du plan de phase, techniques de linéarisation,...) ces techniques restent toutefois limitées [10].

L'apparition des réseaux de neurones artificiels, avec leur capacité d'approximer n'importe quelle fonction non linéaire, c'est avérée un outil très puissant pour la commande des systèmes non linéaires, remédiant ainsi aux lacunes des techniques classiques.

Cependant, la non linéarité n'est pas le seul problème qu'on rencontre dans les systèmes physiques. En effet, il y a beaucoup de systèmes qui présentent une structure complexe; systèmes de grande dimension, composés de plusieurs sous-systèmes interconnectés pouvant avoir des objectifs en conflit [20]. Ces systèmes peuvent poser de sérieux problèmes lors de l'analyse et la synthèse de la commande par des approches centralisées classiques. Pour surmonter ces problèmes, l'intérêt est actuellement porté sur les structures de commande décentralisées qui considèrent un système complexe comme un ensemble de sous-systèmes interconnectés. Chaque sous-système sera commandé par une station de commande n'ayant accès qu'aux informations locales [21].

Dans ce travail, on s'intéresse à l'étude de la commande décentralisée supervisée par réseaux de neurones artificiels.

Dans le premier chapitre, vu qu'on va appliquer cette commande à un bras de robot manipulateur, des notions en robotique sont présentées et les différents modèles du robot sont élaborés. Une étude théorique sur les réseaux de neurones artificiels, incluant les définitions, la classification et les algorithmes d'apprentissage, est exposée dans le second chapitre. On consacre le troisième chapitre à la présentation de la commande décentralisée supervisée. Dans ce cadre, deux approches ont été considérées dans le sens d'une supervision de deux lois de commande (commande décentralisée adaptative, commande par couple calculé).

L'application de cette technique à un bras de robot manipulateur se trouve dans le quatrième chapitre, où des résultats de simulation, incluant des tests de robustesse, sont présentés, ainsi que des commentaires et conclusions.

Enfin, on termine ce travail par une conclusion générale.

Chapitre I

Modélisation du robot Manipulateur

I-1 Introduction :

Il a été estimé, il y a quelques décennies, que le secteur de la robotique va très vite devenir une nouvelle industrie de dimension aussi grande que celle de l'industrie automobile actuelle[1]. En effet, l'invention des robots, qui tiennent leur origine des machines à commande numérique et des télémanipulateurs, et leur incorporation dans l'industrie a fait diminuer considérablement l'intervention directe de l'homme dans la réalisation de tâches matérielles tout en assurant une importante augmentation de la productivité. Ceci nous donne une idée sur les bénéfices qu'ont apporté les robots industriels.

Dans le cadre de notre travail, ce chapitre va présenter des notions de base relatives aux robots, ainsi que les étapes essentielles pour élaborer leurs modèles (géométrique, cinématique et dynamique). Des méthodes systématiques d'obtention de ces modèles ont été développées pour une large classe de robots industriels. Ces modèles sont nécessaires, à la fois, pour la conception et pour la commande de tels mécanismes.

I-2 Notions et définitions :

I-2-1 Définition d'un robot :

Pour le sens commun, un robot est un dispositif mécanique articulé capable d'imiter certaines fonctions humaines telles que la manipulation d'objets ou la locomotion, dans le but de se substituer à l'homme pour la réalisation de certaines tâches matérielles [4].

D'autres définitions ont été proposées pour qualifier un robot, parmi lesquelles on cite :

- Un robot est un système versatile doté d'une mémoire et pouvant effectuer des mouvements comme ceux d'un opérateur humain (définition attribuée par la J.I.R.A : Japan Industrial Robot industry Association) [1].

- Un robot est un manipulateur à fonctions multiples pouvant être programmé pour réaliser automatiquement des tâches variées éventuellement répétitives (définition attribuée par la R.I.A : Robot Institute of America) [1].

- Un robot est un manipulateur multifonctions reprogrammable, capable de déplacer des matériaux, des pièces, des outils ou des appareils spéciaux suivant des chemins programmés en vue d'effectuer des opérations de fabrication (définition attribuée par la Division Internationale de Robotique de la Société des Ingénieurs de Production) [6].

I-2-2 Structure générale d'un robot manipulateur :

Un robot manipulateur est généralement formé des parties suivantes [2]:

- Une structure mécanique qui supporte l'organe terminal à situer.
- Des actionneurs qui agissent sur la structure mécanique pour en modifier la configuration (changer la position de l'organe terminal).
- Des capteurs qui fournissent des informations sur l'état mécanique du robot (capteurs proprioceptifs) ou des informations sur l'environnement du robot (capteurs extéroceptifs). Ces informations sont nécessaires pour la commande du robot.
- Un système de commande qui pilote les actionneurs du robot manipulateur.
- Un système de décision qui assure la fonction de raisonnement et élabore le mouvement du robot manipulateur à partir de la définition de la tâche à exécuter par l'opérateur à l'aide du système de communication.
- Un système de communication qui gère les messages transmis entre le système de décision et l'opérateur.

I-3 Présentation du robot utilisé :

Le robot auquel on s'intéresse est de classe quatre [5]. Il possède trois degrés de liberté : Une translation verticale d_1, une rotation θ_2 et enfin une translation horizontale d_3 (Fig. I-1).

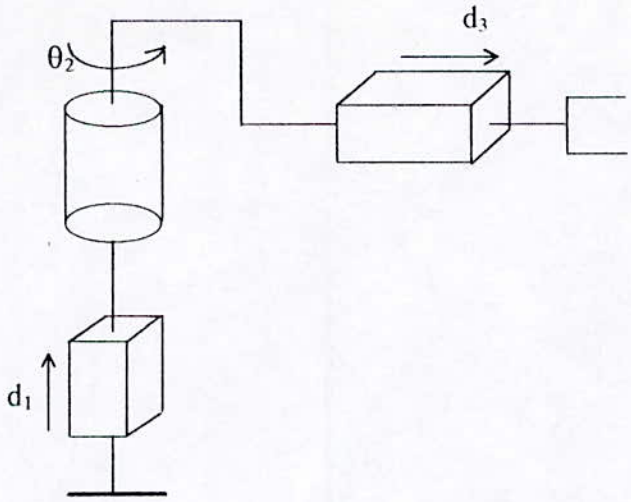


Figure I-1 *Schéma du robot utilisé*

Cette classe de robots est fréquemment utilisée dans l'industrie. On les appelle 'Pick-and-place manipulators' ou robots porteurs [1][5].

Les hypothèses suivantes sont à prendre en considération, pour permettre l'élaboration des différents modèles [5]:

- Les frottements sont de nature linéaire et visqueuse.
- Les liaisons sont supposées rigides.
- Les actionneurs sont idéaux, i.e., leur sortie est directement proportionnelle au signal de commande.
- Les capteurs sont à gain unité, et n'ont pas de composantes dynamiques.

I-4 Modélisation géométrique :

Il est plus naturel, pour un robot, de connaître la situation de son organe terminal plutôt que de considérer la variation de sa configuration. Mais étant donné que les commandes des asservissements des robots agissent sur les grandeurs articulaires, qui définissent la configuration, il est donc nécessaire d'établir une transformation géométrique reliant les variables articulaires q_i et les coordonnées absolues de l'organe terminal x_i (coordonnées opérationnelles).

La transformation de Denavit-Hartenberg permet de réaliser cette correspondance et de donner lieu, ainsi, à ce qu'on appelle modélisation géométrique du robot.

I-4-1 modélisation géométrique directe (MGD) :

Le modèle géométrique direct est la fonction F qui permet d'exprimer la situation de l'organe terminal du robot en fonction de sa configuration :

$$X = F(q) \qquad (I\text{-}1)$$

où X: Vecteur des coordonnées opérationnelles.

q: Vecteur des coordonnées généralisées.

Notations de Denavit-Hartenberg.

Pour implanter la transformation de Denavit-Hartenberg, on attache à chaque liaison S_i un repère $\Re_i=(O_i, x_i, y_i, z_i)$ défini ainsi :

O_i : point d'intersection de la normale commune, des axes des articulations n et n+1, avec l'axe de l'articulation n+1.

z_i : axe de l'articulation entre S_i et S_{i+1}.

x_i : perpendiculaire commune entre z_i et z_{i-1}.

y_i : défini de sorte que (x_i, y_i, z_i) soit un trièdre orthonormé.

On définit quatre paramètres (Fig. I-2) :

α_i : angle (z_{i-1}, z_i) mesuré autour de x_i.

a_i : distance (z_{i-1}, z_i) mesurée suivant x_i.

θ_i : angle (x_{i-1}, x_i) mesuré autour de z_{i-1}.

d_i : distance (x_{i-1}, x_i) mesurée suivant z_{i-1}.

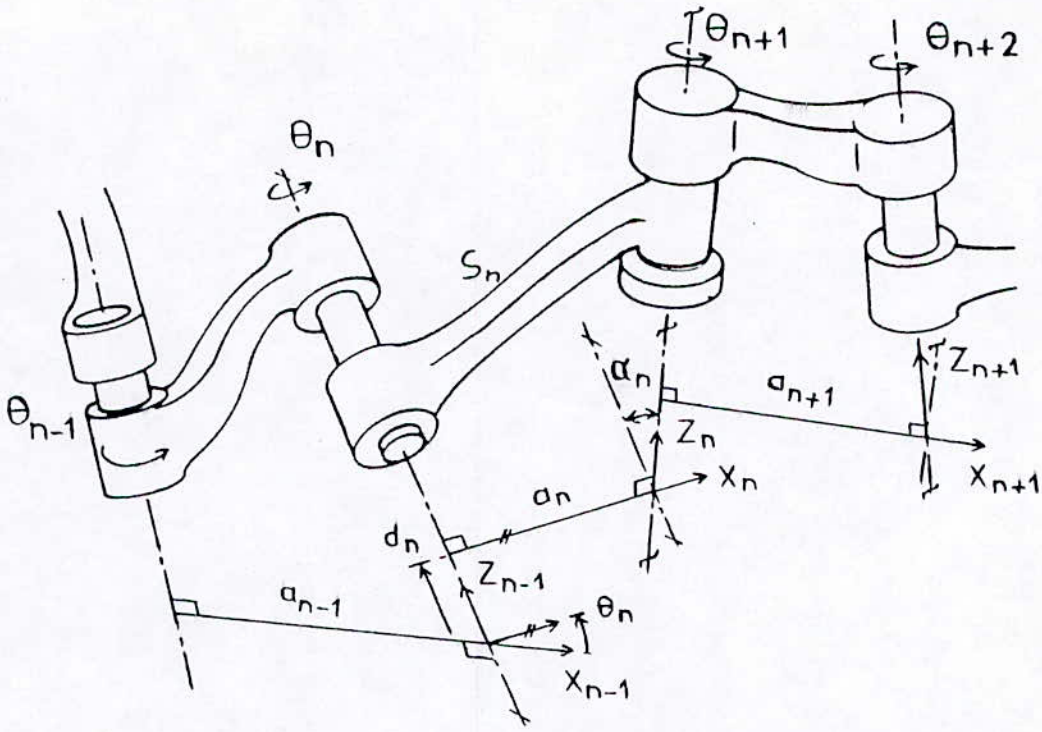


Figure I-2 *Notations de Denavit-Hartenberg*

Cela dit, pour faire coïncider le référentiel $\Re_{i-1}$ au référentiel $\Re_i$, on doit effectuer une succession de 4 transformations élémentaires : translation le long de l'axe z_{i-1} d'une longueur d_i, rotation autour du même axe d'une quantité θ_i, autre translation selon l'axe x_{i-1} de valeur a_i et enfin une rotation d'amplitude α_i autour de x_{i-1} (qui coïncide maintenant avec x_i).

D'où on peut écrire :

$$\Re_{i-1} \xrightarrow{A_i = T(z,d_i) \times R(z,\theta_i) \times T(x,a_i) \times R(x,\alpha_i)} \Re_i \qquad \text{(I-2)}$$

où T : indice de translation.

R : indice de rotation.

A_i: matrice de passage du référentiel $\Re_{i-1}$ au référentiel $\Re_i$.

En utilisant la représentation des coordonnées homogènes [3], on obtient :

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(I-3)}$$

Pour une liaison prismatique, a_i est nul [3].

Pour le robot de classe 4 (Fig. I-3), le tableau ci-dessous contient les valeurs des paramètres de DH.

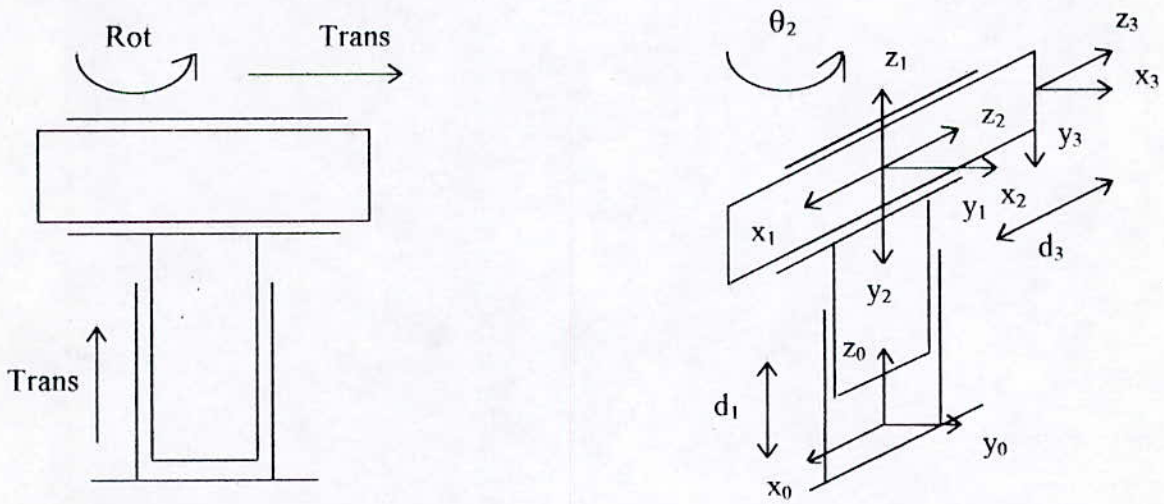


Figure I-3 *Implantation de la transformation de DH*

Liaison	Variables	d_i	θ_i	a_i	α_i
1	d_1	d_1	0°	0	0°
2	θ_2	0	θ_2	0	-90°
3	d_3	d_3	0°	0	0°

Et les matrices de passage d'un repère à un autre sont :

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(I-4)}$$

Le passage du référentiel de base au référentiel lié à l'organe terminal est assuré par la matrice :

$$T_3 = A_1 A_2 A_3 = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 & -d_3\sin\theta_2 \\ \sin\theta_2 & 0 & \cos\theta_2 & d_3\cos\theta_2 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(I-5)}$$

Les coordonnées de l'organe terminal dans son repère (référentiel $\Re_3$) sont données par le vecteur $r_3 = [\,0 \;\; 0 \;\; 0 \;\; 1\,]^T$. Donc ses coordonnées dans le repère $\Re_0$ sont définies par le vecteur :

$$X = r_0 = T_3 r_3 = F(q) \quad \text{(I-6)}$$

où : $X = [x \;\; y \;\; z \;\; 1]^T$

$q = [d_1 \;\; \theta_2 \;\; d_3]^T$

$F(q) = [f_1(q) \;\; f_2(q) \;\; f_3(q) \;\; f_4(q)]^T$

d'où :

$$\begin{cases} x = f_1(q) = -d_3\sin\theta_2 \\ y = f_2(q) = d_3\cos\theta_2 \\ z = f_3(q) = d_1 \\ 1 = f_4(q) = 1 \end{cases} \quad \text{(I-7)}$$

Ainsi, nous avons établi le modèle géométrique direct du robot.

I-4-2 modélisation géométrique inverse (MGI) :

Ayant décrit le modèle géométrique direct d'un manipulateur donnant X = F(q), il s'agit maintenant de résoudre ce dernier système par rapport aux q_i.

En général, si on suppose que $X=[x_1 \; x_2 \ldots x_m]^T$ et $q=[q_1 \; q_2 \ldots q_n]^T$ où m est le nombre de coordonnées opérationnelles et n est le nombre de coordonnées généralisées, la modélisation

géométrique inverse consiste à résoudre un système non linéaire de m équations à n inconnues. Trois cas sont à considérer :

- Si n>m, le manipulateur est dit redondant par rapport à la tâche à effectuer, il y a plus d'inconnues que d'équations et le problème est indéterminé.
- Si n<m, le manipulateur possède moins de degrés de liberté que n'en nécessite l'exécution normale de la tâche caractérisée par les m valeurs x_i. Il existe alors m-n contraintes sur les x_i.
- Si n=m, le système possède une solution unique et peut être résolu pour des configurations simples. Cependant, il est difficile, dans le cas général, de résoudre ce système à cause de sa non linéarité.

Pour le robot de classe 4, et en utilisant (I-7), on tire facilement :

$$\begin{cases} d_1 = z \\ \theta_2 = \arctan\left(-\dfrac{x}{y}\right) \\ d_3 = \sqrt{x^2 + y^2} \end{cases} \quad , \text{ pour } y \neq 0 \text{ et } \theta_2 \neq \frac{\pi}{2} + k\pi, k \in Z \qquad \text{(I-8)}$$

I-5 Modélisation cinématique :

Elle établit le lien entre les dérivées des coordonnées généralisées (vitesses généralisées) et les dérivées des coordonnées opérationnelles.

I-5-1 Modélisation cinématique directe (MCD) :

Le modèle cinématique direct d'un robot manipulateur permet de calculer la différentielle dX, des coordonnées opérationnelles, en fonction de la différentielle dq, des coordonnées généralisées. A partir de (I-1), et pour de petites variations de X, correspondant à de petites variations de q, on a :

$$dX = \left[\frac{\partial F}{\partial q}\right] dq \qquad \text{(I-9)}$$

où les dimensions des matrices sont respectivement : m×1, m×n, n×1

avec n : nombre de degrés de liberté du robot.

m : nombre de degrés de liberté considérées dans l'espace des tâches.

La matrice [∂F/∂q] est dite matrice Jacobienne, notée J(q) et définie par :

$$J(q) = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & & & \\ \frac{\partial f_m}{\partial q_1} & \frac{\partial f_m}{\partial q_2} & \cdots & \frac{\partial f_m}{\partial q_n} \end{bmatrix} \tag{I-10}$$

Pour notre robot et en utilisant (I-7), il est aisément obtenu :

$$J = \begin{bmatrix} 0 & -d_3\cos\theta_2 & -\sin\theta_2 \\ 0 & -d_3\sin\theta_2 & \cos\theta_2 \\ 1 & 0 & 0 \end{bmatrix} \tag{I-11}$$

1-5-2 Modélisation cinématique inverse :

Le modèle cinématique inverse est obtenu en inversant la matrice Jacobienne (si cette dernière est inversible) :

$$dq = \left[J(q)\right]^{-1} dX \tag{I-12}$$

Cette relation exprime la variation des coordonnées généralisées en fonction de la variation des coordonnées opérationnelles.

A partir de (I-11) nous avons :

$$J^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ -\frac{\cos\theta_2}{d_3} & -\frac{\sin\theta_2}{d_3} & 0 \\ -\sin\theta_2 & \cos\theta_2 & 0 \end{bmatrix} \quad \text{avec } d_3 \neq 0 \tag{I-13}$$

Donc pour de petites variations $\Delta q=q(t_i)-q(t_{i-1})$, $\Delta X=X(t_i)-X(t_{i-1})$ nous avons :

$$\begin{bmatrix} d_1(t_i)-d_1(t_{i-1}) \\ \theta_2(t_i)-\theta_2(t_{i-1}) \\ d_3(t_i)-d_3(t_{i-1}) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ \dfrac{-\cos(\theta_2(t_{i-1}))}{d_3(t_{i-1})} & \dfrac{-\sin(\theta_2(t_{i-1}))}{d_3(t_{i-1})} & 0 \\ -\sin(\theta_2(t_{i-1})) & \cos(\theta_2(t_{i-1})) & 0 \end{bmatrix} \begin{bmatrix} x(t_i)-x(t_{i-1}) \\ y(t_i)-y(t_{i-1}) \\ z(t_i)-z(t_{i-1}) \end{bmatrix} \qquad \text{(I-14)}$$

A l'aide de cette relation, on peut éviter le calcul répétitif de l'inversion de la matrice Jacobienne. En effet, connaissant $X(t_i)$, $X(t_{i-1})$ et $q(t_{i-1})$, on peut calculer de façon itérative les différentes variables généralisées $q(t_i)$. Ainsi, on peut dire que la modélisation cinématique complète la modélisation géométrique.

I-6 Génération des trajectoires :

Une trajectoire est l'ensemble des points dans l'espace parcourus par le robot lors de l'exécution d'une tâche. Si une trajectoire est donnée dans l'espace opérationnel (espace des tâches), on peut générer les trajectoires correspondantes dans l'espace des variables articulaires en utilisant le modèle cinématique inverse (équation (I-14)). Plusieurs critères peuvent être considérés pour le choix d'une trajectoire : l'énergie consommée, le temps d'exécution de la tâche, l'évolution continue et lisse de certaines grandeurs physiques...etc.

Dans notre travail, nous avons choisi une famille de trajectoires assurant une continuité en position, vitesse et accélération. Ce sont les trajectoires cycloïdales. Une trajectoire cycloïdale évolue selon la fonction suivante :

$$q(t) = \begin{cases} q_i + (q_f - q_i)\left(\dfrac{t}{t_f} - \dfrac{\sin(2\pi\, t/t_f)}{2\pi}\right) & \text{si } t \le t_f \\ q_f & \text{si } t > t_f \end{cases} \qquad \text{(I-15)}$$

Par dérivation nous obtenons :

$$\dot{q}(t) = \begin{cases} \dfrac{(q_f - q_i)}{t_f}\left(1 - \cos\left(2\pi\, t/t_f\right)\right) & \text{si } t \le t_f \\ 0 & \text{si } t > t_f \end{cases} \qquad \text{(I-16)}$$

$$\ddot{q}(t) = \begin{cases} \dfrac{2\pi\,(q_f - q_i)}{t_f^2}\sin\left(2\pi\, t/t_f\right) & \text{si } t \le t_f \\ 0 & \text{si } t > t_f \end{cases} \qquad \text{(I-17)}$$

où : $q_i = q(0)$

$q_f = q(t_f)$

t_f : temps final.

I-7 Modélisation dynamique :

Dans cette partie, nous allons établir les équations différentielles non linéaires qui relient les efforts actionneurs τ_i (forces généralisées), aux variables articulaires q_i, aux vitesses articulaires $\dot{q}_i$ et aux accélérations articulaires $\ddot{q}_i$. L'ensemble de ces équations constitue le modèle dynamique du manipulateur.

On peut résoudre les équations différentielles du robot pour obtenir les mouvements q_i pour des τ_i donnés dans le but d'une simulation du comportement dynamique (modèle dynamique direct) comme on peut calculer les efforts τ_i pour des mouvements q_i connus (on obtient le modèle dynamique inverse).

Pour obtenir le modèle dynamique, plusieurs approches ont été établies. On peut en citer : le formalisme d'Euler-Lagrange, méthode de Newton-Euler.

I-7-1 Formalisme d'Euler-Lagrange :

Le formalisme des équations de Lagrange nous permet de décrire les équations du mouvement de plusieurs corps articulés les uns par rapport aux autres. Son usage est plus simple, pour le non spécialiste, que d'autres formalismes, même s'il exige souvent des calculs plus longs.

Pour un système mécanique articulé défini par n coordonnées généralisées indépendantes q_i, i=1 à n, l'équation d'Euler-Lagrange s'écrit dans le cas général :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \frac{\partial E_D}{\partial \dot{q}_i} = \tau_i \qquad ; i = \overline{1,n} \tag{I-18}$$

avec L : Lagrangien du système, défini par $L=E_c-E_p$ où E_c : Energie cinétique, E_p : énergie potentielle.

E_D : Energie de dissipation.

τ_i : Forces généralisées.

Ces équations dynamiques vont être dérivées en passant par 6 étapes. D'abord on calcul la vitesse de chaque point de chaque liaison, et on déduit ainsi l'énergie cinétique. Puis l'énergie potentielle sera développée ainsi que l'énergie de dissipation. Enfin, on forme le Lagrangien et on fait les différentiations.

- Calcul de la vitesse d'un point du manipulateur :

Soit un point de la chaîne articulée appartenant à la liaison i. Ses coordonnées r_0 par rapport à un référentiel de base sont données par :

$$r_0 = T_i r_i \tag{I-19}$$

où : T_i est la matrice de passage du référentiel de base au référentiel i et r_i est le vecteur des coordonnées du point considéré dans le référentiel i.

L'intégrale triple dans (I-24) est dite matrice des pseudo-inerties du segment i, et est définie par :

$$J_i = \int_i r_i r_i^T dm = \begin{bmatrix} \int_i x_i^2 dm & \int_i x_i y_i dm & \int_i x_i z_i dm & \int_i x_i dm \\ \int_i x_i y_i dm & \int_i y_i^2 dm & \int_i y_i z_i dm & \int_i y_i dm \\ \int_i x_i z_i dm & \int_i y_i z_i dm & \int_i z_i^2 dm & \int_i z_i dm \\ \int_i x_i dm & \int_i y_i dm & \int_i z_i dm & \int_i dm \end{bmatrix} \quad \text{(I-25)}$$

L'énergie cinétique totale est :

$$k = \sum_{i=1}^{n} \left(k_i + E_{aci} \right) \quad \text{(I-26)}$$

avec $E_{aci} = \frac{1}{2} I_i \dot{q}_i^2$ énergie cinétique introduite par l'actionneur i et I_i moment d'inertie de l'actionneur i qui devient équivalent à la masse dans le cas d'une liaison prismatique [3].

- Calcul de l'énergie potentielle :

L'énergie potentielle d'une chaîne articulée est donnée par :

$$E_p = -\sum_{i=1}^{n} m_i G^T T_i r_i \quad \text{(I-27)}$$

avec $G = \begin{bmatrix} g_x & g_y & g_z & 1 \end{bmatrix}^T$ vecteur de gravitation dans un repère absolu. Dans notre cas $G = \begin{bmatrix} 0 & 0 & -g & 1 \end{bmatrix}^T$.

$T_i r_i$ représente les coordonnées du centre de masse de la liaison i par rapport au repère absolu.

- Calcul de l'énergie de dissipation :

Elle est donnée par :

$$E_D = \frac{1}{2}\sum_{i=1}^{n} f_i \dot{q}_i^2 \tag{I-28}$$

f_i : coefficient de frottement visqueux.

- Le Lagrangien :

$$L = E_c - E_p = \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{i}\sum_{j=1}^{i} \text{trace}\left[\frac{\partial T_i}{\partial q_k} J_i \left(\frac{\partial T_i}{\partial q_j}\right)^T\right] \dot{q}_k \dot{q}_j + \frac{1}{2}\sum_{i=1}^{n} I_i \dot{q}_i^2 + \sum_{i=1}^{n} m_i G^T T_i r_i \tag{I-29}$$

- Calcul des forces généralisées (dérivation des équations dynamiques) :

$$\begin{aligned}\tau_i = &\sum_{k=i}^{n}\sum_{j=1}^{k} \text{trace}\left[\frac{\partial T_k}{\partial q_j} J_k \left(\frac{\partial T_k}{\partial q_i}\right)^T\right] \ddot{q}_j + \sum_{k=i}^{n}\sum_{j=1}^{k}\sum_{m=1}^{k} \text{trace}\left[\frac{\partial^2 T_k}{\partial q_j \partial q_m} J_k \left(\frac{\partial T_k}{\partial q_i}\right)^T\right] \dot{q}_j \dot{q}_m \\ &- \sum_{k=i}^{n} m_k G^T \frac{\partial T_k}{\partial q_i} r_k + I_i \ddot{q}_i + f_i \dot{q}_i\end{aligned} \tag{I-30}$$

En appliquant cette procédure au robot de classe 4, nous obtenons :

$$\begin{cases} E_c = \frac{1}{2}\left((m_1 + m_3)\dot{q}_1^2 + m_3 \dot{q}_3^2 + \left[m_3 (q_3 - l_2)^2 + J_2 + J_3\right]\dot{q}_2^2\right) \\ E_p = (m_1 + m_3) g\, q_1 \\ E_D = \frac{1}{2}\left(f_1 \dot{q}_1^2 + f_2 \dot{q}_2^2 + f_3 \dot{q}_3^2\right) \end{cases} \tag{I-31}$$

où : $[q_1\ q_2\ q_3]=[d_1\ \theta_2\ d_3]$

m_1 : masse de la première liaison.

m_3 : masse de la troisième liaison.

l_2 : distance entre le centre de masse de la liaison 3 et son extrémité.

J_3 : moment d'inertie de la liaison 3 par rapport à un axe vertical passant par son centre de masse.

$J_2=J_1+J_{a1}$ moment d'inertie de la liaison 1 et de son actionneur.

Nous obtenons les équations suivantes :

$$\begin{cases} \tau_1 = (m_1 + m_3)\ddot{q}_1 + (m_1 + m_3)g + f_1\dot{q}_1 \\ \tau_2 = \left(J_2 + J_3 + m_3(q_3 - l_2)^2\right)\ddot{q}_2 + 2m_3(q_2 - l_2)\dot{q}_2\dot{q}_3 + f_2\dot{q}_2 \\ \tau_3 = m_3\ddot{q}_3 - m_3(q_3 - l_2)\dot{q}_2^2 + f_3\dot{q}_3 \end{cases} \quad \text{(I-32)}$$

I-7-2 Modèle dynamique avec charge :

Dans la pratique, lors de l'exécution d'une tâche donnée, le robot porte des charges avec son élément terminal qui peuvent varier d'un moment à un autre. Donc, pour compléter la modélisation dynamique du robot, on doit prendre en considération ce point de vue.

Supposons que le robot de classe 4 porte une charge de masse m_0 avec son élément terminal. Les énergies cinétique et potentielle du robot deviennent :

$$\begin{cases} E_c = \frac{1}{2}\left((m_1 + m_3 + m_0)\dot{q}_1^2 + (m_3 + m_0)\dot{q}_3^2 + \left[m_3(q_3 - l_2)^2 + m_0q_3^2 + J_2 + J_3\right]\dot{q}_2^2\right) \\ E_p = (m_1 + m_3 + m_0)gq_1 \end{cases} \quad \text{(I-33)}$$

D'où le modèle dynamique avec charge (en prenant $\tau_1=k_1u_1$, $\tau_2=k_2u_2$, $\tau_3=k_3u_3$) :

$$\begin{cases} k_1u_1 = (m_1 + m_3 + m_0)\ddot{q}_1 + (m_1 + m_3 + m_0)g + f_1\dot{q}_1 \\ k_2u_2 = \left(J_2 + J_3 + m_3(q_3 - l_2)^2 + m_0q_3^2\right)\ddot{q}_2 + 2\left(m_3(q_3 - l_2) + m_0q_3\right)\dot{q}_2\dot{q}_3 + f_2\dot{q}_2 \\ k_3u_3 = (m_3 + m_0)\ddot{q}_3 - \left(m_3(q_3 - l_2) + m_0q_3\right)\dot{q}_2^2 + f_3\dot{q}_3 \end{cases} \quad \text{(I-34)}$$

I-8 Résultats de simulation :

Les équations (I-34) doivent être mises sous forme d'état dans le but de la simulation du comportement dynamique du robot.

La forme d'état s'écrit :

$$\begin{cases} \dot{x} = Ax + Bu + D \\ y = Cx \end{cases} \qquad \text{(I-35)}$$

où : $x = \left[x_1\ x_2\ x_3\ x_4\ x_5\ x_6\right]^T = \left[q_1\ \dot{q}_1\ q_2\ \dot{q}_2\ q_3\ \dot{q}_3\right]^T$ vecteur d'état.

$y = \left[y_1\ y_2\ y_3\right]^T = \left[q_1\ q_2\ q_3\right]^T$ vecteur des sorties.

$u = \left[u_1\ u_2\ u_3\right]^T$ vecteur des commandes.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\dfrac{f_1}{m_1 + m_3 + m_0} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\dfrac{f_2}{J^*} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -\dfrac{f_3}{m_3 + m_0} \end{bmatrix}; \quad B = \begin{bmatrix} 0 & 0 & 0 \\ \dfrac{k_1}{m_1 + m_3 + m_0} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \dfrac{k_2}{J^*} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \dfrac{k_3}{m_3 + m_0} \end{bmatrix};$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$$D = \left[0 \quad -g \quad 0 \quad -\frac{2\left(m_3\left(x_5 - l_2\right) + m_0 x_5\right)x_4 x_6}{J^*} \quad 0 \quad \frac{\left(m_3\left(x_5 - l_2\right) + m_0 x_5\right)x_4^2}{m_3 + m_0}\right]^T;$$

$$J^* = J_2 + J_3 + m_3\left(x_5 - l_2\right)^2 + m_0 x_5^2.$$

Les simulations ont été faites par la méthode de Runge-Kutta d'ordre 4 avec un pas de simulation $h=10^{-3}$s, et des conditions initiales nulles $x(t=0)=[0\ 0\ 0\ 0\ 0\ 0]^T$.

Les paramètres du robot sont donnés par :

$l_2=0.75$ m; $m_1=20$ Kg; $m_3=10$ Kg; $m_0=0$ Kg; $k1=100$ N.V^{-1}; $k_2=10$ N.V^{-1}; $k_3=10$ N.V^{-1}; $f_1=30$ N.s.m^{-1}; $f_2=7.825$ N.s.m^{-1}; $f_3=20$ N.s.m^{-1}; $J_2=2$ Kg.m^2; $J_3=0.2$ Kg.m^2; $g=9.81$ m.s^{-2}.

Les réponses du robot (en position, vitesse et accélération) à un échelon unitaire sur un horizon de 10 s sont données ci-après (Fig. I-4, I-5 et I-6).

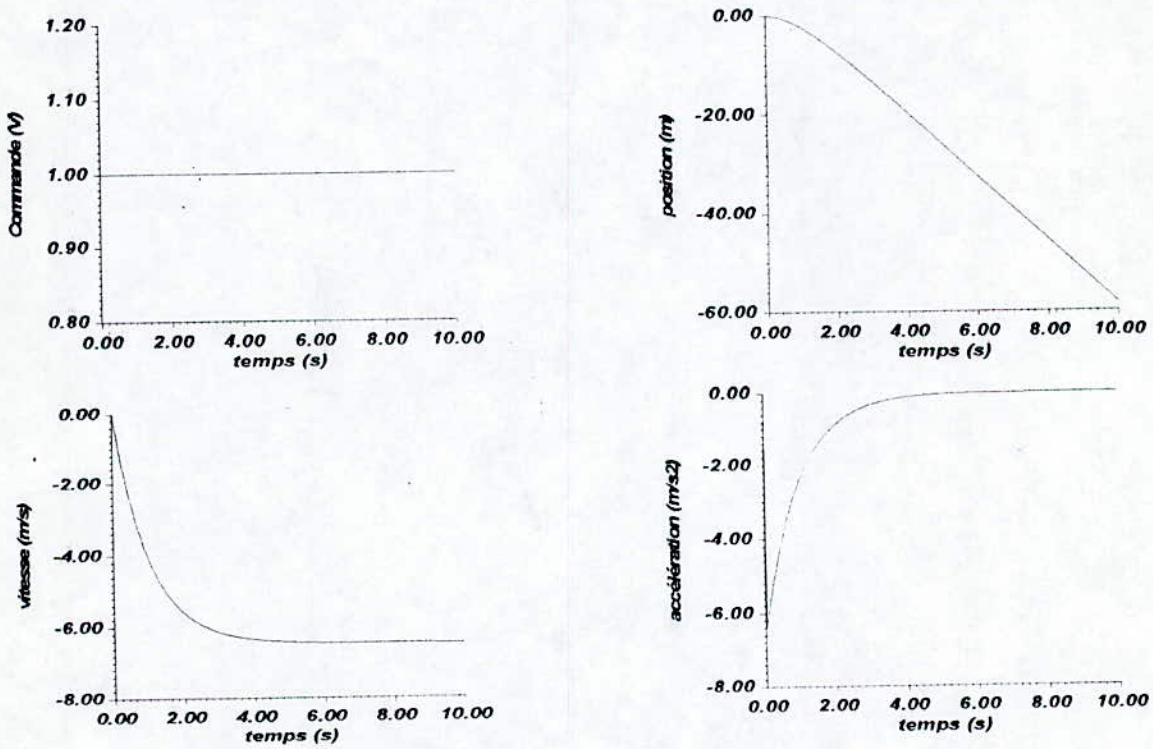


Figure I-4 *Commande, position, vitesse et accélération de l'articulation* 1

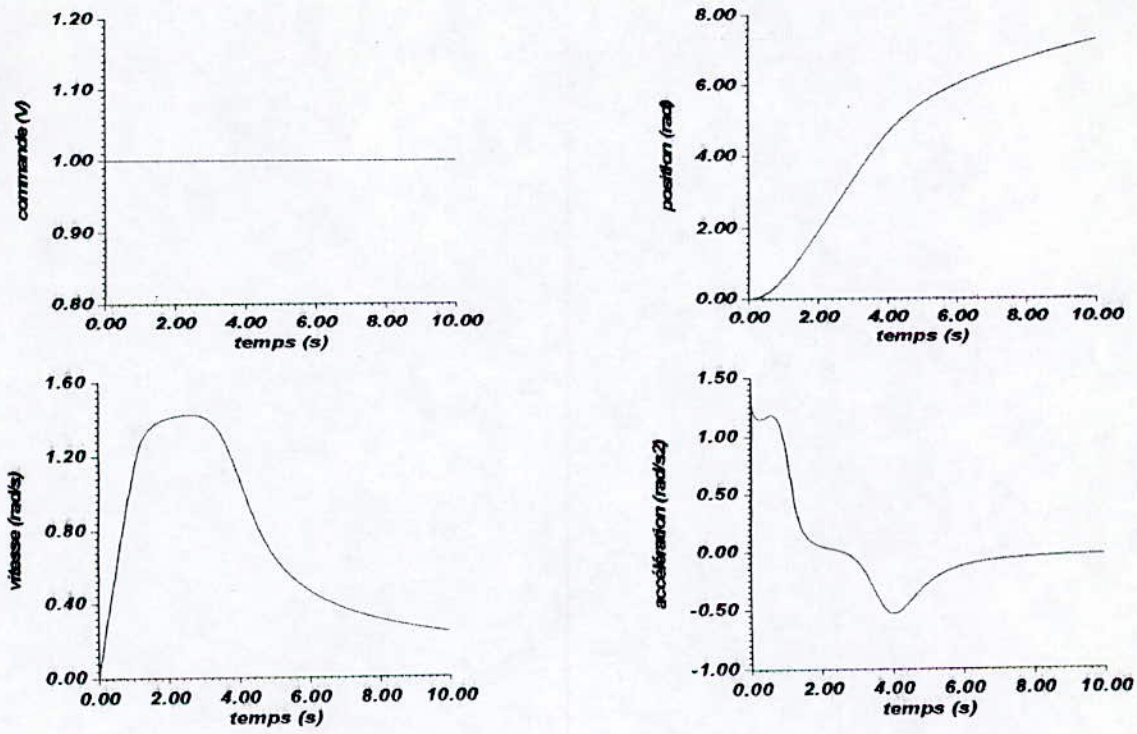


Figure I-5 *Commande, position, vitesse et accélération de l'articulation* 2

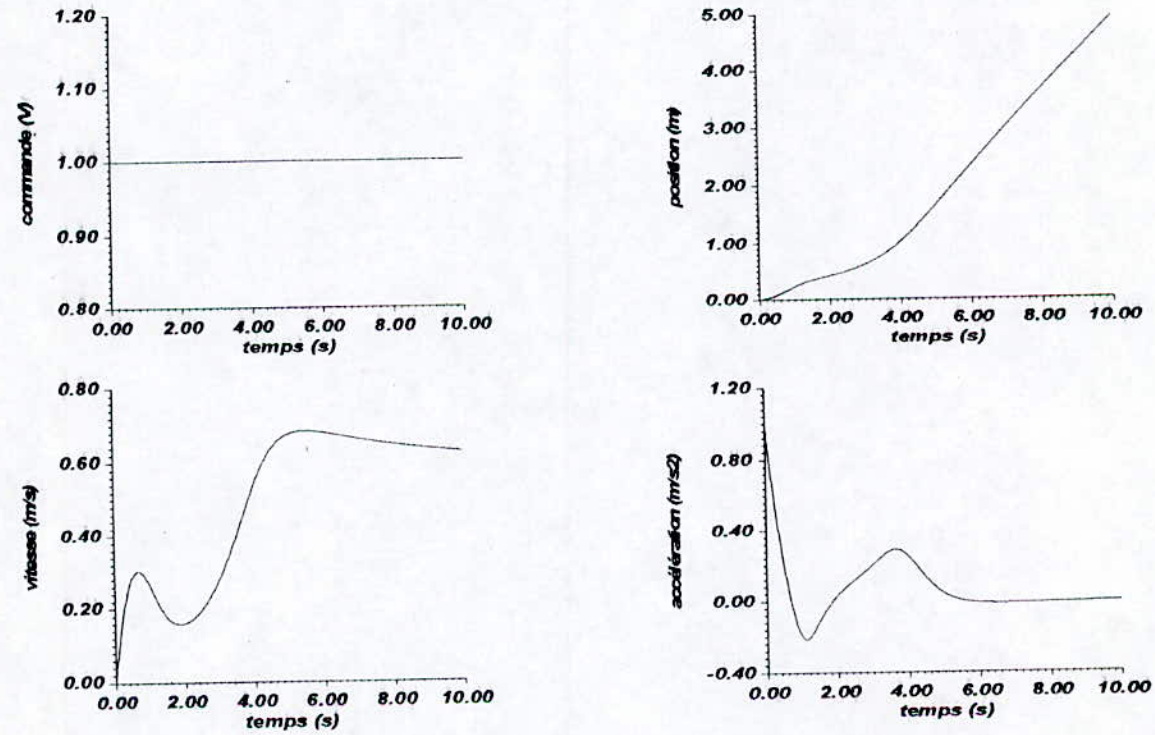


Figure I-6 *Commande, position, vitesse et accélération de l'articulation* 3

I-9 Conclusion :

Ce chapitre a été consacré à la modélisation du robot. Ainsi, après avoir présenté quelques notions sur les robots, les différents modèles de ces derniers ont été élaborés. Le modèle géométrique a permis d'établir la relation entre les coordonnées opérationnelles et les coordonnées généralisées, tandis que le modèle cinématique établit la relation entre les variations de ces grandeurs. Le formalisme d'Euler-Lagrange a été adopté dans le but de la modélisation du comportement dynamique du robot. Ce modèle nous est utile tant pour la simulation en boucle ouverte, que pour la simulation lors de la commande, dans les chapitres qui suivent.

Chapitre II

Etude théorique sur les Réseaux de Neurones Artificiels

II-1 Introduction :

Depuis quelques décennies, les ingénieurs et les informaticiens ont levé un défit qui est la conception d'un calculateur capable de résoudre des problèmes pratiques. En effet, les ordinateurs conventionnels actuels sont limités bien qu'ils aient un pouvoir de calcul puissant. Ils se basent généralement sur une seule unité de traitement qui ne peut effectuer qu'une seule opération à la fois. Ces ordinateurs conviennent bien à l'exécution de séquences d'instructions bien formulées pour eux, mais restent dans l'incapacité d'accomplir certaines tâches qu'un homme ordinaire peut effectuer sans aucune difficulté (la vision, la parole, la reconnaissance de formes....). En effet, un homme est doué d'intelligence. Cette intelligence lui est assuré par un organe très important qui est : *le cerveau*.

En partant de cette idée, les chercheurs ont essayé de comprendre et de modéliser le fonctionnement de cette miraculeuse machine qui est le cerveau humain. Deux groupes de chercheurs ont commencé leurs travaux, chacun ayant sa propre philosophie du problème. Le premier groupe voulait comprendre les méthodes que l'homme utilisait dans son raisonnement, puis analyser ces méthodes de façon logique et avec précision, et enfin programmer ces méthodes. C'est l'école de l'Intelligence Artificielle (IA) qui a soutenu cette idée. Le deuxième groupe voyait qu'il était inutile de faire une programmation directe des activités du cerveau mais il suffisait de construire un ordinateur ayant une structure analogue aux réseaux de neurones biologiques. Puis entraîner cet ordinateur sur les relations et les principes de bases nécessaires pour un être humain. Cet ordinateur sera ensuite capable d'imiter les activités du cerveau. Ainsi, les recherches dans cette voie ont donné naissance par la suite aux Réseaux de Neurones Artificiels.

Dans ce chapitre, nous allons essayer d'éclaircir les notions de neurone et de réseaux de neurones artificiels, leurs caractéristiques ainsi que les méthodes utilisées pour leur apprentissage. Mais d'abord commençons par donner quelques notions de neurophysiologie.

II-2 Notions élémentaires de neurophysiologie :

II-2-1 Physiologie du neurone :

La figure II-1 illustre les constituants principaux d'un neurone biologique. Ce dernier peut être décrit en terme de trois éléments [8]:

- Un corps cellulaire contenant un noyau appelé *cytone*.
- Un axone qui est un prolongement relativement plus long couvert d'une membrane.
- Des dendrites : un autre type de prolongements.

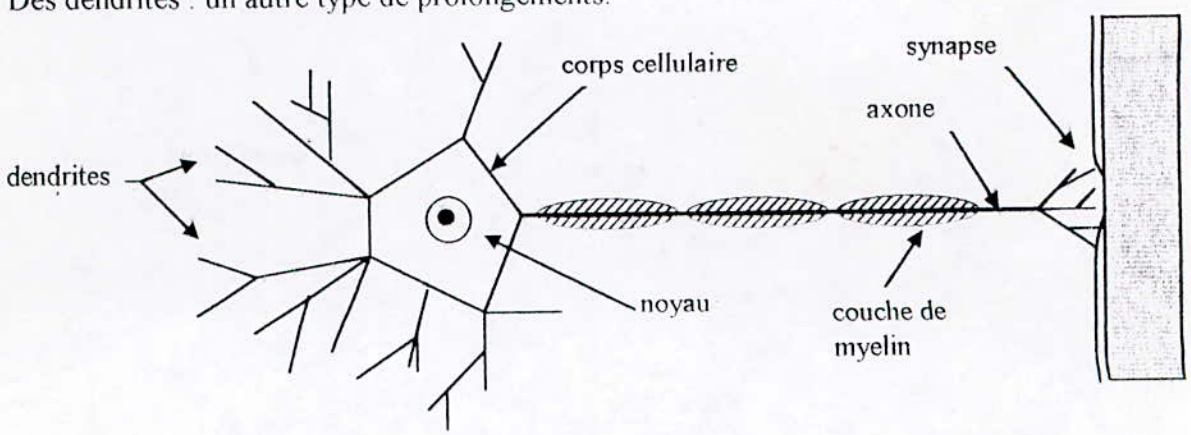


Figure II-1 *Structure générale d'un neurone biologique*

La membrane qui sépare le plasma intracellulaire du fluide extracellulaire joue un rôle important. En effet, grâce à sa perméabilité à certains types d'ions, elle permet de maintenir une différence de potentiel d'équilibre (70 à 100 mV) entre l'intérieur et l'extérieur du neurone. Lorsque le neurone est excité par une impulsion extérieure, cet équilibre est violé et la membrane reprend son travail jusqu'à ce que l'état initial soit rétabli. Ce mécanisme qu'on appelle *potentiel d'action* provoque la transmission d'informations sous forme de potentiel électrique à travers le neurone.

II-2-2 La synapse :

La synapse ou la liaison synaptique est le point d'interconnexion entre deux neurones (fig. II-2). La communication entre un neurone et un autre se fait quand la cellule presynaptique, excitée par un potentiel électrique, libère des substances chimiques dites *neurotransmetteurs* qui

seront absorbées par la cellule postsynaptique. Cette absorption va altérer la perméabilité de la membrane postsynaptique à certains types d'ions, ce qui peut donner naissance à un nouveau signal électrique. En d'autres termes, le message électrique traversant le neurone émetteur devient de nature chimique au niveau de la synapse, puis reprend sa première nature une fois qu'il atteint le neurone récepteur.

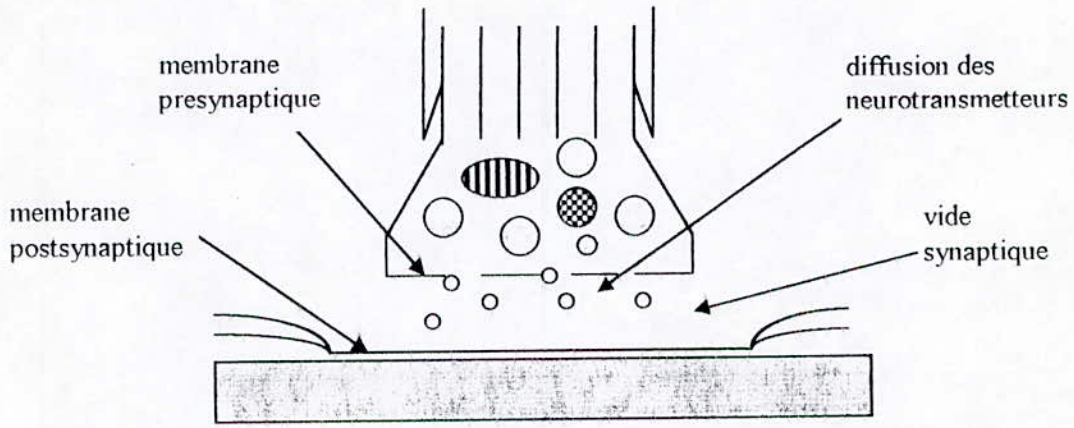


Figure II-2 *La synapse*

II-2-3 Le système nerveux :

Le système nerveux est constitué d'un très grand nombre de neurones fortement interconnectés. En effet, on estime à 10 milliards le nombre de neurones du cerveau humain, et à 36000 le nombre de synapses par neurone. Chaque neurone envoie des impulsions à plusieurs autres neurones (phénomène de divergence) et reçoit des impulsions de plusieurs neurones (phénomène de convergence).

En réalité, les capacités que possède le cerveau et dont un calculateur conventionnel est démuni, ne tiennent pas du neurone en lui-même (ce dernier est très lent par rapport à l'unité de traitement d'un calculateur digital) mais résident dans la distribution du traitement à travers des milliards de neurones qui sont constamment en activité. Le cerveau possède des caractéristiques importantes, qui le diffèrent d'un calculateur digital, parmi lesquelles : l'association, la généralisation et l'auto-organisation. Ce sont ces caractéristiques attrayantes qui ont poussé les chercheurs à essayer de modéliser le cerveau en modélisant son unité de traitement élémentaire qui est le neurone.

II-3 Du neurone biologique au neurone artificiel :

II-3-1 Modèle de McCulloch & Pitts :

Ce fut le premier modèle, proposé par McCulloch & Pitts en 1943 [8]. En se basant sur leur compréhension du neurone biologique, ces deux chercheurs ont suggéré un modèle du neurone, caractérisé par :

- Un sommateur pondéré qui fait la somme pondérée par des gains des potentiels que le neurone reçoit.
- Une fonction seuil : si la somme pondérée dépasse un certain seuil le neurone est activé et fournit un potentiel en sortie, sinon il restera inactif.

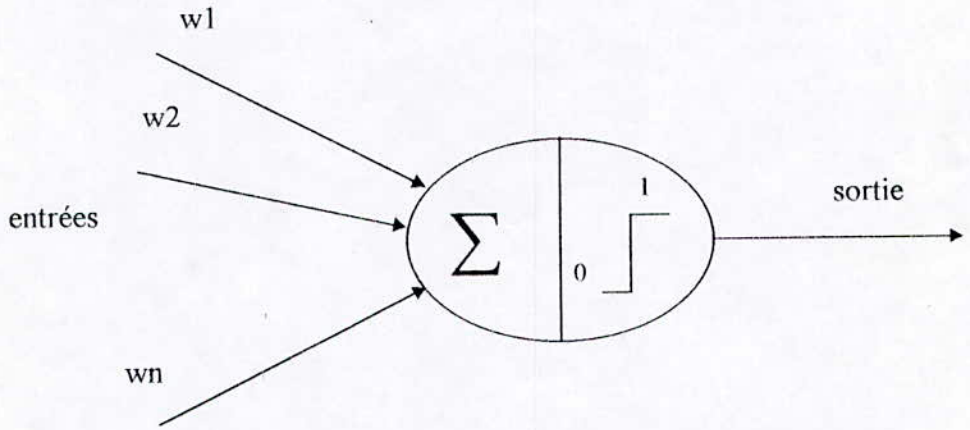


Figure II-3 *Modèle de McCulloch & Pitts*

Le modèle de McCulloch & Pitts a un comportement statique, ce qui n'est pas toujours le cas des neurones biologiques. Ces derniers sont des systèmes dynamiques du fait qu'ils peuvent avoir des rebouclages entre eux. D'où la nécessité d'un modèle plus général.

II-3-2 Modèle général :

Nous allons maintenant décrire un modèle standard et unifié du neurone artificiel. Ce dernier possède 3 composantes [10]:

(1) Un sommateur pondéré.

(2) Un système dynamique linéaire SISO.

(3) Une fonction non linéaire statique dite fonction d'activation.

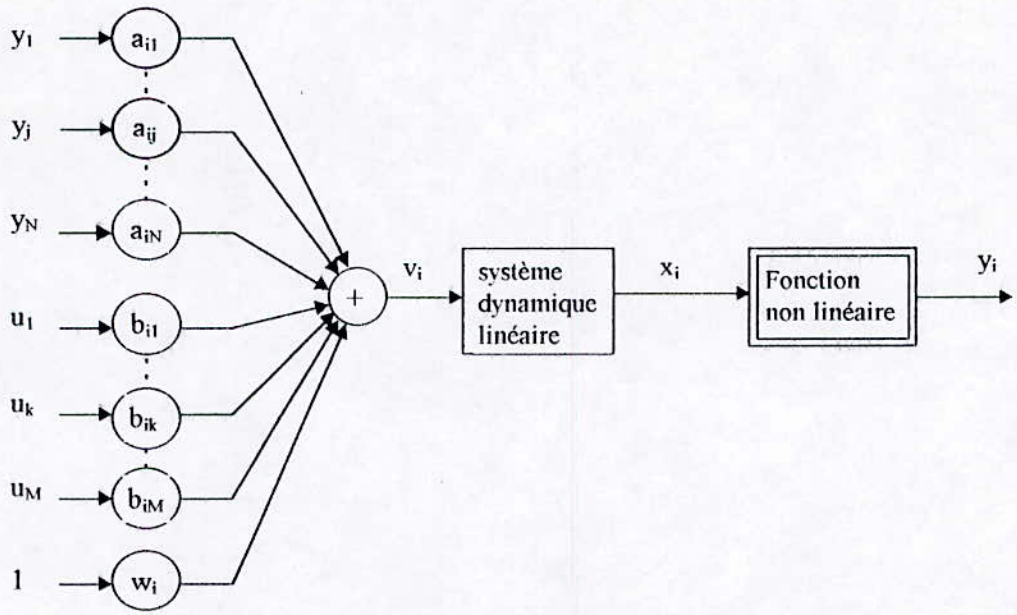


Figure II-4 *Modèle général d'un neurone*

Ces 3 éléments sont décrits ci-dessous.

II-3-2-1 Sommateur pondéré :

Il est décrit par l'équation suivante :

$$v_i(t) = \sum_{j=1}^{N} a_{ij} y_j(t) + \sum_{k=1}^{M} b_{ik} u_k(t) + w_i \qquad \text{(II-1)}$$

où : v_i : somme pondérée du neurone i.

y_j : $j = \overline{1,N}$ sorties de tout les autres neurones reliés au neurone i.

a_{ij} : poids synaptique associé à la liaison entre le neurone j et le neurone i.

u_k : $k = \overline{1,M}$ les entrées du réseau.

b_{ik} : poids synaptique associé à la liaison entre l'entrée k et le neurone i.

w_i : poids synaptique associé à une entrée qui est constamment à 1 (terme de biais).

Sous forme matricielle, l'équation (II-1) devient :

$$v(t) = Ay(t) + Bu(t) + w \tag{II-2}$$

où : le $ij^{ème}$ élément de la matrice A (N×N) est a_{ij}

le $ik^{ème}$ élément de la matrice B (N×M) est b_{ik}

le $i^{ème}$ élément du vecteur colonne v (N×1) est v_i

le $j^{ème}$ élément du vecteur colonne y (N×1) est y_j

le $k^{ème}$ élément du vecteur colonne u (M×1) est u_k

et enfin w est un vecteur colonne (N×1) contenant les biais w_i.

II-3-2-2 Système dynamique linéaire SISO :

Il possède v_i comme entrée et x_i comme sortie et est décrit par la fonction de transfert :

$$H(s) = \frac{X_i(s)}{V_i(s)} \tag{II-3}$$

où $X_i(s)$ et $V_i(s)$ sont respectivement les transformées de Laplace des signaux x_i et v_i. La fonction de transfert H(s) peut prendre l'une des formes suivantes :

$$\begin{aligned} H(s) &= 1 \\ H(s) &= \frac{1}{s} \\ H(s) &= \frac{1}{1+sT} \\ H(s) &= \frac{1}{\alpha_0 s + \alpha_1} \\ H(s) &= e^{-sT} \end{aligned} \tag{II-4}$$

On remarque que les trois premières formes de H(s) ne sont qu'un cas particulier de la quatrième forme.

II-3-2-3 Fonction d'activation :

La fonction d'activation g(.) fournit la sortie y_i du neurone en terme de la sortie x_i de la fonction de transfert :

$$y_i = g(x_i) \qquad \text{(II-5)}$$

On dénombre plusieurs fonctions d'activation dont chacune est réservée à une application particulière. En général cette fonction est monotone, croissante et bornée (à cause de la nature tout ou rien du neurone biologique). On peut faire une double classification des fonctions d'activation, selon qu'elles soient [10]:

- Dérivables / non dérivables
- En forme d'impulsion / en forme d'échelon
- Positives / de moyenne nulle

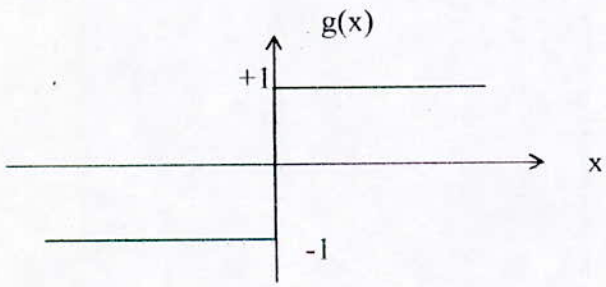
La première classification distingue les fonctions à non linéarité molle des fonctions à non linéarité dure. On a besoin des premières pour la dérivation d'algorithmes d'apprentissage alors que les deuxièmes sont utilisées pour fournir des sorties binaires.

La deuxième classification distingue les fonctions qui n'ont une sortie significative qu'autour de zéro (comme la fonction gaussienne) des fonctions dont le seul changement important est autour de zéro et qui possède des valeurs importantes loin de l'origine.

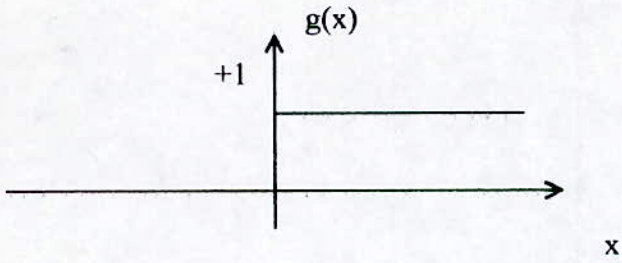
La troisième classification fait la différence entre les fonctions dont la sortie varie entre 0 et 1, et ceux dont la sortie varie entre -1 et 1.

Parmi les fonctions d'activation qui existent nous pouvons citer les suivantes :

- La fonction à seuil :

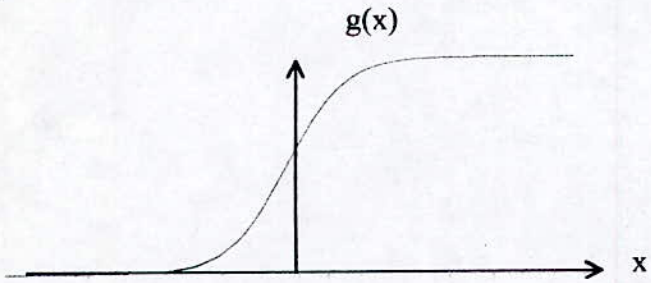


$$g(x) = \begin{cases} +1 & \text{si } x > 0 \\ -1 & \text{ailleurs} \end{cases}$$



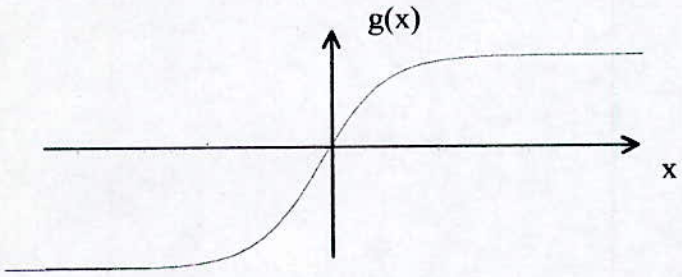
$$g(x) = \begin{cases} +1 & \text{si } x > 0 \\ 0 & \text{ailleurs} \end{cases}$$

- La fonction sigmoïde :



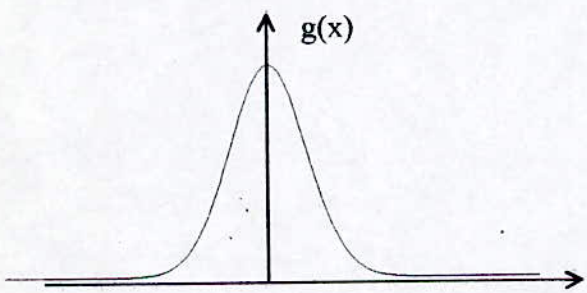
$$g(x) = \frac{1}{1+e^{-cx}} \quad , c > 0$$

- La fonction tangente hyperbolique :



$$g(x) = \tanh(x) = \frac{1-e^{-cx}}{1+e^{-cx}} \quad , c > 0$$

- La fonction gaussienne :



$$g(x) = e^{\left(\frac{-x^2}{\sigma^2}\right)}$$

Dans la plupart des cas, la fonction tangente hyperbolique est utilisée car elle est croissante, bornée et dérivable ce qui permet la dérivation d'algorithmes d'apprentissage.

II-3-3 Réseaux de neurones artificiels :

Dans le cerveau, un très grand nombre de neurones sont interconnectés pour former un réseau et accomplir des activités intelligentes très évoluées. Par analogie, un réseau de neurones artificiels est construit à partir des modèles de neurones (i.e., neurones artificiels) connectés entre eux. A chaque connexion est associé un poids synaptique. La classification des réseaux de neurones artificiels (RNA) peut être effectuée selon deux critères :

1-Premier critère : Dépendance de leur évolution du temps. On distingue alors deux types de réseaux :

1-a) Réseaux statiques :

Leur évolution ne dépend pas du temps. La modification de l'entrée n'entraîne qu'une modification stable de la sortie, mais n'entraîne pas un retour d'information vers cette entrée. Du point de vue architecture, ce type de réseaux est multicouches constitué par des neurones statiques (H(s)=1), de sorte que chaque neurone de la couche i reçoit des informations des neurones de la couche i-1 et est connecté à tous les neurones de la couche i+1 (Fig. II-5).

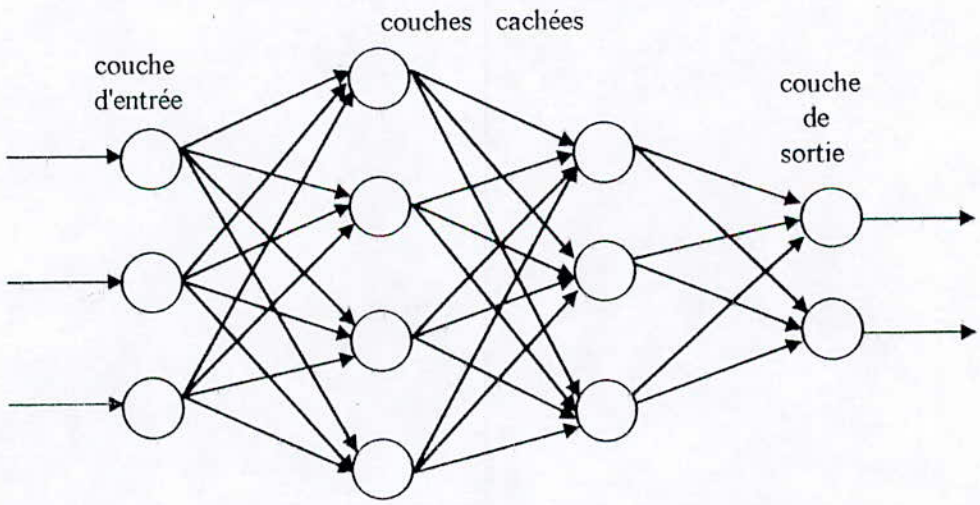


Figure II-5 *Architecture d'un réseau statique*

2-b) Réseaux dynamiques :

Ce sont des réseaux dont l'évolution dépend du temps, ou en d'autre terme, l'état présent du réseau dépend de ces états passés. Donc à priori, ces réseaux contiennent des rebouclages entre neurones (Fig. II-6).

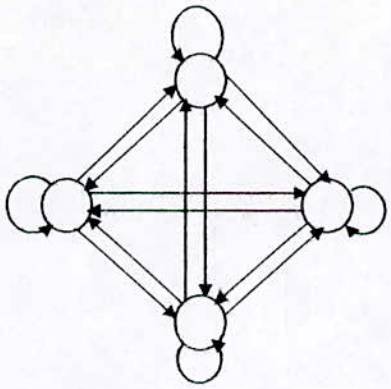


Figure II-6 *Architecture d'un réseau récurrent*

Parmi les réseaux dynamiques, on distingue deux grandes classes : Réseau de Hopfield et Machine de Boltzmann.

Réseau de Hopfield :

C'est une architecture simple qui possède des propriétés très intéressantes, ayant certaines fonctionnalités de type cognitif (mémoire associative). Ce réseau se comporte comme une mémoire associative, restituant une information apprise sous l'effet d'une stimulation qui peut être considérée comme cette information déformée, incomplète ou bruitée. La figure II-7 illustre l'architecture d'un réseau de Hopfield.

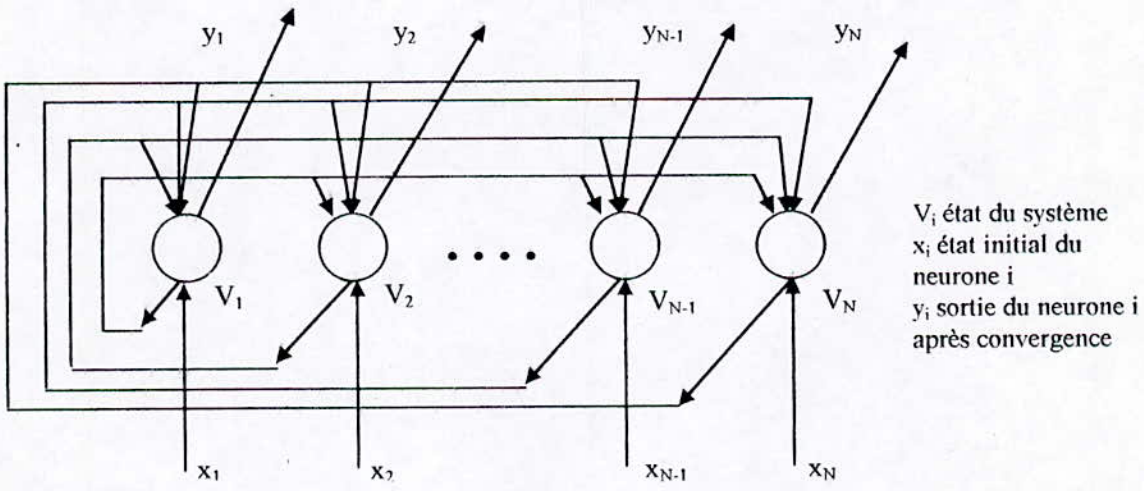


Figure II-7 *Réseau de Hopfield*

Machine de Boltzmann :

La machine de Boltzmann est un réseau de neurones constitué de neurones qui opèrent avec une certaine probabilité, i.e., qui ont un comportement stochastique. Ces neurones peuvent être visibles ou cachés, connectés par des poids synaptiques w_{ij} qui sont symétriques (fig. II-8)

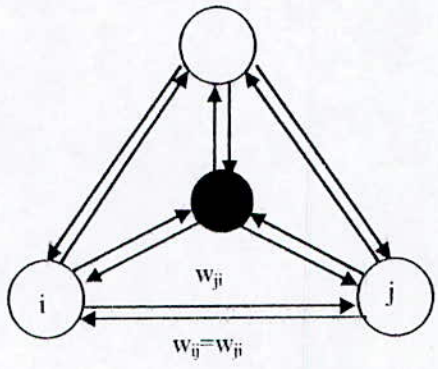


Figure II-8 *Machine de Boltzmann à* 3 *neurones visibles et* 1 *neurone caché*

Dans une machine de Boltzmann, la probabilité que la sortie du $i^{ème}$ neurone prend la valeur 1 est donnée par :

$$P_i = \frac{1}{1 + e^{\left(-\frac{2}{T}s_i\right)}} \qquad \text{(II-6)}$$

où : $\overline{s_i} = \sum_j w_{ij} s_j$ représente l'effet des autres neurones sur le neurone i.

T : est un paramètre correspondant à la température des dynamiques statistiques et qui agit sur la probabilité P_i [11].

L'apprentissage (voir Sec. II-4 pour ce qui est de l'apprentissage des RNA) de se type de réseaux ce fait par la minimisation d'une certaine fonction d'énergie où la température T joue le rôle d'un paramètre.

2-Deuxième critère : Le type d'apprentissage effectué. L'apprentissage peut être supervisé ou non supervisé :

2-a) Apprentissage supervisé :

Dans ce type d'apprentissage, les poids des connexions sont déterminés par le biais d'algorithmes qui consistent à minimiser l'erreur entre la sortie désirée et la sortie du réseau jusqu'à l'obtention d'une performance acceptable.

2-b) Apprentissage non supervisé :

La détermination des poids, dans ce cas, n'est pas en fonction des erreurs, mais en présentant au réseau une quantité suffisante d'exemples contenant des corrélations de sorte que celui-ci en dégage les régularités automatiquement. Ces réseaux sont souvent appelés "auto-organisateurs" (self-organizing neural networks) ou encore à apprentissage compétitif.

II-3-4 Caractéristiques générales des RNA :

Les RNA sont caractérisés par :

1- Le parallélisme :

Dans un RNA plusieurs informations peuvent se propager en parallèle de façon simultanée, à la différence des calculateurs séquentiels classiques qui ne peuvent traiter qu'une seule information à la fois.

2- La capacité d'adaptation :

Grâce a leur capacité d'apprendre, les RNA tiennent compte des nouvelles données du monde extérieur.

3- Pouvoir de généralisation :

A partir des exemples d'apprentissage, les RNA peuvent deviner des points non appris.

4- Résistance aux bruits :

Le réseau de neurone peut filtrer l'information à partir de données bruitées.

5- Systèmes multivariables :

Les réseaux de neurones sont naturellement multivariables. Ils sont directement applicables aux systèmes MIMO.

6- Modélisation non linéaire :

Comme nous avons vu que la fonction d'activation d'un neurone peut être non linéaire, les RNA peuvent approximer n'importe quelle fonction (linéaire ou non linéaire). L'emploi des

réseaux de neurones plutôt que des techniques classiques pour l'approximation de fonction peut être justifié par les arguments suivants :

- Simplicité de mise en œuvre (peu d'analyse mathématique préliminaire).
- Capacité d'approximation universelle prouvée.
- Possibilité de prendre le point de vue "processus = boîte noire".
- Robustesse par rapport à des défaillances internes du réseau (caractère distribué de la représentation).
- Réalisation matérielle parallèle inhérente.

II-4 Apprentissage des Réseaux de Neurones Artificiels :

La caractérisation d'un système par le terme général "réseau de neurones" implique souvent son aptitude à apprendre. L'apprentissage est le processus par lequel le réseau de neurones acquérit la capacité d'effectuer certaines tâches et ce en ajustant ses paramètres internes (poids synaptiques) selon un schéma d'apprentissage spécifique. En fin d'apprentissage, le réseau est sensé nous fournir des sorties aussi proches que possible des sorties désirées.

Il existe plusieurs algorithmes d'apprentissage, parmi lesquels nous pouvons citer : Backpropagation, Fast Backpropagation, recurrent Backpropagation, ROM (Random Optimisation Method),....etc.

Dans cette partie, nous présentons quelques uns de ces algorithmes, avec plus en détail l'algorithme de Backpropagation. C'est ce dernier qui sera utilisé dans la suite de ce travail.

II-4-1 Apprentissage des RNA statiques :

II-4-1-1 Backpropagation :

a) Principe :

Backpropagation (ou rétropropagation de l'erreur) est une méthode d'apprentissage basée sur la minimisation d'un critère quadratique de l'erreur (la somme des carrés des erreurs entre la

sortie désirée et la sortie du réseau). Ceci est fait en changeant continuellement les paramètres du réseau dans la direction de la plus grande descente de l'erreur (procédure de descente du gradient). Le changement de chaque paramètre est proportionnel à sa relative contribution dans la somme des carrés des erreurs.

L'apprentissage par cette méthode s'effectue en deux étapes :

Première étape : un signal d'entrée appliqué au réseau se propage à travers les couches supérieures, pour enfin générer un signal de sortie.

Deuxième étape : La sortie du réseau est comparée avec la sortie désirée et un signal d'erreur est généré. Ce dernier va se propager en sens inverse (d'où le nom Backpropagation) en partant des sorties, provoquant ainsi une modification des poids dans le sens de la minimisation de l'erreur.

b) Dérivation de l'algorithme :

Pour des raisons de simplicité, nous dérivons l'algorithme de Backpropagation pour un réseau à 3 couches. La généralisation à un réseau de plus de 3 couches peut être faite par simple analogie.

Soit donc le réseau à 3 couches de la figure II-9 [8]:

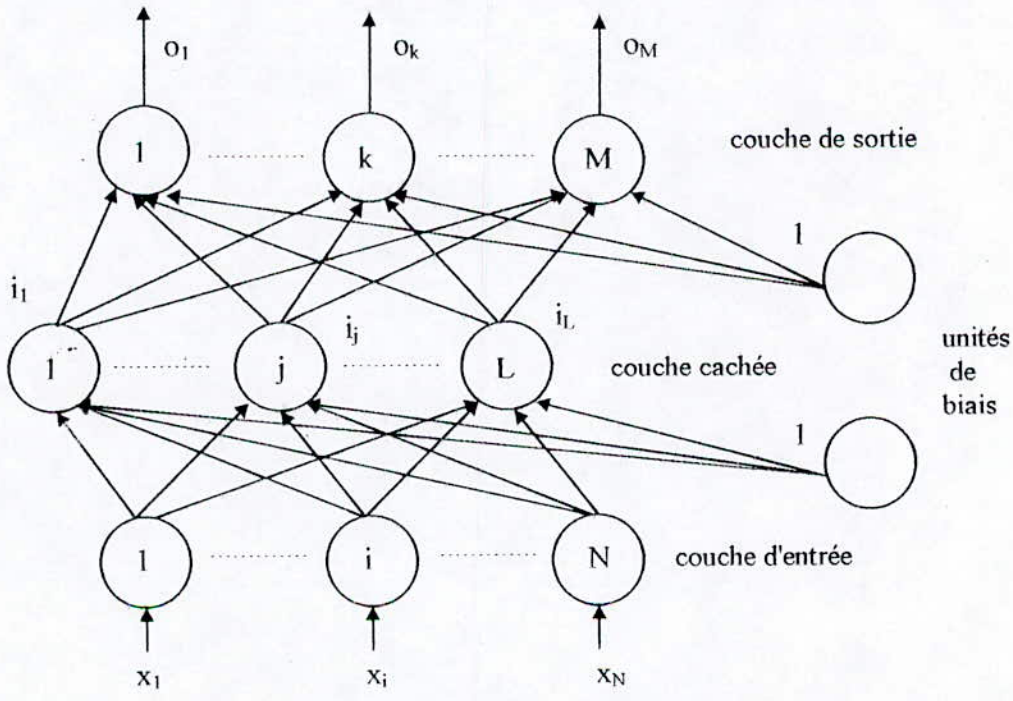


Figure II-9 *Structure d'un réseau à 3 couches*

Nous adoptons les notations suivantes :

q : nombre d'exemple à présenter au réseau lors de l'apprentissage. On sous-entend par "exemple" la paire (X_p, Y_p) où :

$X_p=(x_{p1}\ x_{p2}\ \dots\ x_{pN})^T$ vecteur d'entrée avec N le nombre d'entrées du réseau.

$Y_p=(y_{p1}\ y_{p2}\ \dots\ y_{pM})^T$ vecteur des sorties désirées avec M le nombre de sorties du réseau.

$I_p=(i_{p1}\ i_{p2}\ \dots\ i_{pL})$ vecteur de sortie de la couche cachée avec L le nombre de neurones de la couche cachée.

$O_p=(o_{p1}\ o_{p2}\ \dots\ o_{pM})$ vecteur de sortie du réseau en réponse à X_p.

w^1_{ji} : poids de la connexion entre le $i^{ème}$ neurone de la couche d'entrée et le $j^{ème}$ neurone de la couche cachée.

w^2_{kj} : poids de la connexion entre le $j^{ème}$ neurone de la couche cachée et le $k^{ème}$ neurone de la couche de sortie.

net^1_{pj} : somme pondérée du $j^{ème}$ neurone de la couche cachée, pour l'exemple p.

net^2_{pk} : somme pondérée du $k^{ème}$ neurone de la couche de sortie, pour l'exemple p.

f : fonction d'activation des neurones :

Nous pouvons alors écrire les équations suivantes :

$$\begin{cases} net^1_{pj} = \sum_{i=1}^{N+1} w^1_{ji} x_{pi} \\ i_{pj} = f\left(net^1_{pj}\right) \end{cases} \qquad j = \overline{1,L} \tag{II-7}$$

$$\begin{cases} net^2_{pk} = \sum_{j=1}^{L+1} w^2_{kj} i_{pj} \\ o_{pk} = f\left(net^2_{pk}\right) \end{cases} \qquad k = \overline{1,M} \tag{II-8}$$

Le réseau est entraîné pour minimiser l'erreur totale :

$$E = \frac{1}{2} \sum_{p=1}^{q} \sum_{k=1}^{M} \left(y_{pk} - o_{pk}\right)^2 \tag{II-9}$$

Généralement, au lieu de considérer l'erreur sur tous les exemples, on ne considère que l'erreur sur l'exemple p :

$$E_p = \frac{1}{2}\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)^2 \qquad \text{(II-10)}$$

ceci facilite la dérivation de l'algorithme et conduit au même résultat (c'est la version la plus utilisée dans la littérature). Pour déterminer le changement des poids, on calcul la gradient de E_p par rapport à w_{kj}^2 (pour l'ajustement des poids de la couche de sortie) puis par rapport à w_{ji}^1 (pour l'ajustement des poids de la couche cachée). Pour minimiser l'erreur, on doit faire évoluer les poids dans le sens inverse du gradient.

- Ajustement des poids de la couche de sortie :

$$\begin{aligned}\frac{\partial E_p}{\partial w_{kj}^2} &= -\left(y_{pk} - o_{pk}\right)\frac{\partial f}{\partial\left(net_{pk}^2\right)}\frac{\partial\left(net_{pk}^2\right)}{\partial w_{kj}^2} \\ &= -\left(y_{pk} - o_{pk}\right)f'\left(net_{pk}^2\right)i_{pj}\end{aligned} \qquad \text{(II-11)}$$

et

$$w_{kj}^2(t+1) = w_{kj}^2(t) - \eta\frac{\partial E_p}{\partial w_{kj}^2} \qquad \text{(II-12)}$$

où : η est un paramètre d'apprentissage ajouté pour des considérations pratiques.

d'où, nous obtenons :

$$w_{kj}^2(t+1) = w_{kj}^2(t) + \eta\left(y_{pk} - o_{pk}\right)f'\left(net_{pk}^2\right)i_{pj} \qquad \text{(II-13)}$$

- Ajustement des poids de la couche cachée :

Remarquons d'abord que :

$$\begin{aligned} E_p &= \frac{1}{2}\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)^2 \\ &= \frac{1}{2}\sum_{k=1}^{M}\left(y_{pk} - f\left(net_{pk}^2\right)\right)^2 \\ &= \frac{1}{2}\sum_{k=1}^{M}\left(y_{pk} - f\left(\sum_{j=1}^{L+1} w_{kj}^2 i_{pj}\right)\right)^2 \end{aligned} \qquad \text{(II-14)}$$

On a alors :

$$\begin{aligned} \frac{\partial E_p}{\partial w_{ji}^1} &= -\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)\frac{\partial o_{pk}}{\partial\left(net_{pk}^2\right)}\frac{\partial\left(net_{pk}^2\right)}{\partial i_{pj}}\frac{\partial i_{pj}}{\partial\left(net_{pj}^1\right)}\frac{\partial\left(net_{pj}^1\right)}{\partial w_{ji}^1} \\ &= -f'\left(net_{pj}^1\right)x_{pi}\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)f'\left(net_{pk}^2\right)w_{kj}^2 \end{aligned} \qquad \text{(II-15)}$$

d'où :

$$w_{ji}^1(t+1) = w_{ji}^1(t) + \eta f'\left(net_{pj}^1\right)x_{pi}\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)f'\left(net_{pk}^2\right)w_{kj}^2 \qquad \text{(II-16)}$$

II-4-1-2 Amélioration de Backpropagation :

Il existe plusieurs versions améliorées de Backpropagation qui ont été proposées en vue de surmonter les inconvénients de cette dernière.

a) Fast backpropagation :

Lors de l'implantation de Backpropagation, on a constaté que la diminution de l'erreur était significative durant les premiers cycles d'adaptation (un cycle est un passage sur tous les exemples) , i.e, quand l'estimation de l'erreur était relativement grande. Au fur et à mesure que cette erreur diminuait avec l'apprentissage, la convergence de l'algorithme devenait de plus en

plus lente. Pour éviter ce problème, on utilise l'algorithme de Fast Backpropagation. Ce dernier diffère de celui de Backpropagation par le changement du critère à minimiser (la fonction objective) en vue d'accélérer la convergence même après les premiers cycles d'apprentissage.

Le réseau est entraîné pour minimiser la fonction objective :

$$\begin{aligned} G(\lambda) &= \lambda E + (1-\lambda)E' \\ &= \lambda \sum_{p=1}^{q}\sum_{k=1}^{M} \Phi_2\left(e_{pk}\right) + (1-\lambda)\sum_{p=1}^{q}\sum_{k=1}^{M} \Phi_1\left(e_{pk}\right) \end{aligned} \qquad \text{(II-17)}$$

où : $e_{pk} = y_{pk} - o_{pk}$

$\Phi_2(x) = \frac{1}{2}x^2$

$\Phi_1(\cdot)$ est une fonction définie positive, convexe, continue et dérivable partout.

$\lambda \in [0,1]$

Si λ=1 l'équation (II-17) coïncide avec la fonction objective originale (voir (II-9)).

Si λ=0 l'apprentissage du réseau est basé sur la minimisation de $G(0) = \sum_{p=1}^{q}\sum_{k=1}^{M} \Phi_1\left(e_{pk}\right)$

Quand $0<\lambda<1$ l'équation (II-17) définit une variété de fonctions objectives entre λ=0 et λ=1.

Le problème qui se pose ici est le choix de $G(0) = E'$. On a trouvé que $\Phi_1(\cdot)$ doit être choisie de sorte que $G(0) = E'$ approxime asymptotiquement le critère de l'erreur absolue [9]. Ces exigences sont satisfaites en prenant :

$$\Phi_1(x) = \frac{1}{\beta}\ln\left[\cosh(\beta x)\right] \qquad \text{(II-18)}$$

avec β grand.

Aussi, une loi appropriée pour la réduction de λ de 1 à 0 est la suivante [9]:

$$\lambda = \lambda(E) = e^{\left(-\frac{\mu}{E^n}\right)} \qquad \text{(II-19)}$$

où : μ constante réelle positive.

n nombre entier positif, généralement on le choisi égal à 2.

Pour l'ajustement des poids, le principe est le même que dans Backpropagation, on remplace seulement dans les équations (II-13) et (II-16) $\left(y_{pk} - o_{pk}\right)$ par

$\lambda\left(y_{pk} - o_{pk}\right) + (1-\lambda)\text{tgh}\left[\beta\left(y_{pk} - o_{pk}\right)\right]$.

b) Robust Backpropagation :

C'est une autre version proposée par White en 1989 [13]. Elle est basée sur les statistiques robustes. En effet, White a démontré que Backpropagation est un simple cas d'approximation stochastique, donc elle appartient au domaine des statistiques.

On sous-entend ici par robustesse l'insensibilité du réseau aux échantillons qui sont "étranges". Robust Backpropagation remplace le terme $\left(y_{pk} - o_{pk}\right)$ par une fonction atténuatrice $S\left(y_{pk} - o_{pk}\right)$. Les fonctions :

$$S(x) = \text{tgh}\left(\frac{x}{2}\right) \quad , \quad S(x) = \frac{2x}{1+x^2} \tag{II-20}$$

sont des fonctions statistiquement robustes, alors que $S(x) = x$ (cas de la Backpropagation) ne l'est pas.

c) Backpropagation avec momentum :

Dans cette version, lors du calcul du changement $\Delta w(t)$ des poids, on ajoute une fraction du changement précédent $\Delta w(t-1)$. Ce terme additionnel tend à maintenir une évolution des poids dans la même direction (cela évite de tomber dans des minimums locaux).

II-4-1-3 considérations pratiques :

Avant de commencer l'apprentissage, un certain nombre de paramètres doivent être fixés (dimensionnement du réseau, pas d'apprentissage ...). Ces derniers peuvent engendrer des problèmes s'ils sont mal choisis.

1- Dimensionnement du réseau :

Il n'existe pas de méthodes systématiques pour la détermination du nombre de couches dans le réseau ou bien le nombre de neurones par couche. Néanmoins, il existe des règles pratiques tirées à partir de l'expérience, parmi lesquelles nous pouvons citer:

Règle 1 : plus la relation entre les données en entrée et les sorties désirées est complexe plus il faut augmenter le nombre de neurones par couche.

Règle 2 : pour les réseaux à une seule couche cachée, le nombre de neurones dans la couche cachée est égal à :

$$h = \frac{c}{10(M+N)} \qquad \text{(II-21)}$$

où c : le nombre de vecteurs dans le fichier des exemples.

M: nombre de neurones dans la couche de sortie.

N : nombre d'entrées du réseau.

Règle 3 : le nombre de couches cachées est égal au nombre de relations qui existent entre les neurones de la couche d'entrée et les neurones de la couche de sortie.

2- Paramètre d'apprentissage ou pas de correction η :

Le choix du paramètre d'apprentissage η a un effet important sur les performances du réseau. Généralement, η prend de petites valeurs de l'ordre de 0.05 à .25 pour assurer une convergence vers une solution. Ce paramètre présente en réalité un dilemme : s'il est trop grand, il peut entraîner des oscillations, s'il est trop petit, le temps d'apprentissage devient infini. Pour résoudre ce problème, on choisit η variable, à partir d'une valeur grande qu'on diminuera par la suite.

3- Poids initiaux :

Les poids initiaux sont choisis aléatoirement entre -0.5 et 0.5. Il ne faut jamais initialiser les poids à une même valeur.

4- Minimums locaux :

L'algorithme d'apprentissage peut, ce qui est fréquemment rencontré, converger vers un minimum local du critère d'erreur. Alors l'erreur va stagner autour d'une valeur relativement grande, et les performances requises ne seront jamais atteintes. Pour remédier à ce problème, il existe plusieurs solutions : changer le pas de correction, changer les poids initiaux, utiliser Backpropagation avec momentum, changer la fonction d'activation, augmenter le nombre de couches cachées, augmenter le nombre de neurones par couche.

5- Saturation du réseau :

On sait que les fonctions d'activation dans un réseau de neurones sont généralement bornées, et fournissent des sorties entre 0 et 1 ou entre -1 et 1. Donc, il faut mettre à l'échelle les données présentées au réseau.

Par exemple, si quelqu'un présente au réseau une entrée égale 20000, même avec des poids faibles, les sommes seraient grandes et les sigmoïdes (ou autre fonction d'activation) seraient saturées. A ce moment là, leur dérivées seraient nulles et l'apprentissage va s'arrêter. Ce cas pourrait se présenter quand le pas de correction η ou les valeurs initiales des poids sont très grands.

II-4-2 Apprentissage des RNA dynamiques :

II-4-2-1 Recurrent Backpropagation :

La popularité qu'a connu l'algorithme de Backpropagation a motivé l'application de la méthode de descente du gradient pour l'apprentissage des réseaux dynamiques. Le comportement dynamique d'un réseau récurrent peut être décrit par l'ensemble des équations différentielles non linéaires couplées [9]:

$$\tau \frac{dz_i}{dt} = -z_i + \rho(u_i) + I_i \quad ; \text{i parcourant tous les neurones} \qquad \text{(II-22)}$$

où : z_i : l'activité du $i^{ème}$ neurone.

I_i : terme de biais externe.

$\rho(\cdot)$: fonction d'activation du neurone i.

$u_i = \sum_j w_{ij} z_j$ somme pondérée du $i^{ème}$ neurone.

Quand le système d'équations (II-22) atteint son équilibre on aura $\frac{dz_i}{dt} = 0$, donc :

$$z_i^* = \rho\left(u_i^*\right) + I_i = \rho\left(\sum_j w_{ij} z_j^*\right) + I_i \qquad \text{(II-23)}$$

où l'indice (*) indique l'état d'équilibre.

Les poids synaptiques du réseau sont ajustés sur la base de la minimisation d'une erreur quadratique E évoluée par rapport à l'état d'équilibre :

$$E = \frac{1}{2}\sum_k E_k^2 \qquad \text{(II-24)}$$

où : $E_k = y_k - z_k^*$ avec k parcourant les neurones de la couche de sortie et y_k l'état présent de ces neurones.

Il a été trouvé [9] que l'adaptation des poids synaptiques dépend de l'évolution au cours du temps d'un réseau auxiliaire dont le comportement dynamique est décrit par :

$$\frac{dv_i}{dt} = -v_i + \sum_p v_p \rho'\left(u_p\right) w_{pi} + E_i \qquad \text{(II-25)}$$

Cette équation différentielle contient une somme de non linéarités, en opposition à l'équation (II-22) qui contient la non linéarité d'une somme. Le terme de biais externe, dans le cas du réseau auxiliaire, est l'erreur E_i obtenue à partir du réseau initial.

On peut résumer l'algorithme de Recurrent Backpropagation comme suit :

- Les états stables z_i^* des neurones sont déterminés en laissant le système de l'équation (II-22) évoluer dans le temps jusqu'à atteindre son état d'équilibre.

- Les états stables v_i^* du réseau auxiliaire (équation (II-25)) sont déterminés en laissant ce dernier évoluer vers son état d'équilibre et ce en donnant l'erreur E_i résultante de l'équilibre du réseau initial.

- Les poids synaptiques du réseau sont ajustés par la formule :

$$\Delta w_{pq} = \alpha \rho'\left(u_p^*\right) v_p^* z_q^* \qquad \text{(II-26)}$$

où : α est une constante réelle positive.

II-5 Les réseaux de neurones et la commande :

Les facultés dont jouissent les réseaux de neurones, ont fait de ceux-ci un outil très intéressant pour la commande des systèmes (en particulier les systèmes non linéaires). Dans cette partie, on passe en revue 5 méthodes principales de commande par RNA [14]:

1- Commande supervisée :

Un réseau de neurone artificiel apprend à imiter un être humain ou un programme de calcul qui connaît déjà comment accomplir une tâche donnée.

2- Commande inverse :

Un RNA apprend le modèle inverse d'un système. Ensuite ce réseau est mis en cascade avec le système pour le commander. Si le réseau a bien appris le modèle inverse, la sortie du système sera très proche de la référence fournie par un opérateur humain ou un programme de calcul à l'entrée du réseau.

3- Commande adaptative par RNA :

Dans ce type de commande, un RNA remplace certains blocs dans les schémas de la commande adaptative classique.

4- Backpropagation through time :

L'utilisateur spécifie une fonction d'utilité ou une performance de mesure à maximiser et fournit un modèle de l'environnement extérieur. Backpropagation est utilisée pour calculer les dérivées d'utilités sommées à travers tous les instants futurs et ce par rapport aux actions en cours (actuelles). Ces dérivées sont alors utilisées pour adapter le RNA dont les sorties sont les actions (commandes) ou pour adapter un programme de commande.

5- Méthodes adaptatives critiques :

Dans ce cas aussi, l'utilisateur fournit une fonction ou une mesure à maximiser. Ce problème d'optimisation est résolu en adaptant un RNA additionnel, appelé *réseau critique*. Il évalue le progrès que le système est entrain de faire. En d'autre terme, la sortie du réseau critique peut être vue comme une seconde fonction d'utilité qui, en quelque sorte, représente la somme de la fonction d'utilité originale à travers tous les instants futurs. Le réseau dont les sorties sont les commandes est adapté en vue de maximiser cette seconde fonction d'utilité dans le futur immédiat.

II-6 Conclusion :

Dans ce chapitre, des notions relatives aux réseaux de neurones artificiels ont été présentées. Des rappels de neurophysiologie ont permi d'établir l'analogie entre les réseaux de neurones artificiels et les réseaux de neurones biologiques. En fait, un neurone artificiel n'est rien d'autre qu'un modèle approché du neurone biologique. Dans un second temps, nous avons classifié les réseaux de neurones. Ces derniers peuvent être statiques ou dynamiques, à apprentissage supervisé ou à apprentissage non supervisé. Une partie importante qui est l'apprentissage des réseaux de neurones a été discutée, en mettant l'accent sur l'algorithme de Backpropagation (car c'est ce dernier qui nous intéresse le plus). Enfin, a cause de l'intérêt apporté aux réseaux de neurones dans la commande des systèmes, nous avons présenté, brièvement, le principe de quelques unes des commandes par réseaux de neurones.

Chapitre III

Commande décentralisée supervisée par Réseaux de Neurones Artificiels

III-1 Introduction :

Dans ce chapitre, nous allons présenter une méthode de commande par réseaux de neurones basée sur la supervision d'une loi de commande déjà existante . La structure décentralisée est utilisée dans le sens où le système à commander est constitué de plusieurs sous-systèmes interconnectés. Pour chacun de ces sous-systèmes, une commande supervisée par réseaux de neurones sera synthétisée. Le contrôleur de chaque sous-système n'aura accès qu'aux informations locales.

Dans un premier lieu, la structure de commande décentralisée par réseaux de neurones ainsi que des notions sur la commande supervisée sont exposées. Pour appliquer cette commande, deux approches seront présentées, dans le sens de superviser deux lois de commande différentes. La première loi est celle d'un régulateur adaptatif. La seconde est la commande linéarisante basée sur la méthode du couple calculé. Pour cette dernière, et dans le but de faire une comparaison entre l'approche centralisée et l'approche décentralisée, on fait la supervision dans les deux cas : centralisé et décentralisé.

L'application de cette commande à un bras de robot se fera dans le chapitre qui suit.

III-2 Structure de la commande décentralisée par RNA :

Dans la commande des systèmes complexes, la notion de décentralisation joue un rôle important. Cette notion considère le système comme étant un ensemble de sous-systèmes interconnectés, qui peuvent être géométriquement éloignés. Chaque sous-système sera commandé par une station de commande locale sur laquelle une contrainte de flux d'information est imposée, i.e., elle n'a accès qu'aux mesures des états locaux [21]. Ce type de transfert d'informations aux stations de commande diffère de celui de la commande centralisée classique, dans laquelle le contrôleur peut mesurer toutes les sorties du système et peut commander toutes ses entrées.

La contrainte de décentralisation imposée aux systèmes complexes est due , d'une part, à la difficulté de communication entre les sous-systèmes (systèmes de puissance, systèmes de transport,...) et, d'autre part, au volume de calcul important qui rend une commande centralisée classique impraticable.

Pour ce qui est de la commande décentralisée par réseaux de neurones, la station de commande locale dont nous avons parlé ci-dessus est tout simplement un réseau de neurones artificiel (Fig. III-1). Chaque réseau de neurones est dédié à la commande d'un sous-système et ne reçoit que des informations locales.

L'utilisation des réseaux de neurones dans une structure de commande décentralisée s'avère très intéressante. En effet, grâce à leur pouvoir de généralisation, les réseaux de neurones ont tendance à compenser les interconnections entre sous systèmes.

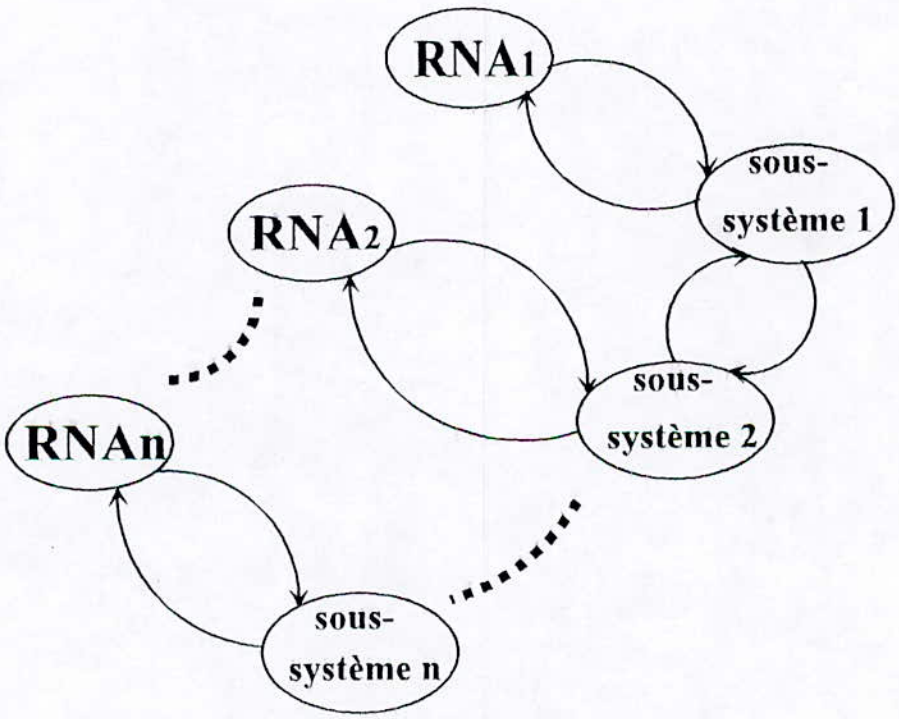


Figure III-1 *Schéma synoptique de la commande décentralisée par RNA*

III-3 Commande supervisée :

III-3-1 Notions et définitions :

Dans la commande supervisée, un réseau de neurones artificiels copie un régulateur existant "copying an existing controller" [14]. Ce régulateur, qui sait déjà comment commander le système, peut être un opérateur humain ou un programme de calcul ("automated controller").

Ce type de commande est implanté, généralement, en 2 phases :

- **Phase d'apprentissage :** lors de cette phase, un réseau de neurones apprend la fonction entrées/sorties du régulateur existant. Pour cela, un fichier d'exemples est généré en laissant ce régulateur commander le système. Puis un algorithme d'apprentissage supervisé sera utilisé.
- **Phase de commande :** une fois l'apprentissage effectué, le régulateur est enlevé et le réseau de neurones est mis à sa place pour commander le système.

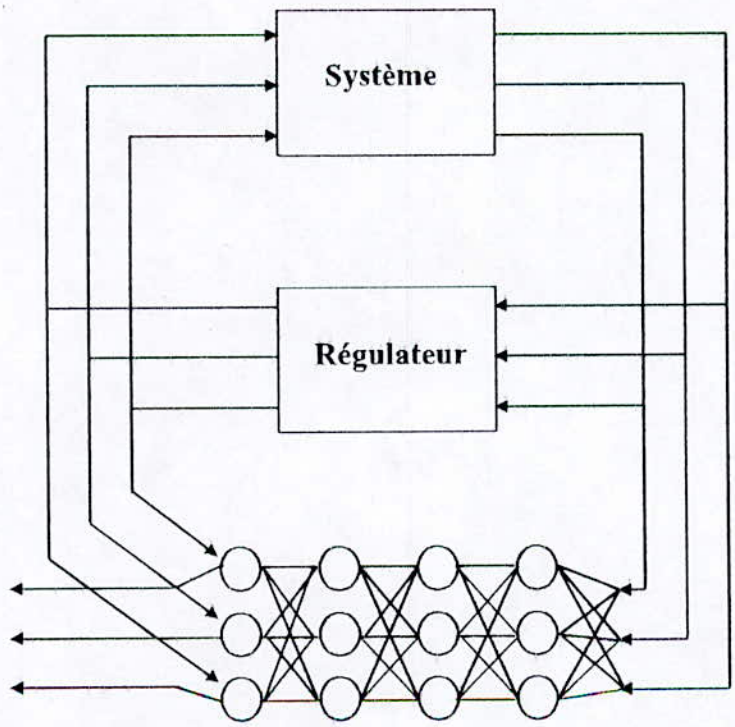


Figure III-2 *Commande supervisée*

L'un des premiers à avoir utiliser ce type de commande était Widrow en 1960 en l'appliquant à un pendule inversé [11]. Widrow a référé à cette commande comme une méthode de construction d'un système expert neuronal par l'acquisition du savoir-faire d'un expert existant. Guez et Selinsky [15] ont fourni une version plus améliorée et plus sophistiquée de la commande supervisée. Ils ont mi en évidence la capacité d'un réseau de neurones à apprendre à commander un système dynamique à partir d'exemples générés par des lois de commande linéaire, non linéaire et par un opérateur humain. Jorgenson [14] a lui aussi fourni un excellent exemple de la commande supervisée dans le domaine de l'aéronautique.

III-3-2 Utilité de la commande supervisée :

La question concernant l'utilité de cette méthode de commande peut être posée. En effet, puisqu'un régulateur effectif existe déjà, et sait accomplir sa tâche, quel est l'intérêt d'avoir une copie de ce régulateur sous forme d'un réseau de neurones? En réalité, il existe plusieurs réponses à cette question [13],[14],[15]:

1) Le régulateur existant peut être un élément impraticable lors de l'utilisation (comme un opérateur humain). Guez et Selinsky ont obtenu d'excellents résultats en supervisant une loi de commande de type "opérateur humain" [15].

2) Comme les calculs (transmission d'informations) dans un réseau de neurones se font en parallèle, la prise de décision et la réaction du réseau aux changements se fera très rapidement, contrairement à un opérateur humain ou un programme de commande implanté dans un calculateur séquentiel.

3) Il n'est pas nécessaire que la loi de commande à superviser soit donnée sous forme explicite, car l'apprentissage des réseaux de neurones se fait à partir d'exemples. Donc, à priori, on peut expliciter sous forme de réseau de neurones (équations mathématiques) une loi de commande présentée sous forme d'exemples. Ceci permettra une analyse.

4) Le caractère distribué de la représentation des réseaux de neurones fournit une meilleure robustesse par rapport aux perturbations (internes et externes) et prévient la dégradation de l'information.

5) Robustesse naturelle relativement aux paramètres non modélisés, due au pouvoir de généralisation des réseaux de neurones.

6) Le réseau de neurones peut former une loi de commande sur la base d'une représentation d'état du système qui est plus facile à mesurer que la représentation requise par le régulateur existant.

Dans ce qui suit, nous allons présenter 2 approches de commande supervisée décentralisée.

III-4 Première approche : supervision d'une loi de commande adaptative (régulateur adaptatif).

III-4-1 Commande décentralisée adaptative [16]:

Le schéma de cette commande décentralisée est présenté sur la figure III-3.

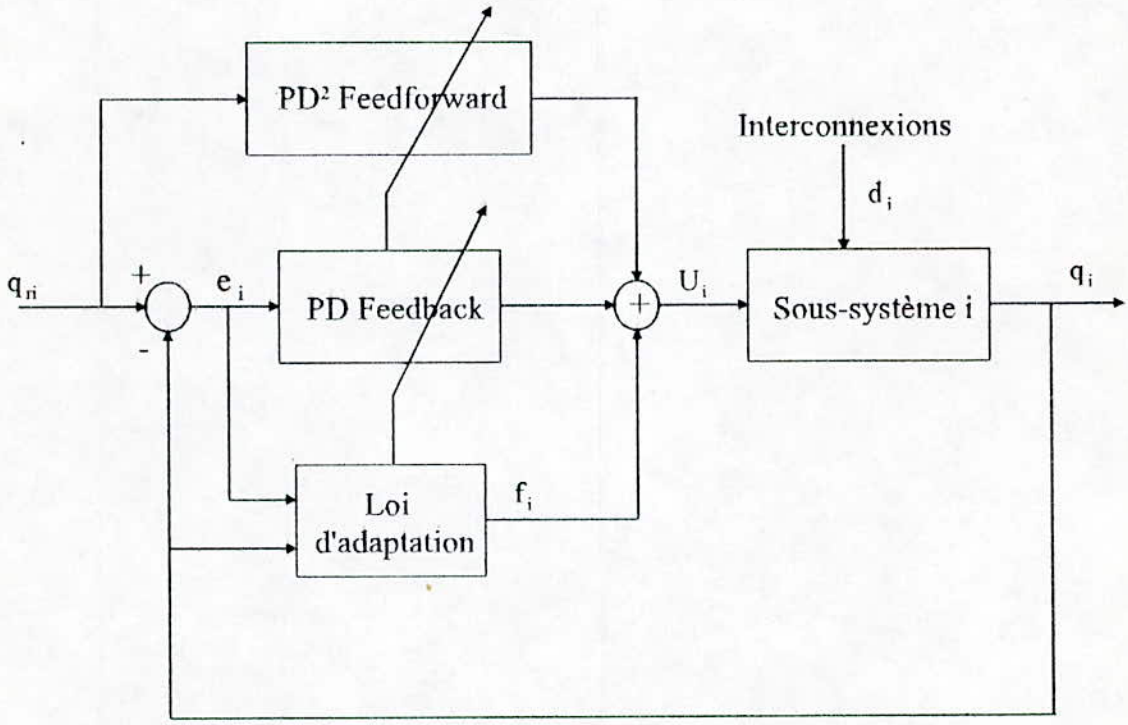


Figure III-3 *Schéma de la commande adaptative décentralisée pour le sous-système i*

Le régulateur adaptatif indépendant dédié à la commande du sous-système i est décrit par :

$$U_i(t) = f_i(t) + \left[k_{i0}(t)e_i(t) + k_{i1}(t)\dot{e}_i(t)\right] + \left[p_{i0}(t)q_{ri}(t) + p_{i1}(t)\dot{q}_{ri}(t) + p_{i2}(t)\ddot{q}_{ri}(t)\right] \qquad \text{(III-1)}$$

où : $e_i(t) = q_{ri}(t) - q_i(t)$ est l'erreur de poursuite en position du sous-système i.

$q_{ri}(t)$: trajectoire désirée.

$q_i(t)$: sortie du sous-système i.

Cette loi de commande est composée de 3 termes :

1) Le terme $f_i(t)$ qui est un signal auxiliaire généré par le schéma d'adaptation pour améliorer les performances de poursuite et compenser la perturbation d_i.

2) Le terme $\left[k_{i0}(t)e_i(t) + k_{i1}(t)\dot{e}_i(t)\right]$ dû à un régulateur PD ("feed-back controller") à gains $k_{i0}(t)$ et $k_{i1}(t)$ adaptatifs.

3) Le terme $\left[p_{i0}(t)q_{ri}(t) + p_{i1}(t)\dot{q}_{ri}(t) + p_{i2}(t)\ddot{q}_{ri}(t)\right]$ fourni par un régulateur "feedforward" à gains ajustables.

Les lois d'adaptation de ces 3 termes, en vue d'assurer une poursuite asymptotique de la trajectoire désirée, sont basées sur une erreur pondérée $r_i(t)$ définie par :

$$r_i(t) = w_{pi}e_i(t) + w_{vi}\dot{e}_i(t) \qquad \text{(III-2)}$$

Pour ajuster le signal auxiliaire :

$$f_i(t) = f_i(0) + \delta_i \int_0^t r_i(t)dt + \rho_i r_i(t) \qquad \text{(III-3)}$$

Pour ajuster les gains du régulateur PD ("feed-back controller") :

$$k_{ij}(t) = k_{ij}(0) + \alpha_{ij} \int_0^t r_i(t)e_i^{(j)}(t)dt + \beta_{ij} r_i(t)e_i^{(j)}(t), \qquad j = 0,1 \qquad \text{(III-4)}$$

Pour ajuster les gains du régulateur "feedforward" :

$$p_{ij}(t) = p_{ij}(0) + \gamma_{ij}\int_0^t r_i(t)\,q_{ri}^{(j)}(t)dt + \lambda_{ij} r_i(t)\,q_{ri}^{(j)}(t)\,, \qquad j = 0,1,2 \tag{III-5}$$

où : δ_{ij}, α_{ij}, γ_{ij} constantes positives.

ρ_i, β_{ij}, λ_{ij} constantes positives ou nulles.

w_{pi}, w_{vi} constantes positives reflétant l'importance relative des erreurs en position et en vitesse dans l'expression de $r_i(t)$.

Du point de vue implantation, le signal auxiliaire $f_i(t)$ peut être généré par un régulateur PID à gains fixes. En effet, en utilisant les équations (III-2) et (III-3) on obtient :

$$f_i(t) = f_i(0) + \rho_i\left[w_{pi}e_i(t) + w_{vi}\dot{e}_i(t)\right] + \delta_i\int_0^t\left[w_{pi}e_i(t) + w_{vi}\dot{e}_i(t)\right]dt \tag{III-6}$$

ou encore :

$$f_i(t) = f_i(0) + \left[\rho_i w_{pi} + \delta_i w_{vi}\right]e_i(t) + \left[\rho_i w_{vi}\right]\dot{e}_i(t) + \left[\delta_i w_{pi}\right]\int_0^t e_i(t)dt \tag{III-7}$$

Ainsi, la loi de commande (III-1) peut être écrite sous forme d'une combinaison d'un régulateur PID ("feed-back controller") et d'un régulateur PD^2 ("feedforward controller") :

$$U_i(t) = U_i(0) + \left[\bar{k}_{ip} + \bar{k}_{iI}\int_0^t dt + \bar{k}_{iv}\frac{d}{dt}\right]e_i(t) + \left[p_{i0} + p_{i1}\frac{d}{dt} + p_{i2}\frac{d^2}{dt^2}\right]q_{ri}(t) \tag{III-8}$$

où : $\begin{cases} \bar{k}_{ip} = k_{i0} + \rho_i w_{pi} + \delta_i w_{vi} \\ \bar{k}_{iI} = \delta_i w_{pi} \\ \bar{k}_{iv} = k_{i1} + \rho_i w_{vi} \end{cases}$ sont les gains ajustables du PID.

et $U_i(0)=f_i(0)$

III-4-2 Commande supervisée décentralisée :

Pour superviser la loi de commande (III-8), on a besoin d'un réseau de neurones pour chaque sous-système i. Chaque réseau n'ayant accès qu'aux informations locales.

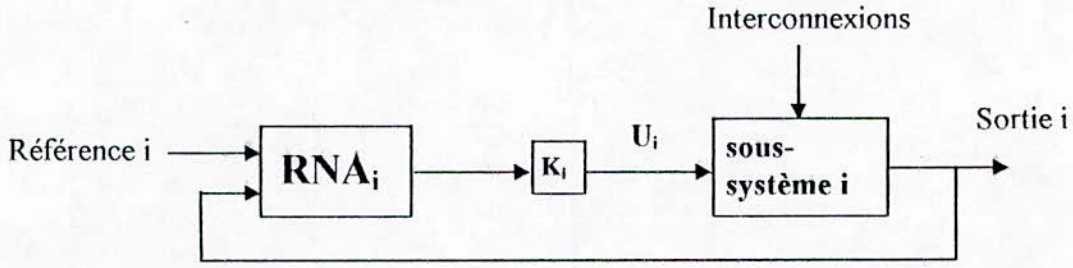


Figure III-4 *Schéma de la commande décentralisée supervisée par RNA pour le sous-système i*

Le bloc K_i dans la figure III-4 est un gain ajouté pour ajuster, éventuellement, la sortie du réseau à la commande désirée.

Pour la structure du réseau, et vue l'équation (III-8) donnant l'expression de la loi de commande à superviser, nous allons utiliser un réseau à 5 entrées et 1 seule sortie. Les entrées sont les grandeurs : $e_i(t)$, $\dot{e}_i(t)$, $q_{ri}(t)$, $\dot{q}_{ri}(t)$, $\ddot{q}_{ri}(t)$, et la sortie sera la commande. Cela permet au réseau de neurones d'implanter le régulateur existant, en lui fournissant le maximum d'informations. L'un peut poser la question sur les gains ajustables, comment le réseau peut en tenir compte? En réalité le réseau tient compte de la variation de ces gains. En effet, leur adaptation est fonction de l'erreur et de la trajectoire désirée (voir les équations (III-4) et (III-5)) qui ont été déjà considérées comme entrées du réseau. Donc, il ne reste qu'à apprendre au réseau un fichier d'exemples du régulateur.

III-5 Deuxième approche : supervision d'une loi de commande non linéaire.

Dans ce paragraphe, nous allons présenter la méthode du couple calculé ainsi que la supervision de cette loi par réseaux de neurones artificiels.

III-5-1 Commande linéarisante "Computed torque method" [17],[18]:

Un robot manipulateur est généralement modélisé comme un ensemble de n corps rigides (liaisons) connectés en série par des articulations, avec une extrémité fixée au sol et l'autre extrémité libre (organe terminal). L'équation dynamique du robot est donnée par :

$$\tau = M(q)\ddot{q} + h(q,\dot{q}) + F \qquad \text{(III-9)}$$

où : τ : vecteur des couples appliqués par les actionneurs.

M : matrice d'inertie du manipulateur.

h : vecteur représentant les forces centrifuges, de coriolis et de frottement.

q : vecteur des positions.

$\dot{q}$: vecteur des vitesses.

F : vecteur des termes inconnus dus aux dynamiques non modélisées et aux perturbations externes.

La loi de la commande linéarisante est donnée par les équations suivantes :

$$\begin{cases} \tau = \hat{M}(q)u + \hat{h}(q,\dot{q}) \\ u = \ddot{q}_r + k_p(q_r - q) + k_v(\dot{q}_r - \dot{q}) = \ddot{q}_r + k_p e + k_v \dot{e} \end{cases} \qquad \text{(III-10)}$$

où : $\hat{M}$ et $\hat{h}$ sont respectivement les estimées de M et h. Elles sont utilisées pour la compensation non linéaire.

q_r, $\dot{q}_r$, $\ddot{q}_r$ sont les vecteurs de position, vitesse et accélération désirées.

k_p et k_v sont des matrices diagonales constantes, avec les éléments dans la diagonale positifs.

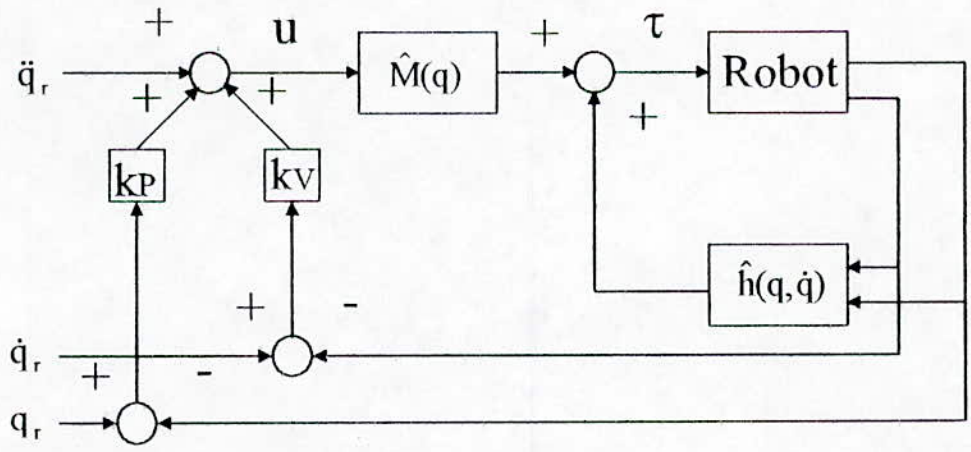


Figure III-5 *Commande par couple calculé*

La dynamique de l'erreur est obtenue en combinant les équations (III-9) et (III-10) :

$$\hat{M}\left(\ddot{q}_r + k_p e + k_v \dot{e}\right) + \hat{h} = M\ddot{q} + h + F \qquad \text{(III-11)}$$

d'où :

$$\ddot{e} + k_v \dot{e} + k_p e = \hat{M}^{-1}\left(\left(M - \hat{M}\right)\ddot{q} + h - \hat{h} + F\right) \qquad \text{(III-12)}$$

On peut facilement voir que l'erreur sera asymptotiquement nulle, si et seulement si F=0 et $\hat{h}, \hat{M}$ sont égaux à h et M. Ce type de commande n'est pas robuste par rapport aux incertitudes non structurées. Les incertitudes non structurées correspondent aux dynamiques non modélisées (frottement non linéaire, retard négligé, modes haute fréquence du manipulateur...).

L'implantation de cette loi de commande sous forme de réseau de neurones, peut en partie résoudre ces problèmes.

III-5-2 Supervision de la commande linéarisante :

III-5-2-1 Dans le cas centralisé :

Dans ce cas, un seul réseau de neurones sera utilisé pour copier cette loi de commande. Ce réseau aura autant de sorties que de sous-systèmes à commander, et aura besoin d'informations sur tous les sous-systèmes. Sous ces considérations, on peut imaginer en conséquence la dimension de ce réseau qui va augmenter et deviendra très grande quand le nombre de sous-système augmente. Cela bien sûr peut causer des problèmes au cours de l'apprentissage.

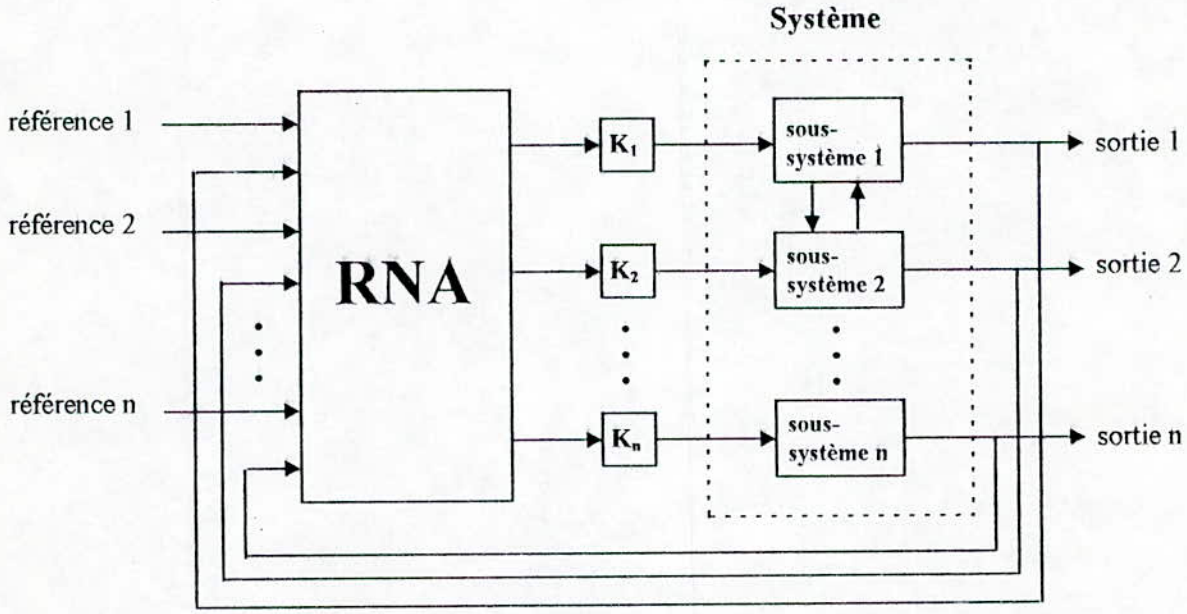


Figure III-6 *Schéma de la commande centralisée supervisée par RNA*

Les gains $K_1, K_2,\ldots,K_n$ sont ajoutés pour ajuster, éventuellement, les sorties du réseau.

III-5-2-2 Dans le cas décentralisé :

L'approche décentralisée pose moins de contraintes que l'approche centralisée. En effet, chaque sous-système sera commandé par un réseau de neurones à part, qui aura une dimension acceptable. Si le nombre de sous-système augmente, on n'a qu'à augmenter le nombre de réseaux

de neurones. Pour chacun de ces derniers l'apprentissage peut être implanter sur un processeur à part, indépendamment des autres RNA.

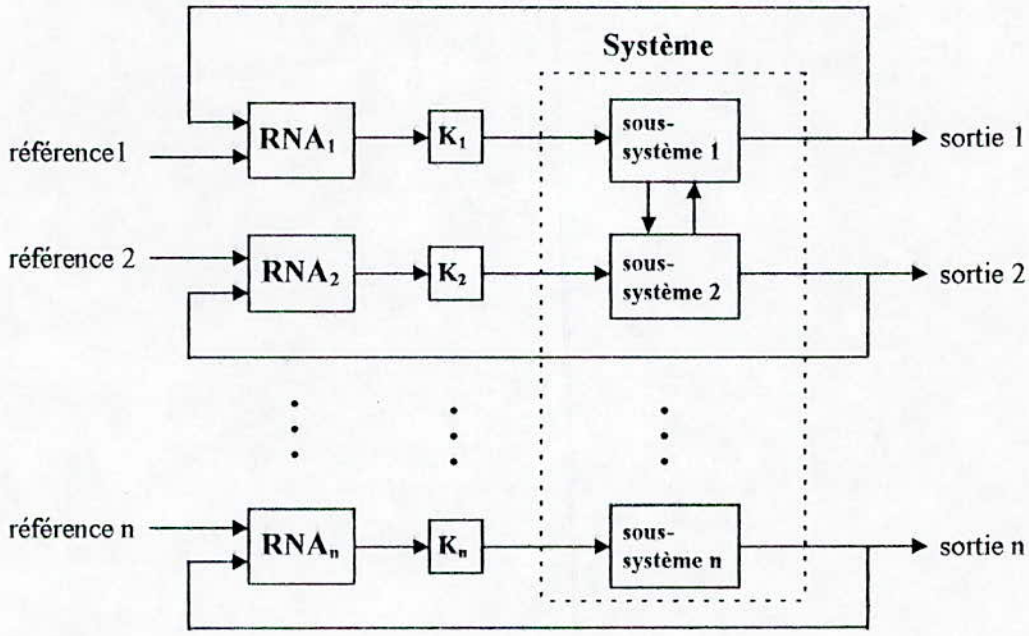


Figure III-7 *Schéma de la commande décentralisée supervisée par RNA*

Le nombre d'entrées de chaque RNA sera égal au nombre de variables locales apparaissant dans l'expression de la loi de commande linéarisante.

III-6 Conclusion :

Dans ce chapitre, nous avons présenté le principe de la commande supervisée par RNA qui consiste simplement à imiter un système de commande existant, en utilisant les propriétés d'approximation et d'apprentissage des réseaux de neurones. Deux approches pour la commande supervisée décentralisée ont été explicitées. Ils trouveront leur application dans le chapitre suivant.

Chapitre IV

Application de la commande décentralisée supervisée par RNA à un bras de robot manipulateur

IV-1 Introduction :

On consacre ce chapitre à l'application de la commande décentralisée supervisée par RNA, décrite dans le chapitre précédant, au robot de classe 4 présenté dans le premier chapitre. Pour cela, des simulations ont été faites ainsi que des tests de robustesse.

IV-2 Première approche : supervision de la loi de commande adaptative.

La loi de commande à superviser est donnée par l'équation (voir Sec. III-4-1) :

$$U_i(t) = U_i(0) + \left[\bar{k}_{ip} + \bar{k}_{it} \int_0^t dt + \bar{k}_{iv} \frac{d}{dt} \right] e_i(t) + \left[p_{i0} + p_{i1} \frac{d}{dt} + p_{i2} \frac{d^2}{dt^2} \right] q_{ri}(t) \qquad \text{(IV-1)}$$

Pour appliquer cette commande et générer les exemples d'apprentissage, le tableau ci-dessous contient les valeurs des constantes utilisées.

	w_p	w_v	δ	α_0	α_1	γ_0	γ_1	γ_2	β_0	β_1	λ_0	λ_1	λ_2
art. 1	400	200	50	100	200	5	5	5	1	1	1	1	1
art. 2	80	40	50	60	100	10	10	10	1	1	1	1	1
art. 3	80	40	50	60	100	3	3	3	1	1	1	1	1

Pour faire la supervision de cette loi, nous avons utilisé 3 réseaux de neurones artificiels. Chaque réseau ayant 5 entrées, une couche cachée de 10 neurones et une seule sortie (Fig. IV-1).

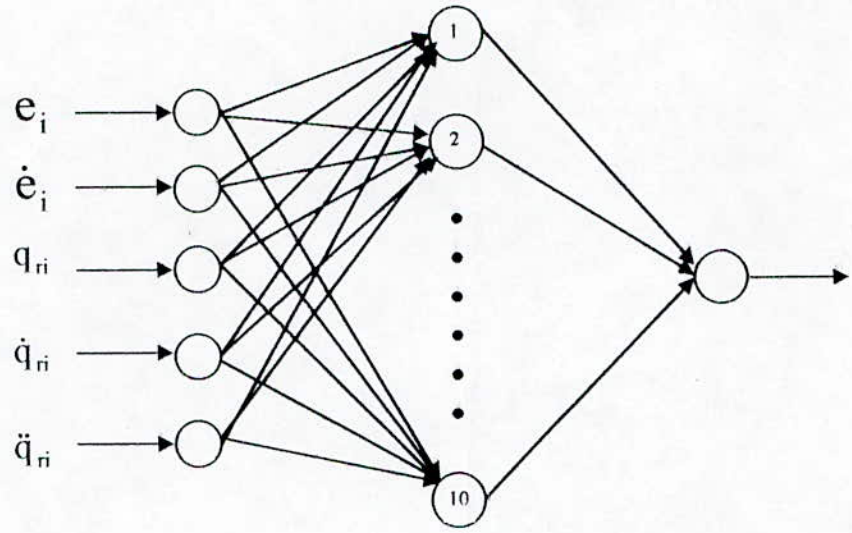
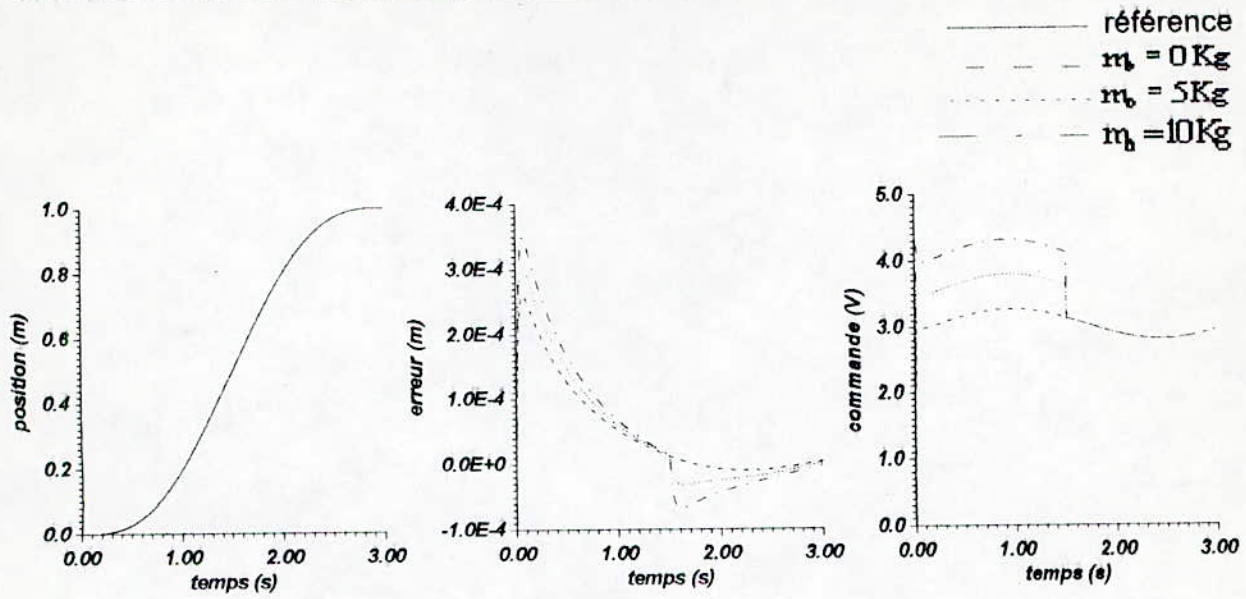


Figure IV-1 *Structure du réseau local utilisée pour l'articulation i (i=*1, 2, 3*)*

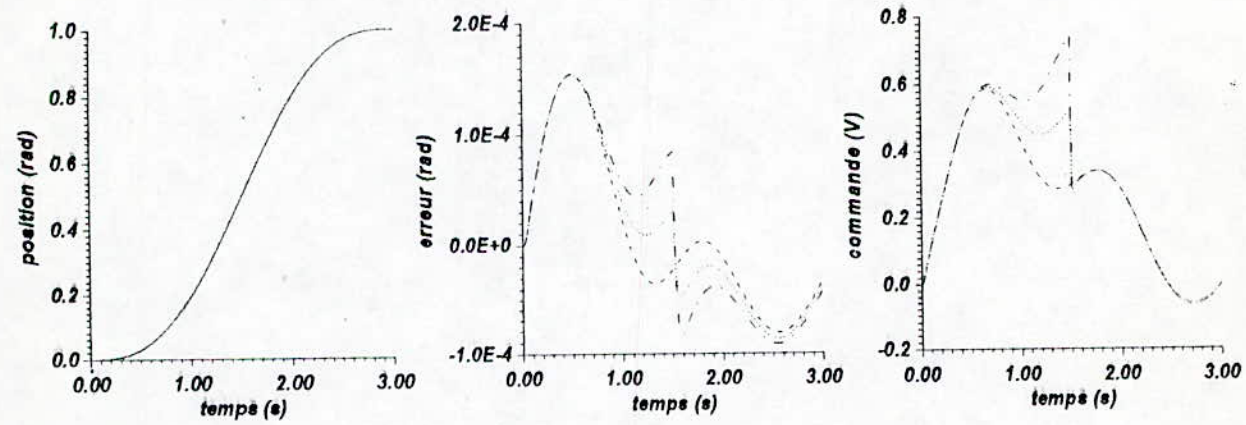
L'apprentissage a été fait par la méthode de Backpropagation en utilisant 150 exemples avec un pas d'apprentissage η de 0.2 pour la première couche et 0.1 pour la deuxième couche. Les poids ont été initialisés aléatoirement entre -0.5 et0.5. Les gains K_i pour ajuster les sorties des réseaux (voir Sec. III-4-2) sont respectivement 4, 1 et 1 pour les RNA 1, 2 et 3. Le nombre d'itérations effectuées lors de l'apprentissage est 1200, 3600 et 600 pour les RNA 1, 2 et 3 respectivement.

On présente sur la figure IV-2 les résultats obtenus par la commande décentralisée adaptative, puis sur la figure IV-3 les résultats obtenus en supervisant cette loi de commande. Dans les deux cas, des tests de robustesse ont été effectués avec une masse m_0 de valeurs : 0, 5 et 10 Kg. Cette masse sera lâchée au milieu du parcourt (t=1.5 s).

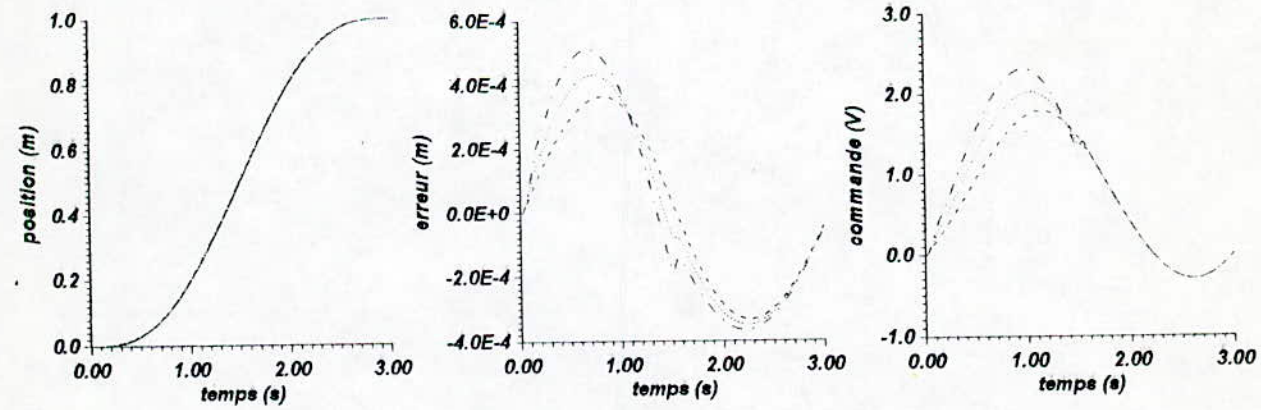
La figure IV-4 illustre une comparaison, pour m_0=10 Kg, entre le régulateur adaptatif décentralisé et le régulateur neuronal décentralisé.



(a) Articulation 1

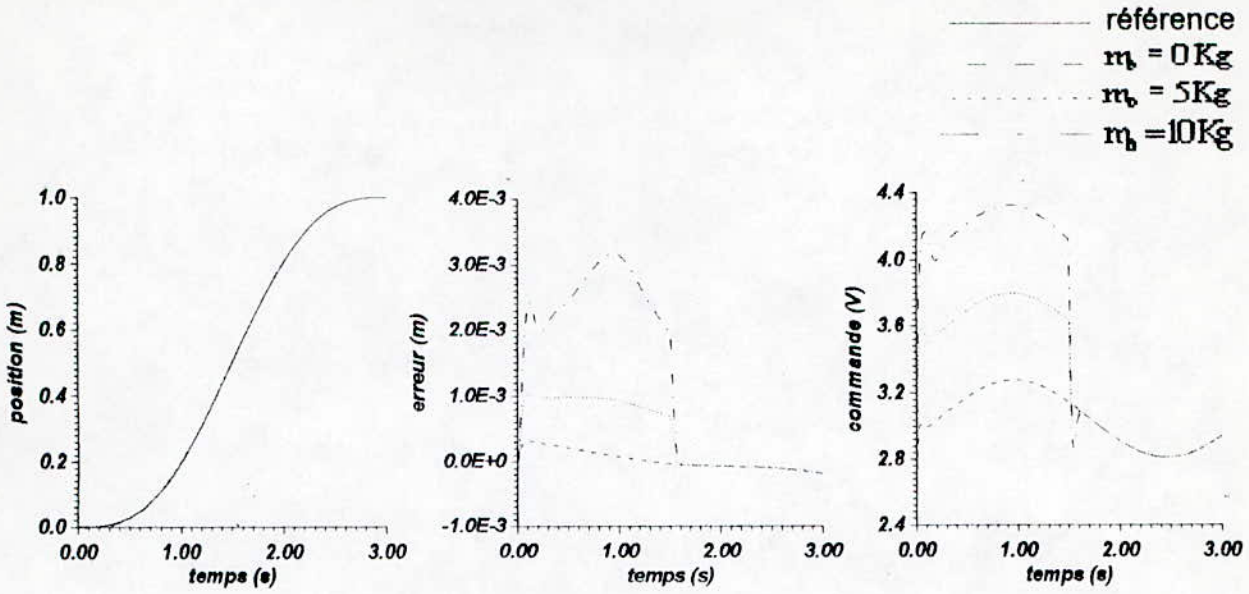


(b) Articulation 2

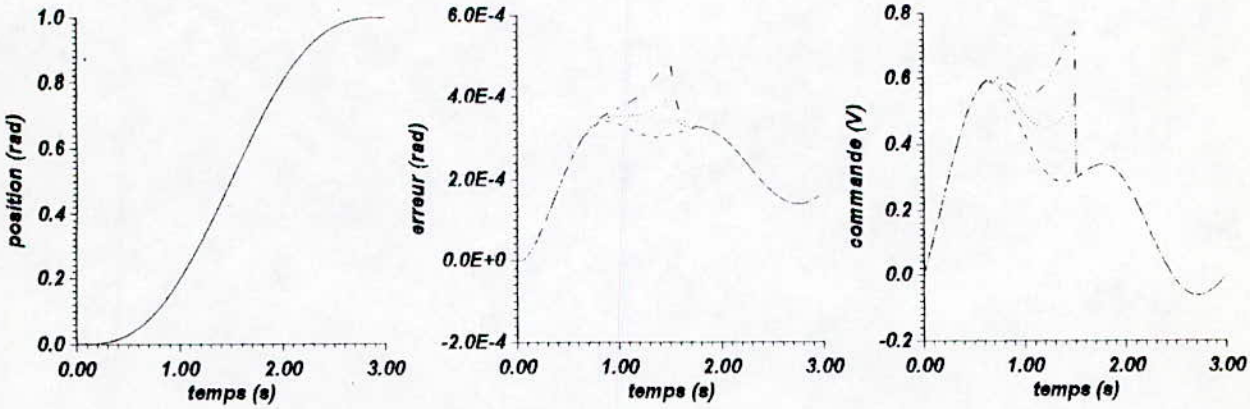


(c) Articulation 3

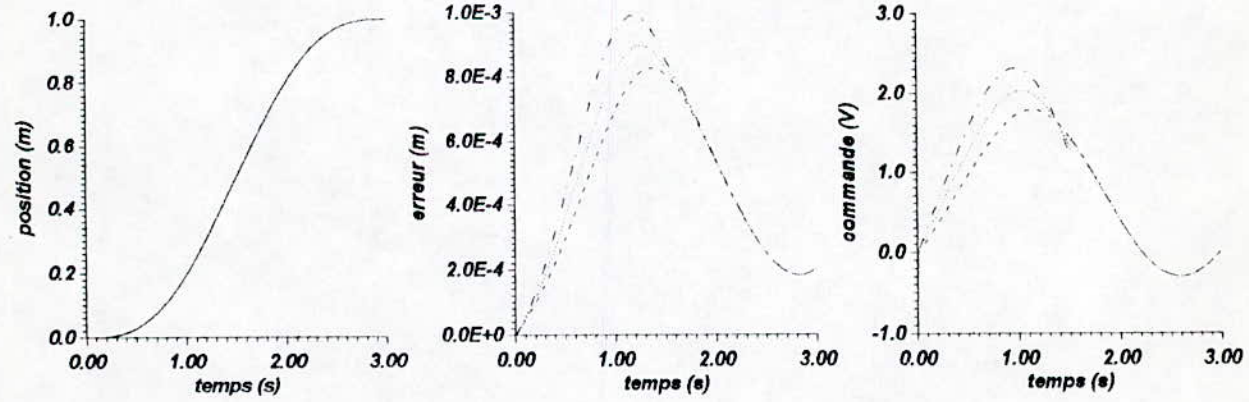
Figure IV-2 *Résultats de la commande décentralisée adaptative*



(a) Articulation 1

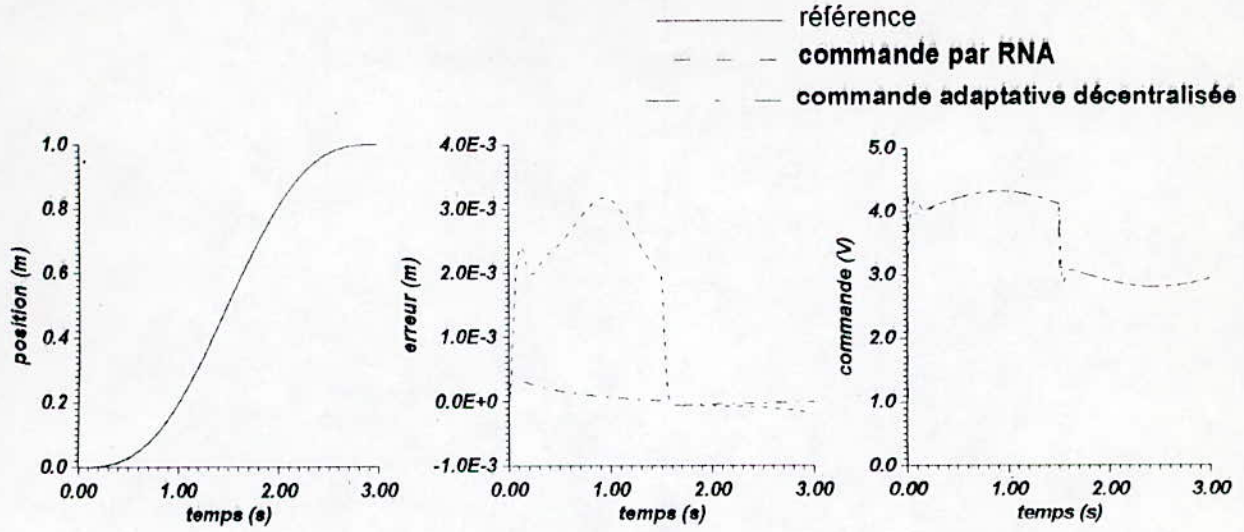


(b) Articulation 2

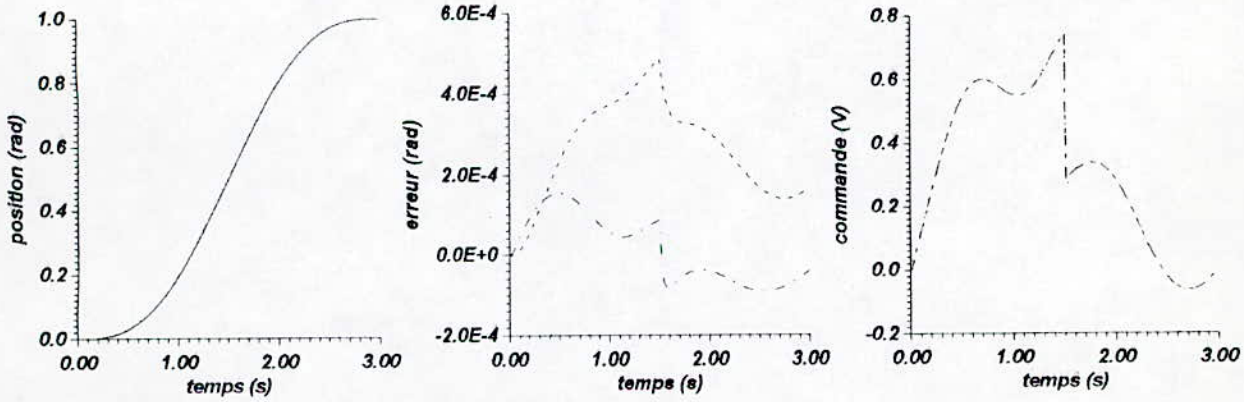


(c) Articulation 3

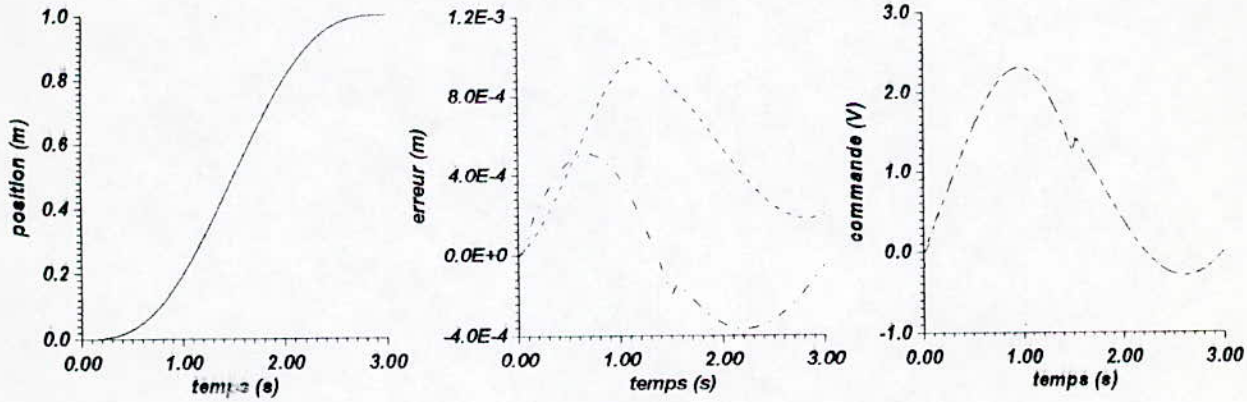
Figure IV-3 *Résultats de la commande décentralisée supervisée*



(a) Articulation 1



(b) Articulation 2



(c) Articulation 3

Figure IV-4 *Comparaison entre la commande décentralisée adaptative et la commande décentralisée supervisée pour* m_0=10 *Kg*

Commentaires :

On constate que le réseau de neurone donne des résultats satisfaisants et bien que le régulateur existant soit adaptatif (gains ajustables), nous avons pu l'implanter sous forme de réseau fixe (gains fixes). Ce réseau donne des performances très proches et présente un caractère robuste (il n'y a pas de perte de performance et de robustesse).

Ici, on note également l'avantage du temps de calcul des réseaux de neurones. En effet, en plus de son architecture parallèle, le réseau nous a évité une étape de calcul supplémentaire qui est celle de l'ajustement des gains du régulateur adaptatif.

IV-3 Deuxième approche : supervision de la loi de commande linéarisante.

On rappelle l'expression de cette loi de commande (voir Sec. III-5-1) :

$$\begin{cases} \tau = \hat{M}(q)u + \hat{h}(q,\dot{q}) \\ u = \ddot{q}_r + k_p(q_r - q) + k_v(\dot{q}_r - \dot{q}) = \ddot{q}_r + k_p e + k_v \dot{e} \end{cases} \qquad \text{(IV-2)}$$

Pour le robot de classe 4, et en utilisant l'équation (I-32), on obtient :

$$\hat{M}(q) = \begin{bmatrix} m_1 + m_3 + m_0 & 0 & 0 \\ 0 & J_2 + J_3 + m_3(q_3 - l_2)^2 + m_0 q_3^2 & 0 \\ 0 & 0 & m_3 + m_0 \end{bmatrix} \qquad \text{(IV-3)}$$

$$\hat{h}(q,\dot{q}) = \begin{bmatrix} (m_1 + m_3 + m_0)q + f_1\dot{q}_1 \\ 2\left(m_3(q_3 - l_2) + m_0 q_3\right)\dot{q}_2\dot{q}_3 + f_2\dot{q}_2 \\ -\left(m_3(q_3 - l_2) + m_0 q_3\right)\dot{q}_2^2 + f_3\dot{q}_3 \end{bmatrix} \qquad \text{(IV-4)}$$

Pour la synthèse de la loi de commande linéarisante ainsi que pour générer les exemples d'apprentissage, nous avons utilisé k_{pi}=20, k_{vi}=10 (i=1, 2, 3), m_0=0 et nous avons négligé l'existence du terme F (terme dû aux dynamiques non modélisées, voir eqt. III-9).

IV-3-1 Supervision dans le cas décentralisé :

Dans l'approche décentralisée, nous avons utilisé 3 RNA. Les structures de ces réseaux sont représentées sur la figure IV-5.

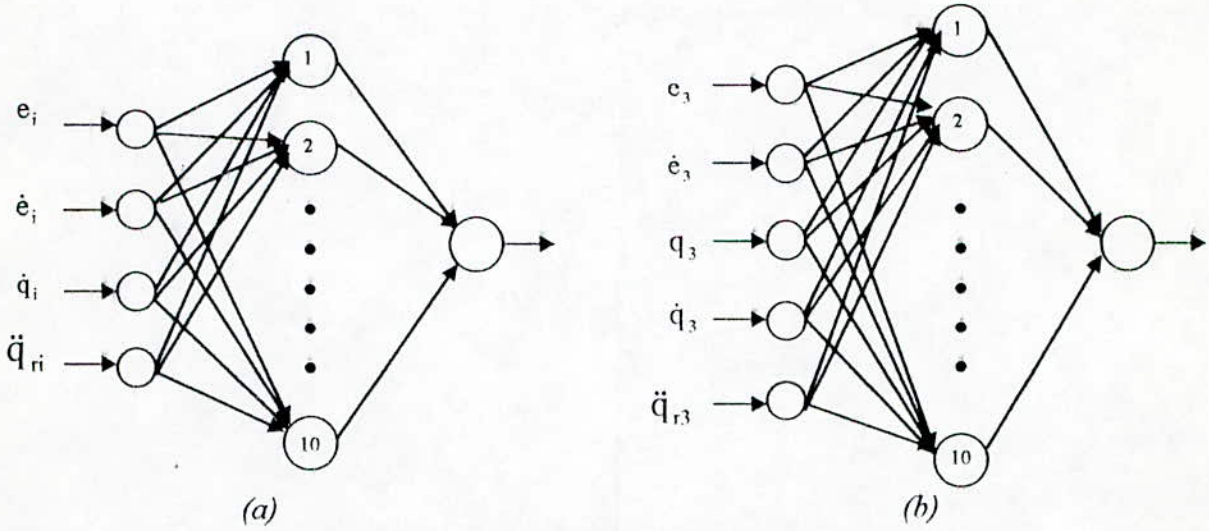


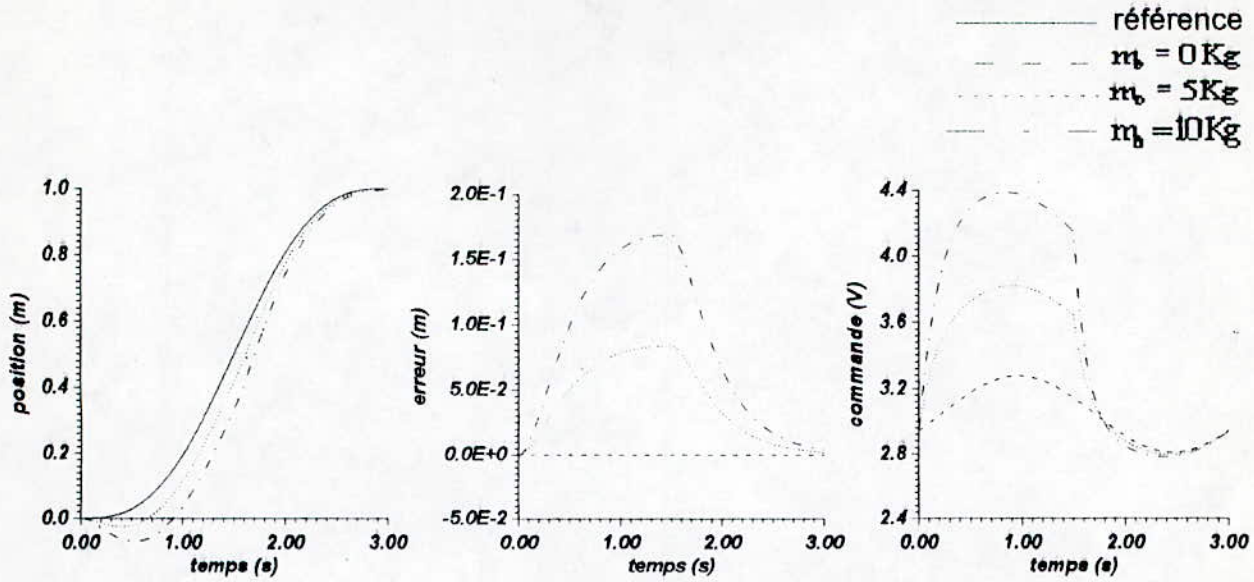
Figure IV-5 *Structures des réseaux utilisés*

(a) pour l'articulation 1 *et* 2 *(i=*1,2*) (b) pour l'articulation* 3

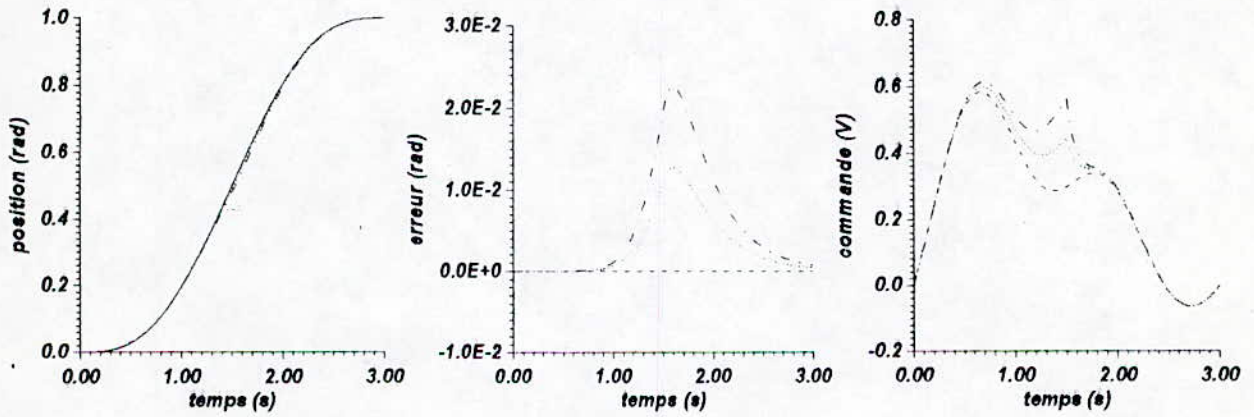
Le nombre d'entrées de chaque réseau est égal au nombre de variables locales apparaissant dans l'expression de la loi de commande linéarisante (eqt. (IV-2), (IV-3) et (IV-4)). L'apprentissage a été fait par la méthode de Backpropagation en utilisant 150 exemples. Les pas d'adaptation pour la première et la deuxième couche sont respectivement 0.2 et 0.1. Les poids ont été initialisés aléatoirement entre -0.5 et 0.5. Les gains K_i pour ajuster les sorties des RNA 1,2 et 3 (voir Sec. III-5-2-2) sont respectivement 4,1 et 1. Le nombre d'itérations effectuées lors de l'apprentissage est 2400,6000 et 600 pour les RNA 1,2 et 3 respectivement.

On présente sur la figure IV-6 les résultats obtenus par la commande linéarisante, puis sur la figure IV-7 les résultats obtenus en supervisant cette loi de commande. Dans les deux cas, la masse m_0 a été variée : 0,5 et 10 Kg, cette masse est lâchée à t=1.5 s.

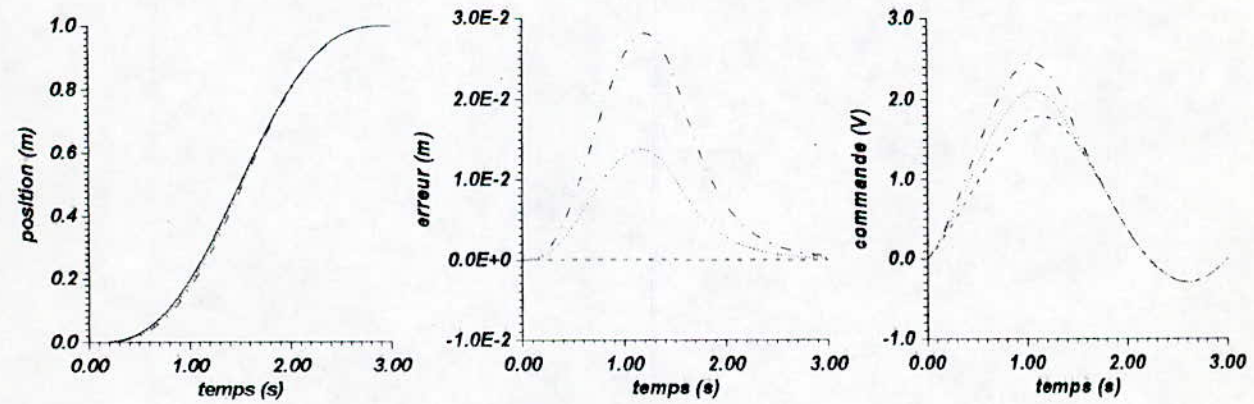
Sur la figure IV-8, on compare la robustesse des deux commandes pour m_0=10 Kg.



(a) Articulation 1

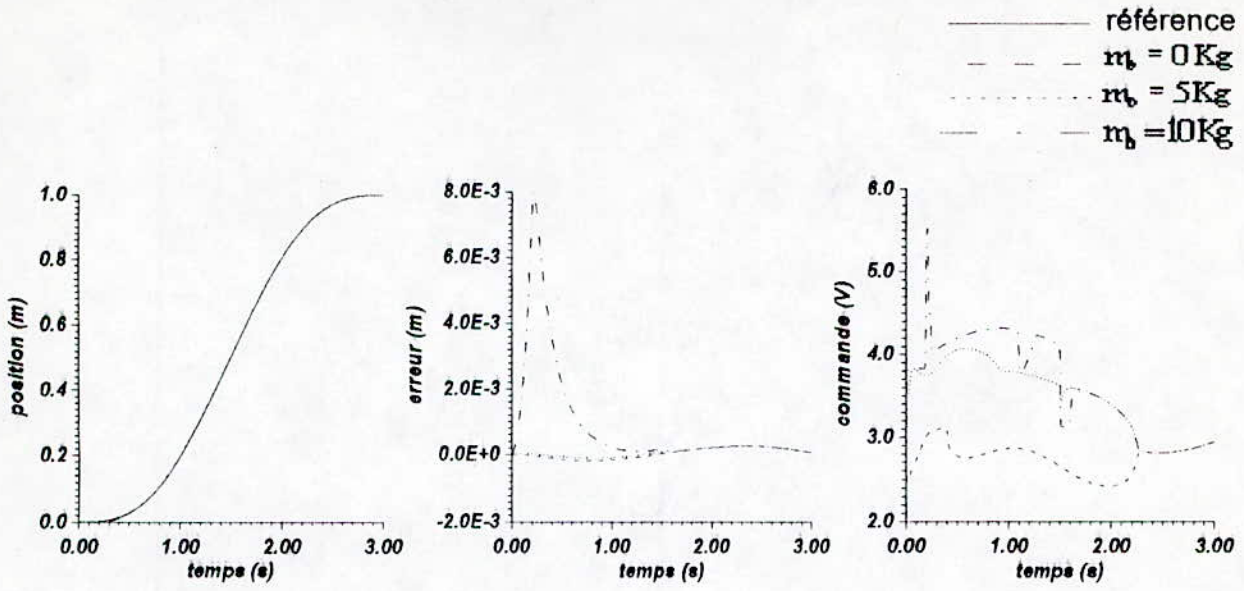


(b) Articulation 2

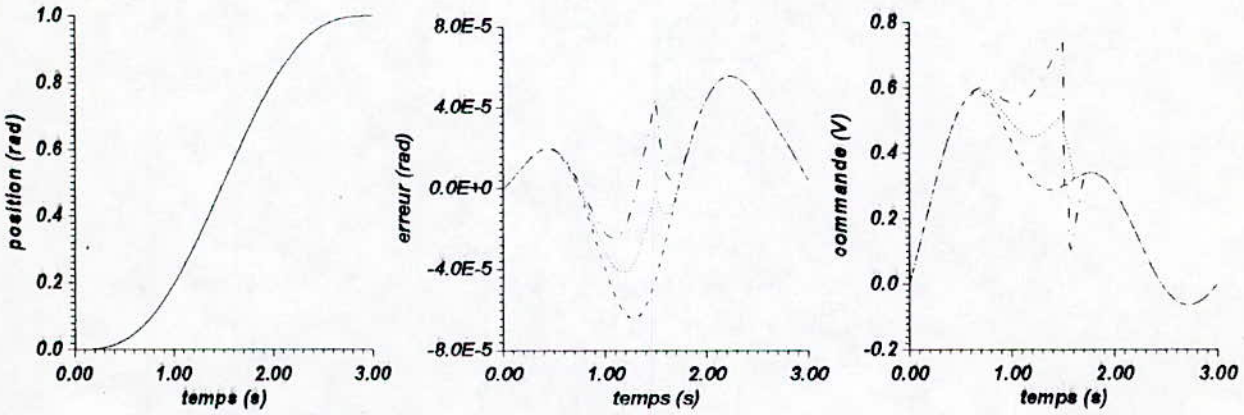


(c) Articulation 3

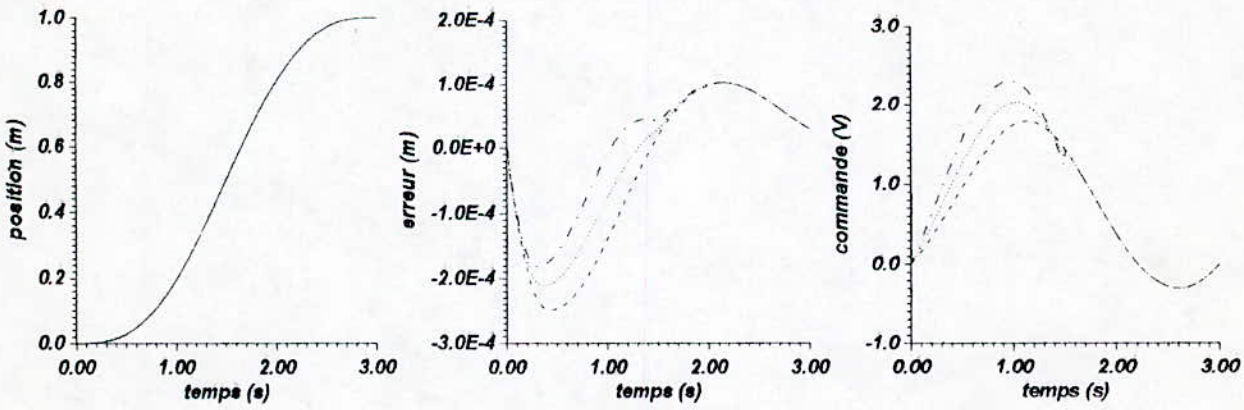
Figure IV-6 *Résultats de la commande linéarisante*



(a) Articulation 1

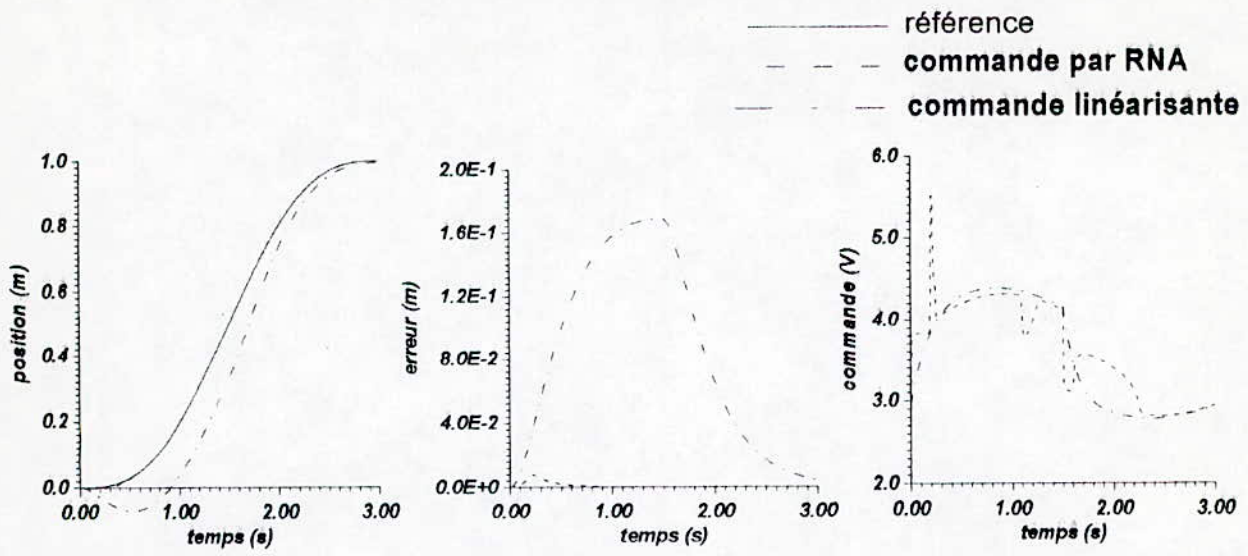


(b) Articulation 2

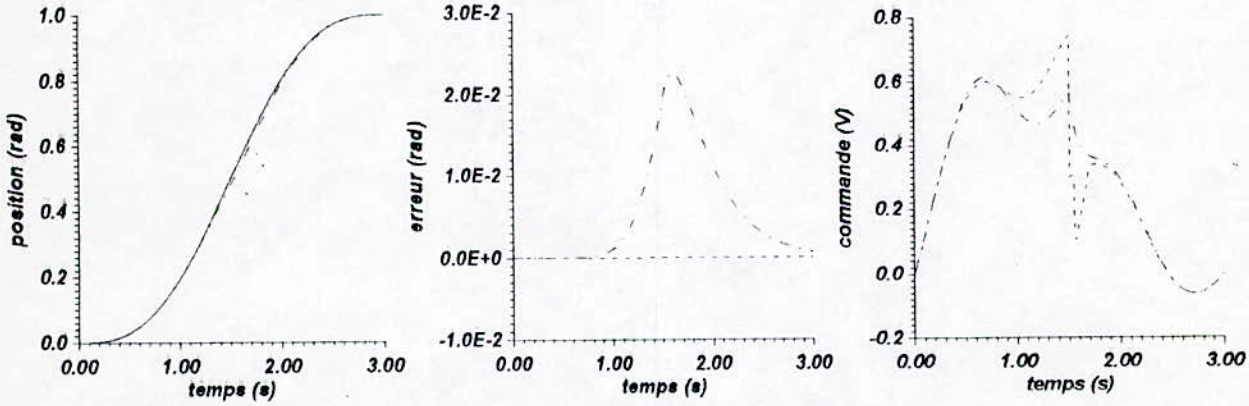


(c) Articulation 3

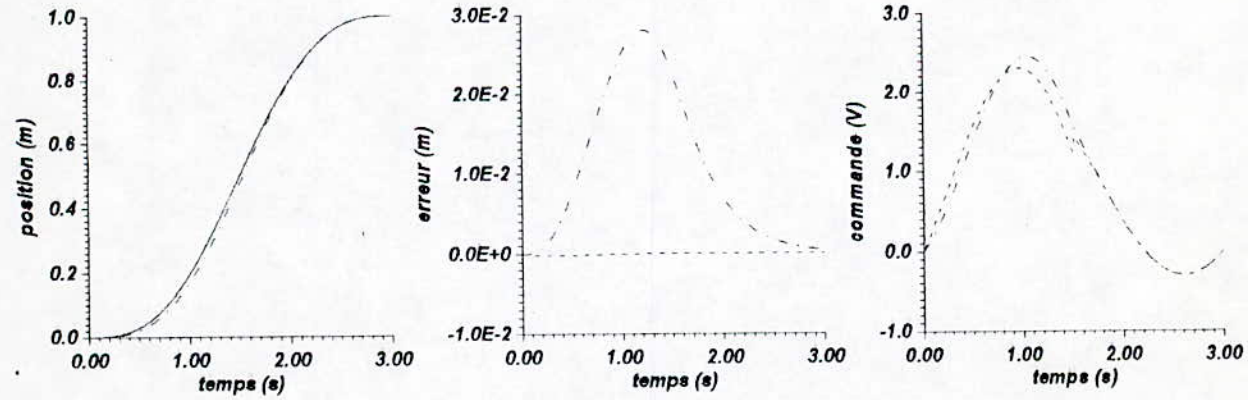
Figure IV-7 *Résultats de la commande décentralisée supervisée*



(a) Articulation 1



(b) Articulation 2



(c) Articulation 3

Figure IV-8 *Comparaison entre la commande linéarisante et la commande décentralisée supervisée pour* m_0=10 *Kg*

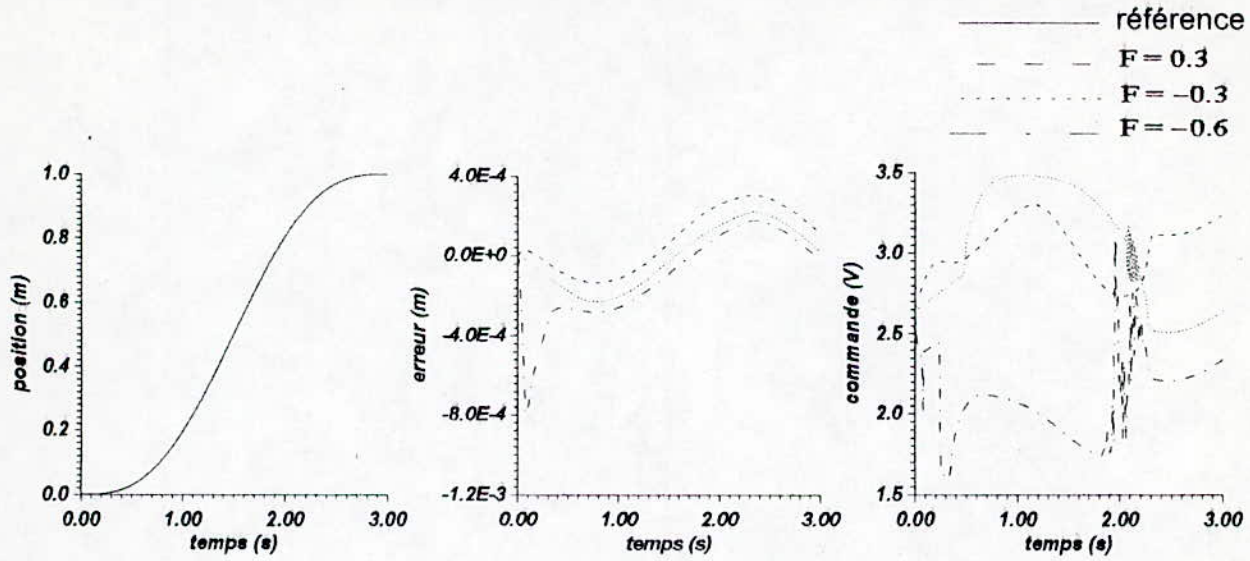
Commentaires :

On remarque que le réseau de neurone présente une meilleure robustesse grâce à son pouvoir de généralisation (robustesse naturelle par rapport aux paramètres non modélisés), ce qui n'est pas le cas pour la commande linéarisante. Les performances de cette dernière se dégradent dés que $\hat{M}$ et $\hat{h}$ n'estiment pas exactement M et h.

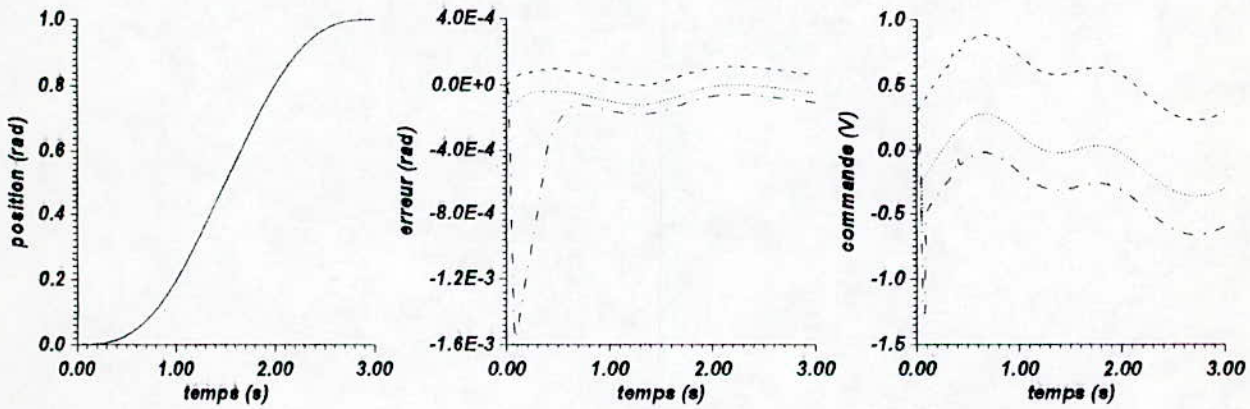
Autres tests de robustesse :

- **Variation du terme F :**

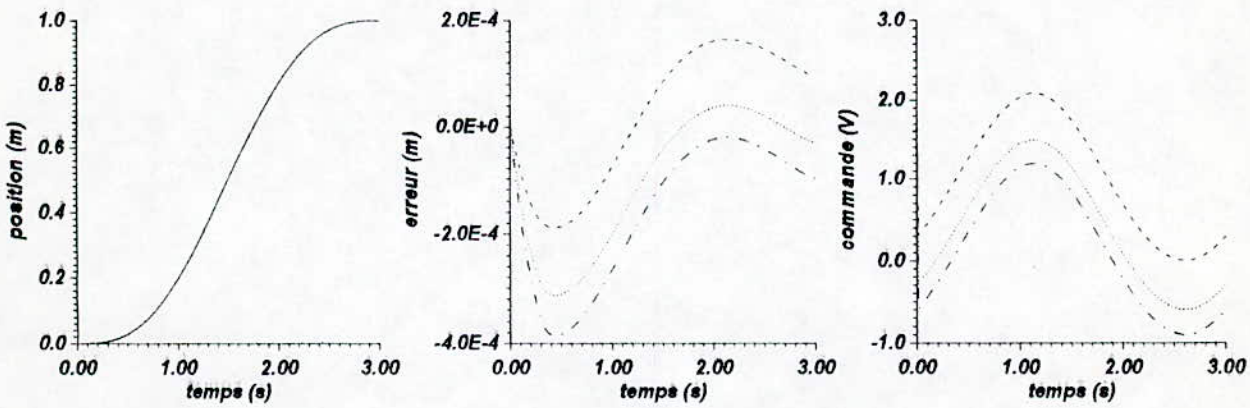
Au préalable et même pour générer les exemples d'apprentissage, nous avons supposé que le terme F (voir eqt. (III-9)) est nul. Maintenant, on introduit un terme F non nul, dont l'origine peut être une dynamique non modélisée ou une perturbation externe. Les performances de l'approche classique et de l'approche par réseaux de neurones sont comparées (Fig. IV-9, Fig. IV-10, Fig. IV-11).



(a) Articulation 1

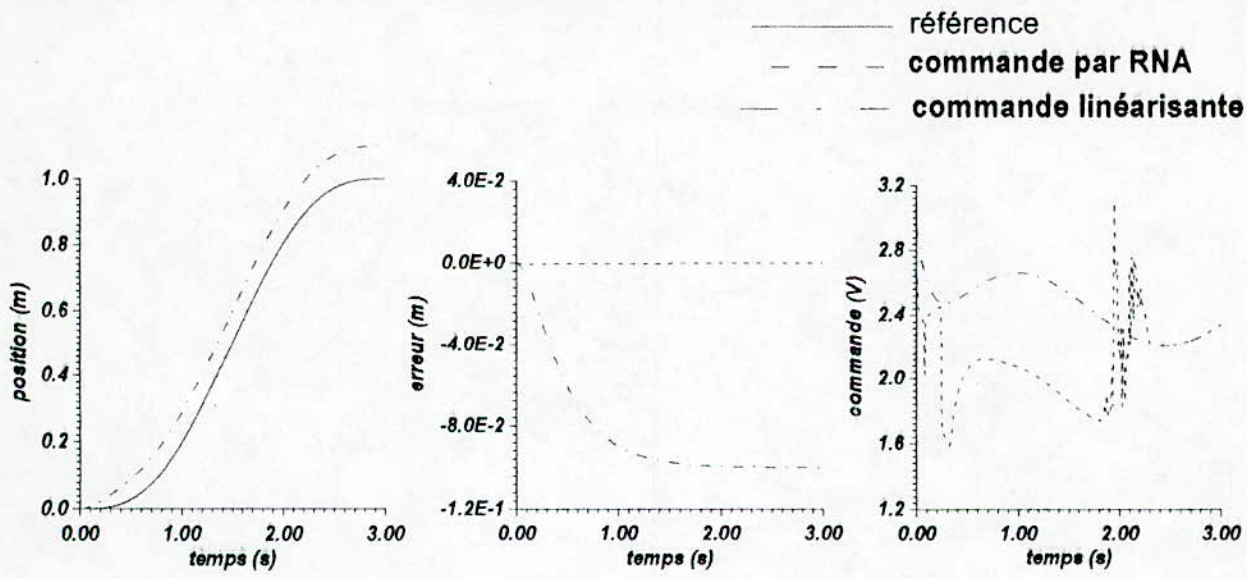


(b) Articulation 2

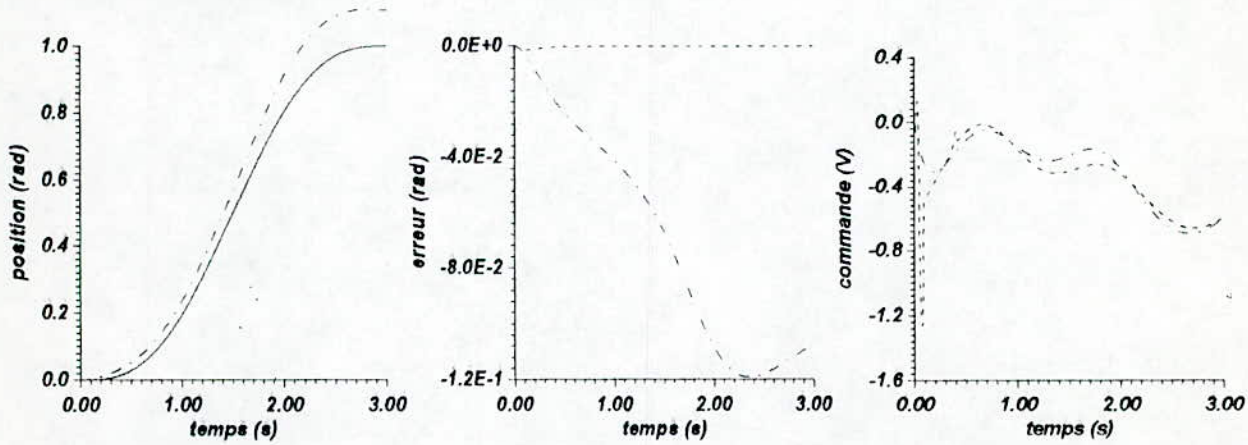


(c) Articulation 3

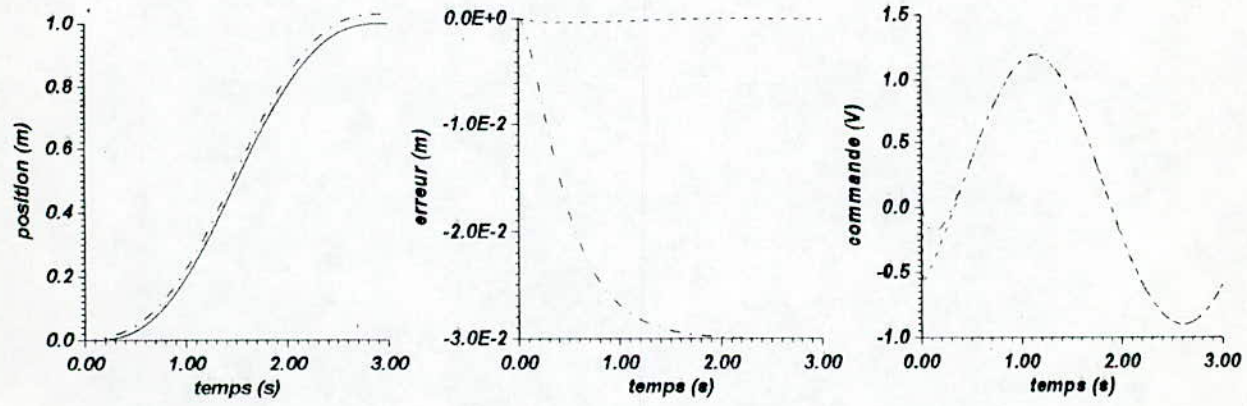
Figure IV-10 *Résultats de la commande décentralisée supervisée*



(a) Articulation 1



(b) Articulation 2



(c) Articulation 3

Figure IV-11 *Comparaison entre la commande linéarisante et la commande décentralisée supervisée pour* F=-0.06

Commentaires :

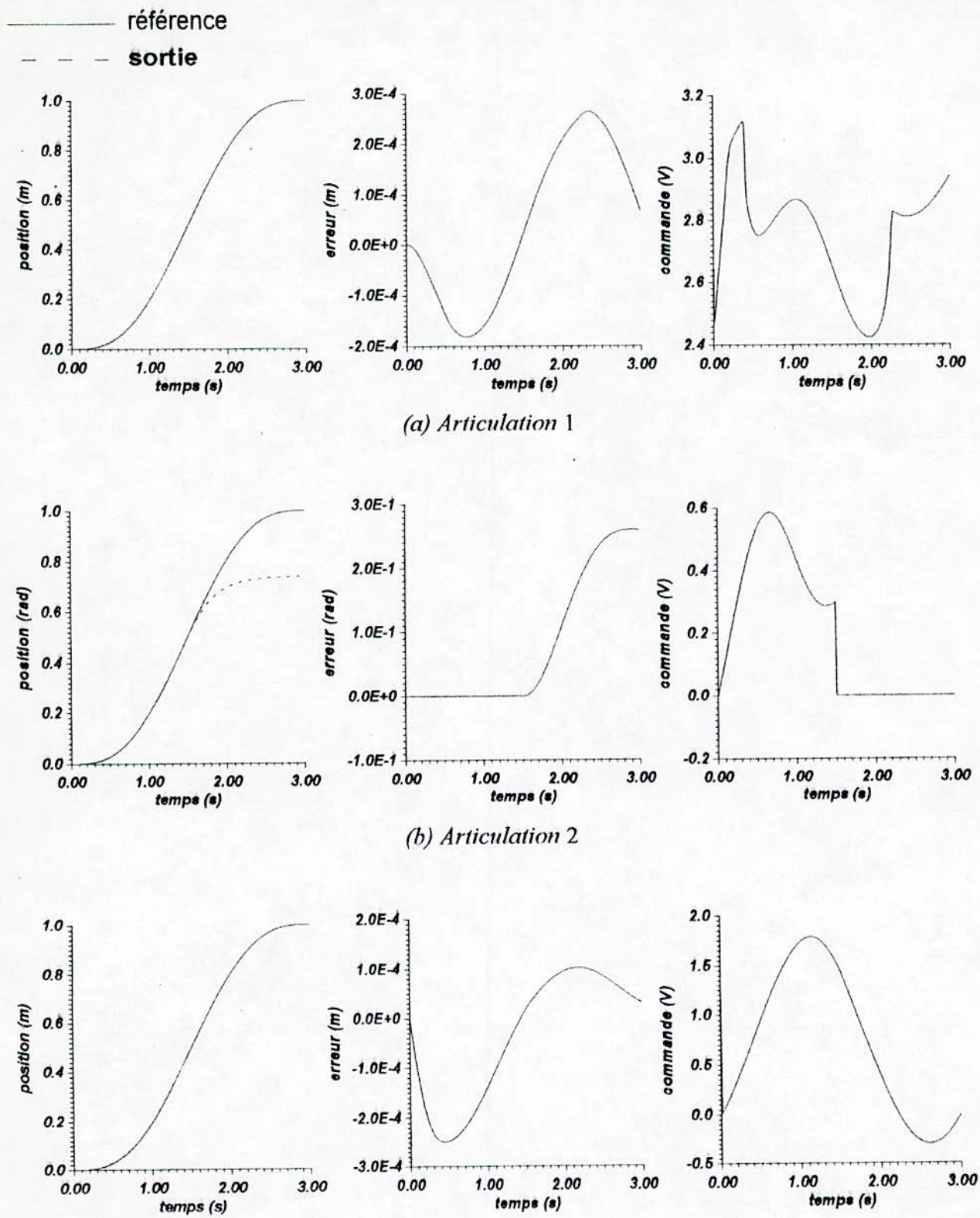
La robustesse du réseau de neurones par rapport à la commande linéarisante est nettement supérieure. Le réseau de neurones a pu compenser le terme F qui peut surgir, en pratique, à n'importe quel moment, alors que la commande linéarisante reste dans l'incapacité de compenser une telle perturbation.

- **Test de rupture de liaison :**

Ce test va mettre en évidence l'effective décentralisation de la commande que nous avons appliqué.

On va supposer qu'à un instant donné, lors de la commande du robot, l'une des deux articulations 2 ou 3, qui sont fortement couplées (voir eqt. (I-34)), ne reçoit pas d'informations (soit l'actionneur tombe en panne, donc la commande sera nulle, soit les capteurs tombent en panne, alors les mesures à fournir au réseau sont nulles) et on vérifie si cette défaillance locale influe sur l'autre articulation.

Sur la figure IV-12 et IV-13, à partir de l'instant 1.5 s , la commande et la mesure de l'articulation 2 sont annulées respectivement et le comportement de l'articulation 3, suite à ce défaut local, est vérifié. Le même test est effectué pour l'articulation 3 (Fig. IV-14 et IV-15).



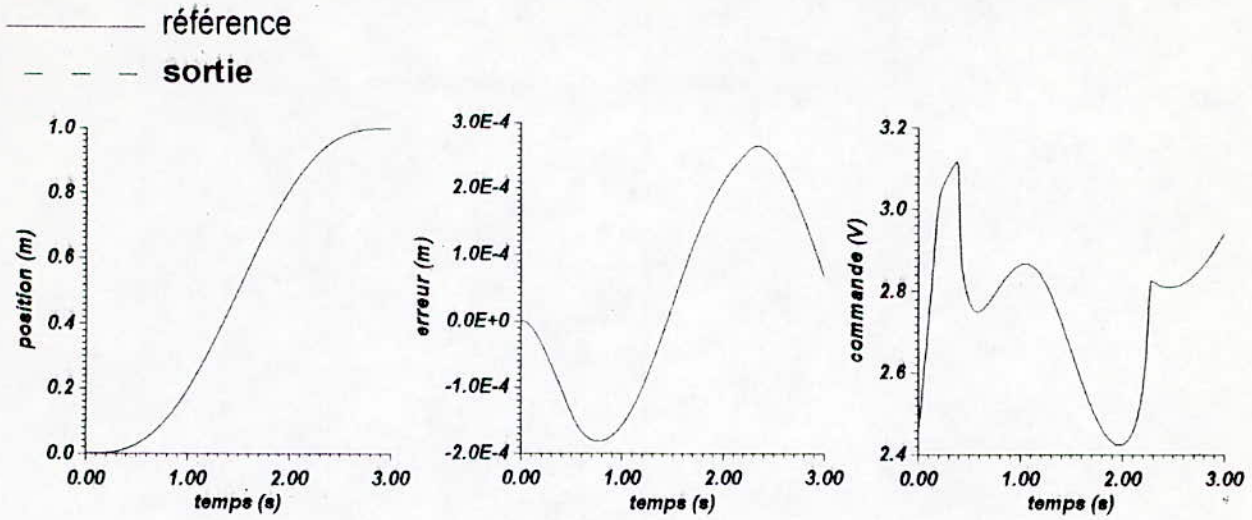
(a) Articulation 1

(b) Articulation 2

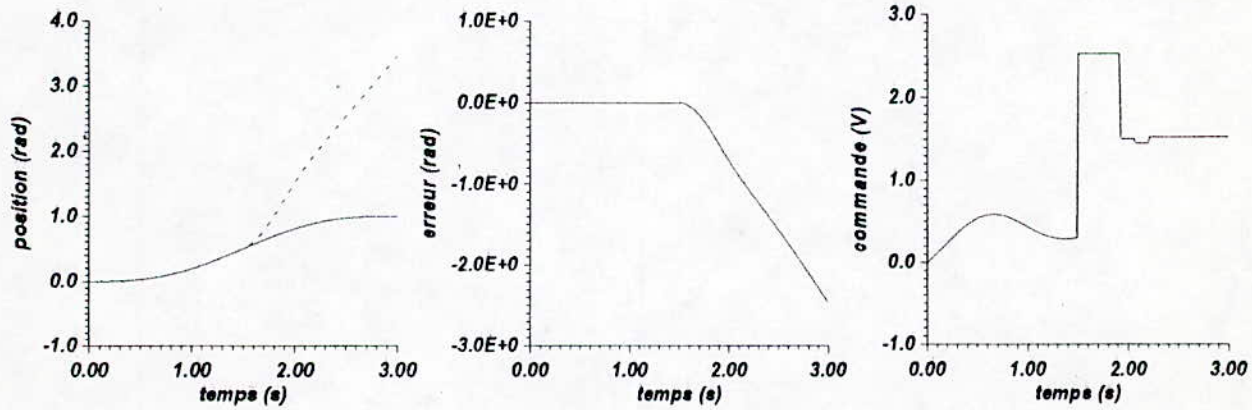
(c) articulation 3

Figure IV-12 *Test de rupture de liaison*

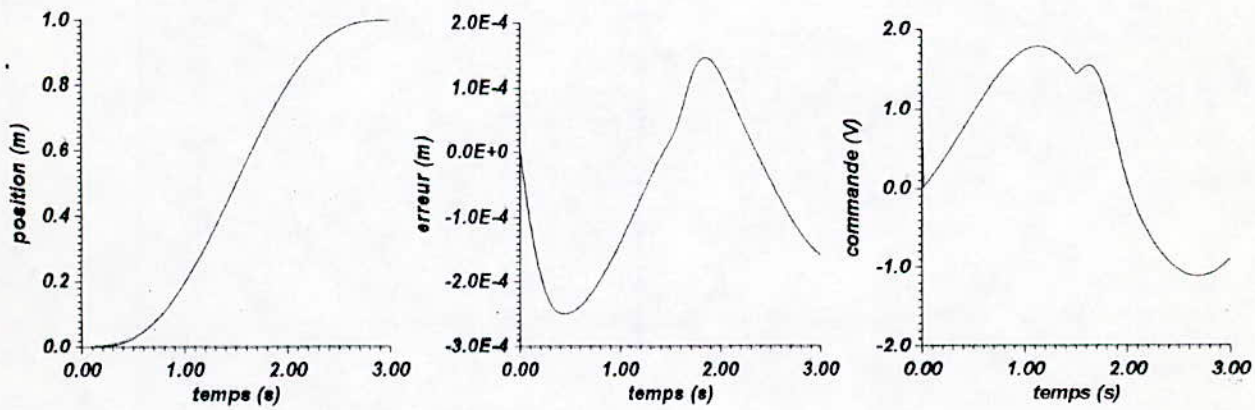
Annulation de la commande de l'articulation 2 *à partir de* t=1.5 *s*



(a) Articulation 1



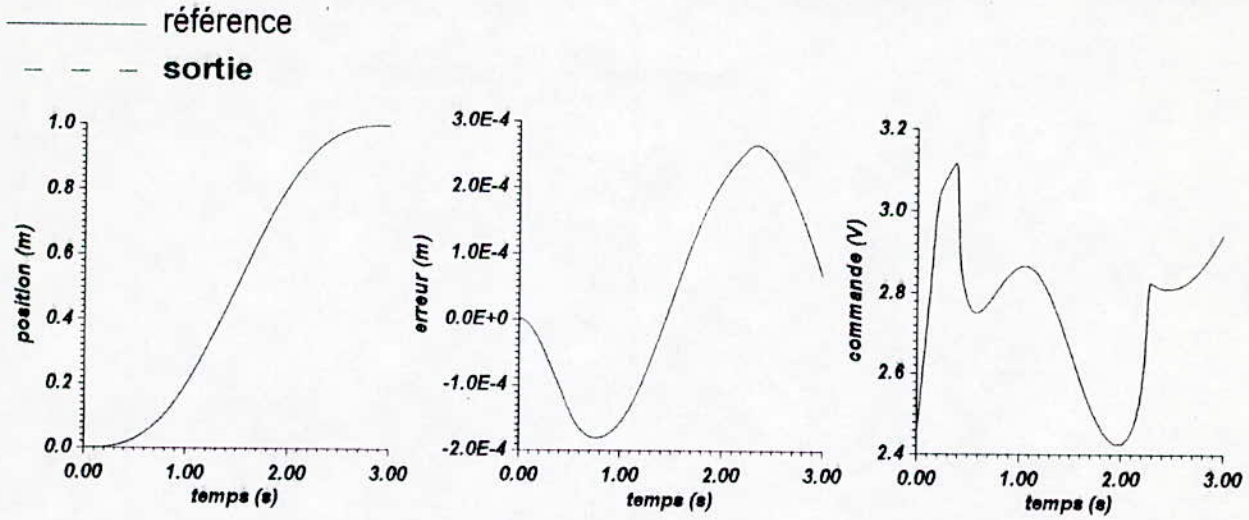
(b) Articulation 2



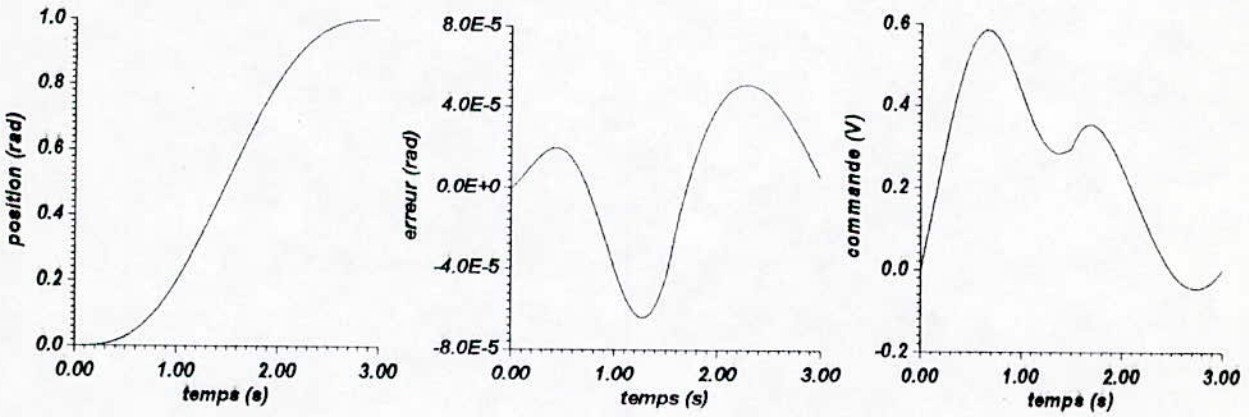
(c) Articulation 3

Figure IV-13 *Test de rupture de liaison*

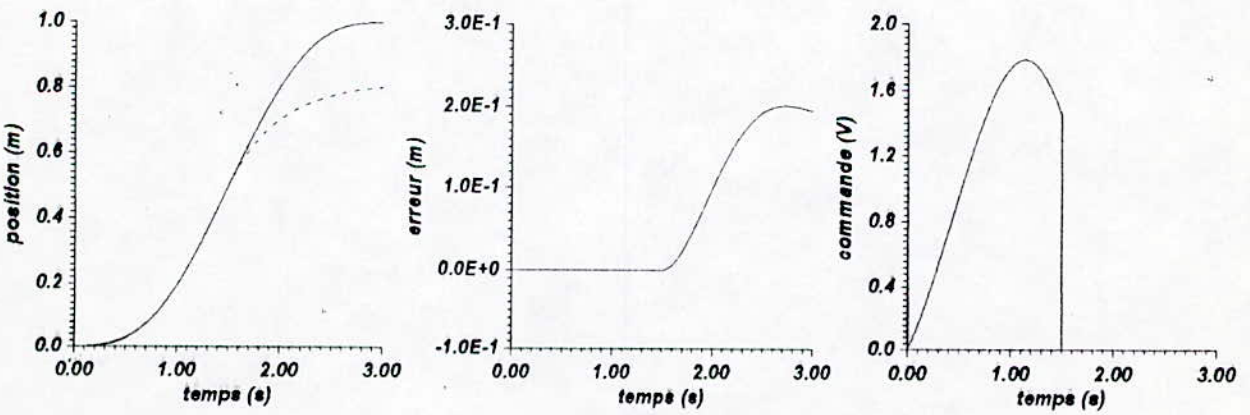
Annulation des mesures pour l'articulation 2 *à partir de t*=1.5 *s*



(a) Articulation 1



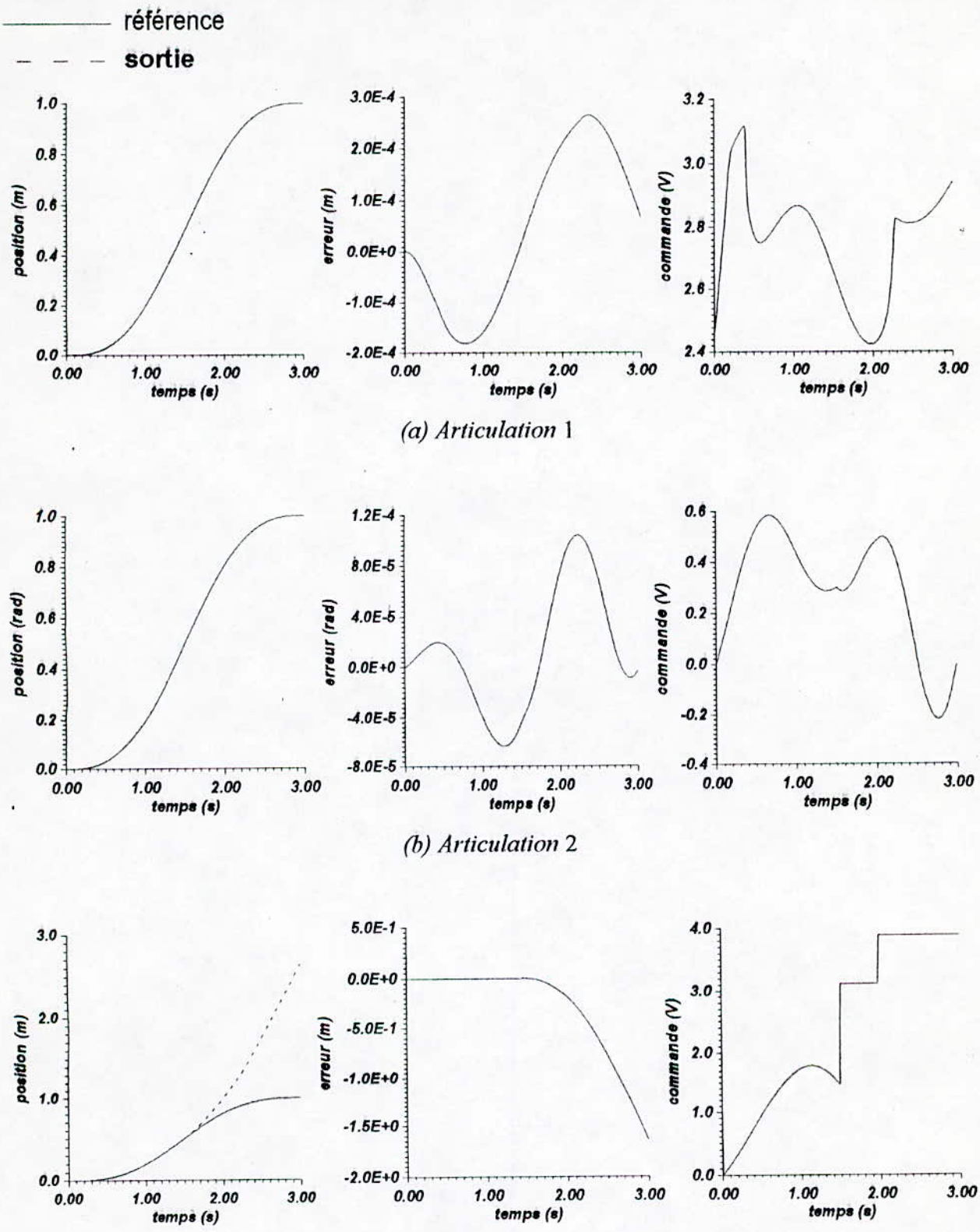
(b) Articulation 2



(c) Articulation 3

Figure IV-14 *Test de rupture de liaison*

Annulation de la commande de l'articulation 3 *à partir de* t=1.5 *s*



(a) Articulation 1

(b) Articulation 2

(c) Articulation 3

Figure IV-15 *Test de rupture de liaison*

Annulation des mesures pour l'articulation 3 *à partir de* t=1.5 *s*

Commentaires :

Suivant les résultats obtenus, on constate qu'une défaillance locale n'a aucune influence sur les autres articulations, ce qui prouve l'effective décentralisation. Ce test illustre une caractéristique intéressante de la commande décentralisée. En effet, l'utilisation de cette commande pour les systèmes complexes empêche la dégradation des performances du système global, lorsqu'un défaut local dans un sous-système surgit. Ainsi, le reste du système ne sera pas affécté.

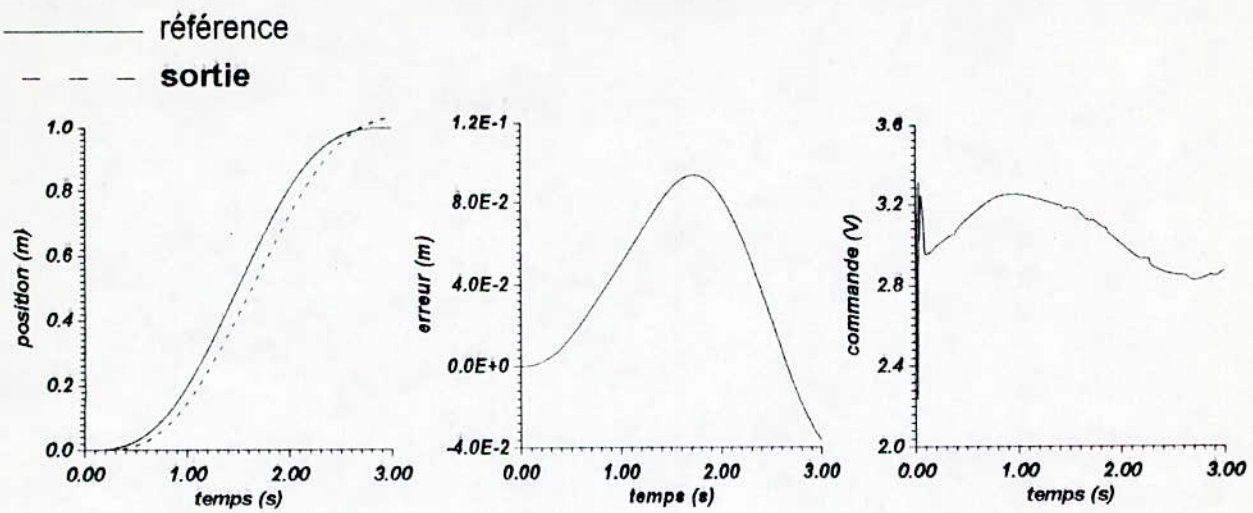
IV-3-2 Supervision dans le cas centralisé :

Dans ce cas, la supervision se fera par un seul RNA qui va commander les 3 articulations du robot en même temps. Ce réseau aura besoin de toutes les informations que nous avons fourni aux 3 réseaux utilisés dans le cas décentralisé. Le réseau aura, donc, 13 entrées $(e_1, \dot{e}_1, \dot{q}_1, \ddot{q}_{r1}, e_2, \dot{e}_2, \dot{q}_2, \ddot{q}_{r2}, e_3, \dot{e}_3, q_3, \dot{q}_3, \ddot{q}_{r3})$ et 3 sorties.

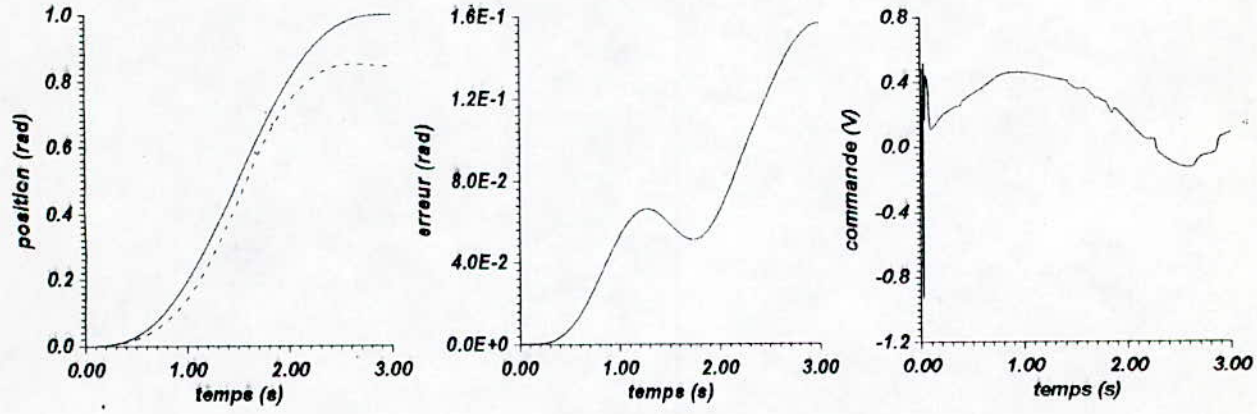
Nous avons d'abord utilisé un réseau à une seule couche cachée dont le nombre de neurones a été varié (10, 15 puis 20 neurones). Malgré que l'apprentissage a été effectué pendant plusieurs cycles d'adaptation, le réseau n'a pu donner des résultats satisfaisants lors de la commande. Alors nous avons été obligés d'augmenter le nombre de couches cachées à 2 couches de 10 neurones chacune. L'apprentissage a été fait par la méthode de Backpropagation sur 150 exemples. Les pas d'adaptation étaient 0.1, 0.08 et 0.06 pour la première, la deuxième et la troisième couche respectivement. Les poids ont été initialisés aléatoirement entre -0.5 et 0.5. les gains K_i pour ajuster les sorties du réseau sont égaux à 1.

Contrairement à l'approche décentralisée, l'approche centralisée nous a causé beaucoup de problèmes lors de l'apprentissage sans toutefois donner de bons résultats lors de la commande. En effet, à cause de la grande dimension du réseau, faire un cycle d'adaptation prend un temps relativement plus grand que celui d'un RNA dans le cas décentralisé. De plus, si dans le cas décentralisé l'apprentissage des 3 RNA (dont la dimension est acceptable) peut être implanté dans 3 calculateurs indépendants, l'apprentissage du réseau dans le cas centralisé (dont la dimension est grande) doit être nécessairement implanté dans un seul calculateur.

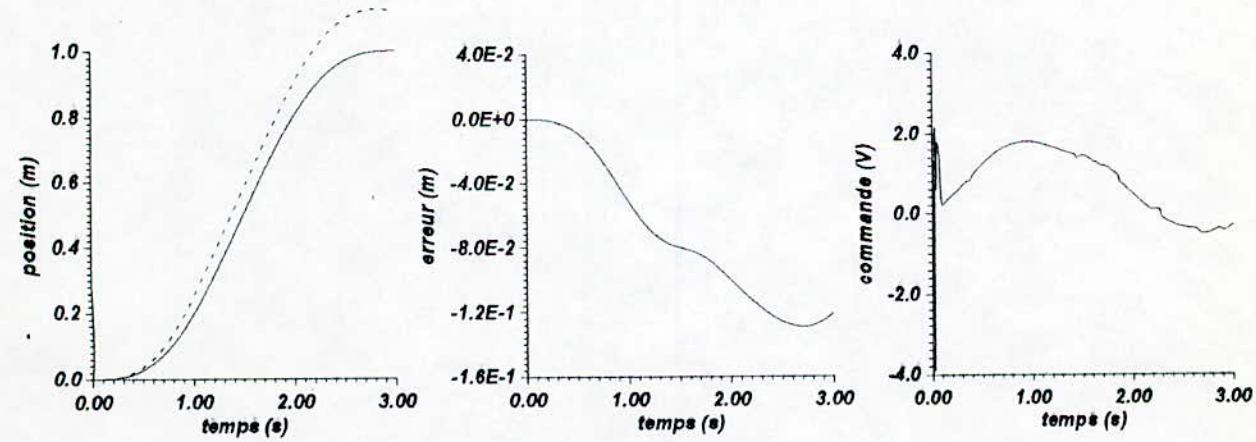
Sur la figure IV-17 on présente les résultats obtenus après 1500 itérations.



(a) Articulation 1



(b) Articulation 2



(c) Articulation 3

Figure IV-17 *Résultats de la commande centralisée supervisée*

Commentaires :

On remarque que le réseau centralisé n'a pu implanter la commande linéarisante. Les performances obtenues par la commande centralisée supervisée sont nettement inférieures à ceux de la commande décentralisée supervisée. On peut même dire que les rapports performances/facilité d'implantation des deux commandes sont nettement différents.

On envisage, pour améliorer les performances de la commande centralisée supervisée soit d'augmenter le nombre de couches cachées, soit d'augmenter le nombre de neurones dans ces couches.

IV-4 Conclusion :

Dans ce chapitre, des résultats de simulation de la commande décentralisée supervisée par réseaux de neurones ont été présentés. On a constaté que cette commande a donné de bons résultats. Le régulateur neuronal décentralisé a pu implanter le régulateur existant et même présentait une meilleure robustesse que ce dernier (cas de supervision de la commande linéarisante). Le régulateur neuronal peut ainsi remédier aux lacunes de certaines techniques de commande classiques. Cependant, le régulateur centralisé supervisé n'a pu implanter le régulateur existant, bien que la dimension du réseau de neurone utilisé a été augmentée et l'apprentissage a été fait pendant plusieurs cycles d'adaptation. Les résultats obtenus ont mi en évidence l'avantage de l'approche décentralisée par rapport à l'approche centralisée, aussi bien du point de vue performances que du point de vue implantation.

Conclusion Générale

Conclusion générale

Dans ce travail, la commande décentralisée supervisée par réseaux de neurones avec une application à un bras de robot manipulateur a été exposée. Dans un premier lieu, nous avons élaboré les différents modèles du robot manipulateur dans le but de simuler son comportement dynamique. Puis une étude générale sur les réseaux de neurones artificiels a été présentée, cela nous a permis de dériver les algorithmes d'apprentissage. Le principe de la commande décentralisée supervisée par réseaux de neurones a été étalé le long du troisième chapitre ainsi que deux lois de commande à superviser, à savoir la commande adaptative décentralisée et la technique du couple calculé.

L'application de la commande décentralisée supervisée à un bras de robot manipulateur s'est avérée très intéressante. En effet, le réseau de neurones a montré d'excellentes capacités à apprendre à imiter un régulateur existant, même si ce dernier est adaptatif. De plus, grâce à son pouvoir de généralisation, le réseau de neurones présentait un caractère robuste et pouvait compenser des perturbations qui n'ont pas été prises en considération lors de l'apprentissage et dont un régulateur classique n'a pu compenser.

Un test de rupture de liaison a mis en évidence l'effective décentralisation de la commande appliquée. En effet, une défaillance locale du système de commande d'une articulation n'a pratiquement aucune influence sur les autres articulations.

Un autre test de comparaison entre la commande supervisée centralisée et la commande supervisée décentralisée a montré l'avantage de cette dernière. L'approche décentralisée présente de meilleures performances et une simplicité du régulateur, ce qui tout à fait le contraire dans l'approche centralisée.

Enfin, comme perspectives de ce travail, on envisage :

- L'utilisation d'autres types de réseaux, comme les réseaux dynamiques, pour la supervision de régulateurs existants.

• Pour résoudre les problèmes de la commande centralisée supervisée, on doit augmenter d'avantage la dimension du réseau de neurones utilisé (par exemple utiliser un réseau à 3 couches cachées) comme on peut changer l'algorithme d'apprentissage.

• L'application de la commande décentralisée supervisée à d'autres types de robots (puma, scara,...).

• La supervision d'autres régulateurs existants qui présentent certains inconvénients (manque de robustesse,...)

Références Bibliographiques

Références Bibliographiques

[1] C.Vibet, *Robots : principes et contrôle*, Ellipse, Paris, 1987.

[2] B.Gorla et M.Renaud, *Modèles des robots manipulateurs application à leur commande*, Cepadues, 1984.

[3] R.P.Paul, *Robot Manipulators : Mathematics, Programming and Control*, MIT Press, Cambridge, MA, 1981.

[4] J.P.Lallemand et S.Zaghloul, *Robotique : Aspects fondamentaux, Modélisation mécanique, CAO robotique, Commande*, Masson, 1994.

[5] D.P.Stoten, "Generalized manipulator dynamics, with regard to model référence adaptive control," *Int. J. Control*, Vol.50, No.6, pp.2249-2268, 1989.

[6] Y.Koren, *La robotique pour l'ingénieur*, McGraw Hill, 1986.

[7] L.Guenfaf, *Etude de différentes stratégies de commande adaptative : application à un bras manipulateur*, Thèse de Magister, ENP, 1995.

[8] J.A.Freeman and D.M.Skapura, *Neural Networks : Algorithms, Applications and Programming techniques*, Addison-Wesley, Massachusetts, 1991.

[9] N.B.Karayiannis and A.N.Venetsanopoulos, *Artificial Neural Networks : Learning Algorithms, Performance Evaluation and Applications*, Kluwer Accademic Publishers, London, 1993.

[10] K .J.Hunt, D.Sbarbaro, R.Zbikowski and P.J.Gowthrop, "Neural Networks for control Systems - ASurvey," *Automatica*, Vol.28, No.6, pp.1083-1112, 1992.

[11] T.Fukuda and T.Shibata, "Theory and Applications of Neural Networks for Industrial Control Systems," *IEEE Trans. on Industrial Electronics*, Vol.39, No.6, pp.472-489, December 1992.

[12] M.Weinfeld, "Réseaux de Neurones," *Techniques de l'ingénieur*, H1990, 1995.

[13] B.Hamzi, S.Labiod, *Identification et commande des systèmes dynamiques par réseaux de neurones*, Projet de Fin d'Etude, ENP, Juillet 1995.

[14] W.T.Miller, R.S.Sutton and P.J.Werbos, *Neural Networks for Control*, MIT Press, Cambridge, MA, 1990.

[15] A.Guez and J.Selinsky, "A Trainable Neuromorphic Controller," *Journal of Robotic Systems*, Vol.5, pp.363-388, 1988.

[16] H.Seraji, "Decentralized Adaptive Control of Manipulators : Theory, Simulation and Experimentation," *IEEE Trans. on Robotics and Automation*, Vol.50, No.2, pp.183-201, April 1989.

[17] T.Ozaki, T.Suzuki, T.Furuhashi, S.Okuma and Y.Uchikawa, "Trajectory Control of Robotics Manipulators Using Neural Networks," *IEEE Trans. on Industrial Electronics*, Vol.30, No.3, pp.195-202, June 1991.

[18] J.E.Slotine and W.Li, *Applied Nonlinear Control*, Prentice-Hall, 1991.

[19] J-M.Randers, *Algorithmes Génétiques et Réseaux de Neurones*, Hermes, 1995.

[20] A.Titli et al., *Analyse et Commande des Systèmes Complexes*, Cepadues, 1979.

[21] E.J.Davison and Ü.Özgüner, "Characterization of Decentralized Fixed Modes for Interconnected Systems," *Automatica*, Vol.19, No.2, pp.169-182, 1983.

ملخص:

هذا العمل يستعرض مبدأ التحكم اللامركزي المراقب بواسطة الشبكات العصبية الإصطناعية. في هذا الإطار، قمنا بتعليم قانوني تحكم للشبكات العصبية الإصطناعية (تحكم لامركزي تلاؤمي، تحكم العزم المحسوب). نتائج تطبيق هذه التقنية في التحكم على ذراع آلي قد عرضت و كذا اختبارات الضلاعة على المنظم العصبي.

كلمات مفتاحية : تحكم لامركزي، شبكات عصبية إصطناعية، تحكم مراقب، تحكم العزم المحسوب.

Abstract :

This work exposes the principle of the decentralized supervised control using neural networks. In this context, two control laws are supervised (decentralized adaptive control, computed torque control). Results of application of this control technique to a robot manipulator are presented as well as robustness tests of the neural controller.

Key words : Decentralized Control, Artificial Neural Networks, Supervised Control, Computed Torque Control.

Résumé :

Ce travail expose le principe de la commande décentralisée supervisée par réseaux de neurones artificiels. Dans ce cadre, deux lois de commande sont supervisées (commande décentralisée adaptative, commande par couple calculé). Les résultats de l'application de cette technique de commande à un bras de robot manipulateur sont présentés ainsi que des tests de robustesse du contrôleur neuronal.

Mots clés : Commande Décentralisée, Réseaux de Neurones Artificiels, Commande Supervisée, Commande par Couple Calculé.