وزارة التعليم العالي والبحث العلمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جامعة فرحات عباس - سطيف
UNIVERSITE FERHAT ABBAS - SETIF
UFAS (ALGERIE)

THESE

Présentée à la Faculté des Sciences de l'Ingénieur
Département d'Electronique
pour l'obtention du Diplôme de

DOCTORAT D'ETAT

en Electronique

par

Mr KARA Kamel

THEME

CONTROLE INTELLIGENT ET SYSTEMES AUTONOMES
(APPLICATION A LA ROBOTIQUE)

Soutenue : le 15 / 03 / 2006 devant la commission d'examen composée de :

ZEHAR Khaled	Prof. à l'Université de Sétif	Président
BENMAHAMMED Khier	Prof. à l'Université de Sétif	Rapporteur
GUESSOUM Abderrezak	Prof. à l'Université de Blida	Examinateur
BELARBI Khaled	Prof. à l'Université de Constantine	Examinateur
MOSTEFAI Mohammed	Prof. à l'Université de Sétif	Examinateur

Résumé

La commande prédictive est devenue actuellement un outil précieux de commande dans divers domaines. Elle est largement étudiée et bien connue dans le cas des systèmes linéaires. L'extension de cette technique à la commande des systèmes non linéaires a récemment fait l'objet de plusieurs travaux de recherches et plusieurs algorithmes, en particulier ceux utilisant la logique floue et les réseaux de neurones, ont été proposés. L'objectif principal de ce travail est de considérer l'application des réseaux de neurones à cette stratégie de commande prédictive.

Le travail présenté envisage les obstacles les plus significatifs rencontrés lors du développement de ces structures de commande prédictive non linéaire : mise en oeuvre d'un modèle non linéaire et solution rapide et fiable de l'algorithme de commande pour une implantation temps réel. Une partie de ce travail est donc consacrée à l'identification des systèmes non linéaires en utilisant les réseaux de neurones. L'autre partie traite le problème de la mise en œuvre de la loi de commande prédictive lorsqu'on utilise un modèle neuronal pour la prédiction du comportement futur du système. Afin de mettre en évidence l'apport des méthodes proposées, des applications, sous forme de simulation, ont été considérés.

Mots clés : *réseaux de neurones, commande prédictive, optimisation, systèmes non linéaires, identification.*

Abstract

The predictive control has become currently a precious tool for control in various domains. It is well known and largely studied in the case of linear systems. The extension of this technique for the control of non linear systems has recently been the subject of many researches, where several algorithms were proposed. The objective of this work is to consider the application of neural networks to this technique of predictive control.

This work deals with the most significant difficulties met during the development of these structures of nonlinear predictive control; that is obtaining a nonlinear model and a fast implementation of the control algorithm for real time applications. Thus, a part of this work is devoted to the identification of nonlinear systems using neural networks. The other part, deals with the problem of the implementation of the predictive control law when using a neural model for the prediction of a future behaviour of the system. In order to highlight the contribution of the proposed methods, simulation examples were considered.

Key words: *neural networks, predictive control, optimization, nonlinear systems, identification.*

REMERCIEMENTS

Je remercie vivement Monsieur Khier Benmahammed , professeur à l'Université de Setif et directeur de ma thèse, Monsieur K. Benmahammed, qui, par sa confiance, ses conseils amicaux, ses explications, a contribué à la réalisation de ce travail. Qu'il trouve ici l'expression de ma profonde gratitude.

Mes remerciements vont aussi à Monsieur Vincent Wertz, professeur à l'Université Catholique de Louvain en Belgique, pour l'honneur qu'il m'a fait en m'accueillant au sein du centre de recherche 'CESAME ' et pour ses conseils, ses encouragements, ses explications éclairées et surtout sa disponibilité.

Je tiens à remercier très chaleureusement Monsieur K. Zehar, professeur à l'Université de Setif, de l'honneur qu'il m'a fait en présidant le jury de cette thèse.

C'est un honneur pour moi que Monsieur A. Guessoum, professeur à l'Université de Blida et Directeur au ministère de l'enseignement supérieur et de la recherche scientifique, ait accepté de participer à ce jury et de consacrer du temps pour se pencher sur mon mémoire malgré ses nombreuses occupations.

Je remercie Monsieur K. Belarbi, professeur à l'Université de Constantine, qui me fait l'honneur et le grand plaisir en acceptant de participer à ce jury.

Je suis également très sensible à l'intérêt que Monsieur M. Mostefai, professeur à l'Université de Sétif, a accordé à mon travail en acceptant d'examiner cette thèse.

Je suis fort reconnaissant au Dr. Nabil Benoudjit qui m'a grandement aidé dans la réalisation de ce travail. Qu'il trouve ici l'expression de ma profonde gratitude.

Mes très vifs remerciements s'adressent au Dr. M.L. Hadjili qui m'a énormément aidé dans la réalisation de ce travail.

Je tiens aussi à adresser mes remerciements à mon beau frère K E. Hemsas, pour les services qui m'a rendu en s'occupant de toute la procédure administrative de ma soutenance.

Enfin, ma profonde gratitude va à ceux qui m'ont aidé de prés ou de loin, que ce soit par leur amitié ou leurs conseils et soutien.

Sommaire

INTRODUCTION

Contexte du travail

La commande des systèmes complexes s'heurte à plusieurs difficultés à savoir : (i) la présence des non linéarités, (ii) les incertitudes et (iii) la complexité des calculs [47]. La solution de ce problème exige des contrôleurs intelligents ayant de grandes capacités d'apprentissage, d'adaptation et de prise de décisions. Les systèmes biologiques et, en particulier l'être humain, en présente un exemple. Ainsi, une imitation, même partielle, de leurs capacités fournit une contribution importante à la théorie des systèmes. Cette imitation peut être faite par l'analyse et la compréhension des aspects structurels et fonctionnels de ces systèmes naturels. Ensuite, la modélisation des composantes d'intelligence ainsi identifiées en utilisant des représentations graphiques et les mathématiques. Les travaux déjà réalisés ont conduit à l'avènement de puissants outils tels que les réseaux de neurones, la logique floue et les algorithmes génétiques. Notre travail, qui s'inscrit dans le domaine de l'intelligence artificielle, a pour objectif l'application des réseaux de neurones à la modélisation et au contrôle des systèmes non linéaires et plus particulièrement le contrôle prédictif à modèle neuronal.

La commande par réseaux de neurones a fait l'objet de nombreux travaux de recherche depuis une dizaine d'années, à cause de la capacité d'apprentissage, d'approximation et de généralisation que possèdent ces réseaux. En effet, différentes méthodes de commande des systèmes non linéaires à base de réseaux de neurones ont été proposées [2], [8], [9], [46]. Elles peuvent être classées en deux grandes catégories : la commande direct, où le régulateur est un réseau de neurones, et la commande par modèle, appelé aussi commande indirect, où un modèle neuronal est utilisé pour modéliser le comportement dynamique du système.

Les avantages que l'on espère tirer des réseaux de neurones lorsqu'on les applique à la commande des systèmes sont en général :

- le traitement parallèle et distribué des informations servant à la commande ;
- l'adaptation et l'apprentissage ;
- l'absence presque totale de restrictions sur les non linéarités du système ;
- la possibilité d'établir une loi de commande avec un minimum d'information a priori sur le système ;
- la rapidité du traitement grâce à une mise en œuvre parallèle possible ;
- la robustesse par rapport au bruit et par rapport aux défaillances internes.

En pratique, on n'a pas encore atteint ces espérances. On manque d'outils théoriques permettant de garantir certaines performances, telles que les garanties de convergence et de stabilité [29], [46]. Il est évident que de telles garanties ne peuvent s'obtenir qu'au prix d'hypothèses assez restrictives sur la tâche à accomplir. D'un côté, en restreignant fort le processus (en le supposant, par exemple, linéaire, à minimum de phase, d'ordre et de temps mort connus), on obtient effectivement des garanties de stabilité pour la commande d'un tel processus ; d'un autre côté, en voulant traiter un processus dynamique tout à fait général (non linéaire et inconnu), on se trouve obligé d'adopter des systèmes de

commande à structure complexe, dont les propriétés de stabilité et d'adaptation ne peuvent être garanties. Il y a donc un juste milieu à trouver en fonction de l'application envisagée.

La commande prédictive est une méthode relativement récente, qui se base, dans la conception de la loi de commande, sur la notion de prédiction. Cette notion de prédiction est présente dans de nombreuses activités humaines telles que la marche, la conduite d'une automobile ou la pratique sportive, activités pour lesquelles on cherche à anticiper une trajectoire prédéfinie située dans le futur. La commande prédictive qui n'a émergé d'une manière industrielle qu'à partir du milieu des années 80, a commencé à donner ses premiers résultats théoriques à la fin des années 70. Il est vrai que le réel essor de cette technique de commande s'est manifesté grâce aux travaux de D. W. Clarke [16] qui a popularisé son concept. A partir de ces travaux, de nombreux résultats d'applications ont été publiés et ce dans des domaines très variés tels que les secteurs de l'industrie chimique, pétrolière, de la robotique et plus généralement de la machine outils. Ces applications industrielles ont toutes un dénominateur commun : la connaissance de la trajectoire à suivre par le système dans le futur sur un certain horizon. La commande prédictive peut être appliquée aux systèmes multivariables et permet de contrôler une large classe de systèmes (instables, à phase non minimale, avec retard). C'est pourquoi elle est apparue très séduisante aux industriels dans le cadre général de problème de poursuite de trajectoires. La commande prédictive est dans la plupart des cas, une commande discrète, même s'il existe malgré tout des commandes prédictives continues.

Motivations et objectifs

Compte tenu du comportement non linéaire de la majorité des processus industriels, l'extension de cette technique de commande prédictive aux cas des systèmes non linéaires s'avère une nécessité impérative. Cette extension est surtout motivée par le fait qu'il n'est pas possible d'obtenir des performances satisfaisantes en utilisant la commande prédictive à modèle linéaire pour la commande des systèmes à point de fonctionnement multiple. En fait, ces dernières années ont permis des développements très rapides, tant au niveau de l'élaboration que de l'application, d'algorithmes de commande prédictive non linéaire, en particulier ceux utilisant les réseaux de neurones et la logique floue. Ces algorithmes diffèrent entre eux par la technique utilisée pour la génération d'une loi de commande implantable de façon réaliste, cependant tous sont basés globalement sur la superposition d'un modèle de prédiction non linéaire à un critère de performances également non linéaire. En envisageant le problème de la commande prédictive à modèle non linéaire, deux difficultés principales peuvent être rencontrés lors de la mise en œuvre de la loi de commande :

- mise en œuvre d'un modèle non linéaire qui peut représenter adéquatement la dynamique du processus ;
- solution rapide et efficace au problème d'optimisation pour calculer en temps réel le signal de commande.

En raison des propriétés d'approximation universelle, de généralisation et d'apprentissage que possèdent les réseaux de neurones, leur utilisation pour la prédiction du comportement futur du système en vue de concevoir une loi de commande prédictive semble surmonter la première difficulté et peut mener à un niveau de performances séduisant. En fait, le premier objectif qui a été fixés pour notre travail est l'emploi des réseaux de neurones, en particulier ceux de type multicouches, pour l'identification des systèmes non linéaires. Les recherches envisagées dans ce contexte ont abouti à la proposition d'une méthode pour l'identification des systèmes non linéaires en présence des perturbations externes. Une formulation mathématique de la solution proposée est donnée, et les deux cas de représentation des perturbations, par un modèle linéaire et par un modèle non linéaire, sont considérés.

La commande prédictive à modèle neuronal, que l'on peut rattacher à la famille de la commande par modèle, ne cesse de connaître ces dernières années un développement croissant. En effet, plusieurs techniques, utilisant l'approche neuronale et permettant de mettre en œuvre ce type de commande ont été proposées. Néanmoins, vu la structure complexe des réseaux de neurones, l'utilisation des modèles neuronaux conduit à un problème d'optimisation non linéaire et non convexe pour lequel une solution analytique n'est pas possible. Les méthodes itératives de l'optimisation non linéaire sont généralement très coûteuses en terme de temps de calcul, ceci est très contraignant dans le cas d'une application en temps réel portant sur des systèmes à dynamique rapide. Le deuxième objectif tracé pour notre travail, est de trouver une solution à la deuxième difficulté que nous avons soulevé en considérant la commande prédictive non linéaire. En fait, dans le but de proposer une solution efficace à ce problème, nous avons en premier lieu reconsidéré l'approche utilisant la méthode de Newton-Raphson. Une amélioration de cette approche, portant sur la stabilité numérique de l'algorithme et qui consiste à utiliser l'algorithme de Jordan, est proposée. Ensuite, une méthode permettant le calcul analytique de la loi de commande est introduite. Cette méthode qui s'appuie sur le principe de la décomposition de la réponse totale du système en ses composantes libre et forcée, offre la possibilité d'être utilisée pour la commande en temps réel des systèmes non linéaires à dynamique rapide. Enfin, une étude comparative permettant de mettre en évidence les performances et l'apport de l'approche proposée est envisagé.

Organisation de la thèse

La thèse est organisée en quatre chapitres. Les deux premiers introduisent les principes fondamentaux des réseaux de neurones et la commande prédictive généralisée (GPC) d'un point de vue général et conceptuel. Le troisième chapitre est principalement consacré à la méthode proposée pour l'identification des systèmes non linéaires en présence des perturbations. Enfin, le dernier chapitre est consacré à la commande prédictive à modèle neuronal et présente les approches développées. Voici un résumé sur les aspects examinés dans chaque chapitre.

Chapitre 1

Le chapitre 1, introductif, présente brièvement les éléments de base des réseaux de neurones nécessaires à la réalisation de ce travail, à savoir le modèle du neurone formel, les architectures et les algorithmes d'apprentissage des réseaux de neurones.

Chapitre 2

Le chapitre 2 présente la stratégie de la commande prédictive et les étapes essentielles de la mise en œuvre d'une loi de commande prédictive. Puis rappelle les bases de la commande prédictive généralisée et les approches, algorithmique et polynomiale, d'élaboration de la loi de commande. Enfin, nous considérons l'application de la GPC à la commande d'un système non linéaire. Le but de ce chapitre est de préciser les bases théoriques qui ont servi à la réalisation de ce travail.

Chapitre 3

Ce chapitre introduit la méthode proposée pour l'identification neuronale des systèmes non linéaires en présence des perturbations. Le chapitre commence par l'exposé des étapes principales de

l'identification en utilisant les réseaux de neurones. Ensuite présente les développements mathématiques de la méthode proposée en considérant les deux cas possibles pour la modélisation des perturbations. Enfin, le chapitre se termine sur des exemples en simulation confirmant la solution proposée.

Chapitre 4

Dans ce chapitre, après avoir formulé le problème de la commande prédictive en utilisant un modèle neuronal, nous introduisons les principes fondamentaux de quelques approches déjà proposées qui serviront à la mise en évidence des résultats obtenus. Ce chapitre présente d'une manière détaillée les étapes de conception d'une loi de commande prédictive en utilisant la méthode de Newton-Raphson, les performances de cette approche sont mises en évidence en considérant plusieurs applications. Nous présentons aussi les développements théoriques de l'approche analytique. Puis nous envisageons l'application de cette dernière approche à la commande d'un réacteur continu parfaitement mélangé, à la commande d'un bras manipulateur et à la commande d'un moteur à courant continu. Enfin, nous considérons une étude comparative montrant l'apport de l'approche analytique, tant du côté performances de poursuite que du côté temps de calcul.

CHAPITRE 1

PRINCIPES FONDAMENTAUX DES RESEAUX DE NEURONES

1 PRINCIPES FONDAMENTAUX DES RESEAUX DE NEURONES

1.1 Introduction

La recherche sur les réseaux de neurones a connu un développement important ces dernières années, tant du côté architecture où plusieurs modèles sont proposés, que du côté algorithmes d'apprentissage utilisés pour entraîner ces réseaux. En effet, ces travaux de recherche ont montré que les réseaux de neurones sont des approximateurs universels, ce qui permet de modéliser n'importe quel système non linéaire, et ils sont principalement, dotés de deux propriétés importantes : l'apprentissage et la généralisation. Ces travaux de recherche ont donné lieu à des applications très intéressantes des réseaux de neurones dans plusieurs domaines, et en particulier le domaine de la commande des systèmes non linéaires [2], [8], [14], [25], [26], [29], [46].

Le but de ce chapitre est de rappeler brièvement les concepts de base des réseaux de neurones intervenant dans la modélisation et la commande des systèmes non linéaires. Nous présentons les éléments de base qui entrent dans leur constitution, leurs architectures ainsi que les méthodes d'apprentissage les plus utilisées dans la commande des systèmes. Nous présentons aussi quelques structures de commande à base des réseaux de neurones fréquemment rencontrées dans la littérature

1.2 Le neurone formel

Le premier modèle du neurone formel date des années quarante. Il a été présenté par Mc Culloch et Pitts [1], [2]. S'inspirant de leurs travaux sur les neurones biologiques ils ont proposé le modèle suivant (figure 1.1) :

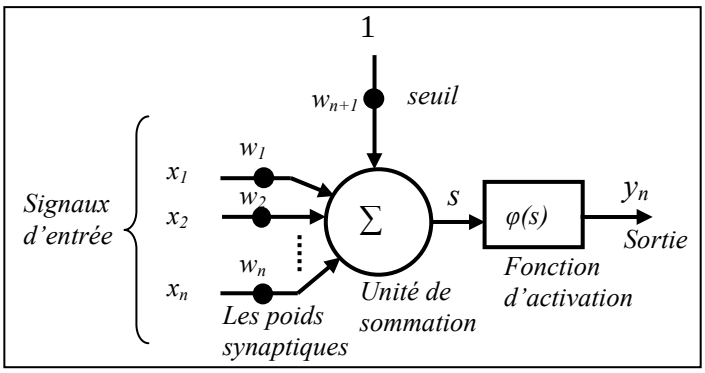


Figure 1.1 : Le neurone formel

Un neurone formel est une fonction algébrique non linéaire et bornée, dont la valeur dépend des paramètres appelés poids synaptiques ou poids des connexions. D'une façon plus générale, un neurone formel est un élément de traitement (opérateur mathématique) possédant *n* entrées (qui sont les entrées externes ou les sorties des autres neurones), et une seule sortie. Ce modèle est décrit mathématiquement par les équations suivantes :

$$\begin{cases} s = \sum_{i=1}^{n+1} w_i x_i \\ y_n = \varphi(s) \end{cases} \tag{1.1}$$

où x_i , w_i , φ , et y_n sont respectivement, les entrées, les poids synaptiques, la fonction d'activation et la sortie du neurone. w_{n+1} est le seuil (le biais) du neurone.

1.3 Fonctions d'activation

Les fonctions d'activation représentent généralement certaines formes de non linéarité. L'une des formes de non linéarité la plus simple, et qui est appropriée aux réseaux discrets, est la fonction signe (figure 1.2 a). Une autre variante de ce type des non linéarités est la fonction de Heaviside (figure 1.2 b). Pour la majorité des algorithmes d'apprentissage il est nécessaire d'utiliser des fonctions continues et différentiables, telles que la fonction sigmoïde unipolaire (figure 1.2 c) et la fonction sigmoïde bipolaire (figure 1.2 d). La classe, la plus utilisée des fonctions d'activation, dans le domaine de la modélisation et de la commande des systèmes non linéaires est la fonction sigmoïde bipolaire [9], [10].

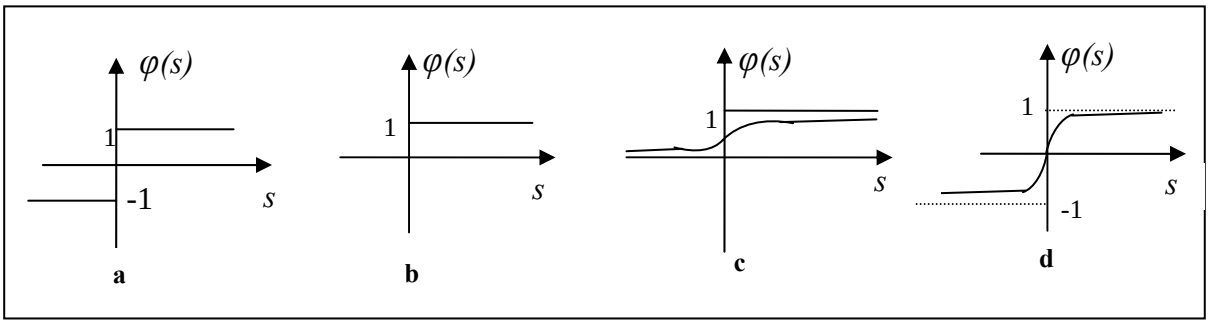


Figure 1.2 : Fonctions d'activation

1.4 C'est quoi un réseau de neurones ?

Un réseau de neurones est un ensemble d'éléments de traitement de l'information, avec une topologie spécifique d'interconnexions (architecture du réseau) entre ces éléments et une loi d'apprentissage pour adapter les poids de connexions (poids synaptiques), il est caractérisé par un parallélisme à grain très fin et à forte connectivité. Nous entendons par là que dans un réseau de neurones donné, l'information est traitée par un grand nombre de processeurs élémentaires très simples, chacun étant relié à d'autres processeurs. Ce processeur très simple est un neurone formel désigné ainsi car son fonctionnement s'inspire d'une modélisation des cellules neuronales biologiques. Ils sont dotés de deux propriétés importantes qui sont à l'origine de leur intérêt pratique dans des domaines très divers [3], [4]:

- **Capacité d'adaptation ou d'apprentissage** qui permet au réseau de tenir compte des nouvelles contraintes ou de nouvelles données du monde extérieur.
- **Capacité de généralisation** qui est son aptitude de donner une réponse satisfaisante à une entrée qui ne fait pas partie des exemples à partir desquels il a appris.

Selon la structure des connexions, on peut distinguer deux types des réseaux de neurones : les réseaux statiques et les réseaux dynamiques [5].

1.4.1 Les réseaux statiques

Dans les réseaux statiques, dits aussi réseaux non récurrents, lorsqu'un signal est présenté en entrée du réseau, la réponse de ce dernier est calculée instantanément, après la traversée de ses couches de l'entrée vers la sortie, dans un sens unique. La sortie de n'importe quel neurone, d'un réseau statique, ne peut pas être appliquée directement sur son entrée ni indirectement à travers d'autres neurones. Un réseau de neurones statique réalise une ou plusieurs fonctions algébriques de ses entrées, par composition des fonctions réalisées par chacun de ses neurones.

Le modèle le plus populaire des réseaux statiques est celui des réseaux multicouches, appelés aussi réseaux MLP (abréviation de multilayered perceptron). Ce réseau est constitué d'une couche d'entrée, d'une couche de sortie et de plusieurs couches cachées. Chaque neurone de la couche l est connecté à tous les neurones de la couche $l+1$. Ce modèle est considéré dans les paragraphes suivants.

1.4.2 Les réseaux dynamiques

Dans un réseau dynamique appelé aussi réseau récurrent, chaque neurone peut recevoir des informations de tous les autres neurones et leur renvoie lui-même des signaux. On peut distinguer deux catégories de ces réseaux : les réseaux entièrement connectés tels que le réseau de Hopfield [4] et les cartes de Kohonen [6], la deuxième catégorie est celle des réseaux à retour d'état et les réseaux à retour de sortie [5]. Les réseaux dynamiques sont moins utilisés dans le domaine de la commande des systèmes que les réseaux statiques ; ceci est peut être lié à la difficulté de l'apprentissage de ce type de réseaux. Plus de détailles peuvent être trouvés dans la référence [7].

1.5 L'apprentissage des réseaux de neurones

L'information que peut acquérir un réseau de neurones est représentée dans les poids des connexions entre les neurones. L'apprentissage consiste donc à ajuster ces poids de telle façon que le réseau présente certains comportements désirés. En d'autres termes, l'apprentissage des réseaux de neurones consiste à ajuster les poids synaptiques de telle manière que les sorties du réseau (le modèle) soient, pour les exemples (les entrées) présentés au réseau lors de l'apprentissage, aussi proches que possibles des sorties "désirées". Il existe trois types d'apprentissage :

L'apprentissage supervisé pour lequel on dispose de la sortie désirée et qui consiste à ajuster les poids synaptiques de telle sorte à minimiser l'écart entre la sortie désirée et la sortie du réseau. C'est le type d'apprentissage utilisé dans le cadre de notre travail.

L'apprentissage non supervisé pour lequel le réseau de neurones organise lui-même les entrées qui lui sont présentées de façon à optimiser un critère de performances donné.

L'apprentissage par renforcement pour lequel le réseau de neurones est informé d'une manière indirecte sur l'effet de son action choisie. Cette action est renforcée si elle conduit à une amélioration des performances.

1.6 Les réseaux MLP statiques

Ce type de réseaux de neurones est très utilisé dans le domaine de la commande des systèmes [2], [8], [9] ; ceci est en raison de la simplicité de leurs algorithmes d'apprentissage. Un réseau MLP est organisé en plusieurs couches : une couche passive dite couche d'entrée, une couche de sortie et une ou plusieurs couches cachées. Les neurones des deux couches consécutives sont entièrement connectés, par contre les neurones de la même couche ne sont pas connectés entre eux. Le nombre total de couches d'un réseau MLP est égale au nombre de couches cachées plus la couche de sortie. La figure 1.3 présente l'exemple d'un réseau MLP à deux couches. Dans la majorité des applications de la commande un réseau de neurone à une seule couche cachée est suffisant [8].

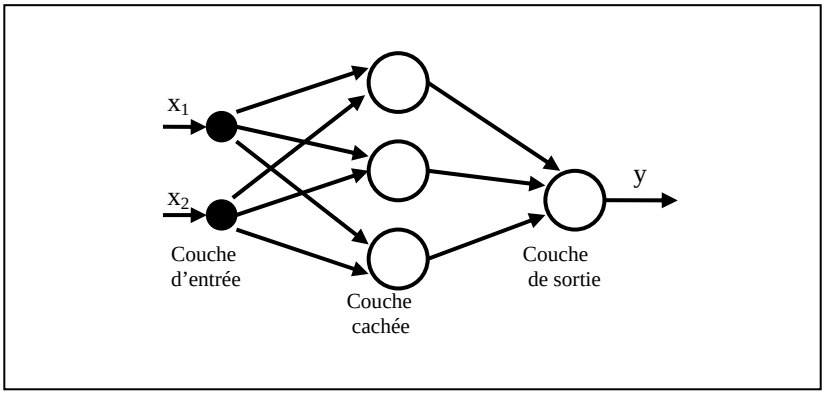


Figure 1.3 : réseau MLP à deux couches

1.6.1 Description mathématique du réseau

Un réseau MLP à L couches, n entrées et m sorties réalise une application de R^n dans R^m. Cette application résulte de la composition des applications locales réalisées par les couches du réseau. La sortie du $j^{ème}$ neurone de la couche l est donnée par :

$$\begin{aligned} s_j^l &= \sum_{i=l}^{n_{l-1}} w_{ji}^l y_{ri}^{l-1} + w_{jn_{l-1}+1}^l \\ y_{rj}^l &= \varphi(s_j^l) \qquad avec \quad l = 1...L,\ j = 1...n_l \end{aligned} \tag{1.2}$$

où n_l est le nombre de neurones sur la couche l, w_{ji}^l est le poids de la connexion entre le $j^{ème}$ neurone de la couche l et le $i^{ème}$ neurone de la couche l-1, $w_{jn_{l-1}+1}^l$ est le biais du $j^{ème}$ neurone de la couche l, et y_{ri}^{l-1} est la sortie du $i^{ème}$ neurone de la couche l-1.

$y_{ri}^0 = x_i$ avec $i=1...n$ est la $i^{ème}$ composante du vecteur d'entrée, $n=n_0$ étant le nombre d'entrées, et $y_{ri}^L = y_{ri}$, avec $i=1...m$, est la $i^{ème}$ composante du vecteur de sortie, $m=n_L$ étant le nombre de sorties.

1.6.2 Algorithmes d'apprentissage

Comme nous l'avons indiqué dans le paragraphe 1.5, l'apprentissage supervisé pour les réseaux de neurones, consiste à adapter les poids synaptiques de telle manière que les sorties du réseau de neurones soient, pour les exemples utilisés lors de l'apprentissage, aussi proches que possibles des sorties désirées. La plupart des algorithmes d'apprentissage des réseaux de neurones sont basés sur les méthodes du gradient [5], [10]: ils cherchent à minimiser une fonction de coût de la forme donnée par l'équation suivante :

$$J = \frac{1}{2}\sum_{q=1}^{m}(y_{rq}(X_p) - y_q^d(X_p))^2 \tag{1.3}$$

où y_{rq} et y^d_q sont les sorties du réseau et les sorties désirées pour le vecteur d'entrée X_p.

Cette optimisation se fait d'une manière itérative, en modifiant les poids en fonction du gradient de la fonction de coût selon la loi d'adaptation suivante :

$$w_{ji}^{l}(k+1) = w_{ji}^{l}(k) - \eta \frac{\partial J}{\partial w_{ji}^{l}} \qquad l = 1,2,\ldots,L \quad j = 1,\ldots,n_l+1, \quad i = 1,\ldots,n_{l-1}+1 \tag{1.4}$$

où : η est une constante positive ($0 < \eta < 1$) appelée "taux d'apprentissage".

Le gradient est calculé par une méthode spécifique aux réseaux de neurones, dite méthode de rétropropagation. Dans cette méthode, de même que l'on est capable de propager un signal provenant des neurones d'entrée vers la couche de sortie, il est possible de calculer le gradient de la fonction de coût en rétropropageant l'erreur commise en sortie vers les couches cachées, d'où le nom rétropropagation. La méthode de rétropropagation est décrite en détail dans [5], [7], [11].

Dans sa version complète, l'algorithme de rétropropagation comporte un certain nombre de paramètres, qui sont difficile à régler, comme par exemple le pas du gradient (taux d'apprentissage). Il est clair que ce dernier paramètre a une importance comme dans toute descente de gradient. S'il est très faible, la convergence risque d'être très lente, et s'il est trop élevé, un problème d'oscillation peut survenir. Plusieurs techniques visant à accélérer la convergence de l'algorithme ont été proposées dans les références suivantes [5], [10].

Une autre méthode utilisée pour l'apprentissage des réseaux MLP est la méthode de Levenberg-Marquardt. Cette méthode, qui est basée sur le calcul de la dérivée seconde de la fonction de coût, est beaucoup plus rapide et efficace que celle de rétropropagation. La loi d'adaptation des poids est dans ce cas, donnée par :

$$w_{ji}^{l}(k+1) = w_{ji}^{l}(k) - \eta (H + \lambda I)^{-1} \frac{\partial J}{\partial w_{ji}^{l}} \qquad l = 1,2,\ldots,L \quad j = 1,\ldots,n_l+1, \quad i = 1,\ldots,n_{l-1}+1 \tag{1.5}$$

où H est la matrice Hessienne de la fonction de coût, I est une matrice identité et λ est un facteur d'adaptation. Les détailles de l'algorithme d'apprentissage en utilisant cette méthode peuvent être consultées dans [8]. Dans toutes les simulations réalisées, dans le cadre de ce travail, nous avons utilisé cet algorithme.

1.7 Choix de la structure du réseau de neurones

En commande neuronale on fait fréquemment appel aux propriétés d'approximation universelle de certains réseaux de neurones pour réaliser une fonction de commande. Ces propriétés désignent la capacité d'approcher une fonction arbitraire d'aussi près que l'on veut, à condition de disposer d'un réseau comportant un nombre suffisant de neurones. Il existe un certain nombre de théorèmes garantissant les capacités d'approximation universelle sous certaines conditions (continuité de la fonction à approcher, nombre suffisant de neurones dans les couches cachées), même si le réseau ne comporte q'une seule couche cachée [13], [14]. Cependant, ces théorèmes ne fournissent pas de méthodes systématiques constructives guidant le choix de la topologie à adopter pour une application particulière. Les questions qui se posent : (i) faut-il utiliser une ou deux couches cachées? (ii) Combien doit-il y avoir de neurones sur la ou les couches cachées?

Nous notons qu'ils n'existent pas de méthodes générales permettant de dimensionner, d'une manière automatique, le réseau en fonction du degré de complexité de l'application envisagée. Citons néanmoins deux approches fréquemment utilisées pour le choix du nombre de neurones [8], [9]. Dans la première approche, on démarre à partir d'un réseau de taille assez grande, ensuite après apprentissage, on essaye de la réduire à sa plus petite valeur possible. La deuxième approche est tout à fait l'inverse de la première ; on démarre à partir d'un réseau de taille réduite, ensuite à chaque fois on ajoute un neurone jusqu'à ce que les performances voulues sont obtenues.

Bien qu'on puisse construire théoriquement des réseaux avec un nombre important de couches cachées, en pratique il est rare d'en avoir plus de trois [52]. Les raisons en sont multiples. La première est

une raison de temps ; l'algorithme d'apprentissage peut être extrêmement lent. Une autre raison est liée à la difficulté d'analyse et d'implémentation du réseau. Il est clair qu'en augmentant indéfiniment le nombre de neurones on peut améliorer la précision de l'apprentissage (c-à-d la propriété d'approximation universelle) ainsi que les propriétés de robustesse par rapport aux défaillances internes du réseau, mais que cela posera des problèmes de mauvaise généralisation (propriété de généralisation). Ceci à cause du phénomène de sur-paramétrisation (overfiting). La propriété de généralisation, qui est très importante dans la commande neuronale, désigne la capacité d'élargir le champ et l'influence de chaque expérience d'apprentissage de façon à fournir une réponse appropriée pour des situations futures plus ou moins similaires. Les propriétés d'approximation universelle et de généralisation sont souvent conflictuelles [52]. Par conséquent, face à cette situation conflictuelle, il faudra trouver une solution de compromis : c'est un des problèmes fondamentaux de l'emploi de réseaux de neurones pour l'approximation de fonctions.

1.8 Sur-apprentissage (overfitting)

Lorsque l'erreur d'apprentissage est significativement meilleure que celle de la généralisation on parle alors de sur-apprentissage (overfitting). Quand ceci se produit, il est possible que le réseau a modélisé le bruit attaché aux données d'apprentissage, plutôt que la relation liant les entrées et les sorties du modèle à apprendre. Une mauvaise généralisation du réseau peut être la cause des éléments suivants :

- l'ensemble de données d'apprentissage est incomplet ;
- le nombre de neurones est important (sur-paramétrisation) ;
- le nombre d'itérations d'apprentissage est élevé.

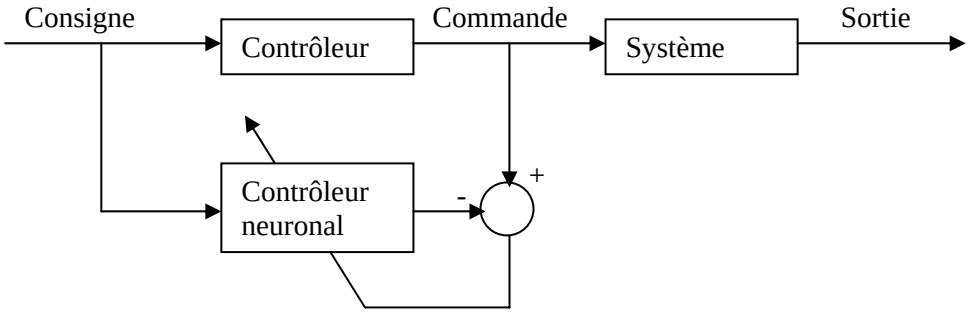
Si la mauvaise généralisation du réseau est la cause de l'utilisation d'un ensemble de données incomplet alors les données de test possédant la plus grande valeur de l'erreur de généralisation sont ajoutées à l'ensemble des données d'apprentissage, un nouveau ensemble de données de test est formé et le réseau est entraîné de nouveau. Pour éviter le problème de sur-apprentissage, on doit aussi limiter le nombre de neurones et celui des itérations d'apprentissage au minimum nécessaire. Une autre manière pour surmonter le problème de sur-apprentissage est d'utiliser la méthode de 'cross validation'. Cette méthode consiste à évaluer à chaque itération d'apprentissage la capacité de généralisation du réseau. A ce fait elle utilise deux ensembles de données pendant la phase d'apprentissage, un ensemble pour faire l'apprentissage et l'autre ensemble pour faire le test. A chaque itération les erreurs d'apprentissage et de test sont calculées, par exemple, dans les cas d'un réseau sur-paramétré et au-delà d'un certain nombre d'itérations d'apprentissage l'erreur d'apprentissage continuer à diminuer par contre celle de test commence à augmenter. Le problème de sur-apprentissage est considéré et illustré par des exemples sous forme de simulation dans la référence [9].

1.9 Commande par réseaux de neurones

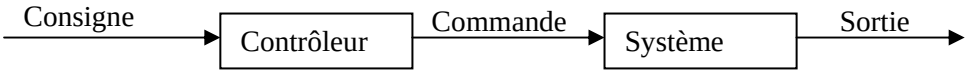
Ces dernières années, un grand intérêt a été porté sur l'utilisation des réseaux de neurones pour l'identification et la commande des systèmes non linéaires. Ceci est dû, principalement, à leurs capacités d'apprentissage et de généralisation ainsi qu'à leur parallélisme. Ces propriétés permettent d'envisager une synthèse de nouvelles lois de commande prometteuses. Des recherches significatives dans le domaine de la commande intelligente utilisant les réseaux de neurones se poursuivent, et plusieurs applications ont trouvé leurs places sur le marché [53]. On peut distinguer deux classes de structures de commande : la commande directe où le contrôleur est un réseau de neurones et la commande indirect pour laquelle la conception de la loi de commande est basée sur l'utilisation d'un modèle neuronal du processus. La méthode de commande que nous avons envisagé dans notre travail appartient à cette deuxième classe. Nous présentons au cours de ce paragraphe quelques structures de commande à base des réseaux de neurones fréquemment rencontrées dans la littérature.

1.9.1 Commande supervisée

Cette technique de commande consiste à imiter un système de commande existant (par exemple, un opérateur humain), en utilisant les propriétés d'approximation et d'apprentissage sur la base d'exemples des réseaux de neurones [54]. La structure de cette commande est illustrée par la figure 1.4. La sortie du contrôleur à imiter sera une fonction à approcher par un réseau de neurones, une fois cette tâche accomplie, le réseau sera implanté à la place de l'ancien contrôleur pour conduire le système en imitant son comportement. On peut s'interroger sur l'intérêt de cette méthode dans la mesure où l'on dispose déjà d'un contrôleur. Plusieurs arguments peuvent répondre à cet objectif, en effet le contrôleur conventionnel qui sert à entraîner le réseau peut être un dispositif difficile à mettre en œuvre en dehors des séries d'expériences qui servent à entraîner le réseau Il peut s'agir d'un opérateur humain qu'on ne peut pas affecté en permanence à la tâche de commande considérée. Cette méthode s'impose aussi, dans le cas où la commande utilisée ne suit pas une loi explicite, d'où la difficulté de la reproduire autrement que par apprentissage. Par ailleurs, cette méthode de contrôle peut fournir un nouveau contrôleur plus général grâce à la caractéristique de généralisation des réseaux de neurones.



a. phase d'apprentissage



b. phase opérationnelle

Figure 1.4 : Structure de la commande supervisée

1.9.2 Commande directe par modèle inverse (apprentissage général)

Cette approche appelée aussi architecture d'apprentissage général [55], [56], procède en deux étapes successives : dans un premier temps on construit le modèle inverse du système en utilisant la sortie du système comme entrée d'un réseau neuronal (Figure 1.5.a). Ce réseau est entraîné par rétropropagation afin de produire une sortie la plus proche possible de la commande. Dans un second temps, le réseau est placé devant le système pour le conduire en boucle ouverte (figure 1.5.b). On en voit directement les inconvénients de cette approche : l'apprentissage se fait hors ligne, les entrées de commande choisies pour la phase d'apprentissage ne conviendront peut être pas pour l'objectif recherché, l'existence est l'unicité d'un modèle inverse est un problème non trivial. On peut songer à combiner les deux phases pour supprimer certains de ces désavantages (figure 1.5.c), mais cette architecture ne cherche pas à réduire l'erreur à la sortie du système.

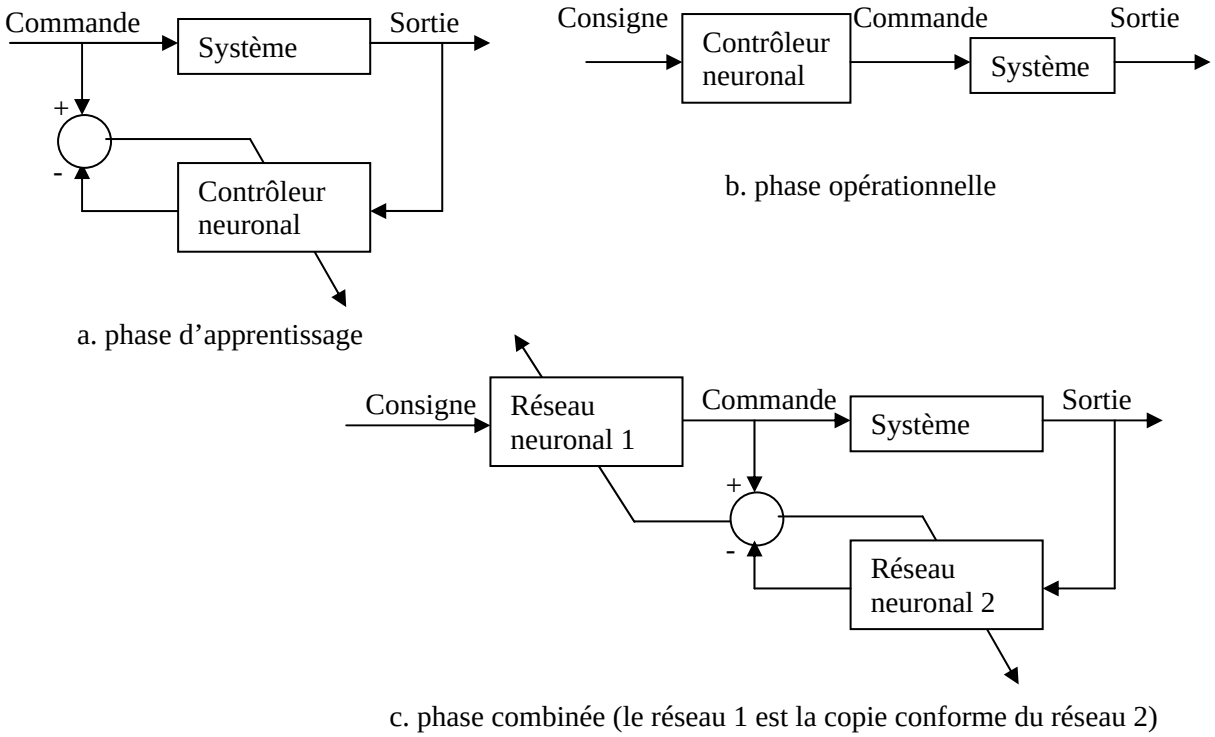


Figure 1.5 : commande directe par modèle inverse (apprentissage général)

1.9.3 Commande directe par modèle inverse (apprentissage spécialisé)

La structure de cette approche est représentée à la figure 1.6 : un réseau neuronal conduit directement le processus et utilise l'erreur à la sortie du processus (ou un indice de performance en général) pour ajuster ces poids en ligne [8], [52]. La plupart des inconvénients de l'apprentissage général sont ainsi évités : il ne faut pas (nécessairement) de phase d'apprentissage préliminaire, l'adaptation est continuelle et il y a naturellement focalisation directe sur le domaine d'intérêt. Cependant ce type d'architecture nécessite une certaine connaissance a priori du processus pour pouvoir calculer correctement les modifications à apporter, par rétropropagation de l'erreur, aux poids. Dans le but de fournir cette information on utilise un réseau neuronal, appelé réseau émulateur, comme modèle du processus et en déduit le Jacobien, par une extension évidente de l'algorithme de rétropropagation. Cette approche peut procéder en deux phases successives : une phase d'identification (figure 1.6.a) suivie d'une phase d'apprentissage du régulateur (figure 1.6.b). Elle peut aussi fonctionner complètement en ligne en imbriquant les deux phases (figure 1.6.c) : on utilise alors simultanément deux erreurs, d'une part l'erreur entre sortie prédite par le réseau émulateur et la sortie réelle du processus et, d'autre part, l'erreur entre la sortie désirée et la sortie réelle du processus. La première erreur est directement rétropropagée dans le réseau émulateur, tandis que la deuxième est rétropropagée dans le réseau régulateur au travers du réseau émulateur (considéré comme une couche supplémentaire du réseau régulateur, mais à poids non modifiables).

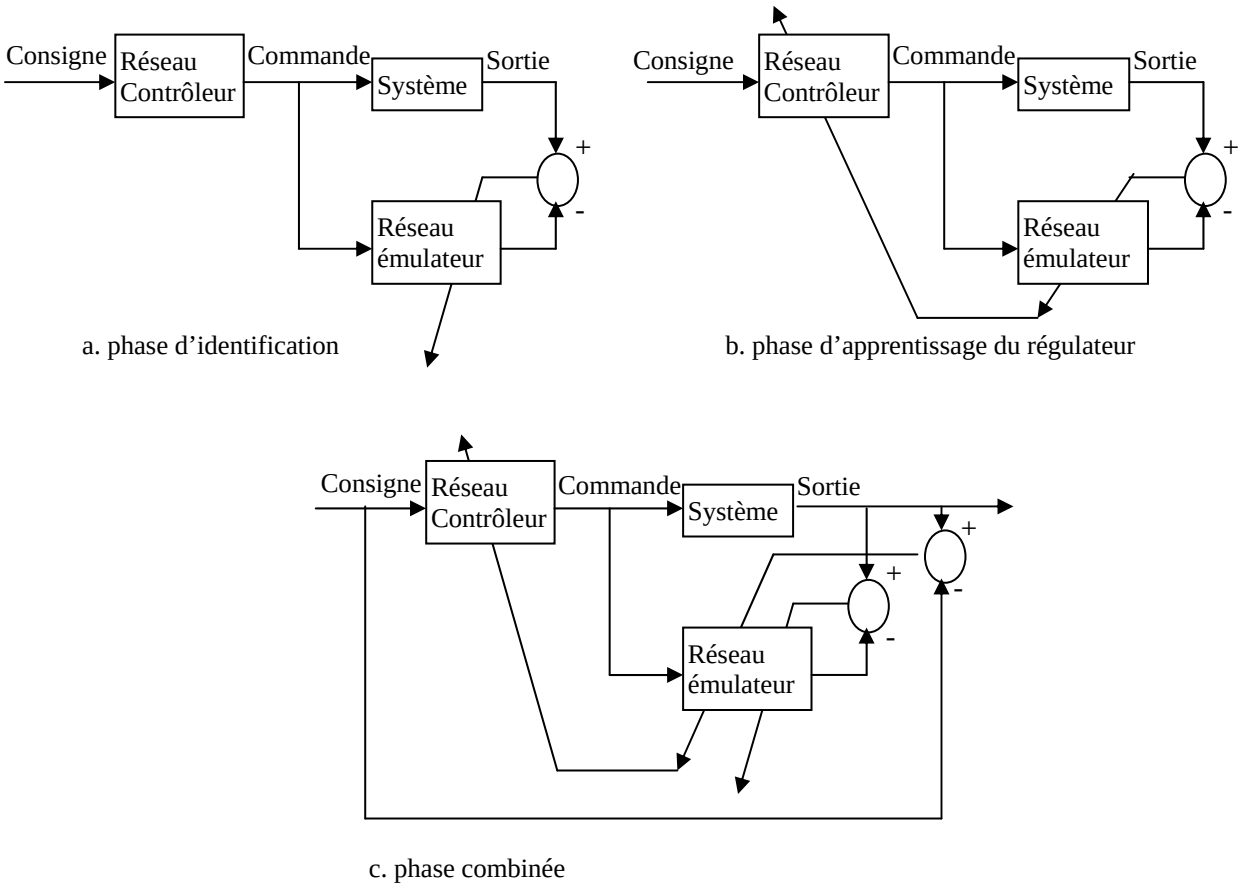


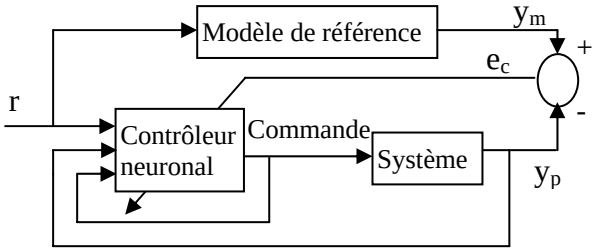
Figure 1.6 : commande directe par modèle inverse (apprentissage spécialisé)

1.9.4 Commande adaptative à modèle de référence

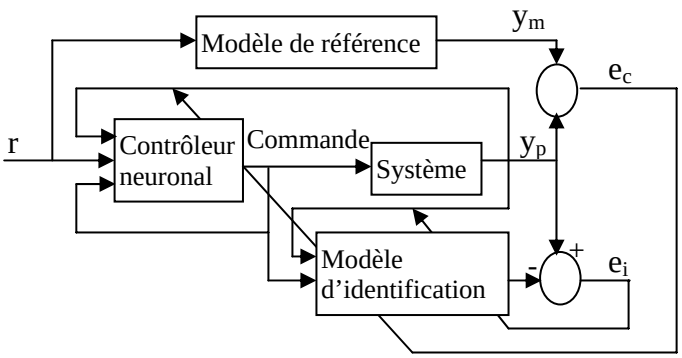
Dans cette stratégie de commande, un modèle de référence est utilisé pour spécifier le comportement désiré du système à commander. Le système de commande, quant à lui est utilisé pour forcer la sortie du système à suivre celle du modèle de référence. En se basant sur ce principe, Narendra [2], [29] a proposé deux structures de commande, l'une directe et l'autre indirecte.

Dans la structure directe (figure 1.7.a), les paramètres du contrôleur sont directement adaptés pour réduire l'erreur en sortie du système. Une fois la phase d'apprentissage est achevée, l'action de contrôle est activée de telle sorte que la sortie du système suive celle du modèle de référence. Puisque le système est placé entre le réseau de neurones et l'erreur de sortie, on ne peut pas ajuster directement les poids du réseau contrôleur. Cette difficulté peut être surmonté en utilisant la structure indirecte.

Dans la structure indirecte (figure 1.7.b), un réseau de neurones est utilisé pour identifier la dynamique directe du système (modèle d'identification), les poids de ce réseau peuvent être adaptés en ligne pour suivre en permanence le comportement dynamique du système. L'erreur utilisée pour adapter les poids du contrôleur neuronal est obtenue en rétropropageant l'erreur de commande (l'écart entre la sortie du modèle de référence et celle du système) dans le modèle d'identification. Cette méthode utilise deux réseaux de neurones, le premier, monté en parallèle du système, assure la fonction d'identification, le deuxième réseau assure la fonction de commande en forçant la sortie du système à suivre celle du modèle de référence.



a. méthode directe



b. méthode indirecte

Figure 1.7 : Commande adaptative à modèle de référence

1.9.5 Autres approches

Les nombreuses applications décrites dans la littérature sont le plus souvent des versions plus ou moins modifiées des approches précédentes. Chacun y ajoute quelques améliorations susceptibles d'en améliorer les performances pour son application spécifique. Citons néanmoins trois approches particulièrement intéressantes. La première est basée sur la régulation par modèle interne[46] ; elle utilise deux réseaux de neurones dont l'un est le modèle direct et l'autre le modèle inverse du système, après apprentissage, ces réseaux sont placés judicieusement dans une boucle de régulation afin d'assurer la réjection de certaines perturbations. La deuxième approche [57] a pour principe de combiner un régulateur linéaire optimal avec un régulateur neuronal ayant pour rôle de compenser les incertitudes (non linéarités) sur le système, non considérées lors de la conception du régulateur linéaire. Elle fait intervenir deux réseaux : l'un, placé en parallèle avec un modèle linéaire du système, a pour fonction de modéliser les incertitudes dont le modèle linéaire ne tient pas en compte ; l'autre se charge de compenser l'effet de ces incertitudes et fonctionne, lui aussi, en parallèle avec un régulateur linéaire construit sur la base du modèle linéaire correspondant. Enfin, la troisième approche [58] procède de façon complètement différente, en se basant sur les capacités de reconnaissance de forme de certains réseaux neuronaux pour identifier un processus à partir de son réponse indicielle ; la phase d'identification se réduit alors à une simple classification et suivant les résultats de cette phase, on active une certaines stratégie de commande. A chaque classe est associé une stratégie, suivant les principes du 'gain scheduling'.

1.10 Conclusion

Nous avons rappelé brièvement, dans ce chapitre, les éléments de base des réseaux de neurones à savoir : le modèle du neurone formel, les différentes architectures des réseaux de neurones et l'apprentissage de ces réseaux. En particulier, nous avons considéré avec plus de détail les réseaux de neurones de type MLP. Nous avons aussi présenté quelques structures de commande à base des réseaux de neurones fréquemment rencontrées dans la littérature. La compréhension des notions exposées, dans ce chapitre, est nécessaire pour la réalisation des travaux qui seront présentés dans les chapitres 3 et 4.

CHAPITRE 2

COMMANDE PREDICTIVE GENERALISEE

2 COMMANDE PREDICTIVE GENERALISEE

2.1 Introduction

La commande prédictive est une méthode relativement récente ; elle a commencé à donner ses premiers résultats théoriques et pratiques à la fin des années 1970 [15]. Dans les années 1980, plusieurs méthodes basées sur les mêmes concepts ont été développées. Parmi ces méthodes, on peut citer la commande prédictive généralisée [16], qui a été la technique la plus largement utilisée par la suite. Toutes ces méthodes prédictives reposent sur la même philosophie ; à savoir créer un effet anticipatif.

Ce chapitre a pour but tout d'abord de présenter les grandes idées communes aux méthodes prédictives, puis, dans le cas particulier de la commande prédictive généralisée (GPC) qui nous intéresse pour la commande prédictive à modèle neuronal, d'introduire le modèle de représentation du système utilisé par la commande prédictive généralisée ainsi que le critère quadratique à minimiser et enfin de développer l'élaboration de la loi de commande correspondante à cette méthode.

2.2 Philosophie de la commande prédictive

Le concept clé de la commande prédictive réside dans la création d'un effet anticipatif. On exploite pour cela les connaissances explicites sur l'évolution de la trajectoire à suivre dans le futur (sur un horizon fini), ensuite, on minimise un critère pourtant sur l'écart entre la sortie prédite et la sortie future désirée. La connaissance à priori de la trajectoire désirée, au moins sur un horizon de quelques points de l'instant présent, permet de tirer partie de toutes les ressources de la méthode restreint nécessairement le domaine d'application à la commande de systèmes pour lesquels la trajectoire à suivre est parfaitement connue. C'est le cas de la commande numérique de machines outils (découpe de pièces), de la commande de bras de robots ou des processus chimiques.

Toutes les méthodes de la commande prédictive sont fondées sur les points suivants :

- Obtention d'un modèle numérique du système pour prédire le comportement futur du système. Cette particularité permet de classer la commande prédictive dans la grande famille des commandes à base de modèles (Model Based Control).
- Calcul d'une séquence de commandes futures de telle sorte à minimiser une fonction de coût quadratique à horizon fini. Cette fonction de coût porte classiquement sur l'erreur entre la trajectoire de référence et la sortie prédite, et l'énergie utilisée pour y parvenir. On peut dire que le but de la stratégie prédictive est de faire coïncider la sortie du système avec la trajectoire (ou consigne) désirée dans le futur, en accord avec le diagramme temporel de la figure 2.1.
- Application du premier élément de la séquence calculée.
- Répétition du processus à la période d'échantillonnage suivante, selon le principe de l'horizon fuyant (receding horizon).

Les principes de la commande prédictive qui viennent d'être exposés permettent d'établir le schéma fonctionnel de la figure 2.2.

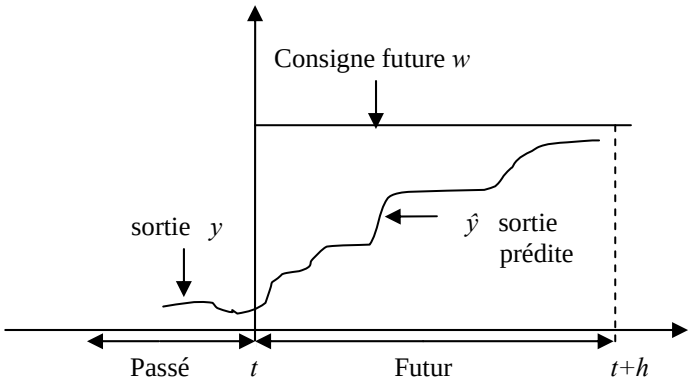


Figure 2.1 : Diagramme temporel de la prédiction à horizon fini.

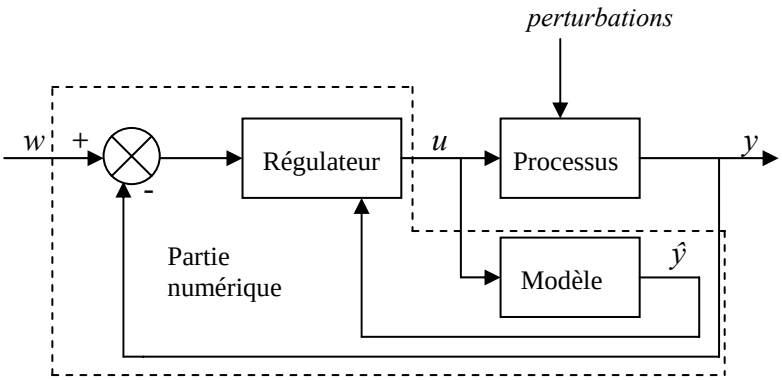


Figure 2.2 : Schéma de principe d'une commande prédictive à base d'un modèle

La différence entre les méthodes prédictives existantes réside, notamment, dans l'utilisation de modèles et critères différents. Une introduction historique sur les diverses méthodes de commande prédictive peut être trouvée dans [17] [18].

Grâces à ses concepts intuitifs et aux bons résultats obtenus, la commande prédictive a été implantée dans un grand nombre d'applications industrielles, tels que les processus chimiques, qui ont été les premiers à utiliser ce type de commande, les processus de distillation, l'industrie pétrolière et les systèmes électromécaniques. La commande prédictive permet de contrôler un grand nombre de processus : instables, à retard pur et à non minimum de phase. Elle est, dans la plupart des cas, une commande discrète, même s'il existe malgré tout des commandes prédictives continues. Le travail réalisé dans le cadre de cette thèse se base sur l'approche discrète de la commande prédictive.

2.3 Modèle de prédiction

L'obtention d'un modèle de prédiction est une condition nécessaire pour toute commande prédictive, puisqu'il permet de prédire le comportement futur du système. Il peut être obtenu directement par transformée en Z de la fonction de transfert continue du système, si l'on dispose de cette fonction, ou être le résultat d'une identification hors ligne du système. La commande prédictive généralisée n'est pas restreinte à un type de représentation particulier, mais historiquement adopte le modèle *CARIMA* (Controlled AutoRegressive Integrated Moving Average), de la forme [16]:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \frac{C(q^{-1})}{\Delta(q^{-1})}\xi(t) \tag{2.1}$$

où $u(t)$, $y(t)$ et $\xi(t)$ sont respectivement l'entrée, la sortie et le signal de perturbation du système. $\xi(t)$ est considéré aléatoire de moyenne nulle. Le polynôme $C(q^{-1})$ modélise l'influence du bruit sur le système. L'introduction de $\Delta(q^{-1})=1-q^{-1}$ dans le modèle de bruit assure une action intégrale dans le correcteur et permet, donc, d'annuler toute erreur statique vis-à-vis d'une entrée ou d'une perturbation en échelon. L'utilisation de ce modèle de perturbation est en fait une conséquence de la présence de perturbations de charge en échelon dans de nombreux processus industriels. Ceci est une conséquence directe du principe du modèle interne [18] : pour contrôler un système sujet à des perturbations instables, la dynamique de la perturbation doit apparaître dans la dynamique du régulateur.

Les polynômes $A(q^{-1})$, $B(q^{-1})$ et $C(q^{-1})$ sont définis par :

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots\ldots + a_{n_a} q^{-n_a} \\ B(q^{-1}) &= b_0 + b_1 q^{-1} + \ldots\ldots + b_{n_b} q^{-n_b} \\ C(q^{-1}) &= 1 + c_1 q^{-1} + \ldots\ldots + c_{n_c} q^{-n_c} \end{aligned} \tag{2.2}$$

2.4 Fonction de coût

La loi de commande est obtenue par minimisation d'une fonction de coût quadratique portant sur les erreurs futures avec un terme de pondération sur les incréments de commande :

$$J = \sum_{j=N_1}^{N_2} (\hat{y}(t+j) - w(t+j))^2 + \lambda \sum_{j=1}^{N_u} \Delta u(t+j-1)^2 \tag{2.3}$$

avec $\Delta u(t+j) = 0$ pour $j \geq N_u$.

$w(t+j)$ est la consigne appliquée à l'instant $t+j$, $\hat{y}(t+j)$ est la sortie prédite à l'instant $t+j$ et $\Delta u(t+j-1)$ est l'incrément de commande à l'instant $t+j-1$.

La fonction de coût, donnée par l'équation (2.3), nécessite la définition de quatre paramètres de réglage :

- N_1 : horizon de prédiction minimal sur la sortie ;
- N_2 : horizon de prédiction maximal sur la sortie ;
- N_u : horizon de prédiction sur la commande ;
- λ : facteur de pondération sur la commande.

Le coefficient λ permet de donner plus ou moins de poids à la commande par rapport à la sortie, de façon à assurer la convergence lorsque le système de départ présente un risque d'instabilité.

2.5 Obtention de la loi de commande

La mise en œuvre de la loi de commande nécessite la définition d'un prédicteur à j-pas et la minimisation de la fonction de coût donnée par l'équation (2.3). Deux approches différentes peuvent

être envisagées pour la définition du prédicteur qui permet de déterminer la sortie prédite à l'instant *t+j*, connaissant la sortie du système jusqu'à l'instant *t* et la commande jusqu'à l'instant *t-1*[17].

2.5.1 Approche algorithmique

En partant de la propriété fondamentale des systèmes discrets linéaires : la réponse d'un système linéaire discret à n'importe quel signal d'entrée peut se calculer par la relation de convolution suivante :

$$y(t) = \sum_{i=0}^{\infty} s(i)\Delta u(t-i)\,, \tag{2.4}$$

où *s(t)* représente la réponse indicielle du système et *Δu(t)* est un incrément du signal d'entrée, il est possible d'écrire l'expression donnant les prédictions *ŷ(t+j/t)* de la manière suivante :

$$\hat{y}(t+j/t) = \sum_{i=1}^{\infty} g_i \Delta u(t+j-i) \quad (\text{en supposant } s(0)=0)\,, \tag{2.5}$$

où $g_i = s(iT_e)$ sont les coefficients de la réponse indicielle du système. T_e étant la période d'échantillonnage.
L'expression précédente peut se décomposer en réponse forcée et réponse libre de la façon suivante :

$$\hat{y}(t+j/t) = y_f(t+j/t) + y_l(t+j/t) \tag{2.6}$$

Où le premier terme de cette équation est donné par :

$$y_f(t+j/t) = \sum_{i=1}^{j} g_i \Delta u(t+j-i), \tag{2.7}$$

il est claire que ce terme dépend uniquement des incréments de commandes futures et présente. Le deuxième terme est donné par :

$$y_l(t+j/t) = \sum_{i=0}^{\infty} g_i \Delta u(t+j-i)\,, \tag{2.8}$$

avec : *Δu(t+j-i)=0* pour *i≤j*. Ce deuxième terme dépend uniquement des incréments de commandes passées ; il s'agit de la réponse libre d'un modèle incrémental.

Ou encore :

$$y_l(t+j) = y_l(t) + \sum_{i=1}^{j} \Delta y_l(t+i) \quad , \tag{2.9}$$

soit finalement :

$$y_l(t+j/t) = y_l(t+j-1/t) + \Delta y_l(t+j) \tag{2.10}$$

Les incréments de la réponse libre sont obtenus à partir du modèle par :

$$\begin{aligned} &A(q^{-1})\Delta y_l(t+j/t) = B(q^{-1})\Delta u(t+j-1) \\ &avec \quad \Delta u(t+j) = 0 \qquad pour \quad j \geq 0 \end{aligned} \tag{2.11}$$

Pour un horizon de prédiction donné N_2, il est possible d'écrire l'équation (2.6) sous forme matricielle. Pour cela posons :

$\hat{Y} = [\,\hat{y}(t+N_1),.....,\hat{y}(t+N_2)]^T$ sorties futures prédites

$\Delta U = [\,\Delta u(t),.....,\Delta u(t+N_u-1)]^T$ incréments des commandes futures

$Y_l = [\,y_l(t+N_1),.....,y_l(t+N_2)]^T$ réponses libres

L'équation de prédiction sera donnée par :

$$\hat{Y} = G\Delta U + Y_l \tag{2.12}$$

G est la matrice des coefficients de la réponse indicielle du modèle (G est de dimension (N_2-N_1+1) x N_u) :

$$G = \begin{bmatrix} g_{N_1} & g_{N_1-1} & g_{N_1-2} & \cdots\cdots \\ g_{N_1+1} & g_{N_1} & g_{N_1-1} & \cdots\cdots \\ . & & & \\ . & & & \\ . & & & \\ g_{N_2} & g_{N_2-1} & g_{N_2-2} & \cdots\cdots g_{N_2-N_u+1} \end{bmatrix} \tag{2.13}$$

avec $g_i = s(iT_e)$, s étant la réponse indicielle du système et T_e la période d'échantillonnage.

La fonction de coût définie par l'équation (2.3) peut également s'écrire sous forme matricielle de la manière suivante :

$$J = (G\Delta U + Y_l - W)^T (G\Delta U + Y_l - W) + \lambda \Delta U^T \Delta U \tag{2.14}$$

avec : $W = [\,w(t+N_1).....w(t+N_2)]^T$

La solution optimale est obtenue par dérivation de (2.14) par rapport au vecteur des incréments de commande :

$$\frac{\partial J}{\partial \Delta U} = 2[G^T G + \lambda I_{N_u}]\Delta U + 2G^T (W - Y_l) = 0$$

soit la solution optimale :

$$\Delta U = [G^T G + \lambda I_{N_u}]^{-1} G^T (W - Y_l) \tag{2.15}$$

la commande à appliquer à l'instant présent est donnée par :

$$u(t) = u(t-1) + \Delta u(t) \tag{2.16}$$

avec $\Delta u(t)$ est le premier élément du vecteur ΔU.

2.5.2 Approche polynomiale

Cette approche permet la synthèse de la loi de commande sous forme polynomiale équivalente, à l'aide de trois polynômes. L'avantage de cette méthode est de fournir des résultats de stabilité s'analysant hors ligne, et de générer des paramètres N_1, N_2, N_u, λ aboutissant à une commande correcte pour la partie temps réel [17].

Le modèle de prédiction est toujours présenté sous forme *CARIMA*, mais il ne sera possible d'obtenir une loi polynomiale que si le prédicteur est lui-même sous forme polynomiale. Ainsi, le prédicteur générant les sorties futures est de la forme [17]:

$$y(t+j) = \underbrace{F_j(q^{-1})y(t) + H_j(q^{-1})\Delta u(t-1)}_{réponse\ \ libre} + \underbrace{G_j(q^{-1})\Delta u(t+j-1) + J_j(q^{-1})\xi(t+j)}_{réponse\ \ forcée} \tag{2.17}$$

Avec G_j représente le futur, F_j, H_j correspondant au présent et au passé, et J_j lié au perturbations. L'équation du modèle *CARIMA* (2.1) (en prenant $c(q^{-1})=1$) combinée avec celle du prédicteur (2.17) fournit le système d'équations suivant :

$$\begin{cases} A(q^{-1})\Delta(q^{-1})y(t+j) = B(q^{-1})\Delta u(t+j-1) + \xi(t+j) \\ [1-q^{-j}F_j(q^{-1})]\,y(t+j) = [G_j(q^{-1}) + q^{-j}H_j(q^{-1})]\,\Delta u(t+j-1) + J_j(q^{-1})\xi(t+j) \end{cases} \tag{2.18}$$

En considérant l'égalité des fonctions de transfert, on obtient les deux équations fondamentales :

$$\begin{cases} \Delta(q^{-1})A(q^{-1})J_j(q^{-1}) + q^{-j}F_j(q^{-1}) = 1 \\ G_j(q^{-1}) + q^{-j}H_j(q^{-1}) = B(q^{-1})J_j(q^{-1}) \end{cases} \tag{2.19}$$

La première équation est une équation diophantine, se résolvant de façon récursive [17]. La résolution des deux équations précédentes permet de déterminer, d'une façon explicite et unique les quatre polynômes inconnus. En faisant l'hypothèse que la meilleure prédiction du terme lié au perturbations est sa moyenne (ici nulle dans le cas du bruit blanc centré), le prédicteur optimal est défini d'une façon unique, dés que les polynômes sont connus, par la relation :

$$\hat{y}(t+j/t) = F_j(q^{-1})y(t) + G_j(q^{-1})\Delta u(t+j-1) + H_j(q^{-1})\Delta u(t-1) \tag{2.20}$$

avec :
$\deg(J_j(q^{-1})) = \deg(G_j(q^{-1})) = j-1$, $\deg(F_j(q^{-1})) = \deg(A(q^{-1}))$, $\deg(H_j(q^{-1})) = \deg(B_j(q^{-1}))$ -1.

Pour simplifier les notations, l'équation précédente peut s'écrire sous la forme matricielle suivante :

$$\hat{Y} = G\Delta U + IF\,y(t) + IH\,\Delta u(t-1) \tag{2.21}$$

où :

$$IF = [F_{N_1}(q^{-1}),.....,F_{N_2}(q^{-1})]^T$$
$$IH = [H_{N_1}(q^{-1}),.....,H_{N_2}(q^{-1})]^T \tag{2.22}$$

G est la matrice formée des coefficients g^j_i des polynômes G_j. Il a été prouvé que les coefficients g^j_i correspondent aux valeurs des coefficients g_i [17]. Cette matrice G est donc identique à celle définie dans l'équation (2.13). On en déduit ensuite la forme matricielle de la fonction de coût :

$$J = (G\Delta U + IF\, y(t) + IH\, \Delta u(t-1) - W)^T (G\Delta U + IF\, y(t) + IH\, \Delta u(t-1) - W) + \lambda \Delta U^T \Delta U^T \tag{2.23}$$

où $\hat{Y}$, ΔU et W sont identiques à ceux définies dans le paragraphe précédent.

La minimisation analytique de l'équation (2.23) fournit la loi de commande optimale :

$$\Delta U = -(G^T G + \lambda I_{N_u})^{-1}(IF\, y(t) + IH\Delta u(t-1) - W) \tag{2.24}$$

La commande à appliquer à l'instant présent est donnée par :

$$u(t) = u(t-1) + \Delta u(t) \tag{2.25}$$

avec $\Delta u(t)$ est le premier élément du vecteur ΔU.

2.6 Choix des paramètres de réglage

La fonction de coût définie par l'équation (2.3) comprend quatre paramètres de réglage que l'utilisateur doit fixer. Nous énonçons ci-dessous quelques idées guidant le choix de ces paramètres, obtenues à partir de l'étude d'un grand nombre de systèmes types monovariables non dérivateurs [16], [17], [19].

- Choix des horizons de prédiction N_1 et N_2

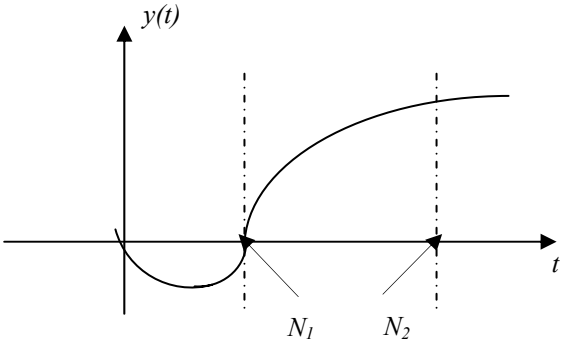


Figure 2.3 : choix de N_1 et N_2 (système à déphasage non minimal)

Le produit N_1T_e est choisi égal au retard pur du système. Ainsi pour un système ne présentant pas de retard ou un retard mal connu ou variable, N_1 est choisi égal à 1.

N_2 est choisi de sorte que le produit N_2T_e soit de l'ordre du temps de réponse du système. En général, il est possible de dire que les valeurs de N_2 au-delà du temps de réponse du système

n'apportent aucune information supplémentaire. Par ailleurs, plus N_2 est grand, plus le système corrigé est stable est lent.

- Choix de l'horizon de prédiction sur la commande N_u

Il convient pour le paramètre N_u de se contenter de $N_u = 1$ afin de tenir compte de sa faible influence sur la marge de phase et la marge de gain [17]. Dans d'autres travaux [20], l'auteur propose de choisir N_u égal au nombre d'états du système mai souligne que la valeur $N_u = 1$ est très souvent suffisante pour beaucoup d'applications relativement simples.

- Choix du facteur de pondération λ

Si $\lambda=0$, on minimise uniquement la différence entre la référence et la sortie prédite. Il peut donc en résulter une commande très forte pouvant faire diverger le processus réel. D'autre part, si λ est très élevé, on pondère alors excessivement la commande qui n'est plus assez dynamique pour obtenir le ralliement à la consigne. Une relation permettant de déterminer rapidement la valeur de λ apportant au système plus de stabilité, est donnée par [17] :

$$\lambda_{opt} = trace(G^T G)$$

2.7 Commande prédictive avec contraintes

En pratique, tous les systèmes sont sujets à des multiples contraintes, provenant à la fois de limitations physiques sur les signaux d'entrée, d'état et de sortie, et des spécifications liées à la sécurité et aux performances. Par exemple, les signaux délivrés par les actionneurs possèdent une amplitude bornée et une vitesse de variation limitée. Grâce à sa formulation temporelle, la commande prédictive permet de prendre en compte explicitement les contraintes dans la phase de synthèse. Cependant, si la résolution analytique du problème d'optimisation dans le cadre prédictif est immédiate en l'absence de contraintes, leur prise en compte génère une charge de calcul considérable, liée à l'utilisation de méthodes itératives. Cet inconvénient restreint de fait les possibilités d'application de la commande prédictive dans le cas de systèmes avec périodes d'échantillonnage faibles.

Le problème de la commande prédictive avec contraintes peut être formulé de la façon suivante :

Minimiser la fonction de coût donnée par :

$$j = \frac{1}{2}\Delta U^T H \Delta U + b^T \Delta U + f_0 , \tag{2.26}$$

avec $H = 2(G^T G + \lambda I)$, $\mathrm{b}^{\mathrm{T}} = 2(Y_l - W)^T G$, $f_0 = (Y_l - W)^T (Y_l - W)$, où $\Delta U, G, Y_l$, et W sont définies dans le paragraphe 2.5.1,

sujet aux contraintes suivantes :

$$\begin{array}{ll} u_{\min} \le u(t) \le u_{\max} & \forall t \ : \text{provient de la limite sur la variable de commande} \\ \delta u_{\min} \le u(t) - u(t-1) \le \delta u_{\max} & \forall t \ : \text{provient de la limite du taux de variation de la variable de commande} \\ y_{\min} \le y(t) \le y_{\max} & \forall t \ : \text{pour des raisons de sécurité} \end{array}$$

D'autres types de contraintes peuvent être considérées afin d'imposer un certain comportement à la réponse du processus, elles peuvent être exprimées de la même manière [19]:

- contraintes d'intervalle : dans certain cas, il est désirable que les variables contrôlées suivent une trajectoire avec une tolérance donnée. Ceci, peut être réalisé en forçant la sortie du système à appartenir à l'intervalle [trajectoire spécifiée – la tolérance trajectoire spécifiée + la tolérance]. C'est-à-dire avoir :

$y_{\min}(t) \le y(t) \le y_{\max}(t)$

- Contraintes de dépassement : dans un certain processus, et pour des raisons différentes, les dépassements sont indésirables. Ce type de contraintes est exprimé de la manière suivante :

$y(t+j) \le w(t)$ pour $j = N_{01}.....N_{02}$, où N_{01} et N_{02} définissent l'horizon sur lequel on peut avoir un dépassement.

- comportement monotone : certains systèmes de commande présentent un comportement oscillatoire, ce comportement est en général indésirable. Des contraintes additionnelles sur les variables contrôlées peuvent être envisagées afin d'avoir un comportement monotone de la réponse du système. Ces contraintes sont de la forme suivantes :

$y(t+j) \le y(t+j+1)$ si $y(t) \prec w(t)$
$y(t+j) \ge y(t+j+1)$ si $y(t) \succ w(t)$

- Comportement non minimum de phase : pour éviter ce type de comportement on peut considérer les contraintes suivantes :

$y(t+j) \ge y(t)$ si $y(t) \prec w(t)$
$y(t+j) \le y(t)$ si $y(t) \succ w(t)$

- Contraintes terminales : on peut trouver ce type de contraintes lorsqu'on considère la technique de commande CRHPC (Constrained Receding Horizon Predictive Control) [59]. Dans cette technique, on force les sorties prédites du système à suivre la trajectoire de référence sur un horizon supplémentaire après l'horizon de prédiction N_2. Ces contraintes peuvent être exprimées en terme de contraintes d'égalité de la manière suivante :

$y_m = [y(t+N_2+1),..., y(t+N_2+m)]^T$

Toutes les contraintes considérées ci-dessus peuvent être exprimées par :

$R\Delta U \le c$ et $A\Delta U = a$. Le problème de la commande prédictive généralisée avec contraintes consiste à minimiser la fonction de coût donnée par l'équation (2.26) sujet à un ensemble de contraintes linéaires ; c'est-à-dire à optimiser une fonction quadratique avec contraintes linéaires. Ceci est connu en tant que problème de programmation quadratique. Le problème de la commande prédictive avec contraintes peut donc, être résolu en utilisant les méthodes d'optimisation quadratique. Deux classes de méthodes semblent à donner les meilleurs résultats : la classe de méthodes désignée par « active set methods » et, plus récemment, la classe de méthodes d'optimisation par point intérieur (Interior Point methods). Les principales algorithmes de la programmation quadratique, utilisés dans le contexte de la commande prédictive avec contraintes, peuvent être consultés dans [19], [21].

2.8 Stabilité de la commande prédictive

Malgré le grand succès de la commande prédictive généralisée dans de nombreux domaines d'application, l'absence de preuves théoriques de stabilité analytique à priori de cette technique de commande est considéré souvent comme un handicap. En fait, la majorité de résultats de stabilité sont limitées au cas de l'horizon infini et il y a un manque d'une théorie claire permettant de trouver les valeurs des paramètres de réglage assurant la stabilité du système en boucle fermée. Un certain nombre de stratégies qui peuvent répondre, sous certaines conditions, à cette préoccupation en donnant des résultats en terme de stabilité, ont été proposées [59], [60], [18], [21].

Une manière pour assurer la stabilité de la boucle corrigée consiste à introduire des contraintes terminales, qui imposent comme condition que la sortie coïncide avec la consigne sur un horizon supplémentaire à la fin de l'horizon de prédiction. Pour prouver la stabilité de cette technique, connue sous le nom CRHPC [59], il suffit d'utiliser la théorie de stabilité de Lyapunov [21]. Une deuxième manière consiste à utiliser des horizons infinis. En fait, il était connu, depuis un certain temps, que la stabilité de la commande prédictive à horizon infini est garantie [18], mais on ne connaissait pas comment traiter les contraintes lorsque l'horizon est infini. Ce problème a trouvé une solution lorsque Rawlings et Muske [60] ont montré que la commande prédictive avec contraintes et à horizon infini peut être approchée par la commande prédictive à horizon fuyant fini et sous contraintes. Il a été aussi montré, dans [18], que la stabilité de la commande prédictive à horizon fini peut, parfois, être garantie même s'il n' y a pas des contraintes terminales. Le problème de la commande prédictive à horizon fini peut être associé à une équation aux différences de Riccati variant dans le temps et dépendante de la valeur optimale de la fonction de coût. Dans la technique "Fake algebraic Riccati equation" [21], on remplace l'équation précédente par une équation algébrique de Riccati (invariant dans le temps) ressemblant à celles rencontrées dans le cas de la commande quadratique linéaire à horizon infini. Si cette équation possède les mêmes propriétés que celles de l'équation algébrique de Riccati réelle correspondant à la commande à horizon infini, alors la stabilité peut être déduite. Enfin, une autre possibilité est basée sur la paramétrisation de Youla. Ainsi, par exemple, les auteurs dans [61] et [62] utilise la paramétrisation de Youla comme outil de robustification de la commande prédictive, en garantissant en premier lieu un certain module pour les pôles de la boucle fermée, en assurant, ensuite, la robustesse en stabilité face à des incertitudes non structurées. Nous tenons à noter que les quatre approches citées ci-dessus sont considérées d'une manière détaillée dans la référence [21].

2.9 Exemple

Dans le souci de faire une comparaison entre la commande prédictive généralisée et les méthodes non linéaires qui vont être proposées dans le chapitre 4, nous considérons ici, l'application de l'algorithme de la commande prédictive généralisé à modèle linéaire sur le système non linéaire décrit par l'équation ci-dessous.

$$\frac{d^2y(t)}{dt^2}+\frac{dy(t)}{dt}+y(t)+y^3(t)=2u(t)-\frac{du(t)}{dt} \tag{2.27}$$

Ce système, qui est à phase non minimal, est souvent considéré dans la théorie des systèmes non linéaires. L'équation aux différences du système linéarisé autour de zéro est donnée par :

$$y(n)-1.895y(n-1)+0.9048y(n-2)=-0.0853u(n-1)+0.1044u(n-2) \tag{2.28}$$

où la période d'échantillonnage $T_e = 0.1\ s$.

Les résultats obtenus pour le jeu de paramètres suivants : $N_1 = 1,\ N_2 = 17,\ N_u = 1$ et $\lambda = 10^{-4}$, sont représentés par les figures (2.4) et (2.5). Nous remarquons que l'erreur de poursuite de la trajectoire de référence augmente avec l'augmentation de l'amplitude de la référence.

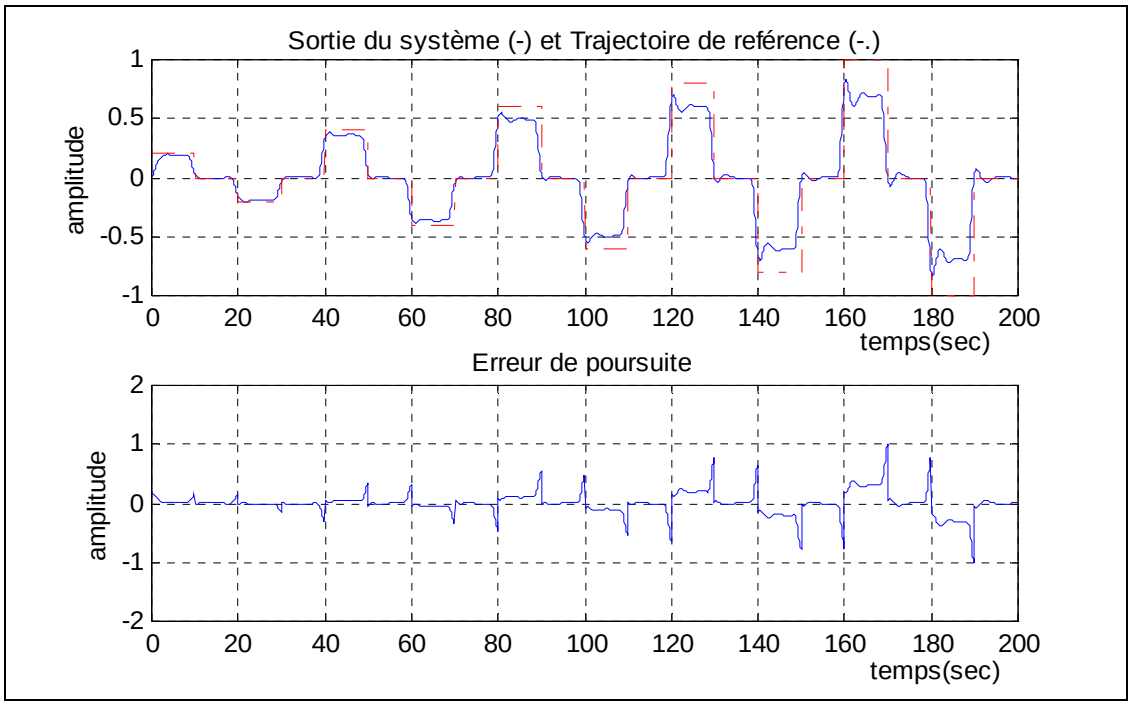


Figure 2.4 : Performances du régulateur prédictif obtenu

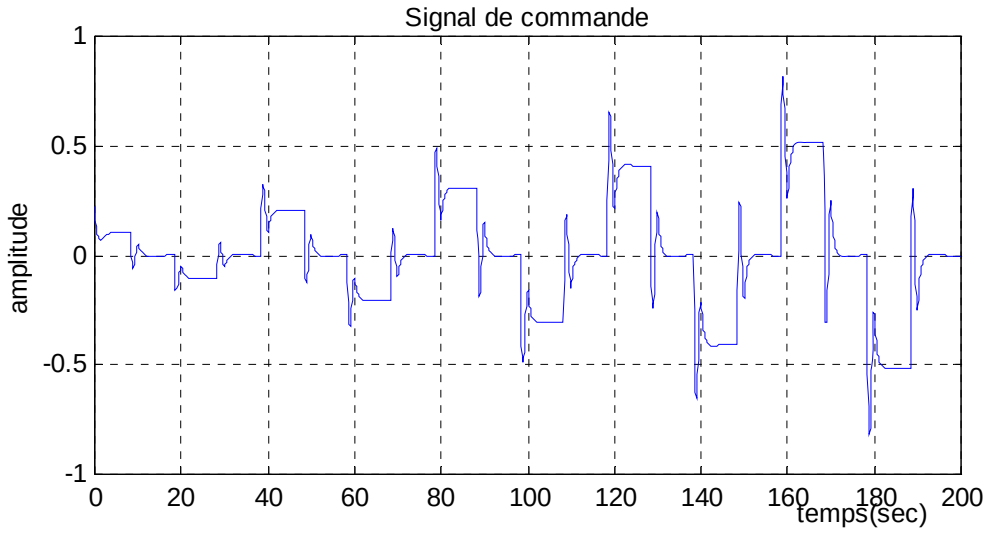


Figure 2.5 : signal de commande

2.10 Conclusion

Nous avons présenté, dans ce chapitre, la stratégie de la commande prédictive généralisée en utilisant la représentation entrée/sortie, bien qu'il soit aussi possible d'utiliser la représentation d'état [21]. La méthodologie prédictive peut se résumer par les deux points fondamentaux suivants :

- Elaboration d'un modèle de prédiction : ce modèle peut être obtenu par discrétisation de la fonction de transfert continue ou par identification ;
- Minimisation analytique d'une fonction de coût quadratique.

On peut aussi envisager de décrire la commande prédictive généralisée en raisonnant avec des modèles non linéaires. Néanmoins l'utilisation de tels modèles s'heurte principalement à deux difficultés majeures:

- La construction des modèles non linéaires fiables est, souvent, une tâche difficile ;
- Le problème d'optimisation devient complexe.

Les réseaux de neurones sont connus pour leur propriété d'approximation universelle des fonctions non linéaires. L'utilisation de tels réseaux peut donc fournir une solution à la première difficulté.

CHAPITRE 3

IDENTIFICATION AVEC LES RESEAUX DE NEURONES

3 IDENTIFICATION AVEC LES RESEAUX DE NEURONES

3.1 Introduction

L'une des étapes fondamentales dans la conception d'une loi de commande prédictive est, l'élaboration d'un modèle de prédiction afin de prévoir le comportement futur du processus. Les performances de ce type de commande dépendent en grande partie de la qualité de tel modèle, il est donc impératif de chercher le modèle qui représente adéquatement le comportement dynamique du système. Vu la capacité d'approximation que possèdent les réseaux de neurones, ils constituent un excellent outil de modélisation des systèmes non linéaires.

Le problème de l'identification des systèmes non linéaires en utilisant les réseaux *MLP*, les réseaux *RBF* (Radial Basis Functions) ainsi que les réseaux récurrents, a été considéré dans [22]. L'objectif de ce chapitre est de présenter, les étapes principales de l'identification des systèmes non linéaires par les réseaux de neurones, et en particulier d'introduire une méthode d'identification de tels systèmes en présence des perturbations [23] [24].

3.2 Identification des systèmes

L'identification, est l'opération de détermination du modèle dynamique d'un système à partir des mesures entrées/sorties. Selon le niveau des connaissances à priori sur le système à identifier, le problème d'identification des systèmes peut être abordé de différentes manières :

- Si la procédure d'identification est exclusivement basée sur les mesures entrées/sortie du système (on ignore tout ou une grande partie des phénomènes physiques mis en jeu), on parle alors de représentation "boîte noire" ;
- Lorsque la procédure d'identification est basée sur certaines connaissances disponibles sur le système, la phrase modélisation "boîte grise " est utilisée.

Dans tous les cas, le problème d'identification peut être formulé de la manière suivante :

Etant donné un ensemble de mesures entrées/sorties :

$$Z(t) = \{y(1), u(1), ..., y(t), u(t)\} \tag{3.1}$$

Le problème est de trouver une application de la forme :

$$\hat{y}(t) = f(Z(t-1)) \tag{3.2}$$

où $\hat{y}(t)$ est un estimé de la sortie $y(t)$.

Si l'identification des systèmes linéaires a déjà été largement étudiée et semble ne plus poser de problèmes insurmontables pour la majorité des cas, il n'en est pas de même pour les systèmes non linéaires. L'objectif de l'identification neuronale des systèmes non linéaires est d'approcher la

fonction *f* dans la relation (3.2) en utilisant un réseau de neurones. En effet, un réseau de neurones peut être considérer comme étant un modèle paramétrique de la forme :

$$\hat{y}(t/\theta) = f(Z(t-1)/\theta) \tag{3.3}$$

où θ est le vecteur des paramètres à déterminer. Ces paramètres, dans ce cas, sont les poids synaptiques du réseau.

Dans la structure donnée par l'équation (3.2), l'ensemble de mesures entrées/sorties augmente d'une façon continue. Il est plus commode d'utiliser un vecteur $\varphi(t)$ des mesures de dimension fixe, le modèle devient alors :

$$\hat{y}(t/\theta) = f(\varphi(t)/\theta) \tag{3.4}$$

$\varphi(t)$ est appelé vecteur des régresseurs, il est donnée par :

$$\varphi(t) = [\, y(t-1),.....,y(t-n), u(t-1),....,u(t-m)] \tag{3.5}$$

En utilisant cette paramétrisation, la procédure d'identification peut se résumer par les étapes indiquées dans la figure (3.1).

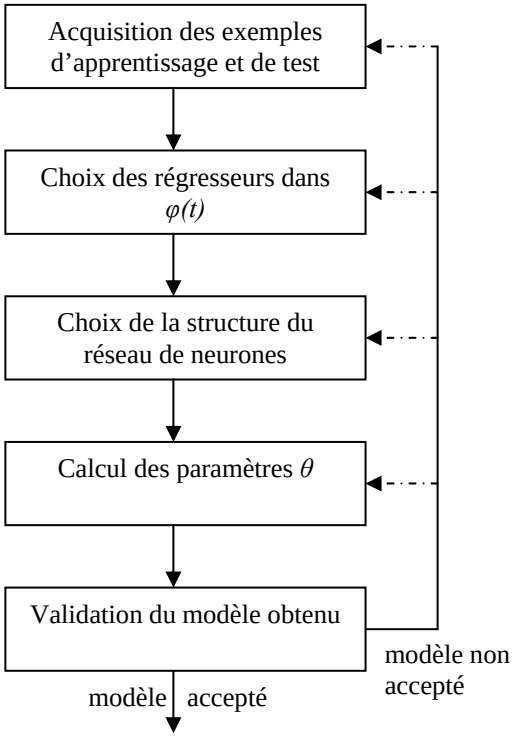


Figure 3.1 : les étapes principales de l'identification

La modélisation des systèmes non linéaires par réseaux de neurones a fait l'objet de nombreux travaux de recherche depuis une dizaine d'années à cause de la capacité d'apprentissage, d'approximation et de généralisation que possèdent ces réseaux [2], [22],[25]-[27], [29]. Cette nouvelle approche fournit une solution efficace à travers laquelle de larges classes des systèmes non linéaires peuvent être modélisés sans une description mathématique précise.

3.3 Acquisition des exemples d'apprentissage et de test

L'objectif de cette étape est d'acquérir les mesures entrées/sorties susceptibles de permettre l'extraction d'un modèle de procédé significatif. Pour cela, on doit appliquer une entrée capable de faire apparaître le maximum d'informations contenues dans le système. Le signal d'entrée qui permet de bien identifier un système non linéaire doit être riche en amplitude et en fréquences.

Dans cet étape on a souvent besoin de :

- choisir la fréquence d'échantillonnage ;
- concevoir le signal d'excitation ;
- appliquer un prétraitement sur les mesures obtenues.

Le prétraitement des données peut inclure : l'opération de filtrage pour éliminer d'éventuelles perturbations ou bruit, et dans le but d'accélérer le processus d'apprentissage des réseaux de neurones, l'opération de réduction des données à la valeur moyenne zéro et à la même valeur de la variance. Il est important de noter que le signal d'entrée choisi ne doit pas exciter les modes dynamiques du système à ne pas inclure dans le modèle. Quelques signaux susceptibles de satisfaire à ces exigences sont donnés dans [8]. La figure (3.2) en présente un exemple.

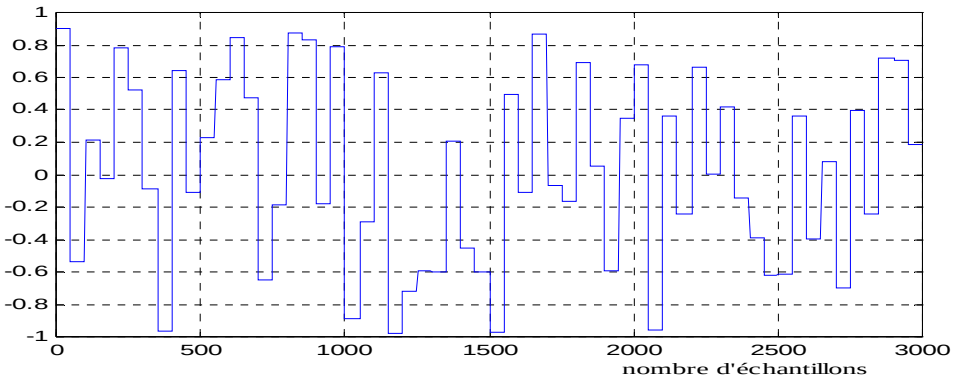


Figure 3.2 : exemple de signaux d'excitation (l'amplitude varie uniformément dans [-1 1])

3.4 Choix des régresseurs

Plusieurs techniques guidant le choix des régresseurs ont été proposées [8], [28]. Cependant, une méthode générale permettant la construction optimale de ces régresseurs n'est pas encore disponible. Les étapes principales de la procédure de choix des régresseurs sont données par l'organigramme de la figure (3.3).

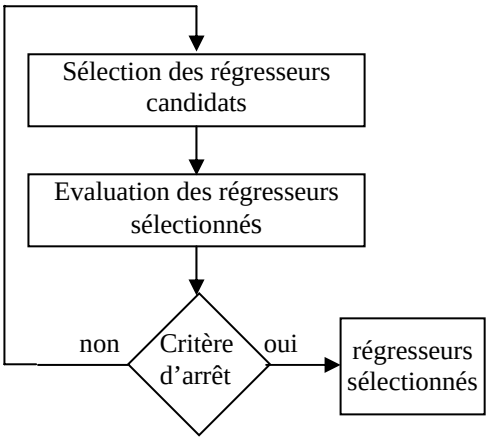


Figure (3.3) : procédure de choix des régresseurs

Il existe deux classes des méthodes de sélection des régresseurs candidats parmi l'ensemble de tout les régresseurs possibles : les méthodes heuristiques et les méthodes pseudo aléatoires. Dans les méthodes heuristiques on peut envisager deux approches différentes :

- dans la première approche, le modèle est construit en ajoutant à chaque fois une nouvelle entrée ;
- dans la deuxième approche, le modèle est construit en utilisant toutes les entrées possibles, ensuite les régresseurs dont leur impact sur les performances du modèle est faible, seront éliminés.

A cause de la taille réduite du modèle d'évaluation (en terme de nombre d'entrées), la première approche est préférable pour le cas des réseaux de neurones. Dans les méthodes pseudo aléatoires les régresseurs candidats à l'évaluation sont obtenus en utilisant les méthodes de la sélection génétique.

Pour évaluer la qualité des régresseurs sélectionnés on peut distinguer deux types des méthodes d'évaluation :

- les méthodes basé sur le modèle : c'est-à-dire de construire d'abord le modèle pour pouvoir ensuite évaluer la qualité des régresseurs ;
- les méthodes indépendantes du modèle : dans ce cas la qualité des régresseurs est évaluée en se basant seulement sur les mesures entrées/sorties.

Les méthodes de sélection et d'évaluation des régresseurs sont données d'une manière détaillée dans [8] et [28].

3.5 Choix de la structure du modèle

L'objet de cette étape est la sélection de la structure du modèle ainsi que l'architecture du réseau de neurones susceptibles de représenter la dynamique du système. Les réseaux multicouches statiques (*MLP*) sont les plus utilisés, à cause de la simplicité de leurs algorithmes d'apprentissage et leurs aptitudes à l'approximation et à la généralisation. L'inconvénient de l'utilisation de cette architecture est le choix du nombre de neurones sur les couches cachées ; il n'existe aucune méthode qui permette de choisir, d'une manière automatique, leur nombre en fonction de l'application envisagée.

Les structures des modèles utilisés pour l'identification des systèmes non linéaires sont une généralisation de ceux utilisés pour l'identification des systèmes linéaires. En effet, par analogie au cas linéaire, les modèles non linéaires suivants ont été proposés :

- Le modèle NFIR (Non linear Finite Impulse Response) : la régression est composée uniquement des entrées passées :

$$\hat{y}(t) = f(u(t-1),...,u(t-n)) \tag{3.6}$$

- Le modèle NARX (Non linear AutoRegressive with eXternal input) : dans ce cas la régression est composée des entrées et sorties passées :

$$\hat{y}(t) = f(u(t-1),...,u(t-n), y(t-1),..., y(t-m)) \tag{3.7}$$

- Le modèle NOE (Non linear Output Error) : la régression est composée des entrées et sorties estimées passées :

$$\hat{y}(t) = f(u(t-1),...,u(t-n), \hat{y}(t-1),..., \hat{y}(t-m)) \tag{3.8}$$

- Le modèle NARMAX (Non linear AutoRegressive, Moving Average with eXternal input) : la régression est composée des sorties et entrées passées ainsi que des erreurs d'estimation :

$$\hat{y}(t) = f(u(t-1),...,u(t-n), y(t-1),..., y(t-m), e(t-1),..., e(t-l)) \tag{3.9}$$

- Le modèle NSS (Non linear State Space models) :

$$\begin{aligned} \hat{x}(t+1) &= f(\hat{x}(t), u(t)) \\ \hat{y}(t) &= g(\hat{x}(t), u(t)) \end{aligned} \tag{3.10}$$

Plusieurs models sont, donc, possibles, le choix d'un modèle précis dépend de l'application, des informations disponibles et de la complexité du modèle. Le modèle NARX, appelé aussi modèle série-parallèle (figure 3.4), est le plus utilisé ; vu sa simplicité et sa structure non récursive (avantage de stabilité et simplicité des algorithmes d'apprentissage). On peut montrer que, sous certaines conditions d'observabilité généralisée, ce modèle peut représenter une large classe de systèmes non linéaires discrets [29]. Le modèle NARMAX et le modèle NOE (appelé aussi modèle parallèle) possèdent une structure récurrente. En plus du problème de stabilité qui, dans ce cas, n'est pas garantie, toute forme de récurrence dans les réseaux de neurones rend l'apprentissage plus complexe. Ainsi, si l'on cherche à ajuster les poids du réseau par une méthode analogue à celle de la descente du gradient, il est toujours possible de le faire, mais avec des dérivées plus difficiles à calculer [30], [31].

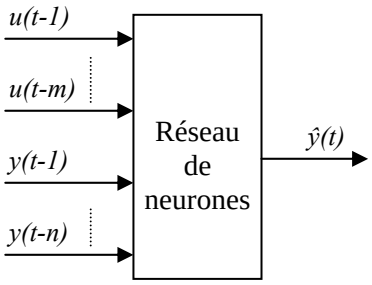


Figure 3.4 : le modèle série-parallèle (NARX)

3.6 Détermination des poids

Après avoir choisi la régression et le modèle, il faut estimer les paramètres de ce dernier. Ces paramètres sont les poids de connexions entres les neurones qui sont adaptés de telle sorte à minimiser un critère de performance. Cette procédure d'adaptation des poids est appelée, dans la littérature sur les réseaux de neurones, apprentissage. Le problème d'apprentissage peut être reformulé de la façon suivante :

Etant donné l'ensemble Z^N de N mesures entrées/sorties :

$$Z^N = \{[y(t), u(t)], T = 1,...., N\} \tag{3.11}$$

et les modèles candidats donnés par l'équation (3.4), le but de l'apprentissage est de déterminer une application de l'ensemble des données Z^N dans l'ensemble des modèles candidats :

$$Z^N \rightarrow \hat{\theta}$$

de telle sorte que la sortie du modèle obtenu approche, avec une précision satisfaisante, la sortie réelle du processus. Le critère de précision le plus utilisé est donné par :

$$V_N(\theta, Z^N) = \frac{1}{2N}\sum_{t=1}^{N}(y(t) - \hat{y}(t/\theta))^2 \tag{3.12}$$

Il s'agit donc, d'adapter les poids du réseau de neurones en minimisant la valeur quadratique moyenne et normalisée de l'erreur de prédiction. Les algorithmes d'adaptation des poids (apprentissage) pour les réseaux multicouches sont très nombreux et peuvent être présentés en deux classes. Dans la première, on trouve les algorithmes basés sur le calcul de la dérivée du critère (3.12), tels que la méthode de la rétropropagation et la méthode utilisant l'algorithme de Levenberg-Marquardt [5], [8], [10], [32]. Les méthodes d'apprentissage utilisant les algorithmes d'optimisation ne nécessitant pas le calcul de la dérivée, constituent la deuxième classe [10]. L'algorithme de Levenberg-Marquardt [8], même s'il ne fournit aucune garantie d'atteindre le minimum global, est cependant largement recommandé.

Il est évident que le choix de la méthode d'apprentissage dépend de la nature en "ligne" ou "hors ligne" de l'identification. Rappelons que le fonctionnement en ligne et en boucle fermée de l'algorithme d'identification pose des problèmes particuliers à cause des facteurs suivants :

- les données d'apprentissage ne sont disponibles que graduellement ;
- on ne maîtrise généralement pas la distribution des données ; celle-ci résulte du régulateur, de la dynamique du système et des objectifs de commande ;
- les conditions d'identifiabilité ne sont pas toujours évidentes à remplir, en particulier la conditions d'excitation persistante (une erreur de prédiction tendant vers zéro ne garantit pas que les paramètres ont atteint leur valeur optimale).

Afin d'éviter le problème de sur-apprentissage (over training) il est souvent souhaitable d'utiliser plus d'un critère pour arrêter l'apprentissage. Un certain nombre de critères d'arrêt, fréquemment utilisés, peuvent être trouvés dans la référence [8].

3.7 Validation du modèle

L'objectif de cette étape est de déterminer si le modèle obtenu représente adéquatement le système non linéaire considéré. Pour cela, on peut distinguer plusieurs critères de validation qui sont exprimés uniquement dans le domaine temporel, contrairement aux systèmes linéaires. La plupart des méthodes de validation utilisent un ensemble de données qui n'ont pas été utilisées dans la phase d'apprentissage, ces données sont appelées données de validation ou de test. Le critère typique le plus utilisé est celui de l'erreur de prédiction. La valeur de cette erreur doit être au dessous d'un certain seuil :

$$\varepsilon(t) = y(t) - \hat{y}(t/\theta) \tag{3.13}$$

Un autre test important est celui de la corrélation de l'erreur de prédiction (residual validation test). Si cette erreur ne contient pas d'information sur les erreurs passées ainsi que sur la dynamique du système, alors on peut dire que le modèle obtenu approxime bien le système. Pour réaliser ce test, en principe, on doit tester si l'erreur de prédiction est non corrélée avec toutes les combinaisons linéaire et non linéaire des données passées. Il est évident que ce test ne peut être réalisé pratiquement, il est donc, usuel de considérer seulement quelques fonctions d'autocorrélation et d'intercorrélation. On doit noter que cette méthode de validation a fait l'objet de plusieurs travaux [33], [34]. Billinges et al proposent, dans le cas des systèmes non linéaires, le test suivant [35] :

$$\begin{cases} \phi_{\varepsilon\varepsilon}(\tau) = \delta(\tau) \\ \phi_{\varepsilon^2 u^2}(\tau) = 0 \\ \phi_{\varepsilon u}(\tau) = 0 \\ \phi_{\varepsilon u^2}(\tau) = 0 \\ \phi_{\varepsilon\varepsilon u}(\tau) = 0 \qquad \forall \tau \end{cases} \tag{3.14}$$

où $\phi(\tau)$ est la fonction de corrélation normalisée.

3.8 Identification en présence des perturbations

Les mesures entrées/sorties, utilisées pour faire l'apprentissage au réseau de neurones, sont souvent entachées de bruit de mesure, ou affectées par des perturbations non mesurées (figure 3.5). Dans la théorie de l'identification, on suppose que les résultats des mesures peuvent être valablement modélisés par la somme d'une fonction inconnue, dite fonction de régression, et d'un terme représentant l'effet de ces perturbations sur la sortie du système (équation (3.15)). Nous pouvons distinguer deux cas différents :

$$y(t/\theta) = f(\varphi(t)/\theta) + v(t) \tag{3.15}$$

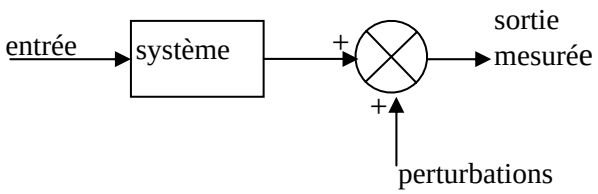


Figure 3.5 : acquisition des mesures entrées/sorties en présence des perturbations

3.8.1 Cas linéaire

Nous supposons que la perturbation peut être représentée par le modèle linéaire suivant :

$$v(t+1) = \sum_{i=0}^{p-1} \alpha_i v(t-i) \tag{3.16}$$

Le système à identifier est alors, décrit par le modèle suivant :

$$\begin{aligned} &y(t+1) = f(U(t), Y(t)) + v(t) \\ &U(t) = [u(t), \ldots, u(t-n+1)] \\ &Y(t) = [y(t), \ldots, y(t-m+1)] \end{aligned} \tag{3.17}$$

A partir des équations (3.16) et (3.17) on peut écrire le terme de la perturbation *v(t)* de la façon suivante :

$$v(t) = \sum_{i=0}^{p-1} \alpha_i y(t-i) - \sum_{i=0}^{p-1} \alpha_i f(U(t-i-1), Y(t-i-1)) \tag{3.18}$$

En remplaçant dans l'équation (3.17) le terme *v(t)* par l'équation (3.18), on obtient :

$$\begin{aligned} y(t+1) = {} & f(U(t), Y(t)) + \\ & \sum_{i=0}^{p-1} \alpha_i y(t-i) - \sum_{i=0}^{p-1} \alpha_i f(U(t-i-1), Y(t-i-1)) \end{aligned} \tag{3.19}$$

Cette nouvelle expression de *y(t+1)* dépend des *(m+p)* valeurs passées de la sortie du système et des *(n+p)* valeurs passées de l'entrée de commande. Il est, donc, possible de l'écrire de la manière suivante :

$$\begin{aligned} &y(t+1) = g(\overline{U}(t), \overline{Y}(t)) \\ &\overline{U}(t) = \left[u(t),...,u(t-n-p+1)\right], \\ &\overline{Y}(t) = \left[y(t),..., y(t-m-p+1)\right] \end{aligned} \tag{3.20}$$

où *g* est une fonction non linéaire qui sera approchée par un réseau de neurones.

Nous avons donc éliminé le terme de la perturbation du modèle d'identification et augmenté la taille du vecteur des régresseurs de la manière indiquée par l'équation (3.20).

3.8.2 Cas non linéaire

Dans le deuxième cas, nous admettons que les différentes perturbations peuvent être modélisées par un modèle non linéaire de la forme générale suivante :

$$\begin{aligned} &v(t+1) = h(V(t)) \\ &V(t) = \left[v(t), v(t-1),..., v(t-p+1)\right], \end{aligned} \tag{3.21}$$

où *h* est une fonction non linéaire.
A partir des équations (3.17) et (3.21), on peut écrire *v(t)* sous la forme :

$$v(t) = y(t+1) - f(U(t), Y(t)) \tag{3.22}$$

De cette dernière équation et de l'équation (3.21), on peut écrire :

$$\begin{aligned} &v(t+1) = h(W(t)) \\ &W(t) = \left[y(t+1) - f(U(t), Y(t)),.....,y(t-p+2) - f(U(t-p+1), Y(t-p+1))\right] \end{aligned} \tag{3.23}$$

En remplaçant dans l'équation (3.17) le terme *v(t)* par l'équation (3.23), on obtient :

$$y(t+1) = f(U(t), Y(t)) + h(W(t-1)) \tag{3.24}$$

Il est clair que cette expression de *y(t+1)* dépend des *(m+p)* et *(n+p)* valeurs passées de la sortie du système et l'entrée de commande, respectivement. Elle peut se mettre sous la forme suivante :

$$\begin{aligned} &y(t+1) = g(\overline{U}(t), \overline{Y}(t)) \\ &\overline{U}(t) = \left[u(t),...,u(t-n+p+1)\right], \\ &\overline{Y}(t) = \left[y(t),..., y(t-m+p+1)\right] \end{aligned} \tag{3.25}$$

où *g* est une fonction non linéaire qui sera approchée par un réseau de neurones.

Donc, nous avons démontré qu'il est possible, quelque soit le modèle des perturbations (linéaire ou non linéaire), de trouver un modèle pour le système à identifier en tenant compte de la présence de ces perturbations.

3.8.3 Simulation

Exemple 1 :

On considère l'identification du système non linéaire décrit par l'équation aux différences définie comme suit :

$$y(t+1) = \frac{y(t)y(t-1)y(t-2)u(t-1)(y(t-2)-1)+u(t)}{1+y(t-1)^2+y(t-2)^2} + v(t) \tag{3.26}$$

Dans tous les cas suivants, nous avons utilisé un réseau de neurones statique à une seule couche cachée (figure 1.3), l'algorithme d'apprentissage basé sur la méthode d'optimisation de Levenberg-Marquardt et un ensemble de 3000 paires de mesures entrée/sortie. Nous en utiliserons 2000 pour faire 500 itérations d'apprentissage au réseau et 1000 pour faire le test de validation. Les données d'apprentissage et de test ont été générées en utilisant l'entrée de commande donnée par la figure (3.2), la réponse correspondante du système est représentée sur la figure (3.6).

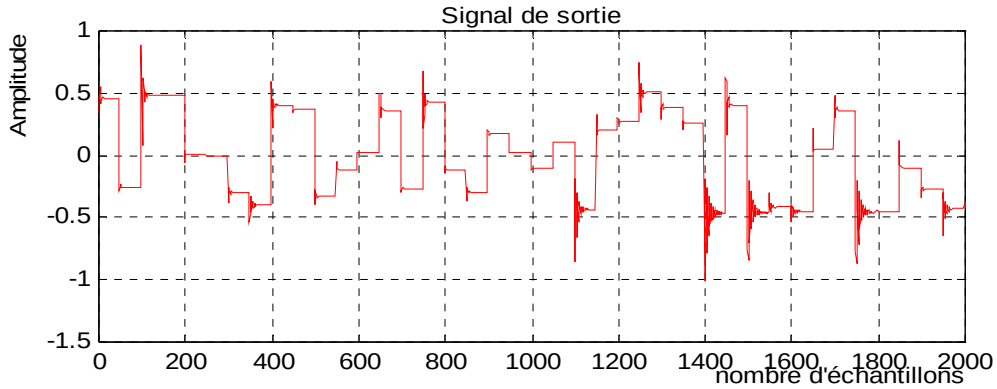


Figure 3.6 : la réponse du système à l'entrée d'excitation

1er Cas :

Nous supposons ici que le système à identifier n'est soumis à aucune perturbation (*v(t)=0*). Le modèle d'identification utilisé est donné par l'équation suivante :

Modèle I : $$\hat{y}(t+1) = NN(u(t), u(t-1), y(t), y(t-1), y(t-2)), \quad (3.27)$$

où *NN* compote 10 neurones sur sa couche cachée.

La réponse du modèle obtenu à l'entrée de test et la sortie correspondante du système sont représentées sur la figure (3.7). Les deux courbes sont superposées et l'erreur de prédiction est très faible.

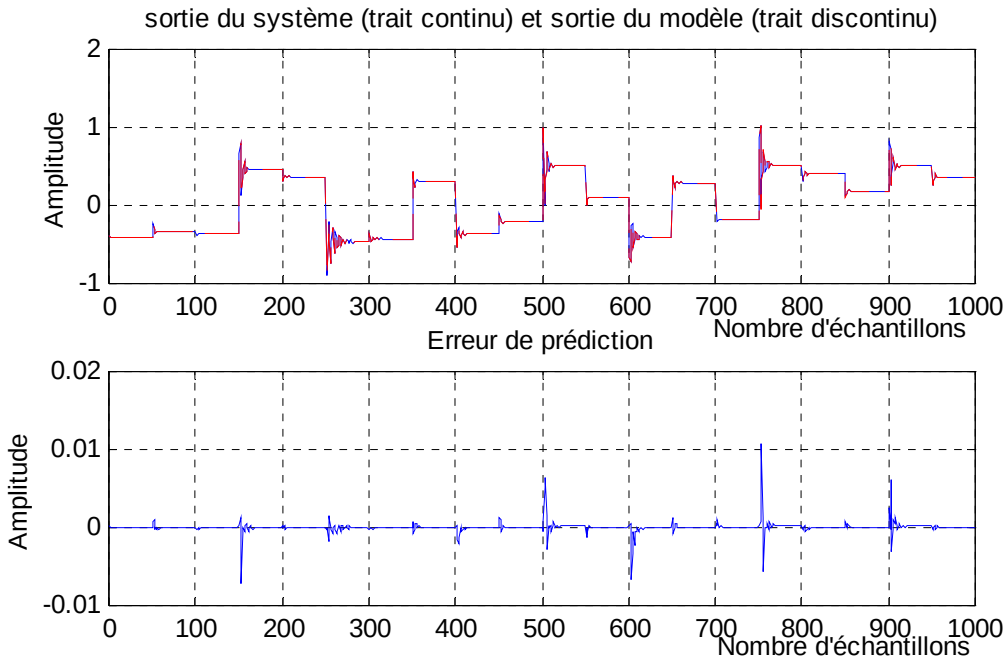


Figure 3.7 : Validation du modèle obtenu pour *v(t)=0*

2ème Cas :

La perturbation est, dans ce cas, non nulle et elle est donnée par le modèle linéaire suivant :

$$v_1(t+1) = cos(\beta)v_1(t) + sin(\beta)v_2(t)$$
$$v_2(t+1) = -sin(\beta)v_1(t) + cos(\beta)v_2(t)$$
$$v(t) = 0.8v_1(t) \qquad \beta = \pi / 6$$

Nous avons utilisé le même modèle que dans le premier cas (*modèle I*), et nous avons augmenté le nombre de neurones dans la couche cachée jusqu'à 20 neurones. L'erreur de prédiction (figure 3.8) est beaucoup plus importante que celle obtenue dans le premier cas. Ceci est dû à l'effet de la perturbation *v(t)* qui n'a pas été pris en compte dans le modèle d'identification utilisé.

3ème Cas :

Afin de tenir en compte de la présence de cette perturbation, nous avons utilisé le modèle donné par l'équation suivante :

Modèle II :
$$\hat{y}(t+1) = NN(u(t), u(t-1), u(t-2), u(t-3), y(t), y(t-1), y(t-2), y(t-3), y(t-4)), \tag{3.28}$$

où *NN* est le même réseau de neurones utilisé dans le cas précédent (20 neurones dans la couche cachée). L'erreur de prédiction, cette fois-ci, est moins importante que celle obtenue dans le 2éme cas (figure 3.8).C'est-à-dire que l'effet de la perturbation a été rejeté par le modèle utilisé.

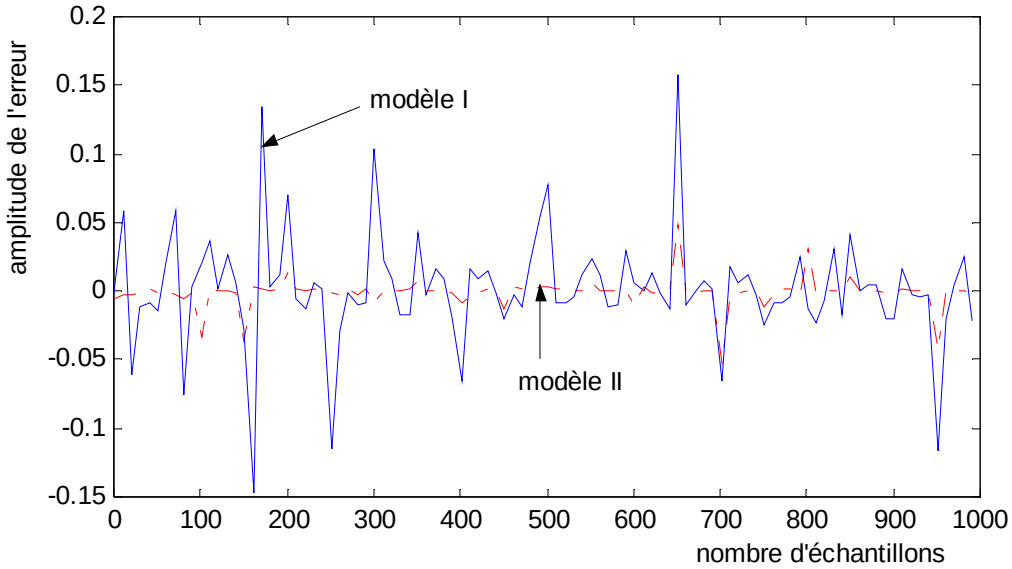


Figure 3.8 : Erreurs de prédiction obtenues (dans le 2ème et 3ème cas)

La figure 3.9 illustre la variation de la valeur du critère d'apprentissage dans les trois cas précédents.

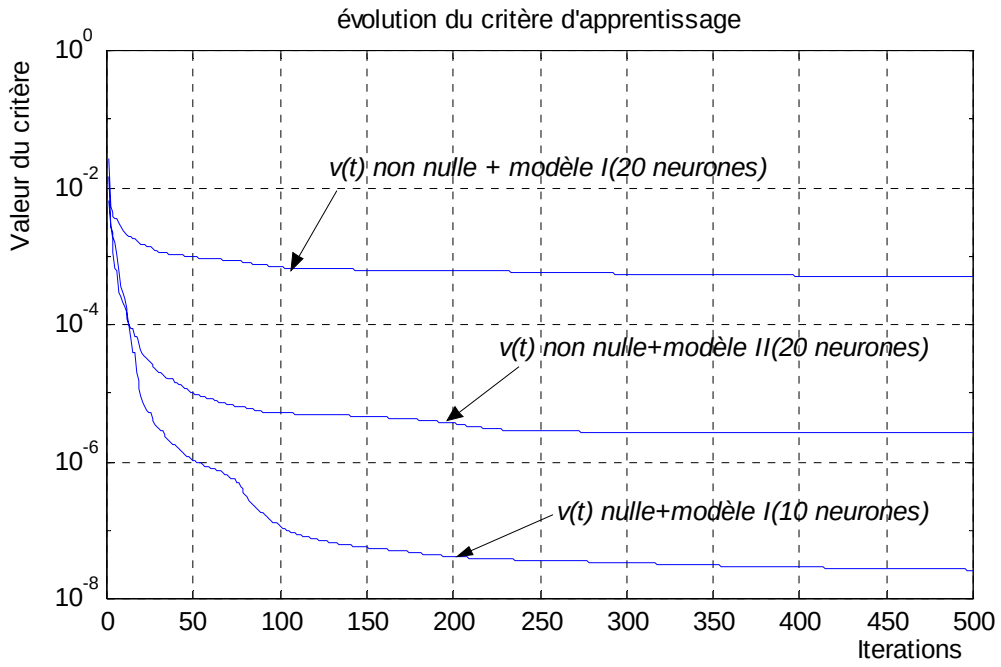


Figure 3.9 : Evolution de la valeur du critère d'apprentissage

Une autre mesure de la qualité du modèle d'identification est celle qui consiste à utiliser la valeur quadratique moyenne normalisée de l'erreur de prédiction, elle est définie par :

$$NMSE = \frac{1}{\sigma^2 N}\sum_{t=1}^{N}(\hat{y}(t)-y(t))^2 \tag{3.29}$$

où σ^2 est la variance de $y(t)$ et N est le nombre d'échantillons des données de test.

Pour faire une comparaison entre les deux modèles, nous avons représenté, sur la même figure, la variation de la valeur de *NMSE* en fonction du nombre d'échantillons des données de test (figure 3.10). Il est claire que le *modèle II* représente mieux le système considéré.

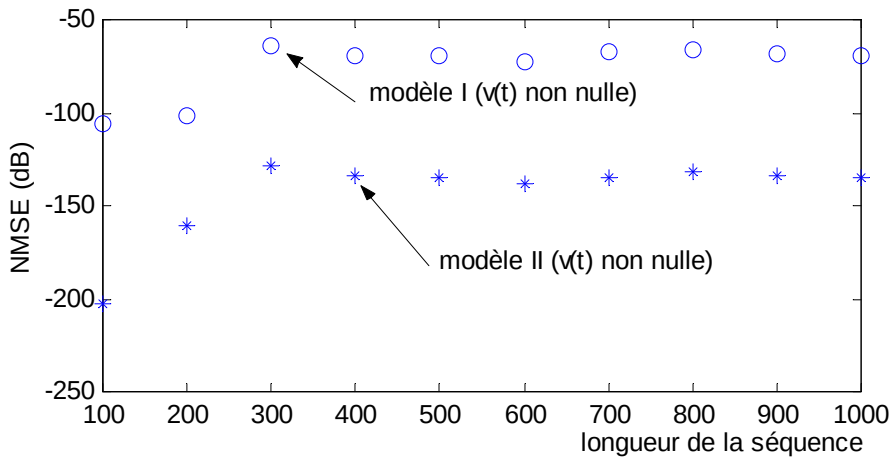


Figure 3.10 : Valeurs de *NMSE* en fonction du nombre d'échantillons des données de test

Exemple 2 :

Dans cet exemple nous considérons l'identification du même système donné par l'équation (3.26), où la perturbation *v(t)* est cette fois-ci représentée par le modèle non linéaire suivant :

$$v_1(t+1) = v_1(t) + 0.2v_2(t)$$
$$v_2(t+1) = -0.2v_1(t) + v_2(t) - 0.1(v_1(t)^2 - 1)v_2(t)$$
$$v(t) = 0.8v_1(t)$$

1er Cas :

Le modèle d'identification utilisé est celui donné par l'expression (3.27), où *NN* est un réseau de neurones possédant 25 neurones sur sa couche cachée. Les résultats obtenus sont représentés sur la figure 3.11. Nous remarquons que l'erreur de prédiction est importante.

2ème Cas :

Afin de prendre en considération l'effet de la perturbation, nous avons utilisé le modèle d'identification donné par l'équation (3.28), où *NN* est le même réseau de neurones utilisé dans le cas précédent.

L'erreur de prédiction, obtenue dans ce cas, est aussi représentée sur la figure 3.11. Elle est moins importante que celle obtenue dans le premier cas.

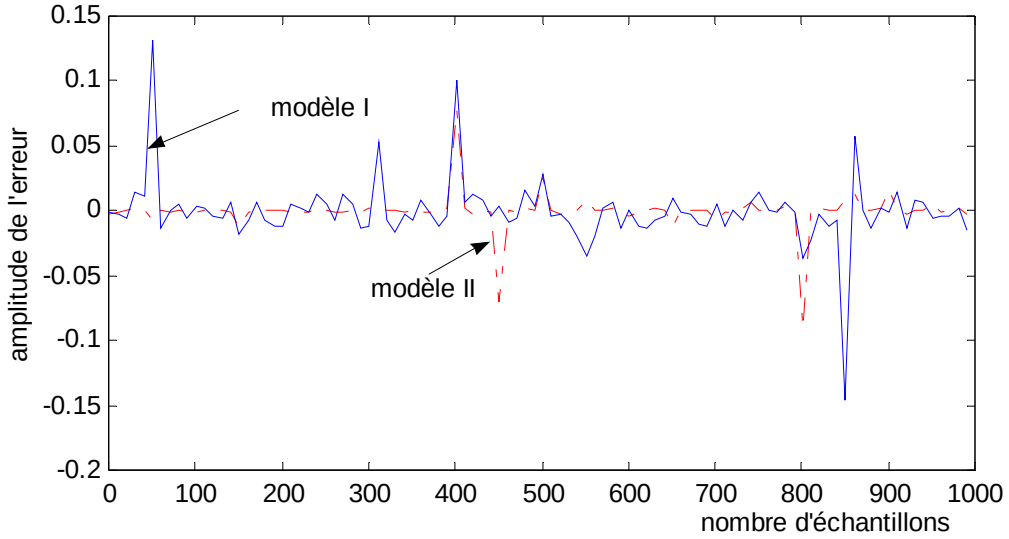


Figure 3.11 : Erreurs de prédiction obtenues (1er et 2ème cas)

Les figures 3.12 et 3.13, où nous avons représenté la variation de la valeur de *NMSE* donnée par l'équation (3.29) et celle de la valeur du critère d'apprentissage, montrent que le modèle obtenu dans le deuxième cas est plus précis que celui obtenu dans le premier cas.

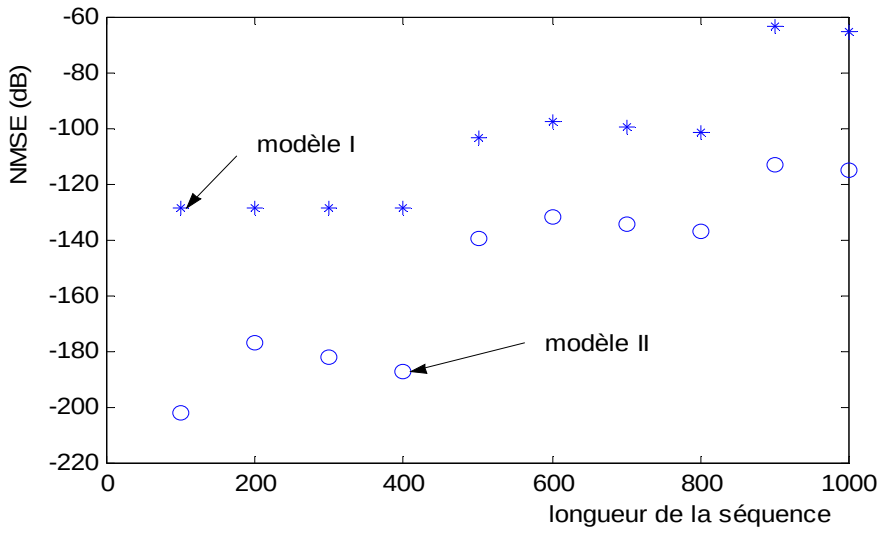


Figure 3.12 : Valeurs de *NMSE* en fonction du nombre d'échantillons des données de test

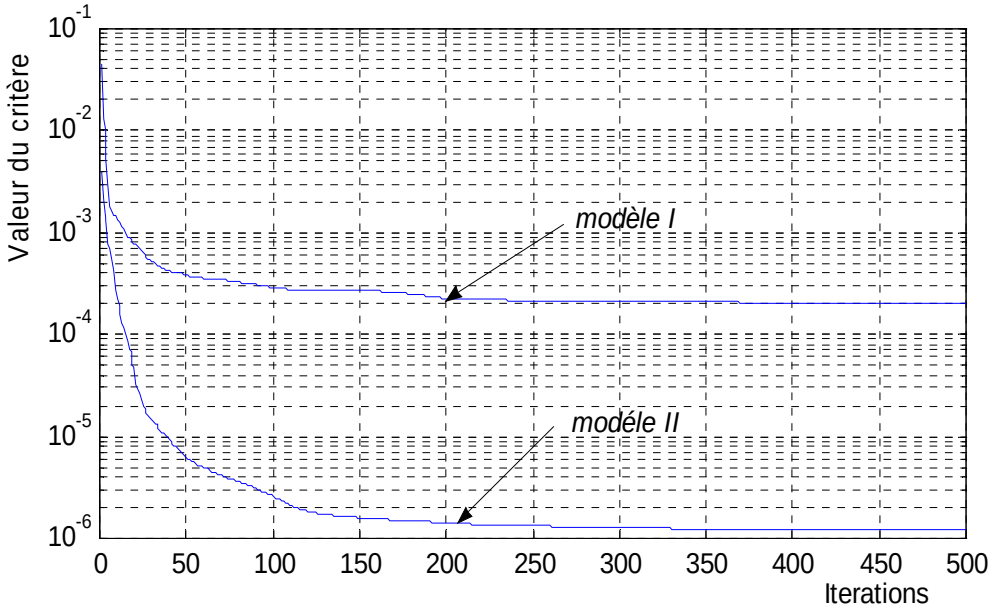


Figure 3.13 : Evolution de la valeur du critère d'apprentissage

3.9 Conclusion

Grâce au développement des méthodologies rigoureuses pour la conception de modèles, les réseaux de neurones sont devenus des outils de modélisation puissants dont les domaines d'applications sont multiples. Ils permettent de réaliser, de manière simple et efficace, des modèles précis, statique ou dynamique.

Notre travail met en évidence la possibilité de tenir en compte des perturbations, lorsqu'on utilise les réseaux de neurones pour la modélisation des systèmes non linéaires. A cet effet nous avons proposé une méthode permettant de construire un modèle du système de précision satisfaisante. Une formulation mathématique de la solution proposée a été donnée, et les deux cas de représentation des

perturbations, par un modèle linéaire et par un modèle non linéaire, ont été considérés. Cette solution consiste à donner plus d'informations au réseau de neurones sur l'histoire de la sortie et de l'entrée de commande du système ; ceci en augmentant la taille du vecteur des régresseurs de la manière expliquée dans le paragraphe 3.7.

CHAPITRE 4

Commande prédictive à modèle neuronal

4 Commande prédictive à modèle neuronal

4.1 Introduction

La commande prédictive par réseaux de neurones, que l'on peut rattacher à la famille de la commande par modèle, ne cesse de connaître ces dernières années un développement croissant. Plusieurs techniques, utilisant l'approche neuronale et permettant de mettre en œuvre ce type de commande ont été proposées. Néanmoins, pour des raisons liées à l'aspect non linéaire du problème d'optimisation, une méthode générale n'est pas encore disponible. En fait, toutes les méthodes proposées utilisent les algorithmes d'optimisation non linéaires [8], [36], [37]. De tels algorithmes sont trop consommateurs du temps de calcul ; ceci est très gênant pour les applications en temps réel. Afin d'alléger ce problème, d'autres méthodes basées sur la linéarisation du modèle de prédiction au voisinage d'un point de fonctionnement [38], [39], ou sur l'utilisation de modèles de formes particulières [40], ont été considérées.

Ce chapitre a pour but dans un premier temps de présenter les méthodologies proposées pour la conception d'une loi de commande prédictive à base d'un modèle neuronal, en particulier l'approche utilisant la méthode de Newton-Raphson [41], puis d'introduire une nouvelle méthode basée sur l'idée de décomposition de la réponse totale du système. Cette nouvelle méthode qui constitue notre contribution principale a fait l'objet des publications [42]- [44].

4.2 Formulation générale de la commande prédictive à modèle neuronal

Le principe de la commande prédictive à modèle neuronal, repose sur la même méthodologie utilisée pour la conception des régulateurs prédictifs linéaires. Les étapes essentielles de cette méthodologie ont été présentées d'une manière détaillée dans le chapitre 2. Le modèle de prédiction qui permet de prévoir le comportement futur du système est, dans ce cas, un réseau de neurones (figure 4.1).

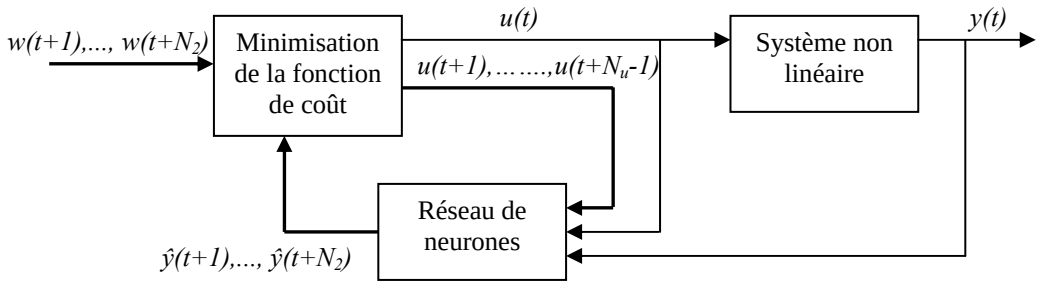


Figure 4.1 : Schéma de principe d'une commande prédictive à base d'un modèle neuronal

Si on admet que le système à contrôler peut être représenté par un modèle NARX :

$$\hat{y}(t) = f[\,y(t-1),....,\,y(t-n),u(t-d),...,u(t-m)] \tag{4.1}$$

où d est le temps mort du système et f est une fonction non linéaire réalisée par un réseau de neurones. Nous devons noter que, pour la réalisation de la fonction non linéaire f, nous pouvons utiliser un réseau de neurones de type *MLP*, *RBF* ou autres structures neuronales. Ainsi, la sortie prédite à l'instant $t+j$ connaissant la sortie du système à l'instant t peut être obtenue, d'une manière récursive, de la façon suivante :

$$\begin{aligned}\hat{y}(t+j/t) = f[\,&\hat{y}(t+j-1),...,\hat{y}(t+j-min(j,n)),y(t-1),...,y(t-max(n-j,0)),\\ &u(t-d+j),...,u(t-d-m+j)]\\ &j = N_1,...,N_2\end{aligned} \tag{4.2}$$

Il est aussi possible d'utiliser N_2-N_1+1 réseaux pour obtenir directement $\hat{y}(t+j/t)$ [36], [44]. Cependant cette dernière méthode, peut mener à des résultats non satisfaisants si l'horizon de prédiction est choisi grand. Car pour avoir une précision acceptable, il est nécessaire d'ajouter une entrée à chaque fois que le pas de prédiction augmente. C'est pour cette raison que la méthode récursive est préférée.

Une fois obtenue le modèle de prédiction, nous devons minimiser la fonction de coût donnée par l'équation (2.3). Ce problème d'optimisation est souvent, sujet à des contraintes sur les variables de commande :

$$\begin{aligned}&u_{min} \le u(t+j-1) \le u_{max} && j = 1,...,N_u\\ &\Delta u_{min} \le \Delta u(t+j-1) \le \Delta u_{max}\\ &\Delta u(t+j) = 0 \quad pour && j \ge N_u\\ &y_{min} \le y(t) \le y_{max}\end{aligned} \tag{4.3}$$

La solution de ce problème fournit la séquence de commandes futures dont seule la première sera effectivement appliquée au système.

Etant donnée que les sorties prédites sont non linéaires par rapport à l'entrée de commande, la minimisation de la fonction de coût, par rapport à l'entrée de commande, est alors un problème d'optimisation non linéaire et non convexe dont la solution est difficile et généralement coûteuse en temps de calcul. Cette solution devient beaucoup plus difficile si on tient en compte des contraintes. Il est important que le choix d'une méthode d'optimisation prenne en considération les points suivants :

- Une convergence rapide de l'algorithme est souvent, une nécessité absolue ;
- La solution doit être trouvée en temps réel ;
- La robustesse numérique de l'algorithme est un facteur primordial ;
- Une précision meilleure que celle du convertisseur numérique / analogique, à utiliser est inutile.

A partir, des points précédents plusieurs options permettent de distinguer une méthode particulière des autres. Deux approches peuvent être envisagées : la résolution numérique en utilisant les algorithmes d'optimisation non linéaire, ou la simplification du problème en considérant des modèles particuliers en vue d'obtenir une forme convexe de la fonction de coût ou de calculer analytiquement la loi de commande.

4.3 Approches utilisant la méthode de Newton

La minimisation directe de la fonction de coût nécessite l'utilisation d'une méthode itérative similaire à celle utilisée dans l'apprentissage des réseaux de neurones multicouches. En partant d'une valeur initiale, le vecteur des incréments de commande est ajusté selon la loi suivante :

$$\Delta U^{(i+1)} = \Delta U^{(i)} + \mu^{(i)} F^{(i)} \tag{4.4}$$

où μ est le pas d'adaptation et F est la direction de recherche.

La méthode de Newton correspond au cas où la direction de recherche est donnée par :

$$F^{(i)} = -H^{-1}(\Delta U^{(i)}) G(\Delta U^{(i)}) \tag{4.5}$$

G et H sont, respectivement, le gradient et la matrice Hessienne de la fonction de coût, ils sont donnés par :

$$G(\Delta U^{(i)}) = \left.\frac{\partial J}{\partial \Delta U}\right|_{\Delta U = \Delta U^{(i)}} \tag{4.6}$$

$$H(\Delta U^{(i)}) = \left.\frac{\partial^2 J}{\partial \Delta U^2}\right|_{\Delta U = \Delta U^{(i)}} \tag{4.7}$$

En plus qu'il n'y a aucune garantie pour que la matrice H soit définie positive, l'implantation directe de cette méthode nécessite le calcul de l'inverse de la matrice H, par conséquent le temps de calcul sera grand. Ceci peut ne pas permettre l'implantation en temps réel de la loi de commande. Ce sont les raisons, pour lesquelles on utilise les versions de la méthode de Newton modifiée.

Une autre approche utilise la méthode quasi Newton (quasi Newton method) qui consiste à définir une approximation pour l'inverse de la matrice Hessienne, puis à utiliser l'algorithme BFGS (Broyden-Fletcher-Goldfrab-Shanno) pour adapter cette approximation à chaque itération. La direction de recherche est, dans ce cas, donnée par :

$$F^{(i)} = -B^{(i)}(\Delta U^{(i)}) G(\Delta U^{(i)}) \tag{4.8}$$

avec B est une approximation de H^{-1}.

En vue, d'assurer la convergence de la méthode (quasi Newton method) le pas d'adaptation est déterminé par un algorithme d'optimisation à une seule variable (line search algorithm).

Une deuxième approche consiste à utiliser la méthode de Levenberg-Marquardt pour laquelle la direction de recherche est calculée en utilisant l'équation suivante :

$$(H(\Delta U^{(i)}) + \lambda^{(i)} I) F^{(i)} = -G(\Delta U^{(i)}) \tag{4.9}$$

où λ est le paramètre de Levenberg-Marquardt et I est une matrice identité.

A chaque itération de la méthode nous devons tester si la matrice $(H(\Delta U^{(i)}) + \lambda^{(i)} I)$ est définie positive. Sinon, nous devons augmenter la valeur de λ jusqu' à ce qu'elle le soit. L'algorithme de factorisation de Cholesky peut être utilisé pour déterminer si la matrice $(H(\Delta U^{(i)}) + \lambda^{(i)} I)$ est définie positive, ensuite pour calculer la direction de recherche à partir de l'équation (4.9).

Enfin, nous devons noter que, le calcul du gradient et de la matrice Hessienne de la fonction de coût dépend de la structure du réseau de neurones utilisé, et que les deux approches, en utilisant un réseau MLP à une seule couche cachée, sont données en détail dans la référence [8].

4.4 Linéarisation instantanée du modèle de prédiction

L'idée de cette approche consiste à linéariser, à chaque instant d'échantillonnage, le modèle neuronal [45]. En supposant que le modèle du système à commander est de la forme :

$$\hat{y}(t) = g(\varphi(t), \theta) \tag{4.10}$$

avec :

$$\varphi(t) = [\, y(t-1), \ldots\ldots, y(t-n), u(t-d), \ldots\ldots, u(t-d-m)]^T \tag{4.11}$$

où d est le temps mort du système.

La linéarisation du modèle (4.10) à l'instant $t = \tau$, autour de son état actuel $\varphi(\tau)$ est donnée par :

$$\tilde{y}(t) = -a_1 \tilde{y}(t-1) - \ldots - a_n \tilde{y}(t-n) + b_0 \tilde{u}(t-d) + \ldots + b_m \tilde{u}(t-m-d) \tag{4.12}$$

où :

$$a_i = -\left.\frac{\partial g(\varphi(t), \theta)}{\partial y(t-i)}\right|_{\varphi(t)=\varphi(\tau)} \qquad b_i = \left.\frac{\partial g(\varphi(t), \theta)}{\partial u(t-d-i)}\right|_{\varphi(t)=\varphi(\tau)} \tag{4.13}$$

$$\tilde{y}(t-i) = y(t-i) - y(\tau-i) \qquad \tilde{u}(t-i) = u(t-i) - u(\tau-i) \tag{4.14}$$

Il est possible de réécrire le modèle (4.12) de la manière suivante :

$$\hat{y}(t) = (1 - A(q^{-1}))y(t) + q^{-d} B(q^{-1}) u(t) + \zeta(\tau) \tag{4.15}$$

où le terme du bais $\zeta(\tau)$ est donnée par :

$$\zeta(\tau) = y(\tau) + a_1 y(\tau-1) + \ldots + a_n y(\tau-n) - b_0 u(\tau-d) - \ldots - b_m u(\tau-d-m) \tag{4.16}$$

et les polynômes $A(q^{-1})$ et $B(q^{-1})$ sont donnés par :

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_n q^{-n} \\ B(q^{-1}) &= b_0 + b_1 q^{-1} + \ldots + b_m q^{-m} \end{aligned} \tag{4.17}$$

Le modèle (4.15), qui est une approximation du modèle non linéaire (4.10), peut être considéré comme étant un modèle linéaire affecté par une perturbation constante $\zeta(\tau)$ dépendante du point de fonctionnement.

Dans le cas d'un réseau MLP à une seule couche cachée, une sortie linéaire et dont la fonction d'activation des neurones cachés est la fonction *tanh*, la dérivée de la sortie par rapport à une entrée $\varphi_i(t)$ s'obtient par :

$$\frac{\partial \hat{y}(t)}{\partial \varphi_i(t)} = \sum_{j=1}^{n_h} w_j w_{jk} \left(1 - tanh^2 \left(\sum_{k=1}^{n_e} w_{jk} \varphi_k(t) + w_{j0} \right) \right) \tag{4.18}$$

où :

$$\hat{y}(t) = \sum_{j=1}^{n_h} w_j \; tanh \left(\sum_{k=1}^{n_e} w_{jk} \varphi_k(t) + w_{j0} \right) + w_0 \tag{4.19}$$

n_e et n_h sont le nombre d'entrées et le nombre de neurones cachés, respectivement.

Une fois que nous avons obtenu les polynômes $A(q^{-1})$ et $B(q^{-1})$ la procédure d'élaboration de la loi de commande prédictive décrite dans le chapitre 2 peut être utilisée.

Une autre approche, qui a été proposé par Botto M. A. et co-auteurs [38], est basée sur la méthode de linéarisation par contre réaction. Le modèle de prédiction utilisée est de la forme :

$$\hat{y}(t) = f(\varphi(t), \theta_1) + g(\varphi(t), \theta_2) u(t-1) \tag{4.20}$$

où :

$$\varphi(t) = [y(t-1), \ldots, y(t-n), u(t-2), \ldots, u(t-m)]^T \tag{4.21}$$

f et g sont obtenues par deux réseaux de neurones. Il est clair que le modèle donné par l'équation (4.20) ne peut représenter qu'une classe des systèmes non linéaires.

Bien que l'idée de la linéarisation s'avère intéressante quant à la facilité d'implantation et au temps de calcul réduit, l'obtention des performances satisfaisantes lorsqu'on s'éloigne du point de fonctionnement, autour duquel la linéarisation a été faite, ou pour les systèmes non linéaires complexes n'est pas évident.

4.5 Utilisation de la méthode de Newton-Raphson

La mise en œuvre de la loi de commande prédictive en utilisant la méthode de Newton-Raphson (Newton-Raphson Neural Predictive Control : NRNPC), consiste à transformer le problème de minimisation de la fonction de coût à un problème de résolution d'un système d'équations linéaires. L'utilisation de cette méthode, connue pour sa convergence quadratique, permet d'éviter le calcul de l'inverse de la matrice Hessienne et par conséquent de réduire le temps de calcul par rapport aux approches utilisant l'algorithme de Newton. Notons que, dans leur travail [37], D. Soloway et al. ont utilisé la décomposition LU pour résoudre le système d'équations. Cependant, en utilisant cette décomposition des problèmes d'instabilité numérique (division par zéro) de l'algorithme peuvent être rencontrés, alors que la robustesse numérique de l'algorithme de commande est un facteur très important. Dans notre travail [41], l'algorithme de Jordan, pour lequel on peut se contenter d'une meilleure stabilité numérique, a été choisi pour la résolution du système d'équations linéaires.

4.5.1 Modèle de prédiction

Nous utilisons un réseau de neurones statique MLP à une seule couche cachée et une seule sortie linéaire (figure 1.3). La fonction d'activation des neurones de la couche cachée est la tangente hyperbolique. Le modèle neuronal du processus, utilisé pour la prédiction des sorties futures est le modèle NARX suivant :

$$\hat{y}(t) = NN\big(u(t-1), \ldots, u(t-m), y(t-1), \ldots, y(t-n)\big) \tag{4.22}$$

La sortie du réseau de neurones *NN*, en appliquant le vecteur d'entrée :

$$\varphi(t) = [u(t-1), \ldots, u(t-m), y(t-1), \ldots, y(t-n)]^T \tag{4.23}$$

est donnée par les expressions suivantes:

$$\hat{y}(t) = \sum_{j=1}^{n_c} W_j \phi_j\big(s_j(t)\big) + W_{n_c+1} \tag{4.24}$$

$$s_j(t) = \sum_{i=1}^{m} w_{ji} u(t-i) + \sum_{i=1}^{n} \left(w_{jm+i} y(t-i) \right) + w_{jn_e+1} \tag{4.25}$$

où :

n_c et $n_e = n+m$ sont les nombres de neurones dans la couche cachée et la couche d'entrée, respectivement. W_j sont les poids des connexions entre la couche cachée et la couche de sortie, w_{ji} sont les poids des connexions entre la couche d'entrée et la couche cachée et ϕ est la fonction d'activation des neurones cachés.

Les sorties prédites à l'instant $t+k$ sont alors, données par :

$$\hat{y}(t+k/t) = \sum_{j=1}^{n_c} W_j \phi\left(s_j(t+k)\right) + W_{n_c+1} \tag{4.26}$$

$$s_j(t+k) = \sum_{i=1}^{m} w_{ji} \begin{cases} u(t+k-i) & ,k-N_u \prec i \\ u(t+N_u) & ,k-N_u \geq i \end{cases} + \sum_{i=1}^{min(k-1,n)} \left(w_{jm+i} \hat{y}(t+k-i) \right) + \sum_{i=k}^{n} \left(w_{jm+i} y(t+k-i) \right) + w_{jn_c+1} \tag{4.27}$$

$$k = N_1, \ldots, N_2$$

4.5.2 Minimisation de la fonction de coût

L'objectif est de trouver le meilleur vecteur de commande $U(t) = [u(t), u(t+1), \ldots\ldots\ldots, u(t+N_u-1)]^T$ minimisant la fonction de coût J donnée par l'équation (2.3). Le minimum de J peut être déterminé d'une manière itérative selon la loi d'adaptation suivante :

$$U(t)^{(k+1)} = U(t)^{(k)} - \left[\frac{\partial^2 J^{(k)}}{\partial U^2(t)} \right]^{-1} \frac{\partial J^{(k)}}{\partial U(t)} \qquad k = 1, \ldots\ldots\ldots, n_{max} \tag{4.28}$$

où n_{max} est le nombre d'itérations maximal, $\frac{\partial J^{(k)}}{\partial U(t)}$ et $\frac{\partial^2 J^{(k)}}{\partial U^2(t)}$ sont le Jacobien et la matrice Hessienne de la fonction de coût à l'itération k, respectivement. Ils sont donnés par :

$$\frac{\partial J^{(k)}}{\partial U(t)} = \left[\frac{\partial J}{\partial u(t)}, \ldots\ldots\ldots\ldots\ldots\ldots, \frac{\partial J}{\partial u(t+N_u-1)} \right]^T \tag{4.29}$$

$$\frac{\partial^2 J^{(k)}}{\partial U^2(t)} = \begin{bmatrix} \frac{\partial^2 J}{\partial u^2(t)} & \ldots\ldots\ldots\ldots\ldots\ldots & \frac{\partial^2 J}{\partial u(t) \partial u(t+N_u-1)} \\ . & & \\ . & & \\ . & & \\ \frac{\partial^2 J}{\partial u(t+N_u-1) \partial u(t)} & \ldots\ldots & \frac{\partial^2 J}{\partial^2 u(t+N_u-1)^2} \end{bmatrix} \tag{4.30}$$

La solution directe de l'équation (4.28) nécessite le calcul de l'inverse de la matrice Hessienne, ceci conduit à un temps de calcul assez important. Afin, d'éviter le calcul de cette matrice inverse, nous mettons l'équation (4.28) sous forme d'un système d'équations linéaires de la forme :

$$\frac{\partial^2 J^{(k)}}{\partial U^2(t)} \Delta U^{(k+1)} = -\frac{\partial J^{(k)}}{\partial U(t)} \tag{4.31}$$

où $\Delta U(t)^{(k+1)} = U(t)^{(k+1)} - U(t)^{(k)}$ est le vecteur des incréments de commande.

La résolution du système d'équations (4.31), par l'algorithme de Jordan, fournit les incréments futurs de commande. Après avoir trouvé $\Delta U^{(k+1)}$, nous pouvons calculer $U^{(k+1)}$ par l'expression suivante:

$$U(t)^{(k+1)} = U(t)^{(k)} + \Delta U(t)^{(k+1)} \tag{4.32}$$

Cette procédure itérative est répétée jusqu'à ce que la précision fixée soit obtenue. Dans la majorité des cas, une itération de cet algorithme est suffisante.

Le calcul de chaque élément du Jacobien ainsi que celui de la matrice Hessienne sont nécessaires pour chaque itération de l'algorithme, ils sont donnés par :

$$\frac{\partial J}{\partial u(t+h)} = 2\sum_{j=N_1}^{N_2}\left[\hat{y}(t+j) - w(t+j)\right]\frac{\partial \hat{y}(t+j)}{\partial u(t+h)} + 2\lambda\sum_{j=1}^{N_u}\left[\Delta u(t+j-1)\right]\frac{\partial \Delta u(t+j-1)}{\partial u(t+h)} \tag{4.33}$$

$$h = 0,\ldots,N_u - 1$$

$$\frac{\partial^2 J}{\partial u(t+l)\partial u(t+h)} = 2\sum_{j=N_1}^{N_2}\left\{\frac{\partial \hat{y}(t+j)}{\partial u(t+l)}\frac{\partial \hat{y}(t+j)}{\partial u(t+h)} + \frac{\partial^2 \hat{y}(t+j)}{\partial u(t+l)\partial u(t+h)}\left[\hat{y}(t+j) - w(t+j)\right]\right\} +$$

$$2\lambda\sum_{j=1}^{N_u}\left\{\frac{\partial \Delta u(t+j-1)}{\partial u(t+l)}\frac{\partial \Delta u(t+j-1)}{\partial u(t+h)} + \Delta u(t+j-1)\frac{\partial^2 \Delta u(t+j-1)}{\partial u(t+l)\partial u(t+h)}\right\} \tag{4.34}$$

$$l = 0,\ldots\ldots,N_u - 1,\quad h = 1,\ldots\ldots N_u - 1$$

$$\frac{\partial \Delta u(t+j-1)}{\partial u(t+h)} = \frac{\partial u(t+j-1)}{\partial u(t+h)} - \frac{\partial u(t+j-2)}{\partial u(t+h)} = \delta(h, j-1) - \delta(h, j-2) \tag{4.35}$$

$$\frac{\partial \Delta u(t+j-1)}{\partial u(t+h)}\frac{\partial \Delta u(t+j-1)}{\partial u(t+l)} = \left(\delta(h, j-1) - \delta(h, j-2)\right)\left(\delta(l, j-1) - \delta(l, j-2)\right) \tag{4.36}$$

où la fonction delta est définie par :

$$\delta(h, j) = \begin{cases} 1 & si \quad h = j \\ 0 & si \quad h \neq j \end{cases}$$

Le terme $\dfrac{\partial^2 \Delta u(t+j-1)}{\partial u(t+l)\partial u(t+h)}$ est pour la plupart des cas nul.

Les termes $\frac{\partial \hat{y}(t+k)}{\partial u(t+h)}$ et $\frac{\partial \hat{y}^2(t+k)}{\partial u(t+h)\partial u(t+l)}$ intervenant dans le calcul du Jacobien et de la matrice Hessienne sont donnés par :

$$\frac{\partial \hat{y}(t+k)}{\partial u(t+h)} = \sum_{j=1}^{n_c} W_j \frac{\partial \phi(s_j(t+k))}{\partial u(t+h)} \tag{4-37}$$

En appliquant la règle de la dérivation composée, on obtient :

$$\frac{\partial \phi(s_j(t+k))}{\partial u(t+h)} = \frac{\partial \phi(s_j(t+k))}{\partial s_j(t+k)} \frac{\partial s_j(t+k)}{\partial u(t+h)} \tag{4.38}$$

$$\frac{\partial s_j(t+k)}{\partial u(t+h)} = \left[\sum_{i=1}^{m} w_{ji} \begin{cases} \delta(k-i,h) & , \quad k-N_u+1 < i \\ \delta(N_u-1,h) & , \quad k-N_u+1 \geq i \end{cases} \right] + \sum_{i=1}^{min(k-1,n)} w_{ji+m} \frac{\partial \hat{y}(t+k-i)}{\partial u(t+h)} \delta_1(k-i-1) \tag{4-39}$$

où δ_1 représente la fonction échelon unité.

$$\frac{\partial^2 \hat{y}(t+k)}{\partial u(t+h)\partial u(t+l)} = \sum_{j=1}^{n_c} W_j \frac{\partial^2 \phi(s_j(t+k))}{\partial u(t+h)\partial u(t+l)} \tag{4-40}$$

$$\frac{\partial^2 \phi(s_j(t+k))}{\partial u(t+h)\partial u(t+l)} = \frac{\partial \phi(s_j(t+k))}{\partial s_j(t+k)} \frac{\partial^2 (s_j(t+k))}{\partial u(t+h)\partial u(t+l)} + \frac{\partial^2 \varphi(s_j(t+k))}{\partial s_j(t+k)^2} \frac{\partial s_j(t+k)}{\partial u(t+k)} \frac{\partial s_j(t+k)}{\partial u(t+l)} \tag{4.41}$$

$$\frac{\partial^2 s_j(t+k)}{\partial u(t+h)\partial u(t+l)} = \sum_{i=1}^{min(k-1,n)} w_{ji+m} \frac{\partial^2 \hat{y}(t+k-i)}{\partial u(t+h)\partial u(t+l)} \delta_1(k-i-1) \tag{4-42}$$

Les étapes principales de la procédure de calcul de la commande sont résumées par les points suivants :

1- spécifier la trajectoire de référence et choisir les paramètres de réglage ;
2- fixer le nombre maximum n_{max} des itérations ;
3- initialiser le vecteur des commandes futures :
   $U(t)=U_0(t)$;
4- calculer les sorties futures : $\hat{y}(t+j), j=N_1,\ldots,N_2$;
5- calculer :
   $$B = -\frac{\partial J}{\partial U(t)} \quad \text{et} \quad A = \frac{\partial^2 J}{\partial^2 U(t)} \ ;$$
6- résoudre le système d'équations : $A\Delta U(t) = B$;
7- calculer le vecteur des commandes futures :
   $U(t) = U_0(t) + \Delta U(t)$, $\qquad U_0(t) = U(t)$;
8- si le nombre maximal d'itérations est atteint, alors la valeur de $u(t)$ est le premier élément du vecteur $U(t)$. Aller à l'étape 4 pour calculer une nouvelle valeur de $u(t)$;
9- sinon, aller à l'étape 4.

4.5.3 Applications

4.5.3.1 Commande d'un système non linéaire d'ordre 2

Nous considérons ici l'application de l'algorithme précédent à la commande prédictive du système non linéaire à phase non minimal donnée par l'équation (2.27).

En vue d'obtenir le modèle de prédiction, nous devons tout d'abord procéder à l'identification du système considéré. A cet effet, nous avons utilisé :

- le modèle NARX suivant :

$$\hat{y}(t) = f\big(y(t-1), y(t-2), u(t-1), u(t-2)\big),$$

- un réseau de neurones MLP possédant 5 neurones sur sa couche cachée et une sortie linéaire,
- Les données d'apprentissage et de test représentées sur la figure 4.2,
- la période d'échantillonnage $T_e = 0.1\ sec$.

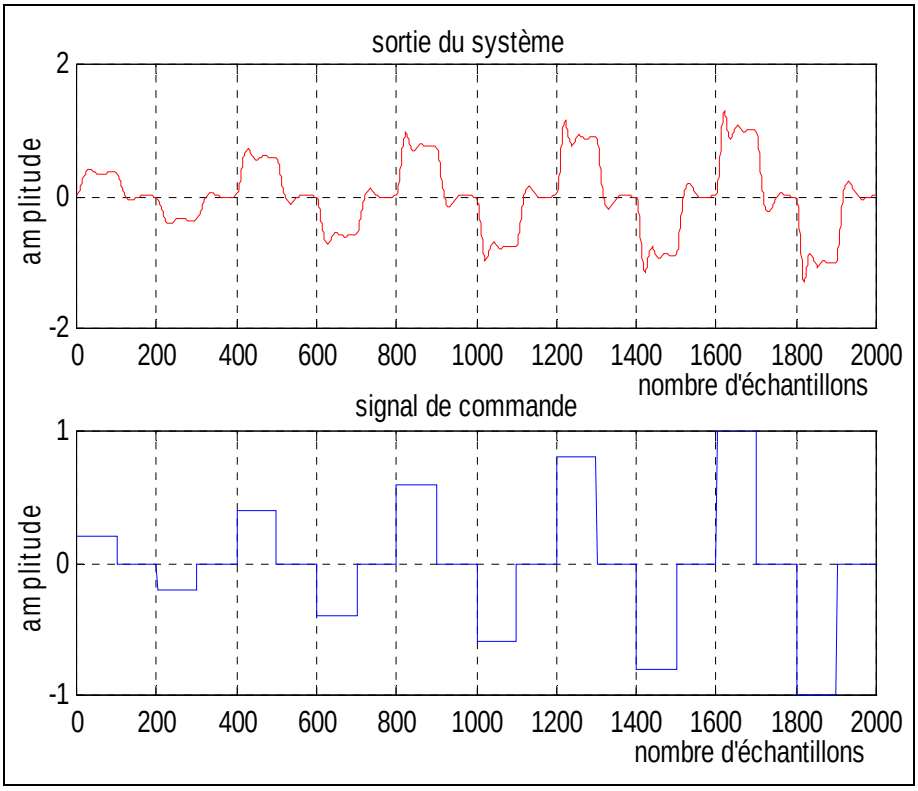


Figure 4.2 : Données d'apprentissage et de test

La réponse du modèle obtenu à l'entrée de test, celle du système à la même entrée et l'erreur de prédiction sont représentées sur la figure 4.3, les deux courbes sont superposées, et l'erreur de prédiction est très faible.

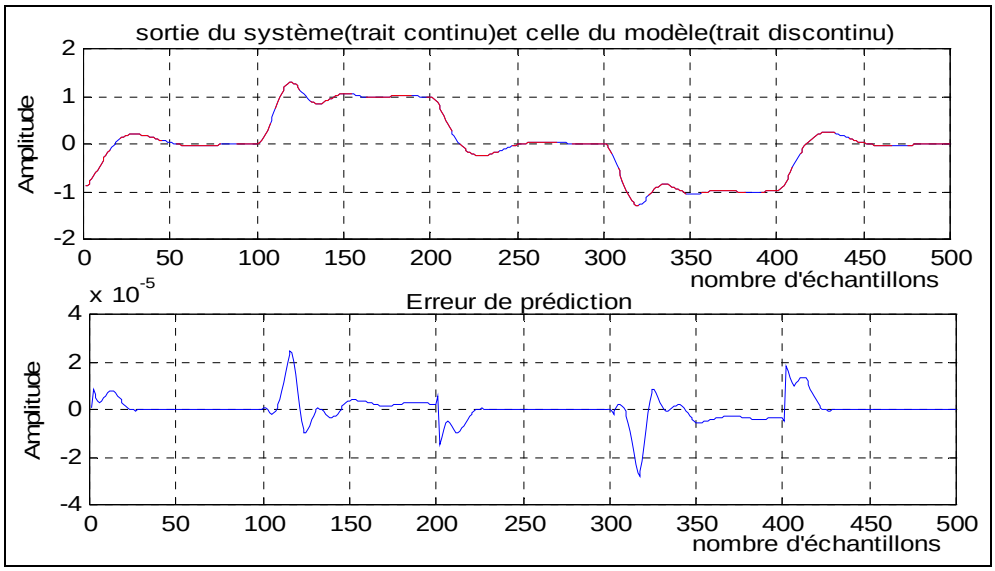


Figure 4.3 : validation du modèle obtenu

Après avoir obtenu le modèle de prédiction et choisi les valeurs des paramètres de réglage, l'algorithme de commande proposé est utilisé pour calculer le signal de commande minimisant la fonction de coût spécifiée. Les figures 4.4 et 4.5 présentent les performances obtenues en choisissant : $N_1=1$, $N_2=8$, $N_u=1$ et $\lambda=10^{-4}$. Nous obtenons une bonne poursuite de la trajectoire de référence, mais avec un effort de commande présentant des pics indésirables et des variations rapides.

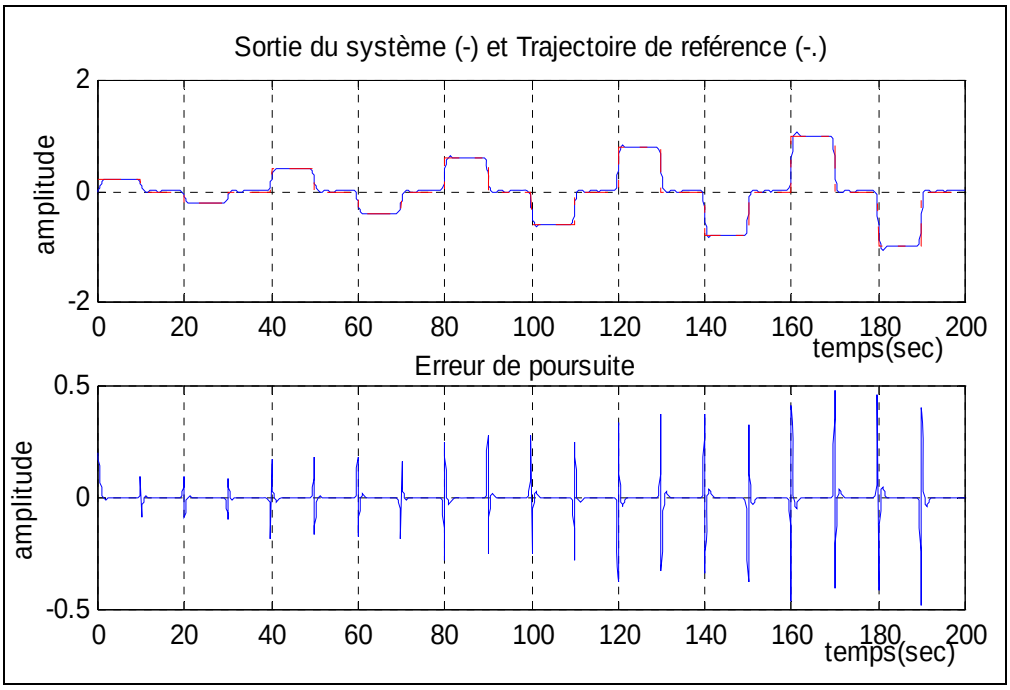


Figure 4.4 : Performances obtenues pour ($N_1=1$, $N_2=8$, $N_u=1$ et $\lambda=10^{-4}$)

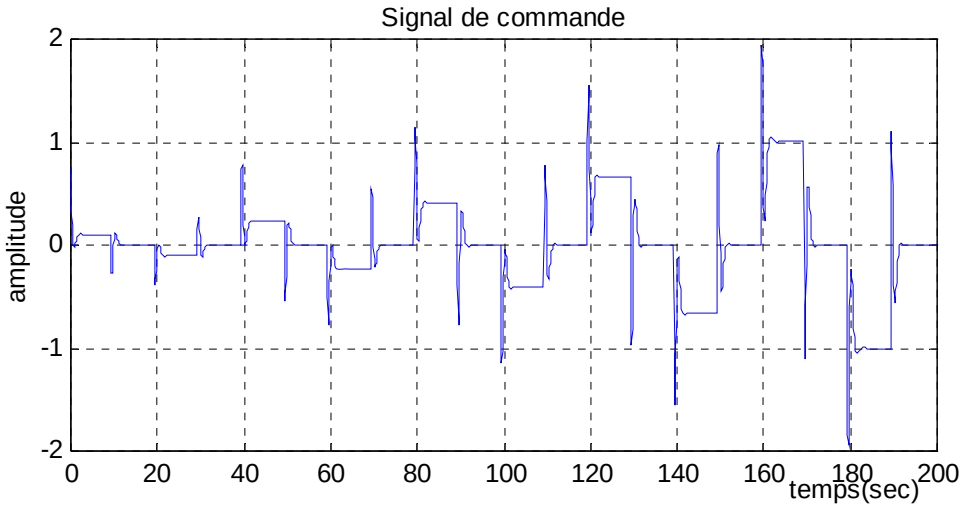


Figure 4.5 : Effort de commande ($N_1=1$, $N_2=8$, $N_u=1$ et $\lambda=10^{-4}$)

Afin de montrer l'effet des paramètres de réglage sur les performances du régulateur prédictif, nous présentons les résultats obtenus pour d'autres valeurs de ces paramètres. Les figures 4.6 et 4.7 montrent les performances obtenues pour les valeurs : $N_1=1$, $N_2=17$, $N_u=1$ et $\lambda=10^{-4}$. Nous obtenons toujours une bonne poursuite de la trajectoire de référence, mais cette fois-ci avec un signal de commande plus ou moins lisse. Sur les figures 4.8 et 4.9 nous avons représenté les performances obtenues pour : $N_1=1$, $N_2=8$, $N_u=1$ et $\lambda=13$, les performances de poursuite sont, dans ce cas, dégradées.

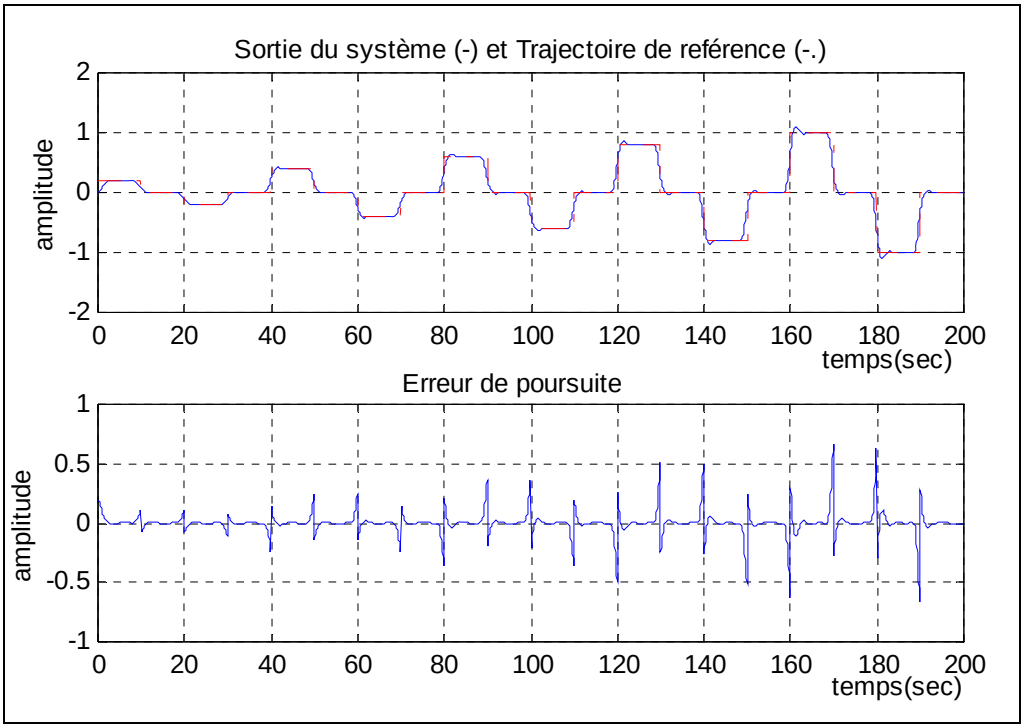


Figure 4.6 : Performances obtenues pour ($N_1=1$, $N_2=17$, $N_u=1$ et $\lambda=10^{-4}$)

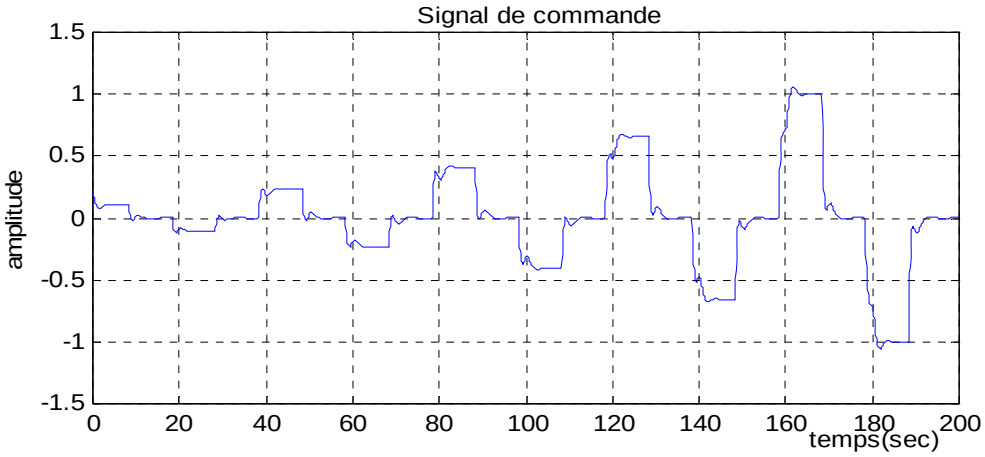


Figure 4.7 : Effort de commande ($N_1=1$, $N_2=17$, $N_u=1$ et $\lambda=10^{-4}$)

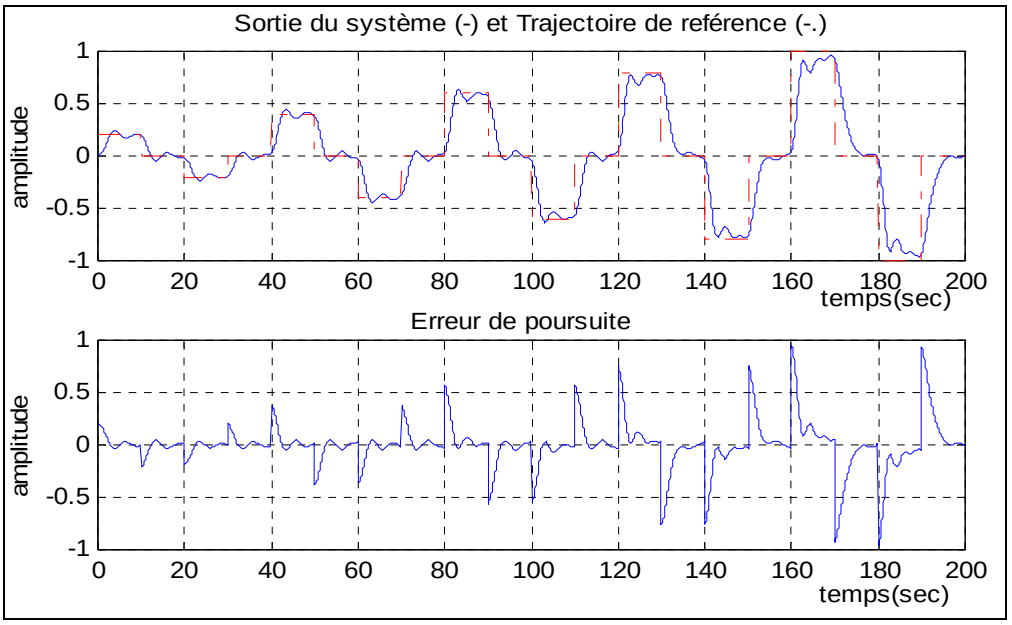


Figure 4.8 : Performances obtenues pour ($N_1=1$, $N_2=8$, $N_u=1$ et $\lambda=13$)

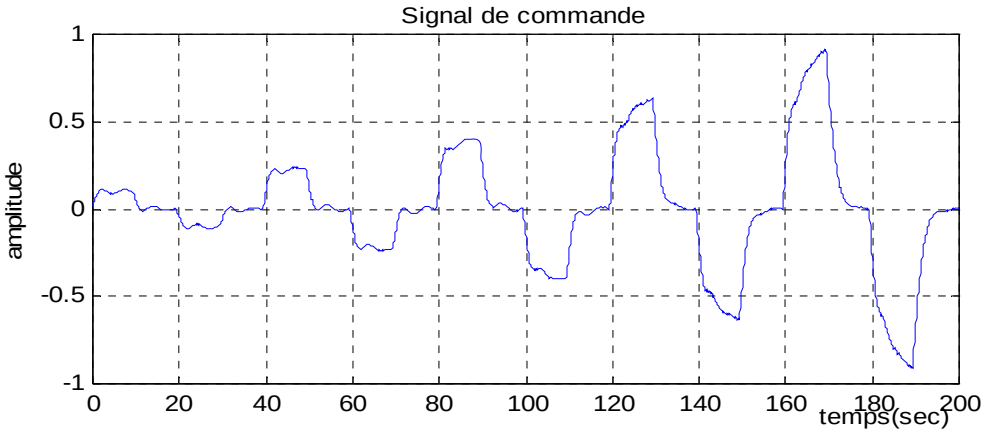


Figure 4.9 : Effort de commande ($N_1=1$, $N_2=17$, $N_u=1$ et $\lambda=13$)

Enfin, pour les mêmes valeurs des paramètres de réglage ($N_1=1$, $N_2=17$, $N_u=1$ et $\lambda=10^{-4}$), la figure 4.10 présente une comparaison entre les performances du régulateur prédictif à modèle neuronal et celles de la commande prédictive généralisée (GPC) appliquée au système linéarisé (équation (2.28)).

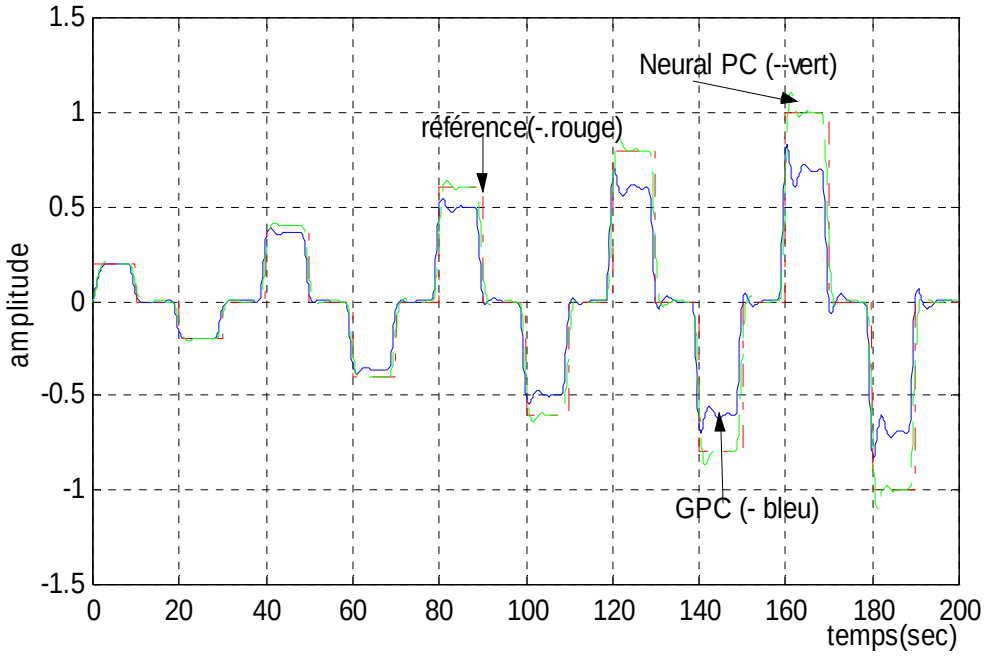
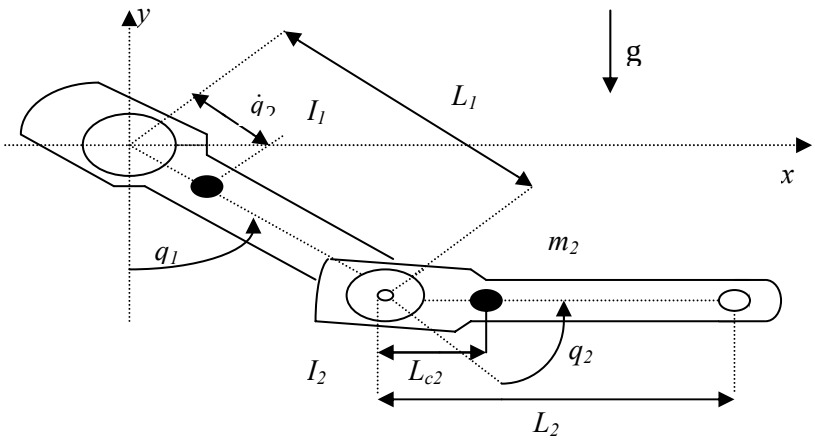


Figure 4.10 : Performances de poursuite, GPC (-) et NPC (--).

4.5.3.2 Commande d'un bras manipulateur

A. Description du bras

Nous considérons la commande du bras manipulateur à deux degrés de liberté représenté sur la figure 4.11.



Notation	Valeur	Unité
L_1	0.45	m
L_2	0.45	m
L_{c1}	0.091	m
L_{c2}	0.048	m
m_1	23.902	kg
m_2	3.88	kg
I_1	1.266	kgm^2/rad
I_2	0.093	kgm^2/rad
g	9.8	m/s^2

Figure 4.11 : Bras manipulateur à 2 degrés de liberté et valeurs de ces paramètres

Le modèle dynamique d'un bras manipulateur à n degrés de liberté est donnée par l'équation matricielle suivante [48], [49]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{4.43}$$

où $M(q)$ est une matrice symétrique définie positive ($n \times n$) appelée matrice d'inertie, $C(q,\dot{q})\dot{q}$ est un vecteur ($n \times 1$) appelé le vecteur de forces centrifuges et Coriolis, $G(q)$ est un vecteur ($n \times 1$) de forces et couples de gravitation, et τ est un vecteur ($n \times 1$) appelé le vecteur de forces extérieurs ou exogènes (les couples et les forces). $q, \dot{q}, et\ \ddot{q}$ sont respectivement, les vecteurs ($n \times 1$) des positions, des vitesses et des accélérations angulaires des articulations.

Dans le cas du bras considéré nous obtenons le modèle découplé suivant :

$$\begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} = \begin{pmatrix} m_1 L_{c1}^2 + m_2 L_1^2 + m_2 L_{c2}^2 + 2 m_2 L_1 L_{c2} \cos(q_2) + I_1 + I_2 & m_2 L_{c2}^2 + m_2 L_1 L_{c2} \cos(q_2) + I_2 \\ m_2 L_{c2}^2 + m_2 L_1 L_{c2} \cos q_2 + I_2 & m_2 L_{c2}^2 + I_2 \end{pmatrix} \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix}$$

$$+ \begin{pmatrix} -m_2 L_1 L_{c2} \sin(q_2)\dot{q}_2 & -m_2 L_1 L_{c2} \sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ m_2 L_1 L_{c2} \sin(q_2)\dot{q}_1 & 0 \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix}$$

$$+ \begin{pmatrix} (m_1 L_{c1} + m_2 L_1) g \cos(q_1) + m_2 g L_{c2} \cos(q_1 + q_2) \\ m_2 g L_{c2} \cos(q_1 + q_2) \end{pmatrix} \tag{4.44}$$

B. Mise en œuvre du contrôleur et résultats obtenus

Chaque articulation est commandée par un contrôleur monovariable (une entrée et une sortie). L'entrée et la sortie de chaque contrôleur sont respectivement, la vitesse angulaire désirée et le couple à appliquer sur l'articulation. La structure de commande est illustrée par la figure 4.12.

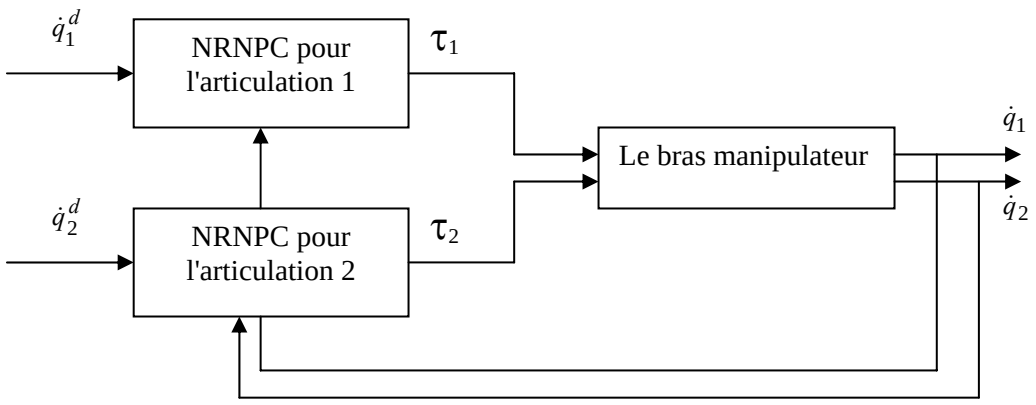


Figure 4.12 : Structure de commande

La première étape dans la mise en oeuvre des deux contrôleurs est l'obtention du modèle de prédiction pour chacune des deux articulations. A cet effet, nous avons utilisé, pour chaque

articulation, un réseau de neurones MLP statique à une seule couche cachée. Les deux modèles, dont la structure est donnée par les équations (4.45) et (4.46), expriment la relation entre le couple appliqué et la vitesse de rotation angulaire de chaque articulation.

$$\hat{\dot{q}}_1(k+1) = f_1(\tau_1(k), \tau_1(k-1), \dot{q}_1(k), \dot{q}_1(k-1), \dot{q}_1(k-2), \dot{q}_1(k-3)) \tag{4.45}$$

$$\hat{\dot{q}}_2(k+1) = f_2(\tau_2(k), \tau_2(k-1), \dot{q}_2(k), \dot{q}_2(k-1), \dot{q}_2(k-2), \dot{q}_2(k-3), \dot{q}_2(k-4)) \tag{4.46}$$

Les données d'apprentissage et de test sont représentées sur la figure 4.13, et les réponses à l'entrée de test des modèles obtenus après apprentissage sont données, respectivement, par les figures 4.14 et 4.15.

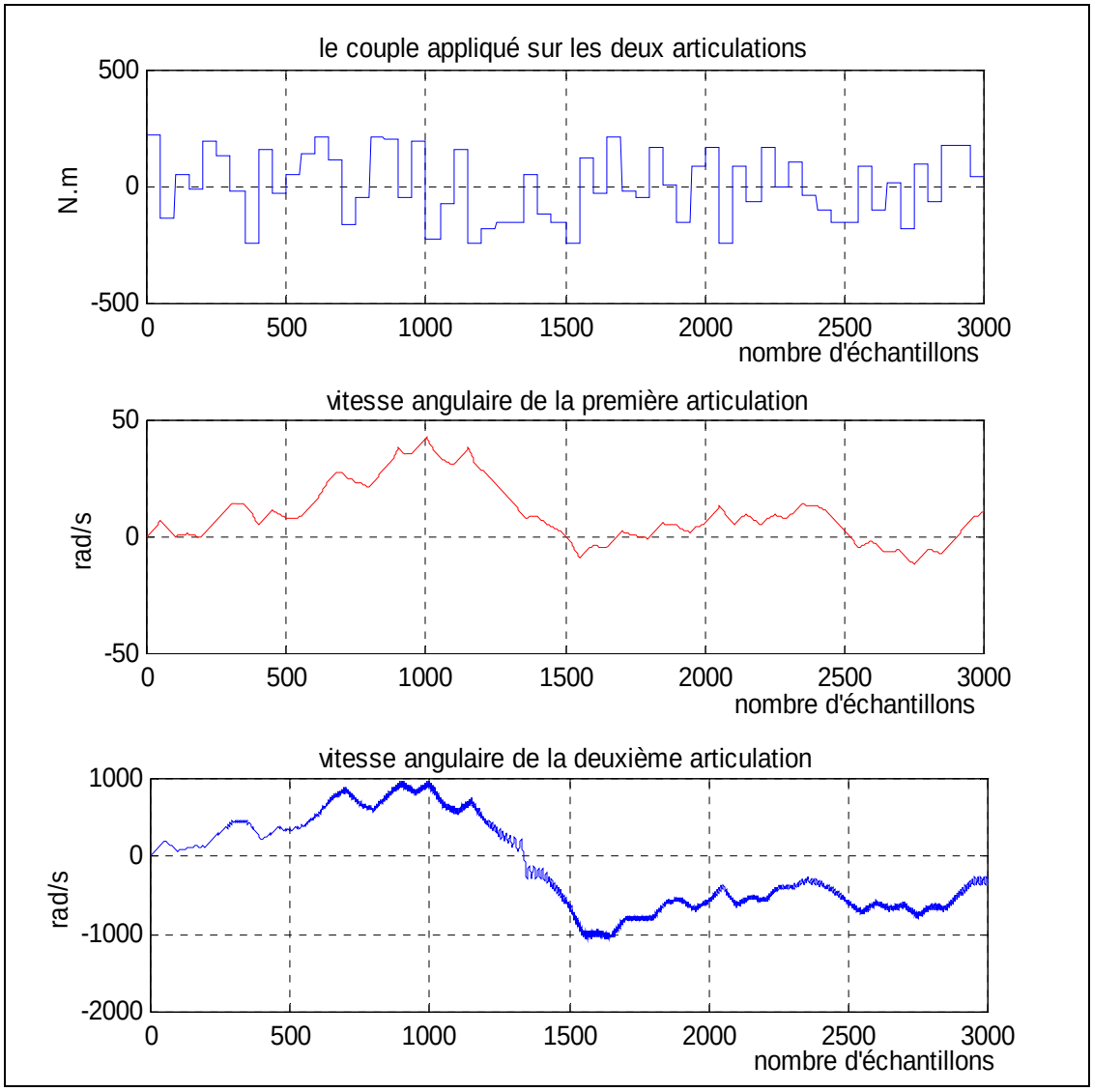


Figure 4.13 : Données d'apprentissage et de test

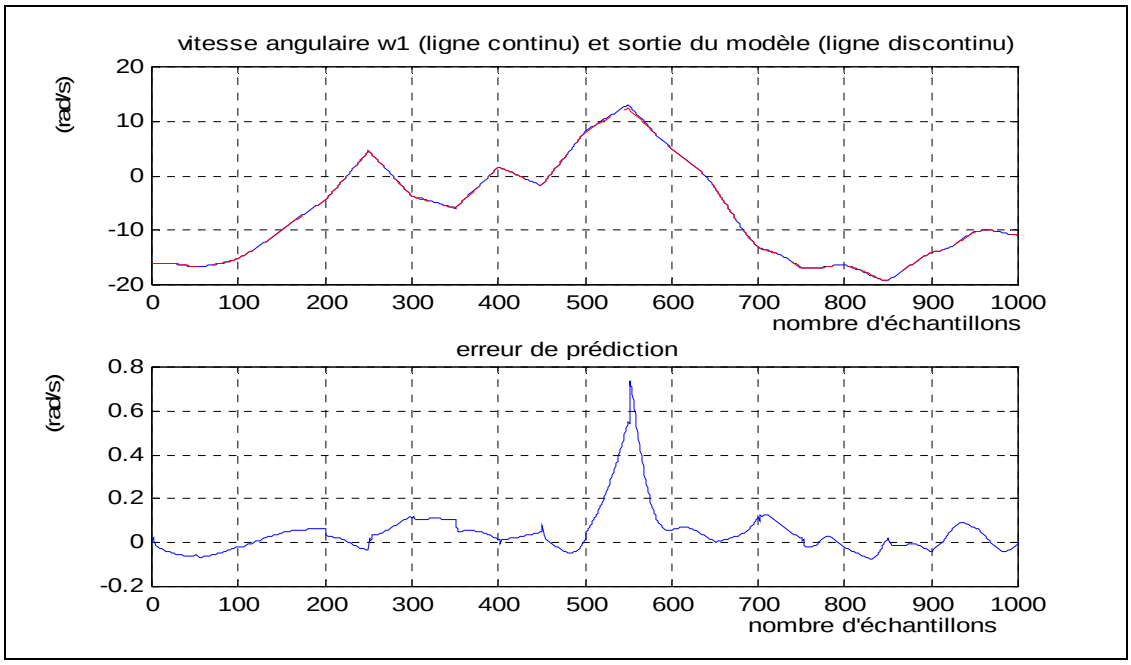


Figure 4.14 : Validation du modèle (articulation 1)

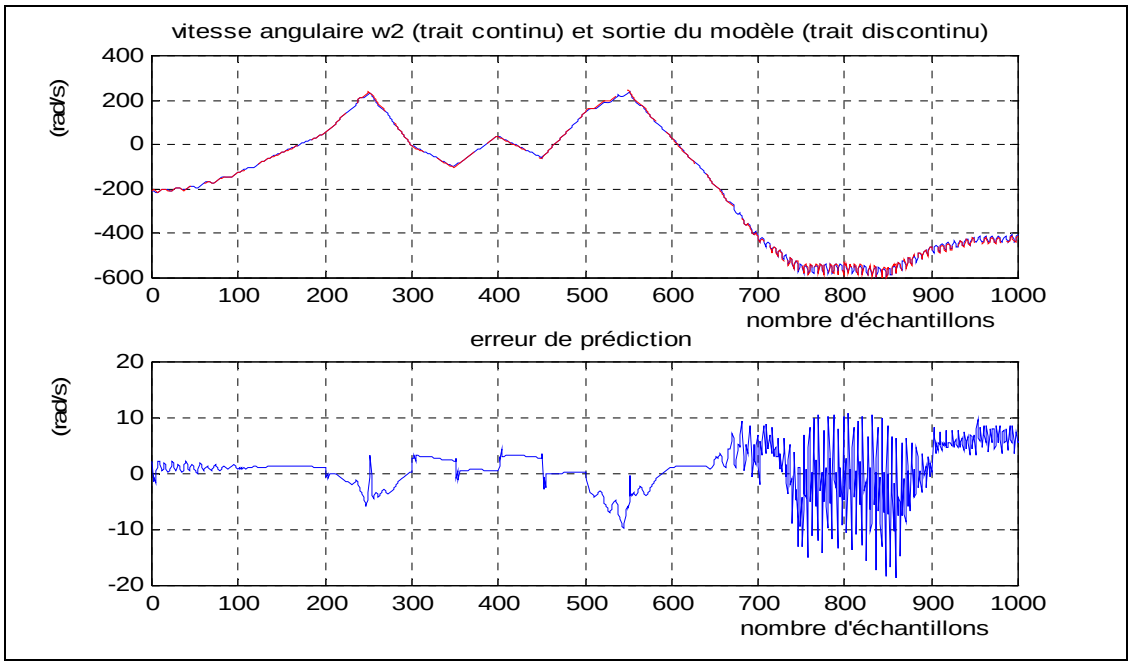


Figure 4.15 : Validation du modèle (articulation 2)

La deuxième étape est l'application de l'algorithme de commande pour calculer les couples de commande, qui permettent aux articulations du bras de suivre les vitesses spécifiées par les trajectoires de référence. Les performances des contrôleurs obtenues, en utilisant les trajectoires de références données par les équations suivantes : $\dot{q}_1^d(t) = 20\,sin(3\pi t)$ et $\dot{q}_2^d(t) = -20\,cos(3\pi t) + 20$, sont représentées sur les figures 4.16.a-c pour la première articulation et les figures 4.17.a-c pour la deuxième articulation. Il est visible, à partir de ces figures, que l'algorithme de commande réalise une bonne performance de poursuite avec une erreur acceptable.

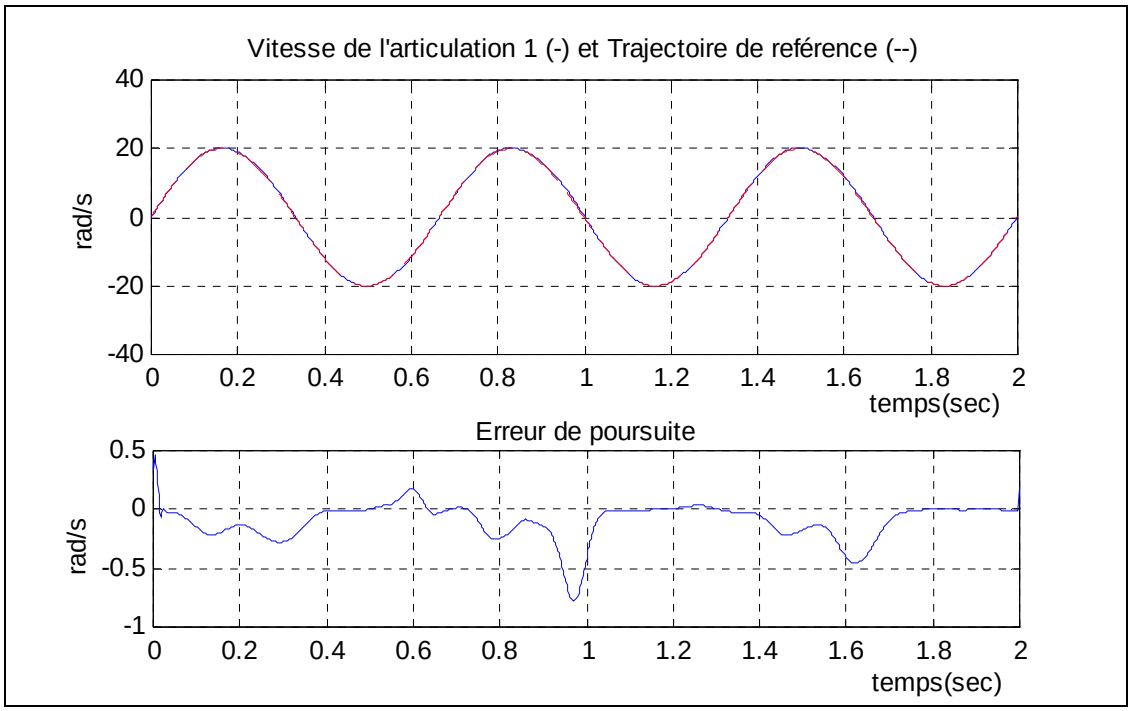


Figure 4.16.a : vitesse angulaire et erreur de poursuite pour l'articulation 1 (N_1=1,N_2= 3, N_u=1 et λ= 10^{-6})

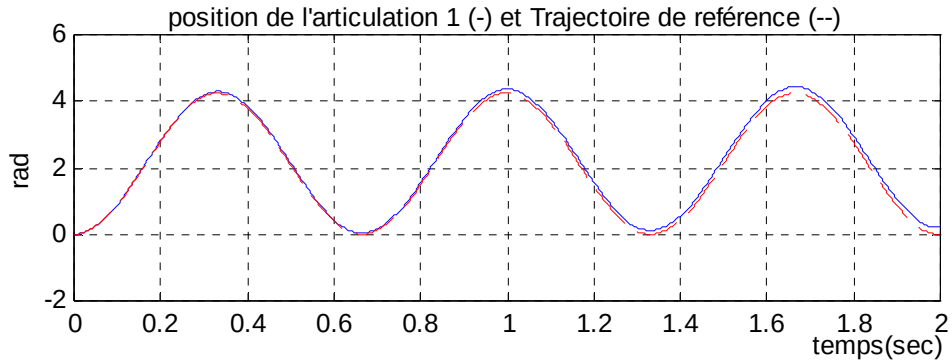


Figure 4.16.b : position de l'articulation 1 (N_1=1,N_2= 3, N_u=1 et λ= 10^{-6})

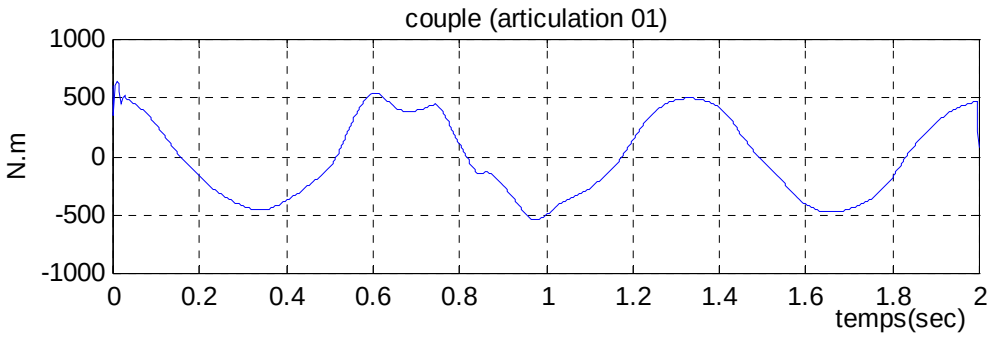


Figure 4.16.c : couple de l'articulation 1 (N_1=1,N_2= 3, N_u=1 et λ= 10^{-6})

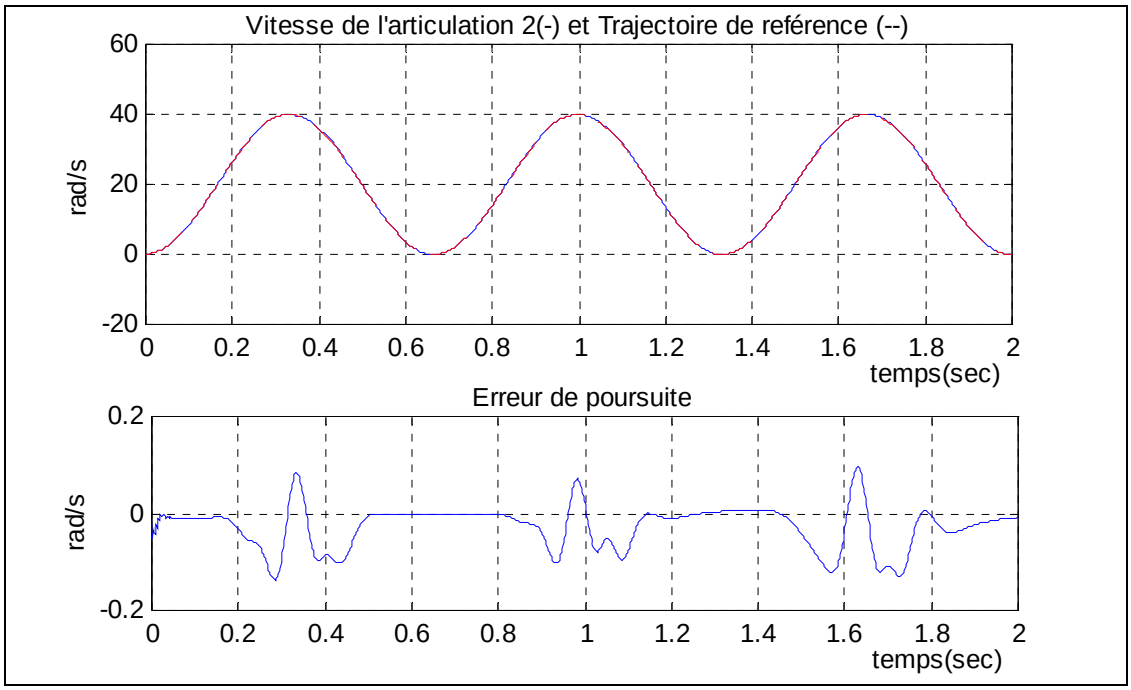


Figure 4.17.a : vitesse angulaire et erreur de poursuite pour l'articulation 2 (N_1=1,N_2= 3, N_u=1 et λ= 10^{-3})

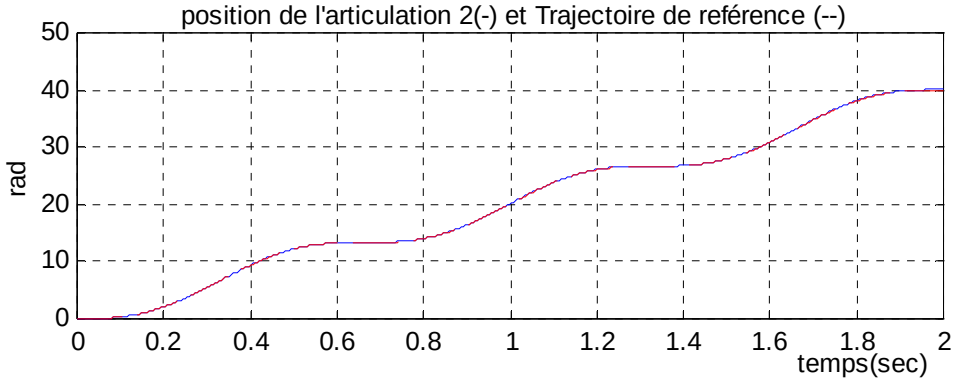


Figure 4.17.b : position de l'articulation 2 (N_1=1,N_2= 3, N_u=1 et λ= 10^{-3})

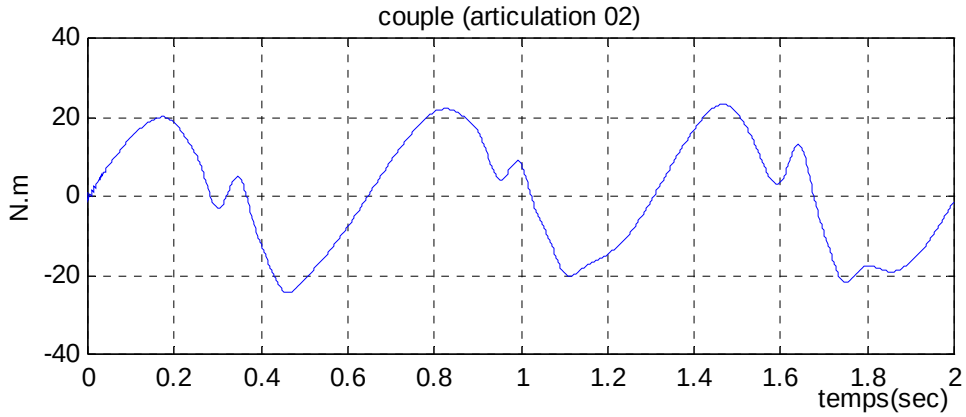


Figure 4.17.c : couple de l'articulation 2 (N_1=1,N_2= 3, N_u=1 et λ= 10^{-3})

4.6 Commande prédictive analytique à modèle neuronal

Dans ce paragraphe nous introduisons une méthode de commande prédictive (Analytic Neural Predictive Control : ANPC) permettant le calcul analytique de la loi de commande [43]. Cette méthode utilise un modèle neuronal et s'appuie, dans le calcul analytique de la loi de commande, sur l'idée de la décomposition, de la réponse totale du système en ses composantes libre et forcée, utilisée dans le cas de la GPC linéaire. Le principe de la méthode est illustré par le schéma bloc de la figure 4.18.

Nous notons que cette idée de décomposition a été utilisée par E. F. Camacho et al [50]. Dans leur travail, la réponse libre est calculée en utilisant le modèle non linéaire du système et la réponse forcée est calculée en utilisant un modèle linéaire obtenu par linéarisation du système autour d'un point de fonctionnement fixe.

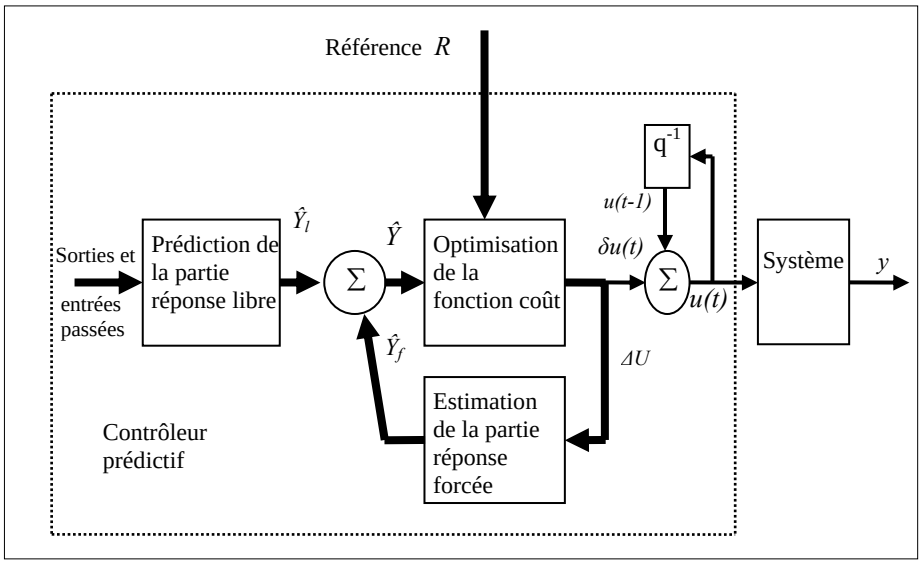


Figure 4.18 : Structure de commande

4.6. 1 Fonctions de coût

La loi de commande est obtenue par minimisation d'une fonction de coût quadratique portant sur les erreurs futures et les incréments de commande :

$$J = \sum_{j=N_1}^{N_2} (r(t+j) - \hat{y}(t+j))^T Q (r(t+j) - \hat{y}(t+j)) + \sum_{j=1}^{N_u} \delta u(t+j-1)^T S\, \delta u(t+j-1) \tag{4.47}$$

où N_1 est l'horizon minimal de prédiction, N_2 l'horizon maximal de prédiction, N_u l'horizon de commande, $r(t)$ la trajectoire de référence, $\hat{y}(t+j)$ les sorties prédites, $\delta u(t) = u(t)-u(t-1)$ l'incrément de commande, et Q et S sont deux matrices définies positive et semi positive respectivement.

L'objectif de la méthode, est de trouver une expression analytique à la loi de commande minimisant cette fonction de coût. Le calcul analytique de la commande permet de réduire considérablement le temps de calcul par rapport aux méthodes itératives, et par conséquent d'obtenir un algorithme de commande qui peut être utilisé dans le cas des systèmes à dynamique rapide.

4.6.2 Modèle de prédiction

Sans perte de généralisation aux autres modèles, nous admettons que le système à commander peut être représenté par le modèle NARX suivant :

$$\begin{aligned} &y(t) = NN\left[Y(t-1), U(t-1)\right] \\ &Y(t-1) = \left[y(t-1),\ldots,y(t-m)\right] \\ &U(t-1) = \left[u(t-d),\ldots,u(t-d-n+1)\right] \end{aligned} \tag{4.48}$$

où : d est le temps mort du système et $NN[\,]$ est le modèle neuronal du système.

Pour déterminer les sorties prédites $\hat{y}(t+k/t)$ nous utilisons un modèle de prédiction similaire à celui présenté dans le paragraphe (2.5.1), et dont ses éléments sont calculés à partir du modèle neuronal du système (modèle non incrémental). Ce modèle est donc de la forme suivante :

$$\hat{y}(t+k/t) = \hat{y}_f(t+k/t) + \hat{y}_l(t+k/t)\,, \tag{4.49}$$

avec :

- $$\hat{y}_f(t+k/t) = \sum_{i=0}^{k-1} g_i \delta u(t+k-i-1/t) \tag{4.50}$$

  où les coefficients g_i sont calculés au point de fonctionnement actuel en utilisant le modèle neuronal. Il est clair que $\hat{y}_f(t+k/t)$ dépend uniquement des incréments de commandes futures et présente. Ce terme correspond à la partie réponse forcée du modèle donné par l'équation (2.6).

- Le terme $\hat{y}_l(t+k/t)$ représente, dans notre cas, la réponse du modèle non linéaire du système aux conditions initiales disponibles à l'instant t, en gardant l'entrée de commande constante et égale à sa dernière valeur $u(t-1)$ pendant tout l'intervalle de prédiction. Il correspond à la partie réponse libre du modèle donné par l'équation (2.6).

Les sorties prédites $\hat{y}_l(t+k/t)$ sont générées d'une manière récursive en utilisant le modèle neuronal du système (figure 4.19) :

$$\begin{aligned} &\hat{y}_l(t+k/t) = NN\left[Y(t+k-1), U(t+k-1)\right] \\ &Y(t+k-1) = \left[y(t+k-1),\ldots,y(t+j),\cdots,y(t+k-m)\right] \qquad k = N_1,\ldots,N_2 \\ &U(t+k-1) = \left[u(t+k-d),\ldots,u(t+i),\ldots,u(t+k-d-n+1)\right] \end{aligned} \tag{4.51}$$

$$y(t+j) = \begin{cases} y(t+j) & si \quad j \prec 0 \\ \hat{y}_l(t+j) + \hat{y}_f(t+j) & si \quad j \geq 0 \end{cases}$$

$$j = k-m,\ldots,k-1$$

$$u(t+i) = \begin{cases} u(t+i) & si \quad i \prec 0 \\ u(t-1) & si \quad i \geq 0 \end{cases}$$

$$i = k-d-n+1,\ldots,k-d$$

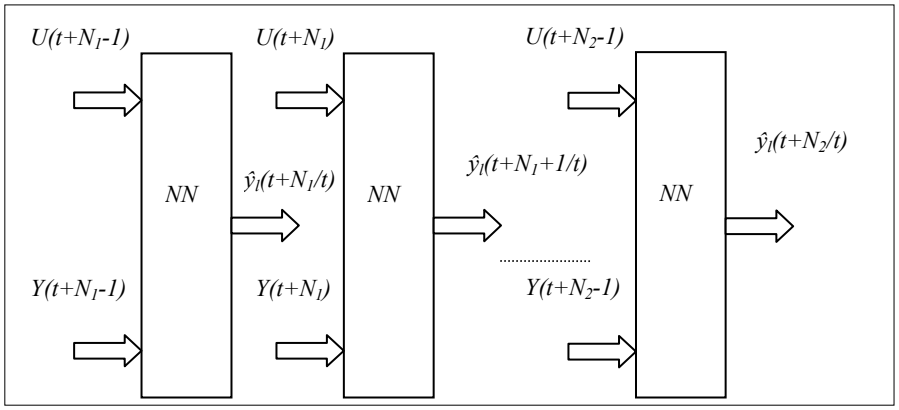


Figure 4.19 : Calcul des valeurs futures $\hat{y}_l(t+k/t)$

Bien que la superposition de deux réponses soit un principe de la théorie des systèmes linéaires, la représentation donnée par l'équation (4.49) est toujours non linéaire. Car les valeurs du terme $\hat{y}_l(t+k/t)$ sont calculées à l'aide du modèle non linéaire du système.

4.6.3 Calcul des coefficients g_i

Les valeurs futures $\hat{y}_s(t+k/t)$ de la réponse du système à un échelon sont calculées en utilisant le modèle neuronal, de la même manière que celle employée pour la prédiction des sorties $\hat{y}_l(t+k/t)$ (figure 4.19). Elles sont données par :

$$\begin{aligned}
&\hat{y}_s(t+k/t) = NN\left[Y(t+k-1)\ ;\ U(t+k-1)\right] \\
&Y(t+k-1) = \left[y_s(t+k-1),...,y_s(t+k-m)\right] \qquad k = N_1,....,N_2 \\
&U(t+k-1) = \left[u(t+k-d),...,u(t+i),.....,u(t+k-d-n+1)\right]
\end{aligned} \tag{4.52}$$

$$u(t+i) = \begin{cases} u(t+i) & si \quad i \prec 0 \\ u(t-1)+du & si \quad i \geq 0 \end{cases}$$

$$i = k-d-n+1,.....,k-d$$

Des équations (4.49) et (4.52) nous avons :

$$\hat{y}_s(t+k/t) - \hat{y}_l(t+k/t) = \sum_{i=0}^{k-1} g_i \delta u(t+k-i-1/t) = g_0 \delta u(t+k-1) + g_1 \delta u(t+k-2) + ... + g_{k-1}\delta u(t)\ ,$$

tout les termes *δu(t+k-1)*,..., *δu(t+1)* seront nuls et il restera uniquement le terme dépendant de *δu(t)*, car :

$$\delta u(t+k-1) = u(t+k-1) - u(t+k-2) = u(t-1) + du - u(t-1) - du = 0\ ,$$

d'où :

$$g_{k-1} = \frac{\hat{y}_s(t+k/t) - \hat{y}_l(t+k/t)}{du} \tag{4.53}$$

où : *du* est l'amplitude de l'impulsion d'entrée ; sa valeur devant être choisie proche de celle de $\delta u(t)$. Puisque la valeur *de* $\delta u(t)$ n'est disponible qu'après la phase d'optimisation, une bonne manière pour choisir la valeur de *du* consiste à prendre $du = \delta u(t-1)$, c'est à dire la valeur de $\delta u(t)$ obtenue dans l'itération précédente. Au régime permanent, $\Delta U(t)$ tend vers zéro, et si on choisit $du = \delta u(t-1)$ l'équation (4.53) devient indéfinie. Il est donc nécessaire de fixer une valeur du_{min} pour le pas *du*.

4.6.4 Minimisation de la fonction de coût

En utilisant la représentation matricielle, il est possible d'écrire l'expression (4.49) sous la forme suivante :

$$\hat{Y}(t) = G\Delta U(t) + \hat{Y}_l(t) \tag{4.54}$$

où :

$$\hat{Y}(t) = \begin{bmatrix} \hat{y}(t+N_1/t) \\ \hat{y}(t+N_1+1/t) \\ \vdots \\ \hat{y}(t+N_2/t) \end{bmatrix} \qquad G = \begin{bmatrix} g_{N_1-1} & g_{N_1-2} & \cdots\cdots g_0 & 0\cdots\cdots\cdots\cdots\cdots 0 \\ g_{N_1} & g_{N_1-1} & \cdots\cdots\cdots\cdots g_0 & 0\cdots\cdots\cdots 0 \\ \vdots & & & \\ \vdots & & & \\ g_{N_2-1} & g_{N_2-2} & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & g_{N_2-N_u} \end{bmatrix}$$

$$\Delta U(t) = \begin{bmatrix} \delta u(t/t) \\ \delta u(t+1/t) \\ \vdots \\ \delta u(t+N_u-1/t) \end{bmatrix} \qquad \hat{Y}_l(t) = \begin{bmatrix} \hat{y}_l(t+N_1/t) \\ \hat{y}_l(t+N_1+1/t) \\ \vdots \\ \hat{y}_l(t+N_2/t) \end{bmatrix} \qquad R(t) = \begin{bmatrix} r(t+N_1) \\ r(t+N_1+1) \\ \vdots \\ r(t+N_2) \end{bmatrix}$$

et la fonction de coût devient :

$$\begin{aligned} J &= (R(t)-\hat{Y}(t))^T Q(R(t)-\hat{Y}(t)) + \Delta U(t)^T S\Delta U(t) \\ &= (R(t)-\hat{Y}_l(t)-G(t)\Delta U(t))^T Q(R(t)-\hat{Y}_l(t)-G(t)\Delta U(t)) + \Delta U^T S\Delta U(t) \end{aligned} \tag{4.55}$$

En dérivant *J* par rapport à $\Delta U(t)$, nous obtenons :

$$\frac{\partial J}{\partial \Delta U(t)} = 2G^T(t)Q(\hat{Y}_l(t) - R(t)) + 2(G^T(t)QG(t) + S)\Delta U(t) \tag{4.56}$$

Puisque la séquence optimale des incréments de commande correspond à $\frac{\partial J}{\partial \Delta U(t)} = 0$, nous obtenons alors :

$$\Delta U(t) = (G^T(t)QG(t) + S)^{-1} G^T(t)Q(R(t) - \hat{Y}_l(t)) \tag{4.57}$$

La valeur optimale de la commande à l'instant *t* est donnée par :

$$u(t) = u(t-1) + \delta u(t) \tag{4.58}$$

où $\delta u(t)$ est le premier élément de $\Delta U(t)$.

4.6.5 Algorithme

Les étapes principales de l'algorithme de calcul de la commande sont résumées par les points suivants :

1- spécifier la trajectoire de référence et choisir les paramètres de réglage ;

2- en utilisant le modèle du système, calculer les valeurs futures $\hat{y}_l(t+k/t)$ (équation (4.51)) ;

3- en utilisant le modèle du système calculer les valeurs futures $\hat{y}_s(t+k/t)$ (équation (4.52)) ;

4- calculer les coefficients g_i (équation (4.53)) et former la matrice $G(t)$;

5- calculer le vecteur des incréments de la commande $\Delta U(t)$ (équation (4.57)) ;

6- la valeur de la commande $u(t)$ à l'instant d'échantillonnage t est donnée par l'équation (4.58) ;

7- aller à l'étape 2 pour calculer une nouvelle valeur de $u(t)$.

4.6.6 Applications

4.6.6.1 Commande d'un réacteur continu parfaitement mélangé

Nous considérons ici l'application de cette stratégie de commande prédictive à modèle neuronal à la commande du CSTR (Continuous Stirred Tank Reactor, figure 4.20). Il s'agit d'un réacteur continu parfaitement mélangé, dans lequel un produit A sera transformé en un autre produit B selon une réaction chimique exothermique. Le volume v utilisé du réacteur est constant et le mélange est considéré comme parfait avec une température T supposée uniforme. Le réacteur travaille en continu, il est alimenté par la charge fraîche de concentration et température initiales C_{a0} et T_0 respectivement, et les produits de la réaction sont soutirés de manière constante. Ce processus de transformation (A vers B) est décrit par les équations d'état suivantes :

$$\begin{aligned} \dot{C}_a(t) &= \frac{q}{v}(C_{a0} - C_a(t)) - k_0 C_a(t) e^{-\frac{E}{RT(t)}} \\ \dot{T}(t) &= \frac{q}{v}(T_0 - T(t)) + k_1 C_a(t) e^{-\frac{E}{RT(t)}} + k_2 q_c(t)(1 - e^{-\frac{k_3}{q_c(t)}}(T_{c0} - T(t)) \end{aligned} \tag{4.59}$$

les constantes k_1, k_2 et k_3 sont données par :

$$k_1 = -\frac{\Delta H k_0}{\rho C_p} \qquad k_2 = \frac{\rho_c C_{pc}}{\rho C_p v} \qquad k_3 = \frac{h_a}{\rho_c C_{pc}}$$

$C_a(t)$ est la concentration du produit A exprimée en *mol/l*, $T(t)$ est la température du mélange exprimée en *Kelvin*, $q_c(t)$ est le débit du liquide de refroidissement utilisé pour contrôler la température de la réaction, il est exprimé en *l/min* et q est le débit du processus exprimé en *l/min*. Les valeurs de toutes les constantes chimiques et thermodynamiques sont données dans la référence [28].

La transformation est une réaction chimique du premier ordre irréversible et exothermique. Le dégagement de la chaleur conduit à l'augmentation de la température *T(t)* du réacteur, dés que cette dernière dépasse la température d'activation relative à la réaction et après un certain pourcentage de conversion, le réactif résultant *B* désiré se transforme en un troisième réactif *C* indésirable. Une autre conséquence plus importante est l'emballement thermique du réacteur qui peut nous mener à une explosion de l'installation. D'où la nécessité d'agir sur la température en utilisant un liquide de refroidissement.

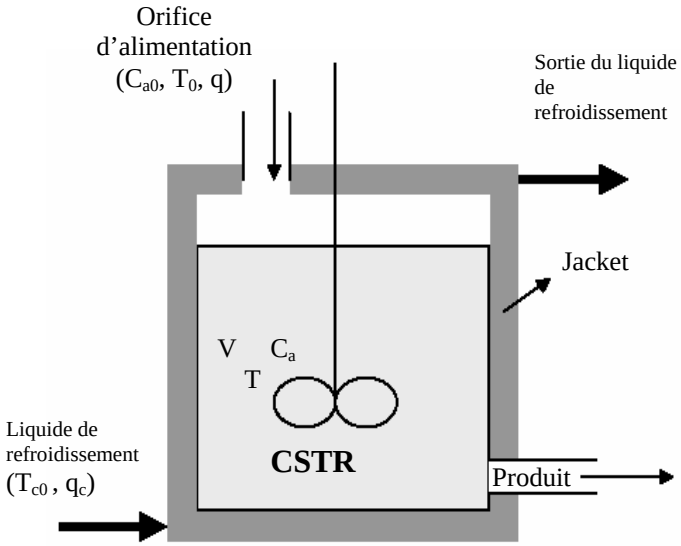


Figure 4.20 : Réacteur continu parfaitement mélangé

Le système d'équations (4.59) montre que la variation de la valeur du débit du liquide de refroidissement $q_c(t)$ (la variable de commande) agit directement sur celle de la température *T(t)*, et la variation de celle-ci agit sur la valeur de la concentration $C_a(t)$ (la variable de sortie). C'est-à-dire, il est possible de contrôler la concentration $C_a(t)$ en contrôlant la température.

Nous avons choisi d'appliquer l'algorithme proposé à la commande du CSTR, car ce dernier présente un comportement dynamique fortement non linéaire (équation (4.59)). En plus, le modèle du CSTR est devenu l'exemple typique pour tester les résultats théoriques obtenus dans le domaine des systèmes non linéaires.

A. Modélisation neuronale du CSTR

Pour l'identification du CSTR, nous avons utilisé :

- un ensemble de données de 7500 échantillons avec une fréquence d'échantillonnage $T_s = 0.1$ min (figure 4.20). Cet ensemble est divisé en deux parties : 4000 échantillons pour faire l'apprentissage au réseau et 3500 pour valider le modèle. Les échantillons de l'apprentissage sont centrés et réduits à la même valeur de la variance (figure 4.22) ;

- un réseau de neurones statique à deux couches, avec 10 neurones sur la couche cachée. La fonction d'activation des neurones cachés est la tangente hyperbolique et celle du neurone de sortie est linéaire.
- un modèle neuronal NARX, dont la structure (figure 4.23) est donnée par l'équation ci-après :

$$\hat{C}_a(k) = NN(C_a(k-1), C_a(k-2), q_c(k-1), q_c(k-2)) \tag{4.60}$$

où : *NN( )* est le réseau de neurone utilisé ;

- une méthode d'apprentissage basée sur l'algorithme de Levenberg-Marquardt a été utilisée pour l'apprentissage du réseau.

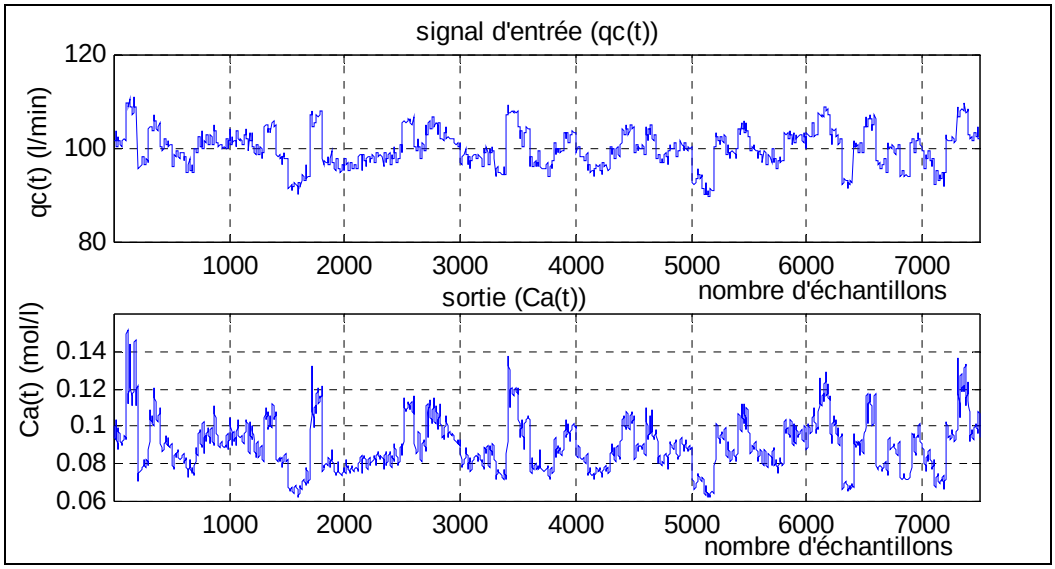


Figure 4.21 : Données d'apprentissage et de validation

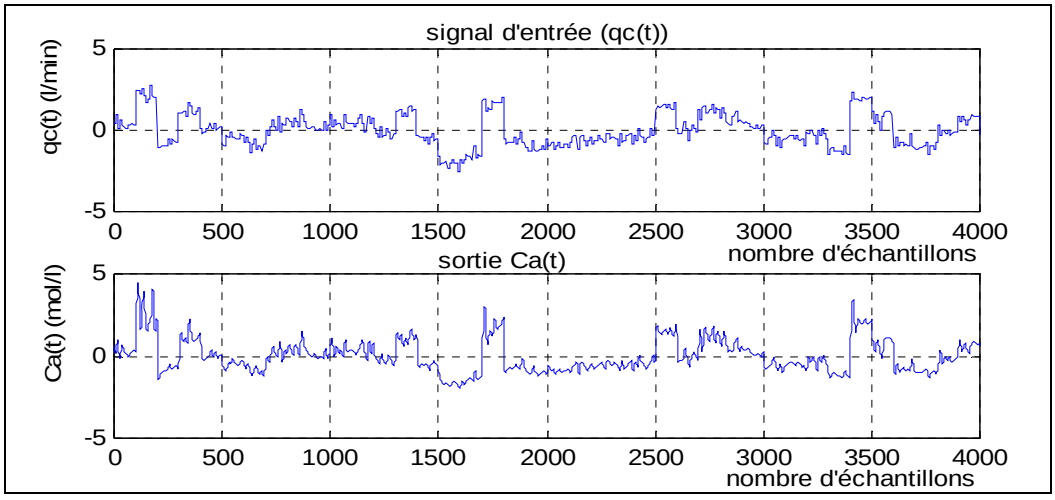


Figure 4.22 : Données d'apprentissage : centrées et réduites

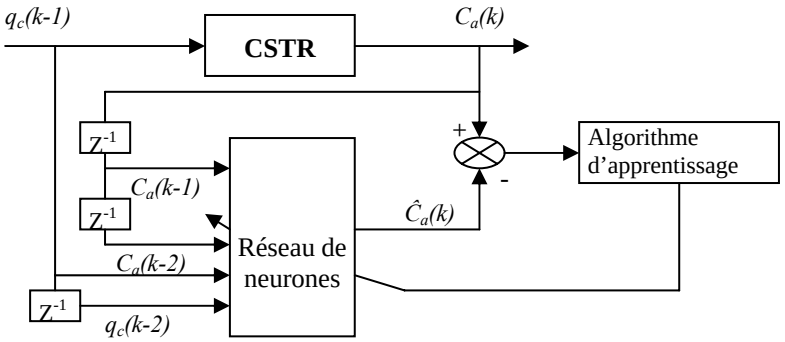


Figure 4.23 : Schéma de l'identification.

La figure 4.24 montre l'évolution de l'erreur quadratique moyenne pendant la phase d'apprentissage. Pratiquement une valeur de l'ordre de 10^{-7} est obtenue au bout de 200 itérations. Les sorties du système et du modèle sont représentées sur la figure 4.25, les deux courbes sont superposées, et l'erreur de prédiction est très faible.

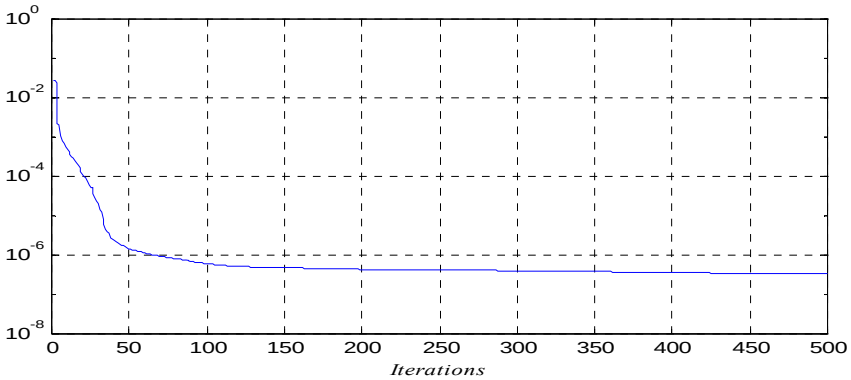


Figure 4.24 : Evolution de l'erreur quadratique moyenne pendant l'apprentissage.

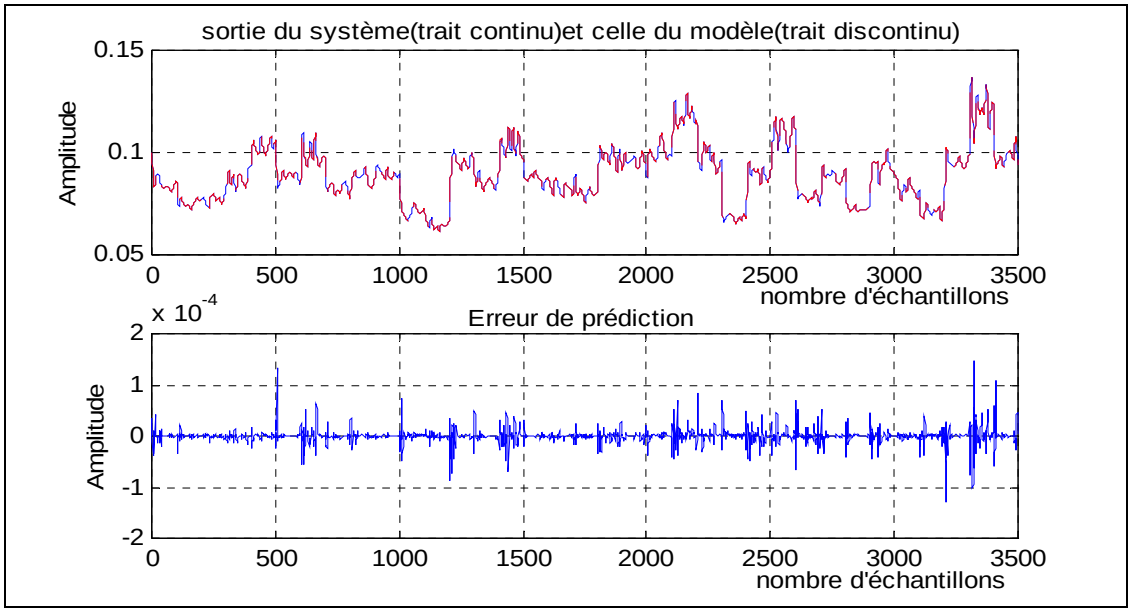


Figure 4.25 : Validation du modèle

B. Mise en œuvre du contrôleur et résultats obtenus

Nous avons utilisé une trajectoire de référence en forme d'escalier (figure 4.26) pour mettre en évidence le caractère non linéaire du système. La durée de chaque pas est de 10 min et les valeurs minimale et maximale de son amplitude sont respectivement 0.08 mol/l et 0.12 mol /l. La loi de commande prédictive a été calculée en utilisant les paramètres suivants :

$$Q = I^{N_2 x N_2} \quad S = 10^{-4} I^{N_u x N_u} \quad du_{\min} = 2\ (l/\min) \quad N_1 = 1 \quad N_2 = 7 \quad N_u = 2$$

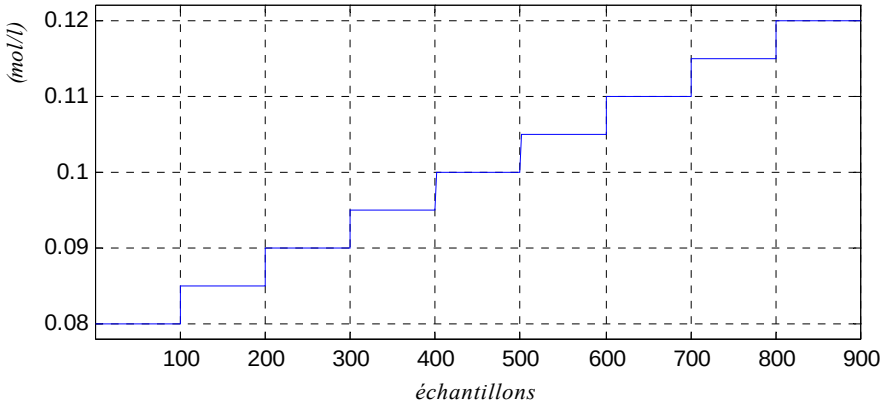


Figure 4.26 : Trajectoire de référence

Les performances du contrôleur obtenu sont représentées sur la figure 4.27. Une bonne poursuite de la référence est obtenue avec un effort de commande plus ou moins lisse (smooth). Ensuite, pour les mêmes valeurs des paramètres définies ci-dessus, nous avons appliqué au milieu de chaque pas une perturbation d'amplitude égale à ±0.005 mol/l. La figure 4.28 montre que le contrôleur compense rapidement l'effet de cette perturbation par des pics d'amplitude raisonnable, et l'erreur de poursuite est très faible (figure 4.29).

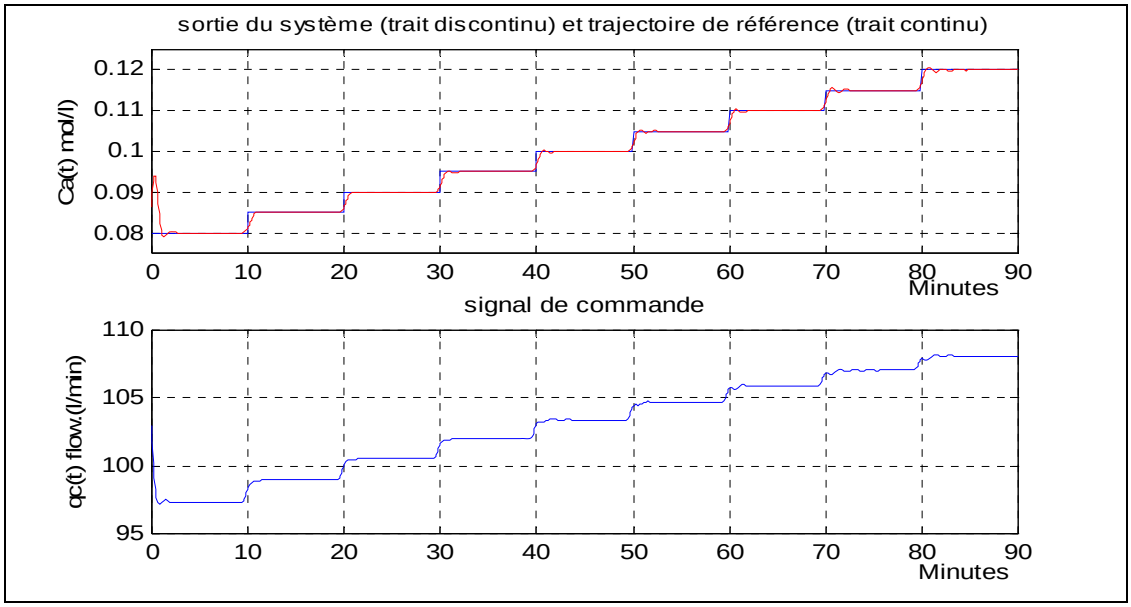


Figure 4.27 : Performances du contrôleur sans perturbations (N_2=7 N_u=2)

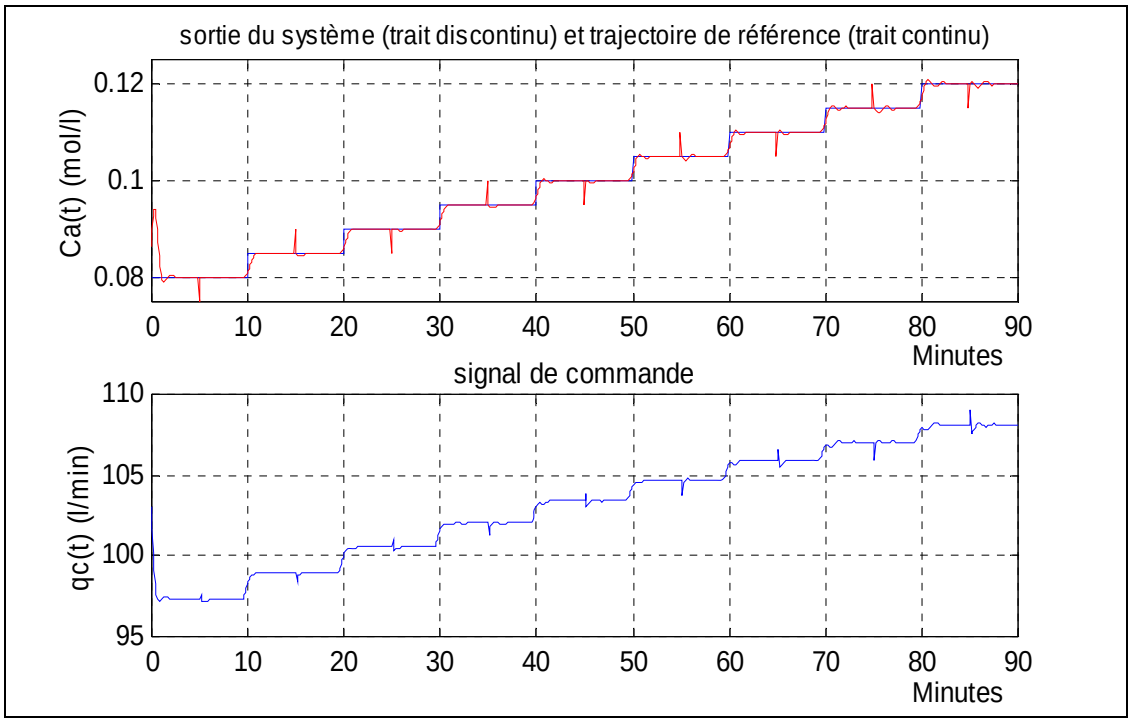


Figure 4.28 : Performance du contrôleur avec perturbations (N_2=7 N_u=2)

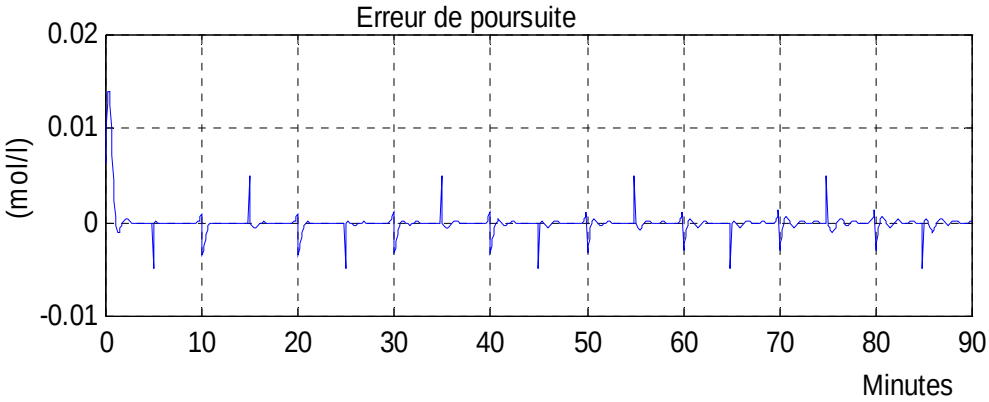


Figure 4.29 : Erreur de poursuite avec perturbation (N_2=7, N_u=2)

Les figures 4.30 et 4.31 présentent les performances obtenues, pour le cas où $N_2 = 10$, $N_u = 2$. Les performances sont meilleures que celles obtenues dans les autres cas. En particulier, nous obtenons une commande suffisamment lisse (sans ondulations).

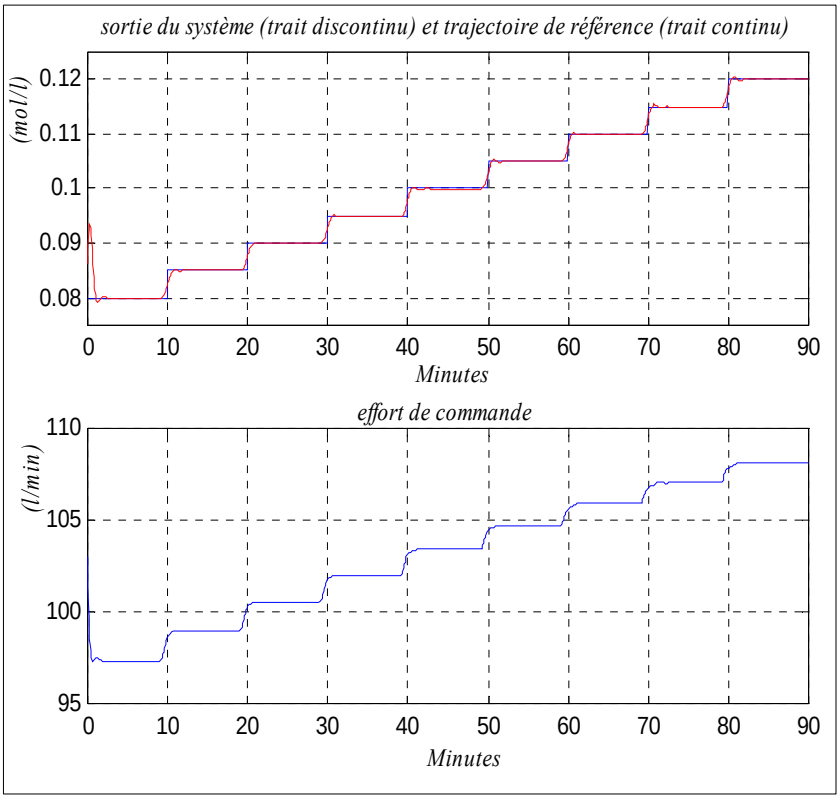


Figure 4.30 : Performances du contrôleur sans perturbations (N_2=10, N_u=2)

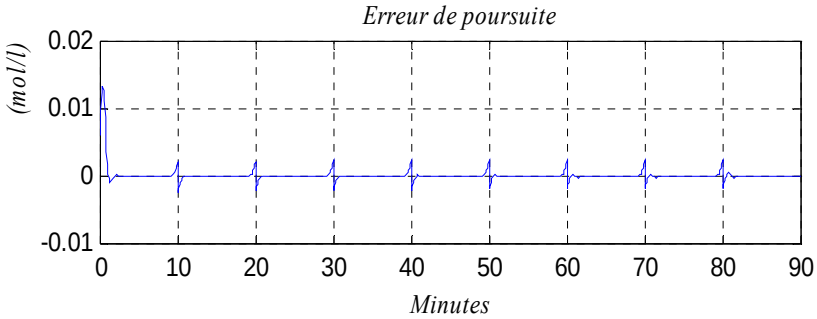


Figure 4.31 : Erreur de poursuite sans perturbation (N_2=10, N_u=2)

4.6.6.2 Commande d'un bras manipulateur

En utilisant l'algorithme de commande (ANPC), nous reconsidérons la commande du bras manipulateur décrit dans le paragraphe 4.5.3.2. Nous utilisons la même structure de commande (figure 4.12), les mêmes trajectoires de référence et les mêmes modèles de prédiction (équations 4.45 et 4.46) développés dans le paragraphe 4.5.3.2. Les résultats de simulation obtenus, sont donnés par les figures 4.32.a-c pour la première articulation et par les figures 4.33.a-c pour la deuxième articulation. Les valeurs des paramètres de réglage sont N_1 =1, N_2 =3, N_u =1, S =10^{-5} et Q =I_{2x2} pour la première articulation et N_1 =1, N_2 =2, N_u =1, S =10^{-2} et Q =I_{2x2} pour la deuxième articulation.

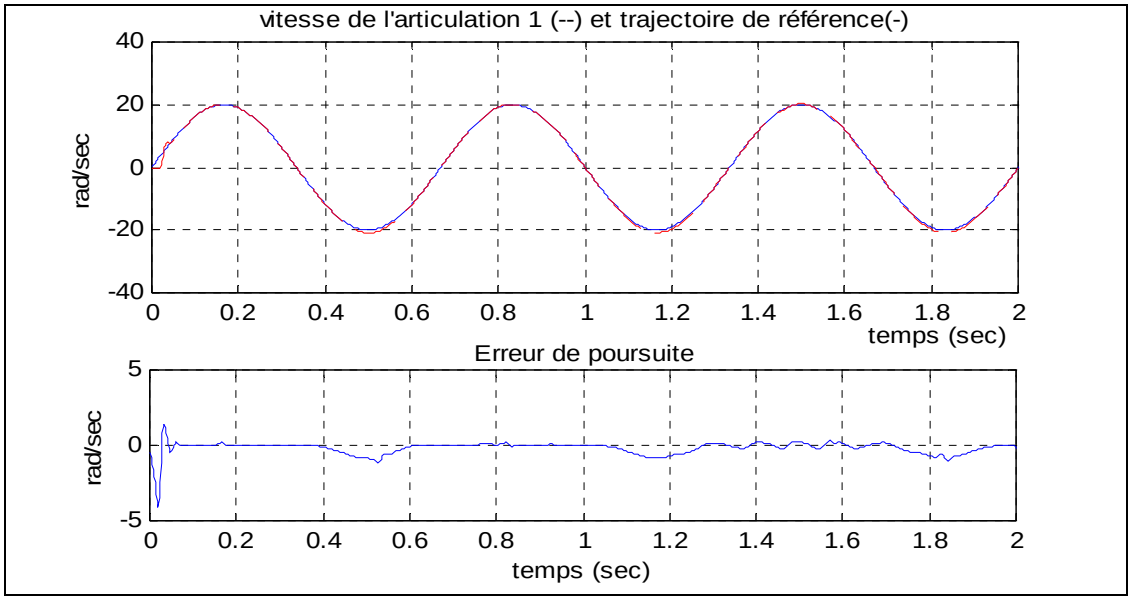


Figure 4.32.a : vitesse angulaire et erreur de poursuite pour l'articulation 1

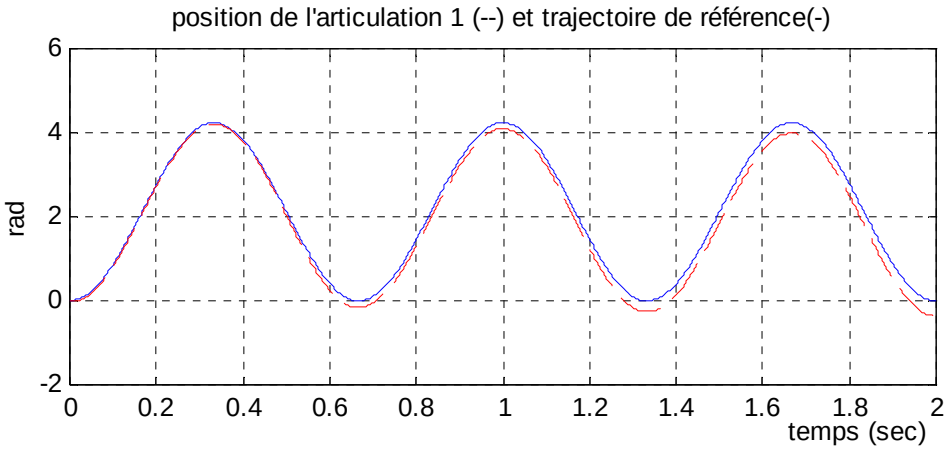


Figure 4.32.b : position de l'articulation 1

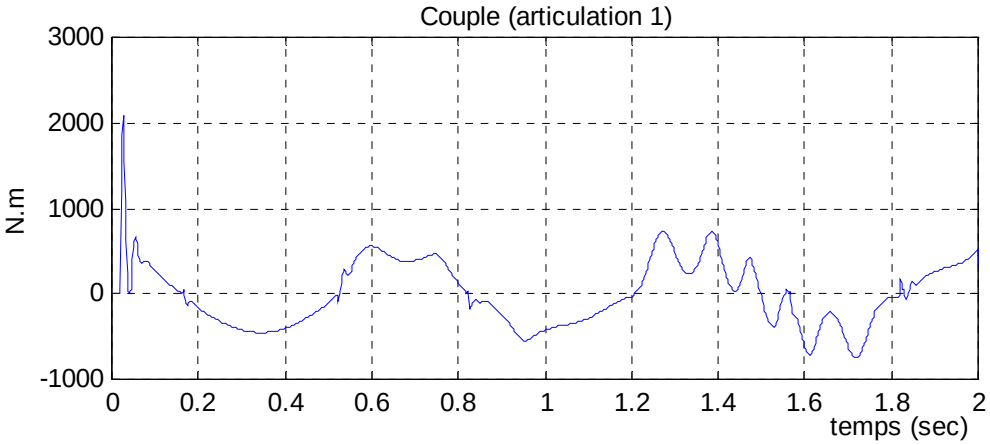


Figure 4.32.c : couple de l'articulation 1

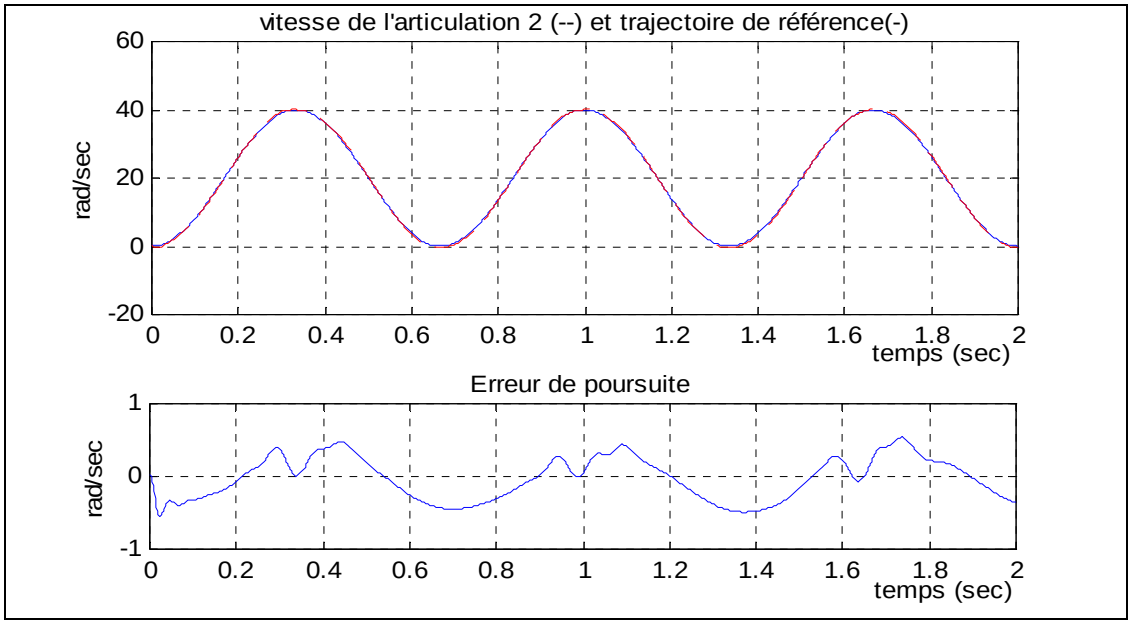


Figure 4.33.a : vitesse angulaire et erreur de poursuite pour l'articulation 2

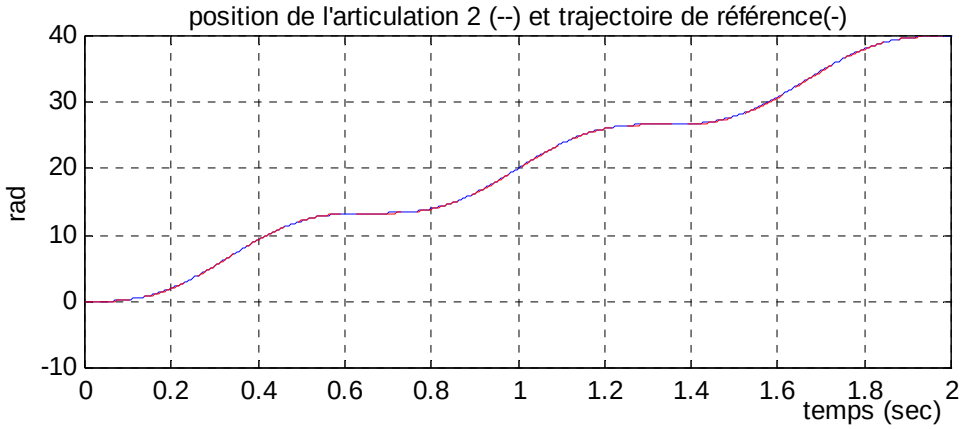


Figure 4.33.b : position de l'articulation 2

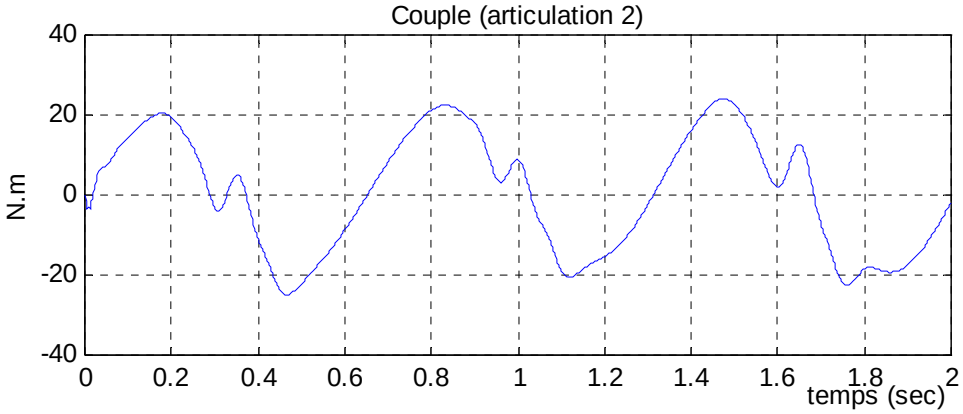


Figure 4.33.c : couple de l'articulation 2

4.6.6.3 Commande d'un moteur à courant continu

Dans ce paragraphe nous envisageons la mise en place de la commande prédictive à modèle neuronal (ANPC) et de la command prédictive généralisée (GPC) sur un moteur à courant continu. Le but est de mettre en évidence l'apport de l'approche proposée (ANPC) par rapport à l'approche linéaire (GPC). Le moteur que nous avons considéré peut être se modéliser par l'équation différentielle suivante [51] :

$$\frac{d^2\Omega}{dt^2}+\left(\frac{Lf+RJ}{LJ}\right)\frac{d\Omega}{dt}+\left(\frac{Rf+K^2}{Lf}\right)\Omega+\frac{R}{LJ}C_{ch}=\frac{K}{LJ}U \tag{4.61}$$

ou en utilisant la représentation d'état par :

$$\begin{cases}\dot{x}_1 = x_2\\ \dot{x}_2 = -a\,x_2 - b\,x_1 - c\,C_{ch} + d\,U\end{cases} \tag{4.62}$$

avec : $a=\frac{Lf+RJ}{LJ}$, $b=\frac{Rf+K^2}{LJ}$, $c=\frac{R}{LJ}$ et $d=\frac{K}{LJ}$

Les valeurs des paramètres du moteur sont :

- Résistance de l'induit R=1.65 Ohm ;
- Self de l'induit L=0.04 H ;
- Constante K=0.518 V.s/rad ;
- Coefficient de frottement f= 0.0055 N.m.s/rad ;
- Moment d'inertie J=0.022 kg.m^2 ;
- Couple de charge C_{ch}=0.0243 N.m ;
- U est la tension aux bornes de l'induit.

Le modèle numérique linéaire du système que nous avons obtenu par identification en utilisant une période d'échantillonnage T_e=5 ms est donné par :

$$H(z)=\frac{0.0069z^{-1}+0.0064z^{-2}}{1-1.8055z^{-1}+0.8126z^{-2}} \tag{4.63}$$

Pour la mise en œuvre de l'algorithme ANPC nous avons utilisé un réseau de neurone MLP statique à une seule couche cachée. La structure du modèle est donnée par l'équation suivante :

$$\hat{\Omega}(t)=f(\Omega(t-1),\Omega(t-2),U(t-1),U(t-2)) \tag{4.64}$$

Les figures 4.34.a et 4.34.b montrent les performances obtenues pour les valeurs des paramètres de réglage N_l=1, N_u=1, λ=10^{-4} et N_2 est égale 8 dans le cas (a) et 12 dans le cas (b). Nous constatons que les performances des deux contrôleurs sont comparables. Plus particulièrement, le temps de réponse obtenu dans le cas de l'approche ANPC est inférieur à celui obtenu dans le cas de l'approche GPC. Ce résultat est obtenu au détriment d'un dépassement d'amplitude faible obtenu dans le cas de l'approche ANPC.

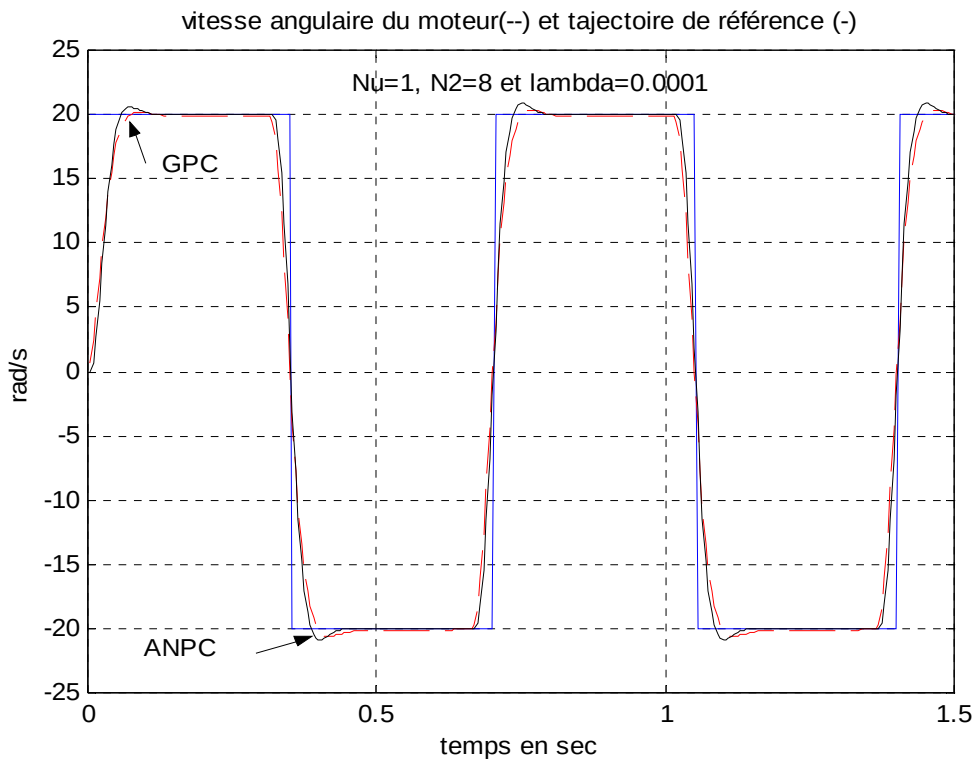


Figure 4.34.a : vitesse angulaire du moteur et trajectoire de référence

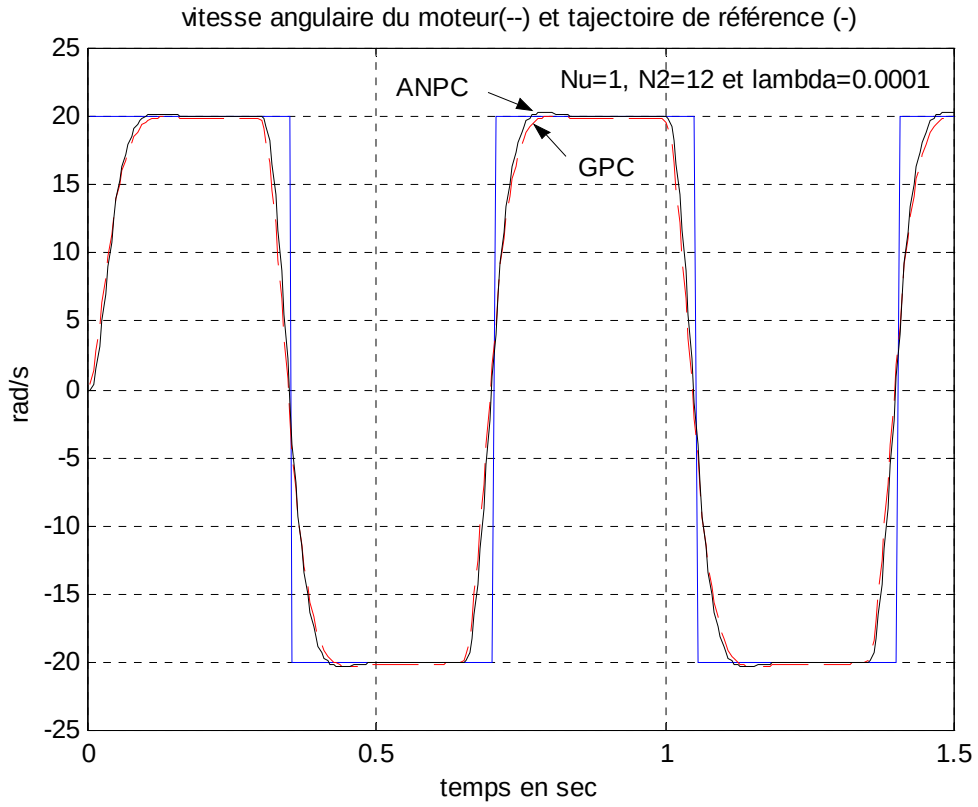


Figure 4.34.b : vitesse angulaire du moteur et trajectoire de référence

Dans le but de conclure sur la sensibilité des deux approches aux changements des paramètres du système, nous considérons la variation de la valeur de la résistance (R) de l'induit du moteur. Notons que la variation de celle-ci induit la variation des coefficients a, b et c de l'équation (4.62). Pour les mêmes valeurs des paramètres de réglage : N_1=1, N_u=1, λ=10^{-4} et N_2=8, nous avons fait la simulation pour les trois cas suivant : (i) R varie de 25 % de sa valeur nominale (figure 4.35.a), (ii) R varie de 50 % de sa valeur nominale (figure 4.35.b) et (iii) R varie de 100 % de sa valeur nominale (figure 4.35.c).

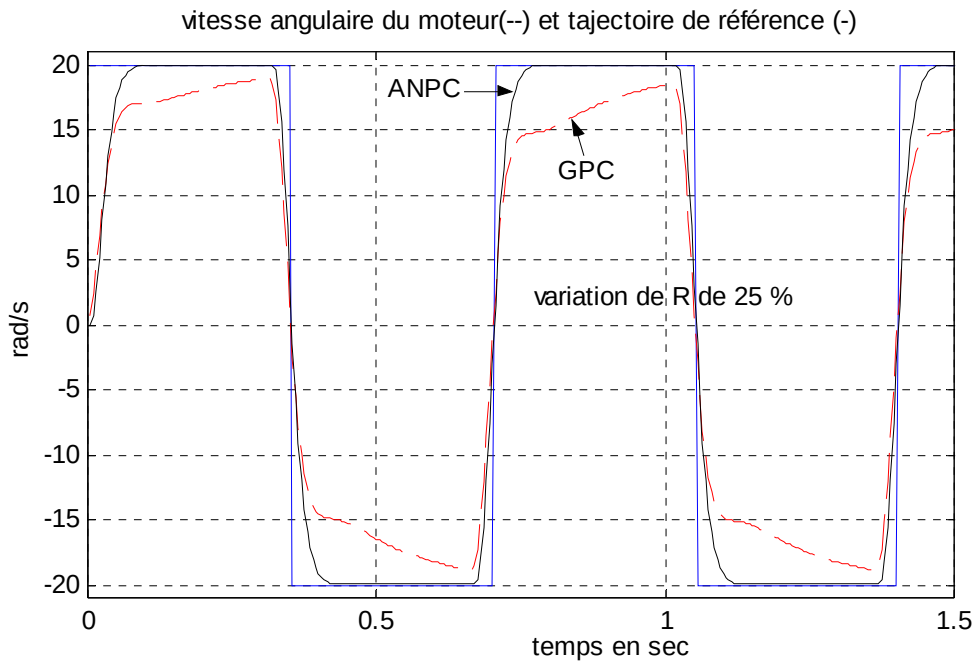


Figure 4.35.a : Variation de R de 25 % de sa valeur nominale

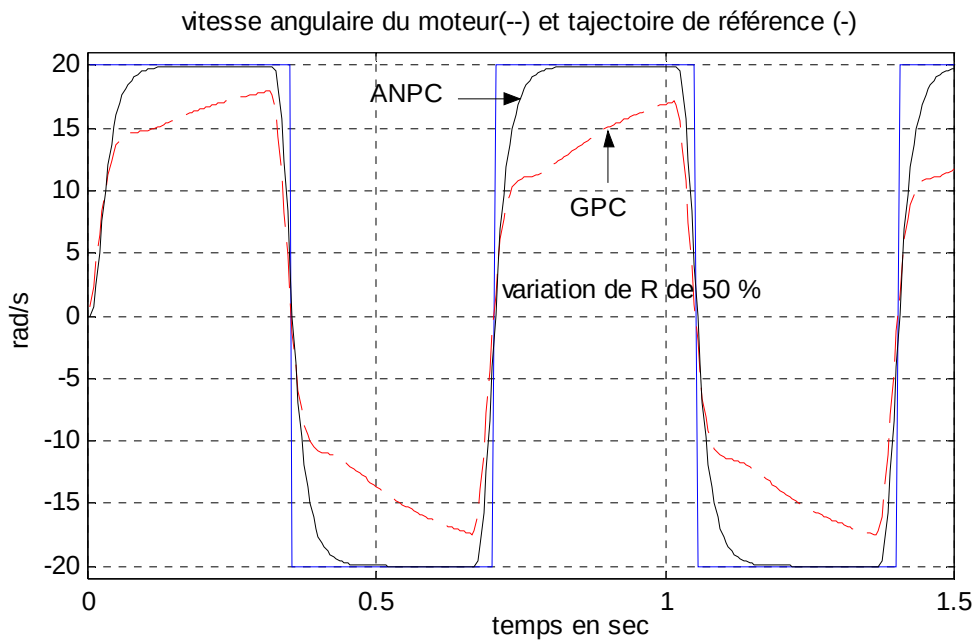


Figure 4.35.b : Variation de R de 50 % de sa valeur nominale

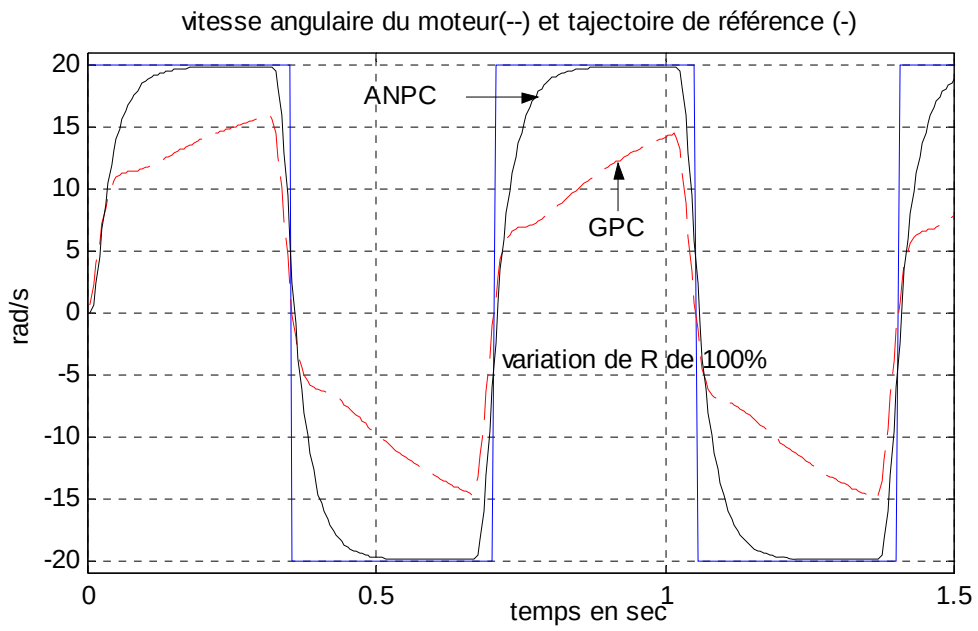


Figure 4.35.c : Variation de *R* de 100 % de sa valeur nominale

Un deuxième cas important est celui de la variation du moment d'inertie *J*, induisant la variation de tous les coefficients de l'équation 4.62. Les variations des paramètres sont donc ici plus sévères que précédemment, et sont typiques du comportement d'un axe de robot sur lequel la charge serait modifiée (changement d'outil par exemple). Pour les mêmes valeurs des paramètres de réglage données précédemment, nous avons considéré deux cas : (i) une variation de 25 % de *J* (figure 4.36.a) et (ii) une variation de 100 % de *J* (figure 4.36.b).

A partir de cette étude, nous pouvons conclure que notre approche est beaucoup moins sensible aux changements des paramètres du système que l'approche classique GPC. Ceci pourrait être expliqué par la capacité de généralisation des réseaux de neurones.

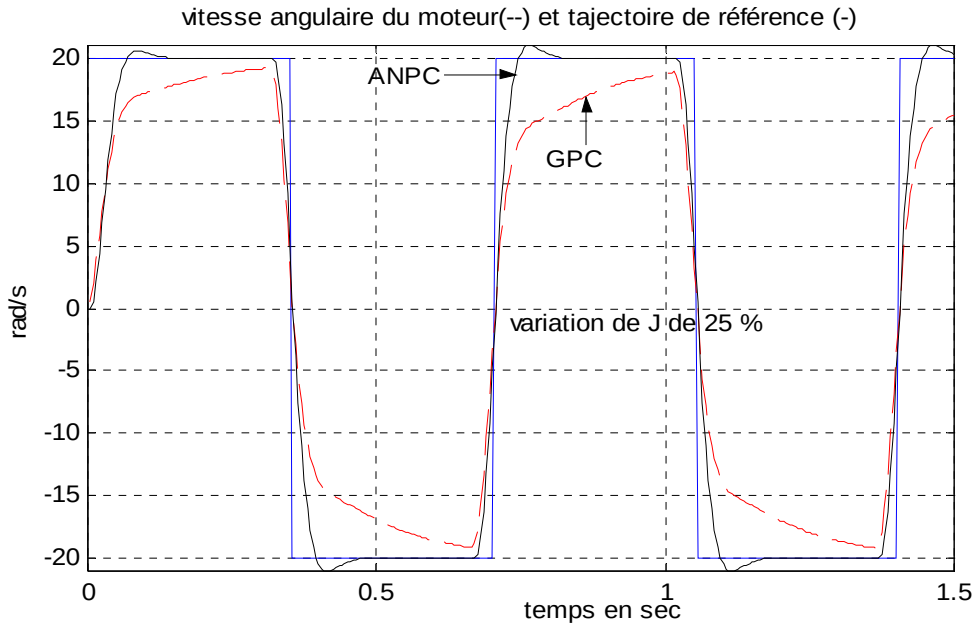


Figure 4.36.a : Variation de *J* de 25 % de sa valeur nominale

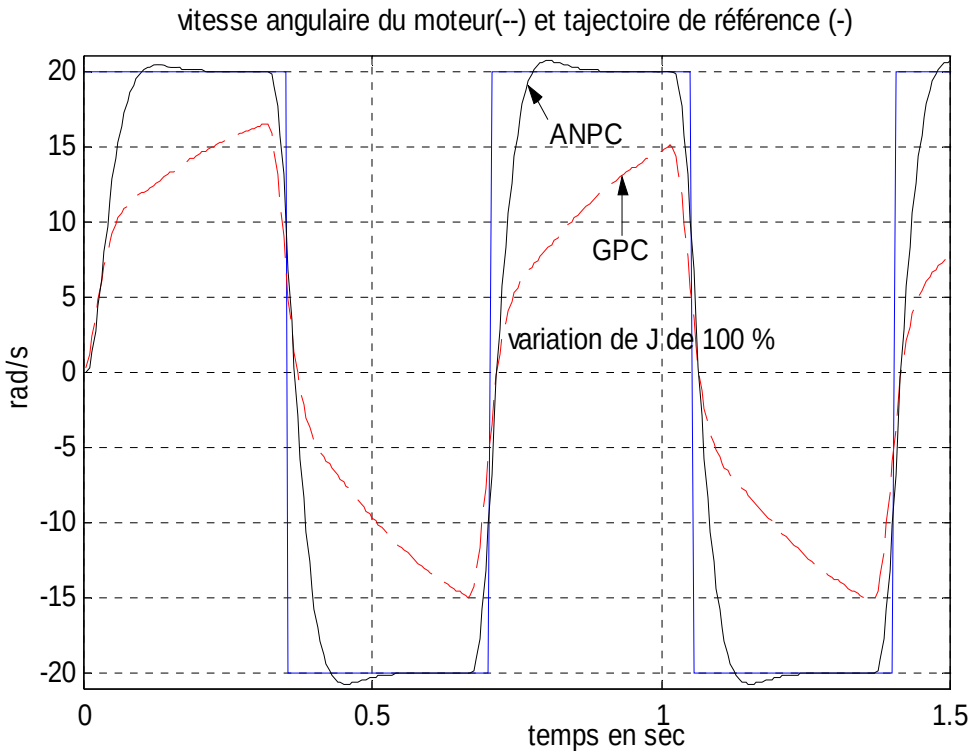


Figure 4.36.b : Variation de *J* de 100 % de sa valeur nominale

4.7 Etude comparative

Afin de conclure sur la rapidité de l'algorithme proposé (ANPC), qui est un facteur très important dans l'évaluation de tous les régulateurs, et de comparer les performances de cette approche à celles de l'approche itérative et de la commande prédictive généralisée linéaire, nous avons repris le problème de la commande du système non linéaire décrit par l'équation (2.27). La figure 4.37 présente le résultat obtenu en utilisant l'algorithme de commande basé sur la méthode de Levenberg-Marquardt (Levenberg-Marquardt based Neural Predictive Control : LMNPC). Les performances obtenues en appliquant l'algorithme ANPC sont représentées sur la figure 4.38. Notons que les valeurs utilisées des paramètres de réglage sont : $N_1=1$, $N_2=17$, $N_u=1$, $\lambda=10^{-4}$, et que la valeur quadratique moyenne de l'erreur de poursuite est égale à : 9 10^{-3} pour l'algorithme LMNPC, 8.4 10^{-3} pour l'algorithme NRNPC et à 7 10^{-3} pour l'algorithme ANPC. Sur la figure 4.39 nous avons représenté une partie de la sortie du système obtenue en utilisant les deux algorithmes (ANPC et NRPC) et la commande GPC. Une mauvaise poursuite de la trajectoire de référence, dans le cas de la GPC, est obtenue. En revanche, dans le cas des algorithmes ANPC et NRNPC, nous obtenons une bonne poursuite. Ceci justifie l'utilisation des modèles non linéaires et en particulier les réseaux de neurones.

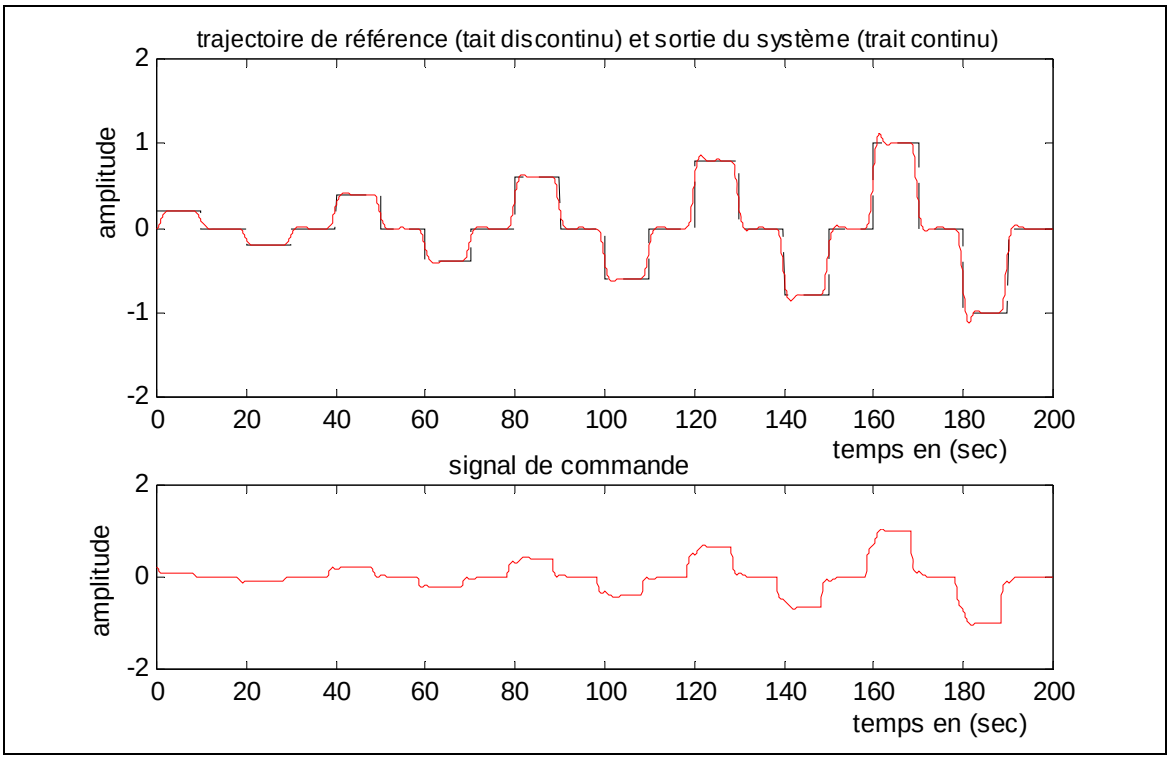


Figure 4.37 : performances obtenues dans le cas de LMNPC

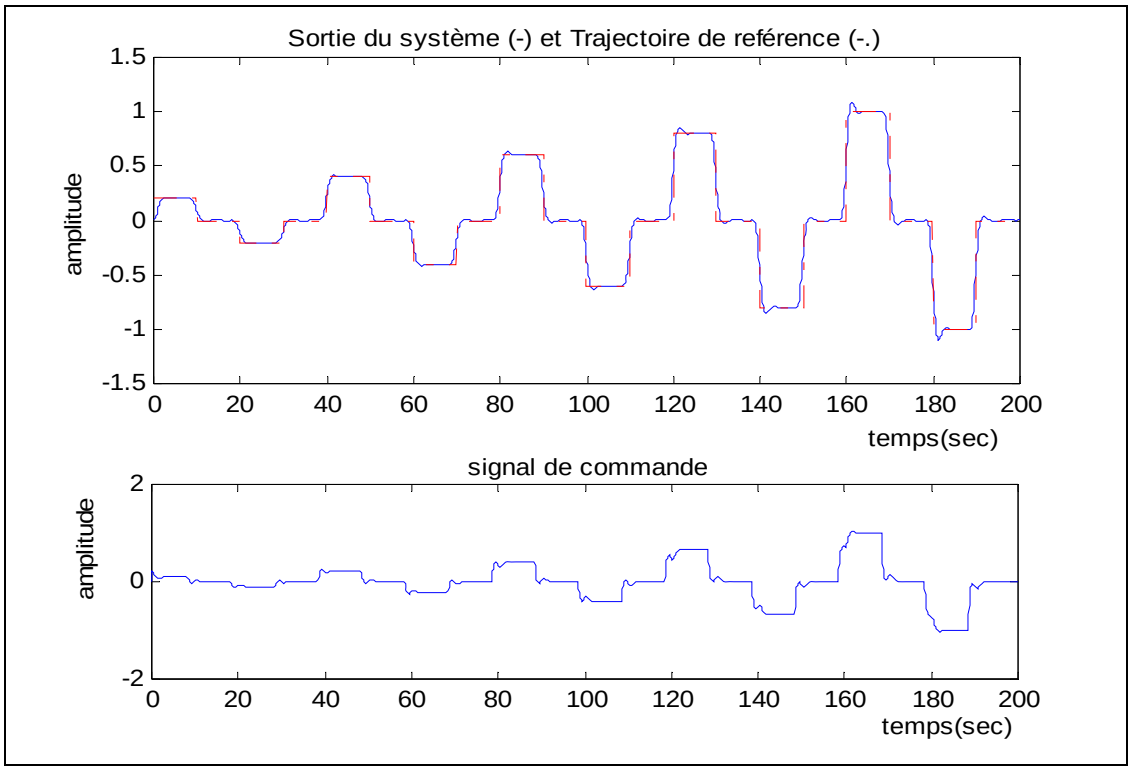


Figure 4.38 : performances obtenues dans le cas de ANPC

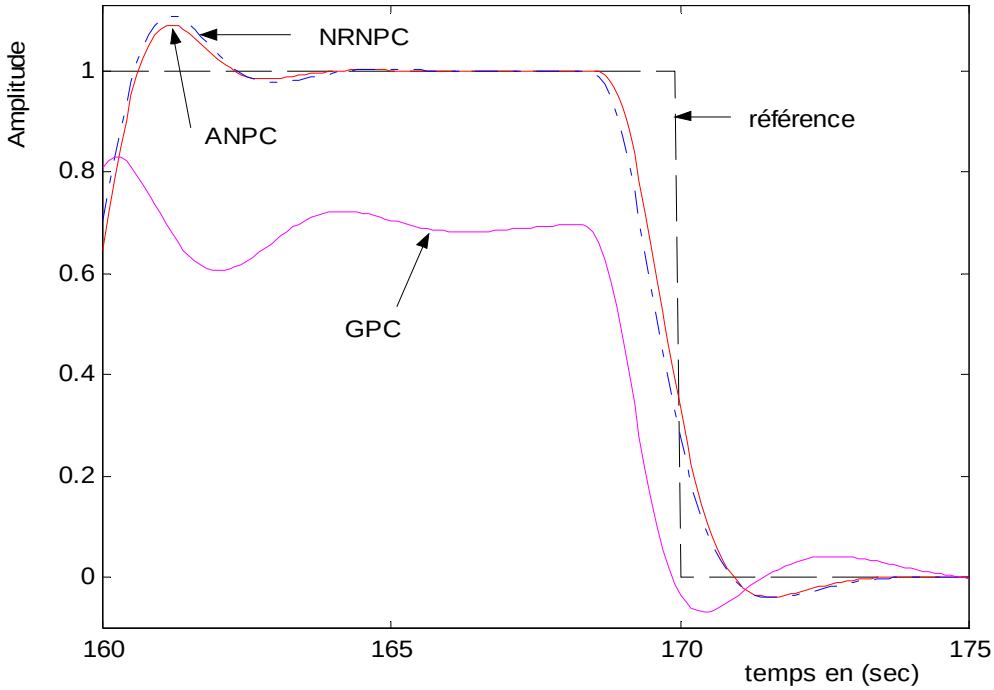


Figure 4.39 : Performances obtenues dans le cas de la commande GPC, ANPC et NRNPC

Enfin, le tableau 4.1 présente le temps nécessaire au calcul d'une valeur du signal de commande en fonction des valeurs de N_2 et N_u, la valeur de N_1 étant toujours fixée à 1. Le nombre d'itérations maximal, pour l'approche LMNPC, est fixé à 5 et celui de l'approche NRNPC est fixé à 1. La simulation a été faite sur un PC *(P. IV)* d'horloge égale à *1.7 GHz*- et les algorithmes ont été codés en utilisant le logiciel matlab. A partir de ce tableau, nous pouvons conclure que l'approche ANPC est beaucoup plus rapide que les approches NRNPC et LMNPC, qui demandent un temps de calcul assez important. L'approche ANPC peut être utilisée facilement pour la commande en temps réel des systèmes à dynamique rapide.

ANPC				NRNPC				LMNPC			
N_u \ N_2	**5**	**10**	**15**	N_u \ N_2	**5**	**10**	**15**	N_u \ N_2	**5**	**10**	**15**
1	1.32	2.531	3.718	**1**	3.679	7.304	10.922	**1**	12.211	21.133	32.43
2	1.351	2.555	3.75	**2**	7.851	15.695	23.602	**2**	14.633	33.359	50.031
3	1.367	2.57	3.773	**3**	13.609	27.14	40.672	**3**	19.601	41.773	62.929
4	1.398	2.601	3.828	**4**	21.312	41.515	62.132	**4**	25.086	62.328	88.125
5		2.632	3.843	**5**		60.57	87.945	**5**		72.148	107.20

Tableau 4.1 : temps de calcul (en milliseconde) d'une valeur de la commande

4.8 Conclusion

Dans ce chapitre, après avoir formulé le problème de la commande prédictive à modèle neuronal, nous avons exposé brièvement le principe de quelques approches déjà proposées pour mettre en évidence les motivations de ce travail. Dans le but d'avoir un algorithme de commande numériquement stable, nous avons proposé d'utiliser avec la méthode de Newton-Raphson l'algorithme de Jordan. L'efficacité de cette approche a été mise en évidence en considérant son application à la commande d'un système non linéaire à phase non minimale et à la commande d'un bras manipulateur. Ensuite, nous avons proposé une nouvelle stratégie de commande prédictive des systèmes non linéaires à base d'un modèle neuronal. Une formulation de l'algorithme de conception du contrôleur a été donnée en tenant compte d'un temps de retard et d'un horizon minimal de prédiction quelconques. L'approche proposée se caractérise par les points suivants :

- simplicité d'implantation : ne nécessitant que l'estimation des coefficients de la réponse du système à un échelon ;
- temps de calcul réduit ;
- robustesse vis-à-vis des perturbations ;
- moins sensible aux changements des paramètres du système par rapport à l'approche linéaire.

En plus, l'algorithme proposé peut être utilisé pour la commande prédictive de n'importe quel système non linéaire ; ceci se justifie par l'aptitude des réseaux de neurones à la modélisation de tels systèmes. Les performances de cette méthode sont mises en évidence en considérant plusieurs applications. Nous avons obtenu de bons résultats, en particulier la bonne poursuite de la trajectoire de référence, l'obtention d'une action de commande suffisamment lisse et la réduction du temps de calcul. Enfin, les performances des approches proposées (NRNPC et ANPC) ont été comparées à celles des approches LMNPC et GPC linéaire. Les résultats obtenus montrent l'apport de ces approches, tant du coté précision de poursuite que du coté temps de calcul réduit.

CONCLUSIONS ET PERSPECTIVES

La conception d'une loi de commande prédictive peut être divisée en deux étapes fondamentales :

- Une étape de modélisation qui consiste à construire un modèle capable de prédire, avec une bonne précision, le comportement dynamique du système à contrôler, sur un horizon fini ;
- Une étape d'optimisation non linéaire qui consiste à trouver la loi de commande minimisant une fonction de coût donnée.

Notre travail part de la problématique liée à l'élaboration d'un modèle non linéaire, qui approche le mieux possible le comportement du système, et s'étend à la recherche d'une solution rapide et efficace au problème d'optimisation non linéaire.

Motivé par la capacité d'approximation universelle des fonctions non linéaires et les propriétés de généralisation et d'adaptation que possèdent les réseaux de neurones, nous avons choisi d'utiliser un modèle neuronal pour la prédiction des sorties futures du système. En particulier, nous avons choisi les réseaux MLP statiques à cause de la simplicité de leurs algorithmes d'apprentissage. Notre travail, dans ce contexte de modélisation, a abouti au développement d'une méthode d'identification par réseaux de neurones des systèmes non linéaires sujet à des perturbations externes. Une formulation mathématique de la solution proposée a été donnée, et les deux cas de représentation des perturbations, par un modèle linéaire et par un modèle non linéaire, ont été considérés. Cette solution consiste, en augmentant la taille du vecteur des régresseurs, à donner plus d'informations au réseau de neurones sur l'histoire de la sortie et de l'entrée de commande du système. L'efficacité de la méthode a été mise en évidence par des exemples en simulation.

Bien que les réseaux de neurones constituent un outil, simple et puissant, de modélisation des systèmes non linéaires, leur utilisation dans le cadre de la commande prédictive conduit à un problème d'optimisation, non linéaire, complexe et dont la solution par les méthodes itératives est généralement coûteuse en terme du temps de calcul. Il est donc important de développer d'autres techniques qui permettent de surmonter cette contrainte. Le travail que nous avons réalisé, dans ce contexte, a permis de proposer les contributions suivantes :

Au début, nous avons considéré l'approche utilisant la méthode de Newton-Raphson. Grâce à la convergence quadratique de la méthode de Newton-Raphson, il est possible d'obtenir un temps de calcul relativement faible en comparaison avec les approches utilisant la méthode de Newton. Une version de cette approche, numériquement stable et qui utilise l'algorithme de Jordan au lieu de la décomposition 'LU', a été proposée. Les performances de cette approche ont été mises en évidence en considérant son application à la commande d'un système non linéaire à phase non minimale et à la commande d'un bras manipulateur. Ensuite, nous avons proposé une nouvelle stratégie basée sur le principe de décomposition de la réponse totale du système. Une formulation de l'algorithme de conception du contrôleur a été donnée en tenant compte d'un temps de retard et d'un horizon minimal de prédiction quelconques. L'approche proposée, qui se caractérise par les points suivants, permet de calculer analytiquement la loi de commande :

- simplicité d'implantation : ne nécessitant que l'estimation des coefficients de la réponse du système à un échelon ;
- temps de calcul réduit ;
- robustesse vis-à-vis des perturbations ;
- moins sensible aux changements des paramètres par rapport à l'approche linéaire.

En plus, l'algorithme proposé peut être utilisé pour la commande prédictive de n'importe quel système non linéaire ; ceci se justifie par l'aptitude des réseaux de neurones à la modélisation de tels systèmes. Les performances de cette méthode sont mises en évidence en considérant plusieurs applications (commande d'un réacteur continu parfaitement mélangé, commande d'un bras manipulateurs et commande d'un moteur à courant continu). Nous avons obtenu des résultats satisfaisants, en particulier la bonne poursuite de la trajectoire de référence, l'obtention d'une action de commande suffisamment lisse et la réduction du temps de calcul. Enfin, les performances des approches proposées (NRNPC et ANPC) ont été comparées à celles des approches LMNPC et GPC linéaire, les résultats obtenus montrent l'apport de ces approches, tant du coté précision de poursuite que du coté temps de calcul réduit.

Le travail présenté est développé pour un système mono-variable. Son application à des systèmes multi-variables est possible et elle est immédiate.

Plusieurs perspectives à ce travail peuvent être envisagées. Une première démarche consiste à examiner l'extension de la méthode proposée au cas des systèmes à paramètres variables en introduisant un mécanisme d'adaptation adéquat. En effet, l'ajustement automatique, en ligne et en temps réel, des paramètres du contrôleur permet de maintenir un même niveau de performances. Etant donné que les performances de la commande prédictive dépendent essentiellement du modèle de prédiction, il est raisonnable de penser à ajuster en ligne les poids synaptique du réseau de neurones (le modèle de prédiction), en utilisant un algorithme d'apprentissage rapide, de telle sorte qu'il s'adapte au changements des paramètres du système. Afin de minimiser la charge de calcul pour les applications effectuées en ligne, il faut penser à introduire un test de mise à jour conditionnel permettant de n'ajuster les poids du réseau que si les variations des paramètres du système sont suffisamment significatives.

L'une des raisons que fait de la méthodologie prédictive une technique de commande puissante et très utilisée dans le domaine industriel, est la prise en compte des contraintes pour la mise au point de la loi de commande. Il est donc recommandable de considérer le problème de la commande prédictive à modèle neuronal avec contraintes. La prise en compte des contraintes et l'utilisation d'un modèle neuronal posent un véritable problème d'optimisation non linéaire et non convexe. Un travail devrait donc se faire dans ce contexte pour développer des méthodes permettant de trouver une solution rapide et efficace à ce problème. Les méthodes d'optimisation par point intérieur (interior point methods) pourrait être considérées.

Une autre perspective qui mérite d'être étudiée consiste à envisager le problème de stabilité. En effet, Il n'existe aucune théorie générale permettant de démontrer d'une façon systématique, la stabilité des systèmes non linéaires. Toutes les méthodes développées sont basées sur le principe de l'hyperstabilité de Lyapunov pour lequel le choix de la fonction de Laypunov n'est pas toujours évident. Ce problème de stabilité devient beaucoup plus complexe pour les systèmes de commande à base des réseaux de neurones.

Enfin, il apparaît nécessaire de sortir du cadre de la simulation, au sein duquel on a pu montrer l'intérêt du travail effectué, pour tester de façon expérimentale les bénéfices réellement attendus.

Références bibliographiques

[1] P. J. W Melsa, "Neural Networks: A conceptual overview", Technical Report TRC - 89 - 08, Tellabs Research Center, Mishawaka, Aug. 1989

[2] K. S. Narendra, *Neural networks for identification and control*, Center for systems science, Yale University

[3] E. Davalo, P. Naim, *Des réseaux de neurones*, 2 ème édition, Eyrolles 1990

[4] H. Abdi, *Les réseaux de neurones*, Presses universitaire de Grenoble, 1994

[5] D. R. Hush and B. G. Horne, "Progress in supervised Neural Networks", *IEEE Signal Processing Magazine*, vol. 10, no. 1,pp 8-39, Jan. 1993.

[6] T. Kohonen, *Self organization and associative memory*, 2 eme édition, Springer Verlag Berlin Heidelberg, 1988.

[7] P. K. Simpson, *Artificial neural systems*, Pergamon press, Elmsford, New York, 1989

[8] Norgaard M., Ravn O., Poulsen N. K., Hansen L. K., *Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner's Handbook*, Springer-Verlag, London Limited 2000, Great Britain, 2nd printing, 2001.

[9] J. W. Hines, *Fuzzy and Neural Approaches in Engineering*, John Wiley & Sons, Inc., 1997

[10] Jang J. – S. R., Sun C. –T., Mizutani E., *Neuro-Fuzzy and Soft Computing*, Prentice Hall Upper Saddle River, NJ 07458, 1997.

[11] B. Widrow and M. A. Lehr, "30 Years of adaptive neural Networks : Perceptron, Madaline and Backpropagation", Proceedings of The IEEE, vol. 78, no. 9, pp 1415-1441, Sep. 1990

[12] S. Z. Qin, H. T. Su and T. J. MacAvoy, "Comparaison of four neural net learning methods for dynamic system identification", IEEE Trans. Neural Networks, vol. 3 no. 1,pp 122-130, Jan. 1993

[13] Hornik K., Stinchcombe M., White H., "Multilayer Feed forward Networks are Universal Approximators", *Neural Networks*, vol. 2, pp. 359-366, 1989

[14] White H., *Artificiel neural networks. Approximation and learning theory*, Blackwell Publishers, 1992

[15] Jacques Richalet, *Pratique de la commande predictive*, Hermes, Paris 1993

[16] D. W. Clarke, C., Mohtadi, P.S. Tuffs, "Generalized predictive control – part I. The basic algorithm; Part II. Extention and interpretations", *Automatica* 23(2), 137-160, 1987

[17] Patrick Boucher, Didier Dumur, *la commande prédictive*, technip, Paris, 1996

[18] R. B. Bitmead, M. Givers et V. Wertz, *Adaptive optimal control. The thinking Man's GPC*, Prentice Hall. Englewood Cliffs, N. J. 1990

[19] Eduardo F. Camacho, Carlos Bordons, *Model predictive control*, springer-verlag, London, 1999

[20] M. K., Maaziz, *Commande prédictive de systèmes non linéaires: Application à la commande de machines asynchrones*. Thèse de Doctorat, Université Paris Sud Orsay, Ecole supérieure d'électricité, sept. 2000

[21] J. M., Maciejowski, *Predictive control with constraints*, Prentice Hall, Pearson Education Limited, England, 2002

[22] K. Kara, Application des réseaux de neurones à l'identification des systèmes non linéaires, Thèse de Magister, Université de Constantine 1995.

[23] K. Kara, M. L. Hadjili, K. Benmahammed, "Modélisation neuronale des systèmes non linéaires en présence des perturbations", *Conférence internationale sur les systèmes de Télécommunications, d'Electronique Médicale et d'Automatique, CISTEMA'2003*, Tlemcen les 27- 29 sep. 2003

[24] K Kara, T. Missoum, T. Abed, K. Benmahammed, "Disturbed Nonlinear Systems Identification using Neural Networks", *Proc. of the third IEEE International Conference on Systems, signals & Devices*, Vol. 1, Sousse, Tunisia, March 21-24, 2005

[25] Eric Ranco and Peter J.Gawthrop, "Neural networks for modelling and control", Technical report: csc 97008, November 10, 1997, *Center for system and control, department of Mechanical Engineering, University of Glasgow*, U.K.

[26] S. R. Chu, R. Shoureshi and N. Tenorio, "Neural Network for system identification", *IEEE Control Systems Magazine*, vol. 10, no. 3, pp 31-35, Apr. 1990

[27] S. Chen and S. A. Billings, "Neural Networks for nonlinear system modeling and Identification", *Int. J. Contro*l, vol. 56, no. 2, pp 319-346, Aug. 1992.

[28] Jairo J. Espinosa Oviedo, Fuzzy Modeling and Control, Thèse de Doctorat, Katholieke Universiteit Leuven, Belgique, april. 2001.

[29] K. S. Narendra and K. Parthasarathy, "Identification and Control of dynamical systems using neural networks", *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp 4-27, Mar. 1990

[30] K. S. Narendra and K. Parthasarathy, "Gradient methods for optimization of dynamical systems containing neural networks", *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp 252-262, 1991

[31] P. J. Werbos, "Backpropagation through time: What it does and how to do it", *Proceedings of IEEE*, vol. 78, no. 10, pp 1550-1560, Oct. 1990

[32] P. Van der Smagt, "Minimization methods for training feed-forward neural networks", *Neural Networks* 7 (1), pp. 1-11, 1994

[33] Billings S. A., Jamaluddin H. B. and Chen S., "Properties of neural networks with applications to modeling non linear dynamical systems", *International journal of control*, 55 (1), 193-224, 1992

[34] Billings S. A. and Zhu Q. M., "Nonlinear model validation using correlation tests", *International journal of control*, 60(6),1107-1120, 1994

[35] Billings S. A. and Voon W. S. F., "Correlation based model validity tests for non linear models", *International journal of control*, 44(1),235-244, 1986

[36] Sorensen P. H., Norgaard M., Ravn O., Poulsen N. K., "Implementation of neural network based non-linear predictive control", *Neurocomputing*, vol. 28, 1999, pp. 37-51

[37] Soloway D., Haley P. J., "Neural Generalized Predictive Control: A Newton-Raphson Implementation", *Proceedings of the 1996 IEEE International Symposium on intelligent Control*, pp. 277-282, 1996

[38] Botto M. A., Van Den Boom Ton J. J., Krijgsman A., José Sà Da Costa, " Predictive control based on neural network models with I/O linearization", *Int. Journal of Control*, Vol. 72 n°. 17, 1538-1554, 1999

[39] Botto M. A., Costa J. S. D., "A comparison of nonlinear predictive control techniques using neural network models", *Journal of Systems Architecture*, vol. 44, 1998, p. 579-616

[40] Liu G. P., Kadirkamanathan V., Billings S. A., "Predictive control for non-linear systems using neural networks", *Int. Journal of Control*, Vol. 71, n° 6, p. 1119-1132, 1998

[41] K. Kara, T. Abed, K. Benmahammed, "Commande prédictive à modèle neuronal en utilisant l'algorithme de Newton-Raphson", *quatrième conférence sur le génie électrique (CGE'04),12-13* avril 2005, EMP Bordj El Bahri, Alger

[42] K. Kara, M. L. Hadjili, K. Benmahammed, " Contrôle Prédictive par Réseaux de Neurones-Application au Contrôle d'un Réacteur Continu Parfaitement Mélangé", *Conférence magrébine en génie électrique, CMGE'04*, 11 – 12 /04/2004, Constantine, Algérie

[43] K. Kara, M. Hadjili, V. Wertz, K. Benmahammed, "Commande prédictive à modèle neuronal", *Journal Européen des Systèmes Automatisés*, vol. 38/1-2, pp. 125-143, 2004

[44] K. Kara, T. Abed, K.Benmahammed, V. Wertz, "Unconstrained Neuronal Predictive Control", *in the third IEEE International Conference on Systems, signals & Devices*, Vol. 1, March 21-24, 2005 – Sousse, Tunisia

[44] C. Kambhampati, J. D. Mason, K. Warwick, "A stable one-ahead predictive control of non linear systems" , *Automatica* 36 (2000), pp. 485-495.

[45] M. Nørgaard, P. H. Sørensen, N. K. Poulsen, O. Ravn, L. K. Hansen, "Intelligent Predictive control of nonlinear processes using neural networks", *Proceedings of the 1996 IEEE International Symposium on intelligent Control*, pp. 301-306, 1996

[46] Hunt K., Sbarbaro D., Zbikowski R., Gawthrop P., " Neural networks for control system-A Survey", *Automatica*, vol. 28, n°6, 1992, pp. 1083-1112

[47] K. S. Narendra, S. Mukhopadhyay "Intelligent Control Using Neural Networks", *IEEE Control Systems Mag*. pp. 11-18, Apr. 1992

[48] Rogelio Lozano, Damia Taoutaou, *Commande adaptative et applications*, Hermès science Europe, 2001

[49] Etienne Dombre, Wisama Khalil, *Modélisation et commande des robots*, Hermès, Paris, 1988

[50] E. F. Camacho, M. Berenguel, "Application of generalized predictive control to a solar power plant", *Proc. of the third IEEE conference on control applications*, Glasgow, U.K., 1994, pp. 1657-1662

[51] R. Chauprade, *Electronique de puissance, Commande des moteurs à courant continu*, Paris, 1988

[52] Jean – Michel Renders, *Algorithmes génétiques et réseaux de neurones application à la commande de processus*, Hermès, Paris, USA 1995

[53] Stamatios V. Kartalopolos, *Understanding Neural Networks and Fuzzy Logic; basic concepts and applications*, IEEE press, 1996

[54] Madan M. Gupta, Dandina H. Rao Khalil, *Neuro-Control Systems: Theory and applications*, IEEE press, USA 1994

[55] Kuperstein M., "Neural network model for adaptive hande-eye coordination for single postures", *Science*, vol. 239, pp. 1308-1311, 1988

[56] Levin E., Gwirtzman R., Inbar G., "Neural architecture for adaptive system modeling an control", *Neural Networks*, vol. 4, pp. 185-191, 1991

[57] Iiguni Y., Sakai H., Tokumaru H., "A nonlinear regulator design in presence of system uncertainties using multilayer neural networks", *IEEE Trans. On Neural Networks*, vol. 2 n°4, pp. 410-417, 1991

[58] Kumar S. S., Guez A., "ART-Based adaptative pole placement for neurocontrollers", *Neural Networks*, Vol.4, pp. 319-335, 1991

[59] D. W. Clark, R. Scattaloni, "Constrained Receding –horizon predictive control", *Proceedings IEE*, 138(4), pp. 347-354, july 1991

[60] J.B. Rawlings, K.R. Muske, "The stability of constrained receding horizon control", *IEEE Trans. on Automatic Control*, 38, pp. 1512-1516, 1993

[61] R.A.J. de Vries, T.J.J. Van den Boom, "Robust stability constraints for predictive control", *European Conference*, Brussels, 1997

[62] R.A.J. de Vries, T.J.J. Van den Boom, "Constrained Robust Predictive Control", *European Conference*, Rome Italy 1995