N° d'ordre : 03 / 2010-D / EL

الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIEN

Faculté D'Electronique et D'Informatique

جامعة هواري بومدين للعلوم والتكنولوجيا

U S T H B

Thèse

Présentée pour l'obtention du grade de

Docteur

Spécialité : Contrôle de processus et Robotique

Par

MAOUCHE Amin Riad

Thème

VERS UNE COMMANDE NEURONALE DE ROBOTS FLEXIBLES

Soutenue publiquement le 11 Novembre 2010, devant le jury composé de :

Mme	Fatima Boumghar	Professeur	à l'*USTHB*	Présidente
Mr	Mokhtar Attari	Professeur	à l'*USTHB*	Dir. de thèse
Mr	Mustapha Hamerlain	Directeur de Recherche	au *CDTA*	Examinateur
Mr	Farès Boudjema	Professeur	à l'*ENP*	Examinateur
Mr	Djamel Boukhetala	Professeur	à l'*ENP*	Examinateur
Mme	Noura Achour	Maitre de conférences A	à l'*USTHB*	Examinatrice

بسم الله الرحمان الرحيم

e-mail: armaouche@umbb.dz

web site: www.lins.usthb.dz

Dédicaces

A la mémoire de ma mère et à mon père,

A ma femme et à mes enfants

A mon frère, à sa femme et à ses enfants

A ma sœur et à son mari

A toute ma grande famille et ma belle famille

A mon promoteur et à toute l'équipe du Laboratoire LINS

A mes amis et collègues enseignants

Remerciements

الحمد لله رب العالمين

Cette thèse a été réalisée au sein du Laboratoire d'Instrumentation (LINS) de la faculté d'Électronique et Informatique (FEI) de l'Université des Sciences et de la Technologie Houari Boumediene (USTHB).

Je tiens tout particulièrement à exprimer ma profonde reconnaissance à Monsieur **Mokhtar ATTARI**, Professeur à la Faculté d'Electronique et d'Informatique de l'USTHB et directeur de la présente thèse pour m'avoir orienté au cours de mon travail, pour m'avoir fait profité de ses connaissances, pour les nombreuses discussions scientifiques et techniques et pour sa rigueur et son exigence qui ont amplement influencés la qualité de ce travail.

Je remercie les membres du jury qui m'ont fait l'honneur d'accepter d'examiner ce travail:

- Madame **Fatima BOUMGHAR**, Professeur à la Faculté d'Electronique et d'Informatique de l'USTHB pour l'honneur qu'elle me fait en présidant ce jury.
- Monsieur **Mustapha HAMERLAIN**, Directeur de recherches au CDTA, Alger, pour avoir mobilisé ses compétences et son temps pour juger ce travail.
- Monsieur **Farès BOUDJEMA,** Professeur à l'ENP, Alger, pour me faire l'honneur une nouvelle fois, après mon travail de Magister, d'examiner mon travail de Doctorat.
- Monsieur **Djamel BOUKHETALA**, Professeur à l'ENP, Alger, pour avoir accepté de faire parti de mon jury de thèse et pour l'attention accordée.
- Madame **Noura ACHOUR**, Maitre de Conférences à la Faculté d'Electronique et d'Informatique de l'USTHB pour l'honneur qu'elle me fait en acceptant de faire parti du jury, une nouvelle fois, après mon Magister, et pour l'intérêt qu'elle porte à mon travail.

Je ne peux pas oublier les membres de ma famille, qui m'ont soutenu durant cette thèse, ainsi que mes amis pour leur aide et leur encouragement. Je tiens aussi à exprimer toute mon amitié à mes collègues ainsi qu'à tous les membres du LINS. Merci, enfin, à toute personne qui a contribué d'une manière ou d'une autre à l'aboutissement de ce travail.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Depuis la révolution industrielle, une discipline a marqué l'évolution du monde technologique: la Robotique. L'avènement des robots dans l'industrie a permis de soulager l'homme de travaux répétitifs et très contraignants tels que: le déplacement d'objets lourds, les tâches d'assemblages, les microsoudures etc. Ceci avec plus d'efficacité et de précision.

Les robots sont aussi utilisés dans des applications en milieu hostile, telles que: le positionnement de satellites dans l'espace, la manipulation de substances dangereuses (substances explosives, radioactives ou toxiques) ou bien encore dans des travaux sous marins.

La compétition incessante dans l'industrie conduit à une nécessaire augmentation de la productivité en préservant la qualité et en diminuant le coût de revient des produits. Cependant, les robots manipulateurs existants souffrent encore de faiblesses qui les empêchent de mener à bien certaines tâches et limitent leurs champs d'action.

Ces robots sont en effet rigides, lourds et encombrants ce qui se traduit par une grande énergie consommée et une vitesse d'exécution lente. La consommation peut constituer un point crucial quand l'énergie est limitée, comme c'est le cas dans les applications spatiales. Des recherches ont donc été entamées afin d'alléger la structure des robots et augmenter ainsi leurs performances.

Toutefois, l'augmentation du rapport 'charge transportée / masse propre du robot', ainsi que l'exécution de mouvements à forte accélération engendrent une déformation de la structure du robot et l'apparition d'oscillations 'basses fréquences' dues à la flexibilité des bras.

Ces phénomènes physiques doivent être pris en compte lors de la modélisation et de l'élaboration de la loi de commande pour préserver la qualité et la précision des tâches auxquelles est destiné le robot.

Le travail que nous présentons porte sur le contrôle de processus dynamiques complexes et plus spécifiquement sur la commande de robots manipulateurs à bras flexibles.

La tâche de commande d'un robot flexible consiste en deux sous tâches. Premièrement, le suivi de la trajectoire de consigne pour les variables rigides qui sont les positions et les vitesses articulaires. Deuxièmement, l'amortissement des variables élastiques qui sont, dans notre cas, les flèches et les rotations élastiques de section au bout de chaque bras.

Les principales difficultés pour la commande des robots flexibles sont les suivantes.

A la différence d'un robot rigide, le robot flexible est un système sous actionné car il possède plus de sorties à contrôler (variables rigides et élastiques) que d'entrées de commande (couples articulaires appliqués). Cela se traduit par la présence d'équations de couplage dynamique entre les variables rigides et élastiques dont il faudra tenir compte.

Un robot flexible est un système à paramètres répartis et régi par des équations complexes aux dérivées partielles. L'utilisation d'un tel modèle directement dans la commande augmente la complexité de celle-ci et par la même son temps de calcul et ainsi son temps de réponse.

Les erreurs de modélisation, les incertitudes sur l'estimation des paramètres physiques ainsi que les différentes perturbations externes influent beaucoup sur la qualité du contrôle.

Les robots flexibles sont naturellement plus sensibles aux variations de la charge transportée, ce qui nécessite l'utilisation de commandes particulièrement performantes et robustes afin de compenser les écarts de trajectoires induits par cette variation.

Pour faire face à ces difficultés, nous avons élaboré une commande hybride originale qui trouve son fondement dans la technique de modélisation de type 'boite grise'.

Le modèle semi-physique ou modèle type 'boite grise', peut être perçu comme un compromis entre le modèle à base de connaissances et le modèle statistique. Il peut englober la somme des connaissances que l'on possède sur le processus (lois physiques, chimiques, etc.) en plus de fonctions paramétrées, dont les paramètres sont déterminés par la mesure.

Une commande, basée sur une modélisation de type 'boite grise' est très utile lorsqu'un modèle à base de connaissances existe mais est incomplet et ne peut être amélioré avec une analyse plus approfondi, ou peut l'être mais au détriment d'un temps de calcul et d'une complexité tels qui le rendent inutilisable dans les problèmes de contrôle en temps réel.

La loi de commande que nous proposons est une combinaison de deux contrôleurs. Le premier contrôleur est basé sur une approximation des fonctions non linéaires, du modèle dynamique utilisé dans la commande, par un réseau de neurones artificiels. Son but est d'assurer un contrôle stable des variables rigides tout en amortissant les vibrations des variables élastiques.

L'utilisation des réseaux de neurones, en lieu et place des fonctions non linéaires complexes dans la commande, réduira son temps de calcul et augmentera de ce fait sa réactivité et donc ses performances dans la pratique.

Le deuxième contrôleur est constitué d'un réseau de neurones adaptatif qui fonctionne 'en ligne'. L'utilisation d'une commande adaptative permet de prendre en charge les erreurs dues aux incertitudes structurées et non structurées et rend possible la compensation, jusqu'à un certain point, de phénomènes physiques, tel que les frottements, dont la représentation mathématique est difficile à réaliser.

Cette thèse est divisée en deux grandes parties. La première partie concerne la modélisation et regroupe le chapitre I et le Chapitre II. La deuxième partie concerne la commande et regroupe le chapitre III et le Chapitre IV.

Au Chapitre I, un bref exposé est présenté sur la méthode d'obtention du modèle dynamique d'un robot à bras flexibles dans le cas général. Nous détaillons par la suite le calcul concernant le cas d'un robot flexible plan horizontal, à un et deux axes. Nous considérerons, par la suite, ce dernier comme prototype dans la simulation des lois de commande élaborées.

Le Chapitre II est consacré à la modélisation par réseaux de neurones artificiels (RNA). Nous présentons, dans un premier temps, un rappel historique sur la naissance et l'évolution des RNA. Nous donnons ensuite plus de détaille sur le type de réseau utilisé dans la commande ainsi que la méthode adoptée pour l'apprentissage.

Au chapitre III, une brève revue bibliographique sur la commande de robots flexibles est présentée. Nous exposons par la suite, quelques commandes classiques à base de modèle de connaissances ainsi que quelques propriétés dynamiques fondamentales utilisée dans la commande.

Le Chapitres IV, débute par la présentation d'une commande non linéaire classique à base de modèle de connaissances. Une analyse critique après simulation, nous montre les limites de cette commande et la nécessité de la modifier et de la compléter pour obtenir une commande plus efficace. Nous introduisons alors, les RNA dans la loi de commande pour former un nouveau type de contrôleur hybride. En fin nous montrons par une série de tests en simulation l'efficacité de la commande proposée et sa robustesse face à une incertitude importante sur l'estimation des paramètres dynamique du robot.

Finalement, nous résumons l'essentiel des résultats obtenus ainsi que les principales perspectives de ce travail, dans la conclusion générale.

Chapitre I

Modélisation dynamique du robot flexible

I.1 Introduction

Les avantages indéniables des robots flexibles en terme de rapidité, de dextérité, de légèreté et de coût de revient poussent les industriels à s'équiper de plus en plus de ce type de robots. Néanmoins, l'utilisation des robots flexibles pose de nouveaux problèmes par rapport aux robots rigides. En effet, l'allégement des bras du robot associé à l'augmentation de la vitesse d'exécution des tâches engendrent des oscillations qui peuvent dégrader les performances du robot en termes de temps de réponse et de précision dans l'exécution des tâches.

Une étude dynamique incluant le caractère flexible du robot dans le modèle doit donc être effectué afin de prendre en compte ces oscillations et pouvoir les atténuer par la suite grâce à une commande adéquate.

De nombreuses recherches ont été entamées afin d'établir le modèle dynamique des robots flexibles. D'un point de vue méthodologique, différentes approches ont été utilisées et l'on peut citer entre autres : la méthode des masses concentrées [Ren 1980, Pot 1983], la méthode des éléments finis [Sun 1981, Uso 1986, Bay 1987, Son 1988, Jon 1990,Ser 1994], l'approche modale [Can 1984, Boo 1987, Bar 1988, Oak 1989a, Oak 1989b, Del 1990, Del 1991, Ahm 2008], la méthode de perturbation singulière [Sic 1988], la méthode des Bonds-Graphs [Mar 1977, Yaz 1988, Sam 1990], l'approche utilisant l'algèbre de Lie [Gua 1989], l'approche fréquentielle [Boo 1983, Bay 1989, Ser 1990, Spe 1990, Kel 1993].

En fait, les méthodes de modélisation dynamique des robots flexibles sont, d'une manière générale, une combinaison des méthodes de modélisation dynamique des robots rigides et des méthodes d'analyse des structures flexibles telles que la méthode des éléments finis et l'analyse modale [Tza 1989, Che 1990, Mit 1992].

Toutes ces méthodes de modélisation utilisent un premier jeu de coordonnées relatif à un mouvement rigide de la structure, en général de grande amplitude, auquel on superpose de petits déplacements élastiques.

Pour notre part, nous avons utilisé un modèle formel basés sur une approche 'modèle référencé corps rigide' et sur une discrétisation par éléments finis [Che 1990, Mao 1999]. Après un bref rappel de ces deux notions, nous établissons le modèle dynamique de façon générale, à partir du formalisme de Lagrange dont les coefficients sont calculés grâce à une technique simple et rapide. Enfin, nous présentons le modèle dynamique des robots à un et deux bras flexibles.

I.2 Choix des coordonnées

Le 'modèle référencé corps rigide' consiste à utiliser un système de coordonnées qui distingue les mouvements du solide rigide des mouvements dus aux déformations élastiques. Nous nous plaçons dans notre travail, dans le cadre de la mécanique des corps déformables où, aux mouvements de solide rigide de grande amplitude, se superposent de petits mouvements et déformations réversibles linéaires par rapport aux charges appliquées. La structure est supposée être constituée d'éléments mécaniques homogènes et isotropes qui peuvent être assimilés à des barres ou des poutres.

Pour déterminer la position d'un point M du bras du robot, on distingue un champ de solide rigide $\mathbf{u_r}(M)$, virtuel, associé aux degrés de liberté de chacune des articulations et un champ de déplacements élastiques $\mathbf{u_e}(M)$ mesuré par rapport à la configuration non déformée de chaque corps. Le déplacement résultant global est:

$$\mathbf{u}(M) = \mathbf{u_r}(M) + \mathbf{u_e}(M) \qquad \text{(I.1)}$$

I.3 Caractéristiques de la dynamique des structures

Dans l'étude de la mécanique des structures on peut distinguer deux approches : statique et dynamique. L'étude de la dynamique des structures diffère de l'étude statique par deux aspects importants. Le premier est, par définition, la variation au cours du temps du problème dynamique. En effet, contrairement au cas statique, la charge et la réponse du système sont fonction du temps et la solution n'est donc pas unique. Le deuxième aspect est plus profond. Supposons un bras de robot flexible sujet à une charge statique p, les moments internes ainsi que la flexion du bras dépendent directement de cette charge et peuvent être calculés à partir de p grâce aux principes d'équilibre des forces. Alors que si la charge $p(t)$ est appliquée d'une façon dynamique, les déplacements du bras qui en résultent seront associés aux accélérations qui produisent des forces d'inertie qui s'opposent à ces accélérations.

Ainsi, les moments fléchissants doivent-ils équilibrer non seulement les forces extérieures mais aussi les forces d'inertie qui résultent de l'accélération du bras. Les forces d'inertie qui résistent de cette façon aux accélérations de la structure sont ceux qui caractérisent le mieux le problème dynamique.

D'une manière générale, si les forces d'inertie représentent une part importante de la charge totale équilibrée par les forces élastiques internes de la structure, alors le caractère dynamique doit être pris en compte dans la résolution du problème. Si par contre, la lenteur des mouvements est telle que l'on puisse négliger les forces d'inertie, l'analyse du problème pour n'importe quel instant donné peut être effectuée à partir d'une approche statique et ce même si la charge et la réponse du système varient au cours du temps.

I.4 Méthode de discrétisation

Le problème tel qu'il vient d'être exposé, montre que l'analyse d'un tel système est assez compliquée du fait que les forces d'inertie résultent du déplacement de la structure du robot qui, en retour, est lui même influencé par l'amplitude des forces d'inertie. Ce cycle interdépendant de cause et d'effet ne peut être résolu qu'en formulant le problème en termes d'équations différentielles couplées.

De plus, comme la masse du bras est uniformément distribué sur toute sa longueur, il faudrait calculer les déplacements et accélérations en tout point le long de l'axe si l'on veut déterminer entièrement les forces d'inertie mises en jeux. Dans ce cas, on doit développer le problème en termes d'équations aux dérivées partielles du fait que la position le long du bras ainsi que le temps, doivent être pris comme variables indépendantes.

Pour étudier ce système de dimension infinie, la solution consiste souvent à ramener le problème à un système d'équations aux dérivées ordinaires, de dimension finie, en discrétisant successivement les variables d'espace et de temps. Pour cela on peut utiliser deux méthodes de discrétisation: par éléments finis ou par décomposition modale.

Dans le cas de la discrétisation par éléments finis (adoptée dans notre travail), le champ de déplacement élastique $\mathbf{u_e}(M, t)$ est approximé par sous domaines:

$$\mathbf{u_e}(M, t) = \mathbf{N_e}(M)\ \mathbf{q_e}(t) \tag{I.2}$$

où, $\mathbf{N_e}(M)$ représente la matrice de fonctions de forme (de type polynomial, en général) et $\mathbf{q_e}(t)$ représente le vecteur des degrés de liberté élastique aux nœuds défini par le déplacement et/ou les dérivées du déplacement en ces nœuds.

La difficulté de cette approche réside dans le choix judicieux des fonctions de forme et des éléments finis, en plus du fait qu'elle introduit souvent un grand nombre de degrés de liberté. Elle présente cependant l'avantage d'assurer la convergence vers la solution exacte.

En ce qui concerne la décomposition modale, ou décomposition sur base de fonctions propres, le champ $\mathbf{u_e}$ (M, t) est décrit comme la résultante des contributions d'un nombre fini de modes:

$$\mathbf{u_e}(M, t) = \sum_{i=1}^{m} \phi_i(M)\, q_{e_i}(t) \tag{I.3}$$

avec, $\phi_i(M)$: $i^{ème}$ fonction propre et $q_{e_i}(t)$: $i^{ème}$ composante modale.

Les fonctions propres ϕ_i sont obtenues en effectuant une analyse modale de chacun des corps de la structure soit à partir d'un modèle à éléments finis, soit à partir d'une approche expérimentale [Lou 1997].

La difficulté de cette méthode réside dans le choix du nombre de modes propres et des conditions aux limites qui définissent la base. Cependant elle est très utilisée car son élaboration est moins ardue que celle de la précédente et elle permet une écriture plus condensée grâce à un nombre limité de modes.

I.5 Formulation de l'équation du mouvement

Nous utilisons, ici, le formalisme de Lagrange pour décrire le mouvement dynamique du robot flexible. Pour cela nous devons d'abord calculer l'énergie cinétique totale $\mathbf{T}$ ($\mathbf{q}$, $\dot{\mathbf{q}}$) et l'énergie potentielle totale $\mathbf{U}(\mathbf{q})$ du système. On note: $\mathbf{q} = [\mathbf{q_r}, \mathbf{q_e}]^T$, $\dot{\mathbf{q}} = \frac{d\mathbf{q}}{dt}$, $\ddot{\mathbf{q}} = \frac{d\dot{\mathbf{q}}}{dt}$, avec $\mathbf{q_r}$ et $\mathbf{q_e}$ représentant les degrés de liberté rigides et élastiques respectivement.

Si on considère les frottements et les efforts extérieurs sur l'organe terminal nuls, on peut écrire l'équation suivante [Dom 1988]:

$$\mathbf{L_r \Gamma} = \frac{d}{dt}(\nabla_{\dot{\mathbf{q}}}\mathbf{T}) - \nabla_{\mathbf{q}}\mathbf{T} + \nabla_{\mathbf{q}}\mathbf{U} \qquad \text{(I.4)}$$

avec,

$\mathbf{L_r \Gamma} = \left[\Gamma_1 .. \Gamma_{n_r}, 0..0\right]^T$ vecteur (n) désignant les couples appliqués sur les articulations.

$\nabla_{\dot{\mathbf{q}}}\mathbf{T} = \left[\frac{\partial \mathbf{T}}{\partial \dot{q}_1} ... \frac{\partial \mathbf{T}}{\partial \dot{q}_n}\right]^T$ représente le gradient de $\mathbf{T(q,\dot{q})}$ par rapport à $\dot{\mathbf{q}}$.

$\nabla_{\mathbf{q}}\mathbf{T} = \left[\frac{\partial \mathbf{T}}{\partial q_1} ... \frac{\partial \mathbf{T}}{\partial q_n}\right]^T$ représente le gradient de $\mathbf{T(q,\dot{q})}$ par rapport à $\mathbf{q}$.

$\nabla_{\mathbf{q}}\mathbf{U} = \left[\frac{\partial \mathbf{U}}{\partial q_1} ... \frac{\partial \mathbf{U}}{\partial q_n}\right]^T$ représente le gradient de $\mathbf{U}(\mathbf{q})$ par rapport à $\mathbf{q}$.

$n = n_r + n_e$ étant le nombre total de degrés de liberté du system, n_r le nombre de degrés de liberté rigides et n_e le nombre de degrés de liberté élastiques.

On peut aussi réécrire l'équation (I.4) sous la forme générale suivante [Che 1990]:

$$\mathbf{L_r\,\Gamma} = \mathbf{A(q)}\,\ddot{\mathbf{q}} + \mathbf{B(q)}\left[\dot{\mathbf{q}}\dot{\mathbf{q}}\right] + \mathbf{C(q)}\left[\dot{\mathbf{q}}^2\right] + \mathbf{Q(q)} + \mathbf{K}\,\mathbf{q} \tag{I.5}$$

où,

• **A(q)** est la matrice (n x n) symétrique, définie, positive, de l'énergie cinétique, aussi appelée matrice d'inertie du robot et peut être calculée grâce à la relation [Dom 1988]:

$$\mathbf{A}\,\dot{\mathbf{q}} = \nabla_{\dot{\mathbf{q}}}\mathbf{T} \quad \text{car} \quad \mathbf{T} = \frac{1}{2}\,\dot{\mathbf{q}}^{\mathrm{T}}\,\mathbf{A}\,\dot{\mathbf{q}} \tag{I.6}$$

• **B(q)** est la matrice (n x $n\cdot(n-1)/2$) des termes de Coriolis, ses éléments peuvent être calculés à partir de la relation:

$$\mathbf{B}_{i,jk} = \frac{\partial \mathbf{A}_{ij}}{\partial q_k} + \frac{\partial \mathbf{A}_{ik}}{\partial q_j} - \frac{\partial \mathbf{A}_{jk}}{\partial q_i} \tag{I.7}$$

avec i allant de 1 à n, j allant de 1 à n-1 et pour chaque j : k variant de j +1 à n.

• $\left[\dot{\mathbf{q}}\dot{\mathbf{q}}\right]$ est un vecteur ($n\cdot(n-1)/2$) donné par: $[\dot{q}_1\dot{q}_2 \ldots \dot{q}_1\dot{q}_n, \dot{q}_2\dot{q}_3 \ldots \dot{q}_2\dot{q}_n, \ldots, \dot{q}_{n-1}\dot{q}_n]^{\mathrm{T}}$.

• **C(q)** est la matrice (n x n) des termes centrifuges tel que:

$$\mathbf{C}_{ij} = \frac{\partial \mathbf{A}_{ij}}{\partial q_j} - \frac{1}{2}\frac{\partial \mathbf{A}_{jj}}{\partial q_i} \tag{I.8}$$

avec i et j allant de 1 à n.

• $\left[\dot{\mathbf{q}}^2\right]$ est un vecteur (n) donné par: $\left[\dot{q}_1{}^2 \ldots \dot{q}_n{}^2\right]^{\mathrm{T}}$

• **Q(q)** est le vecteur (n) des forces de gravité dont les éléments sont donnés par l'équation suivante:

$$\mathbf{Q}_i = \frac{\partial \mathbf{U_G}}{\partial q_i} \tag{I.9}$$

où, $\mathbf{U_G}$ représente l'énergie potentielle de gravité et i varie de 1 à n.

• $\mathbf{K}$ est la matrice (n x n) constante de raideur. Elle est calculée à partir de la relation [Che 1990]:

$$\mathbf{K}\,\mathbf{q} = \nabla_{\mathbf{q}}\,\mathbf{U_D} \tag{I.10}$$

avec, $\mathbf{U_D}$ étant l'énergie potentielle de déformation élastique et on a:

$$\mathbf{U_D} = \frac{1}{2}\,\mathbf{q}^{\mathrm{T}}\,\mathbf{K}\mathbf{q} \tag{I.11}$$

D'une façon générale, on peut écrire la matrice raideur $\mathbf{K}$ comme suit [Che 1990]:

$$\mathbf{K} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K_e} \end{bmatrix} \tag{I.12}$$

où les n_r premières lignes et colonnes de $\mathbf{K}$ sont nulles puisque $\mathbf{U_D}$ ne dépend pas de $\mathbf{q_r}$.

On peut décomposer $\mathbf{K_e}$ en sous-matrice de raideur $\mathbf{K_{e_i}}$ avec $\mathbf{K_e} = \begin{bmatrix} \mathbf{K_{e_1}} & \dots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{K_{e_b}} \end{bmatrix}$ pour i allant de 1 à b, b étant le nombre d'éléments finis considérés.

L'équation (I.5) décrit le modèle dynamique du robot à bras flexibles, elle représente une généralisation du modèle dynamique du robot rigide.

Les éléments des matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{Q}$, $\mathbf{K}$ sont détaillés en annexe A pour le cas du robot à un et deux bras flexibles.

Pour simplifier la manipulation du modèle on utilisera la forme condensée suivante:

$$\mathbf{L_r}\,\boldsymbol{\Gamma} = \mathbf{A(q)}\,\ddot{\mathbf{q}} + \mathbf{h(q,\dot{q})}\,\dot{\mathbf{q}} + \mathbf{K}\,\mathbf{q} \tag{I.13}$$

avec,

$$\mathbf{h(q,\dot{q})}\,\dot{\mathbf{q}} = \mathbf{B(q)}\left[\dot{\mathbf{q}}\dot{\mathbf{q}}\right] + \mathbf{C(q)}\left[\dot{\mathbf{q}}^2\right] + \mathbf{Q(q)} \tag{I.14}$$

I.6 Modélisation de robots flexibles à un et deux axes

Nous présentons dans ce qui suit le calcul détaillé du modèle dynamique des robots à un et deux bras flexibles (voir les Figures I.1 et I.3). On considère, dans notre travail, que les déplacements se font sur un plan horizontal (OXY).

Le terme de pesanteur $\mathbf{Q}$ n'étant pas présent on pourra alléger les expressions finales, notamment celles qui concernent le robot à deux bras flexibles. De plus ce type de robot étant, en pratique, surtout utilisé en aérospatial (satellite, stations, etc.), il n'est donc pas sujets à l'attraction terrestre.

I.6.1 Cas du robot flexible à un axe

La représentation schématique du robot flexible à un axe est donnée par la Figure I.1. On utilise un modèle discrétisé à un élément fini, ce qui conduit à trois degrés de liberté: $\mathbf{q} = [\mathbf{q_r}, \mathbf{q_e}]^T$ où, $\mathbf{q_r} = [\theta]$ est la position articulaire et $\mathbf{q_e} = [f, \alpha]^T$ représente les déplacements élastiques, flèche et rotation de section, en bout de bras, respectivement (Pham, 1992).

Assimilons le bras du robot flexible à une poutre de longueur L, de section S et de masse volumique ρ (voir Figure I.2), la masse de la poutre est désignée par m (avec $m = \rho S L$); alors le champ de déplacement en tout point M appartenant au volume de cette poutre est calculé comme la résultante de deux composantes.

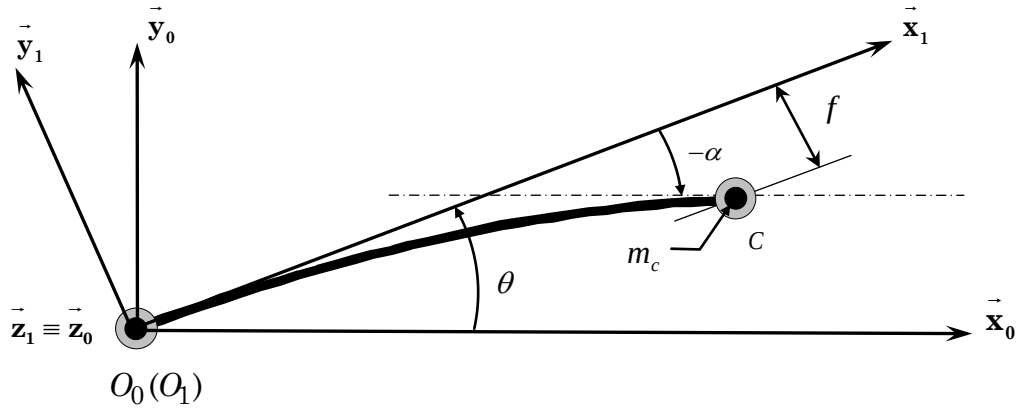

Figure I.1 Robot à un bras flexible.

La première composante est représentée par le champ de déplacement rigide de la poutre non déformée (Figure I.2), il est donné par le vecteur:

$$\mathbf{u_r}(M) = x\,\vec{\mathbf{x}} + y\,\vec{\mathbf{y}} + z\,\vec{\mathbf{z}} \tag{I.15}$$

La deuxième composante est représentée par le champ de déplacement élastique mesuré par rapport à cette configuration non déformée (Figure I.1), il est donné par [Pha 1992]:

$$\mathbf{u_e}(M) = -y\,f_{,x}\,\vec{\mathbf{x}} + f\,\vec{\mathbf{y}} + 0\,\vec{\mathbf{z}} \tag{I.16}$$

avec, f représentant la flèche de déformation élastique au point M d'abscisse x et $f_{,x} = \dfrac{\partial f}{\partial x}$ représentant la tangente à la déformée au point M.

Finalement, les coordonnées de M dans le repère **R** sont données par le vecteur:

$$\mathbf{OM(R)} = (x - y\,f_{,x})\,\vec{\mathbf{x}} + (y + f)\,\vec{\mathbf{y}} + z\,\vec{\mathbf{z}} \tag{I.17}$$

A partir de cette équation on peut calculer la vitesse de déplacement du point M par rapport à O. Celle ci est la résultante de deux vitesses: la vitesse de translation $\mathbf{V_{tr}(R)}$ et la vitesse de rotation $\mathbf{V_{rot}(R)}$.

La vitesse de translation $\mathbf{V_{tr}(R)}$ de M relativement à O est décrite par l'équation:

$$\mathbf{V_{tr}(R)} = \left[\frac{\mathrm{d}}{\mathrm{d}t}(x - y f_{,x})\right]\vec{\mathbf{x}} + \left[\frac{\mathrm{d}}{\mathrm{d}t}(y + f)\right]\vec{\mathbf{y}} + \left[\frac{\mathrm{d}}{\mathrm{d}t}(z)\right]\vec{\mathbf{z}} \tag{I.18}$$

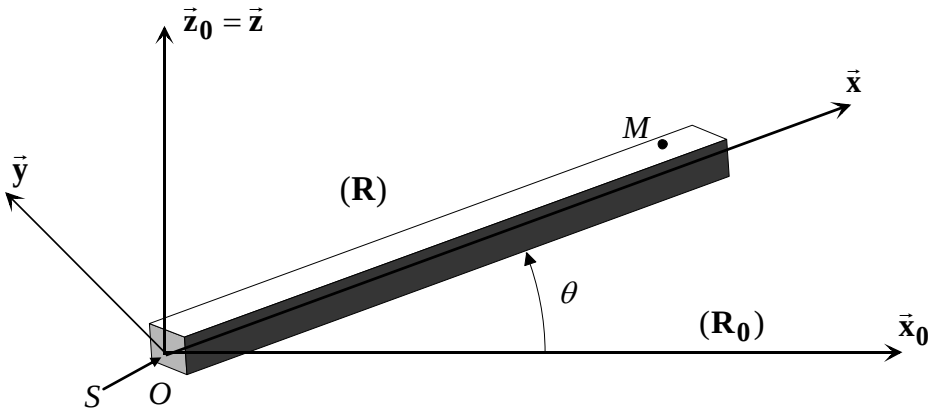

Figure I.2 Poutre non-déformée.

On obtient, donc l'expression suivante:

$$\mathbf{V}_{\mathrm{tr}}(\mathbf{R}) = -y\dot{f}_{,x}\ \vec{\mathbf{x}} + \dot{f}\ \vec{\mathbf{y}} + 0\ \vec{\mathbf{z}} \tag{I.19}$$

La vitesse de rotation $\mathbf{V}_{\mathrm{rot}}(\mathbf{R})$ du vecteur $\mathbf{OM(R)}$ par rapport à O s'écrit:

$$\mathbf{V}_{\mathrm{rot}}(\mathbf{R}) = \mathbf{\Omega}_{\mathrm{R}}^{\mathrm{R0}}(\mathbf{R}) \wedge \mathbf{OM(R)} \tag{I.20}$$

avec, $\mathbf{\Omega}_{\mathrm{R}}^{\mathrm{R0}}(\mathbf{R}) = [0,\ 0,\ \dot{\theta}]$ représente la vitesse de rotation angulaire du repère $\mathbf{R}$ par rapport au repère $\mathbf{R_0}$ exprimée dans $\mathbf{R}$.

On déduit donc l'expression de la vitesse de rotation $\mathbf{V}_{\mathrm{rot}}(\mathbf{R})$:

$$\mathbf{V}_{\mathrm{rot}}(\mathbf{R}) = [-(y+f)\ \dot{\theta}\ ]\ \vec{\mathbf{x}} + [(x - y\ f_{,x})\ \dot{\theta}\ ]\ \vec{\mathbf{y}} + 0\ \vec{\mathbf{z}} \tag{I.21}$$

Par conséquent, on obtient la vitesse de déplacement du point M par rapport à O exprimée dans $\mathbf{R}$,

$$\mathbf{V}_{M}^{O}(\mathbf{R}) = -[(y+f)\ \dot{\theta} + y\dot{f}_{,x}\ ]\ \vec{\mathbf{x}} + [(x - y\ f_{,x})\dot{\theta} + \dot{f}\ ]\ \vec{\mathbf{y}} + 0\ \vec{\mathbf{z}} \tag{I.22}$$

D'autre part, si la flèche $f(x, t)$ est exprimée en fonction des variables élastiques de bout de bras $\mathbf{q_e}$ au moyen de la fonction d'interpolation $\mathbf{N_e}$, on a la propriété fondamentale suivante [Che 1990]:

$$f(x, t) = \mathbf{N_e}(x)\ \mathbf{q_e}(t) \tag{I.23}$$

avec, $\mathbf{N_e} = [N_1,\ N_2]$ où $N_1 = \frac{3x^2}{L^2} - \frac{2x^3}{L^3}$ et $N_2 = L\left(-\frac{x^2}{L^2} + \frac{x^3}{L^3}\right)$.

Finalement, on peut calculer l'énergie cinétique $\mathbf{T}_m$ de la poutre grâce à l'équation [Dom 1988]:

$$2\,\mathbf{T}_m = \rho \int_0^L \left[\int_S [\mathbf{V}_M^O(\mathbf{R})]^2\, ds\right] dx \tag{I.24}$$

Sachant que $\int_S y\, ds = 0$ puisque l'on suppose les sections symétriques par rapport à l'axe $\vec{\mathbf{z}}$ et que $\int_S y^2\, ds = I$, où I représente le moment quadratique de section.

L'expression de l'énergie cinétique $\mathbf{T}_m$ devient alors:

$$2\mathbf{T}_m = \rho\int_0^L [\, I\dot{\theta}^2 + S\mathbf{q_e}^{\mathrm{T}}\mathbf{N_e}^{\mathrm{T}}\mathbf{N_e}\mathbf{q_e}\dot{\theta}^2 + I\dot{\mathbf{q}}_\mathbf{e}^{\mathrm{T}}\mathbf{N_{e,x}}^{\mathrm{T}}\mathbf{N_{e,x}}\dot{\mathbf{q}}_\mathbf{e} + 2I\dot{\theta}\mathbf{N_{e,x}}\dot{\mathbf{q}}_\mathbf{e} + Sx^2\dot{\theta}^2\,]\,dx$$

$$+\ \rho\int_0^L [\, I\mathbf{q_e}^{\mathrm{T}}\mathbf{N_{e,x}}^{\mathrm{T}}\mathbf{N_{e,x}}\mathbf{q_e}\dot{\theta}^2 + 2Sx\dot{\theta}\mathbf{N_e}\dot{\mathbf{q}}_\mathbf{e} + S\mathbf{q_e}^{\mathrm{T}}\mathbf{N_e}^{\mathrm{T}}\mathbf{N_e}\dot{\mathbf{q}}_\mathbf{e}\,]\,dx \qquad \text{(I.25)}$$

ou bien encore,

$$2\mathbf{T}_m = \rho I L\dot{\theta}^2 + \rho S\mathbf{q_e}^{\mathrm{T}}\begin{bmatrix} \frac{13L}{35} & \frac{-11L^2}{210} \\ \frac{-11L^2}{210} & \frac{L^3}{105} \end{bmatrix}\mathbf{q_e}\dot{\theta}^2 + \rho I\dot{\mathbf{q}}_\mathbf{e}^{\mathrm{T}}\begin{bmatrix} \frac{6}{5L} & \frac{-1}{10} \\ \frac{-1}{10} & \frac{2L}{15} \end{bmatrix}\dot{\mathbf{q}}_\mathbf{e}$$

$$+\ \rho S\frac{L^3}{3}\dot{\theta}^2 + \rho I\mathbf{q_e}^{\mathrm{T}}\begin{bmatrix} \frac{6}{5L} & \frac{-1}{10} \\ \frac{-1}{10} & \frac{2L}{15} \end{bmatrix}\mathbf{q_e}\dot{\theta}^2 + \rho S\dot{\mathbf{q}}_\mathbf{e}^{\mathrm{T}}\begin{bmatrix} \frac{13L}{35} & \frac{-11L^2}{210} \\ \frac{-11L^2}{210} & \frac{L^3}{105} \end{bmatrix}\dot{\mathbf{q}}_\mathbf{e}$$

$$+\ 2\rho S\dot{\theta}\begin{bmatrix} \frac{7L^2}{20} & \frac{-L^3}{20} \end{bmatrix} + 2I\dot{\theta}\begin{bmatrix} 1 & 0 \end{bmatrix}\dot{\mathbf{q}}_\mathbf{e} \qquad \text{(I.26)}$$

L'équation (I.26) représente la contribution à l'énergie cinétique de l'élément fini poutre. Pour avoir l'énergie cinétique totale **T** du robot, il faudra prendre aussi en compte l'énergie cinétique de la masse concentrée m_c qui représente la charge et des inerties concentrées J_A et J_B à l'origine et à l'extrémité du bras, respectivement, qui correspondent aux parties rigides du robot:

$$\mathbf{T} = \mathbf{T}_m + \frac{1}{2}J_A\dot{\theta}^2 + \frac{1}{2}J_B(\dot{\theta}+\dot{\alpha})^2 + \frac{1}{2}m_c\,\mathbf{V}_C^O(\mathbf{R})^2 \qquad \text{(I.27)}$$

Notons que le premier terme à droite de l'équation (I.27) représente l'énergie cinétique du bras flexible. Le deuxième terme est dû au moment d'inertie de la partie de la masse du moteur (actionneur) relativement au bras (partie mobile du moteur encrée à l'origine du bras). Le troisième terme est dû au moment d'inertie de la masse concentrée à l'extrémité du bras. Le quatrième terme représente l'énergie cinétique de la charge au point *C*.

L'équation (I.27) est ensuite utilisée avec (I.6), (I.7) et (I.8) pour déduire les expressions des matrices **A**, **B** et **C**. Celles-ci sont données en Annexe A.2.

Remarquons que la matrice de pesanteur **Q** est nulle, car nous avons considéré un robot horizontal, quant à la matrice raideur **K**, elle est de la forme :

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 12EI/L^3 & -6EI/L^2 \\ 0 & -6EI/L^2 & 4EI/L \end{bmatrix} \tag{I.28}$$

Si nous supposons connue la longueur du bras L, le modèle dynamique peut s'écrire linéairement en fonction du vecteur de paramètres dynamiques du robot [Gau 1991]:

$$\mathbf{L_r}\,\boldsymbol{\Gamma} = \mathbf{D}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})\,\mathbf{X} \tag{I.29}$$

où, $\mathbf{D(q,\dot{q},\ddot{q})}$ est appelée matrice d'information du système (voir Annexe D) et **X** représente le vecteur des paramètres dynamiques du robot flexible.

Dans le cas du robot à un bras flexible et si m représente la masse du bras on a:

$$\mathbf{X} = [J_A + J_B,\ J_B,\ m_c,\ \rho I,\ \rho SL = m,\ EI]^{\mathrm{T}} \tag{I.30}$$

I.6.2 Cas du robot flexible à deux axes

La représentation schématique du robot à deux bras flexibles est donnée par la Figure I.3. Pour décrire les mouvements de ce robot on utilise un modèle discrétisé à un élément fini par bras, ce qui conduit à six variables au total, deux variables rigides et quatre variables élastiques [Che 1990].

Ces variables représentent les sorties du système, ils sont décrits par le vecteur $\mathbf{q} = [\mathbf{q_r},\ \mathbf{q_e}]^{\mathrm{T}}$. Le vecteur $\mathbf{q_r} = [\theta_1,\ \theta_2]^{\mathrm{T}}$ regroupe les positions articulaires de chaque bras et le vecteur $\mathbf{q_e} = [f_1,\ \alpha_1,\ f_2,\ \alpha_2]^{\mathrm{T}}$ regroupe les déplacements élastiques, flèches et rotations de sections, au bout de chaque bras.

Le vecteur des couples articulaires est donné par $\Gamma = [\Gamma_1,\ \Gamma_2]^{\mathrm{T}}$ et représente l'entrée de notre système.

On calcule l'énergie cinétique du premier et du deuxième bras $\mathbf{T}_{m_1}$ et $\mathbf{T}_{m_2}$ respectivement comme dans le cas du robot à un axe. On déduit l'énergie cinétique totale du robot comme étant:

$$\mathbf{T} = \mathbf{T}_{m_1} + \mathbf{T}_{m_2} + \frac{1}{2}J_{A_1}\dot{\theta}_1^{\,2} + \frac{1}{2}J_{B_1}(\dot{\theta}_1+\dot{\alpha}_1)^2 + \frac{1}{2}J_{A_2}(\dot{\theta}_1+\dot{\alpha}_1+\dot{\theta}_2)^2$$

$$+ \frac{1}{2}J_{B_2}(\dot{\theta}_1+\dot{\alpha}_1+\dot{\theta}_2+\dot{\alpha}_2)^2 + \frac{1}{2}m_{c_1}V(O_2)^2 + \frac{1}{2}m_{c_2}V(C)^2 \qquad \text{(I.31)}$$

où, J_{A_i} et J_{B_i} représentent les inerties concentrées à l'origine et à l'extrémité du bras i, respectivement (i = 1, 2).

Notons que le premier et le deuxième terme à droite de l'équation (I.31) représentent l'énergie cinétique du premier et du deuxième bras flexible, respectivement. Le troisième terme est dû au moment d'inertie de la partie de la masse du premier moteur (actionneur) relative au premier bras (partie mobile du moteur 1 ancrée à l'origine du bras 1). Le quatrième et le cinquième terme sont dus au moment d'inertie de la portion de masse du second moteur relative au premier bras et de la portion de masse du second moteur relative au deuxième bras, respectivement. Le sixième terme est dû au moment d'inertie de la masse au point C (la charge). Le septième et le huitième terme sont dus à l'énergie cinétique de la masse au point O_2 et C, respectivement.

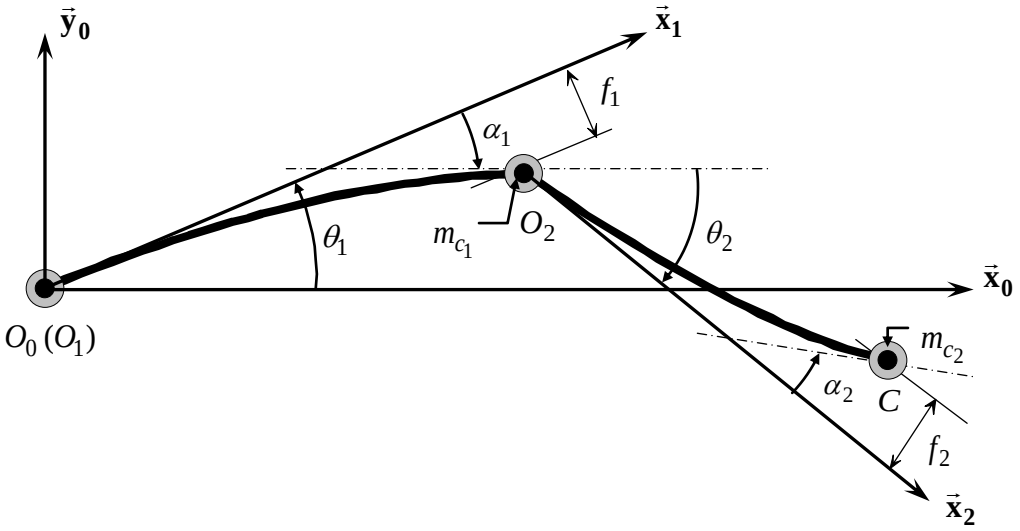

Figure I.3 Robot à deux bras flexibles.

La matrice d'inertie **A** calculée suivant l'équation (I.6) est donnée en annexe A.2. Les matrices **B** et **C** sont ensuite obtenues par dérivation de **A** suivant les équations (I.7) et (I.8).

Comme dans le cas à un axe, les termes de pesanteur sont également nuls, le robot évoluant dans un plan horizontal. Les sous-matrices raideur ont la forme suivante :

$$\mathbf{K}_{\mathbf{e}_i} = \begin{bmatrix} \dfrac{12\,E_i I_i}{{L_i}^3} & \dfrac{-6\,E_i I_i}{{L_i}^2} \\ \dfrac{-6\,E_i I_i}{{L_i}^2} & \dfrac{4\,E_i I_i}{L_i} \end{bmatrix} \tag{I.32}$$

La propriété de linéarité du modèle par rapport aux paramètres dynamiques (I.29) est aussi valable pour le cas du robot à deux bras flexible. Le vecteur **X** vaut dans ce cas:

$$\mathbf{X} = [J_{A_1} + J_{B_1},\ J_{B_1},\ m_{c_1} + m_{c_2},\ \rho_1 I_1,\ m_1,\ E_1 I_1,$$

$$J_{A_2} + J_{B_2},\ J_{B_2},\ m_{c_2},\ \rho_2 I_2,\ m_2,\ E_2 I_2]^{\mathrm{T}} \tag{I.33}$$

m_1 et m_2 représentant les masses des bras 1 et 2, respectivement.

I.7 Conclusion

Dans ce chapitre une méthode simple et rapide est décrite pour l'obtention du modèle dynamique d'un robot flexible en utilisant la discrétisation par éléments finis associé au formalisme de Lagrange. Cette méthode a été utilisée, par la suite, pour le calcul des coefficients dynamiques d'un robot horizontal à un et deux bras flexibles.

On remarque, selon les expressions obtenues, que le modèle dynamique d'un robot flexible est beaucoup plus complexe que celui d'un robot rigide. En effet la principale différence réside dans le fait qu'un robot flexible à la différence d'un robot rigide, est un système sous actionné car il possède plus de sorties à contrôler (variables rigides et élastiques) que d'entrées de commande (couples articulaires appliqués).

Ceci se traduit par la présence d'équations de couplage dynamique entre les variables rigides et élastiques dont il faudra tenir compte lors du contrôle. Plus de détails serons donnés aux chapitres III et IV, qui seront consacrés à la commande des robots flexibles.

Supposons maintenant que le modèle dynamique du système considéré soit inconnu ou trop complexe pour être établi entièrement. Nous devons donc utiliser un modèle statistique pour représenter le système. Ce type de modélisation est basé sur la théorie de l'apprentissage par les données, car on utilise exclusivement les valeurs des entrées et sorties du système pour établir un modèle.

Les réseaux de neurones artificiels sont le plus souvent utilisés dans ce type de modélisation pour leurs propriétés d'approximations universelles, de parcimonie et de rapidité de calcul. Nous utiliserons ce type de modèle en association avec le modèle dynamique pour établir un nouveau type de commande hybride, qui fera l'objet du Chapitre IV.

Le Chapitre suivant présente quelques généralités sur les réseaux de neurones artificiels ainsi que quelques outils qui seront utilisés dans le Chapitre IV.

Chapitre II

Modélisation par réseaux de neurones

II.1 Introduction

Toutes les fonctions biologiques de l'être humain utilisent un réseau de neurone très complexe: le cerveau. Ainsi, l'être humain possède, en moyenne, cent milliards de neurones qui lui permettent de lire, écrire, respirer, réfléchir etc.

Les scientifiques ont toujours été fascinés par le fonctionnement du cerveau humain et surtout par sa capacité à apprendre, à reconnaître et à s'adapter à de nouvelles situations. Ils commencent à peine à élucider son mécanisme complexe de fonctionnement. Ainsi, les fonctions biologiques y compris la mémoire sont stockées dans les neurones et leurs interconnexions.

L'apprentissage est décrit comme l'établissement de nouvelles connexions entre les neurones ou la modification de connexions qui existent déjà. Se basant sur ces connaissances, les scientifiques tentent de développer des techniques utilisant des algorithmes mathématiques pour modéliser le fonctionnement du neurone afin de construire un réseau artificiel de neurones formels et par conséquent approcher l'intelligence du cerveau.

Les Réseaux de Neurones Artificiels (RNA) n'ont pas la puissance du cerveau humain mais ils peuvent être entraînés pour atteindre des performances relativement élevées. Les RNA sont donc une modélisation simplifiée du fonctionnement complexe du cerveau biologique. Ils miment sa façon d'organiser, de stocker et de traiter l'information.

II.1.1 Bref historique

Les premiers travaux sur les RNA datent de la fin du 19ème et le début du 20ème siècle, effectués par une équipe multidisciplinaire de chercheurs physiciens et psychologues tels que Helmholtz, Mach et Pavlov. Leurs travaux ont porté essentiellement sur l'établissement de théories générales sur l'apprentissage, le conditionnement, la vision, mais n'ont pas donné de modèles mathématiques précis décrivant le fonctionnement d'un neurone.

La vision moderne des réseaux de neurones a commencé en 1940 avec les travaux de Warren McCulloch et Walter Pitts [Mcc 1943]. Ils ont introduit le premier modèle mathématique du neurone et ont montré qu'un simple réseau de neurones peut calculer une quelconque fonction arithmétique ou logique.

Un peu plus tard en 1949, Donal Hebb, psychologue Canadien, a proposé une théorie fondamentale pour l'apprentissage des réseaux de neurones. Pour expliquer les effets d'apprentissage en fonction de l'expérience, il propose que les cellules apprennent à modifier l'intensité des connexions qui les relient en fonction de leur activité simultanée. C'est la 'loi de Hebb'.

Vers la fin des années 50, Frank Rosenblatt a présenté un réseau appelé 'perceptron', et ses règles d'apprentissages [Ros 1958]. Il a démontré sa capacité à reconnaître des formes. Ce fut la première application concrète et pratique des réseaux de neurones artificiels.

En 1960, Bernard Widrow et Ted Hoff ont introduit de nouveaux algorithmes d'apprentissage ('Least Mean Square error' ou 'LMS algorithm') pour l'entraînement de réseaux linéaires [Wid 1960, Wid 1985, Wid 1990], dont la structure et les capacités sont similaires aux perceptrons.

Vers la fin des années soixante Marvin Minsky et Seymour Papert [Min 1969] ont démontré les limitations des perceptrons et des règles d'apprentissage établies précédemment par Rosenblatt, Widrow et Hoff. Influencés par les résultats de Minsky et Papert, et handicapés par les moyens informatiques de l'époque qui ne permettaient pas d'effectuer des calculs complexes, plusieurs chercheurs ont abandonné leurs travaux. Durant une décennie, l'intérêt de la plupart des chercheurs pour les RNA a, ainsi, fortement baissé.

Quelques importants travaux ont cependant continué durant les années 70, tels que ceux de Teuvo Kohonen [Koh 1972], James Anderson [And 1972] et Stephen Grossberg [Gro 1976].

Durant les années 80, l'avènement de l'outil informatique, le développement des micro-ordinateurs personnels et des stations de travail ont donné un nouvel essor au domaine. De nouveaux concepts ont, également, été introduits par le physicien Jhon Hopfield [Hop 1982] comme le nouveau type de réseau de neurone qu'il a introduit et qui porte son nom. La seconde raison du développement des réseaux de neurones artificiels est l'invention de l'algorithme de rétro propagation des erreurs pour l'apprentissage des réseaux multicouches [Rum 1986]. Cet algorithme a enfin répondu aux critiques de Minsky et Papert de 1960.

Ces nouveaux développements ont donné une renaissance au domaine qui a connu, dès lors, constamment la découverte de nouvelles théories et de nouveaux algorithmes. Les réseaux de neurones sont devenus un domaine de recherche multidisciplinaire attractif, qui réunit des électroniciens, des mathématiciens, des informaticiens, des biologistes, etc. Des recherches récentes parlent même de culture de réseaux de neurones artificielles avec des réseaux de neurones biologiques (connectique RNA/RNB) [Arb 2003].

II.1.2 Quelques domaines d'applications

Les RNA sont utilisés dans des domaines aussi divers que variés. On peut classer les réseaux de neurone artificiels suivant leurs applications en trois grandes catégories:

- Réseaux de neurones pour la classification ou la discrimination. Ces réseaux sont utilisés pour séparer un ensemble de données entrées/sorties et les classées en catégories suivant un critère donné. Ils trouvent leurs applications en robotique pour la reconnaissance de trajectoires, en instrumentation électronique pour la classification de signaux de capteurs, en traitement du signal ou de l'image pour la reconnaissance de caractères ou la reconnaissance d'objets par exemple.

Ils sont aussi utilisés en médecine pour le diagnostic de maladies, ou bien encore en pharmaco chimie pour la synthèse de nouvelles molécules à effets thérapeutiques et la conception de nouveaux médicaments, etc.

- Réseaux de neurones pour la modélisation et l'approximation. Si la fonction que doit réaliser le réseau est connue analytiquement, il réalise alors une approximation de fonction en trouvant la meilleure fonction non linéaire qui la représente.

Des études on montré qu'un réseau multicouche avec une seule couche cachée de neurones sigmoïdes et une couche de sortie linéaires permet d'approximer n'importe quelle fonction d'intérêt avec une précision arbitraire à condition de disposer de suffisamment de neurones sur la couche cachée [Dre 2005].

Si la fonction que doit réaliser le réseau est inconnue analytiquement, mais que l'on dispose de valeurs de cette fonction qui proviennent de mesures chimiques, physiques etc., le réseau réalise une modélisation ou une régression. C'est essentiellement dans ce type d'application que sont utilisés les réseaux de neurones non bouclés à apprentissage supervisé. Nous utiliserons ce type de réseaux au Chapitre IV pour l'élaboration d'une commande adaptative originale.

- Réseaux de neurones pour l'optimisation combinatoire. Ici, les RNA sont utilisés sans entrainement préalable, on laisse le réseau évoluer de manière aléatoire pour trouver une solution optimale. On les utilise pour répondre à des problèmes d'optimisation tels que: comment minimiser le coût de production à travers un ordonnancement des tâches et une utilisation optimale des ressources humaines et matérielles ? Comment accroitre la productivité ? etc.

II.2 Modèle mathématique

Un RNA est constitué d'un ensemble d'opérateurs non linéaires interconnectés, formants une famille de fonctions non linéaires, qui permet de construire, par apprentissage, une très large classe de modèles mathématiques. L'élément de base du réseau est le neurone formel.

II.2.1 Le neurone formel

Le neurone formel est une fonction algébrique paramétrée qui agit sur des variables appelées stimulus ou entrées du neurone. Le résultat de la fonction est appelé sortie du neurone. Le modèle mathématique d'un neurone artificiel à entrées multiples est illustré par la Figure II.1.

Le neurone artificiel à entrées multiples est constitué d'un intégrateur qui effectue la somme pondérée de ses entrées représentées par le vecteur $\mathbf{P} = \left[p_1, p_2, p_3, .., p_R\right]^{\mathrm{T}}$, pondérée par le vecteur des poids $\mathbf{W} = \left[w_{1,1}, w_{1,2}, w_{1,3}, .., w_{1,R}\right]$. Chaque entrée p_i est multipliée par le poids $w_{1,i}$ correspondant, comme indiqué sur la Figure II.1.

Le potentiel n du neurone s'écrit:

$$n = \mathbf{W}\mathbf{p} + b \tag{II.1}$$

avec, b scalaire représentant le biais.

La sortie a du neurone est alors donnée par:

$$a = f(\mathbf{W}\mathbf{p} + b) \tag{II.2}$$

avec, f fonction d'activation du neurone.

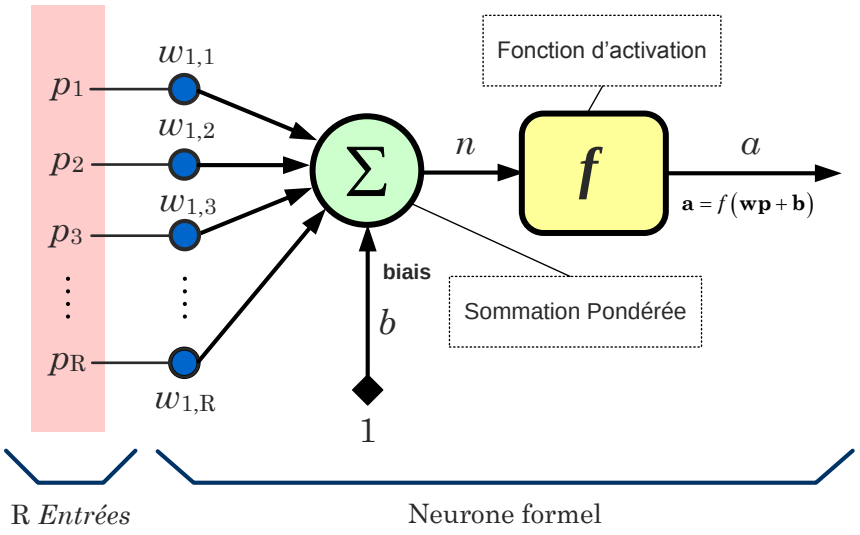

Figure II.1 Schéma d'un neurone artificiel à entrées multiples.

Le biais peut être considéré comme une entrée constante b ou une entrée égale à 1 avec un poids de connexion égal à b. Au Chapitre IV nous avons adopté cette deuxième possibilité dans notre réseau de neurone.

Nous avons ainsi considéré le biais comme une entrée supplémentaire, fixe, égale à 1 et nous avons intégré sa pondération dans la matrice des poids **W**.

Le poids d'un neurone artificiel représente la force de sa connexion. Un poids négatif inhibe une entrée alors qu'un poids positif la renforce.

Pour pouvoir simuler un réseau de neurones on considère le temps discret. C'est à dire qu'on suppose que tous les neurones sont synchrones et qu'à chaque instant t tous les neurones calculent simultanément leurs potentiels et produisent la sortie $a(t) = f\ (n(t))$. Ce qui permet d'éliminer le facteur temps des équations. Rappelons que dans les réseaux biologiques, les neurones sont asynchrones.

II.2.2 La fonction d'activation

La fonction de transfert ou bien encore fonction d'activation sert à introduire une non-linéarité dans le fonctionnement du neurone.

Il existe plusieurs types de fonctions d'activation, la Figure II.2 représente les quatre fonctions les plus utilisées: la fonction 'seuil', la fonction 'linéaire', la fonction 'sigmoïde' et la fonction 'tangente hyperbolique'.

- La fonction seuil présente deux intervalles. En dessous du seuil, le neurone est non-actif et sa sortie vaut soit 0 soit -1. Au dessus du seuil, le neurone est actif et sa sortie vaut 1.

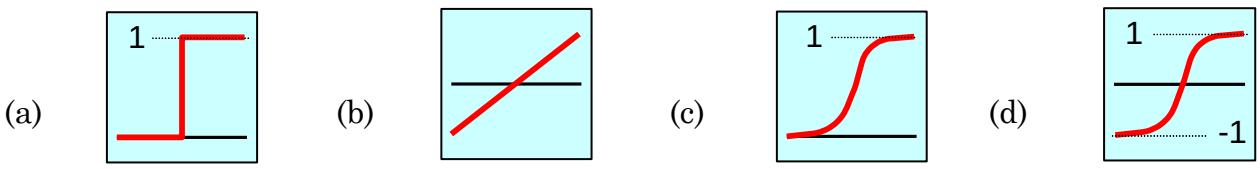

Figure II.2 Fonctions de transfert: (a) fonction seuil, (b) fonction linéaire, (c) fonction sigmoïde et (d) fonction tangente hyperbolique.

▪ La fonction linéaire reproduit généralement le potentiel n vers la sortie a du neurone:

$$f(n) = n \tag{II.3}$$

le neurone formel se contente ici de réaliser la somme pondérée de ses entrées.

▪ La fonction sigmoïde est définie par:

$$f(n) = \frac{1}{1 + e^{-\lambda n}} \tag{II.4}$$

avec, λ scalaire.

Une variante de la fonction sigmoïde est la fonction tangente hyperbolique qui a la même forme mais dont les asymptotes horizontales sont -1 et 1. Elle est définie par:

$$f(n) = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}} \tag{II.5}$$

cette fonction est souvent utilisé dans l'algorithme de retro-propagation car le calcul de sa dérivée est simple [Wid 1990].

Le choix de la fonction de transfert va dépendre de la nature du problème à résoudre. Ainsi, par exemple, la fonction seuil, dont les sorties valent 0 si l'entrée est inférieure à 0, et 1 si l'entrée est supérieure à 0, est utilisée pour les tâches de classification dure.

La fonction sigmoïde et la fonction tangente hyperbolique, du fait de leurs caractéristiques non linéaires, sont utilisées dans les problèmes de modélisation, d'approximation de fonctions mathématiques ou de classification floue, etc.

II.3 Les réseaux de neurones

II.3.1 Définition

L'intérêt des neurones réside dans les propriétés qui résultent de leur association en réseau, c'est-à-dire de la composition des fonctions non linéaires réalisées par chacun de ses neurones [Dre 2005].

Un RNA est donc une association de plusieurs neurones interconnectés entre eux et qui effectuent des calculs simultanément de façon parallèle.

Un réseau est défini par:

- ✓ son architecture: c'est-à-dire la topologie des connexions entre neurones
- ✓ ses caractéristiques de calculs: sa fonction de transfert, sa fonction de combinaison
- ✓ ses règles d'apprentissage.

II.3.2 Différents types de réseaux de neurones

On peut classer les réseaux de neurones artificiels suivant la direction dans laquelle évoluent les signaux à travers le réseau. Dans ce cas, on distingue deux catégories de réseaux. Les réseaux bouclés et les réseaux non bouclés.

Dans un réseau non bouclé, l'information circule des entrées vers la sortie sans retour vers les entrées. Le signal, en se déplaçant dans le réseau suivant le sens des connexions ne peut pas repasser deux fois par le même neurone.

Le réseau non bouclé réalise une (ou plusieurs) fonction non linéaire par composition de toutes les fonctions réalisées par les neurones des couches cachées.

Ce type de réseau est utilisé dans la classification et l'approximation de fonctions non linéaires. C'est celui que nous avons adopté dans notre travail.

Dans un réseau bouclé, lorsqu'on se déplace dans le réseau suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient en arrière (un cycle).

Ce type d'architecture est utilisé pour modéliser des processus dynamiques. En effet, lorsque l'état du système à un moment donné dépend non seulement de l'entrée mais aussi de l'état précédent du système, on reboucle alors les sorties correspondantes dans le réseau à ces variables d'états vers l'entrée du réseau.

Cependant on peut toujours revenir d'une architecture de réseau bouclé à une architecture de réseau direct (non bouclé) tout simplement en ajoutant des entrées (variables d'état) supplémentaires correspondantes aux sorties précédentes du système (état précédant du système), mais ceci aura pour conséquence de rendre plus complexe le réseau, avec plus d'entrées et plus de neurones à gérer [Dre 2005].

Le réseau bouclé, utilisé sans apprentissage trouve aussi son application dans les problèmes d'optimisation combinatoire et d'analyse de données. Deux exemples bien connus de réseaux bouclés, sont le réseau de Hopfield et le réseau d'Elman.

Enfin, on peut classer les réseaux de neurones artificiels suivant leurs fonctions de combinaison. Considérons un neurone quelconque, la fonction de combinaison est la fonction qui réalise la somme pondérée. Cette fonction peut changer selon le type de réseau: MLP (Multi-Layer Perceptron) ou RBF (Radial Basis Function).

Les réseaux de type MLP calculent une combinaison linéaire des entrées, c'est-à-dire que la fonction de combinaison renvoie le produit scalaire entre le vecteur des entrées et le vecteur des poids synaptiques [Mao 2010b].

Les réseaux de type RBF calculent dans la fonction de combinaison la distance entre l'entrée et le centre d'une fonction à base radiale (de forme gaussienne ou ondelette en général), c'est-à-dire que la fonction de combinaison renvoie la norme euclidienne du vecteur issue de la différence vectorielle entre les vecteurs d'entrées et les centres d'un jeu de fonctions à base radiale [Mao 2010a].

Les deux types de réseaux sont très utilisés dans l'approximation de fonctions non linéaires.

Dans ce travail, nous avons utilisé des réseaux de type MLP pour leurs simplicités et à cause du fait que dans les RBF, la complexité (le nombre de fonctions) croit d'une manière exponentielle avec la taille de l'espace d'entrée utilisée [Pri 2000].

II.3.3 Architecture d'un réseau à couches

Les réseaux non bouclés à couches tels que les réseaux MLP (Multi Layer Perceptron) sont constitués de neurones organisés en séries de couches qui sont interconnectés. La Figure II.3 représente la topologie d'un réseau à couches.

Dans ce type de réseaux on distingue la couche d'entrée qui correspond aux variables d'entrées du système et la dernière couche appelée couche de sortie, elle est constituée d'un ou de plusieurs neurones et correspond aux sorties du système. Les couches intermédiaires qui se trouvent entre la couche d'entrée et la couche de sortie sont appelées couches cachées.

Le nombre de neurones dans une couche cachée est à optimiser pour un système donné et n'est pas nécessairement le même pour chaque couche cachée.

Un neurone d'une couche cachée reçoit le résultat du traitement des neurones de la couche précédente (leurs sorties), traite à son tour l'information et sa sortie va constituer une entrée pour la couche suivante. Ainsi le calcul se fait de proche en proche en allant des entrées vers la sortie. C'est la propagation du réseau. La (ou les) fonction obtenue sera donc une composition de toutes les fonctions réalisées par tous les neurones du réseau.

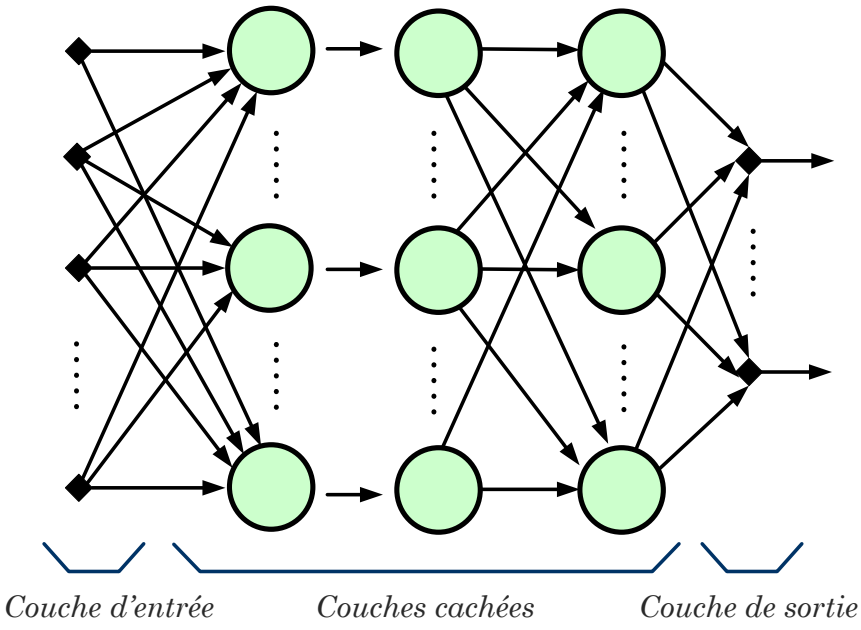

Figure II.3 Représentation schématique d'un réseau à couches.

La Figure II.4 représente une couche de S neurones qui est connecté à R entrées. Les R entrées de la couche forment le vecteur **p**. Chaque élément j de ce vecteur est connecté à tous les neurones i de la couche et est associé au poids $w_{i,j}$. Le poids $w_{i,j}$ désigne la connexion entre le neurone i et son entrée j. Souvent $w_{i,j}$ désigne le poids d'une connexion qui a comme point de départ le neurone j et comme point d'arrivée le neurone i, mais ceci n'est pas une règle générale.

Nous avons considéré dans ce chapitre les notations suivantes:

- L'ensemble des poids d'une couche forment une matrice $\mathbf{W}=\begin{bmatrix} w_{1,1} & \cdots & w_{1,R} \\ \vdots & \ddots & \vdots \\ w_{S,1} & \cdots & w_{S,R} \end{bmatrix}$,
- l'ensemble des S biais va constituer le vecteur $\mathbf{b}=\left[b_1,b_2,\cdots,b_S\right]^{\mathrm{T}}$,
- l'ensemble des S potentiels créés va constituer le vecteur $\mathbf{n}=\left[n_1,n_2,\cdots,n_S\right]^{\mathrm{T}}$,
- les sorties des S neurones de la couche vont constituer le vecteur $\mathbf{a}=\left[a_1,a_2,\cdots,a_S\right]^{\mathrm{T}}$.

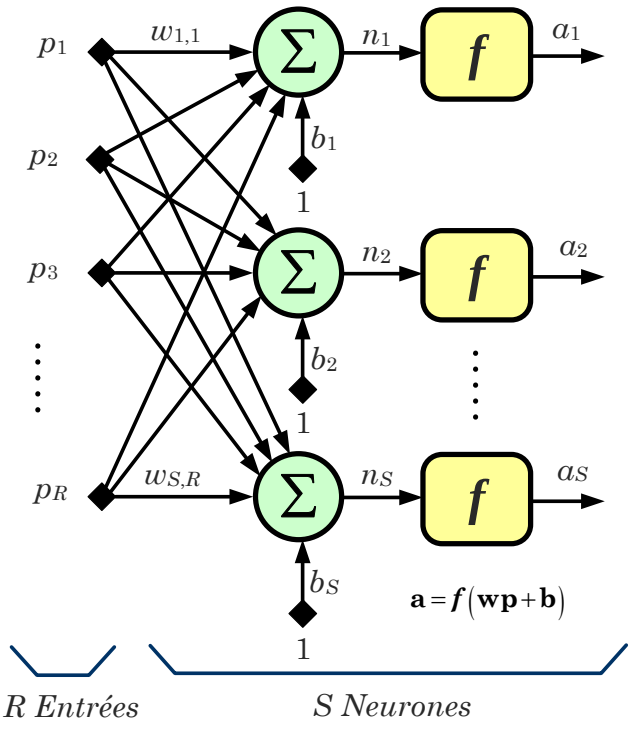

Figure II.4 Schéma d'une couche à S neurones et R entrées.

Une couche de S neurones peut également être représentée sous une forme matricielle telle que décrite par la Figure II.5 et par l'équation suivante:

$$\mathbf{a} = \boldsymbol{f}(\mathbf{W}\mathbf{p}+\mathbf{b}) \tag{II.6}$$

les fonctions d'activations au sein d'une même couche sont généralement identiques.

Ainsi, pour spécifier l'architecture d'un réseau il faut choisir le nombre de couches cachées et le nombre de neurones correspondants ainsi que le type de fonction de transfert de chaque couche; le nombre d'entrées et le nombre de neurones dans la couche de sortie dépendant quand à eux directement de la configuration du problème à résoudre ou du système à modéliser.

Le choix de l'architecture influe à la fois sur les capacités de calcul du réseau et sur le type d'apprentissage susceptible d'être utilisé. La couche cachée doit comporter suffisamment de neurones pour bien représenter les non linéarités de la fonction à approximer ou du système à modéliser.

Même si aucune méthode systématique n'existe pour déterminer de façon exacte le nombre de neurones dans la couche cachée ou le nombre de couches cachées, on a quand même la propriété d'approximation universelle suivante: « Toute fonction bornée, avec un nombre fini de discontinuités, peut être approximée uniformément, avec une erreur arbitraire, sur une région finie, en utilisant un réseau ayant une seule couche cachée à fonctions d'activations non linéaires et dérivables et une couche de sortie à fonctions d'activations linéaires [Hor 1989, Hor 1990, Hor 1991, Att 1995, Att 1999, Kec 2001, Arb 2003, Dre 2005, Mao 2007, Mao 2008b, Mao 2008a, Mao 2009c, Mao 2010b] ».

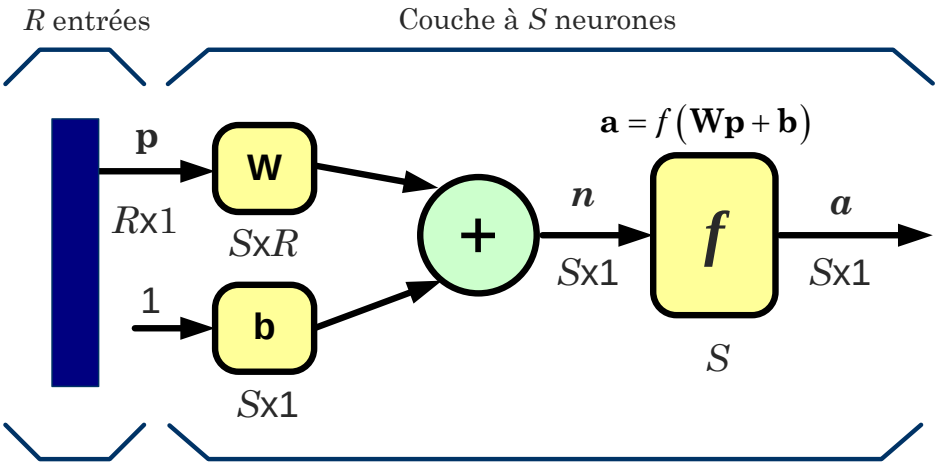

Figure II.5 Représentation matricielle d'une couche de S neurones et R entrées.

De plus, on a la propriété de parcimonie suivante. Le nombre de paramètres nécessaires pour effectuer une approximation avec une erreur donnée, augmente exponentiellement avec le nombre de variables, dans un modèle linéaire relativement à ces paramètres (cas d'une approximation avec un polynôme), alors il augmente d'une manière linéaire en utilisant un modèle non linéaire par rapport à ceux-ci (cas du MLP à couche cachée sigmoïde par exemple) [Bar 1993, Dre 2005].

Ceci justifie notre choix au Chapitre IV d'utiliser un réseau MLP à une couche cachée, de fonction d'activation tangente hyperbolique, pour répondre au problème d'approximation des fonctions non linéaires dans la commande.

II.4 Types d'apprentissage dans un RNA

Pour pouvoir entraîner un réseau de neurones artificiel il faut disposer d'un ensemble suffisant d'exemples c'est-à-dire de couples d'entrées/sorties.

Une des spécifications d'un réseau de neurone est son type d'apprentissage c'est-à-dire la manière avec laquelle va se comporter le réseau avec les exemples présentés lors de la phase d'apprentissage. On distingue habituellement trois types: l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage hybride. De plus cet apprentissage peut s'effectuer d'une manière incrémentale ou par lot (batch) [Gup 2003].

II.4.1 Apprentissage supervisé

Dans ce cas on fournit dans les exemples présentés au réseau, les entrées à traiter mais aussi les sorties ou réponses attendues. On connaît donc pour chaque exemple k l'entrée $p(k)$ et la sortie désirée $y^d(k)$ correspondante.

Le réseau effectue une évaluation du vecteur d'entrées $\mathbf{p}(k)$ (si le système possède plusieurs entrées), puis compare les valeurs obtenues avec le vecteur de sortie désiré $\mathbf{y^d}(k)$ (si le système possède plusieurs sorties). Le réseau modifie ensuite ses paramètres internes afin de minimiser l'erreur constatée.

Ce type d'apprentissage est utilisé en modélisation, en commande de processus, en approximation et en classification. Nous l'avons donc adopté dans notre travail.

II.4.2 Apprentissage non supervisé

Dans ce cas, on ne présente au réseau que les vecteurs d'entrées ensuite le réseau se base sur ses propriétés d'auto-organisation pour essayer de trouver des corrélations entre les données. Donc, le réseau est amené à regrouper les données selon des critères de ressemblance a priori inconnus et les classe dans un nombre fini de catégories (Clustering). Cette méthode est aussi utilisée en optimisation combinatoire où le réseau essaye de trouver par lui-même une solution optimale à un problème d'optimisation donné [Dre 2005].

II.4.3 Apprentissage hybride

Certains auteurs [Pal 1996, Lal 1996], utilisent le terme d'apprentissage hybride pour parler d'un couplage supervisé/non supervisé. Dans ce cas il s'agit de mettre en parallèle ou en série un réseau entraîné en mode supervisé et un autre en mode non supervisé. Ce type d'apprentissage est plus rare et encore mal exploré.

II.4.4 Les modes d'apprentissage

L'apprentissage supervisé peut s'effectuer suivant deux modes: le mode incrémental et le mode par lot (batch).

Dans le mode incrémental, les poids des connexions et les biais sont ajustés à chaque fois qu'une entrée est présentée au réseau alors que dans le mode batch, les poids et les biais ne sont modifiés qu'après la présentation de tous les ensembles d'entrées.

Dans le Chapitre IV nous avons considéré le mode par lot lors de l'approximation des fonctions non linéaires dans la commande. L'apprentissage s'effectuant ici 'hors ligne' c'est à dire avant de commander le système [Mao 2008b, Mao 2008a, Mao 2010b].

Nous avons réservé le mode incrémental pour la commande neuronale adaptative car l'apprentissage s'effectue 'en ligne' c'est-à-dire pendant la commande. Les poids de la commande neuronale adaptative sont donc mis à jour à chaque présentation de l'erreur sur la poursuite de la trajectoire [Mao 2007, Mao 2009c, Mao 2009b].

II.5 Règles de base de l'apprentissage

L'objectif de l'apprentissage est de fournir une méthode au réseau afin qu'il puisse ajuster ses paramètres lorsqu'on lui présente des exemples à traiter.

On appelle 'apprentissage' ou 'entraînement' des réseaux de neurones la procédure qui consiste à modifier les paramètres des neurones du réseau (des poids de connexions entre les neurones et des biais du réseau) afin qu'il reproduise au mieux les valeurs désirées. Il existe de nombreuses règles qui ont été mises au point pour permettre l'apprentissage d'un réseau. La plus utilisée est l'apprentissage par correction d'erreur.

II.5.1 Apprentissage par correction d'erreur

Cet apprentissage est basé sur la correction de l'erreur observée à la sortie du réseau. Plusieurs fonctions d'erreur existent, nous avons retenu celle correspondant aux moindres carrés.

Considérons un réseau multicouche à S neurones dans la couche de sortie. Supposons que nous disposons de K exemples ou couples d'entrées/sorties ($\mathbf{p}$, $\mathbf{y^d}$). $\mathbf{p}$ est le vecteur d'entrée et $\mathbf{y^d}$ est le vecteur de sortie désiré. Soit $\mathbf{a}$ le vecteur de sortie réel du réseau.

Le nombre de neurones dans la couche de sortie étant égale au nombre de variables de sorties du système, on a $\mathbf{a}=[a_1,a_2,..,a_S]$ et $\mathbf{y^d}=[y_1^d,y_2^d,..,y_S^d]$. La sortie du réseau $\mathbf{a}$ est calculée, de proche en proche, de la couche d'entrées vers la couche de sortie. C'est la propagation avant ou relaxation du réseau.

L'erreur observée à la sortie du réseau au niveau d'un neurone *i*, quand on présente un exemple *k* en entrée, est donnée par:

$$e_i(k) = y_i^d(k) - a_i(k) \tag{II.7}$$

L'apprentissage par correction d'erreur consiste à minimiser la fonction d'erreur *E* aussi appelée fonction de coût. Elle est définie, ici, par la somme des carrés des erreurs effectuée sur chaque neurone de la couche de sortie, à la présentation de l'exemple *k*:

$$E(k) = \sum_{i=1}^{S} e_i(k)^2 \tag{II.8}$$

Le calcul de cette fonction d'erreur est ensuite utilisé pour modifier les poids dans le réseau de la manière suivante.

Soit $w_{i,j}$ le poids de la connexion qui relie la sortie du neurone j d'une couche à l'entrée du neurone i de la couche suivante. La nouvelle valeur du poids $w_{i,j}$ à l'itération (k+1) s'écrit en fonction de la valeur du poids à l'itération précédente (k) et de la variation $\Delta w_{i,j}$ de la manière suivante:

$$w_{i,j}(k+1) = w_{i,j}(k) + \Delta w_{i,j}(k) \tag{II.9}$$

La variation $\Delta w_{i,j}$ est quand à elle calculée relativement à la fonction d'erreur *E* de la manière suivante:

$$\Delta w_{i,j}(k) = -\mu \frac{\partial E(k)}{\partial w_{i,j}} \tag{II.10}$$

où, μ est un scalaire multiplicateur appelé pas du gradient ou taux d'apprentissage (learning rate), il est compris entre 0 et 1.

Cette méthode est définie comme 'la méthode de descente du gradient'. Le calcul du gradient $\frac{\partial E(k)}{\partial w_{i,j}}$ sera détaillé plus loin.

L'apprentissage consiste donc, à modifier les paramètres du réseau (les poids et les biais) de façon à faire converger les sorties obtenues par le réseau vers les sorties désirées. Il continue jusqu'à ce que l'écart entre la valeur calculée par le réseau et la valeur donnée dans la série d'apprentissage soit acceptable. On dit alors que le système a convergé. La Figure II.6 illustre le principe d'apprentissage par correction d'erreur.

Pour minimiser la fonction d'erreur il faut calculer ses dérivés par rapport aux poids et aux biais. L'apprentissage est donc un procédé d'optimisation itératif basé sur le calcul du gradient de la fonction d'erreur.

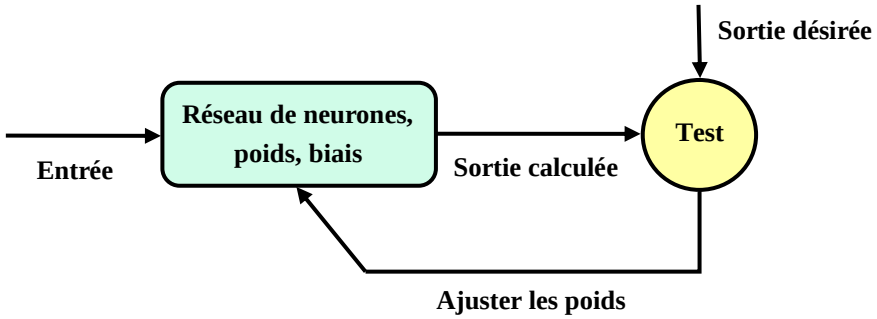

Figure II.6 Illustration du principe d'apprentissage supervisé.

II.5.2 Règle de Widrow-Hoff

Les premières règles d'apprentissage ont été établies pour un neurone linéaire par Widrow et Hoff en 1960 [Wid 1960]. Le schéma représentant un neurone linéaire à R entrées est donné par la Figure II.7.

La fonction réalisée, ici, par le neurone doit passer le plus près possible, au sens des moindres carrés, des points utilisés pour l'apprentissage. Il faut alors minimiser les écarts quadratiques.

La fonction de coût est donc une fonction minimum quadratique moyenne ou LMS (Least Mean Square) de l'erreur obtenue sur l'ensemble des K exemples, telle que décrite par l'équation suivante [Kro 1996]:

$$E = \frac{1}{K}\sum_{k=1}^{K} e^2(k) \qquad \text{(II.11)}$$

L'apprentissage selon Widrow et Hoff est un algorithme de descente du gradient tel que décrit par l'équation (II.10). Pour minimiser les erreurs quadratiques il faut calculer les dérivés de la fonction d'erreur par rapport aux poids et aux biais.

Pour évaluer le gradient de la fonction d'erreur il suffit d'évaluer le gradient de l'erreur relative à chaque exemple k et de faire ensuite la somme de tous les gradients [Kro 1996]:

$$\frac{\partial e^2(k)}{\partial w_{1,j}} = 2e(k)\frac{\partial e(k)}{\partial w_{1,j}} = 2e(k)\frac{\partial (y^d(k) - a(k))}{\partial w_{1,j}} \tag{II.12}$$

Dans le cas d'un neurone linéaire, la sortie $a(k)$ s'écrit:

$$a(k) = n(k) = \sum_{i=1}^{R} w_{1,i} \cdot p_i(k) + b \tag{II.13}$$

On remplace l'équation (II.13) dans l'équation (II.12), on obtient:

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial}{\partial w_{1,j}}\left[y^d(k) - \left(\sum_{i=1}^{R} w_{1,j} \cdot p_i(k) + b \right) \right] = -p_j(k) \tag{II.14a}$$

$$\frac{\partial e(k)}{\partial b} = \frac{\partial}{\partial b}\left[y^d(k) - \left(\sum_{i=1}^{R} w_{1,j} \cdot p_i(k) + b \right) \right] = -1 \tag{II.14b}$$

Finalement, les variations des poids et du biais s'écrivent:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\,\mu \cdot e(k) \cdot \mathbf{p}(k)^{\mathrm{T}} \tag{II.15a}$$

$$b(k+1) = b(k) + 2\,\mu \cdot e(k) \tag{II.15a}$$

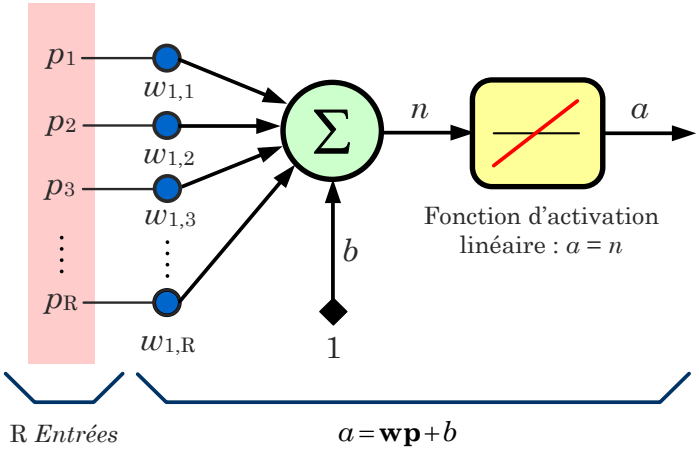

Figure II.7 Schéma représentant un neurone linéaire à R entrées.

Ces règles ont été étendues au cas d'une couche à S neurones linéaires. Les variations des poids et des biais seront exprimées sous la forme matricielle suivante:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\,\mu \cdot \mathbf{e}(k) \cdot \mathbf{p}(k)^{\mathrm{T}} \tag{II.16a}$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\,\mu \cdot \mathbf{e}(k) \tag{II.16a}$$

avec, le vecteur d'entrée $\mathbf{p} = [p_1, p_2, .., p_R]^{\mathrm{T}}$, la matrice des poids $\mathbf{W} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,R} \\ \vdots & \ddots & \vdots \\ w_{S,1} & \cdots & w_{S,R} \end{bmatrix}$, le vecteur de biais $\mathbf{b} = [b_1, b_2, .., b_S]^{\mathrm{T}}$, le vecteur d'erreur $\mathbf{e} = [e_1, e_2, .., e_S]^{\mathrm{T}}$ et μ est le taux d'apprentissage ou pas du gradient (learning rate), il est compris entre 0 et 1.

II.6 Méthode de rétro propagation du gradient

Nous avons vu que la modification des poids et des biais dépend du calcul du gradient de la fonction d'erreur. Le problème de calcul de ce gradient se pose dans le cas d'un réseau multicouche avec des fonctions d'activation non linéaires.

En effet, Comment faire varier les poids de chaque connexion en tenant compte de l'erreur qui n'a été mesurée que sur la couche de sortie? Ce problème a été identifié et a donné lieu à la mise au point dans les années 80 de l'algorithme de 'rétro propagation' [Rum 1986].

II.6.1 Le principe

L'algorithme de retro propagation du gradient est basé sur la généralisation de la règle d'apprentissage de Widrow et Hoff, appliquées aux réseaux multicouches à fonctions d'activation non linéaires mais différentiables. L'algorithme standard est un algorithme de descente du gradient. L'application répétée des règles de dérivation des fonctions composées permet de calculer le gradient de la fonction d'erreur.

Le nom retro propagation provient de la manière dont est calculé le gradient de la couche de sortie vers les entrées. Il existe des variantes de cet algorithme qui sont basées sur d'autres techniques d'optimisation comme la méthode du gradient conjugué et la méthode de Newton [Arb 2003].

II.6.2 La mise en œuvre

Pour illustrer la méthode de retro propagation du gradient, prenons le cas d'un réseau multicouches à une couche cachée (celui considéré dans notre travail), le cas d'un réseau de neurones à plusieurs couches cachées étant facilement transposable.

La première couche est définie par le vecteur d'entrée du réseau $\mathbf{p} = [p_1, p_2, .., p_R]^{\mathrm{T}}$.

La deuxième couche est la couche cachée elle contient H neurones, chaque neurone est caractérisé par un biais b^h, un potentiel n^h, une fonction d'activation f_h et une sortie a^h, l'écriture sous forme de vecteur donne: $\mathbf{b^h} = [b_1^h, b_2^h, .., b_H^h]^{\mathrm{T}}$, $\mathbf{n^h} = [n_1^h, n_2^h, .., n_H^h]^{\mathrm{T}}$ et $\mathbf{a^h} = [a_1^h, a_2^h, .., a_H^h]^{\mathrm{T}}$.

La dernière couche est la couche de sortie, elle contient S neurones, chaque neurone est caractérisé par un biais b^s , un potentiel n^s, une fonction d'activation f_s et une sortie a^s, c'est-à-dire que $\mathbf{b^s} = [b_1^s, b_2^s, .., b_S^s]^{\mathrm{T}}$, $\mathbf{n^s} = [n_1^s, n_2^s, .., n_S^s]^{\mathrm{T}}$ et $\mathbf{a^s} = [a_1^s, a_2^s, .., a_S^s]^{\mathrm{T}}$. La sortie désirée est quand à elle donnée par: $\mathbf{y^d} = [y_1^d, y_2^d, .., y_S^d]^{\mathrm{T}}$.

Généralement les fonctions d'activations au sein d'une même couche ont la même forme. Chaque entrée r est reliée à un neurone j de la couche cachée par une connexion de poids $w_{j,r}^h$. Tout neurone j de la couche cachée est reliée à un neurone i de la couche de sortie par une connexion de poids $w_{i,j}^s$. Le regroupement sous forme matricielle donne: $\mathbf{W^h} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,R} \\ \vdots & \ddots & \vdots \\ w_{H,1} & \cdots & w_{H,R} \end{bmatrix}$ et $\mathbf{W^s} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,H} \\ \vdots & \ddots & \vdots \\ w_{S,1} & \cdots & w_{S,H} \end{bmatrix}$.

La méthode de rétro propagation est basée sur la minimisation de l'erreur quadratique. Les poids et les biais sont modifiés dans le sens où la fonction de coût décroît le plus rapidement, donc dans le sens du gradient négatif.

L'erreur commise à la sortie du réseau sur un exemple k est calculée comme suit:

$$E(k) = \frac{1}{2} \cdot \sum_{i=1}^{S} \left[y_i^d(k) - a_i^s(k) \right]^2 \tag{II.17}$$

L'erreur moyenne lors du passage complet des K exemples s'écrit:

$$E = \frac{1}{K} \sum_{k=1}^{K} E(k) \tag{II.18}$$

La mise à jour d'un poids quelconque $w_{i,j}$ ou d'un biais b_i du réseau suit la règle d'apprentissage par correction d'erreur et est donnée par l'équation:

$$w_{i,j}(k+1) = w_{i,j}(k) - \mu \frac{\partial E(k)}{\partial w_{i,j}} \tag{II.19a}$$

$$b_i(k+1) = b_i(k) - \mu \frac{\partial E(k)}{\partial b_i} \tag{II.19b}$$

où, $\frac{\partial E(k)}{\partial w_{i,j}}$ est le gradient de l'erreur relativement au poids considéré.

Comme on le note, la méthode de rétro propagation du gradient n'est pas une règle d'apprentissage à part entière car elle est basée essentiellement sur la règle d'apprentissage par correction d'erreur. Son originalité tient, seulement, du fait quelle propose une méthode pour le calcul du gradient $\frac{\partial E(k)}{\partial w_{i,j}}$ [Dre 2005].

La mise à jour des poids (et des biais) du réseau peut se faire de deux manières. Le gradient est calculé et les poids sont modifiés après chaque présentation d'un exemple entrée/sortie k. C'est le mode incrémental. Ou bien alors, les poids et les biais du réseau sont calculés après passage de tous les exemples entrée/sortie (1 passe). Le gradient final utilisé est la moyenne de tous les gradients calculés à chaque exemple. C'est le mode d'apprentissage par lot (batch).

- **Calcul des paramètres de la couche de sortie**

On doit calculer pour tous les poids $w_{i,j}^s$ le gradient $\frac{\partial E(k)}{\partial w_{i,j}^s}$. Pour cela, on utilise la règle de dérivation des fonctions composées de la manière suivante:

$$\frac{\partial E(k)}{\partial w_{i,j}^s} = \frac{\partial E(k)}{\partial n_i^s} \cdot \frac{\partial n_i^s}{\partial w_{i,j}^s} \tag{II.20a}$$

$$\frac{\partial E(k)}{\partial b_i^s} = \frac{\partial E(k)}{\partial n_i^s} \cdot \frac{\partial n_i^s}{\partial b_i^s} \tag{II.20b}$$

Les termes $\frac{\partial n_i^s}{\partial w_{i,j}^s}$ et $\frac{\partial n_i^s}{\partial b_i^s}$ sont facilement calculés car le potentiel du neurone i dépend directement de ses poids et de son biais. On a:

$$\frac{\partial n_i^s}{\partial w_{i,j}^s} = \frac{\partial}{\partial w_{i,j}^s}\left\{\sum_{c=1}^{H}\left[w_{i,c}^s \cdot a_c^h\right] + b_i^s\right\} \tag{II.21}$$

Le deuxième terme de l'équation (II.21) est nul sauf pour l'élément $w_{i,j}^s$, on aura donc:

$$\frac{\partial n_i^s}{\partial w_{i,j}^s} = a_j^h \tag{II.22a}$$

$$\frac{\partial E(k)}{\partial b_i^s} = 1 \tag{II.22b}$$

En remplaçant les équations (II.22) dans les équations (II.20) on trouve:

$$\frac{\partial E(k)}{\partial w_{i,j}^s} = a_j^h \cdot \frac{\partial E(k)}{\partial n_i^s} \tag{II.23a}$$

$$\frac{\partial E(k)}{\partial b_i^s} = \frac{\partial E(k)}{\partial n_i^s} \tag{II.23b}$$

Le terme $\frac{\partial E(k)}{\partial n_i^s}$ est appelé 'sensibilité', il est calculé de la manière suivante:

$$\frac{\partial E(k)}{\partial n_i^s} = \frac{\partial E(k)}{\partial a_i^s} \cdot \frac{\partial a_i^s}{\partial n_i^s} \tag{II.24}$$

or, on sait que, $a_i^s = f_s(n_i^s)$, donc l'équation précédente devient:

$$\frac{\partial E(k)}{\partial n_i^s} = \frac{\partial E(k)}{\partial a_i^s} \cdot f_s{}'(n_i^s) \tag{II.25}$$

comme on considère ici la couche de sortie, on a:

$$\frac{\partial E(k)}{\partial a_i^s} = \frac{\partial}{\partial a_i^s}\left\{\frac{1}{2}\cdot\sum_{c=1}^{S}\left[y_c^d(k)-a_c^s(k)\right]^2\right\} = -(y_i^d(k)-a_i^s(k)) \tag{II.26}$$

donc l'équation (II.25) précédente devient:

$$\frac{\partial E(k)}{\partial n_i^s} = -\left[y_i^d(k)-a_i^s(k)\right]\cdot f_s{}'(n_i^s) \tag{II.27}$$

On peut finalement écrire la mise à jour des poids et des biais dans la couche de sortie de la manière suivante:

$$w_{i,j}^s(k+1) = w_{i,j}^s(k) + \mu\cdot a_j^h(k)\cdot\left[y_i^d(k)-a_i^s(k)\right]\cdot f_s{}'(n_i^s) \tag{II.28a}$$

$$b_i^s(k+1) = b_i^s(k) + \mu\cdot\left[y_i^d(k)-a_i^s(k)\right]\cdot f_s{}'(n_i^s) \tag{II.28b}$$

sachant que le terme $a_j^h = f_h\left(\sum_{r=1}^{R}\left[p_r\cdot w_{j,r}^h\right]+b_j\right)$.

- **Calcul des paramètres de la couche cachée**

Par analogie on peut réécrire directement les équations (II.23) de la manière suivante:

$$\frac{\partial E(k)}{\partial w_{j,r}^h} = p_r \cdot \frac{\partial E(k)}{\partial n_j^h} \tag{II.29a}$$

$$\frac{\partial E(k)}{\partial b_j^h} = \frac{\partial E(k)}{\partial n_j^h} \tag{II.29b}$$

Le terme de sensibilité $\frac{\partial E(k)}{\partial n_j^h}$ est calculé de la manière suivante:

$$\frac{\partial E(k)}{\partial n_j^h} = \frac{\partial E(k)}{\partial a_j^h} \cdot \frac{\partial a_j^h}{\partial n_j^h} \tag{II.30}$$

on sait que, $a_j^h = f_h(n_j^h)$, donc l'équation précédente devient:

$$\frac{\partial E(k)}{\partial n_j^h} = \frac{\partial E(k)}{\partial a_j^h} \cdot f_h{}'(n_j^h) \tag{II.31}$$

comme on considère ici une couche cachée et non la couche de sortie, on va écrire le terme $\frac{\partial E(k)}{\partial a_j^h}$ de la manière suivante [Kro 1996]:

$$\frac{\partial E(k)}{\partial a_j^h} = \sum_{i=1}^{S} \frac{\partial E(k)}{\partial n_i^s} \cdot \frac{\partial n_i^s}{\partial a_j^h} = \sum_{i=1}^{S} \left[\frac{\partial E(k)}{\partial n_i^s} \cdot \frac{\partial}{\partial a_j^h} \sum_{c=1}^{H} \left[w_{i,c}^s \cdot a_c^h \right] \right] = \sum_{i=1}^{S} \left[\frac{\partial E(k)}{\partial n_i^s} \cdot w_{i,j}^s \right] \tag{II.32}$$

le terme $\frac{\partial E(k)}{\partial n_i^s}$ étant calculé par l'équation (II.27), l'équation (II.31) devient:

$$\frac{\partial E(k)}{\partial n_j^h} = -f_h{}'(n_j^h) \cdot \sum_{i=1}^{S} \left[\left[y_i^d(k) - a_i^s(k) \right] \cdot f_s{}'(n_i^s) \cdot w_{i,j}^s \right] \tag{II.33}$$

On peut finalement écrire la mise à jour des poids et des biais dans la couche cachée de la manière suivante:

$$w_{j,r}^{h}(k+1) = w_{j,r}^{h}(k) + \mu \cdot p_r \cdot f_h^{'}(n_j^h) \cdot \sum_{i=1}^{S} \left[\left[y_i^d(k) - a_i^s(k) \right] \cdot f_s^{'}(n_i^s) \cdot w_{i,j}^s \right] \quad \text{(II.34a)}$$

$$b_j^h(k+1) = b_j^h(k) + \mu \cdot f_h^{'}(n_j^h) \cdot \sum_{i=1}^{S} \left[\left[y_i^d(k) - a_i^s(k) \right] \cdot f_s^{'}(n_i^s) \cdot w_{i,j}^s \right] \quad \text{(II.34b)}$$

Le réglage des paramètres des couches en amont dépend donc du réglage des paramètres des couches en aval. C'est-à-dire que le sens des calculs se fait de la couche de sortie vers la couche d'entrée en passant par les couches cachées. C'est de là que provient le terme retro propagation (back propagation). L'algorithme de cette méthode est donné en annexe B.

II.7 Critères d'arrêt et problèmes rencontrées lors de l'apprentissage

II.7.1 Critères d'arrêts d'un algorithme d'apprentissage

Il existe de nombreux critères qui permettent d'arrêter le processus d'apprentissage d'un réseau. On peut citer parmi ceux ci: la limitation du nombre maximum d'itérations, la fixation d'une erreur minimale requise, critère de validation croisée etc.

- **Fixer le nombre maximum de cycles d'entraînement**

L'entraînement d'un réseau multicouche par rétro propagation se fait par des présentations répétées des couples entrées/sorties de l'ensemble d'apprentissage. Chaque présentation est appelée cycle (epoch). Après chaque cycle le réseau apprend encore plus. Cependant un apprentissage trop répété risque de diminuer les performances du réseau en réduisant sa capacité à bien se comporter face à des données qu'il n'a pas encore rencontrées. On préfère donc fixer un nombre maximum de cycles pour limiter le temps de calcul et éviter le problème de sur-apprentissage.

- **Fixer une erreur minimale**

On peut fixer une borne inférieure sur l'erreur quadratique moyenne ou sur sa racine carrée. Lorsque l'erreur calculée diminue en dessous de cette borne, on arrête l'apprentissage et on considère que l'algorithme a convergé.

- **Le critère de validation croisée**

Pour appliquer ce critère, nous commençons tout d'abord par diviser l'espace des exemples en deux parties, l'un pour l'apprentissage et l'autre pour la validation. L'ensemble de validation sera utilisé pour le calcul de l'indice de performance. La technique consiste à arrêter l'apprentissage lorsque l'indice de performance calculé sur les données de validation cesse de s'améliorer ou commence à décroitre.

Cette méthode est très utile lorsque les données sont entachées de bruit ou si l'on veut que le réseau garde toute sa capacité de généralisation, elle permet d'éviter le problème de sur-apprentissage. Nous avons considéré cette méthode dans notre travail, plus de détails sur son utilité et utilisation sont donnés au Chapitre IV.

II.7.2 Différents problèmes rencontrés lors de l'apprentissage

Nous citons ci-après quelques problèmes rencontrés lors de l'apprentissage des RNA.

- **Le sur-apprentissage et le problème de richesse des données**

Comme nous l'avons vu précédemment le problème de sur-apprentissage apparait lorsque le réseau est entrainé de manière trop répété avec le même ensemble d'apprentissage ou avec uniquement des données trop proches. Ceux-ci ne possédant pas la richesse nécessaire, le réseau n'arrivera pas à recréer toutes les caractéristiques du système.

De ce fait même si le réseau arrive à minimiser l'erreur sur l'ensemble d'apprentissage grâce à un nombre d'itérations élevé, il verra quand même ses performances décroitre lorsqu'on lui présente des données différentes de celle de l'apprentissage ou des données teintées de bruit. Le réseau perd ainsi sa capacité de généralisation.

- **Le phénomène de saturation**

Lorsqu'on utilise des fonctions d'activation non linéaires telles que les fonctions sigmoïdes dans la couche de sortie, il arrive que le processus de convergence devienne tellement lent que le réseau cesse d'apprendre. Ceci est du au fait que les valeurs de la fonction, calculées par le réseau, se trouvent coincées dans un intervalle où la dérivée est pratiquement nulle. Or, la mise à jour des poids dans un algorithme d'apprentissage, comme celui de retro propagation, dépend essentiellement du calcul de la dérivée des fonctions d'activation.

Pour éviter ce type de problème, il est recommandé de normaliser les entrées d'un réseau. Ceci consiste à la transformation linéaire des couples entrée/sortie en des valeurs comprises entre [0,1] ou [-1,1] de façon à rester dans l'intervalle de 'courbure' de la fonction d'activation. En effet sur cet, intervalle on dit que la fonction garde ses propriétés de 'discrimination'.

- **Le phénomène des minima locaux**

La méthode de rétro propagation est une technique de descente de gradient qui doit converger vers le minimum de la surface que constitue la fonction d'erreur. Elle n'offre, par conséquent, aucune garantie quant à la convergence de l'algorithme.

La surface de la fonction d'erreur d'un réseau à fonction d'activation non linéaire est plus complexe que la surface d'un réseau linéaire. Elle peut avoir plusieurs minima locaux. Il peut arriver que le processus de minimisation de la fonction d'erreur tombe dans un minimum local sans pouvoir en sortir, surtout si le taux d'apprentissage est petit.

En effet, si le taux d'apprentissage est trop petit, la variation des poids risque de ne pas être suffisamment importante pour que l'algorithme sorte de ce minimum local.

D'un autre coté choisir, un taux d'apprentissage élevé risque de faire en sorte que l'algorithme finit par osciller autour du point terminal vers lequel il doit converger sans jamais l'atteindre. Ceci est du au fait que le pas du gradient est trop grand. Le réglage du taux d'apprentissage doit se faire donc au cas par cas et en fonction des résultats obtenus.

Notons enfin, que certaines techniques utilise un pas de gradient variable, ceci a pour conséquence aussi d'augmenter la rapidité de convergence de l'algorithme.

Une astuce, simple, pour réduire le problème des minima locaux est la suivante. A chaque cycle d'entraînement, il est recommandé de permuter ou de générer aléatoirement l'ordre de présentation des exemples pour éviter de tomber dans le même minimum local que lors du cycle précédent. Changer l'ordre de présentation des données lors de l'apprentissage permet de générer des trajectoires de minimisation différentes et augmente les chances de tomber sur le minimum global recherché.

II.8 Conclusion

Nous avons présenté dans ce chapitre un bref rappel sur les réseaux de neurones artificiels. Le but de ce Chapitre n'est pas d'être exhaustif mais seulement de présenter la configuration des réseaux de neurones ainsi que les méthodes d'apprentissage que nous avons utilisé dans notre travail. Les réseaux de neurones artificiels sont très utilisés en contrôle de processus et robotique pour la modélisation de processus.

Cette modélisation de type 'boite noire' trouve tout son intérêt lorsque le système considéré est trop complexe pour être déterminé par un modèle de connaissances. De plus, l'utilisation d'un modèle de connaissances complexe dans la commande augmente énormément son temps de calcul et ainsi son temps de réponse et risque de réduire ses performances, tel que nous le montrons plus loin dans cette thèse.

Pour notre part nous avons utilisé des réseaux des neurones artificiels de type MLP à apprentissage supervisé au Chapitre IV, dans le but d'approximer les fonctions non linéaires utilisées dans notre commande et dans l'élaboration d'un contrôleur neuronal adaptatif.

Avant d'introduire cette commande, nous présentons d'abord au chapitre suivant, quelques généralités sur la commande des robots flexibles.

Chapitre III

Commandes classiques de robots flexibles

III.1 Introduction

La commande des robots flexibles représente un challenge important pour de nombreux chercheurs. L'allégement des structures mécaniques et les performances demandées amènent à considérer les effets de la flexibilité dans la synthèse de lois de commandes performantes. Certains domaines sont particulièrement intéressés par ces problèmes de commande de structures flexibles : les grandes structures mécaniques (par exemple des plates-formes de forage), la télé-robotique spatiale (bras manipulateur des navettes spatiales etc.), et même la robotique 'classique' avec les structures allégées, des joints flexibles, des performances accrues avec excitation des modes naturels de vibration de tout élément mécanique etc.

Commander un robot flexible est une tâche ardue du fait que ce dernier présente des caractéristiques qui le distinguent du robot rigide classique. En effet, un robot manipulateur à bras flexibles subit une variation importante de la dynamique en fonction de sa configuration et de la charge transportée. De plus, c'est un système sous actionné car il possède plus de sorties à contrôler que d'entrées de commandes. D'un point de vue mathématique, c'est un système qui est régi par des équations aux dérivées partielles d'ordre infini. Enfin, l'ajout de variables dues à l'élasticité des bras engendre des équations de mouvement plus complexes surtout si le robot possède plusieurs degrés de liberté.

Cependant, la plupart des techniques de commande de robots flexibles sont inspirées des commandes classiques de robots rigides. Une étude bibliographique nous permet de relever quelques techniques importantes dans ce domaine.

Une stratégie de commande à plusieurs étages a été utilisée par [Boo 1975, Hil 1991, Ush 1991, Lin 1992, Kho 1995, Aza 2003, Moh 2005], elle consiste à superposer à la commande du corps rigide les techniques de 'shaping' ou la correction des effets élastiques.

D'autres travaux utilisent les techniques de découplage [Des 1988, Che 1989], d'autres encore sont basés sur la méthode de la perturbation singulière [Sic 1988, Spo 1995, Par 2002] ou utilisent le contrôle par retour d'état [Ryu 2004] ainsi que les commandes non linéaires adaptatives [Mao 2009a].

Les commandes basées sur la logique floue et les réseaux de neurones artificiels ont aussi été utilisées car ils permettent une réduction de la complexité et un calcul plus rapide de la loi de commande ce qui constitue un point critique pour le suivi de trajectoires dynamiques en temps réel [Lew 1995, Kuo 2001, Che 2003, Tia 2004, Tan 2006, Mao 2007, Mao 2008b, Mao 2008a, Mao 2009c, Mao 2009b, Mao 2010b, Mao 2010a].

Avec le récent développement des technologies des capteurs et actionneurs, plusieurs chercheurs se sont tournés vers les méthodes de contrôle par suppression des vibrations des structures flexibles en utilisant des matériaux 'intelligents' comme les alliages à mémoire de forme 'Shape Memory Alloys' (SMA) [Ela 2001], les matériaux magnéto-rhéologiques 'Magnetorheological (MR) materials' [Giu 2001], les matériaux électro-rhéologiques 'Electrorheological (ER) materials' [Len 1999] et les transducteurs piézo-électriques 'Piezoelectric transducers (PZT)' [Shi 2001, Sun 2004, Sha 2005] etc.

D'une manière générale on peut distinguer deux types d'approches pour l'élaboration de la loi de commande. Les commandes basées sur le modèle de connaissances (type boite blanche) telles que les commandes linéaires et non linéaires et les commandes basées sur un modèle statistique (type boite noire) telles que les commandes à base de réseaux de neurones.

Les commandes basées sur le modèle de connaissances présentent l'avantage d'utiliser les informations physiques ainsi que les lois de la dynamique pour l'élaboration de commandes stables et efficaces. Cependant leurs performances sont souvent tributaires de l'exactitude du modèle utilisé ainsi que de la bonne précision avec laquelle les paramètres physiques du système sont mesurés.

Ceci conduit souvent à l'élaboration de commandes soit incompatibles avec une dynamique rapide de la trajectoire à suivre, à cause d'un modèle trop complexe et donc une réactivité limitée de la commande, soit à une commande imprécise à cause de l'allégement effectué sur le modèle [Kur 2005, Gu 2005, Hu 2007, Mik 2007].

Les commandes basées sur le modèle statistique présentent l'avantage d'êtres rapides et ne nécessitent pas une connaissance des lois physiques qui régissent le système à contrôler.

Cependant elle nécessite généralement, comme dans le cas des réseaux de neurones, un entrainement hors ligne avant de pouvoir être utilisées. Leurs performances sont tributaires de la base de données et de la méthode utilisée pour l'entrainement.

De plus, la configuration du modèle utilisé n'est pas unique et plusieurs essais sont nécessaires pour trouver la structure du modèle qui donnera de bonnes performances. Le problème le plus grave rencontré avec ce type de commande provient du faite que si le système évolue en dehors des plages de variations des entrées/sorties avec lesquelles a été effectué l'entrainement, la réaction de la commande sera alors imprévisible [Bis 1995, Gup 2003].

Nous présentons dans ce chapitre quelques commandes classiques, utilisant les modèles à base de connaissances. Des commandes originales utilisant un modèle à base de connaissances associé à un modèle statistique seront présentées au chapitre suivant et représenteront notre contribution personnelle au domaine de la commande de robots flexibles.

Nous présentons aussi à la fin de ce chapitre quelques définitions et théorèmes sur la stabilité des systèmes non linéaires.

III.2 Approches basées sur le modèle de connaissances

On peut classer les commandes utilisant le modèle à base de connaissances en deux catégories: les commandes linéaires et les commandes non linéaires.

Les commandes linéaires sont basées sur une linéarisation des équations de mouvement autour d'un point de fonctionnement ils ne peuvent donc conduire qu'à des performances locales.

L'application de ces méthodes à des mouvements de grandes vitesses et amplitudes et à des robots à plusieurs degrés de liberté est inadaptée. En effet, les non-linéarités et les effets de couplage entre les bras deviennent alors importants et néfastes à la commande [Tze 1990].

Les commandes non linéaires sont basées quand à elles sur le modèle non linéaire du robot garantissant une applicabilité générale et des performances homogènes en toutes circonstances. On peut considérer ici le modèle complet ou avec simplification de quelques termes si le modèle est trop complexe pour être utilisé dans la commande.

Dans ce cas, l'utilisation de commandes non linéaires adaptatives parait être un choix judicieux pour compenser les erreurs de modélisation, les variations paramétriques ou les perturbations extérieures [Slo 1987, Pri 1999, Mik 2007, Mao 2009a].

III.3 Commandes linéaires

Pour utiliser les commandes linéaires avec notre robot flexible nous devons tout d'abord linéariser le modèle dynamique autour d'un point de fonctionnement. Nous présenterons ensuite deux types de commandes linéaires fréquemment utilisées en robotique classique.

La première est la commande par retour d'état qui est basée sur la technique de placement de pôles. La seconde est la commande linéaire quadratique (LQ) qui est basée quant à elle sur l'optimisation d'un critère de coût.

Nous citerons les avantages et les inconvénients de chacune des deux méthodes et nous conclurons quant à la nécessité d'utiliser des commandes plus performantes comme les commandes non-linéaires.

III.3.1 Linéarisation du modèle

Réécrivons l'équation (I.5) du Chapitre 1, décrivant le modèle dynamique du robot flexible, sous la forme condensée suivante:

$$\mathbf{L_r}\,\mathbf{\Gamma} = \mathbf{A(q)}\,\ddot{\mathbf{q}} + \mathbf{H(q,\dot{q})} \tag{III.1}$$

avec,

$$\mathbf{H(q,\dot{q})} = \mathbf{B(q)}\left[\dot{\mathbf{q}}\dot{\mathbf{q}}\right] + \mathbf{C(q)}\left[\dot{\mathbf{q}}^2\right] + \mathbf{Q(q)} + \mathbf{K}\,\mathbf{q} \tag{III.2}$$

On peut linéariser notre modèle autour d'un point de fonctionnement en différentiant l'équation (III.1):

$$\Delta\ddot{\mathbf{q}} = -\,\mathbf{A}^{-1}\left(\frac{\partial \mathbf{H}}{\partial \mathbf{q}} + \frac{\partial \mathbf{A}}{\partial \mathbf{q}}\ddot{\mathbf{q}}\right)\Delta\mathbf{q} - \mathbf{A}^{-1}\frac{\partial \mathbf{H}}{\partial \dot{\mathbf{q}}}\,\Delta\dot{\mathbf{q}} + \mathbf{A}^{-1}\,\mathbf{L_r}\,\Delta\mathbf{\Gamma} \tag{III.3}$$

où, $\Delta\mathbf{q} = (\mathbf{q} - \mathbf{q_0})$ avec $\mathbf{q_0}$ la valeur de $\mathbf{q}$ au point de fonctionnement.

On déduit de cette équation la représentation sous forme d'état suivante:

$$\begin{bmatrix} \Delta\dot{\mathbf{q}} \\ \Delta\ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n,n} & \mathbf{I}_{n,n} \\ -\mathbf{A}^{-1}\left(\frac{\partial \mathbf{H}}{\partial \mathbf{q}} + \frac{\partial \mathbf{A}}{\partial \mathbf{q}}\ddot{\mathbf{q}}\right) & -\mathbf{A}^{-1}\frac{\partial \mathbf{H}}{\partial \dot{\mathbf{q}}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{q} \\ \Delta\dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n,n_r} \\ \mathbf{A}^{-1}\,\mathbf{L_r} \end{bmatrix} \Delta\mathbf{\Gamma} \tag{III.4}$$

tel que, $\mathbf{0}_{n,n}$, $\mathbf{0}_{n,n_r}$ sont des matrices de zéros de dimension nxn et nxn_r, respectivement. $\mathbf{I}_{n,n}$ représente la matrice identité de dimension nxn.

Rappelons que :

- n est le nombre de variables au totale = n_r (nombre de variables rigides) + n_e (nombre de variables élastiques) = 2 + 4 = 6 pour le cas du robot à deux bras flexibles,
- $\mathbf{L_r}$ est un vecteur de dimension n dont les n_r premiers éléments sont 1 et les n_e suivants 0.

Si on choisit le point de fonctionnement comme étant un point d'équilibre tel que $\ddot{\mathbf{q}} = \dot{\mathbf{q}} = \mathbf{0}$ et $\mathbf{q_e} = \mathbf{0}$, on obtient finalement:

$$\dot{\boldsymbol{X}} = \boldsymbol{A}\,\boldsymbol{X} + \boldsymbol{B}\,\Delta\boldsymbol{\Gamma} \tag{III.5}$$

où,

- le vecteur d'état est $\boldsymbol{X} = \begin{bmatrix} \Delta\mathbf{q} \\ \Delta\dot{\mathbf{q}} \end{bmatrix}$,
- la matrice d'évolution du système est $\boldsymbol{A} = \begin{bmatrix} \mathbf{0}_{n,n} & \mathbf{I}_{n,n} \\ -\mathbf{A}^{-1}\dfrac{\partial \mathbf{H}(\mathbf{q},\mathbf{0})}{\partial \mathbf{q}} & -\mathbf{A}^{-1}\dfrac{\partial \mathbf{H}(\mathbf{q},\mathbf{0})}{\partial \dot{\mathbf{q}}} \end{bmatrix}$,
- et la matrice d'application de la commande est $\boldsymbol{B} = \begin{bmatrix} \mathbf{0}_{n,n_r} \\ \mathbf{A}^{-1}\mathbf{L_r} \end{bmatrix}$.

Appliquons les résultats obtenus à notre robot. Dans le cas du robot à un bras flexibles nous avons (voir Annexe A.2.1):

- la matrice d'inertie **A** est constante au premier ordre.
- $\dfrac{\partial \mathbf{H}(\mathbf{q},\mathbf{0})}{\partial \mathbf{q}} = \mathbf{K}$ (matrice de raideur) et $\dfrac{\partial \mathbf{H}(\mathbf{q},\mathbf{0})}{\partial \dot{\mathbf{q}}} = \mathbf{0}$.

Ce qui conduit à un système linéaire invariant décrit par la représentation d'état suivante:

$$\begin{bmatrix} \Delta\dot{\mathbf{q}} \\ \Delta\ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n,n} & \mathbf{I}_{n,n} \\ -\mathbf{A}^{-1}\mathbf{K} & \mathbf{0}_{n,n} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{q} \\ \Delta\dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n,n_r} \\ \mathbf{A}^{-1}\mathbf{L_r} \end{bmatrix} \Delta\boldsymbol{\Gamma} \tag{III.6}$$

Dans le cas du robot à deux bras flexibles nous devons aussi annuler les termes de couplages dans la matrice **A** (notamment les termes S_{24} et C_{24} en Annexe A.2.2), nous obtenons alors un système linéaire invariant tel que décrit par l'équation (III.6). On a vérifié ensuite la commandabilité du système en utilisant le théorème de Kalman.

Disposant d'un modèle linéarisé, toutes les solutions classiques de commandes linéaires (approches fréquentielles ou temporelles) sont utilisables.

Nous présentons succinctement dans ce qui suit deux commandes linéaires très utilisées en robotique: la commande par retour d'état et la commande linéaire quadratique. Des résultats détaillés ainsi que des tests en simulation de ces deux commandes appliquées au robot flexible peuvent êtres retrouvé dans notre thèse de Magister [Mao 1999].

III.3.2 Commande par retour d'état (placement de pôles)

Contrairement au correcteur classique qui utilise une représentation par fonction de transfert, la commande par placement de pôles ainsi que la commande linéaire quadratique utilisent la représentation d'état pour modéliser le système. Dans ce type de commande le système utilise toutes les informations nécessaires, c'est à dire ses variables d'état, pour se corriger. Ceci est en opposition avec les correcteurs classiques où seules les sorties sont 'rebouclées'.

Cette technique utilise le fait que les caractéristiques de la réponse et donc les performances du système dépendent de la position des pôles. Donc pour améliorer la stabilité et les performances du système on va considérer le placement des pôles comme un paramètre réglable du système en faisant varier un gain **k** dans la commande.

La commande par retour d'état exige une condition moins forte que la contrôlabilité, et il suffit que le système soit stabilisable c'est à dire qu'il suffit que les pôles instables seulement soient contrôlables pour que la méthode converge [Sha 1993].

Le modèle linéarisé de notre système peut être décrit par la représentation d'état suivante:

$$\dot{\mathbf{X}} = \boldsymbol{A}\,\mathbf{X} + \boldsymbol{B}\,\mathbf{U}$$

$$\mathbf{Y} = \boldsymbol{C}\,\mathbf{X} \qquad \text{(III.7)}$$

avec, $\mathbf{U} = \Delta\boldsymbol{\Gamma}$ vecteur de commande et $\boldsymbol{C}$ est un vecteur de dimension $2n$ dont les n_r premiers éléments sont à 1 et les autres sont à 0, c'est à dire qu'on s'intéresse aux positions articulaires en sortie.

Afin de stabiliser le système en boucle fermée ou augmenter sa stabilité (marge de gain et marge de phase), il est nécessaire d'avoir des pôles à partie réelle négative. Supposons que l'on veuille obtenir les pôles suivants: $p_1, p_2, \ldots, p_n$.

Alors l'équation caractéristique correspondant à ces pôles s'écrit [Sha 1993]:

$$\Delta_d(p) = p^n + \lambda_1 p^{n-1} + \ldots + \lambda_n \qquad \text{(III.8)}$$

où, $\lambda_1 \ldots \lambda_n$ sont les coefficients de l'équation caractéristique du système.

Le calcul du système à retour d'état en boucle fermée est obtenu en introduisant une entrée de consigne **E** (voir la Figure III.1), ce qui donne:

$$\mathbf{U} = -\mathbf{k}\,\mathbf{X} + \bar{\mathbf{N}}\mathbf{E} \qquad \text{(III.9)}$$

Le gain constant $\bar{\mathbf{N}}$ peut être facilement calculé de façon à produire une erreur statique nulle pour des entrées de consigne constantes.

Pour calculer le gain de retour d'état **k**, plusieurs méthodes existent. On peut citer parmi celles-ci, la méthode d'Ackermann pour les systèmes mono-entrée mono-sortie qui aboutit à la relation suivante [Par 2001]:

$$\mathbf{k} = [0\ 0\ \ldots\ 1]\ \mathbf{Cont}^{-1}\,\Delta_d(\boldsymbol{A}) \qquad \text{(III.10)}$$

où,

- $\Delta_d(\boldsymbol{A}) = \boldsymbol{A}^n + \lambda_1 \boldsymbol{A}^{n-1} + \ldots + \lambda_{n-1} \boldsymbol{A} + \lambda_n \mathbf{I}_{n,n}$ avec $\mathbf{I}_{n,n}$ matrice (nxn) d'identité,
- **Cont** est la matrice de contrôlabilité de rang n du system en boucle ouverte définie par: $[\boldsymbol{B}\ \boldsymbol{AB}\ \boldsymbol{A}^2\boldsymbol{B} \ldots \boldsymbol{A}^n\boldsymbol{B}]$.

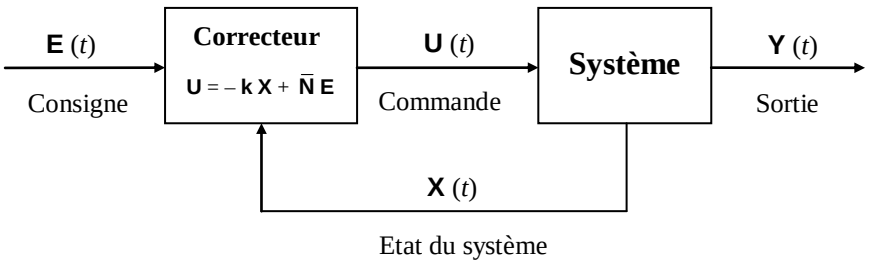

Figure III.1 Commande par retour d'état.

On aboutit finalement aux équations qui décrivent le système corrigé en boucle fermée:

$$\dot{\mathbf{X}} = (\boldsymbol{A} - \boldsymbol{B}\,\mathbf{k})\,\mathbf{X} + \boldsymbol{B}\bar{\mathbf{N}}\mathbf{E}$$

$$\mathbf{Y} = \boldsymbol{C}\,\mathbf{X} \qquad \text{(III.11)}$$

La commande par placement de pôles est une technique efficace puisque en plus de stabiliser le système, elle permet, par un choix judicieux des pôles, de modifier et d'améliorer la réponse temporelle de notre système [Mao 1999].

Cependant la principale difficulté de cette méthode réside justement dans la façon de choisir ces pôles. Un choix optimal permettrait d'avoir par exemple un temps de réponse plus court avec un dépassement moins élevé. On peut donc améliorer les performances de notre système par un meilleur placement de pôles.

Bien que le choix d'avoir un système rapide avec un petit dépassement soit délicat à mettre en œuvre du fait que ces deux caractéristiques soient en général antagonistes, il existe cependant des méthodes d'optimisation qui permettent de trouver le placement de pôles adéquat.

Nous présentons succinctement dans ce qui suit la méthode linéaire quadratique qui répondra à ces exigences.

III.3.3 Commande linéaire quadratique

Historiquement, le développement de la commande linéaire quadratique (LQ) a commencé dans les années 1960 en parallèle avec les programmes de recherche spatiale. L'application de cette méthode à la commande de structures flexibles a fait l'objet de nombreuses recherches, entre autres [Bal 1978, Sch 1984, Pha 1992, Mao 1999].

La commande LQ appartient à la puissante famille des commandes optimales et elle vise à minimiser un critère quadratique (fonction de coût) qui dépend de l'état du système et de l'amplitude de la commande appliquée.

Soit le système linéaire, contrôlable et invariant dans le temps, décrit par l'équation (III.7). Considérons la fonction de coût suivante [Han 2004]:

$$\mathbf{J} = \frac{1}{2}\int_0^{\infty} (\mathbf{X}^{\mathrm{T}}\mathbf{Q}\,\mathbf{X} + \mathbf{U}^{\mathrm{T}}\mathbf{R}\mathbf{U})\,\mathrm{d}t \qquad \text{(III.12)}$$

avec, **Q** matrice de pondération réelle symétrique semi-définie positive ($\mathbf{Q} \geq 0$) et **R** matrice de pondération réelle symétrique définie positive ($\mathbf{R} > 0$).

Le problème est de minimiser **J** en respectant l'entrée de commande **U**(t). Une interprétation simple de la fonction de coût peut être faite en considérant un système scalaire (du premier ordre); la fonction **J** devient:

$$\mathbf{J} = \frac{1}{2}\int_0^{\infty} (\mathrm{Q}\,\mathbf{X}^2 + \mathrm{R}\,\mathbf{U}^2)\,\mathrm{d}t \qquad \text{(III.13)}$$

Il apparaît bien ainsi que **J** représente la somme pondérée de l'énergie de l'état et de la commande. Si R est très grand devant Q, l'énergie de commande est grandement pénalisée. Ceci se traduit physiquement par de plus petits moteurs, actionneurs et gain d'amplifications nécessaires pour 'implémenter' la loi de commande. De même si Q est très grand devant R, l'état sera grandement pénalisé. Ceci conduit à un système très amorti et donc évite d'avoir de larges fluctuations et de grands dépassements au niveau des variables d'état.

La commande optimale **U** qui permet de minimiser la fonction **J** dans (III.12) se présente alors sous la forme simple [Mik 2007]:

$$\mathbf{U} = -(\mathbf{R}^{-1}\,\boldsymbol{B}^{\mathrm{T}}\,\mathbf{P})\,\mathbf{X} \qquad \text{(III.14)}$$

où **P** est une solution symétrique semi-définie positive de l'équation algébrique de Ricatti:

$$\dot{\mathbf{P}} = \mathbf{P}\,\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}}\mathbf{P} - \mathbf{P}\,\boldsymbol{B}\,\mathbf{R}^{-1}\,\boldsymbol{B}^{\mathrm{T}}\mathbf{P} + \mathbf{Q} = 0 \qquad \text{(III.15)}$$

La commande linéaire quadratique est, par sa nature, adaptée au système à plusieurs entrées et sorties car on cherche, ici, à minimiser un critère de performance global tenant en compte toutes les entrées et sorties du système.

La commande LQ permet d'assurer une bonne stabilité du système ainsi que des performances optimales [Mao 1999]. De plus, le calcul de cette commande est facilement programmable sur ordinateur. La seule difficulté dans l'emploi de cette méthode réside dans le choix des matrices de pondération **Q** et **R**.

D'une manière générale les deux types de commande précédemment décrites, s'adaptent bien au problème de régulation où le correcteur essaie de maintenir une sortie constante face à des perturbations internes ou externes. Notre modèle linéarisé s'inscrit dans ce contexte puisque l'on travaille autour d'un point de fonctionnement. Il ne faut donc pas qu'il y ait de grandes variations de la sortie pour rester dans le domaine de validité du modèle. Or ceci est rarement le cas en robotique où les problèmes sont souvent du type poursuite de trajectoire avec des consignes qui peuvent varier d'une façon importante au cours du temps. L'inconvénient majeur de ces méthodes incombe donc plus au modèle utilisé qu'aux méthodes elles mêmes.

L'autre inconvénient du modèle linéarisé est qu'il induit une perte significative d'informations qui peut même aller jusqu'à déstabiliser le modèle obtenu, alors que le système non-linéaire original était lui parfaitement stable. Un exemple bien connu de phénomènes non-linéaires qui sont souvent négligés lors de la modélisation est l'effet des frottements. Les frottements peuvent jouer un rôle très important dans la stabilité des systèmes physiques. Ils interviennent comme amortisseurs en consommant à chaque mouvement une partie de l'énergie du système. Les éléments non-linéaires compliquent souvent les équations mathématiques mais en les négligeant on risque de rendre le modèle non-conforme avec la réalité du système.

Afin d'obtenir en simulation un comportement du robot le plus fidèle possible et le plus proche de la réalité nous nous devons donc d'utiliser le modèle non-linéaire. Nous avons par conséquent utilisé dans notre travail de thèse, des commandes adaptées à ce type de modèle. Quelques généralités sur les commandes non-linéaires sont présentées dans le paragraphe suivant. Au Chapitre IV, nous détaillerons les commandes que nous avons élaborées pour un contrôle efficace des robots flexibles.

III.4 Commandes non linéaires

Nous nous intéresserons particulièrement, ici, aux commandes conçues à partir du modèle mathématique du système considéré. Cependant il existe d'autres types de commandes, comme les commandes 'floues' basées sur un raisonnement logique et proche du langage humain ou bien encore les commandes 'neuronales' basées, quand à elles, sur une étude statistique de la réponse du système. Nous avons associé la commande non-linéaire à la commande neuronale pour former un nouveau type de commande hybride que nous présenterons dans le Chapitre IV.

Parmi les commandes non-linéaires classiques, deux catégories peuvent être distinguées. La première concerne les commandes dynamiques, la seconde inclut les commandes issues de la théorie des systèmes linéaires telle que la commande H∞, les commandes dites à 'structure variable' telle que la commande par mode de glissement etc. Cette dernière est basée sur une logique de commutation: la loi de commande est discontinue et change de structure en atteignant un ensemble de surfaces de commutation prédéterminées dans l'espace d'état.

Nous nous sommes intéressés dans notre travail aux commandes dynamiques pour les raisons suivantes:

- ✓ ces lois de commande sont particulièrement bien adaptées lorsque l'on souhaite poursuivre une trajectoire avec une vitesse élevée et une bonne précision dynamique. Elles prennent en compte la dynamique fortement non linéaire et couplée du système. Elles offrent ainsi des résultats de convergence globale des signaux d'erreur vers zéro et une réponse uniforme indépendamment de la zone de travail atteignable du robot. Ceci n'est notamment pas le cas des commandes linéaires présentées précédemment et qui ne sont valables qu'autour d'un point de fonctionnement,
- ✓ le phénomène de flexion et par conséquent les variables élastiques doivent être pris en compte au niveau de la loi de commande, afin de les amortir rapidement et assurer une bonne stabilité au système,
- ✓ une analyse théorique de la stabilité de ces commandes peut être menée. Les lois de commandes sont en effet synthétisées à partir d'une démonstration de convergence des différents signaux d'erreurs.

Suivant que les paramètres de la commande soient constants ou variables les commandes non-linéaires dynamiques peuvent se différencier en deux catégories. Les commandes à paramètres fixes tout au long du mouvement et les commandes adaptatives où la structure de la commande reste inchangée mais dans laquelle des lois d'adaptation viennent faire varier les paramètres du modèle de façon à prendre en compte l'évolution de la dynamique du système.

La commande à paramètres fixes est la plus simple. Toutefois elle ne peut s'envisager seule dans la mesure où il existe des non-linéarités qui sont difficilement modélisables. C'est notamment notre cas en ce qui concerne les frottements par exemple. Il est par conséquent essentiel de considérer une commande adaptative qui compensera ces non-linéarités, les erreurs de modélisation, la dérive des paramètres etc.

La commande adaptative permet d'ajuster la réponse du système en fonction des variations volontaires ou non d'inertie ou de charge au cours du mouvement. Elle permet aussi de compenser jusqu'à un certain point les phénomènes physiques difficilement modélisables ou non considérés lors de l'étape de modélisation.

Parmi les commandes adaptatives, on distingue deux classes: les commandes adaptatives indirectes et les commandes adaptatives directes [Gui 1995]:

a) Les commandes adaptatives indirectes:

Les commandes adaptatives indirectes ou commandes 'auto-ajustables' sont des commandes à structure fixe dans lesquelles les paramètres du système sont identifiés en ligne à partir des données entrées/sorties du processus puis réinjectés vers un bloc de synthèse de commande. La structure de ce type de commande est représentée par la Figure III.2.

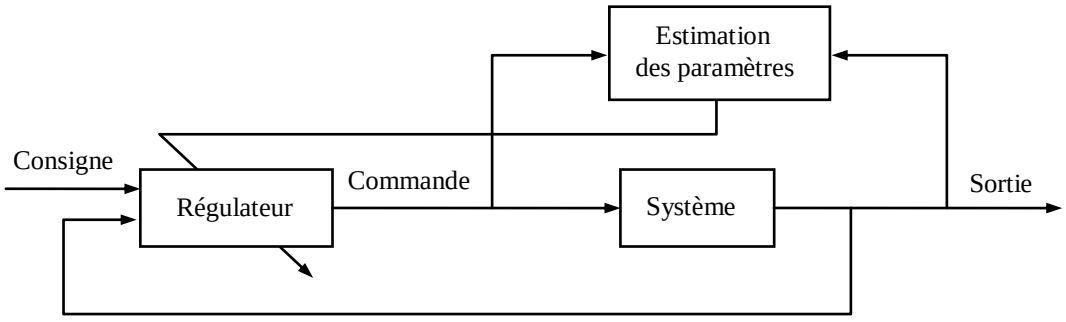

Figure III.2 Structure de la commande adaptative indirecte.

L'inconvénient majeur de ces commandes est que les paramètres doivent converger vers leurs vraies valeurs pour que la stabilité en boucle fermée soit assurée. Ce qui implique que les trajectoires désirées doivent être suffisamment riches, or ceci est souvent en désaccord avec la tâche à réaliser.

De plus cette approche nécessitant un modèle le plus exact possible, elle ne convient pas en pratique pour la commande de systèmes fortement non-linéaires et ou le modèle utilisé pour la commande est une approximation ou une réduction du modèle complet du système.

Or l'utilisation du modèle complet dans la commande, dans le cas de ces systèmes, engendre une lois de commande complexe et trop lourde pour être compatible avec les contraintes de la commande en temps réel tel que le suivi de trajectoire dynamique rapide.

b) Commandes adaptatives directes :

La structure générale des commandes adaptatives directes ou commandes par modèle de référence est donnée par la Figure III.3. Il s'agit de lois de commandes intégrant tout ou une partie du modèle et estimant en ligne les paramètres susceptibles de varier au cours du mouvement.

Les lois d'adaptation sont déterminées non pas de façon à ce que les paramètres convergent vers leur vraie valeur mais de façon à ce que le système global soit stable selon un critère de minimisation de l'énergie.

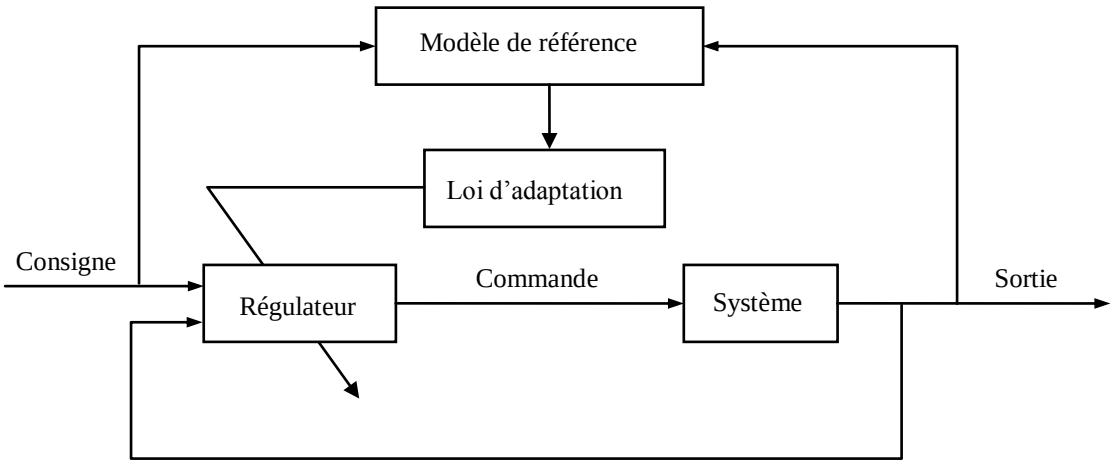

Figure III.3 Structure de la commande adaptative directe.

Avant d'approfondir, dans le chapitre suivant, les commandes non-linéaires utilisées, il est essentiel de poser un certain nombre de définitions et théorèmes sur lesquels est basée la théorie des systèmes non linéaires ainsi que les propriétés propres à notre système.

Nous présentons en annexe C, quelques définitions et théorèmes sur la stabilité des systèmes non linéaires. En annexe D, nous relèverons les propriétés du modèle dynamique de notre robot flexible. Dans le Chapitre suivant, nous utiliserons ces informations dans l'élaboration des différentes lois de commande ainsi que dans la démonstration de stabilité.

III.5 Conclusion

Nous avons présenté dans ce chapitre deux types de commandes basées sur le modèle de connaissances, les commandes linéaires et les commandes non-linéaires.

Les commandes linéaires utilisent le modèle linéarisé de notre robot flexible. Cependant ce modèle n'étant valable qu'autour d'un point de fonctionnement, ces commandes sont surtout utilisées pour les problèmes de régulation ou pour de faibles déplacements du bras manipulateur. Or ceci est rarement le cas en robotique où les problèmes sont souvent du type poursuite de trajectoire avec des consignes qui peuvent varier d'une façon importante au cours du temps.

Nous nous sommes donc orientés vers les commandes non-linéaires. Nous avons retenus dans le cadre de notre travail, la commande non linéaire à paramètres fixes pour sa simplicité et la commande adaptative non linéaire directe pour ses propriétés de compensation des erreurs de modélisations et des variations des paramètres au cours du mouvement. Nous décrivons en détail ces deux commandes dans le chapitre suivant.

Une analyse des performances de ces deux commandes conclura quand à la nécessité d'employer un nouveau type de commande que nous avons élaboré. Ce nouveau type de commande hybride associera la commande non-linéaire à la commande à base de réseaux de neurones pour une efficacité et des performances accrues lors du contrôle.

Chapitre IV

Commande hybride de robots flexibles

IV.1 Introduction

Une nouvelle famille de commandes 'intelligentes' est présentée dans ce Chapitre. Elle représente notre propre contribution au domaine de la commande de robots flexibles. Cette famille est constituée de commandes basées sur la modélisation semi-physique ou modèle de type 'boite grise'. Cette technique a pour but de combiner le meilleur des deux mondes : le modèle à base de connaissances ou modèle de type 'boite blanche' avec le modèle statistique ou modèle de type 'boite noire'.

Le modèle à base de connaissances ou modèle de type 'boite blanche' est une description mathématique du phénomène qui se produit dans le système. Cette description est fondée sur les lois de la physique ou de la chimie, biologie, sociologie, etc.

Typiquement, les équations impliquées dans le modèle sont des équations de cinématique, thermodynamique, résistance des matériaux etc. Elles contiennent des paramètres qui ont un sens physique (masse, rigidité de flexion, coefficient de diffusion etc.) et elles peuvent aussi contenir un petit nombre de paramètres qui sont déterminés par régression à partir des mesures [Dre 2005].

Par opposition, un modèle de type 'boite noire' est une description paramétrée du processus basée sur la théorie d'apprentissage statistique. Tous les paramètres du modèle sont estimés à partir de mesures effectuées sur le processus. Ce modèle ne prend pas en considération une connaissance préalable du processus (ou alors seulement des connaissances minimes).

Très souvent, les dispositifs ou les algorithmes qui peuvent apprendre à partir de données sont décrits comme intelligents. Les facultés mentales d'apprentissage de l'homme, de généralisation, de mémorisation et de prédiction devraient constituer le fondement de chaque dispositif artificiel ou système intelligent [Er 2009].

Même si nous sommes encore loin d'atteindre quelque chose de similaire à l'intelligence humaine, plusieurs produits utilisant les Réseaux de Neurones Artificiels (RNA), les Machines à Vecteur de Support (MVS) et les Modèles à base de Logique Floue (MLF) présentent les propriétés précédentes [Kas 1998].

Un système de commande intelligent est entre autres caractérisé par sa capacité à faire face à une grande quantité de données bruitées, arrivants simultanément de différents capteurs. De plus, ce type de commande arrive à compenser les incertitudes dans son modèle afin de produire la loi de commande adéquate [Kec 2001, Mao 2010b].

Le modèle semi-physique ou modèle de type 'boite grise', peut être perçu comme un compromis entre le modèle à base de connaissances et le modèle statistique. Il peut englober toute les connaissances qu'a l'ingénieur ou l'expert sur son processus (ou une bonne partie des connaissances) et, en addition, reposer aussi sur des fonctions paramétrées, dont les paramètres sont déterminés par la mesure.

Cette combinaison rend possible la prise en charge de tous les phénomènes physiques qui ne sont, soit pas modélisés du tout car non connus, soit pas modélisés avec la bonne précision au travers d'une connaissance préalable [Dre 2005].

Une commande basée sur une modélisation de type 'boite grise' est très utile lorsqu'un modèle à base de connaissances existe mais est incomplet et ne peut être amélioré avec une analyse plus approfondi, ou peut l'être mais au détriment d'un temps de calcul et d'une complexité tels qui le rendent inutilisable notamment dans les problèmes de contrôle en temps réel [Mao 2008b, Mao 2008a, Mao 2010b].

Les systèmes physiques sont par nature non linéaires et sont généralement gouvernés par des équations aux dérivés partielles complexes. Le modèle dynamique utilisé dans l'architecture de la commande est donc généralement un modèle approché. Or, les erreurs de modélisation introduites par cette approximation influencent grandement les performances du contrôleur.

Le choix d'une commande adaptative, à base de réseaux de neurones artificiels, permet de prendre en charge ces erreurs de modélisation et rend possible la compensation, jusqu'à un certain point, des phénomènes physiques, tel que les frottements, dont la représentation mathématique est difficile à réaliser [Mao 2007, Mao 2009c, Mao 2009b, Mao 2010a].

Le temps de réponse de la loi commande nous parait être un facteur déterminant pour l'élaboration d'une commande dont le comportement sera réellement efficace dans la pratique. L'utilisation des réseaux de neurones va nous permettre d'alléger considérablement la loi de commande et va accélérer son calcul, augmentant ainsi l'efficacité de cette dernière lors du contrôle [Mao 2008b, Mao 1008a, Mao 2010b].

Dans ce chapitre nous présentons une approche hybride pour palier au problème de commande de robots flexibles. Cette approche est basée sur le modèle de type 'boite grise'. Cette commande originale permet de faire face aux perturbations et incertitudes structurés et non structurées lors du contrôle tout en amortissant les vibrations des bras. Ce qui constitue une contribution à l'amélioration des performances de la commande de robots flexibles [Mao 2007, Mao 2008b, Mao 2008a, Mao 2009c, Mao 2010b].

L'architecture de commande que nous proposons est constituée de deux contrôleurs. Le premier contrôleur est basé sur l'approximation par les réseaux de neurones des fonctions non linéaires extraits du modèle dynamique et utilisés dans la commande. Son but est de fournir un contrôle stable des positions et vitesses articulaires ainsi que l'amortissement des vibrations de chaque bras. Ensuite, une commande neuronale adaptative est ajoutée pour compenser les non linéarités qui ne sont pas prise en charge par le modèle, de même que les incertitudes sur l'estimation des paramètres du modèle.

Finalement, La robustesse de la commande hybride proposée est testée face à une incertitude importante sur l'estimation des paramètres physiques du robot et comparée aux résultats obtenus avec une commande non linéaire classique. Les résultats en simulation montrent l'efficacité de la stratégie de commande proposée.

IV.2 Commande non linéaire à paramètres fixes

Afin d'élaborer la commande hybride basée sur la modélisation de type 'boite grise', nous devons d'abord présenter la commande non linéaire qui, elle, est basée entièrement sur le modèle de connaissances c'est-à-dire le modèle dynamique du robot flexible.

IV.2.1 Principe

La commande non-linéaire présentée est une généralisation de la loi 'computed torque' classiquement utilisée dans la commande de robots manipulateurs rigides [Slo 1987]. Elle est constituée en partie du modèle non linéaire réduit du robot flexible complété par une partie correspondant à un contrôleur proportionnel et dérivé (PD) [Pha 1992].

Réécrivons l'équation (I.13) du Chapitre I, qui décrit le modèle dynamique du robot flexible, de la manière suivante:

$$\begin{bmatrix} \boldsymbol{\Gamma} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{A_r} & \mathbf{A_{re}} \\ \mathbf{A_{er}} & \mathbf{A_e} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_\mathbf{r} \\ \ddot{\mathbf{q}}_\mathbf{e} \end{bmatrix} + \begin{bmatrix} \mathbf{h_r} & \mathbf{h_{re}} \\ \mathbf{h_{er}} & \mathbf{h_e} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_\mathbf{r} \\ \dot{\mathbf{q}}_\mathbf{e} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K_e} \end{bmatrix} \begin{bmatrix} \mathbf{q_r} \\ \mathbf{q_e} \end{bmatrix} \qquad \text{(IV.1)}$$

on en déduit que:

$$\boldsymbol{\Gamma} = \mathbf{A_r}\,\ddot{\mathbf{q}}_\mathbf{r} + \mathbf{h_r}\,\dot{\mathbf{q}}_\mathbf{r} + \mathbf{A_{re}}\,\ddot{\mathbf{q}}_\mathbf{e} + \mathbf{h_{re}}\,\dot{\mathbf{q}}_\mathbf{e} \qquad \text{(IV.1a)}$$

$$\mathbf{0} = \mathbf{A_{er}}\,\ddot{\mathbf{q}}_\mathbf{r} + \mathbf{h_{er}}\,\dot{\mathbf{q}}_\mathbf{r} + \mathbf{A_e}\,\ddot{\mathbf{q}}_\mathbf{e} + \mathbf{h_e}\,\dot{\mathbf{q}}_\mathbf{e} + \mathbf{K_e}\,\mathbf{q_e} \qquad \text{(IV.1b)}$$

Nous pouvons alors utiliser la loi de commande suivante:

$$\boldsymbol{\Gamma}_\mathbf{NL} = \mathbf{A_r}\,\ddot{\mathbf{q}}_\mathbf{r}^\mathbf{d} + \mathbf{h_r}\,\dot{\mathbf{q}}_\mathbf{r}^\mathbf{d} + \mathbf{K_{pr}}\,\tilde{\mathbf{q}}_\mathbf{r} + \mathbf{K_{vr}}\,\dot{\tilde{\mathbf{q}}}_\mathbf{r} \qquad \text{(IV.2)}$$

où,

$\mathbf{q}_\mathbf{r}^\mathbf{d}$, $\dot{\mathbf{q}}_\mathbf{r}^\mathbf{d}$ et $\ddot{\mathbf{q}}_\mathbf{r}^\mathbf{d}$ définissent la trajectoire angulaire désirée,

$\tilde{\mathbf{q}}_\mathbf{r} = \mathbf{q}_\mathbf{r}^\mathbf{d} - \mathbf{q}_\mathbf{r}$ et $\dot{\tilde{\mathbf{q}}}_\mathbf{r} = \dot{\mathbf{q}}_\mathbf{r}^\mathbf{d} - \dot{\mathbf{q}}_\mathbf{r}$ sont les erreurs en positions et vitesses angulaires,

$\mathbf{K_{pr}}$ et $\mathbf{K_{vr}}$ sont des matrices définies positives de gain.

IV.2.2 Etude de la stabilité

On se place dans le cas idéal où il n'y a pas d'erreur sur l'évaluation des paramètres dynamiques $\mathbf{X}$ du robot flexible**.**

En soustrayant la loi de commande (IV.2) à l'équation dynamique (IV.1a), on obtient l'équation d'erreur suivante:

$$\mathbf{A_r}\,\ddot{\tilde{\mathbf{q}}}_\mathbf{r} + \mathbf{A_{re}}\,\ddot{\tilde{\mathbf{q}}}_\mathbf{e} + \mathbf{h_r}\,\dot{\tilde{\mathbf{q}}}_\mathbf{r} + \mathbf{h_{re}}\,\dot{\tilde{\mathbf{q}}}_\mathbf{e} + \mathbf{K_{pr}}\,\tilde{\mathbf{q}}_\mathbf{r} + \mathbf{K_{vr}}\,\dot{\tilde{\mathbf{q}}}_\mathbf{r} = \mathbf{0} \qquad \text{(IV.3)}$$

avec,

$\tilde{\mathbf{q}}_\mathbf{e} = \mathbf{0} - \mathbf{q_e} = -\mathbf{q_e}$ et $\dot{\tilde{\mathbf{q}}}_\mathbf{e} = \mathbf{0} - \dot{\mathbf{q}}_\mathbf{e} = -\dot{\mathbf{q}}_\mathbf{e}$ erreurs de stabilisation élastique en position et en vitesse respectivement.

D'autre part, réécrivons l'équation de couplage (IV.1b) en fonction des variables d'erreur de suivi et de stabilisation:

$$\mathbf{A_{er}}\,\ddot{\tilde{\mathbf{q}}}_\mathbf{r} + \mathbf{A_e}\,\ddot{\tilde{\mathbf{q}}}_\mathbf{e} + \mathbf{h_{er}}\,\dot{\tilde{\mathbf{q}}}_\mathbf{r} + \mathbf{h_e}\,\dot{\tilde{\mathbf{q}}}_\mathbf{e} + \mathbf{K_e}\,\tilde{\mathbf{q}}_\mathbf{e} = \mathbf{A_{er}}\,\ddot{\mathbf{q}}_\mathbf{r}^\mathbf{d} + \mathbf{h_{er}}\,\dot{\mathbf{q}}_\mathbf{r}^\mathbf{d} \qquad \text{(IV.4)}$$

En utilisant les équations (IV.3) et (IV.4), l'équation d'erreur globale devien :

$$\begin{bmatrix} \mathbf{A_r} & \mathbf{A_{re}} \\ \mathbf{A_{er}} & \mathbf{A_e} \end{bmatrix}\begin{bmatrix} \ddot{\tilde{\mathbf{q}}}_\mathbf{r} \\ \ddot{\tilde{\mathbf{q}}}_\mathbf{e} \end{bmatrix} + \begin{bmatrix} \mathbf{h_r} & \mathbf{h_{re}} \\ \mathbf{h_{er}} & \mathbf{h_e} \end{bmatrix}\begin{bmatrix} \dot{\tilde{\mathbf{q}}}_\mathbf{r} \\ \dot{\tilde{\mathbf{q}}}_\mathbf{e} \end{bmatrix} + \begin{bmatrix} \mathbf{K_{pr}} & \mathbf{0} \\ \mathbf{0} & \mathbf{K_e} \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{q}}_\mathbf{r} \\ \tilde{\mathbf{q}}_\mathbf{e} \end{bmatrix} \ldots$$

$$+ \begin{bmatrix} \mathbf{K_{vr}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \dot{\tilde{\mathbf{q}}}_\mathbf{r} \\ \dot{\tilde{\mathbf{q}}}_\mathbf{e} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{A_{er}}\,\ddot{\mathbf{q}}_\mathbf{r}^\mathbf{d} + \mathbf{h_{er}}\,\dot{\mathbf{q}}_\mathbf{r}^\mathbf{d} \end{bmatrix} \qquad \text{(IV.5)}$$

ou sous forme compacte:

$$\mathbf{A}\,\ddot{\tilde{\mathbf{q}}} + \mathbf{h}\,\dot{\tilde{\mathbf{q}}} + \mathbf{K_p}\,\tilde{\mathbf{q}} + \mathbf{K_v}\,\dot{\tilde{\mathbf{q}}} + \mathbf{s_1} = \mathbf{0} \qquad \text{(IV.6)}$$

où,

$\mathbf{K_p} = \begin{bmatrix} \mathbf{K_{pr}} & \mathbf{0} \\ \mathbf{0} & \mathbf{K_e} \end{bmatrix}$, $\mathbf{K_v} = \begin{bmatrix} \mathbf{K_{vr}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ sont des matrices constantes positives définies

et $\mathbf{s_1} = -\begin{bmatrix} \mathbf{0} \\ \mathbf{A_{er}}\,\ddot{\mathbf{q}}_\mathbf{r}^\mathbf{d} + \mathbf{h_{er}}\,\dot{\mathbf{q}}_\mathbf{r}^\mathbf{d} \end{bmatrix}$.

Pour étudier la stabilité du système global, on considère la fonction de Lyapunov suivante:

$$\mathbf{V} = \frac{1}{2}\,\dot{\tilde{\mathbf{q}}}^{\mathrm{T}}\,\mathbf{A}\,\dot{\tilde{\mathbf{q}}} + \frac{1}{2}\,\tilde{\mathbf{q}}^{\mathrm{T}}\,\mathbf{K_p}\,\tilde{\mathbf{q}} \tag{IV.7}$$

la dérivation de $\mathbf{V}$ donne:

$$\dot{\mathbf{V}} = \dot{\tilde{\mathbf{q}}}^{\mathrm{T}}\,(\mathbf{A}\,\ddot{\tilde{\mathbf{q}}} + \frac{1}{2}\,\dot{\mathbf{A}}\,\dot{\tilde{\mathbf{q}}} + \mathbf{K_p}\,\tilde{\mathbf{q}}) \tag{IV.8}$$

en utilisant (IV.6), on obtient:

$$\dot{\mathbf{V}} = \dot{\tilde{\mathbf{q}}}^{\mathrm{T}}\,(\frac{1}{2}\,\dot{\mathbf{A}} - \mathbf{h})\,\dot{\tilde{\mathbf{q}}} - \dot{\tilde{\mathbf{q}}}^{\mathrm{T}}\,(\mathbf{K_v}\,\dot{\tilde{\mathbf{q}}} + \mathbf{s_1}) \tag{IV.9}$$

La propriété de passivité du robot flexible (voir l'annexe D) implique que la matrice $(\frac{1}{2}\,\dot{\mathbf{A}} - \mathbf{h})$ est antisymétrique, on a alors:

$$\dot{\mathbf{V}} = \dot{\mathbf{V}}_1 + \dot{\mathbf{V}}_2 \tag{IV.10}$$

avec,

$$\dot{\mathbf{V}}_1 = -\,\dot{\tilde{\mathbf{q}}}_{\mathbf{r}}^{\mathrm{T}}\,\mathbf{K_{vr}}\,\dot{\tilde{\mathbf{q}}}_{\mathbf{r}} \tag{IV.11}$$

et,

$$\dot{\mathbf{V}}_2 = \dot{\tilde{\mathbf{q}}}_{\mathbf{e}}^{\mathrm{T}}\,(\mathbf{A_{er}}\,\ddot{\mathbf{q}}_{\mathbf{r}}^{\mathbf{d}} + \mathbf{h_{er}}\,\dot{\mathbf{q}}_{\mathbf{r}}^{\mathbf{d}}) \tag{IV.12}$$

Pendant le contrôle du mouvement du robot flexible, on peut distinguer deux phases différentes. La première phase est la phase de régulation statique autour d'une configuration ou phase de positionnement. La deuxième est la phase de poursuite dynamique de trajectoire qui est fonction du temps.

Nous étudierons l'équation (IV.10) séparément sur ces deux phases. Selon la deuxième méthode de Lyapunov (voir l'annexe C), la commande est asymptotiquement stable si les conditions suivantes sont remplies :

- ✓ $\mathbf{V}$ est strictement positive partout sauf en $\tilde{\mathbf{q}} = \mathbf{0}$ où elle est nulle,
- ✓ $\dot{\mathbf{V}}$ est strictement négative partout sauf en $\tilde{\mathbf{q}} = \mathbf{0}$ où elle est nulle.

- **Phase de positionnement**

Lorsque l'organe terminal du robot se situe autour d'une position finale désirée, on peut écrire que $\ddot{\mathbf{q}}_{\mathbf{r}}^{\mathbf{d}} = \dot{\mathbf{q}}_{\mathbf{r}}^{\mathbf{d}} = \mathbf{0}$ ce qui annule $\dot{\mathbf{V}}_{\mathbf{2}}$. Il reste alors:

$$\dot{\mathbf{V}} = \dot{\mathbf{V}}_1 = -\dot{\tilde{\mathbf{q}}}_{\mathbf{r}}^{\mathrm{T}} \mathbf{K}_{\mathbf{vr}} \dot{\tilde{\mathbf{q}}}_{\mathbf{r}} \qquad \text{(IV.13)}$$

Donc, il est clair que:

- si $\dot{\tilde{\mathbf{q}}}_{\mathbf{r}} \neq \mathbf{0}$ alors $\dot{\mathbf{V}} < \mathbf{0}$ et $\mathbf{V} > \mathbf{0}$,
- si $\dot{\tilde{\mathbf{q}}}_{\mathbf{r}} = \mathbf{0}$ on a $\dot{\mathbf{V}} = \mathbf{0}$ et donc $\mathbf{V}$ est constante, aussi d'après (IV.7) on déduit que $\tilde{\mathbf{q}}$ est constante. Maintenant si on considère l'équation dynamique des erreurs (IV.6) avec le fait que $\ddot{\tilde{\mathbf{q}}} = \dot{\tilde{\mathbf{q}}} = \mathbf{0}$, le vecteur $\tilde{\mathbf{q}}$ doit être nul, ce qui annule par la même $\mathbf{V}$.

La stabilité asymptotique étant assurée, le positionnement articulaire et la stabilisation des variables élastiques sont ainsi réalisés.

- **Phase de poursuite de la trajectoire**

Supposons que $\mathbf{K}_{\mathbf{vr}}$ soit très grand et que $\ddot{\mathbf{q}}_{\mathbf{r}}^{\mathbf{d}}$ ainsi que $\dot{\mathbf{q}}_{\mathbf{r}}^{\mathbf{d}}$ soient petits, alors, l'allure de la fonction $\dot{\mathbf{V}}$ sera pratiquement celle de $\dot{\mathbf{V}}_{\mathbf{1}}$ qui est strictement négative. La loi de commande reste donc asymptotiquement stable comme dans le cas précédent.

En pratique, afin de minimiser les effets de $\dot{\mathbf{V}}_{\mathbf{2}}$ par rapport à ceux de $\dot{\mathbf{V}}_{\mathbf{1}}$ il faudra veiller à ce que les vitesses et accélérations de consigne ne soient pas trop importantes pour un réglage de $\mathbf{K}_{\mathbf{vr}}$ donné, de façon à ce que $\dot{\mathbf{V}}$ reste essentiellement négative assurant ainsi la stabilité de la commande.

Nous pouvons considérer $\dot{\mathbf{V}}_{\mathbf{2}}$ comme étant un terme de perturbation qui exprime l'idée suivante: plus rapide est la dynamique désirée, plus importantes sont les vibrations et les erreurs de suivi [Pha 1992].

IV.2.3 Etude de la robustesse paramétrique

Considérons maintenant le cas où les paramètres dynamiques estimés $\hat{\mathbf{X}}$ utilisés pour la commande diffèrent des paramètres dynamiques réels $\mathbf{X}$ du robot flexible.

On a vu au Chapitre I et tel que détaillé en annexe D, que le modèle dynamique du robot flexible pouvait s'écrire linéairement par rapport aux paramètres dynamiques $\mathbf{X}$.

D'après l'équation (D.2) de l'annexe D on peut écrire le couple articulaire désiré $\mathbf{\Gamma_d}$ de la manière suivante:

$$\mathbf{\Gamma_d} = \mathbf{D_r}\,\mathbf{X} \tag{IV.14}$$

Alors que le couple articulaire réel $\mathbf{\Gamma}$, calculé à partir de la loi de commande, s'écrit quand à lui comme suit,

$$\mathbf{\Gamma} = \mathbf{D_r}\,\hat{\mathbf{X}} \tag{IV.15}$$

En soustrayant l'équation (IV.15) de l'équation (IV.14) nous obtenons l'erreur $\tilde{\mathbf{\Gamma}}$ effectuée lors du calcul du couple articulaire:

$$\tilde{\mathbf{\Gamma}} = \mathbf{D_r}\,\tilde{\mathbf{X}} \tag{IV.16}$$

avec, $\tilde{\mathbf{X}} = \mathbf{X} - \hat{\mathbf{X}}$ étant le vecteur d'erreurs d'estimation.

Par conséquent l'équation (IV.3) n'est plus égale à zéro mais à $\tilde{\mathbf{\Gamma}}$ et finalement l'équation globale des erreurs (IV.6) devient:

$$\mathbf{A}\,\ddot{\tilde{\mathbf{q}}} + \mathbf{h}\,\dot{\tilde{\mathbf{q}}} + \mathbf{K_p}\,\tilde{\mathbf{q}} + \mathbf{K_v}\,\dot{\tilde{\mathbf{q}}} + \mathbf{s_1} + \mathbf{s_2} = \mathbf{0} \tag{IV. 17}$$

où, $\mathbf{s_2} = -\begin{bmatrix} \mathbf{D_r}\,\tilde{\mathbf{X}} \\ \mathbf{0} \end{bmatrix}$.

L'erreur sur les paramètres dynamiques introduit donc un nouveau terme de perturbation représenté par $\mathbf{s_2}$.

Dans le cas où l'écart entre les valeurs des paramètres dynamiques réels ($\mathbf{X}$) du robot flexible et les valeurs des paramètres dynamiques ($\hat{\mathbf{X}}$) utilisées dans la commande est très important, on introduit un algorithme d'adaptation pour modifier en ligne $\hat{\mathbf{X}}$ afin que la stabilité du système global continue à être assurée. Il s'agit de la commande non linéaire adaptative que nous présentons ci après.

IV.3 Commande non linéaire adaptative

IV.3.1 Principe

Afin d'assurer la stabilité de la loi de commande (IV.2) en présence d'incertitudes sur les valeurs des paramètres dynamiques ($\hat{\mathbf{X}}$) utilisés dans la commande, on considère la fonction de Lyapunov suivante :

$$\mathbf{V} = \frac{1}{2}\dot{\tilde{\mathbf{q}}}^{\mathrm{T}}\mathbf{A}\,\dot{\tilde{\mathbf{q}}} + \frac{1}{2}\tilde{\mathbf{q}}^{\mathrm{T}}\mathbf{K_p}\,\tilde{\mathbf{q}} + \frac{1}{2}\tilde{\mathbf{X}}^{\mathrm{T}}\mathbf{K_g}^{-1}\tilde{\mathbf{X}} \qquad \text{(IV.18)}$$

où, $\mathbf{K_g}$ est une matrice constante définie positive de gain d'adaptation.

En dérivant l'équation (IV.18) nous obtenons:

$$\dot{\mathbf{V}} = \dot{\tilde{\mathbf{q}}}^{\mathrm{T}}(\mathbf{A}\,\ddot{\tilde{\mathbf{q}}} + \frac{1}{2}\dot{\mathbf{A}}\,\dot{\tilde{\mathbf{q}}} + \mathbf{K_p}\,\tilde{\mathbf{q}}) + \tilde{\mathbf{X}}^{\mathrm{T}}\mathbf{K_g}^{-1}\dot{\tilde{\mathbf{X}}} \qquad \text{(IV.19)}$$

or, comme $\mathbf{X}$ est constant, on a aussi:

$$\dot{\tilde{\mathbf{X}}} = -\dot{\hat{\mathbf{X}}} \qquad \text{(IV.20)}$$

donc l'expression de $\dot{\mathbf{V}}$ devient:

$$\dot{\mathbf{V}} = \dot{\tilde{\mathbf{q}}}^{\mathrm{T}}(\frac{1}{2}\dot{\mathbf{A}} - \mathbf{h})\,\dot{\tilde{\mathbf{q}}} - \dot{\tilde{\mathbf{q}}}^{\mathrm{T}}(\mathbf{K_v}\,\dot{\tilde{\mathbf{q}}} + \mathbf{s_1} + \mathbf{s_2}) - \tilde{\mathbf{X}}^{\mathrm{T}}\mathbf{K_g}^{-1}\dot{\hat{\mathbf{X}}} \qquad \text{(IV.21)}$$

En utilisant la propriété de passivité du robot flexible et en développant les différents termes de l'équation précédente on trouve finalement:

$$\dot{\mathbf{V}} = -\dot{\tilde{\mathbf{q}}}_{\mathbf{r}}^{\mathrm{T}}\mathbf{K_{vr}}\,\dot{\tilde{\mathbf{q}}}_{\mathbf{r}} + \dot{\tilde{\mathbf{q}}}_{\mathbf{e}}^{\mathrm{T}}(\mathbf{A_{er}}\,\ddot{\mathbf{q}}_{\mathbf{rd}} + \mathbf{h_{er}}\,\dot{\mathbf{q}}_{\mathbf{rd}}) + \tilde{\mathbf{X}}^{\mathrm{T}}(\mathbf{D_r}^{\mathrm{T}}\,\dot{\tilde{\mathbf{q}}}_{\mathbf{r}} - \mathbf{K_g}^{-1}\dot{\hat{\mathbf{X}}}) \qquad \text{(IV.22)}$$

En considérant la loi d'adaptation de type intégral suivante [Mao 2009a]:

$$\dot{\hat{\mathbf{X}}} = \mathbf{K_g}\,\mathbf{D_r}^{\mathrm{T}}\,\dot{\tilde{\mathbf{q}}}_{\mathbf{r}} \tag{IV.23}$$

on remarque que le dernier terme de $\dot{\mathbf{V}}$ est supprimé et on retrouve le cas de la commande à paramètres fixes. L'algorithme que nous avons élaboré pour cette loi de commande est donné en annexe E. La structure de la commande adaptative non linéaire est décrite par la Figure IV.1:

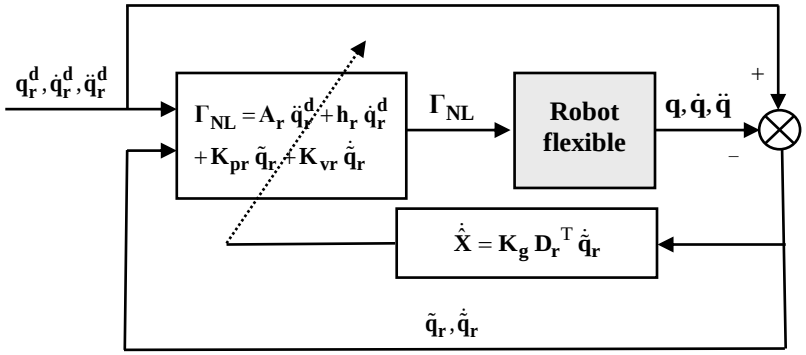

Figure IV.1 Le système de commande adaptatif non linéaire.

IV.3.2 Tests et simulations

Afin d'étudier les performances de la commande non linéaire à paramètres fixes et de la commande non linéaire adaptative directe, nous nous sommes proposés d'utiliser une trajectoire dynamique possédant une forme sinusoïdale:

$$\theta^{\mathrm{d}}(t) = \pi \sin\left(\frac{2\pi}{T}t\right) \tag{IV.24}$$

avec, $T = 30$ sec.

Nous avons aussi testé ces commandes avec un autre type de trajectoires dans [Mao 2009a], avec des résultats de simulation similaires.

Supposons que les valeurs des paramètres physiques du robot flexible sont tels que décrit en annexe A.1. Pour tester la robustesse de la commande non linéaire adaptative directe, nous allons considérer le cas extrême où l'erreur d'estimation sur les paramètres dynamique $\mathbf{X}$ du robot à deux bras flexibles est telle que:

$$\hat{\mathbf{X}} = \mathbf{X}/100 \tag{IV.25}$$

Nous utilisons ensuite les paramètres dynamiques estimés $\hat{\mathbf{X}}$ dans le calcul des matrices $\mathbf{A_r}$ et $\mathbf{h_r}$. Ceci va conduire la commande non linéaire à produire un couple articulaire incorrecte. Nous allons voire comment la commande adaptative va réagir face à cette erreur et comment elle va la corriger.

Les Figures IV.2 à IV.11 illustrent les résultats obtenus avec la commande adaptative non linéaire directe appliquée au robot manipulateur à deux bras flexibles. Ces figures décrivent l'évolution de: la position angulaire, l'erreur en position, la flèche, la vitesse angulaire et l'erreur sur la vitesse angulaire, pour l'articulation 1 et 2, respectivement.

La trajectoire désirée (consigne) est donnée dans les Figures IV.2, IV.5, IV.7 et IV.10 en pointillés. Les résultats de la commande non linéaire à paramètres fixes, décrite par l'équation (IV.2), sont reportés en ligne discontinue pour comparaison. Le Tableau IV.1 et le Tableau IV.2 présentent l'erreur maximale ainsi que l'erreur quadratique moyenne (RMS) de la position et de la vitesse angulaire obtenus avec les deux types de commande utilisés.

D'après les Figures IV.4 et IV.9, on voit que la flexion est bien contenue avec les deux types de commandes, avec des vibrations inférieures à 0.12 m pour la première articulation et inférieures à 0.03 m pour la seconde.

On remarque la présence d'oscillations plus nombreuses avec la commande adaptative car le couple articulaire généré avec cette dernière varie plus vite qu'avec la commande à paramètres fixes. Cependant leur amplitude n'augmente pas et la flèche reste bornée.

La trajectoire désirée impose une variation rapide du mouvement du fait que l'accélération angulaire est elle-même sinusoïdale sur ce type de trajectoire. On peut voir son impact dans le contrôle de la vitesse angulaire sur les Figures IV.5 et IV.10; spécialement quand l'accélération change de signe, de négative à positive, à l'instant t = 15 sec.

Le contrôle en vitesse de la trajectoire, obtenu avec la commande adaptative non linéaire, est correcte et l'erreur de suivi est satisfaisante. En comparaison, on note que la trajectoire de la vitesse angulaire obtenue avec la commande non linéaire à paramètres fixes dévie fortement de la consigne (voir les Figures IV.5 et IV.10).

D'après le Tableau IV.1 et le Tableau IV.2, on remarque que l'erreur en vitesse, obtenue avec la commande à paramètres fixes, atteint 0.32 rad/sec (18.3 deg/sec) pour la première articulation et 0.31 rad/sec (17.6 deg/sec) pour la seconde. La commande adaptative donne de meilleurs résultats, avec une erreur en vitesse inférieure à 0.024 rad/sec (1.4 deg/sec) pour la première articulation et inférieure à 0.043 rad/sec (2.5 deg/sec) pour la seconde (voir les Figures IV.6 et IV.11).

Pour le contrôle en position (voir les Figures IV.2 et IV.7), on note que la trajectoire angulaire obtenue avec la commande adaptative suit bien la trajectoire désirée; avec une erreur ne dépassant pas 0.0035 rad (0.2 deg) pour la première articulation et ne dépassant pas 0.031 rad (1.8 deg) pour la seconde (voir les Figures IV.3 et IV.8). Quand à la commande à paramètres fixes, l'erreur obtenue sur la position angulaire, dépasse facilement 0.93 rad (53 deg) pour la première articulation et 0.55 rad (31 deg) pour la seconde.

IV.3.3 Analyse et critique de la commande adaptative non linéaire

Les résultats de la commande adaptative non linéaire, s'ils sont bons en simulation, ne peuvent refléter les performances de cette commande dans la réalité, car un grave problème vient entacher les performances de celle ci dans la pratique.

En effet la commande adaptative non linéaire lors de nos tests a nécessité l'utilisation d'un pas (intervalle de temps entre chaque itération) très petit de 0.001 sec pour obtenir ces résultats. Or la simulation du suivi d'une trajectoire de 30 sec a nécessité sur un ordinateur avec un micro processeur de type pentium IV de fréquence 2.4 Ghz, 30 minutes.

Chaque itération lors du calcul de la commande adaptative a donc nécessité 0.06 sec, c'est-à-dire, largement plus que le 0.001 sec requis. On peut donc conclure qu'avec la puissance des microcalculateurs et microprocesseurs actuels, cet commande n'est pas compatible, étant donné la complexité de notre modèle dynamique, avec les contraintes du contrôle en temps réel et verra ses performances chuter en fonction de la rapidité de variation de la trajectoire à suivre.

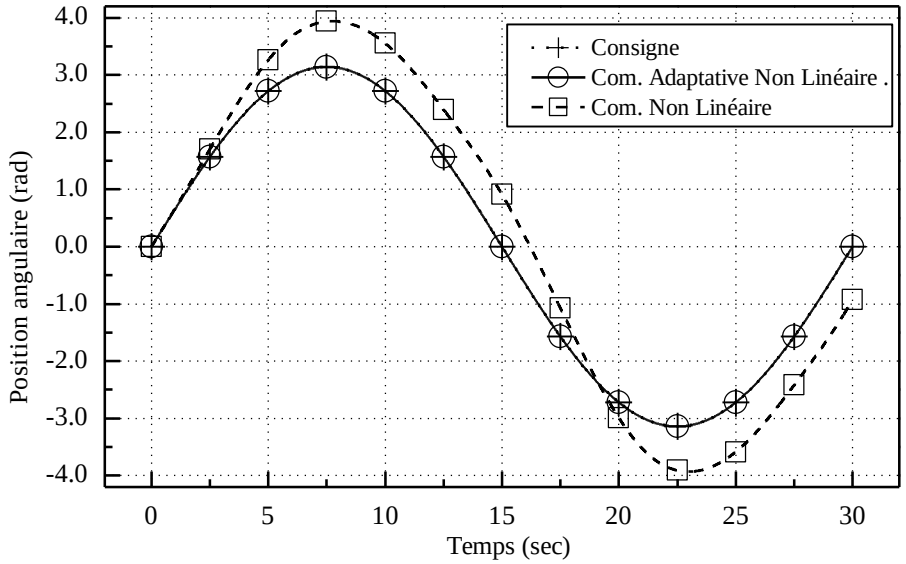

Figure IV.2 Evolution de la position angulaire θ_1 (rad)

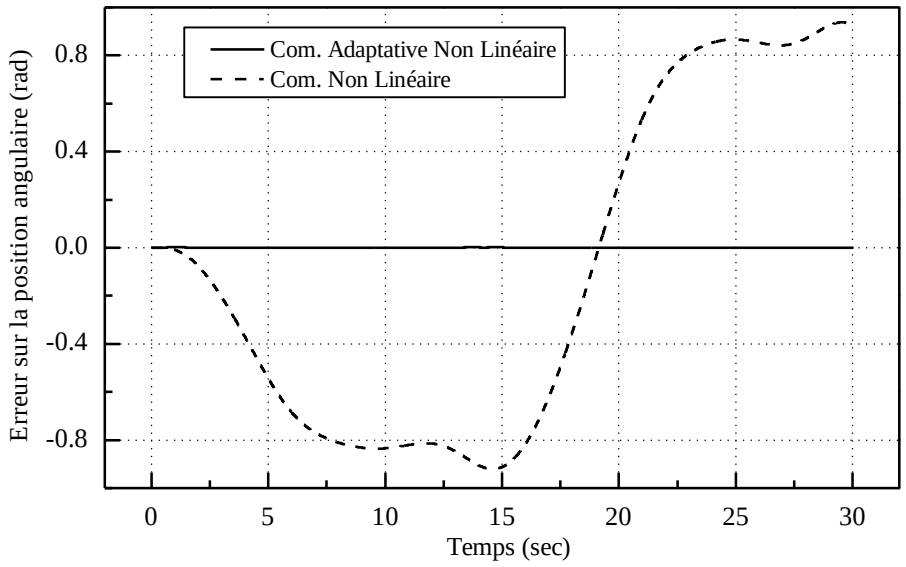

Figure IV.3 Evolution de l'erreur sur la position angulaire $\tilde{\theta}_1$ (rad)

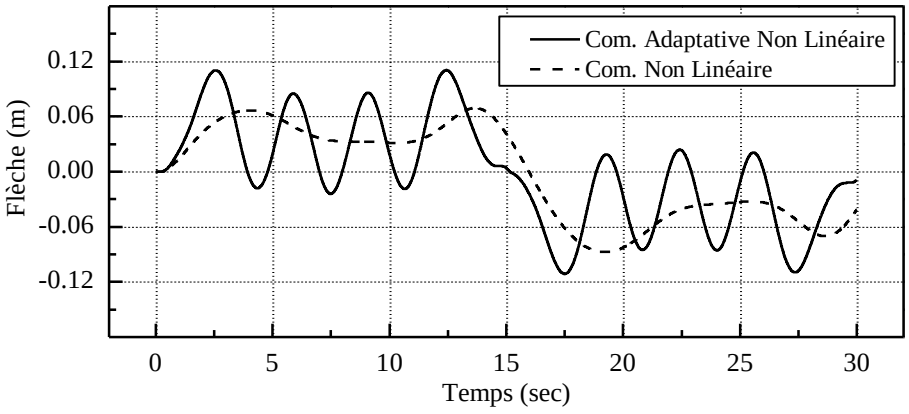

Figure IV.4 Evolution de la flèche f_1 (m)

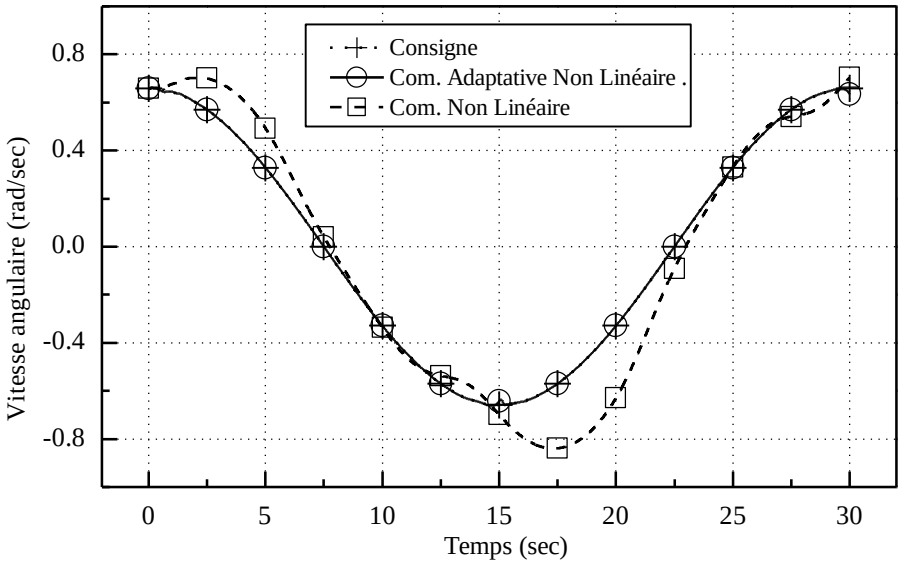

Figure IV.5 Evolution de la vitesse angulaire $\dot{\theta}_1$ (rad/sec)

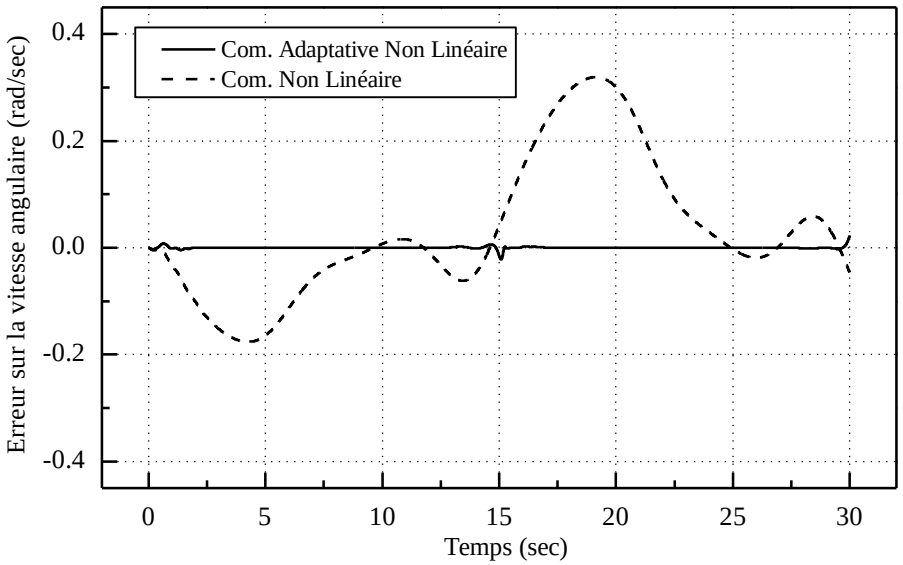

Figure IV.6 Evolution de l'erreur sur la vitesse angulaire $\tilde{\dot{\theta}}_1$ (rad/sec)

Tableau IV.1

ERREUR DE SUIVI DE TRAJECTOIRE POUR L'ARTICULATION 1

	Erreur Maximum		Erreur RMS	
Variable	θ_1 (rad)	$\dot{\theta}_1$ (rad/sec)	θ_1 (rad)	$\dot{\theta}_1$ (rad/sec)
Commande Adaptative Non Linéaire	3.42×10^{-3}	2.33×10^{-2}	9.17×10^{-4}	2.16×10^{-3}
Commande Non Linéaire	9.38×10^{-1}	3.19×10^{-1}	7.06×10^{-1}	1.36×10^{-1}

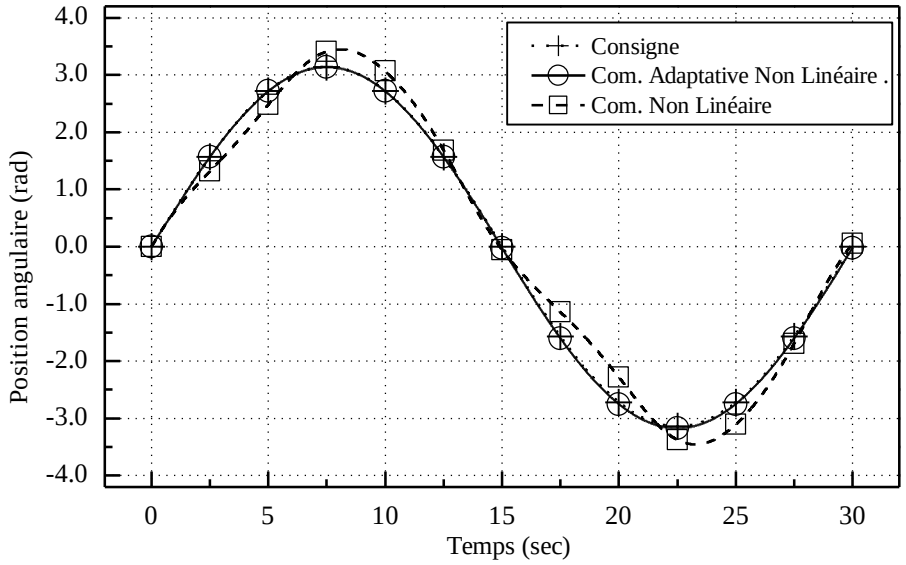

Figure IV.7 Evolution de la position angulaire θ_2 (rad)

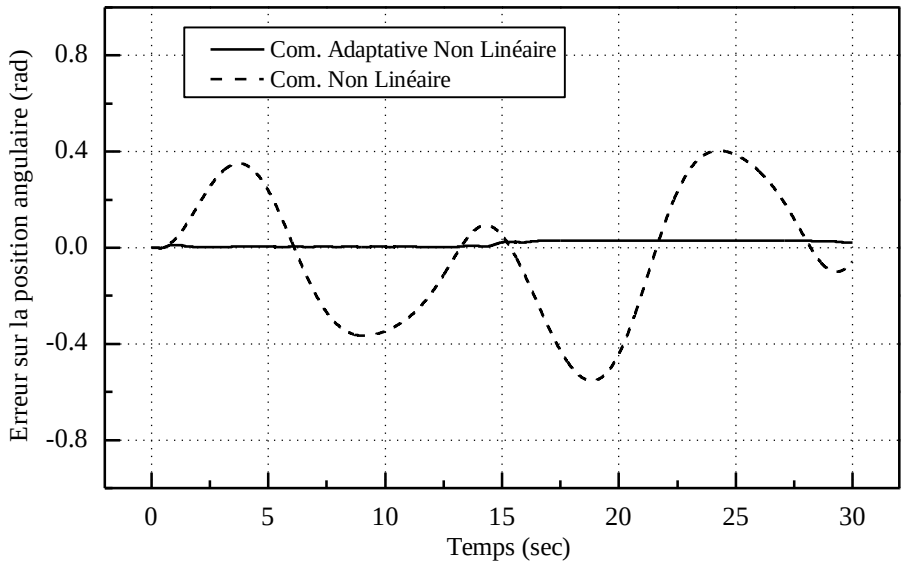

Figure IV.8 Evolution de l'erreur sur la position angulaire $\tilde{\theta}_2$ (rad)

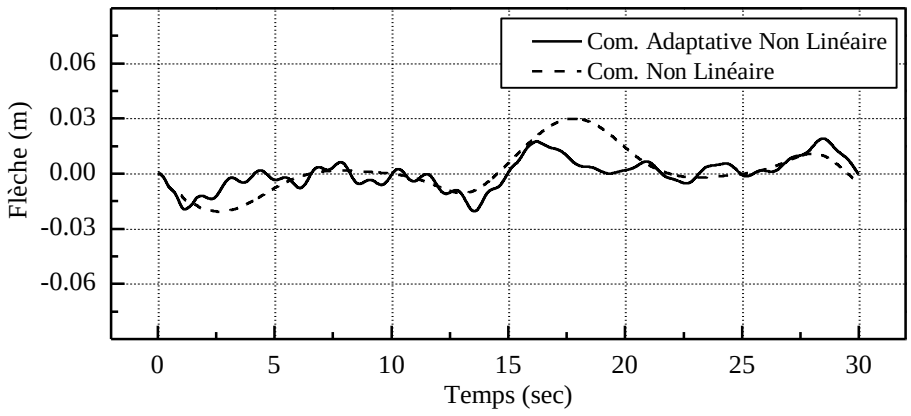

Figure IV.9 Evolution de la flèche f_2 (m)

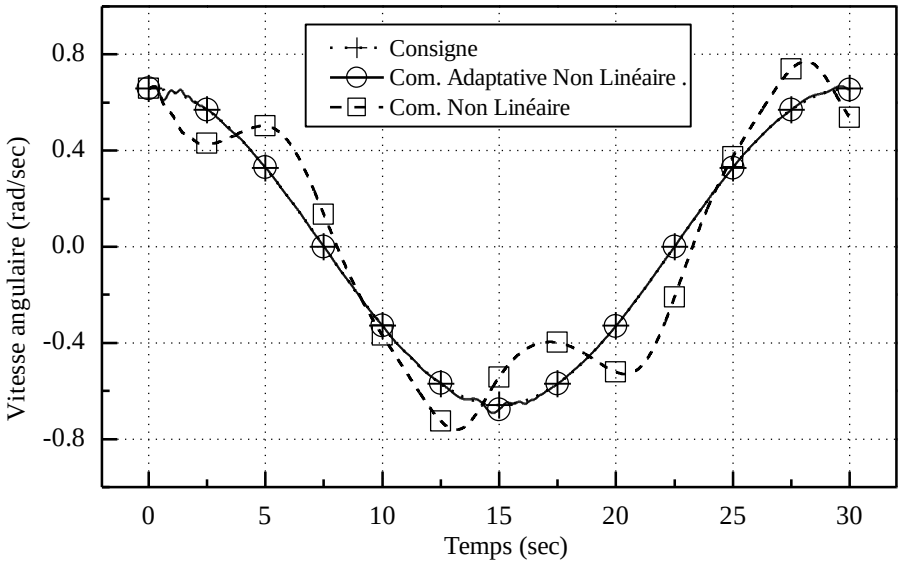

Figure IV.10 Evolution de la vitesse angulaire $\dot{\theta}_2$ (rad/sec)

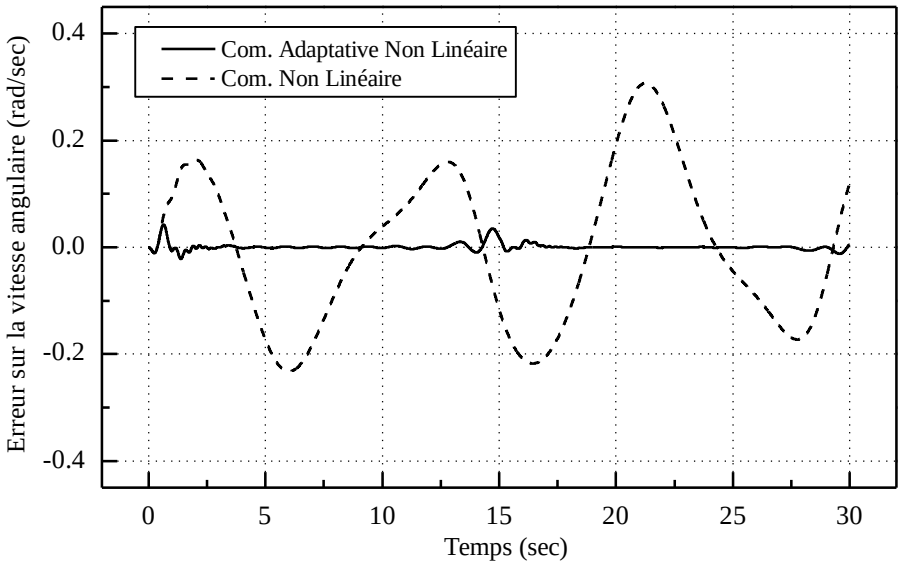

Figure IV.11 Evolution de l'erreur sur la vitesse angulaire $\dot{\tilde{\theta}}_2$ (rad/sec)

Tableau IV.2

ERREUR DE SUIVI DE TRAJECTOIRE POUR L'ARTICULATION 2

	Erreur Maximum		Erreur RMS	
Variable	θ_2 (rad)	$\dot{\theta}_2$ (rad/sec)	θ_2 (rad)	$\dot{\theta}_2$ (rad/sec)
Commande Adaptative Non Linéaire	3.06×10^{-2}	4.24×10^{-2}	2.07×10^{-2}	6.65×10^{-3}
Commande Non Linéaire	5.54×10^{-1}	3.07×10^{-1}	2.78×10^{-1}	1.45×10^{-1}

IV.4 Commande hybride à base de RNA

Pour palier aux inconvénients des commandes non linéaires précédentes, nous proposons une nouvelle structure de commande. Cette structure hybride est une combinaison de deux contrôleurs. La construction du premier contrôleur est basée sur l'approximation des fonctions non linéaires dans l'équation (IV.2) par des réseaux de neurones pour réduire le temps de calcul et augmenter ainsi la réactivité de la commande [Mao 2008b, Mao 2008a, Mao 2010b].

Le deuxième contrôleur est basé, quand à lui, sur un réseau de neurones adaptatif. Ici le réseau est entrainé 'en ligne' pour compenser les erreurs dues aux incertitudes structurées (incertitudes sur l'estimation des paramètres physiques dans le modèle) et non structurés (perturbations externes et phénomènes physiques non pris en charge par le modèle), améliorant ainsi la précision du contrôleur globale [Mao 2007, Mao 2009c].

IV.4.1 Réduction du temps de réponse avec les RNA

La loi de commande non linéaire présentée dans l'équation (IV.2) présente des avantages majeurs, car elle utilise les informations extraites des équations du modèle dynamique pour le contrôle du robot. Des caractéristiques physiques comme la passivité du système peuvent alors être utilisées [Kur 2005] pour élaborer des commandes stables, comme vu précédemment.

L'inconvénient de cette méthode est que l'utilisation des équations du modèle dynamique dans la loi de commande peut conduire à un contrôleur complexe. Le temps de calcul que nécessite ce contrôleur peut alors être très grand.

Ceci est typiquement le cas avec les robots manipulateurs à bras flexibles car ils sont généralement gouvernés par des équations différentielles non linéaires complexes.

Ce qui conduit inévitablement à un modèle lourd à manipuler. L'utilisation d'un tel modèle dans la loi de commande peut alors être incompatible avec les contraintes du contrôle en temps réel.

Pour éviter ce problème nous proposons d'approximer la partie du modèle qui est utilisé dans la loi de commande par un réseau de neurones. Le point essentiel qui fait des RNA un outil idéal pour les systèmes de contrôle est qu'ils fonctionnent avec des algorithmes de régression non linéaire qui peuvent modéliser des systèmes de grandes dimensions tout en gardant une extrême flexibilité due à leurs possibilités d'apprentissage. De plus, le calcul de la réponse du réseau est très rapide et peut être réalisé en parallèle sur des machines à plusieurs unités de calcul.

L'utilisation des équations du modèle dynamique dans l'entrainement du réseau de neurones présentent plusieurs avantages. Les données sont facilement et rapidement acquises par simulation. Le jeu de données, ainsi obtenu, n'est pas teinté de bruits de mesures.

Les valeurs d'entrées/sorties peuvent être produites avec un nombre suffisant pour permettre une très bonne approximation du modèle. De plus, il est possible de générer un jeu d'entrées/sorties qui a une meilleure représentation du système, du fait que le modèle dynamique, contrairement au système réel, peut être simulé sur une large plage de fonctionnement, ce qui permet de mieux caractériser ces non linéarités.

Nous proposons donc, dans le cadre de nôtre travail, d'approximer les fonctions $\mathbf{A_r(q_r, q_e)}$ et $\mathbf{h_r(q_r, q_e, \dot{q}_r, \dot{q}_e)}$ utilisées dans la loi de commande (IV.2) avec les réseaux de neurones artificielles $\mathbf{A_rNN}$ et $\mathbf{h_rNN}$, respectivement. Nous utiliserons ensuite leurs sorties en addition avec la partie 'Proportionnelle et Dérivée' pour élaborer le premier contrôleur [Mao 2008b, Mao 2008a, Mao 2010b]:

$$\mathbf{\Gamma_{NN}} = \mathbf{A_rNN}\,\ddot{\mathbf{q}}_\mathbf{r}^\mathbf{d} + \mathbf{h_rNN}\,\dot{\mathbf{q}}_\mathbf{r}^\mathbf{d} + \mathbf{K_{pr}}\,\tilde{\mathbf{q}}_\mathbf{r} + \mathbf{K_{vr}}\,\dot{\tilde{\mathbf{q}}}_\mathbf{r} \qquad \text{(IV.26)}$$

Dans notre schéma de construction du réseau de neurones pour $\mathbf{A_rNN}$ et $\mathbf{h_rNN}$, nous avons utilisé pour chacun d'eux, un réseau de neurones à trois couches qui consistent en une couche d'entrée, une couche cachée et une couche de sortie (voir la Figure IV.12). Nous exposons dans l'annexe F les réglages adoptés pour chacun des deux réseaux de neurones $\mathbf{A_rNN}$ et $\mathbf{h_rNN}$.

Nous avons adopté l'algorithme de rétro propagation (back-propagation) pour effectuer un apprentissage supervisé du réseau de neurones [Gup 2003].

Comme nous l'avons présenté au Chapitre II, lors de l'apprentissage avec l'algorithme de rétro propagation, on peut distinguer deux phases, une phase directe (forward phase) et une phase de retour (backward phase).

Dans la phase directe, le signal d'entrée se propage dans le réseau couche après couche, produisant la réponse $\mathbf{Y}$ à la sortie du réseau (voir la Figure IV.12):

$$\mathbf{Y} = f_o(f_h(\mathbf{Xi} \cdot \mathbf{Wij}) \cdot \mathbf{Wjk}) \qquad \text{(IV.27)}$$

où, $\mathbf{Xi}$ est le signal d'entrée, $\mathbf{Y}$ est la sortie du réseau de neurones considéré ($\mathbf{A_r NN}$ ou $\mathbf{h_r NN}$). Dans notre schéma de contrôle, les signaux d'entrée de la couche d'entrée de $\mathbf{A_r NN}$ sont les positions angulaires rigides et élastiques des deux bras: $[\theta_1, \theta_2, f_1, \alpha_1, f_2, \alpha_2]^{\mathrm{T}}$. Pour $\mathbf{h_r NN}$, les entrées sont les positions et vitesses angulaires, rigides et flexibles des deux bras: $[\theta_1, \theta_2, f_1, \alpha_1, f_2, \alpha_2, \dot{\theta}_1, \dot{\theta}_2, \dot{f}_1, \dot{\alpha}_1, \dot{f}_2, \dot{\alpha}_2]^{\mathrm{T}}$. La somme pondérée des sorties des deux première couches est donnée par le produit $\mathbf{Xi} \cdot \mathbf{Wij}$. Les éléments Wij_{ij} représentent les poids entre les neurones i de la couche d'entrée et les neurones j de la couche cachée. Les éléments Wjk_{jk} représentent les poids entre les neurones j de la couche cachée et les neurones k à la couche de sortie.

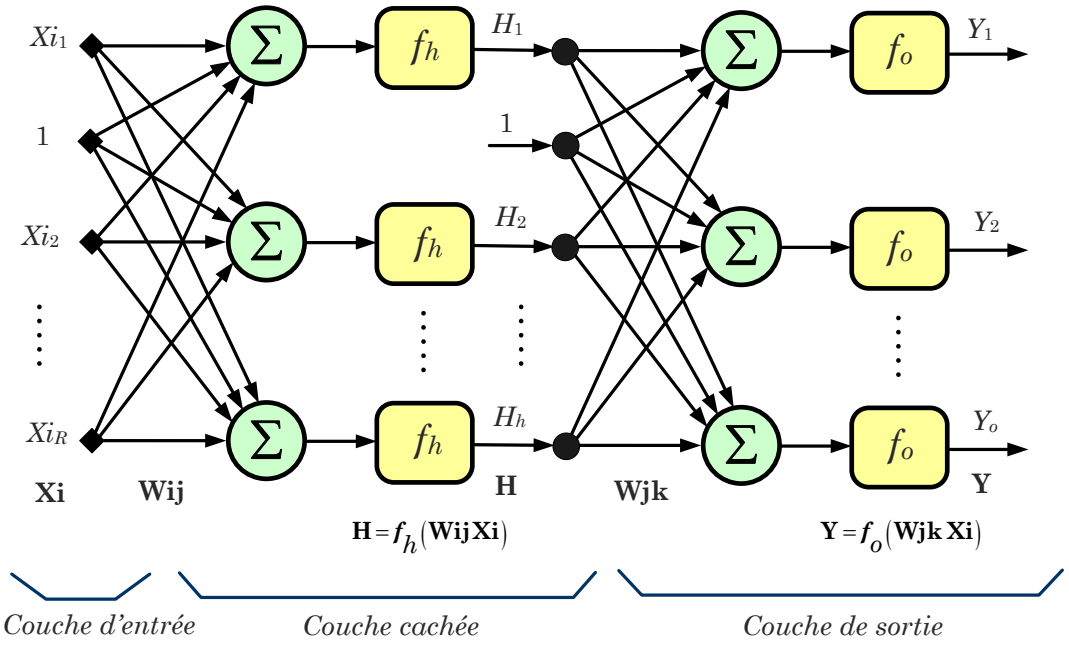

Figure IV.12 Réseau de neurones de type MLP à une couche cachée

Dans notre travail, nous avons considéré la fonction f_o comme une fonction linéaire et la fonction f_h comme une fonction tangente hyperbolique exprimée par:

$$f_h(x) = \frac{2}{1+e^{-2x}} - 1 \tag{IV.28}$$

Les réponses des réseaux $\mathbf{A_r NN}$ et $\mathbf{h_r NN}$ produits, sont comparées avec les réponses des fonctions $\mathbf{A_r}$ et $\mathbf{h_r}$, respectivement. Les signaux d'erreurs, ainsi générés, sont alors propagés en sens inverse (de la sortie vers l'entrée) à travers le réseau et forment la phase de retour.

Dans la phase de retour, la loi de réglage des poids 'delta rule learning' utilise l'écart entre la sortie désirée et la sortie réelle du réseau pour modifier les poids et ainsi réduire l'erreur.

L'entrainement est, ici, effectué 'hors ligne' de sorte à ne pas perturber le contrôle en temps réel du système. Les poids du réseau sont ajustés de façon à minimiser la fonction d'erreur (ou fonction de coût) suivante [Mao 2008b, Mao 2008a, Mao 2010b]:

$$\mathbf{E_{NN}} = \frac{1}{2}(\mathbf{Y^d} - \mathbf{Y})^2 \tag{IV.29}$$

où, $\mathbf{Y^d}$ et $\mathbf{Y}$ sont les sorties désirées et réelles, respectivement, du réseau de neurones considéré ($\mathbf{A_r NN}$ ou $\mathbf{h_r NN}$).

La loi de réglage des poids 'delta rule learning' pour les neurones dans la couche de sortie est donnée par [Arb 2003]:

$$\delta k_k = Y_k^d - Y_k \tag{IV.30}$$

Les poids d'interconnexion Wjk_{jk} entre la couche de sortie et la couche cachée sont changés à partir de la fonction d'erreur par une quantité:

$$\Delta Wjk_{jk} = \mu \cdot \delta k_k \cdot H_j \tag{IV.31}$$

où, μ est le taux d'apprentissage (learning rate) et H_j est la sortie du j^{ieme} neurone dans la couche cachée.

La loi de réglage des poids pour les neurones dans la couche cachée est donnée par:

$$\delta j_j = (1 - H_j^2) \cdot \sum_k (\delta k_k \cdot Wjk_{jk}) \qquad \text{(IV.32)}$$

Les poids d'interconnexion Wjk_{jk} entre la couche cachée et la couche d'entrée sont changés par une quantité:

$$\Delta Wij_{ij} = \mu \cdot \delta j_j \cdot Xi_i \qquad \text{(IV.33)}$$

Nous avons entrainé les réseaux de neurones correspondants à $\mathbf{A_r NN}$ et $\mathbf{h_r NN}$ par le biais de différentes trajectoires dynamiques: sinusoïdales, gaussiennes etc.

Nous avons fait particulièrement attention au critère d'arrêt. Nous considérons, en effet, que le simple fait de fixer le nombre d'itérations ou bien laisser l'entrainement se faire jusqu'à atteindre une certaine erreur prédéfinie, n'est pas recommandable.

La raison est que nous voulons que notre réseau de neurones se comporte bien face à un jeu de données test, c'est-à-dire nous voulons que le réseau réponde convenablement face à des trajectoires qu'il n'a pas apprises précédemment lors de la phase d'entrainement. Cette propriété du réseau est connue sous le nom de 'généralisation' [Bis 1995].

Nous voulons donc entrainer notre réseau de neurones de façon à ce qu'il ait une bonne capacité de généralisation.

L'erreur dans le jeu de données d'entrainement décroit avec les itérations quand le réseau de neurones a assez de degrés de liberté (nombre de couches et ou nombre de neurones par couches) pour représenter la courbe des entrées/sorties.

Cependant en augmentant le nombre d'itérations, on peut amener le réseau à se rappeler du jeu d'entrainement (le réseau répond de façon exacte mais uniquement face aux données d'entrainement), plutôt que d'essayer de trouver les non linéarités ou singularités sous-jacentes du système. Ce problème est connu dans les modèles statistiques sous le nom de sur-apprentissage (overfitting ou overtraining).

Pour éviter ce problème nous avons utilisé la méthode de validation croisée (cross validation). Dans ce cas, les performances sur un jeu de données de validation (données entrées/sorties obtenus à partir de trajectoires que le réseau n'a encore jamais rencontrées) doivent être mesurées régulièrement, durant la phase d'apprentissage [Dre 2005].

Pour notre part, nous avons effectué une mesure toutes les cinquante passes sur le jeu de données d'entrainement. L'apprentissage est stoppé quand les performances sur le jeu de validation commencent à diminuer, malgré le fait que les performances sur le jeu d'entrainement continuent d'augmenter.

De cette manière nous obtenons un réseau qui se comporte bien face aux données d'apprentissage mais aussi face à des données qu'il n'a pas encore rencontrées. Le réseau de neurones ainsi obtenu garde toute sa capacité de généralisation.

IV.4.2 Construction de la commande adaptative neuronale

Supposons, maintenant, qu'il existe des incertitudes structurées, c'est-à-dire supposons que les paramètres dynamiques estimés $\hat{\mathbf{X}}$, utilisés dans le modèle dynamique, soient différents des paramètres dynamiques réels $\mathbf{X}$ du robot flexible.

Ceci va conduire les fonctions $\mathbf{A_r}$ et $\mathbf{h_r}$ et donc les réseaux de neurones $\mathbf{A_r NN}$ et $\mathbf{h_r NN}$ a produire une réponse fausse. Par conséquent les couples articulaires, calculés à partir de l'équation (IV.26), seront erronés.

Supposons, de plus, qu'il existe des incertitudes non structurées dues à des phénomènes physiques non modélisés comme les frottements par exemple. Une équation de mouvement, plus générale, du robot flexible plan horizontal, est alors donnée par:

$$\mathbf{L_r \Gamma} = \mathbf{A(q)\ddot{q}} + \mathbf{B(q)\dot{q}\dot{q}} + \mathbf{C(q)\dot{q}^2} + \mathbf{K q} + \mathbf{F(q,\dot{q})} \qquad \text{(IV.34)}$$

où, $\mathbf{F(q,\dot{q})}$ représentent les incertitudes non structurées, incluant les frottements (visqueux et dynamiques) et autres perturbations.

Nous allons alors ajouter un second contrôleur au système de commande pour réduire l'écart entre le modèle et le système réel et compenser les incertitudes structurées et non structurées rencontrées. Ce second contrôleur sera basé sur un réseau de neurones adaptatif [Mao 2007, Mao 2009c].

L'idée fondamentale dans l'utilisation du réseau de neurones adaptatif dans le deuxième contrôleur est de produire une sortie qui forme une partie du couple articulaire global. Cette partie du couple est utilisée pour guider le mouvement des articulations du robot de façon à ce qu'ils correspondent à la trajectoire articulaire désirée. L'erreur entre les valeurs des positions et vitesses articulaires désirées et réelles est alors utilisée pour entrainer 'en ligne' le réseau de neurones du deuxième contrôleur.

Dans notre schéma de construction du réseau de neurones pour la commande adaptative, nous utilisons aussi trois couches. Des fonctions sigmoïdes et linéaires sont utilisées dans la couche cachée et la couche de sortie, respectivement (voir l'annexe F).

Les signaux d'entrée de la première couche sont les positions et vitesses angulaires des deux bras: $[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T$ et les signaux de sorties $\mathbf{Y}$ de la dernière couche sont les couples articulaires: $\mathbf{\Gamma_{AN}} = [\Gamma_{AN_1}, \Gamma_{AN_2}]^T$ (voir Figure IV.13).

L'apprentissage est, ici, effectué 'en ligne' et les paramètres du réseau sont ajustés de façon à minimiser la fonction d'erreur (fonction de coût) suivante:

$$\mathbf{E_{AN}} = \frac{1}{2}(\mathbf{Y^d} - \mathbf{Y})^2 = \frac{1}{2}(\mathbf{K_{pn}}\,\tilde{\mathbf{q}}_\mathbf{r} + \mathbf{K_{vn}}\dot{\tilde{\mathbf{q}}}_\mathbf{r})^2 \qquad \text{(IV.35)}$$

où, $\mathbf{Y^d}$ et $\mathbf{Y}$ sont les sorties désirées et réelles, respectivement, du réseau de neurones, $\mathbf{K_{pn}}$ et $\mathbf{K_{vn}}$ sont des matrices constantes positives définies de gain.

Comme l'apprentissage s'effectue en ligne, nous devons absolument réduire son temps de calcul. Le taux d'apprentissage (learning rate) est réglé en fonction de l'apprentissage du réseau, les minimums locaux potentiels et l'amplitude du changement des poids. Ce taux peut alors être trop grand ou trop petit par rapport à ce que nécessite le réseau.

Un facteur d'inertie (momentum factor) η est alors utilisé pour accompagner le réseau lors de la phase d'apprentissage [Kro 1996]:

$$\Delta \mathbf{W}(t+1) = \mu \cdot \frac{\partial \mathbf{E}}{\partial \mathbf{W}} + \eta \cdot \Delta \mathbf{W}(t) \qquad \text{(IV.36)}$$

où, **W** désigne **Wij** ou **Wjk**, t est l'indice de la $n^{\text{ième}}$ présentation ($n^{\text{ième}}$ passe) et η est la constante qui détermine l'effet du changement de poids précédent.

Quand aucun facteur d'inertie n'est utilisé, il peut se passer beaucoup de temps avant que le minimum ne soit atteint avec un petit taux d'apprentissage, alors que si on utilise un grand taux d'apprentissage on risque de ne jamais atteindre exactement le minimum à causes des oscillations (on reste autour du minimum sans jamais l'atteindre car le pas est trop grand). Quand on ajoute le terme relatif au facteur d'inertie le minimum est atteint plus rapidement (car le pas devient variable et dépend de la grandeur du changement précédent). Ceci conduit la commande neuronale adaptative à produire une réponse plus rapide. Un meilleur contrôle est alors réalisé.

L'architecture de contrôle globale de la commande hybride appliquée au robot flexible est présentée dans la Figure IV.13. Elle est décrite par l'équation suivante:

$$\mathbf{\Gamma_H} = \mathbf{\Gamma_{NN}} + \mathbf{\Gamma_{AN}} \qquad \text{(IV.37)}$$

où, $\mathbf{\Gamma_H}$ est la sortie du contrôleur hybride (couple articulaire); $\mathbf{\Gamma_{NN}}$ est la sortie du premier contrôleur, basée sur le modèle neuronal du robot, comme définie par l'équation (IV.26); $\mathbf{\Gamma_{AN}}$ est la sortie du second contrôleur, basée sur la commande neuronale adaptative.

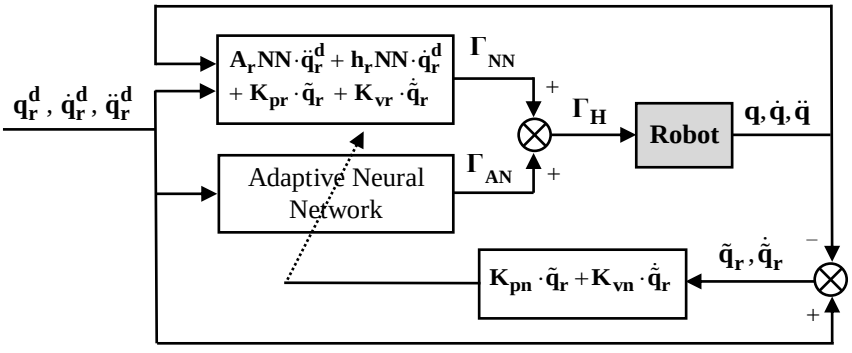

Figure IV.13 Architecture de la commande hybride

IV.4.3 Tests et simulations

Les performances de la commande hybride proposée, est testée sur une trajectoire dynamique ayant une accélération de type 'Bang-Bang' avec une vitesse initiale et finale nulles:

$$\ddot{\theta}_1^{\mathrm{d}}(t) = \ddot{\theta}_2^{\mathrm{d}}(t) = \begin{cases} 16\pi/T^2 & \text{for} \quad t \in \left[0, T/4\right] \\ -16\pi/T^2 & \text{for} \quad t \in \left[T/4, 3T/4\right] \\ 16\pi/T^2 & \text{for} \quad t \in \left[3T/4, T\right] \end{cases} \tag{IV.38}$$

avec, $\theta_1^{\mathrm{d}}(0) = \theta_2^{\mathrm{d}}(0) = 0$ et $\dot{\theta}_1^{\mathrm{d}}(0) = \dot{\theta}_2^{\mathrm{d}}(0) = 0$.

Nous donnons aussi à titre indicatif les résultats de la commande non linéaire à paramètres fixes décrite par l'équation (IV.2). Nous avons pris $T = 30$ sec. La vitesse angulaire maximum est atteinte pour $t = T/4$ et pour $t = 3T/4$ et sa valeur absolue est $4\pi/T$ rad/sec ou bien encore 24 deg/sec.

Les différentes matrices de gains sont ajustées de la manière suivante:

- dans la loi de commande non linéaire (IV.2) et dans le premier contrôleur de la commande hybride (IV.26): $\mathbf{K_{pr}} = \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \end{bmatrix}$ et $\mathbf{K_{vr}} = \begin{bmatrix} 4 & 0 \\ 0 & 0.8 \end{bmatrix}$,
- dans la fonction d'erreur (IV.35) du deuxième contrôleur de la commande hybride: $\mathbf{K_{pn}} = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.4 \end{bmatrix}$ et $\mathbf{K_{vn}} = \begin{bmatrix} 4.8 & 0 \\ 0 & 1.5 \end{bmatrix}$.

Pour tester la robustesse paramétrique de la commande hybride, mettons nous dans les conditions de l'équation (IV.25). C'est-à-dire supposons que les paramètres dynamiques estimés $\hat{\mathbf{X}}$, utilisés dans le modèle dynamique, soient très différents des paramètres dynamiques réels du robot flexible.

Ceci va conduire les fonctions $\mathbf{A_r}$ et $\mathbf{h_r}$ et donc les réseaux de neurones $\mathbf{A_r NN}$ et $\mathbf{h_r NN}$ à produire une réponse erronée. Par conséquent, les couples articulaires, calculés à partir de l'équation (IV.26), seront erronés. Nous verrons par la suite comment la commande adaptative neuronale corrigera cette erreur.

Notre but ici est de simuler une importante erreur due soit à la mauvaise estimation des paramètres du modèle soit à la non prise en compte dans le modèle d'un phénomène physique soit encore une perturbation externe. Nous supposons que si notre commande peut prendre en charge cette importante erreur, elle pourra à fortiori prendre en charge une erreur plus petite.

Dans le but de simplifier les tests, les paramètres dynamiques du robot sont mal estimés de la même manière dans l'équation (IV.25). Même si ce n'est pas toujours le cas en pratique, ceci n'affectera en rien les performances de la commande hybride, car dans la loi de commande neuronale adaptative on considère l'erreur globale c'est-à-dire la résultante de toutes les erreurs.

Les Figures IV.14 à IV.23 illustrent les résultats obtenus avec la commande hybride appliquée au robot à deux bras flexibles. Ces figures décrivent l'évolution de: la position angulaire, l'erreur en position, la flèche, la vitesse angulaire et l'erreur sur la vitesse angulaire, pour l'articulation 1 et 2, respectivement.

La trajectoire désirée (consigne) est donnée dans les Figures IV.14, IV.17, IV.19 et IV.22 en pointillés. Les résultats de la commande non linéaire à paramètres fixes, décrite par l'équation (IV.2), sont reportés en ligne discontinue pour comparaison.

Le Tableau IV.3 et le Tableau IV.4 présentent l'erreur maximale ainsi que l'erreur quadratique moyenne (RMS) de la position et de la vitesse angulaire obtenus avec les deux types de commande utilisés.

La trajectoire désirée impose une variation rapide de l'accélération aux instants $t_1 = T/4 = 7.5$ sec et $t_2 = 3T/4 = 22.5$ sec. Ce changement radicale, d'une accélération positive à négative pour l'instant t_1 et de négative à positive pour l'instant t_2, éprouve grandement le contrôleur. On peut voir son impact sur le contrôle de la vitesse angulaire dans les Figures IV.17 et IV.22. Cependant le suivi de la trajectoire, en vitesse, obtenu avec la commande hybride est fidele à la consigne. En comparaison, on note que la trajectoire de la vitesse angulaire obtenue avec la commande non linéaire à paramètres fixes dévie fortement de la consigne.

D'après les Tableaux IV.3 et IV.4, on note que l'erreur en vitesse, obtenue avec la commande non linéaire à paramètres fixes, atteint 0.19 rad/sec (10.9 deg/sec) pour la première articulation et 0.13 rad/sec (7.7 deg/sec) pour la seconde.

La commande hybride donne de meilleurs résultats, avec une erreur en vitesse inférieure à 0.0052 rad/sec (0.3 deg/sec) pour la première articulation et inférieure à 0.009 rad/sec (0.5 deg/sec) pour la seconde (voir les Figures IV.18 et IV.23).

Pour le contrôle en position (voir les Figures IV.14 et IV.19), on note que la trajectoire angulaire obtenue avec la commande hybride suit bien la trajectoire désirée; avec une erreur ne dépassant pas 0.0031 rad (0.2 deg) pour la première et la deuxième articulation (voir les Figures IV.15 et IV.20). Quand à la commande à paramètres fixes, l'erreur obtenue sur la position angulaire, dépasse 0.43 rad (24 deg) pour la première articulation et 0.34 rad (19 deg) pour la seconde (voir les Tableaux IV.3 et IV.4).

D'après les Figures IV.16 et IV.21, on voit que la flexion est bien contenue avec les deux types de commandes, avec une flèche inférieure à 0.05 m pour la première articulation et inférieur à 0.02 m pour la seconde.

On remarque la présence d'oscillations plus nombreuses avec la commande hybride car le couple articulaire généré par la commande adaptative neuronale varie plus vite qu'avec la commande non linéaire à paramètres fixes. Cependant l'amplitude des oscillations n'augmente pas et la flèche reste bornée.

D'après les résultats obtenus quelques conclusions intéressantes peuvent être déduites. L'utilisation du contrôleur basé sur le modèle du robot flexible ($\mathbf{\Gamma_{NN}}$) tout seul, réduit la précision de la commande dans la présence d'incertitude structurées et non structurées. Alors que, l'utilisation du contrôleur adaptatif ($\mathbf{\Gamma_{AN}}$) tout seul, accroit la flèche des bras et aucun amortissement des oscillations n'est réalisé ce qui peut mener à un système instable. La combinaison de ces deux techniques de contrôle donne un bon compromis entre stabilité et précision. Les résultats obtenus en simulation montrent l'efficacité de la commande hybride proposée.

De plus, l'utilisation des réseaux de neurones dans la commande nous a permis de réduire d'une manière significative le temps de calcul de la commande puisque la simulation de la trajectoire sur la même configuration matérielle que précédemment dure moins d'une seconde. Ceci, ajouté au fait que cette commande ne nécessite qu'un pas d'itération de 0.01 sec, nous rend très confiant quand au comportement de cette commande dans la pratique.

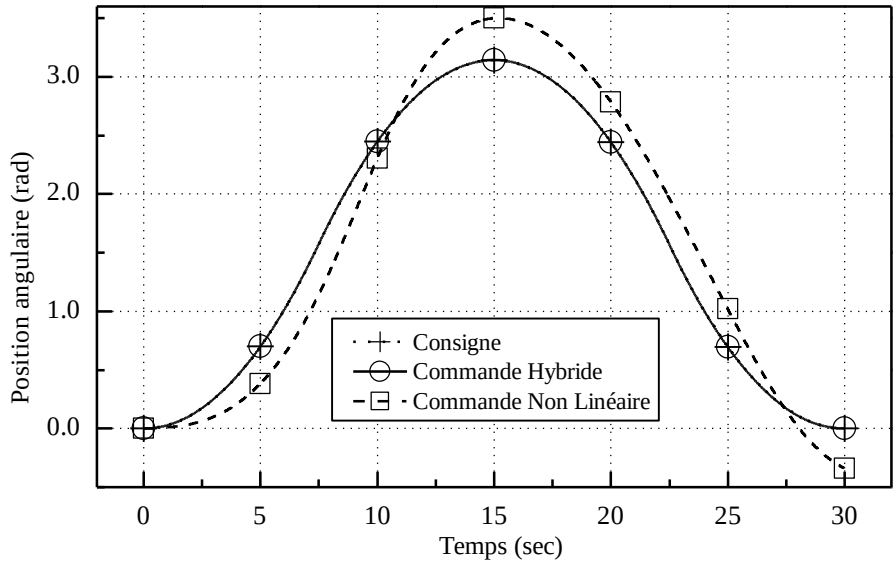

Figure IV.14 Evolution de la position angulaire θ_1 (rad)

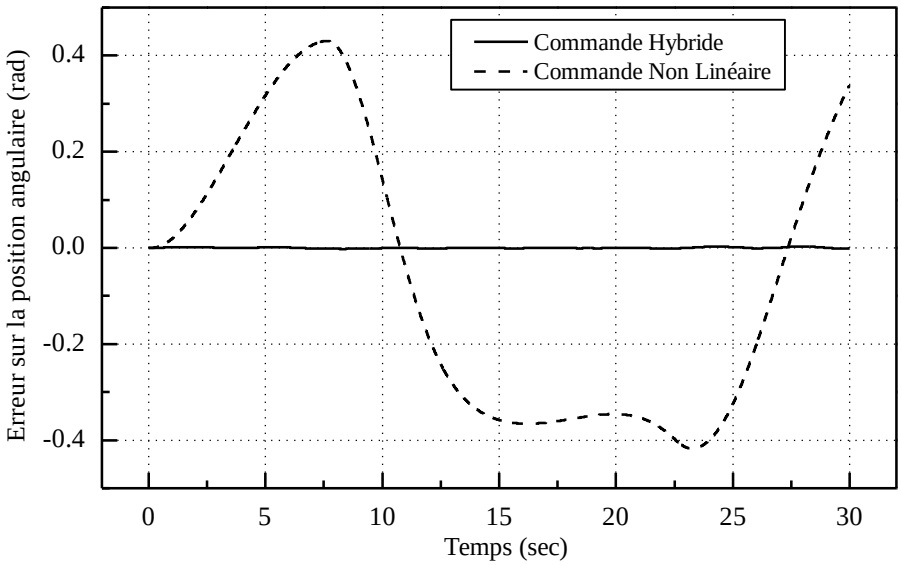

Figure IV.15 Evolution de l'erreur sur la position angulaire $\tilde{\theta}_1$ (rad)

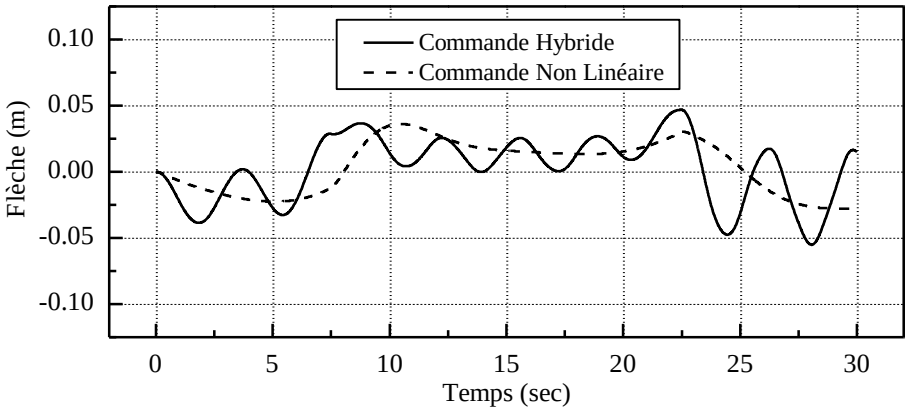

Figure IV.16 Evolution de la flèche f_1 (m)

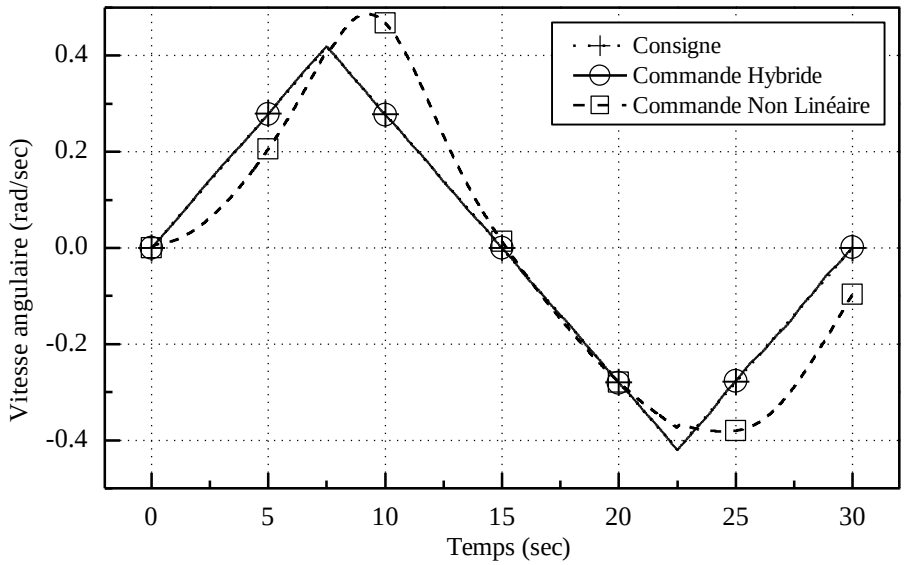

Figure IV.17 Evolution de la vitesse angulaire $\dot{\theta}_1$ (rad/sec)

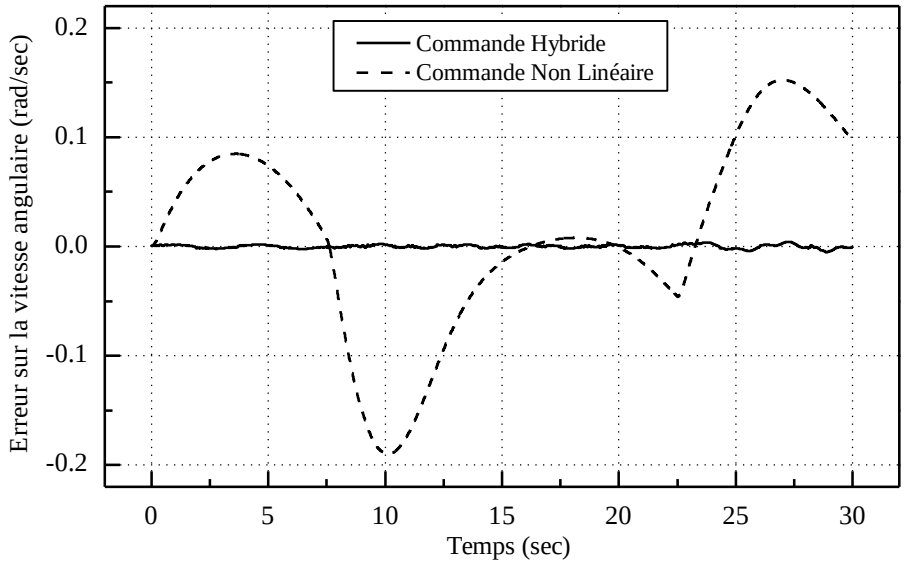

Figure IV.18 Evolution de l'erreur sur la vitesse angulaire $\tilde{\dot{\theta}}_1$ (rad/sec)

Tableau IV.3

ERREUR DE SUIVI DE TRAJECTOIRE POUR L'ARTICULATION 1

	Erreur Maximum		Erreur RMS	
Variable	θ_1 (rad)	$\dot{\theta}_1$ (rad/sec)	θ_1 (rad)	$\dot{\theta}_1$ (rad/sec)
Commande Hybride	3.07×10^{-3}	5.19×10^{-3}	1.14×10^{-3}	1.60×10^{-3}
Commande Non Linéaire	4.31×10^{-1}	1.90×10^{-1}	2.98×10^{-1}	8.92×10^{-2}

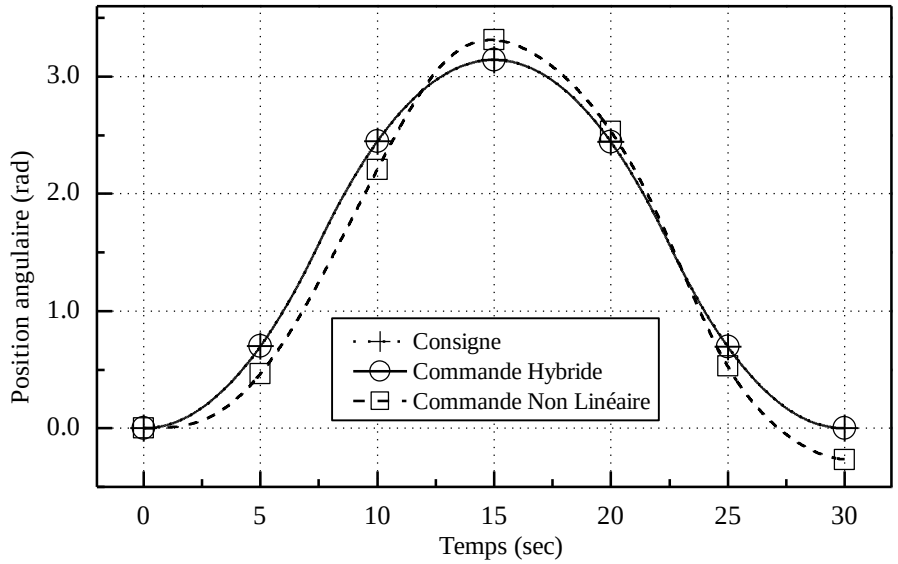

Figure IV.19 Evolution de la position angulaire θ_2 (rad)

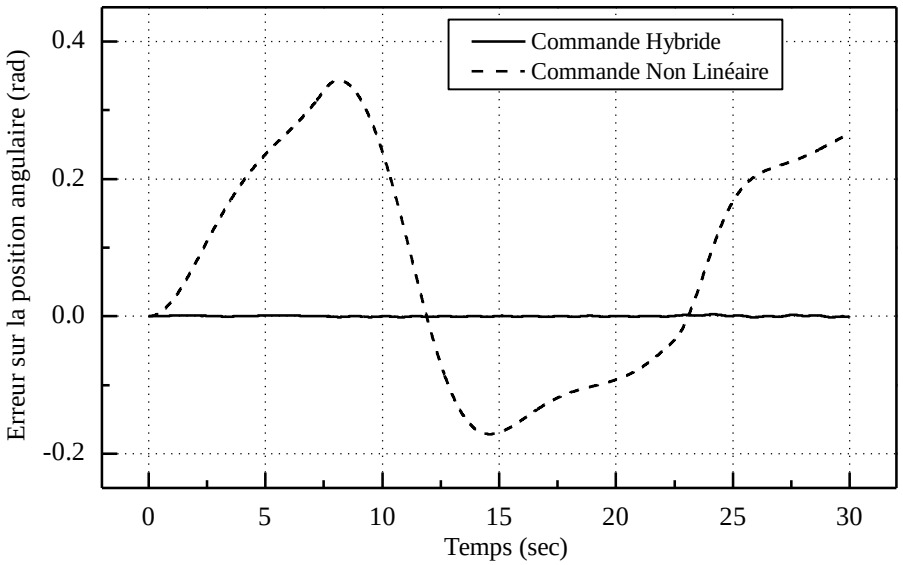

Figure IV.20 Evolution de l'erreur sur la position angulaire $\tilde{\theta}_2$ (rad)

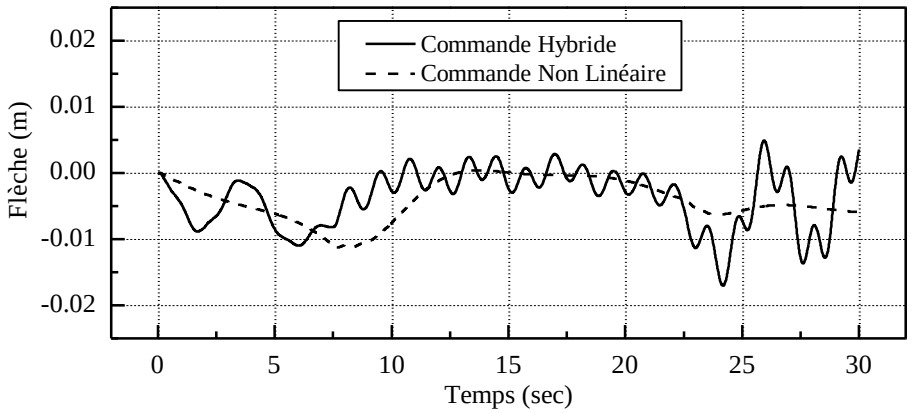

Figure IV.21 Evolution de la flèche f_2 (m)

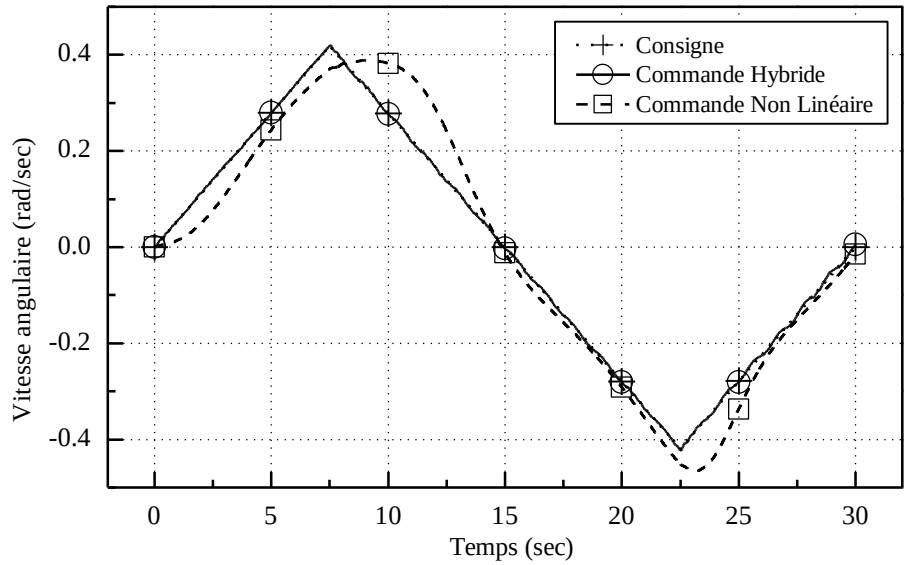

Figure IV.22 Evolution de la vitesse angulaire $\dot{\theta}_2$ (rad/sec)

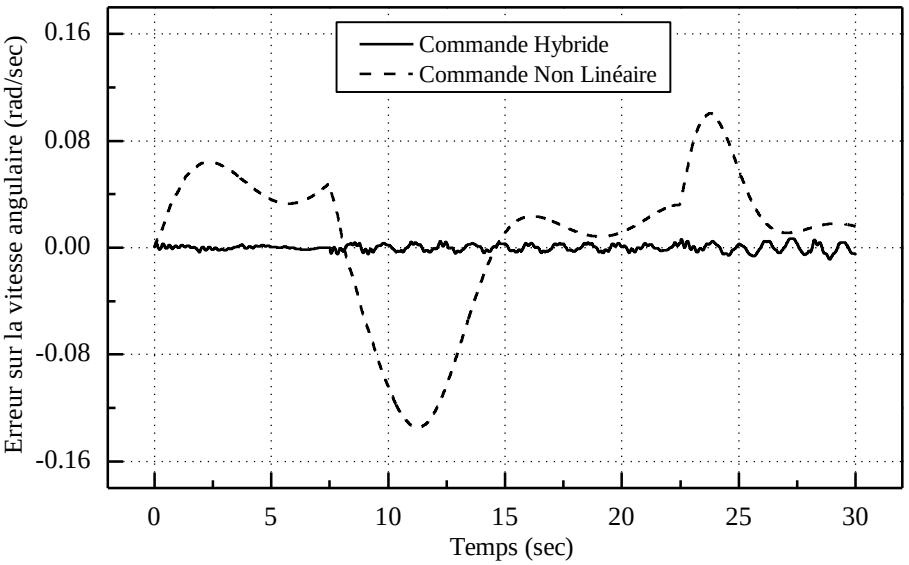

Figure IV.23 Evolution de l'erreur sur la vitesse angulaire $\dot{\tilde{\theta}}_2$ (rad/sec)

Tableau IV.4

ERREUR DE SUIVI DE TRAJECTOIRE POUR L'ARTICULATION 2

	Erreur Maximum		Err. Quadratique Moyenne	
Variable	θ_2 (rad)	$\dot{\theta}_2$ (rad/sec)	θ_2 (rad)	$\dot{\theta}_2$ (rad/sec)
Commande Hybride	2.85×10^{-3}	8.67×10^{-3}	8.43×10^{-4}	2.55×10^{-3}
Commande Non Linéaire	3.44×10^{-1}	1.35×10^{-1}	1.83×10^{-1}	5.53×10^{-2}

Nous donnons également dans les Figures IV.24 et IV.25, à titre indicatif, l'évolution des couples articulaires, appliqués à la première et à la deuxième articulation, respectivement. Pour comparaison, les couples articulaires générés par la commande hybride sont donnés en lignes continues et ceux générés par la commande non linéaire sont donnés en lignes discontinues.

Nous retrouvons, ici, une caractéristique physique des robots à bras flexibles. Nous remarquons, que l'évolution de la flèche (voir les Figures IV.16 et IV.21) suit en sens inverse l'évolution du couple articulaire (voir les Figures IV.24 et IV.25). En effet, à chaque fois qu'un couple articulaire est appliqué, une flèche apparaît au bout du bras flexible. L'amplitude de la flèche est proportionnelle à celle couple mais elle est de sens inverse, car elle est induite par l'inertie du bras et de la charge transportée en bout de bras. Ce résultat valide, au passage, le modèle du robot flexible que nous avons élaboré et qui décrit fidèlement ce comportement.

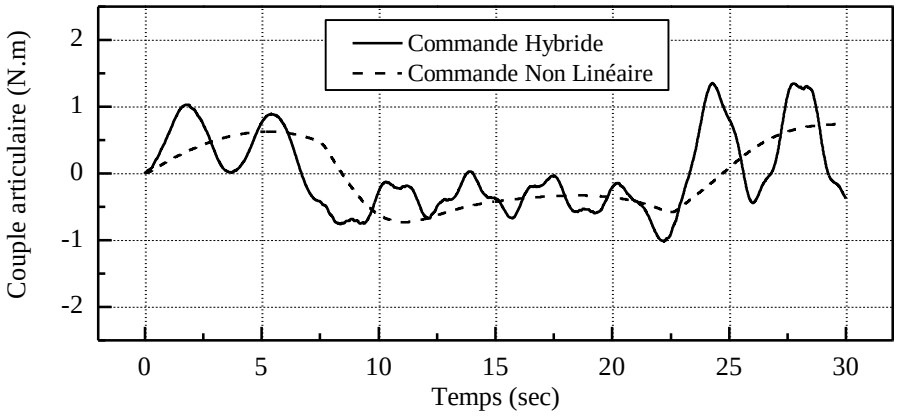

Figure IV.24 Evolution du couple articulaire Γ_1 (N.m)

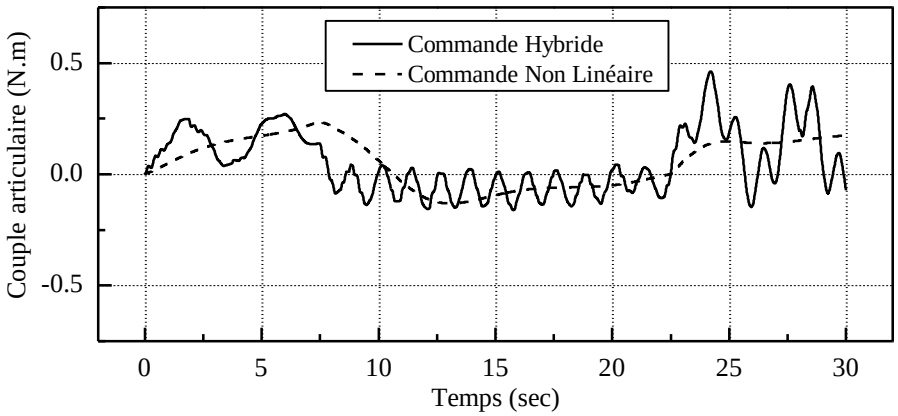

Figure IV.25 Evolution du couple articulaire Γ_2 (N.m)

Nous remarquons aussi que le couple, généré par les deux types de commandes, a une amplitude relativement faible. Ceci permet d'utiliser des moteurs de petites tailles qui consomment relativement peut d'énergie. Ce qui n'est pas le cas avec des robots rigides. La consommation peut constituer un point crucial, lorsque l'énergie est limitée, telle que dans les systèmes embarqués ou en aérospatial, où l'on trouve, d'ailleurs, la plupart des applications actuelles des robots flexibles.

IV.5 Conclusion

Une commande originale, efficace et robuste, a été présentée dans ce Chapitre, dans le but de contrôler la trajectoire articulaire des robots flexibles et d'amortir les vibrations des bras.

Dans un premier temps nous avons présenté une commande non linéaire basée sur le modèle dynamique non linéaire du robot flexible. Ici, on a utilisé les propriétés physiques du robot dans l'élaboration de la commande ce qui nous a permis d'avoir un contrôle qui suit la forme de la trajectoire de consigne tout en amortissant les vibrations dues à la flexion des bras.

Ensuite une étude de la robustesse de cette première loi de commande nous a contraints à ajouter une partie adaptative à la commande pour faire face aux incertitudes structurées et non structurées.

Enfin l'utilisation des réseaux de neurones en lieu et place des fonctions non linéaires du modèle puis dans l'élaboration de la partie adaptative de la commande, nous a permis de réduire significativement le temps de calcul et ainsi le temps de réponse de la commande et d'améliorer les performances de celle-ci, comme l'ont montrés, les tests réalisés.

Conclusion générale

Le travail que nous avons présenté porte sur le contrôle de processus dynamiques complexes et plus spécifiquement sur la commande de robots manipulateurs à structure flexible.

Notre but a été de rechercher des techniques de commandes intelligentes, qui augmentent les performances du contrôleur et réduisent son temps de réponse. L'idée principale, ici, a été de combiner deux techniques de commande, la commande à base de modèle de connaissances comme la commande non linéaire et la commande à base de modèle statistique comme la commande neuronale.

Pour aboutir à ce résultat nous avons divisé notre travail en quatre étapes. Au Chapitre I nous avons établi le modèle dynamique d'un robot plan horizontal à deux bras flexibles. Ceci nous a permis de tenir compte, dans la commande, des non linéarités de notre système et du couplage dynamique entre les variables rigides et élastique.

L'utilisation de l'étude dynamique du mouvement du robot flexible dans la loi de commande présente des avantages majeurs. En effet, celle-ci peut alors utiliser les informations extraites des équations du modèle dynamique ainsi que les propriétés physiques du robot (comme la passivité) pour effectuer un contrôle stable de la trajectoire et amortir les vibrations.

Dans le Chapitre II, nous avons relevé quelques propriétés fondamentales des réseaux de neurones artificiels, telles que la capacité d'approximation universelle, la parcimonie, la rapidité de calcul et l'apprentissage par les données mesurées sur le système réel. Nous avons ensuite utilisé ces propriétés dans l'élaboration de notre loi de commande.

Le Chapitre III nous a permis, après une brève revue bibliographique, de classer les commandes existantes en deux catégories. Les commandes à base de modèle de connaissances et les commandes à base de modèles paramétrés à apprentissage par les données.

Au Chapitre IV, nous avons montré la nécessité d'utiliser des commandes plus élaborées pour le contrôle des robots flexibles. En réponse, nous avons proposé une architecture de commande hybride, originale, qui regroupe les avantages des modèles à base de connaissances et des modèles statistiques. Puis nous avons effectué une série de test pour montrer l'efficacité et la robustesse de la commande proposée.

La loi de commande que nous avons présentée dans notre travail fonctionne par l'association de deux contrôleurs.

Le premier contrôleur et basé sur une approximation des fonctions non linéaires, du modèle dynamique utilisé dans la commande, par un réseau de neurones artificiels.

Le deuxième contrôleur est constitué d'une commande adaptative neuronale. L'apprentissage, ici, du réseau s'effectue en ligne et à partir des mesures réelles sur le processus à contrôler.

Le but du premier contrôleur est d'assurer un contrôle stable des variables rigide tout en amortissant les vibrations des variables élastiques. Le contrôleur adaptatif sert, quand à lui, à compenser les erreurs dues aux incertitudes structurées (incertitudes sur l'estimation des paramètres physiques) et non structurés (perturbations externes et phénomènes physiques non pris en charge par le modèle), améliorant ainsi la précision de la commande globale.

Les tests effectués en simulations sur un robot plan horizontal à deux bras flexibles ont été réalisés avec succès. Ils ont démontré la robustesse en performances de l'architecture de contrôle proposée faces à une erreur importante dans le modèle du robot. L'utilisation des réseaux de neurones nous a permis d'alléger considérablement la construction de la loi de commande et de réduire significativement son temps de calcul et ainsi sa réactivité. Ceci nous laisse confiant quant à l'efficacité de cette commande originale dans la pratique.

Les principales perspectives qui s'inscrivent dans le prolongement de ce travail concernent trois points.

Il serait intéressant d'étudier les performances des réseaux de neurones à base radiale dans l'approximation des fonctions du modèle dynamique ou dans la commande adaptative.

Une étude de stabilité de la commande globale, associant le premier et le deuxième contrôleur, reste à être effectuée.

En fin, il nous semble nécessaire de réaliser des tests expérimentaux, afin de valider et de mettre en valeur la commande originale proposée.

Références bibliographiques

[Ahm 2008] Ahmad M.A., Mohamed Z. and Hambali N., 'Dynamic modeling of a two-link flexible manipulator system incorporating payload', *3rd IEEE Conference on Industrial Electronics and Applications (ICIEA 2008)*, pp. 96-101, 2008.

[Ala 2007] Alam M.S. and Tokhi M.O., 'Hybrid fuzzy logic control with genetic optimization for a single-link flexible manipulator', *Engineering Applications of Artificial Intelligence*, DOI:10.1016/j.engappai.2007.08.002, 2007.

[And 1972] Anderson J.A., 'A simple neural network generating an interactive memory', *Mathematical Biosciences*, Vol. 14, pp.197-220, 1972.

[Arb 2003] Arbib M.A., *The Handbook of Brain Theory and Neural Networks*, The MIT press, USA, ISBN: 0-262-01197-2, 2003.

[Att 1999] Attari M., Boudjema F., Bouhedda M. and Bouallag S., 'A decentralized neural architecture based A/D converter with binary coded outputs', *Computer Standards and Interface*, Elsevier, Vol. 21, Issue 2, page 102, ISSN: 0920-5489, June 1999.

[Att 1995] Attari M., Boudjema F. and Heniche M., 'An artificial neural network to linearize a G (Tungsten vs. Tungsten 26% Rhenium) thermocouple characteristic in the range of zero to 2000°C', *IEEE International Symposium on Industrial Electronics (ISIE 1995)*, Vol. 1, pp. 176-180, Athens, Greece, 1995.

[Aza 2003] Azad A.K.M., Tokhi M.O. and Anand N. 'Teaching of control for complex systems trough simulation', *Proceedings of the 2003 ASEE/WFEO International Colloquium, American Society for Engineering Education*, 2003.

[Ben 1991] Benallegue A., 'Contribution à la commande dynamique adaptative des robots manipulateurs rapides', *Thèse de Doctorat de l'Université Pierre et Marie Curie*, Paris VI, France, 1991.

[Bal 1978] Balas M.J., 'feedback control of flexible system', *IEEE Transaction on Automatic Control*, Vol. 23, No. 4, pp. 673-679, 1978.

[Bar 1993] Barron A., 'Universal approximation bounds for superposition of a sigmoidal function', *IEEE Transactions on Information Theory*, Vol. 39, pp. 930-945, 1993.

[Bar 1988] Barbieri E. and Ozguner U., 'Unconstrained and constrained mode expansions for a flexible slewing link', *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 110, pp. 416-421, 1988.

[Bay 1989] Bayo E., Papadopoulus P., Serna M.A. and Stubbe J., 'Inverse dynamics and kinematics of a multi-link elastic robot : An iterative frequency domain approach', *International Journal of Robotics Research*, Vol. 8, No 6, pp. 49-62, 1989.

[Bay 1987] Bayo E., 'A finit element approach to control the end-point motion of a single-link flexible robot', *Journal of Robotic Systems*, Vol. 4, pp. 63-75, 1987.

[Bis 1995] Bishop C.M., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[Boo 1987] Book W.J. and Hastings G.G., 'A linear dynamic model for flexible robotic manipulators", *IEEE Control System Magazine*, Vol. 7, No 1, pp. 61-64, 1987.

[Boo 1983] Book W.J. and Majette M., 'Controller design for flexible distributed parameter mechanical arms via combined state space and frequency domain technics', *Transaction on ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 105, p 245-254, 1983.

[Boo 1975] Book W.J., Maizza-Neto O. and Whitney D.E., 'Feedback control of two beam, two joint systems with distributed flexibility', *Dynamic Systems, Measurement, and Control*, Vol. 97, No. 4, pp. 424-431, 1975.

[Can 1984] Cannon R.H.Jr and Schmitz E., 'Initial experiments on the end-point control for a flexible one-link robot', *International Journal of Robotics Research*, Vol. 3, N° 3, pp. 62-75, 1984.

[Che 1990] Chedmail P., 'Synthèse de robots et de sites robotisés. Modélisation de robots souples', *Thèse de Doctorat*, ENSM, Nantes, France, 1990.

[Che 1989] Chedmail P. and Khalil W., 'Non linear decoupling control of flexible robots', *4th International Conference on Advanced Robotics*, Columbus, pp. 138-145, 1989.

[Che 2003] Cheng X.P. and Patel R.V., 'Neural network based tracking control of a flexible macro–micro manipulator system', *Neural Networks*, Vol. 16, pp. 271-286, 2003.

[Che 2000] Cheong J., Chung W. and Youm Y. 'Bandwidth modulation of rigid subsystem for the class of flexible robots', *IEEE International Conference on Robotics and Automation*, San Francisco, USA, pp. 1478-1483, 2000.

[Dam 1995] Damaren C.J., 'Passivity analysis for flexible multilink space manipulators', *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 2, pp. 272-279, 1995.

[Del 1991] De Luca A. and Siciliano B., 'Closed-form dynamic model of planar multilink lightweight robots', *IEEE Transaction on Systems, Man and Cybernetics*, Vol. 21, No 4, pp. 826-839, 1991.

[Del 1990] De Luca A., Lanari L., Lucibello P., Panzieri S. and Ulivi G., 'control experiments on a two-link robot with a flexible forearm', *29th IEEE Conference on Decision and Control*, Honolulu, USA, 1990.

[Des 1988] De Shutter J., Van Brussels H., Adam M., Froment A. and Faillot J.L., 'Control of flexible robots using generalized non-linear decoupling', *Proceedings of IFAC, SYROCO*, pp. 113-118, 1988.

[Dom 2007] Dombre E. and Khalil W., *Robot Manipulators. Modeling, Performance Analysis and Control*, ISTE Ltd., London, U.K, 2007.

[Dom 1988] Dombre E. and Khalil W., *Modélisation et commande des robots*, Hermes, Paris, France, 1988.

[Dre 2005] Dreyfus G., *Neural Networks Methodology and Applications*, Springer, Germany, ISBN-10 3-540-22980-9, 2005.

[Ela 2001] Elahinia M.H. and Ashrafiuon H., 'Nonlinear control of a shape memory alloy actuated manipulator', *ASME Journal Of Vibration Acoustic*, Vol. 123, No. 4, pp. 487-495, 2001.

[Er 2009] Er M.J. and Zhou Y. *Theory and Novel Applications of Machine Learning*. In-Tech, Croatia, ISBN: 978-3-902613-55-4.

[Far 2009] Farrel J.A. and Polycarpou M.M., *Adaptive approximation based control*, Wiley-Interscience Inc., New Jersey, USA, 2009.

[Gau 1991] Gautier M., Pham C.M. and Chedmail P., 'Energy based determination of identifiable parameters of flexible link robots', *Proceedings of SYROCO*, pp. 487-495, 1991.

[Gau 1990] Gautier M. 'Contribution à la modélisation et à l'identification des robots', *Thèse de Doctorat d'Etat de l'Université de Nantes,* ENSM, France, 1990.

[Giu 2001] Giurgiutiu V., Jichi F., Berman J. and Kamphaus J.M. 'Theoretical and experimental investigation of magnetostrictive composite beams', *Smart Material Structure*, Vol. 10, No 5, pp. 934-935, 2001.

[Gros 1976] Grossberg S., 'Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural features detectors', *Biological Cybernetics*, Vol. 23, pp. 121-134, 1976.

[Gu 2005] Gu D.W., Petkov P.Hr. and Konstantinov M.M., *Robust control design with Matlab*,. Springer, p. 335, 2005.

[Gua 1989] Guay P., 'Modèle mécanique des mécanismes : Formulation à l'aide de groupes de Lie des déplacements', *Thèse de Doctorat en mécanique*, INSA de Lyon, France, 1989.

[Gui 1995] Guihard M., 'Etude de lois de commande adaptatives d'actionneurs pneumatiques pour le contrôle dynamique d'un robot marcheur', *Thèse de Doctorat de l'Université Pierre et Marie Curie*, Paris, France, 1995.

[Gup 2003] Gupta M.M., Jin L. and Homma N. *Static and Dynamic Neural Networks*, Wiley-Interscience, USA, ISBN: 0-471-21948-7, 2003.

[Han 2004] Hangos K.M., Bokor J. and Szederkényi G. *Analysis and Control of Nonlinear Process Systems*, Springer, London, GB, 2004.

[Hil 1991] Hillsley K.L. and Yurkovitch S. 'Vibration control of two link flexible robot arm', *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 2121-2126, 1991.

[Hop 1982] Hopfield J.J., 'Neural networks and physical systems with emergent collective computational abilities'. *Proceedings of the National Academy of Sciences*, Vol. 79, pp. 2554-2558, 1982.

[Hor 1991] Hornik K., 'Approximation capabilities of multilayer feedforward networks', *Neural Networks*, Vol. 4, pp. 251-257, 1991.

[Hor 1990] Hornik K., Stinchcombe M. and White H., 'Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks', *Neural Networks*, Vol. 3, pp. 551-560, 1990.

[Hor 1989] Hornik K., Stinchcombe M. and White H., 'Multilayer feedforward networks are universal approximators', *Neural Networks*, Vol. 2, pp. 359-366, 1989.

[Hor 1990] Horowitz R. and Sadegh N., 'An exponentially stable adaptive control law for robot manipulators', *IEEE Transactions on Robotics and Automation*, Vol. 6, n° 4, pp. 491-496, 1990.

[Hu 2007] Hu T., Zhu J. and Sun Z., 'Robust adaptive neural control of a Class of MIMO Nonlinear Systems' Tsinghua Science & Technology, Vol. 17, pp. 14-21, 2007.

[Jon 1990] Jonker B., 'A finit element analysis of flexible manipulators', *International Journal of Robotics Research*, Vol. 9, No 4, pp 59-74, 1990.

[Kas 1998] Kasabov N.K. *Foundations of neural networks, fuzzy systems, and knowledge engineering*, The MIT press, USA, ISBN: 0-262-11212-4, 1998.

[Kec 2001] Kecman V., *Learning and soft computing: support vector machines, neural networks and fuzzy logic models*, The MIT press, UK, ISBN: 0-262-11255-8, 2001.

[Kel 1993] Kelemen M. and Bagchi A., 'Modeling and feedback control of a flexible arm of a robot for prescribed frequency-domain tolerances', *Automatica*, Vol. 29, n° 4, pp. 899-909, 1983.

[Kha 2002] Khalil, H. K., *Nonlinear Systems*. Prentice-Hall, 2002.

[Kho 1995] Khorrami F., Jain S. and Tzes A., 'Experimental results on adaptive nonlinear control and input preshaping for multi-link flexible manipulators', *Automatica*, Vol. 31, pp. 83-97, 1995.

[Koh 1972] Kohonen T., 'Correlation matrix memories', *IEEE Transactions on computers*, Vol. 21, pp. 353-359, 1972.

[Kro 1996] Krose B. and Smagt P., *An Introduction to Neural Networks*, Amsterdam University Press, Netherland, 1996.

[Kuo 2001] Kuo C.F.J. and Lee C.J., 'Neural network control of a rotating elastic manipulator', *Computers and Mathematics with applications*, Vol. 42, pp. 1009-1023, 2001.

[Kur 2005] Kurfess T.R. *Robotics and Automation Handbook*, CRC Press, USA, ISBN: 0-8493-1804-1, 2005.

[Lal 1996] Lallement Y., 'Intégration neuro-symbolique et intelligence artificielle: application et implémentation parallèle'. *Thèse de Doctorat de l'université Henri Poincarré de Nancy I*, France, 1996.

[Lan 1988] Landau I.D. and Horowitz R., 'Synthesis of adaptive controllers for robot manipulators using a passive feedback systems approach', *IEEE International Conference on Robotics and Automation*, Philadelphia, USA , pp. 1028-1033, 1988.

[Lan 1979] Landau I.D. *Adaptive control. The model reference approach*, Marcel Dekker Inc., New-York, USA, 1979.

[Len 1999] Leng J. and Asundi A. 'Active vibration control system of smart structures based on FOS and ER actuator', *Smart Material Structure*, Vol. 8, No. 2, pp. 252-256, 1999.

[Lin 1992] Lin Y.J. and Lee T.S., 'Comprehensive dynamic modeling and motion/force control of flexible manipulators', *Mechatronics*, Vol. 2, pp. 129-148, 1992.

[Lew 1995] Lewis F.L., Liu K. and Yesildirek A. 'Neural net robot controller with guaranteed tracking performance'. *IEEE Transaction on Neural Networks*. Vol. 6 No3, pp. 703-715, 1995.

[Lou 1997] Loudini M., 'Modélisation, analyse et méthodologies de commande linguistique floue d'un bras manipulateur de robot flexible', *Thèse de Magistère de l'ENP*, Alger, Algérie, 1997.

[Mao 2010a] Maouche A.R. and Attari M., 'Nonlinear adaptive RBFNN control of a one-link flexible manipulator', *IEEE 1st International Conference on Machine and Web Intelligence (ICMWI'2010)*, Algiers, Algeria, pp. 144-149, ISBN 978-1-4244-8610-6, October 2010.

[Mao 2010b] Maouche A.R., 'Intelligent control', in *Motion control*, Casolo F. Eds., Croatia: In-Teh Publication, pp. 31-50, ISBN 978-953-7619-55-8, January 2010.

[Mao 2009a] Maouche A.R. and Attari M., 'Adaptive nonlinear control of a one-link flexible manipulator', *3rd International Conference on Electrical Engineering (ICEE'2009)*, pp. 124-129, Boumerdès, Algeria, December 2009.

[Mao 2009b] Maouche A.R. and Attari M., 'CMAC based adaptive control of a flexible link manipulator', *IEEE 35th International Conference on Industrial Electronics CONtrol (IECON 2009)*, pp. 1480-1485, Porto, Portugal, ISBN: 978-1-4244-4648-3, DOI: 10.1109/IECON.2009.5414718, November 2009.

[Mao 2009c] Maouche A.R. and Attari M. 'Hybrid adaptive neural control for flexible manipulators', *International Journal of Intelligent Systems Technologies and Applications (IJISTA)*, Inderscience Ltd., Switzerland, Vol. 7, No. 4, pp. 396-413, ISSN (Online): 1740-8873, ISSN (Print): 1740-8865, DOI: 10.1504/IJISTA.2009.028055, September 2009.

[Mao 2008a] Maouche A.R. and Attari M. 'Adaptive neural controller of a rotating flexible manipulator', *IEEE 19th International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM 2008)*, pp. 517-522, Ischia, Italy, ISBN: 978-1-4244-1663-9, DOI: 10.1109/SPEEDHAM.2008.4581211, June 2008.

[Mao 2008b] Maouche A.R. and Attari M. 'Neural network-based adaptive control of a two-link flexible manipulator', *The Mediterranean Journal of Measurement and Control*, SoftMotor Ltd., United Kingdom, Vol. 4, No. 2, pp. 66-75, ISSN: 1743-9310, April 2008.

[Mao 2007] Maouche A.R. and Attari M. 'Hybrid control strategy for flexible manipulators', *IEEE 17th International Symposium on Industrial Electronics (ISIE 2007)*, pp. 50-55, Vigo, Spain, ISBN: 978-1-4244-0755-2, DOI: 10.1109/ISIE.2007.4374571, June 2007.

[Mao 1999] Maouche A.R., 'Conception et mise en œuvre de lois de commande pour un robot flexible. Optimisation des performances.', *Thèse de Magister de l'USTHB*, Alger, Algérie, 1999.

[Mar 1977] Margolis D.L. and Karnopp D.C., 'Bond-graphs for flexible multibody systems', *Proceedings IUTAM Conference on Multibody Dynamics*, pp. 208-219, Munich, Germany, 1977.

[Mik 2007] Mikles J. and Fikar M., *Process Modelling, identification and control*, Springer, 2007.

[Min 1969] Minsky M.L. and Papert S., *Perceptrons: An Essay in Computational Geometry*, Cambridge, MIT Press, 1969.

[Mit 1992] Mitrouchev P., 'Méthodologie pour l'élaboration de modèles de comportement de robot-manipulateurs sujets à des déformations élastiques, *Thèse de Doctorat de l'UFR des sciences et techniques de l'université de Franche-Comté*, France, 1992.

[Moh 2005] Mohamed Z., Martins J.M., Tokhi M.O., Sa Da Costa J. and Botto, M.A., 'Vibration control of a very flexible manipulator system', *Control Engineering Practice*, Vol. 13, pp. 267-277, 2005.

[Oak 1989a] Oakley C.M. and Cannon R.H., 'End-point control of a two-link manipulator with a very flexible forearm : Issues and experiments', *Proceedings of the American Control Conference*, Pittsburgh, USA, 1989.

[Oak 1989b] Oakley C.M. and Cannon R.H., 'Equations of motion for an experimental planar two-link flexible manipulator', *ASME Winter Annual Meeting*, pp. 267-278, San Francisco, USA, 1989.

[Pal 1996] Pal S.K. and Srimani P.K., 'Neurocomputing: motivation, models and hybridization', *Computer*, Vol. 29, No 3, pp. 24-28, 1996.

[Par 2001] Paraskevopoulos P.N., *Modern control engineering*, Marcel Dekker Inc., New York, USA, 2001.

[Par 2002] Park N.C., Yang H.S., Park H.W. and Park Y.P., 'Position/vibration Control of two-degree-of-freedom arms having one flexible link with artificial pneumatic muscle actuators', *Robotic and Autonomous Systems*, Vol. 40, pp. 239-253, 2002.

[Pha 1992] Pham C.M., 'Identification and control of flexible robots', *Thèse de Doctorat de l'université de Nantes*, France, 1992.

[Pot 1983] Potkonjac V. and Vukobratovic M., *Dynamics of manipulation robots: Theory and application, Scientific fundamentals of robotics 1*, Springer-Verlag, Berlin, Germany, 1983.

[Pri 1999] Priest W., Stevens G.T.Jr. and Liu K., *Mechanical engineering handbook. Robotics*, CRC Press LLC, 1999.

[Pri 2000] Principe J.C., 'Artificial neural networks', Dorf, R. C., Editor, *The Electrical Engineering Handbook*, Chapter 20, CRC Press LLC, 2000.

[Ren 1980] Renaud M., 'Contribution à la modélisation et à la commande dynamique des robots manipulateurs', *Thèse de Doctorat de l'université Paul Sabatier*, Toulouse, France, 1980.

[Ros 1958] Rosenblatt F., 'The perceptron: a probabilistic model for information storage and organization in the brain', *Psychological review*, Vol. 65, pp. 386-408, 1958.

[Ryu 2004] Ryu J.H., Kwon D.S. and Hannaford B., 'Control of a flexible manipulator with non collocated feedback: Time-domain passivity approach', *IEEE Transaction on Robotics*, Vol. 20, No. 4, pp. 776-780, 2004.

[Rum 1986] Rumelhart D.E., Hinton G.E. and Williams R.J., 'Learning internal representations by error propagation', Rumelhart D.E. and McClelland J.L., editors. *Parallel Data Processing: Exploration in the Microstructure of Cognition*, Vol. 1, Chapter 8, pp. 318-362, the M.I.T. Press, Cambridge, 1986.

[Sam 1990] Samanta B., 'Dynamics of flexible multibody systems using Bond-Graphs and Lagrange multipliers', *Transaction on ASME Journal of Mechanical Design*, Vol. 112, No 1, pp. 30-35, 1990.

[Ser 1994] Serna M.A. and Carrera E., 'A general solution for the inverse dynamics of flexible robots', *Proceedings of the European Robotics and Intelligent Systems Conference*, Malaga, Spain, 1994.

[Ser 1990] Serna M.A. and Bayo E., 'Off-line trajectory planning for flexible manipulators', *Proceeding of the IFIP TC7 Conference on Modeling the innovation : Communications, Automation and Information Systems*, pp 213-220, Rome, Italy, 1990.

[Sci 1988] Siciliano B. and Book W.J., 'A singular perturbation approach to control of lightweight flexible manipulators', *The International Journal of Robotics Research*, Vol. 7, pp. 79-90, 1988.

[Sch 1984] Schmitz E., 'Initial experiments on the end-point control of a flexible one link robot', *International Journal of Robotics Research*, Vol. 3, No. 3, pp. 62-75, 1984.

[Sha 1993] Shahian B. and Hassul M., *Control system design using MATLAB*, Prentice-Hall Inc, New Jersey, USA, 1993.

[Sha 2005] Shan J., Liu H.T. and Sun D., 'Slewing and vibration control of a single-link flexible manipulator by positive feedback (PPF)', *Mechatronics*, Vol. 15, pp. 487-503, 2005.

[Shi 2001] Shin H.C. and Choi S.B., 'Position control of a two-link flexible manipulator featuring piezoelectric actuators and sensors', *Mechatronics*, Vol. 11, No. 6, pp. 707-729, 2001.

[Slo 1991] Slotine J.J.E. and Lie W., *Applied nonlinear control*, Prentice-Hall Inc, New Jersey, USA, 1991.

[Slo 1987] Slotine J.J.E. and Lie W. 'On the adaptive control of robot manipulators'. *International Newspaper of Robotics Research*, Vol. 6, pp. 49-59, 1987.

[Son 1988] Soni A.H. and Naganathan G., 'Nonlinear modeling of kinematic and flexibility effects in manipulator design', *ASME Journal of Mechanics, Transmissions, and Automation in Design*, Vol. 110, No 3, pp. 243-254, 1988.

[Spe 1990] Spector V.A. and Flashner H., 'Modeling and design implications of noncollocated control in flexible systems", *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 112, p 186-193, 1990.

[Spo 1995] Spong, M.W., 'Adaptive Control of flexible joint manipulators: comments on two papers', *Automatica*, Vol. 31, pp. 585-590, 1995.

[Sun 2004] Sun D., Mills J.K., Shan J.J. and Tso S.K., 'A PZT actuator control of a single-link flexible manipulator based on linear velocity feedback and actuator placement', *Mechatronics*, Vol. 14, No. 4, pp. 381-401, 2004.

[Sun 1981] Sunada W. and Dubowsky S., 'The application of finite element method to the dynamic analysis of flexible spatial and co-planar linkage systems', *ASME Journal of Mechanical Design*, Vol. 103, No 3, pp. 643-651, 1981.

[Tan 2006] Tang Y., Sun F., Sun Z., 'Neural network control of flexible-link manipulators using sliding mode', *Neurocomputing*, Vol. 70, pp. 288-295, 2006.

[Tia 2004] Tian L. and Collins C., 'A dynamic recurrent neural network-based controller for a rigid-flexible manipulator system', *Mechatronics*, Vol. 14, pp. 471-490, 2004.

[Tza 1987] Tzafestas S.G. and Kanoh H., 'Dynamic studies of flexible robot manipulators in distributed parameter systems : Modeling and simulation', *Proceeding of IMACS/IFAC 1987*, Hiroshima, Japan; also in: Futagami T., Tzafestas S.G. and Sunahara Y., (1989) 'Distributed–parameter systems: modeling and simulation', *Elsevier Science Publishers B.V (North Holland),* p 329-344, 1987.

[Tze 1990] Tzes A.P. and Yurkovich A., 'Experiments in identification and control of flexible link manipulators', *IEEE Control System Magazine*, pp. 41-46, 1990.

[Ush 1991] Ushiyama M. and Konno A., 'Computed acceleration control for the vibration suppression of flexible robotic manipulators', *5th International Conference on Advanced Robotics*, Pisa, Italy, pp. 126-131, 1991.

[Uso 1986] Usoro P.B., Nadira R. and Mahil S.S., 'A finite element / Lagrange approach to modeling lightweight manipulators', *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 108, No 3, pp. 198-205, 1986.

[Wid 1990] Widrow B. and Lehr M.A., '30 years of adaptive neural networks: Perceptron, madaline, and backpropagation', *Proceedings of the IEEE*, Vol. 78, pp.1415-1441, 1990.

[Wid 1985] Widrow B. and Stearns S.D., *Adaptive signal processing*, Engle-wood Cliffs, Prentice-Hall, 1985.

[Wid 1960] Widrow B. and Hoff M.E., 'Adaptive switching circuits', *1960 IRE WESCON Convention Record*, New York: IRE Part 4, pp.96-104, 1960.

[Yaz 1988] Yazman A., 'Modélisation des robots flexibles par les Bond-Graphs : Application à l'analyse de leurs performances dynamiques', *Thèse de Doctorat en automatique*, Paris 11, France, 1988.

Annexe

A. Modèle dynamique

A.1 Notations et descriptions

- **Paramètres physiques du robot flexible**

Les valeurs des paramètres physiques du robot à deux bras flexibles sont données par le Tableau A.1. La configuration des bras est illustrée par la Figure A.1.

Tableau A.1

NOTATIONS ET VALEURS DES PARAMETRES PHYSIQUES DU ROBOT FLEXIBLE

Paramètres physiques	**Bras 1**	**Bras 2**
Longueur du bras (m)	$L_1 = 1.00$	$L_2 = 0.50$
Inertie concentrée à l'origine du bras (kg m^2)	$J_{A_1} = 1.80\ 10^{-3}$	$J_{A_2} = 1.85\ 10^{-4}$
Inertie concentrée à l'extrémité du bras (kg m^2)	$J_{B_1} = 4.70\ 10^{-2}$	$J_{B_2} = 0.62$
Masse du bras (kg)	$m_1 = 1.26$	$m_2 = 0.35$
Masse concentrée à l'extrémité (kg)	$m_{c_1} = 4.0$	$m_{c_2} = 1.0$
Masse volumique (kg/m^3)	$\rho_1 = 7860$	$\rho_2 = 7860$
Module de Young	$E_1 = 1.98\ 10^{11}$	$E_2 = 1.98\ 10^{11}$
Moment quadratique de section (m^4)	$I_1 = 3.41\ 10^{-11}$	$I_2 = 6.07\ 10^{-12}$

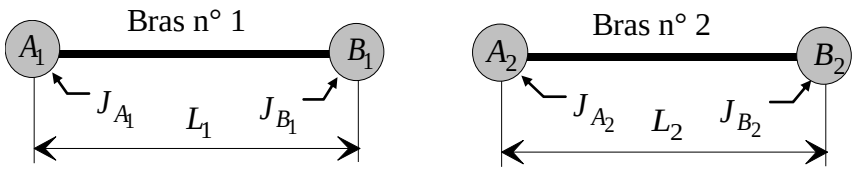

Figure A.1 Bras 1 et 2 du robot flexible à deux axes.

Les notations et vecteurs utilisés pour la description du robot à un et deux bras flexibles sont donnés ci-dessous.

- **Robot flexible à un axe**

$\mathbf{X} = [\, J_A + J_B,\ J_B,\ m_c,\ \rho I,\ \rho SL = m, EI \,]^{\mathrm{T}}$

$\mathbf{q} = [\, \theta,\ f,\ \alpha \,]^{\mathrm{T}}$

$\dot{\mathbf{q}} = [\, \dot{\theta},\ \dot{f},\ \dot{\alpha} \,]^{\mathrm{T}}$

$\dot{\mathbf{q}}\dot{\mathbf{q}} = [\, \dot{\theta}\dot{f},\ \dot{\theta}\dot{\alpha},\ \dot{f}\dot{\alpha} \,]^{\mathrm{T}}$

$\dot{\mathbf{q}}^2 = [\, \dot{\theta}^2,\ \dot{f}^2,\ \dot{\alpha}^2 \,]^{\mathrm{T}}$

$\ddot{\mathbf{q}} = [\, \ddot{\theta},\ \ddot{f},\ \ddot{\alpha} \,]^{\mathrm{T}}$

- **Robot flexible à deux axes**

$\mathbf{X} = [\, J_{A_1} + J_{B_1},\ J_{B_1},\ m_{c_1} + m_{c_2},\ \rho_1 I_1,\ m_1,\ E_1 I_1,\ J_{A_2} + J_{B_2},\ J_{B_2},\ m_{c_2},\ \rho_2 I_2,\ m_2,\ E_2 I_2 \,]^{\mathrm{T}}$

$\mathbf{q} = [\, \theta_1,\ \theta_2,\ f_1,\ \alpha_1,\ f_2,\ \alpha_2 \,]^{\mathrm{T}}$

$\dot{\mathbf{q}} = [\, \dot{\theta}_1,\ \dot{\theta}_2,\ \dot{f}_1,\ \dot{\alpha}_1,\ \dot{f}_2,\ \dot{\alpha}_2 \,]^{\mathrm{T}}$

$\dot{\mathbf{q}}\dot{\mathbf{q}} = [\, \dot{\theta}_1\dot{\theta}_2,\ \dot{\theta}_1\dot{f}_1,\ \dot{\theta}_1\dot{\alpha}_1,\ \dot{\theta}_1\dot{f}_2,\ \dot{\theta}_1\dot{\alpha}_2,\ \dot{\theta}_2\dot{f}_1,\ \dot{\theta}_2\dot{\alpha}_1,\ \dot{\theta}_2\dot{f}_2,\ \dot{\theta}_2\dot{\alpha}_2,\ \dot{f}_1\dot{\alpha}_1,$

$\dot{f}_1\dot{f}_2,\ \dot{f}_1\dot{\alpha}_2,\ \dot{\alpha}_1\dot{f}_2,\ \dot{\alpha}_1\dot{\alpha}_2,\ \dot{f}_2\dot{\alpha}_2 \,]^{\mathrm{T}}$

$\dot{\mathbf{q}}^2 = [\, \dot{\theta}_1^2,\ \dot{\theta}_2^2,\ \dot{f}_1^2,\ \dot{\alpha}_1^2,\ \dot{f}_2^2,\ \dot{\alpha}_2^2 \,]^{\mathrm{T}}$

$\ddot{\mathbf{q}} = [\, \ddot{\theta}_1,\ \ddot{\theta}_2,\ \ddot{f}_1,\ \ddot{\alpha}_1,\ \ddot{f}_2,\ \ddot{\alpha}_2 \,]^{\mathrm{T}}$

- **Autres notations**

Pour l'écriture des coefficients dynamiques du modèle nous avons adoptés les notations suivantes:

$X_i = \mathbf{X}(i)$ $\qquad$ $L_i = L_i$

$q_i = \mathbf{q}(i)$ $\qquad$ $S_{24} = \mathrm{Sin}(\theta_2 + \alpha_1)$

$\dot{q}_i = \dot{\mathbf{q}}(i)$ $\qquad$ $C_{24} = \mathrm{Cos}(\theta_2 + \alpha_1)$

A.2 Coefficients dynamiques

Nous présentons ici les matrices **A**, **B**, **C** et **K** du robot à un bras flexible ainsi que la matrice **A** du robot à deux bras flexibles. D'après les équations (I.13) et (I.14) vues au chapitre I et comme le terme de pesanteur **Q** n'existe pas, le robot évoluant dans un plan horizontal, nous pouvons écrire le modèle dynamique du robot sous la forme générale suivante :

$$\mathbf{L_r}\,\mathbf{\Gamma} = \mathbf{A(q)}\,\ddot{\mathbf{q}} + \mathbf{B(q)}\left[\dot{\mathbf{q}}\dot{\mathbf{q}}\right] + \mathbf{C(q)}\left[\dot{\mathbf{q}}^2\right] + \mathbf{K}\,\mathbf{q} \qquad \text{(A.1)}$$

- **Robot flexible à un axe**

$$\mathbf{L_r}\,\mathbf{\Gamma} = [\,\Gamma_1, 0, 0\,]^T$$

$$A(1,1) = X_1 + X_3(L^2 + {q_2}^2) + X_4\left(L + \frac{6{q_2}^2}{5L} + \frac{2L{q_3}^2}{15} - \frac{q_2 q_3}{5}\right) +$$

$$X_5\left(\frac{L^2}{3} + \frac{13{q_2}^2}{35} + \frac{L^2{q_3}^2}{105} - \frac{11L q_2 q_3}{105}\right)$$

$$A(1,2) = A(2,1) = X_3 L + X_3 + X_5\frac{7L}{20}$$

$$A(1,3) = A(3,1) = X_2 - X_5\frac{L^2}{20}$$

$$A(2,2) = X_3 + X_4\frac{6}{5L} + X_5\frac{13}{35}$$

$$A(2,3) = A(3,2) = -X_4\frac{1}{10} - X_5\frac{11L}{210}$$

$$A(3,3) = X_2 + X_4\frac{2L}{15} + X_5\frac{L^2}{105}$$

$$\mathbf{B} = \begin{bmatrix} B(1,1) & B(1,2) & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 0 & 0 & 0 \\ C(2,1) & 0 & 0 \\ C(3,1) & 0 & 0 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & K(2,2) & K(2,3) \\ 0 & K(3,2) & K(3,3) \end{bmatrix}$$

$$B(1,1) = 2X_3 q_2 + X_4\left(\frac{12q_2}{5L} - \frac{q_3}{5}\right) + X_5\left(\frac{26q_2}{35} - \frac{11Lq_3}{105}\right)$$

$$B(1,2) = X_4\left(\frac{4Lq_3}{15} - \frac{q_2}{5}\right) + X_5\left(\frac{2L^2 q_3}{105} - \frac{11Lq_2}{105}\right)$$

$$C(2,1) = -X_3 q_2 + X_4\left(\frac{q_3}{10} - \frac{6q_2}{5L}\right) + X_5\left(\frac{11Lq_3}{210} - \frac{13q_2}{35}\right)$$

$$C(3,1) = X_4\left(\frac{q_2}{10} - \frac{2Lq_3}{15}\right) + X_5\left(\frac{11Lq_2}{210} - \frac{L^2 q_3}{105}\right)$$

$$K(2,2) = X_6\frac{12}{L^3}$$

$$K(2,3) = K(3,2) = -X_6\frac{6}{L^2}$$

$$K(3,3) = X_6\frac{4}{L}$$

- **Robot flexible à deux axes**

$$\mathbf{L_r}\,\boldsymbol{\Gamma} = [\,\Gamma_1,\ \Gamma_2\,,\ 0,\ 0,\ 0,\ 0\,]^T$$

$$A(1,1) = X_1 + X_3(L_1^2 + q_3^2) + X_4 L_1 + X_5\frac{L_1^2}{3} + X_7 + X_9(2q_5(q_3 C_{24} - L_1 S_{24}) +$$

$$2L_2(q_3 S_{24} + L_1 C_{24}) + L_2^2 + q_5^2) + X_{10} L_2 + X_{11}\left(\frac{L_2^2}{3} + L_1^2 + q_3^2 +\right.$$

$$\left. L_2(q_3 S_{24} + L_1 C_{24}) + \left(\frac{q_6 L_2}{6} - q_5\right)(L_1 S_{24} - q_3 C_{24})\right)$$

$$A(1,2) = X_7 + X_9(q_5(q_3 C_{24} - L_1 S_{24}) + L_2(q_3 S_{24} + L_1 C_{24}) + L_2^2 + q_5^2) + X_{10} L_2$$

$$+ X_{11}\left(\frac{L_2^2}{3} + \frac{L_2}{2}(q_3 S_{24} + L_1 C_{24}) + \frac{1}{2}\left(\frac{q_6 L_2}{6} - q_5\right)(L_1 S_{24} - q_3 C_{24})\right)$$

$$A(1,3) = X_3 L_1 + X_4 + X_5 \frac{7L_1}{20} + X_9(L_2 C_{24} - q_5 S_{24})$$

$$+ X_{11}(L_1 + \frac{L_2 C_{24}}{2} + \frac{S_{24}}{2}(\frac{q_6 L_2}{6} - q_5))$$

$$A(1,4) = X_2 - X_5 \frac{L_1^2}{20} + X_9(q_5(q_3 C_{24} - L_1 S_{24}) + L_2(q_3 S_{24} + L_1 C_{24}) + L_2^2 + q_5^2)$$

$$+ X_7 + X_{11}(\frac{L_2^2}{3} + \frac{L_2}{2}(q_3 S_{24} + L_1 C_{24})$$

$$+ \frac{1}{2}(\frac{q_6 L_2}{6} - q_5)(L_1 S_{24} - q_3 C_{24})) + X_{10} L_2$$

$$A(1,5) = X_9(q_3 S_{24} + L_1 C_{24} + L_2) + X_{10} + X_{11}(\frac{1}{2}(q_3 S_{24} + L_1 C_{24}) + \frac{7L_2}{20})$$

$$A(1,6) = X_8 - X_{11}(\frac{L_2}{12}(q_3 S_{24} + L_1 C_{24}) - \frac{L_2^2}{20})$$

$$A(2,2) = X_7 + X_9(L_2^2 + q_5^2) + X_{10} L_2 + X_{11} \frac{L_2^2}{3}$$

$$A(2,3) = X_9(L_2 C_{24} - q_5 S_{24}) + X_{11}(\frac{L_2 C_{24}}{2} + S_{24}(\frac{q_6 L_2}{12} - q_5))$$

$$A(2,4) = X_7 + X_9(L_2^2 + q_5^2) + X_{10} L_2 + X_{11} \frac{L_2^2}{3}$$

$$A(2,5) = X_9 L_2 + X_{10} + X_{11} \frac{7L_2}{20}$$

$$A(2,6) = X_8 - X_{11} \frac{L_2^2}{20}$$

$$A(3,3) = X_3 + X_4 \frac{6}{5L_1} + X_5 \frac{13}{35} + X_{11}$$

$$A(3,4) = -X_4 \frac{1}{10} - X_5 \frac{11L_1}{210} + X_9(L_2 C_{24} - q_5 S_{24}) + X_{11}(\frac{L_2 C_{24}}{2} + S_{24}(\frac{q_6 L_2}{12} - q_5))$$

$$A(3,5) = X_9 C_{24} + X_{11} \frac{C_{24}}{2}$$

$$A(3,6) = -X_{11}\frac{L_2 C_{24}}{12}$$

$$A(4,4) = X_2 + X_4\frac{2L_1}{15} + X_5\frac{L_1^2}{105} + X_7 + X_9(L_2^2 + q_5^2) + X_{10}L_2 + X_{11}\frac{L_2^2}{3}$$

$$A(4,5) = X_9 L_2 + X_{10} + X_{11}\frac{7L_2}{20}$$

$$A(4,6) = X_8 - X_{11}\frac{L_2^2}{20}$$

$$A(5,5) = X_9 + X_{10}\frac{6}{5L_2} + X_{11}\frac{13}{35}$$

$$A(5,6) = -X_{10}\frac{1}{10} - X_{11}\frac{11L_2}{210}$$

$$A(6,6) = X_8 + X_{10}\frac{2L_2}{15} + X_{11}\frac{L_2^2}{105}$$

B. Algorithme d'apprentissage par rétro propagation

Prenons la configuration du réseau de neurones telle que décrite dans le sous-Chapitre II.5. L'algorithme d'entraînement par la méthode de rétro propagation peut être résumé par les différentes étapes suivantes:

1- Initialiser tous les poids et les biais du réseau en prenant des valeurs aléatoires, ou en suivant une méthode d'initialisation des poids spécifique.

2- Pour chaque couple d'exemple ($\mathbf{p}$, $\mathbf{y^d}$) de l'ensemble d'apprentissage :

   a) Propager les entrées vers l'avant à travers les couches du réseau pour calculer la sortie $\mathbf{a^s}$ correspondante.

   b) Calculer la fonction d'erreur par rapport aux valeurs désirées suivant l'équation (II.17).

   c) Rétro propager l'erreur vers l'arrière à travers les couches du réseau pour calculer le gradient de la fonction d'erreur, en utilisant les équations (II.23), (II.27), (II.29) et (II.33).

   d) Ajuster les poids de connexion et les biais de toutes les couches, en utilisant les équations (II.19a) et (II.19b) ou bien encore directement avec les équations (II.29) et (II.34).

   e) Si un des critères d'arrêt (erreur minimale, nombre d'itérations, etc.) est atteint alors arrêter les calculs.

   f) Sinon, ajouter une itération en reprenant à l'étape 2.

C. Définitions et théorèmes sur la stabilité des systèmes non linéaires

C.1 Définition de la notion de stabilité

Commençons par définir la stabilité globale asymptotique [Ben 1991]. Soit un système dynamique décrit par l'équation différentielle suivante:

$$\dot{\mathbf{x}} = f(\mathbf{x}, t) \tag{C.1}$$

où $\mathbf{x}$ est le vecteur d'état de dimension n du système et $f(\,)$ une fonction qui peut être non-linéaire.

Soit $\Phi(\mathbf{x_0}, t_0)$ la solution du système différentiable par rapport au temps telle que, $\forall\, \mathbf{x_0}, t_0$:

$$\Phi(\mathbf{x_0}, t_0) = \mathbf{x_0} \quad \text{et} \quad \dot{\Phi}(\mathbf{x_0}, t_0) = f(\Phi(\mathbf{x_0}, t_0)\,,\, t) \tag{C.2}$$

L'état $\mathbf{x_e}$ du système est dit globalement asymptotiquement stable si $\forall\, \mathbf{x_0} \in \mathbb{R}^n$, les conditions suivantes sont vérifiées:

1) $f(\mathbf{x_e}\,,\, t) = \mathbf{0}\,,\ \forall\, t \in \mathbb{R}^n$

2) $\forall \varepsilon > 0, \exists \delta(\varepsilon, t_0) > 0$ tel que $\left\| \mathbf{x_0} - \mathbf{x_e} \right\| \le \delta(\varepsilon, t_0) \implies \left\| \Phi(\mathbf{x_0}, t_0) - \mathbf{x_e} \right\| \le \varepsilon\,,\ \forall\, t \ge t_0$

3) $\lim\limits_{t \to \infty} \left\| \Phi(\mathbf{x_0}, t_0) - \mathbf{x_e} \right\| = \mathbf{0}$

Le système est dit asymptotiquement stable s'il n'existe qu'une zone $\delta(t_0) > 0$ pour laquelle l'état d'équilibre est stable.

Les deux premières conditions sont nécessaires pour avoir un système stable (au sens de Lyapunov) alors que la troisième le rend asymptotiquement stable. La stabilité asymptotique est donc plus restreinte que la stabilité au sens de Lyapunov (stabilité simple).

Un système possédant par exemple des pôles simples imaginaires purs étant par conséquent juste oscillant est l'exemple même d'un système qui est stable au sens de Lyapunov mais qui ne l'est pas asymptotiquement.

Nous présentons, dans ce qui suit, deux approches couramment utilisées pour démontrer la stabilité d'un système non-linéaire : la deuxième méthode de Lyapunov et l'hyperstabilté de Popov. Nous présentons plus en détail la première car nous l'utilisons pour démontrer la stabilité de nos commandes non-linéaires.

C.2 Deuxième méthode de Lyapunov

Le principe de base de cette méthode est le suivant [Slo 1991, Kha 2002]. Considérons un système physique isolé. Si la variation $\mathrm{d}E/\mathrm{d}t$ de l'énergie $E(\mathbf{x})$, $\mathbf{x}$ étant le vecteur d'état du système, est négative pour tout $\mathbf{x}$, excepté pour l'état d'équilibre $\mathbf{x_e}$, alors l'énergie continue à décroître jusqu'à atteindre son minimum $E(\mathbf{x_e})$.

Dans la théorie de la stabilité de Lyapunov, l'énergie $E(\mathbf{x})$ est remplacée par une fonction auxiliaire $V(\mathbf{x})$ scalaire ou vectorielle, qui est appelée fonction de Lyapunov.

Par analogie avec le cas de l'énergie, on exigera de cette fonction V [Kha 2002]:

- ✓ qu'elle soit positive partout sauf en $\mathbf{x_e} = \mathbf{0}$ où elle est nulle, l'origine étant supposée point d'équilibre,
- ✓ que sa dérivée $\dot{V}(\mathbf{x})$ soit négative sauf en $\mathbf{x_e}$ où elle est nulle.

Si V répond aux conditions précédentes alors on peut affirmer que l'état du système reviendra vers l'origine, si le système venait à être perturbé.

- **Formulation mathématique** [Gui 1995, Kha 2002]:

D'après la théorie de Lyapunov on peut dire que l'état d'équilibre $\mathbf{x_e}$ est globalement asymptotiquement stable si :

1) $V(\mathbf{0}, t) = 0 \quad \forall \ t \in \mathbb{R}$,

2) $V(\mathbf{x},t) \geq \alpha(\|\mathbf{x}\|), \forall t$, avec $\alpha()$ fonction scalaire continue non croissante vérifiant $\alpha(0)=0$,

3) $V(\mathbf{x},t) \to \infty$ quand $\|\mathbf{x}\| \to \infty, \forall t$,

4) $\dot{V} = \dfrac{dV}{dt} \leq -\gamma(\|\mathbf{x}\|) < 0, \forall \mathbf{x} \in \mathbb{R}^n, \forall t$; et $\gamma()$ fonction scalaire continue vérifiant $\gamma(0) = 0$.

La difficulté principale de la deuxième méthode de Lyapunov est la construction de la fonction auxiliaire *V*. Le problème réside dans la rareté de méthodes systématiques, en particulier pour les systèmes non-linéaires pour lesquels le caractère suffisant (mais non nécessaire) des critères de stabilité ne permet pas toujours de conclure.

Parmi les techniques de construction des fonctions de Lyapunov existantes on peut citer la méthode de Lur'e qui se base sur la représentation modale du système (forme canonique de Lure) et la méthode de Krasovski que nous présentons ci-après.

- **Construction de Krasovski** [Slo 1991]

C'est une technique de recherche systématique des fonctions de Lyapunov. Elle permet de déterminer les contraintes à imposer aux paramètres pour assurer la stabilité du système.

Soit un système non linéaire décrit par:

$$\dot{\mathbf{x}} = \mathbf{X}(\mathbf{x}) \tag{C.3}$$

avec, $\mathbf{x} \in \mathbb{R}^n$. On construit une forme quadratique des seconds membres X_i :

$$V = \mathbf{X}^{\mathrm{T}} \mathbf{P} \mathbf{X} \tag{C.4}$$

avec, **P** matrice symétrique nxn. D'où on déduit:

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \mathbf{X}^{\mathrm{T}}(\mathbf{J}^{\mathrm{T}} \mathbf{P} + \mathbf{P}\mathbf{J})\mathbf{X} \tag{C.5}$$

avec **J**, Jacobien associé à **X** (**x**).

On peut ensuite imposer $\mathrm{d}V/\mathrm{d}t < 0$ et en déduire les conditions sur *V* et les contraintes sur les paramètres. Notons enfin que cette technique permet d'étudier la stabilité de structure du système, cependant on n'a que des conditions suffisantes.

- **Autres méthodes de constructions** [Gui 1995]

D'une manière générale, trois grandes approches sont possibles pour trouver des fonctions de Lyapunov:

1) Chercher une fonction V de type donné

Il existe des fonctions qui 'réussissent' souvent comme fonctions de Lyapunov pour les systèmes étudiés en mécanique non linéaire. Les principales sont les suivantes :

a) Fonction quadratique des variables (Lyapunov):

$$V = \mathbf{x}^{\mathrm{T}} \mathbf{P}\, \mathbf{x} \tag{C.6}$$

où $\mathbf{P}$ est une matrice symétrique ;

b) Fonction quadratique des seconds membres du système (Krasovski):

$$V = \mathbf{X}^{\mathrm{T}} \mathbf{P}\, \mathbf{X} \tag{C.7}$$

c) Fonction quadratique plus intégrale (Lur'e):

$$V = \mathbf{x}^{\mathrm{T}} \mathbf{P}\, \mathbf{x} + \int_{\mathbf{0}}^{\mathbf{x}} f(\mathbf{u})\, \mathrm{d}\mathbf{u} \tag{C.8}$$

où f (u) est une fonction assujettie à certaines contraintes.

2) Utiliser la dérivée $\dfrac{\mathrm{d}V}{\mathrm{d}t}$

On peut quelquefois partir de la dérivée temporelle et 'remonter' à la fonction V. Ce procédé permet notamment de construire de façon systématique des fonctions de Lyapunov pour systèmes linéaires.

3) Utiliser le gradient $\dfrac{\partial V}{\partial \mathrm{x}_i}$

On peut enfin se donner à *priori* des fonctions de $\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3$

$$V_1 = \frac{\partial V}{\partial \mathrm{x}_1} \qquad V_2 = \frac{\partial V}{\partial \mathrm{x}_2} \qquad V_3 = \frac{\partial V}{\partial \mathrm{x}_3} \tag{C.9}$$

Celles-ci doivent vérifier les égalités

$$\frac{\partial V_1}{\partial x_2} = \frac{\partial V_2}{\partial x_1} \qquad \frac{\partial V_2}{\partial x_3} = \frac{\partial V_3}{\partial x_2} \qquad \frac{\partial V_3}{\partial x_1} = \frac{\partial V_1}{\partial x_3} \tag{C.10}$$

ou bien encore

$$\mathbf{rot}\,(V_1,\, V_2,\, V_3) \;=\; 0 \tag{C.11}$$

rot étant le rotationnel.

Puis il faut déduire le terme $\frac{\mathrm{d}V}{\mathrm{d}t}$ et 'remonter' à $V(x_1,x_2,x_3)$. On étudie alors le signe de V et de $\frac{\mathrm{d}V}{\mathrm{d}t}$ en vue d'appliquer le théorème de Lyapunov; c'est la méthode dite du 'gradient variable'.

C.3 Méthode de Popov

La théorie de Popov sur la stabilité est quand à elle directement liée à la notion de passivité du système [Lan 1988].

Les définitions et propriétés relatives aux systèmes passifs peuvent être retrouvés dans les ouvrages [Far 2006, Lan 1979].

Nous présentons ici quelques propriétés utilisées pour les démonstrations de stabilité.

Soit un système décrit par :

$$f(\mathbf{x},\, \dot{\mathbf{x}},\, \mathbf{u},\, t,\, t_0) \;=\; \mathbf{0}\,, \quad \mathbf{y} \;=\; h\,(\mathbf{x},\, \mathbf{u}) \tag{C.12}$$

où $\mathbf{u} \in \mathbb{R}^m$ est l'entrée, $\mathbf{x} \in \mathbb{R}^n$ est l'état du système, $\mathbf{y} \in \mathbb{R}^m$ est la sortie, f et h sont des fonctions continues par morceaux par rapport à tous leurs arguments, de plus f possède des solutions pour tout état initial $\forall t_0 \le t \le t_1$.

- **Inégalité de Popov**

Tout système passif vérifie l'inégalité suivante qui lie l'entrée **u** et la sortie **y**:

$$\int_{t_0}^{t_1} \mathbf{y}^{\mathrm{T}}(\tau)\, \mathbf{u}(\tau)\, \mathrm{d}\tau \geq -\gamma^2 \qquad \text{(C.13)}$$

- **Stabilité d'un système passif en boucle fermée**

Tout système non linéaire qui vérifie l'inégalité de Popov en contre-réaction sur un bloc linéaire de fonction de transfert **H** (p) est asymptotiquement stable si **H** (p) est strictement positive réelle (SPR).

Rappelons qu'une matrice **H** (p) de fonctions réelles est strictement positive si :

✓ tous les pôles des éléments de **H** (p) ne sont pas dans le demi-plan droit de Laplace: Re(poles) ≥ 0,

✓ la matrice **H** (jw) + $\mathbf{H}^{\mathrm{T}}(jw)$ est Hermitienne.

Une matrice **A** (p) est Hermitienne si : **A** (p) = $\mathbf{A}^{\mathrm{T}}(p^*)$ avec p^* conjugué de p.

❖ Les deux approches que nous venons de présenter et qui permettent d'appréhender le problème de la stabilité des systèmes non linéaires sont basées sur un critère d'énergie. En effet, l'approche de Popov est fondée sur le fait que l'énergie totale d'un système est égale à la somme de l'énergie initiale, de l'énergie fournie et de l'énergie dissipée ; cette dernière étant toujours négative car le système ne fournit pas de lui-même de l'énergie. L'approche de Lyapunov, quand à elle, est basée sur le choix d'une fonction qui est bien souvent le reflet de l'énergie.

D. Propriétés du modèle dynamique

L'objectif de cette partie est de souligner quelques propriétés de notre modèle dynamique donné par (I.13) que nous utiliserons pour élaborer la loi de commande et démontrer sa stabilité.

- **Définie positivité et symétrie de la matrice d'inertie A**

Lors de la simulation numérique de notre robot, nous avons dû intégrer le système implicite d'équations différentielles non linéaire que constitue le modèle dynamique inverse et qui a nécessité l'inversion de la matrice **A**, ceci a été rendu possible du fait que la définie positivité de **A** garantit l'existence de la matrice inverse $\mathbf{A}^{-1}$.

La stabilité des algorithmes de commande nécessite souvent la propriété de définie positivité de la matrice d'inertie **A** notamment dans le cadre d'une fonction de Lyapunov.

- **Définie positivité et symétrie de la sous-matrice raideur $\mathbf{K_e}$**

Cette propriété sera exploitée dans la démonstration de stabilité de la loi de commande non-linéaire. Elle exprime le fait qu'un robot flexible possède des caractéristiques naturelles de stabilité que l'on peut utiliser afin d'obtenir des lois de commande simples et efficaces.

- **Passivité du système global**

La matrice **h** dans (I.13) est de dimension nxn, elle peut être calculée à partir de **A** comme suit [Kur 2005]:

$$\mathbf{h}\,(i,j) \;=\; \frac{1}{2}\sum_{k=1}^{n}\dot{q}_k\left[\frac{\partial\,\mathbf{A}(i,j)}{\partial\,q_k}+\frac{\partial\,\mathbf{A}(i,k)}{\partial\,q_j}-\frac{\partial\,\mathbf{A}(j,k)}{\partial\,q_i}\right] \tag{D.1}$$

les éléments $\mathbf{h}\,(i, j)$ sont appelés symboles de Kristoffel.

Nous avons alors la propriété fondamentale suivante : $\dot{\mathbf{A}}-2\mathbf{h}$ est une matrice antisymétrique [Damaren 1995, Che 2000, Dom 2007].

Cette propriété a été exploitée pour l'élaboration commande stable de robots manipulateurs [Lan 1988, Hor 1990, Slo 1991, Gui 1995, Kur 2005, Far 2006]. Elle exprime la passivité énergétique du robot en tant que structure mécanique. Nous l'utiliserons dans le cadre de la démonstration de stabilité de la loi de commande non linéaire.

- **Linéarité par rapport aux paramètres dynamiques**

On a vu au Chapitre I, que le modèle dynamique du robot flexible s'exprimait linéairement en fonction d'un jeu de paramètre $\mathbf{X}$ appelés paramètres dynamiques. Ces paramètres combinent les paramètres inertiels classiques avec les paramètres de raideur élastiques [Gau 1990]. Cette propriété est utilisée lors de l'élaboration de la loi de commande non-linéaire adaptative.

Rappelons l'équation (I.29) du Chapitre I [Gau 1991]:

$$\mathbf{L_r}\,\boldsymbol{\Gamma} = \mathbf{D}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})\,\mathbf{X} \tag{D.2}$$

où, $\mathbf{D}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ est la matrice d'information du système.

Nous présentons, dans ce qui suit, le détail de la matrice $\mathbf{D}$ dans le cas du robot à un bras flexible; l'écriture pour le cas du robot à deux bras flexibles étant plus lourde mais suivant le même raisonnement pour l'obtention de $\mathbf{D}$.

Dans l'équation du modèle dynamique (I.13) on obtient, en tirant en facteur les paramètres dynamiques du robot flexible, l'équation suivante:

$$\begin{bmatrix}\Gamma\\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{D_r}\\ \mathbf{D_e}\end{bmatrix} \cdot \mathbf{X} \tag{D.2}$$

ou bien encore:

$$\begin{bmatrix}\Gamma\\ 0\\ 0\end{bmatrix} = \begin{bmatrix} D(1,1) & D(1,2) & D(1,3) & D(1,4) & D(1,5) & D(1,6)\\ D(2,1) & D(2,2) & D(2,3) & D(2,4) & D(2,5) & D(2,6)\\ D(3,1) & D(3,2) & D(3,3) & D(3,4) & D(3,5) & D(3,6)\end{bmatrix} \cdot \begin{bmatrix}X_1\\ X_2\\ X_3\\ X_4\\ X_5\\ X_6\end{bmatrix} \tag{D.3}$$

avec,

$D(1, 1) = \ddot{q}_1$,

$D(1, 2) = \ddot{q}_3$,

$D(1, 3) = (L^2 + q_2^2)\ddot{q}_1 + L\ddot{q}_2 + 2q_2\dot{q}_1\dot{q}_2$,

$$D(1, 4) = (L + \frac{6q_2^2}{5L} + \frac{2Lq_3^2}{15} - \frac{q_2 q_3}{5})\ddot{q}_1 + \ddot{q}_2 + (\frac{12q_2}{5L} - \frac{q_3}{5})\dot{q}_1\dot{q}_2 + (\frac{4Lq_3}{15} - \frac{q_2}{5})\dot{q}_1\dot{q}_3,$$

$$D(1, 5) = (\frac{L^2}{3} + \frac{13q_2^2}{35} + \frac{L^2 q_3^2}{105} - \frac{11Lq_2 q_3}{105})\ddot{q}_1 + \frac{7L}{20}\ddot{q}_2 - \frac{L^2}{20}\ddot{q}_3 + (\frac{26q_2}{35} - \frac{11Lq_3}{105})\dot{q}_1\dot{q}_2 + (\frac{2L^2 q_3}{105} - \frac{11Lq_2}{105})\dot{q}_1\dot{q}_3,$$

$D(1, 6) = D(2, 1) = D(2, 2) = 0$,

$D(2, 3) = L\ddot{q}_1 + \ddot{q}_2 - q_2\dot{q}_1^2$,

$D(2, 4) = \ddot{q}_1 + \frac{6}{5L}\ddot{q}_2 - \frac{1}{10}\ddot{q}_3 + (\frac{q_3}{10} - \frac{6q_2}{5L})\dot{q}_1^2$,

$D(2, 5) = \frac{7L}{20}\ddot{q}_1 + \frac{13}{35}\ddot{q}_2 - \frac{11L}{210}\ddot{q}_3 + (\frac{11Lq_3}{210} - \frac{13q_2}{35})\dot{q}_1^2$,

$D(2, 6) = \frac{12}{L^3}q_2 - \frac{6}{L^2}q_3$,

$D(3, 1) = D(3, 3) = 0$,

$D(3, 2) = \ddot{q}_1 + \ddot{q}_3$,

$D(3, 4) = -\frac{1}{10}\ddot{q}_2 + \frac{2L}{15}\ddot{q}_3 + (\frac{q_2}{10} - \frac{2Lq_3}{15})\dot{q}_1^2$,

$D(3, 5) = -\frac{L^2}{20}\ddot{q}_1 - \frac{11L}{210}\ddot{q}_2 + \frac{L^2}{105}\ddot{q}_3 + (\frac{11Lq_2}{210} - \frac{L^2 q_3}{105})\dot{q}_1^2$,

$D(3, 6) = -\frac{6}{L^2}q_2 + \frac{4}{L}q_3$.

E. Algorithme de la commande non linéaire adaptative

Nous présentons ci après l'algorithme que nous avons élaboré pour la commande non-linéaire adaptative appliquée au robot flexible.

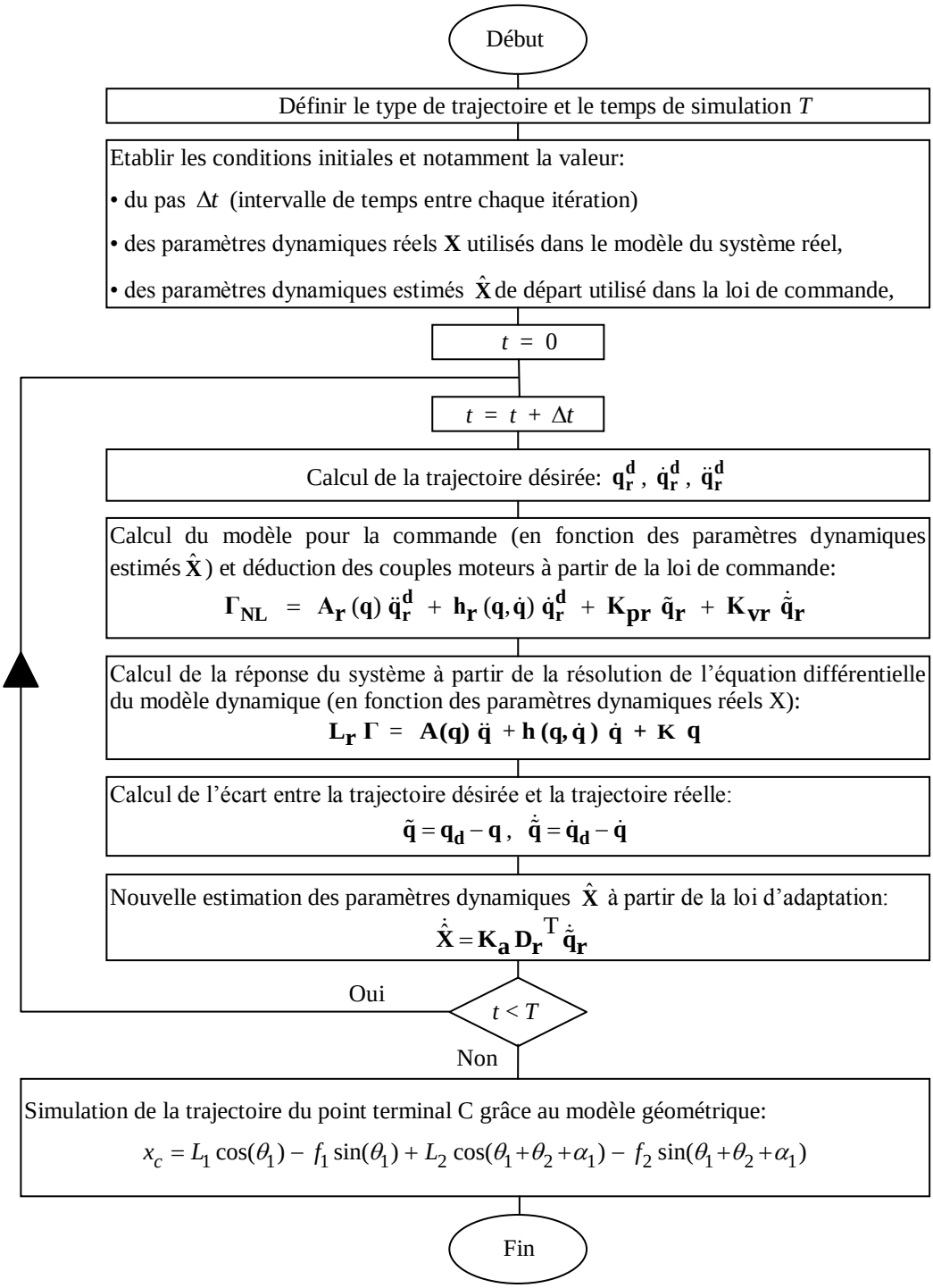

Figure E.1 Algorithme de la commande non-linéaire adaptative.

F. Réglage des RNA utilisés dans la commande hybride

Nous avons élaboré au Chapitre IV une commande hybride originale. Cette commande fonctionne par l'association de deux contrôleurs. Les caractéristiques de ces deux contrôleurs sont données ci-après.

- **Contrôleur 1**

Le premier contrôleur utilise deux réseaux de neurones artificielles $\mathbf{A_r NN}$ et $\mathbf{h_r NN}$ pour approximer les fonctions non linéaires des matrices $\mathbf{A_r(q_r, q_e)}$ et $\mathbf{h_r(q_r, q_e, \dot{q}_r, \dot{q}_e)}$.

Nous avons utilisé dans l'architecture de $\mathbf{A_r NN}$ et $\mathbf{h_r NN}$, un réseau de neurones de type MLP à trois couches qui consistent en une couche d'entrée, une couche cachée et une couche de sortie, pour chacun d'eux. On a considéré, une fonction d'activation tangente hyperbolique dans la couche cachée et une fonction d'activation linéaire dans la couche de sortie, cette configuration garantissant la propriété d'approximation universelle du réseau (voir Chapitre II).

Les signaux d'entrée de la couche d'entrée de $\mathbf{A_r NN}$ correspondent aux variables utilisées dans la matrice $\mathbf{A_r(q_r, q_e)}$. Ce sont donc, les positions angulaires rigides et élastiques des deux bras: $[\theta_1, \theta_2, f_1, \alpha_1, f_2, \alpha_2]^{\mathrm{T}}$. Les sorties correspondent aux fonctions réalisées par les éléments de la matrice $\mathbf{A_r(q_r, q_e)}$ de dimension (2x2). Nous avons donc en sortie 4 signaux correspondant à: $A_r(1,1)$, $A_r(1,2)$, $A_r(2,1)$ et $A_r(2,2)$. Le réseau de neurone utilisé ici, contient 8 neurones dans la couche caché.

Les signaux d'entrée de $\mathbf{h_r NN}$ correspondent aux variables utilisées dans la matrice $\mathbf{h_r(q_r, q_e, \dot{q}_r, \dot{q}_e)}$. Ce sont donc, les positions et vitesses angulaires, rigides et flexibles des deux bras: $[\theta_1, \theta_2, f_1, \alpha_1, f_2, \alpha_2, \dot{\theta}_1, \dot{\theta}_2, \dot{f}_1, \dot{\alpha}_1, \dot{f}_2, \dot{\alpha}_2]^{\mathrm{T}}$. Les sorties du réseau correspondent aux fonctions réalisées par les éléments de la matrice $\mathbf{h_r(q_r, q_e, \dot{q}_r, \dot{q}_e)}$ de dimension (2x2). Les quatre signaux de sortie correspondent à: $h_r(1,1)$, $h_r(1,2)$, $h_r(2,1)$ et $h_r(2,2)$. La couche cachée contient ici, 12 neurones.

La méthode de rétro propagation du gradient a été utilisée lors de l'entrainement de ces deux réseaux avec un pas de gradient $\mu = 0.6$. Le mode d'entrainement par lot (voir Chapitre II) a été adopté du fait que l'apprentissage s'effectue ici hors ligne, c'est-à-dire, avant d'appliquer la commande.

Comme exemples pour l'entrainement, nous avons choisi un ensemble de fonctions sinusoïdales. Pour le critère d'arrêt de l'algorithme d'apprentissage nous avons considéré la méthode de validation croisée (voir le sous-Chapitre IV.4.1).

Nous donnons à titre indicatif, dans la Figure F.1, le résultat de la simulation du réseau $\mathbf{A_r NN}$ (l'approximation de $\mathbf{h_r NN}$ étant du même ordre). Nous avons considéré la trajectoire de type 'Bang bang' utilisé dans le test de la commande hybride (voir le sous-Chapitre IV.4.3). Les résultats correspondent aux fonctions non linéaires réalisés par les éléments $A_r(1,1)$, $A_r(1,2)$, $A_r(2,1)$ et $A_r(2,2)$, respectivement. La courbe désirée est donnée en trait bleu et la courbe simulée par le réseau de neurones en trait rouge.

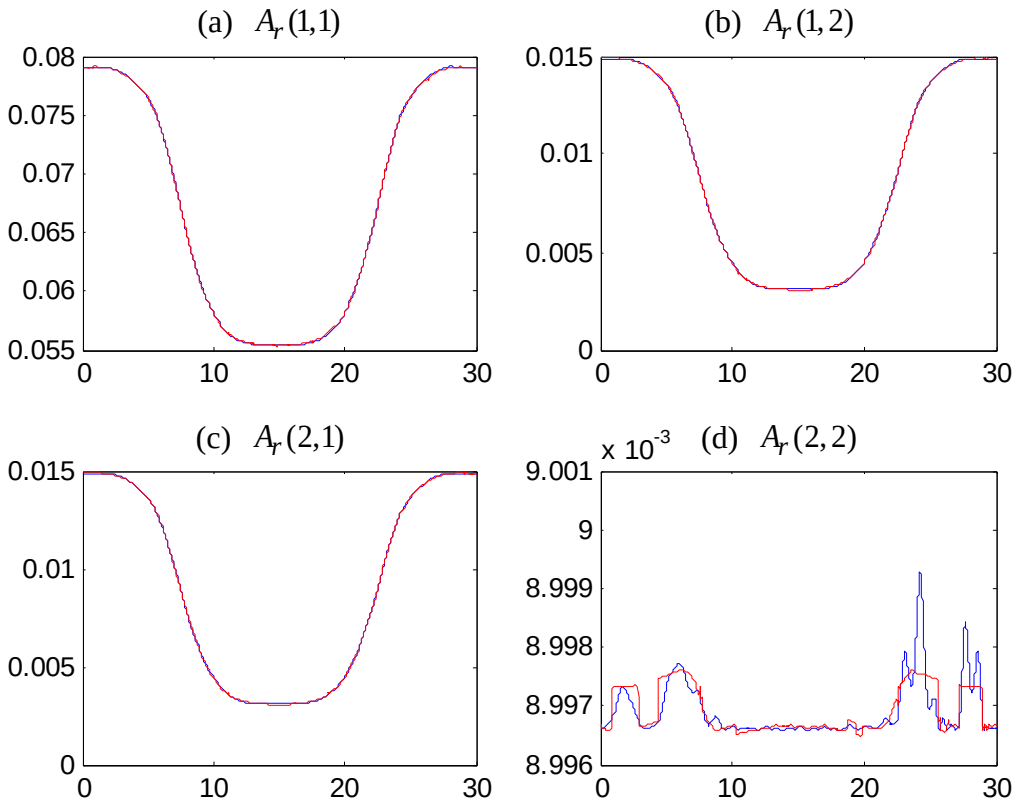

Figure F.1 Résultats de simulation du réseau $\mathbf{A_r NN}$.

On remarque d'après les résultats obtenus, une très bonne approximation des éléments $A_r(1,1)$, $A_r(1,2)$ et $A_r(2,1)$. Notons que le réseau a bien reproduit la similitude entre les éléments $A_r(1,2)$ et $A_r(2,1)$, la matrice $\mathbf{A_r(q_r, q_e)}$ étant symétrique.

L'approximation de l'élément $A_r(2,2)$ est moins précise mais suit globalement le contour de la courbe désirée. Ceci s'explique par le fait que la fonction correspondant à l'élément $A_r(2,2)$ contient beaucoup plus de non linéarités et a une forme très irrégulière par rapport aux fonctions relatives à $A_r(1,1)$, $A_r(1,2)$ et $A_r(2,1)$.

Nous pouvons toutefois améliorer la précision de l'approximation en augmentant le nombre de neurones du réseau, afin de générer plus de non linéarités. Nous avons aussi obtenus de très bons résultats dans l'approximation des quatre éléments de la matrice en utilisant un réseau de neurones pour chaque élément de la matrice.

Cependant ces solutions tendent à accroitre la complexité de la commande (et donc diminue sa réactivité); de plus nous n'avons remarqué aucune amélioration notable dans le contrôle. Nous avons donc gardé la configuration originale du réseau pour simplifier la commande et réduire son temps de calcul.

- **Contrôleur 2**

Le deuxième contrôleur est constitué d'un réseau de neurone adaptatif (fonctionnant en ligne) de type MLP. Trois couches sont utilisées, dont une cachée à 6 neurones. La fonction d'activation est de forme tangente hyperbolique dans la couche cachée et linéaire en sortie.

Les signaux d'entrée de la première couche sont au nombre de quatre et représentent les positions et vitesses angulaires: $[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T$ et les deux signaux de sorties de la dernière couche sont les couples articulaires: $\mathbf{\Gamma_{AN}} = [\Gamma_{AN_1}, \Gamma_{AN_2}]^T$.

On a utilisé pour l'entrainement, la méthode de rétro propagation du gradient avec un terme d'inertie $\eta = 0.5$ (momentum) et un pas d'apprentissage $\mu = 0.7$. Nous avons adopté, le mode d'entrainement incrémentale car l'apprentissage se fait, ici, en ligne c'est-à-dire pendant la commande et donc au fur et à mesure que les données arrivent au réseau.

Nous pouvons dire, d'après les résultats de simulation dans le sous-Chapitre IV.4.3, que le réseau adaptatif a bien rempli son rôle. En effet sa rapidité et son efficacité ont permis de réduire l'erreur de suivi pendant le contrôle, malgré la dynamique importante de la trajectoire de consigne et l'erreur significative introduite lors de l'estimation des paramètres du robot flexible.

ملخص

البحث المقدم في هذه الأطروحة يصف نهج هجين لمشكلة التحكم على الروبوت المرن الأذرع في ظل الأخطاء المنتظمة وغير المنتظمة في تمثيل النظام. في هذا العمل، تم أولا، بناء وحدة تحكم عصبونية على أساس معادلة ديناميكية الحركة للروبوت المرن الأذرع. الهدف منها إنتاج تحكم سريع ومستقر لموضع و سرعة المفاصل و تخليص الأذرع من الذبذبات. ثم أ ضيفت وحدة تحكم عصبونية متكيّفة بهدف تعويض اللا خطيات المجهولة والديناميكيات الغير ممثلة، مما أدى إلى تحسين دقة التحكم. وقد تم اختبار متانة وحدة التحكم الهجينة في ظل اضطرابات معتبرة و قورنت قدراتها بقدرات وحدة تحكم لا خطية كلاسيكية. أظهرت نتائج المحاكاة فعالية إستراتيجية التحكم الهجين المقترحة.

كلمات البحث : التحكم العصبوني المتكيّف، الروبوت المرن الأذرع، الديناميكيات اللا خطية.

RÉSUMÉ

Le travail de recherche présenté dans cette thèse décrit une approche hybride au problème de la commande des robots manipulateurs à bras flexibles, face à des incertitudes structurées et non structurées dans le modèle. Dans ce travail, un premier contrôleur neuronal, basé sur l'équation dynamique du robot flexible à été élaboré. Son but est de produire un contrôle rapide et stable des positions et vitesses articulaires du robot et d'amortir les oscillations des bras. Un deuxième contrôleur, neuronal adaptatif, a été ensuite ajouté afin de compenser les non-linéarités non identifiées et les dynamiques non modélisées, améliorant ainsi la précision du contrôle. La robustesse de la commande hybride, ainsi constituée, a été testée face à des perturbations importantes et ses performances ont été comparées à celles d'une commande non-linéaire classique. Les résultats obtenus en simulation ont montré l'efficacité de la stratégie de commande hybride proposée.

Mots clés: commande neuronale adaptative, robot manipulateur à bras flexibles, dynamiques non-linéaires.

ABSTRACT

The research tasks presented in this thesis describe a hybrid approach to the problem of controlling flexible link manipulators for both structured and unstructured uncertainties conditions. First, a neural network controller, based on the robot's dynamic equation of motion, is elaborated. It aims to produce a fast and stable control of the joint position and velocity, and to dump the vibrations of the arms. Then, an adaptive neural controller is added to compensate the unknown nonlinearities and unmodeled dynamics, thus enhancing the accuracy of the control. The robustness of the hybrid controller has been tested under important disturbances and compared to a classical nonlinear controller. Simulation results showed the effectiveness of the proposed hybrid control strategy.

Keywords: adaptive neural network control, flexible link manipulator, nonlinear dynamics.