République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université El-Hadj Lakhdar- Batna

Faculté de Technologie
Département de Génie Electrique
Filière ELECTRONIQUE

MEMOIRE DE MAGISTER

En Electronique

Option : Robotique

Présenté par

MELAKHESSOU LAKHDAR

Ingénieur d'État en Electronique Option : Contrôle
De l'Université de Batna

Thème

Contrôle et identification des systèmes non linéaires par les techniques neuronales

---

Devant le jury composé de :
Soutenue Le / /

Noureddine SLIMANE,	M.C.A	Univ. Batna	Président
Djemai ARAR,	M.C.A	Univ. Batna	Rapporteur
A/El-Krim BOUKABOU	M.C.A	Univ. Jijel	Examinateur
Ridha BENZID,	M.C.A	Univ. Batna	Examinateur

Remerciements

Ce travail a été réalisé au sein du département de Génie Electrique de l'université de Batna.

Mes sincères remerciements à Monsieur Djemai ARAR, Maître de Conférences à *l'Université de Batna d'avoir pris le temps de juger ce travail et de m'avoir fait l'honneur d'être rapporteur de mon mémoire de magister.*

J'adresse mes sincères remerciements à Monsieur Noureddine SLIMANE, Maître de Conférences à l'Université de Batna et responsable de la formation, d'avoir accepté d'examiner ce travail et de présider le jury.

Toute ma profonde gratitude va également à Monsieur A.El Krim BOUKABOU, Maître de Conférences à l'université de Jijel, à Monsieur Ridha BENZID, Maître de Conférences à l'Université de Batna pour l'intérêt qu'ils ont témoigné à l'égard de ce travail et d'avoir accepté de participer au jury de thèse.

Enfin, je tiens tout particulièrement à remercier Monsieur et Madame BENOUDJIT pour leur aide.

Je dédie ce modeste travail

A mes parents
A ma petite famille
A mes frères et sœurs
A tous mes amis

Table des matières

Table des matières

Chapitre III : Contrôle neuronal des systèmes non linéaires

Chapitre IV : Contrôle neuronal adaptatif d'un bras manipulateur à deux degrés de libertés

Introduction générale

Introduction

La théorie de contrôle fournit des outils d'analyse et de synthèse parfaitement adaptée aux systèmes linéaires. Cependant en pratique, ces méthodes ne s'avèrent pas toujours applicables à cause de non linéarité des systèmes réels et parce qu'il n'est pas toujours possible de linéariser le système à commander, de ce fait des modèles non linéaires ont été considérés.

Il n'y a pas de méthodes générales utilisables pour l'identification des systèmes non linéaires. Cela est dû à la complexité des systèmes non linéaires. Cependant, plusieurs techniques classiques ont été établies pour l'identification de certaines classes de systèmes non linéaires [1]. Parmi ces techniques, il y a celles concernant l'estimation des paramètres pour des structures particulières. [2], et celles basées sur les séries de Volterra et de Wiener. Ces techniques classiques d'identification ne sont pas applicables pour des systèmes non linéaires à structures inconnues. D'où la nécessite de voir d'autres approches.

Le terme « identification » couvre à la fois une démarche et un ensemble de techniques dont l'objet est la détermination de modèle de comportement d'un procédé physique à partir de mesures caractéristiques de son fonctionnement dynamique. Ce modèle ne cherche qu'à reproduire " au mieux" un fonctionnement dynamique dans un contexte donné, sans se préoccuper de la signification physique éventuelle des paramètres dont il dépend.

Identifier un procédé, c'est donc déterminer, à partir de mesures expérimentales caractérisant son fonctionnement dynamique, les valeurs des paramètres du modèle le plus simple possible et conduisant à un comportement jugé comparable.

L'application des techniques neuronales pour l'identification et le contrôle des systèmes non linéaires peut fournir des nouvelles solutions pour ce problème, nous parlons alors d'identification neuronale et de contrôle neuronal.

En effet, les capacités d'identification des réseaux de neurones, en particulier ceux de type multicouches, leur aptitude à la généralisation et leur adaptativité, nous conduisent aujourd'hui à étudier et développer des architectures modulaires à base de réseaux de neurones en identification., donc l'identification neuronale c'est en fin de compte la

présentation d'un système ou un processus sous la forme d'un modèle neuronal. En fait, il s'agit de construire un modèle neuronal représentant le système non linéaire, tout en ajustant ses paramètres de telle sorte que sa sortie s'approche le plus possible de celle du système inconnu.

La possibilité d'apprentissage peut réduire l'effort humain lors de la conception des contrôleurs et permet de découvrir des structures de contrôle plus efficaces que celles déjà connues. La propriété de traitement parallèle fournit aux contrôleurs à base de réseaux de neurones l'habilité d'une réponse rapide aux variations complexes de l'environnement [3].

L'objectif du présent travail est de mettre en évidence les capacités des réseaux de neurones dans l'identification et le contrôle des systèmes non linéaires.

Le premier chapitre donne un aperçu général sur les réseaux de neurones; les définitions et les concepts de base [4][5].

Le deuxième chapitre détaille le problème d'identification des systèmes non linéaires en utilisant les réseaux de neurones et donne une solution pour le problème de rejection des perturbations additives en entrée et en sortie [6][7].

Le chapitre trois traitera l'utilisation de réseaux de neurones dans le contrôle des systèmes non linéaires, Le contrôle avec modèle inverse et le contrôle adaptatif avec modèle de référence, seront plus détaillées [6][8].

Dans le quatrième chapitre nous présenterons l'identification et le contrôle neuronal adaptatif indirect avec modèle de référence d'un bras manipulateur à deux degrés de libertés [9].

Enfin, la conclusion générale présente le bilan de ce travail ainsi que les perspectives envisagées.

CHAPITRE I

Les réseaux de neurones artificiels

I.1 Introduction

Les réseaux de neurones artificiels connaissent depuis quelques années un succès croissant dans divers domaines des Sciences de l'Ingénieur. Ils mènent à élaborer une technique de traitement de données qui fera bientôt partie de la boite à outils de tout ingénieur préoccupé de tirer le maximum d'informations pertinentes des données qu'il possède: faire des prévisions, élaborer des modèles, reconnaître des formes ou des signaux, etc.

L'objectif principal de la recherche sur réseaux de neurones était d'accroître nos connaissances sur le mécanisme cérébral via l'élaboration de systèmes artificiels capables de reproduire des calculs complexes, similaires à ceux qu'effectue le cerveau humain, donc l'étude approfondie des concepts de base de ces réseaux est toujours nécessaire pour qu'on puisse aboutir à des solutions pour des applications différentes.

Dans ce chapitre, nous faisons une présentation générale des réseaux de neurones. D'abord, nous donnons un aperçu sur les éléments de base qui entrent dans leur constitution, à savoir le modèle du neurone biologique et celui du neurone formel, ainsi que la définition d'un réseau de neurone et ses propriétés. Les structures de connexions entres les neurones, et les différents types d'apprentissage sont présentées.

I.2 Modèle d'un neurone

Le neurone artificiel (ou cellule) est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones appartenant à un niveau situé en amont (on parlera de neurones "amonts"). À chacune des entrées est associé un poids w représentatif de la force de la connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones appartenant à un niveau situé en aval (on parlera de neurones "avals"). À chaque connexion est associée un poids [4]. (Figure I.1)

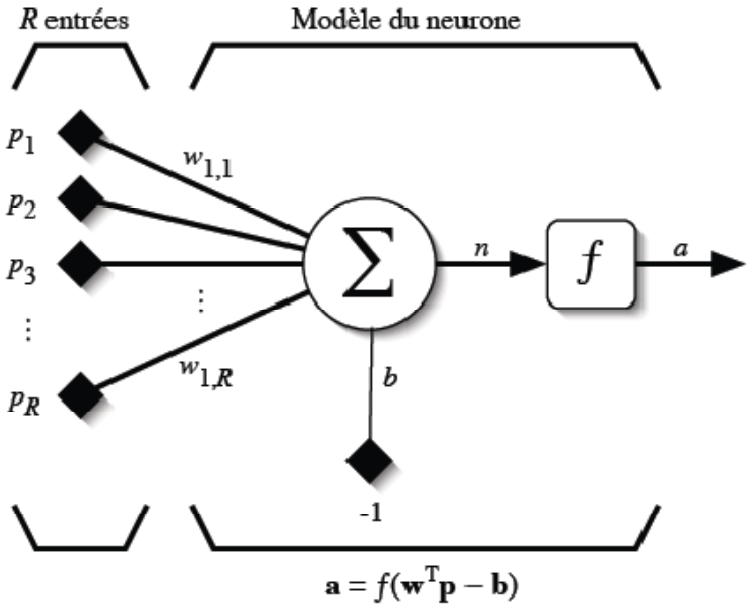


Fig. I.1 - Modèle d'un neurone artificiel [5]

Les R entrées du neurone correspondent au vecteur $p = [p_1, p_2, \dots, p_R]^T$ alors que $w = [w_{1,1}, w_{1,2}, \dots, w_{1,R}]^T$ représente le vecteur des poids du neurone. La sortie n de l'intégrateur est donnée par l'´equation suivante :

$$n = \sum_{j=1}^{R} w_{1,j} p_j - b = w_{1,1}p_1 + w_{1,2}p_2 + \cdots + w_{1,R}p_{1R} - b, \quad \text{(I.1)}$$

Que l'on peut aussi écrire sous forme matricielle :

$$n = w^T p - b \quad \text{(I.2)}$$

Cette sortie correspond à une somme pondérée des poids et des entrées moins ce qu'on nomme le biais b du neurone. Le résultat n de la somme pondérée s'appelle le niveau d'activation du neurone. Le biais b s'appelle aussi le seuil d'activation du neurone. Lorsque le niveau d'activation atteint ou dépasse le seuil b, alors l'argument de f devient positif (ou nul). Sinon, il est négatif.

On peut faire un parallèle entre ce modèle mathématique et certaines informations que l'on connait (ou que l'on croit connaitre) à propos du neurone biologique. Le neurone biologique est une cellule vivante spécialisée dans le traitement des signaux électriques. Les neurones sont reliés entre eux par des liaisons appelées axones. Ces axones vont eux-mêmes

jouer un rôle important dans le comportement logique de l'ensemble. Ces axones conduisent les signaux électriques de la sortie d'un neurone vers l'entrée (synapse) d'un autre neurone.

Les neurones font une sommation des signaux reçus en entrée et en fonction du résultat obtenu vont fournir un courant en sortie [4]. (Figure I.2)

La structure d'un neurone se compose de trois parties [4][5]:

- La somma : ou cellule d'activité nerveuse, au centre du neurone.
- L'axone : attaché au somma qui est électriquement actif, ce dernier conduit l'impulsion conduite par le neurone.
- Dendrites : électriquement passives, elles reçoivent les impulsions d'autres neurones.

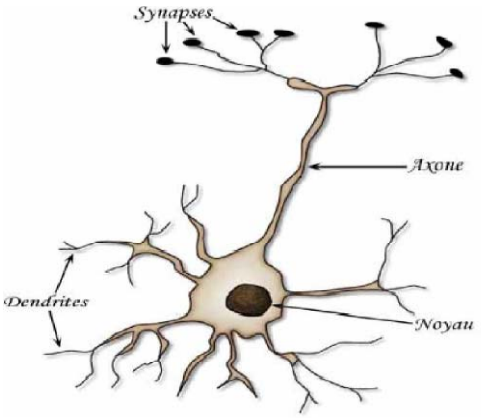


Fig. I.2 - Schéma d'un neurone biologique

Un poids d'un neurone artificiel représente donc l'efficacité d'une connexion synaptique. Un poids négatif vient inhiber une entrée, alors qu'un poids positif vient l'accentuer. Il est important de retenir que ceci est une grossière approximation d'une véritable synapse qui résulte en fait d'un processus chimique très complexe et dépendant de nombreux facteurs extérieurs encore mal connus. Il faut bien comprendre que notre neurone artificiel est un modèle pragmatique qui, comme nous le verrons plus loin, nous permettra d'accomplir des tâches intéressantes. La vraisemblance biologique de ce modèle ne nous importe peu. Ce qui compte est le résultat que ce modèle nous permettra d'atteindre [5].

Un autre facteur limitatif dans le modèle que nous nous sommes donnés concerne son caractère discret. En effet, pour pouvoir simuler un réseau de neurones, nous allons rendre le temps discret dans nos équations. Autrement dit, nous allons supposer que tous les neurones sont synchrones, c'est-à-dire qu'à chaque temps t, ils vont simultanément calculer

leurssomme pondérée et produire une sortie a(t) = f (n(t)). Dans les réseaux biologiques, tous les neurones sont en fait asynchrones[4][5].

Revenons donc à notre modèle tel que formulé par l'´equation (I.2) et ajoutons la fonction d'activation f pour obtenir la sortie du neurone :

$$a = f(n) = f(w^T p - b) \qquad \text{(I.3)}$$

En remplaçant $\boldsymbol{w^T}$ par une matrice d'une seule ligne $\boldsymbol{W = w^T}$ d'une seule ligne, on obtient une forme générale que nous adopterons tout au long de cette référence :

$$a = f(Wp - b) \qquad \text{(I.4)}$$

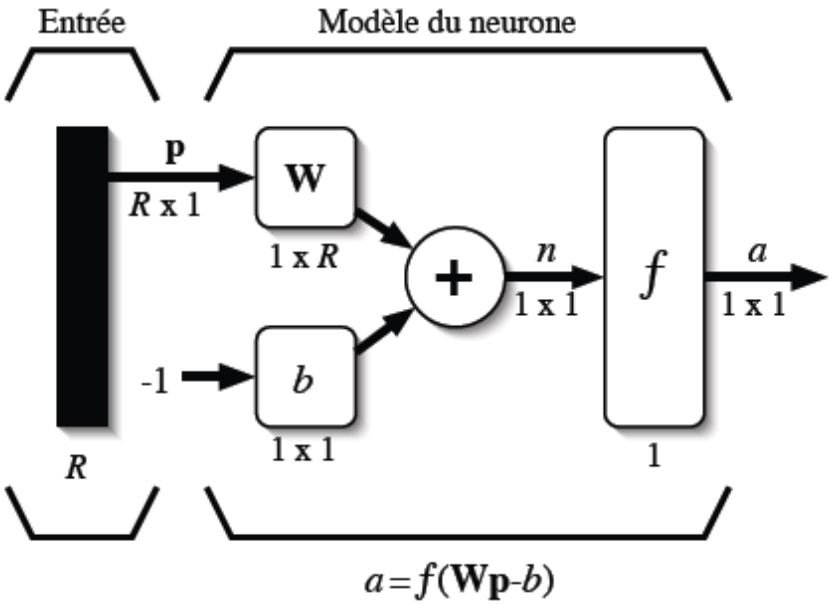


Fig. I.3 - Représentation matricielle du modèle d'un neurone artificiel[5]

L'´equation (I.4) nous amène à introduire un schéma de notre modèle plus compact que celui de la figure I.1. La figure I.3 illustre celui-ci. On y représente les R entrées comme un rectangle noir (le nombre d'entrées est indiqué sous le rectangle). De ce rectangle sort le vecteur p dont la dimension matricielle est R×1. Ce vecteur est multiplié par une matrice W qui contient les poids (synaptiques) des neurones. Dans le cas d'un neurone simple, cette matrice possède la dimension 1×R. Le résultat de la multiplication correspond au niveau d'activation qui est ensuite comparé au seuil b (un scalaire) par soustraction. Finalement, la

sortie du neurone est calculée par la fonction d'activation f. La sortie d'un neurone est toujours un scalaire.

I.3 Fonctions de transfert

Jusqu'à présent, nous n'avons pas spécifié la nature de la fonction d'activation de notre modèle. Il se trouve que plusieurs possibilités existent. Différentes fonctions de transfert pouvant être utilisées comme fonction d'activation du neurone sont énumérées au tableau I.1. Les trois les plus utilisées sont les fonctions «seuil» (en anglais «hardlim»), «linéaire» et «sigmoïde».

Comme son nom l'indique, la fonction seuil applique un seuil sur son entrée. Plus précisément, une entrée négative ne passe pas le seuil, la fonction retourne alors la valeur 0 (on peut interpréter ce 0 comme signifiant faux), alors qu'une entrée positive ou nulle dépasse le seuil, et la fonction retourne 1 (vrai). Utilisée dans le contexte d'un neurone, cette fonction est illustrée à la figure I.4.a. On remarque alors que le biais b dans l'expression de a=hardlim($\mathbf{w}^T\mathrm{p} - b$) (équation I.4) détermine l'emplacement du seuil sur l'axe $\mathbf{w}^T\mathrm{p}$, ou la fonction passe de 0 à 1.

Le tableau I.1 résume les fonctions de transfert couramment utilisées.

Nom de la fonction	Relation d'entrée/sortie	Icône	Nom Matlab
seuil	$a = 0$ si $n < 0$ $a = 1$ si $n \geq 0$		hardlim
seuil symétrique	$a = -1$ si $n < 0$ $a = 1$ si $n \geq 0$		hardlims
linéaire	$a = n$		purelin
linéaire saturée	$a = 0$ si $n < 0$ $a = n$ si $0 \leq n \leq 1$ $a = 1$ si $n > 1$		satlin
linéaire saturée symétrique	$a = -1$ si $n < -1$ $a = n$ si $-1 \leq n \leq 1$ $a = 1$ si $n > 1$		satlins
linéaire positive	$a = 0$ si $n < 0$ $a = n$ si $n \geq 0$		poslin
sigmoïde	$a = \frac{1}{1+\exp^{-n}}$		logsig
tangente hyperbolique	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
compétitive	$a = 1$ si n maximum $a = 0$ autrement		compet

TAB. I.1 – Fonctions de transfert a = f(n) [5]

La fonction linéaire est très simple, elle affecte directement son entrée à sa sortie :

$$a = n \qquad \text{(I.5)}$$

Appliquée dans le contexte d'un neurone, cette fonction est illustrée à la figure I.4.b. Dans ce cas, la sortie du neurone correspond à son niveau d'activation dont le passage à zéro se produit lorsque $w^T p = b$.

La fonction de transfert sigmoïde est quant à elle illustrée à la figure I.4.c. Son équation est donnée par :

$$a = \frac{1}{1+exp^{-n}} \qquad \text{(II.6)}$$

Elle ressemble soit à la fonction seuil, soit à la fonction linéaire, selon que l'on est loin ou prés de b, respectivement. La fonction seuil est non-linéaire car il y a une discontinuité lorsque $\boldsymbol{w}^T p = b$.De son côté, la fonction linéaire est tout à fait linéaire. Elle ne comporte aucun changement de pente. La sigmoïde est un compromis intéressant entre les deux précédentes. Notons finalement, que la fonction «tangente hyperbolique» est une version symétrique de la sigmoïde.

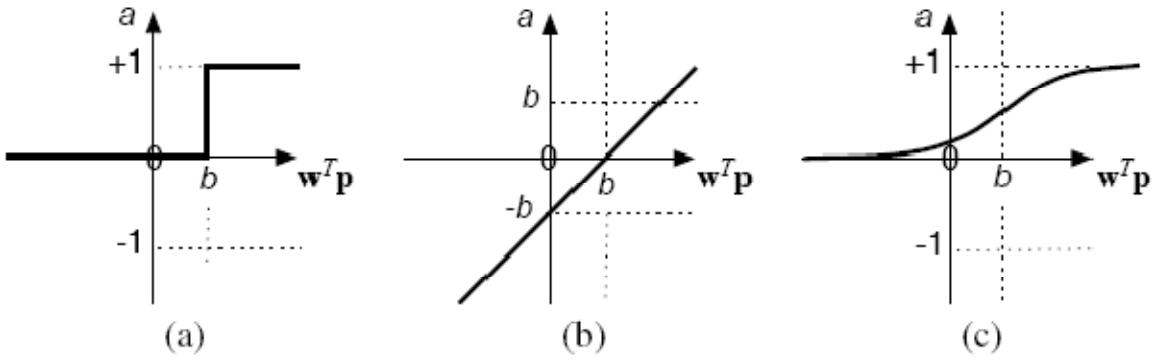


Fig. I.4 – Fonction de transfert : (a) du neurone «seuil» ; (b) du neurone «linéaire», et (c) du neurone «sigmoïde» [5]

I.4 Architecture de réseau

Un réseau de neurones est un maillage de plusieurs neurones, généralement organisé en couches. Pour construire une couche de S neurones, il s'agit simplement de les assembler comme à la figure (I.5). Les S neurones d'une même couche sont tous branchés aux R entrées. On dit alors que la couche est totalement connectée.

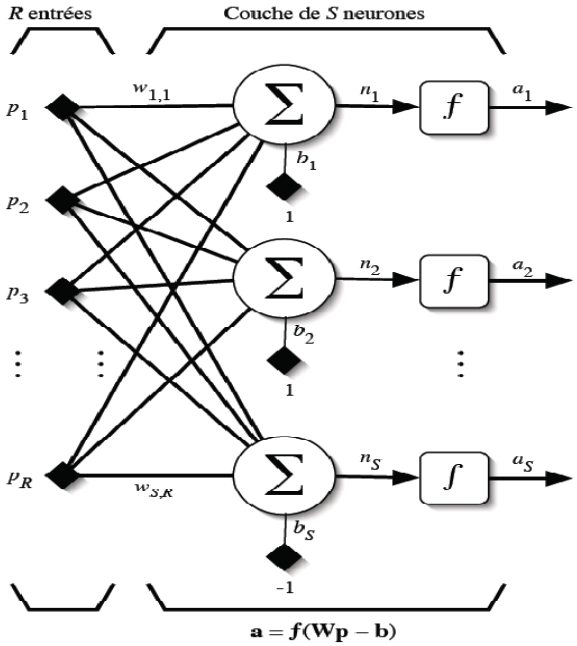


Fig. I.5 – Couche de S neurones [5]

Un poids $w_{i,j}$est associé à chacune des connexions. Nous noterons toujours le premier indice par i et le deuxième par j (jamais l'inverse). Le premier indice (rangée) désigne toujours le numéro de neurone sur la couche, alors que le deuxième indice (colonne) spécifie le numéro de l'entrée. Ainsi, $w_{i,j}$ désigne le poids de la connexion qui relie le neurone i à son entrée j. L'ensemble des poids d'une couche forme donc *une matrice* **W** *de dimension S × R :*

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix} \qquad \text{(I.7)}$$

Notez bien que S ≠ R, dans le cas général (les nombres de neurones et d'entrées sont indépendants). Si l'on considère que les S neurones forment un vecteur de neurones, alors on peut créer les vecteurs $b = [b_1 b_2 \dots b_s]^T$, $n = [n_1 n_2 \dots n_s]^T$ *et* $a = [a_1 a_2 \dots a_s]^T$. Ceci nous amène à la représentation graphique simplifiée, illustrée à la figure (I.6). On y retrouve, comme à la figure (I.3), les mêmes vecteurs et matrice. La seule différence se situe au niveau de la taille, ou plus précisément du nombre de rangées (S), de *b*, *n*, *a* et **W**.

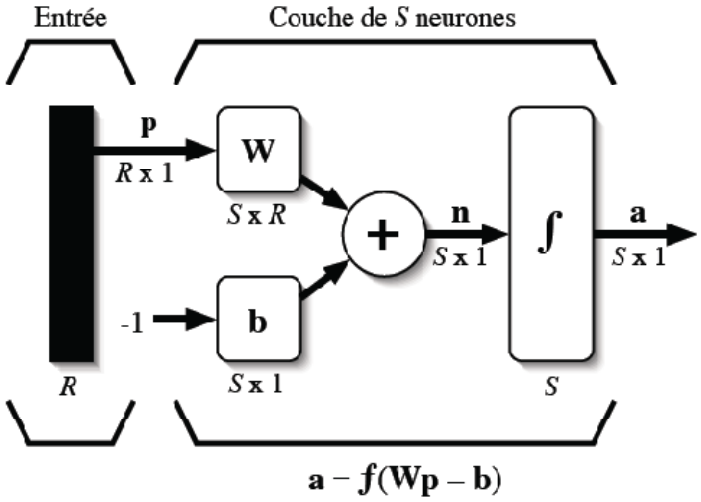


Fig. I.6 – Représentation matricielle d'une couche de S neurones[5]

Finalement, pour construire un réseau, il ne suffit plus que de combiner des couches comme à la figure (I.7). Cet exemple comporte R entrées et trois couches de neurones comptant respectivement S1, S2 et S3 neurones. Dans le cas général, de nouveau, S^1, S^2, S^3. Chaque couche possède sa propre matrice de poids W^k, où k désigne l'indice de couche. Dans le contexte des vecteurs et des matrices relatives à une couche, nous emploierons toujours un exposant pour désigner cet indice. Ainsi, les vecteurs $b^k, n^k et\ a^k$ sont aussi associés à la couche k.

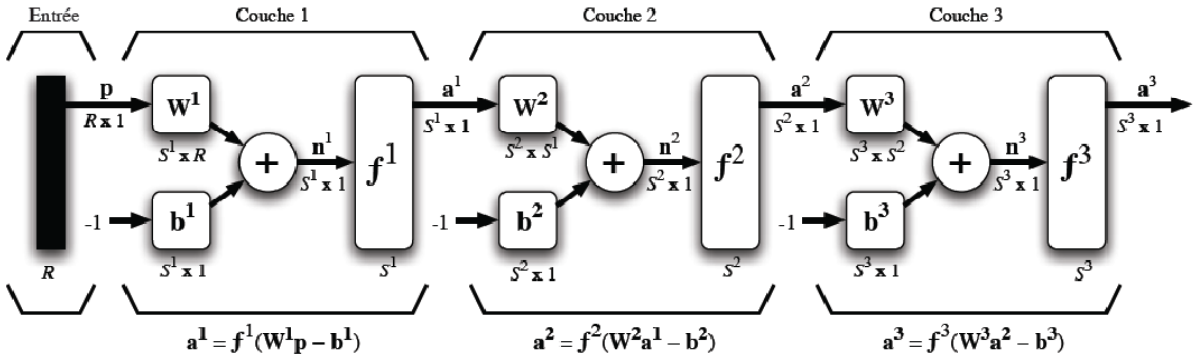


Fig. I.7 – Représentation matricielle d'un réseau de trois couches [5]

Il est important de remarquer dans cet exemple que les couches qui suivent la première ont comme entrée la sortie de la couche précédente. Ainsi, on peut enfiler autant de couches que l'on veut, du moins en théorie. Nous pouvons aussi fixer un nombre quelconque de neurones sur chaque couche. En pratique, nous verrons plus tard qu'il n'est cependant pas

souhaitable d'utiliser trop de neurones. Finalement, notez aussi que l'on peut changer de fonction de transfert d'une couche à l'autre. Ainsi, toujours dans le cas général

$$f^1 \neq f^2 \neq f^3.$$

La dernière couche est nommée «couche de sortie». Les couches qui précédent la couche de sortie sont nommées «couches cachées», le réseau de la figure (I.7) possède donc deux couches cachées et une couche de sortie.

Les réseaux multicouches sont beaucoup plus puissants que les réseaux simples à une seule couche. En utilisant deux couches (une couche cachée et une couche de sortie), à condition d'employer une fonction d'activation sigmoïde sur la couche cachée, on peut entraîner un réseau à produire une approximation de la plupart des fonctions, avec une précision arbitraire (cela peut cependant requérir un grand nombre de neurones sur la couche cachée). Sauf dans de rares cas, les réseaux de neurones artificiels exploitent deux ou trois couches.

Entraîner un réseau de neurones signifie modifier la valeur de ses poids et de ses biais pour qu'il réalise la fonction entrée/sortie désirée. Pour spécifier la structure du réseau, il faut aussi choisir le nombre de couches et le nombre de neurones dans chaque couche.

Tout d'abord, rappelons que le nombre d'entrées du réseau (R), de même que le nombre de neurones sur la couche de sortie est fixé par les spécifications du problème que l'on veut résoudre avec ce réseau. Par exemple, si la donnée du problème comporte quatre variables en entrée et qu'elle exige de produire trois variables en sortie, alors nous aurons simplement R = 4 et S^M = 3, où M correspond à l'indice de la couche de sortie (ainsi qu'au nombre de couches). Ensuite, la nature du problème peut aussi nous guider dans le choix des fonctions de transfert. Par exemple, si l'on désire produire des sorties binaires 0 ou 1, alors on choisira probablement une fonction seuil pour la couche de sortie. Il reste ensuite à choisir le nombre de couches cachées ainsi que le nombre de neurones sur ces couches, et leurs fonction de transfert. Il faudra aussi fixer les différents paramètres de l'algorithme d'apprentissage. Finalement, la figure I.8 illustre le dernier élément de construction que nous emploierons pour bâtir des réseaux dit «récurrents».

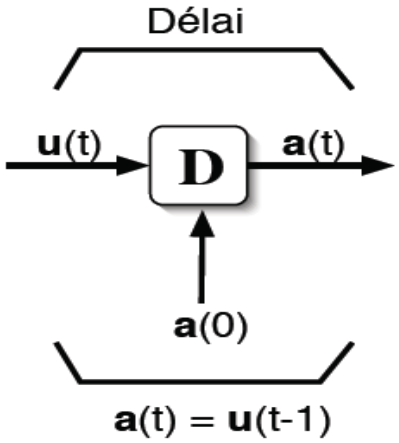


Fig. I.8 – élément de retard [5]

Il s'agit d'un registre à décalage qui permet d'introduire un retard dans une donnée que l'on veut acheminer dans un réseau. La sortie retardée*a(t)* prend la valeur de l'entrée u au temps $t - 1$. Cet élément de retard présuppose que l'on peut initialiser la sortie au temps $t = 0$ avec la valeur *a(0)*. Cette condition initiale est indiquée à la figure I.8 par une flèche qui entre par le bas de l'élément.

I.5 Le perceptron multicouche

Le perceptron multicouches (noté MLP pour Multi Layer Perceptron en anglais) est directement inspiré du raisonnement présenté au dessus. L'idée principale est de grouper des neurones dans une couche. On place ensuite bout à bout plusieurs couches et connecte complètement les neurones de deux couches adjacentes. Les entrées des neurones de la deuxième couche sont donc en fait les sorties des neurones de la première couche. Les neurones de la première couche sont reliés au monde extérieur et reçoivent tous le même vecteur d'entrée (c'est en fait l'entrée du réseau). Ils calculent alors leur sorties qui sont transmises aux neurones de la deuxième couche, etc. Les sorties des neurones de la dernière couche forment la sortie du réseau [10].

I.6 Apprentissage

Le besoin d'apprentissage se manifeste lorsque l'information à priori est incomplète, et son type dépend du degré de plénitude de cette information [5][11], Comme 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. Mathématiquement l'apprentissage est défini par [5][11]:

$$\frac{\partial W}{dt} \neq 0$$

Où West la matrice des poids

On peut distinguer trois types d'apprentissage [5][11], un apprentissage supervisé, apprentissage non supervisé et apprentissage par renforcement.

I.6.1 L'apprentissage (supervisé)

Ce type d'apprentissage nécessite que la réponse désirée du système à entrainer soit connue à priori (présence d'un maître qui fournit la réponse désiré), et il est effectue de la façon suivante [5][11]: en présente au réseau les valeurs d'entrée et on calcule sa sortie actuelle correspondante ensuite les poids sont ajustés de façon à réduire l'écart entre la réponse désirée et celle du réseau en utilisant l'erreur de sortie (la différence entre 1a réponse du réseau et celle désirée). Cette procédure est répétée jusqu'a ce qu'un critère de performance soit satisfait. Une fois 1a procédure d'apprentissage est achevée, les coefficients synaptiques prennent des valeurs optimales au regard des configurations mémorisées et le réseau peut être opérationnel.

I.6.2 Apprentissage non supervisé

Dans ce cas, la connaissance à priori de la sortie désirée n'est pas nécessaire, et la procédure d'apprentissage est basée uniquement sur les valeurs d'entrées. Le réseau s'auto organisé de façon àoptimisé une certaine fonction de coût, sans qu'on lui fournit 1a réponse désirée [5][11]. Cette propriété est appelée auto-organisation.

I.6.3 Apprentissage par renforcement

L'idée de base de l'apprentissage par renforcement est inspirée des mécanismes d'apprentissage chez les animaux. Dans ce type d'apprentissage on suppose qu'il n'existe pas de maître (superviseur) qui peut fournir la réponse correcte, mais le système à entrainer 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 du système entrainé, et les éléments qui contribuent dans La génération de cette action sont soit récompenses ou punis [5][11].

I.7 La rétro-propagation

Les algorithmes d'optimisation de fonction efficaces utilisent en général la différentielle de la fonction considérée (c'est à dire son gradient car elle est à valeurs réelles). Quand les fonctions de transfert utilisées dans les neurones sont différentiables, et quand la fonction distance est aussi différentiable, l'erreur commise par un MLP est une fonction différentiable des coefficients synaptiques du réseau. L'algorithme de rétro-propagation permet justement de calculer le gradient de cette erreur de façon efficace : le nombre d'opérations (multiplications et additions) à faire est en effet proportionnel au nombre de connexions du réseau, comme dans le cas du calcul de la sortie de celui-ci. Cet algorithme rend ainsi possible l'apprentissage d'un MLP [10].

I.8 Conclusion

Dans ce chapitre, nous avons donné en bref une description sur les réseaux de neurones artificiels.À partir du comportement du cerveau humain et d'un modèle neuronal biologique simple, les chercheurs ont arrivé à construire des modèles neuronaux artificiels plus complexes. Les réseaux de neurones présentent donc une très grande diversité,en effet, un type de réseau neuronal est défini par sa topologie, sa structure interne et son algorithme d'apprentissage. Selon la nature des connexions, plusieurs architectures des réseaux de neurones peuvent être obtenues et différents types d'apprentissage peuvent être utilisés. On a présenté l'architecture neuronale MLP par laquelle, nous avons concentré notre projet, car elle forme le modèle qui a le plus d'application [5][8], et dont les techniques d'apprentissage sont les mieux connues[5][8].

CHAPITRE II

Identification neuronale et rejection des perturbations dans les systèmes non linéaires

II.1 Introduction

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[12][13]. En effet, 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.

L'identification, c'est l'opération de détermination du modèle dynamique d'un système à partir des mesures entrées/sorties. Souvent la sortie mesurée des systèmes est entachée du bruit. Cela est dû soit à l'effet des perturbations agissant à différents endroits du procédé, soit à des bruits de mesure. Ces perturbations introduisent des erreurs dans l'identification des paramètres du modèle [14][15][16]. Dans ce chapitre, nous présentons une méthode d'identification à base de réseaux de neurones permettant d'obtenir un modèle de précision satisfaisante en présence de telles perturbations. Nous parlons alors d'identification neuronale qui permet d'approximer un système quelconque par un modèle neuronal [17][18][19].

D'abord nous résumons brièvement les différentes étapes de la conception d'un modèle en utilisant les réseaux de neurones. Nous décrivons ensuite, la procédure de modélisation en présence des perturbations.

II.2 Modélisation des processus

Les systèmes réels sont difficiles à étudier, donc on est amené à les représenter mathématiquement pour pouvoir les commander. Le problème posé est: comment obtient-on ce modèle mathématique? C'est ce qu'on appelle identification ou modélisation des processus[20].

Un processus est un objet soumis à des actions externes, à l'intérieur duquel des grandeurs interagissent, et sur lequel on peut faire des mesures.

Un modèle est une représentation de la réalité visible ou observable. Modéliser c'est «remplacer du visible compliqué avec de l'invisible simple» [21], Le but de toute modélisation de processus est de construire un modèle, c'est-à dire une représentation mathématique de son fonctionnement. Ce modèle sera utilisé pour effectuer des prédictions de la sortie du processus, ou pour l'apprentissage d'un correcteur, ou encore pour simuler le processus au sein d'un système de commande [22].

II.3 Différents type de modèle

On distingue deux types de modèles:

II.3.1 Modèle « boite noire»

Le modèle boite noire constitue la forme la plus primitive du modèle mathématique, il est réalisé uniquement à partir de données expérimentales ou observations ; il peut avoir une valeur prédictive dans un certain domaine de validité, mais il n'a aucune valeur explicative de la structure physique du système[21].

II.3.2 Modèle de connaissance

A l'opposé du modèle « boite noire», le modèle de connaissance, que l'on pourrait qualifier de « boite blanche», est issu de l'analyse des phénomènes physiques, chimiques, biologiques, etc., dont résulte la grandeur que l'on cherche à modéliser. Ces phénomènes sont mis en équation à l'aide des connaissances théoriques disponibles au moment de l'élaboration du modèle. Ces modèles sont beaucoup plus riches de signification que les modèles de représentation « boite noire » et contiennent toutes les informations utiles sur les processus. Ils sont, par contre, beaucoup plus onéreux et difficiles à obtenir [21].

II.4 Représentation des systèmes non linéaires

La représentation des systèmes dynamiques non linéaires multi-entrée multi-sortie (MIMO) en utilisant les équations d'état, est donnée par [12] [19]:

$$x(k+1) = f[x(k), u(k)]$$
$$y(k+1) = h[x(k)] \qquad \text{(II.1)}$$

Où {u(k)}, {x(k)}, {y(k)} sont des séquences à temps discret,

$$u(k) = \left[u_1(k), u_2(k), \dots, u_p(k)\right]^T$$
$$x(k) = [x_1(k), x_2(k), \dots, x_n(k)]^T$$
$$y(k) = [y_1(k), y_2(k), \dots, y_m(k)]^T \qquad \text{(II.2)}$$

Ainsi, l'équation (II.1) représente un système non linéaire d'ordre II, à p-entrées m-sorties avec u(k), x(k) et y(k) représentant respectivement les vecteurs d'entrée, d'état et de sortie à l'instant k.

Une autre représentation très utilisée, est 1a représentation entrée-sortie non linéaire (connue sous le nom de modèle non linéaire ARMA, ou NARMA modèle) C'est cette représentation qui permet aux réseaux de neurones d'être largement utilisés pour la prédiction, l'identification et le contrôle [12][19].

Soit une séquence d'entrée de longueur *l,* commençant à l'instant k et dénotée par $U_l(k)$, tel que:

$$U_l(k) = \{u(k), u(k+1), \dots, u(k+l-1)\} \qquad \text{(II.3)}$$

En utilisant l'équation (II.1) nous avons les équations d'états

$$\begin{cases} y(k) = \psi_1[x(k)] \\ y(k+1) = \psi_2[x(k), u(k)] \\ \dots\dots \qquad \dots\dots \\ y(k+n-1) = \psi_n[x(k), U_{n-1}(k)] \end{cases} \qquad \text{(II.4)}$$

Donc :

$y_n(k) = \psi[x(k), U_{n-1}(k)]$ Où $\psi: x * U_{n-1} \longrightarrow y_n$

D'un autre coté, x(k) peut être exprimée par:

$$x(k) = \phi[y_n(k), U_{n-1}(k)] \qquad \text{(II.5)}$$

La sortie à l'instant (k+n) est donnée par:

$$y(k+n) = \psi_{n+1}[x(k), U_n(k)]$$

$$y(k+n) = \psi_{n+1}[\phi[y_n(k), U_{n-1}(k)], U_n(k)]$$

$$y(k+n) = F[y(k+n-1), \dots, y(k), u(k+n-1), \dots, u(k)] \qquad \text{(II.6)}$$

Ou bien par équivalence:

$$y(k+1) = F[y(k), \dots, y(k-n+1), u(k), \dots, u(k-n+1)] \qquad \text{(II.7)}$$

F :$\Re^{2n} \longrightarrow \Re$ est une fonction non linéaire qui relie les valeurs passées des entrées et des sorties àla valeur présente de la sortie. La représentation MIMO peut être facilement dérivée, et dans ce cas u et y dans (II.7) sont des vecteurs.

II.5 Identification des systèmes

L'identification d'un système consiste principalement à sélectionner la structure du modèle.Et l'estimation des paramètres de celui-ci. En d'autres termes construire un modèle acceptable montré en figure (II.1), qui produirait une sortie $\hat{y}(k)$ approximant $y(k)$ lorsqu'il est soumis à la même entrée $u(k)$ que celle du système à identifier et ceci dans le but d'optimiser un certain critère de performance basé sur l'erreur entre la sortie du système à identifier et la sortie du modèle d'identification [23][24][25][19]. Il est à rappeler qu'il existe plusieurs techniques d'identification des systèmes non linéaires, disponibles dans la littérature [1][2][19].

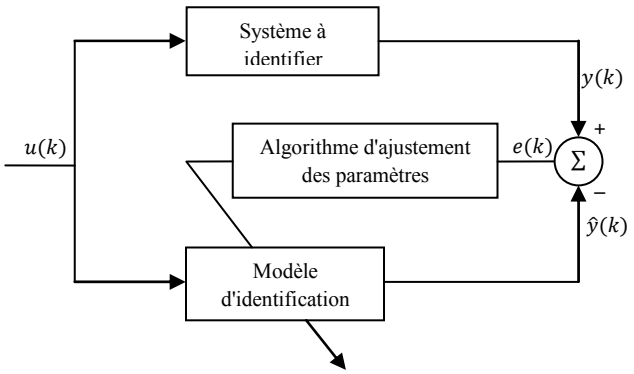


Figure II.1: Identification

L'identification se fait en général en deux étapes: l'étape qualitative (caractérisation) et l'étape quantitative (estimation des paramètres) [6][11][19].

II.5.1 Étape qualitative

Elle consiste à fixer les formes des équations qui décrivent le processus, Pour les systèmes non linéaire NARENDRA a proposé quatre classes de modèles d'identification entrées-sorties, ceux-là sont décrits par les équations suivantes[19] :

- ✓ **Modèle I :**

$$y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + g[u(k), u(k-1), \dots, u(k-m+1)] \qquad \text{(II.8)}$$

- ✓ **Modèle II :**

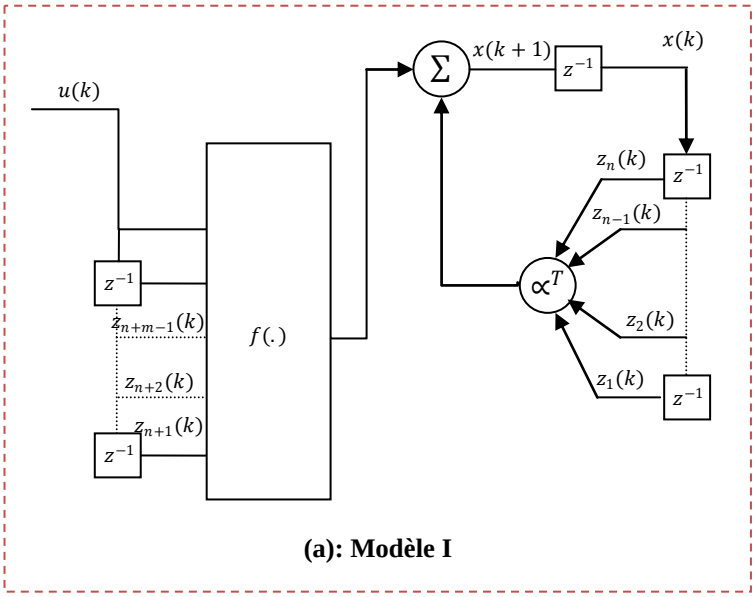
$$y(k+1) = f[y(k), y(k-1), \dots, y(k-n+1)] + \sum_{i=0}^{m-1} \beta_i u(k-i) \qquad \text{(II.9)}$$

- ✓ **Modèle III :**

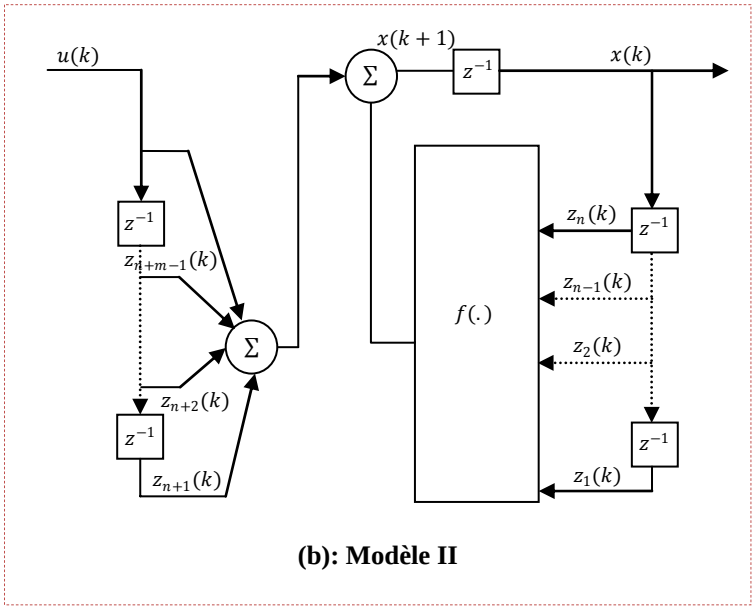
$$y(k+1) = f[y(k), y(k-1), \dots, y(k-n+1)] + g[u(k), u(k-1), \dots, u(k-m+1)] \quad \text{(II.10)}$$

✓ **Modèle IV :**

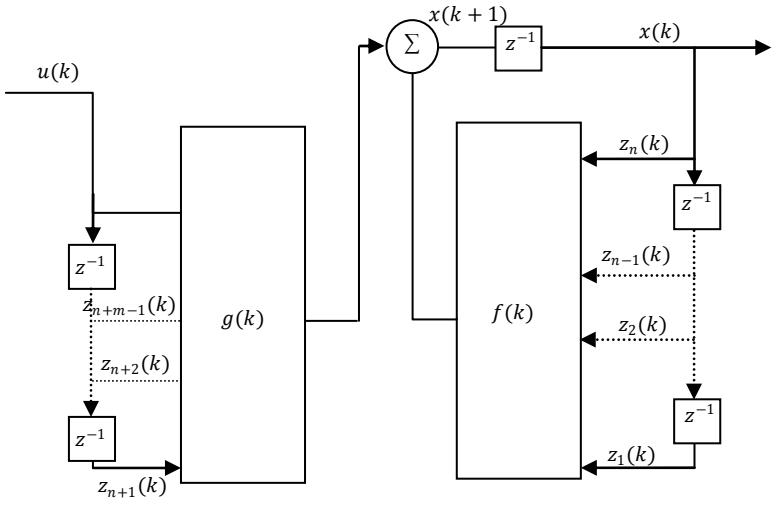
$$y(k+1) = f[y(k), y(k-1), \dots, y(k-n+1); u(k), u(k-1), \dots, u(k-m+1)] \quad \text{(II.11)}$$



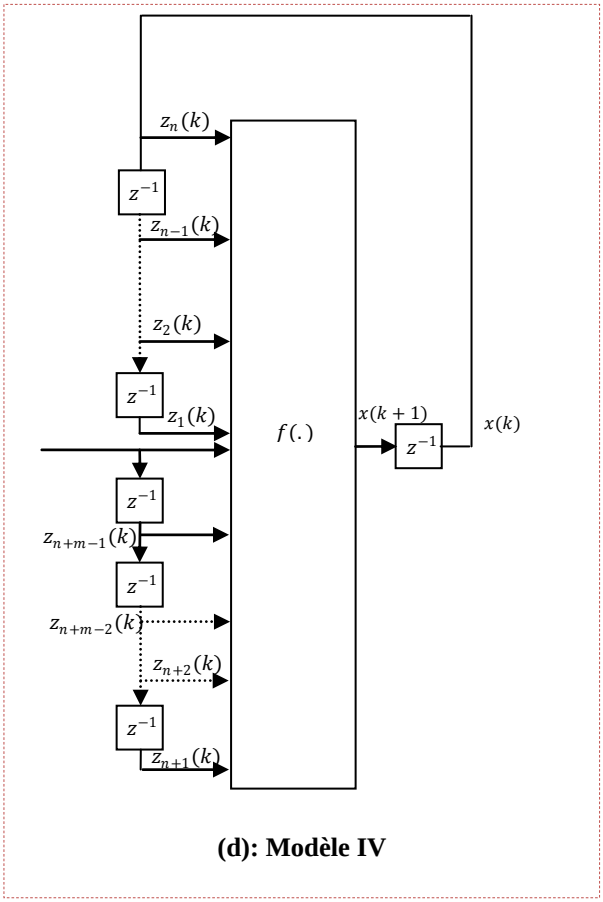
(a): Modèle I



(b): Modèle II



(c): Modèle III



(d): Modèle IV

Figure II.2: **Modèles** : Représentation des systèmes SISO

Où [u(k), y(k)] représente la paire d'entrée-sortie d'un système SISO à l'instant k, et m ≤ n. Le modèle IV est la représentation de systèmes NARMA la plus générale.

De nouveaux modèles simplifiant les algorithmes de contrôle et de ce fait applicables àune large classe de systèmes non linéaires [12][19]. Ces deux modèles nous en sont référés par modèle V et modèle VI. Ils sont décrits par :

- ✓ **Modèle V :**

$$y(k+1) = f[y(k), \dots, y(k-n+1)] + \sum_{i=0}^{n-1} g_i[y(k), \dots, y(k-n+1)]u(k-i) \quad \text{(II.12)}$$

- ✓ **Modèle VI :**

$$y(k+1) = f[y(k), \dots, y(k-n+1), u(k-1), \dots, u(k-n+1)] + g[y(k), \dots, y(k-n+1), u(k-1), \dots, u(k-n+1)]u(k) \quad \text{(II.13)}$$

II.5.2 Étape quantitative

Elle consiste à déterminer les valeurs numériques des paramètres (poids des connexions) qui interviennent dans le modèle d'identification. On doit définir un critère qui exprime quantitativement l'écart entre le système et le modèle. Ce critère devra être minimisé [26][27].

II.6 Les structures d'identification

Le principe général de l'identification est simple et consiste à placer en parallèle le modèle d'identification et le processus à identifier, comme le montre la Figure II.3. Le modèle d'identification reçoit en entrée la commande *u*(*k*) appliquée et éventuellement la sortie *y*(*k-1)* précédente du processus. Il est entraîné à produire la nouvelle sortie (ou le nouvel état) *y*(*k*) du processus, cette méthode d'identification est souvent appelée **méthode série-parallèle.** Elle est aussi souvent considérée comme plus stable car le modèle d'identification est régulièrement (recalculé) en utilisant l'état réel du processus, donné par :

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1), \dots, y(k-n+1); u(k), u(k-1), \dots, u(k-m+1)] \quad \text{(II.14)}$$

Par opposition à la méthode **parallèle** présentée au Figure II.4. Dans cette dernière méthode, le modèle d'identification ne reçoit pas en entrée la sortie réelle $y(k-1)$ du processus mais la sortie $\hat{y}(k-1)$ qu'il a lui-même prédite au pas de temps précède.

Cette approche sera d'intérêt si la boucle donnant l'état du système est remplacée par une connexion récurrente du modèle d'identification. Ce dernier est alors libre de développer sa propre représentation de l'espace des états du processus. Elle est donnée par [27][28]:

$$\hat{y}(k+1) = \hat{f}[\hat{y}(k), \hat{y}(k-1), \dots, \hat{y}(k-n+1); u(k), u(k-1), \dots, u(k-m+1)\ ] \quad \text{(II.15)}$$

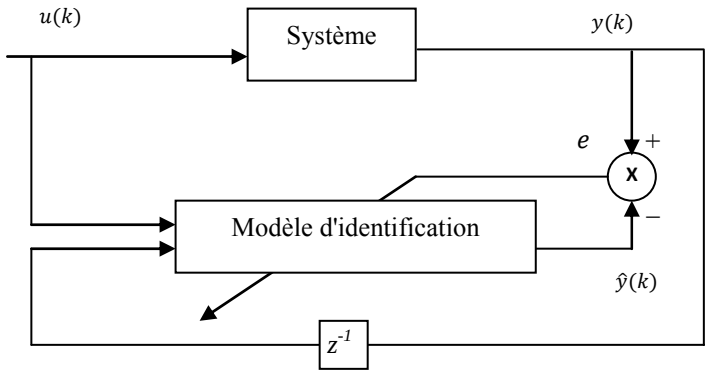


Figure II.3 : identification de processus par la méthode série-parallèle

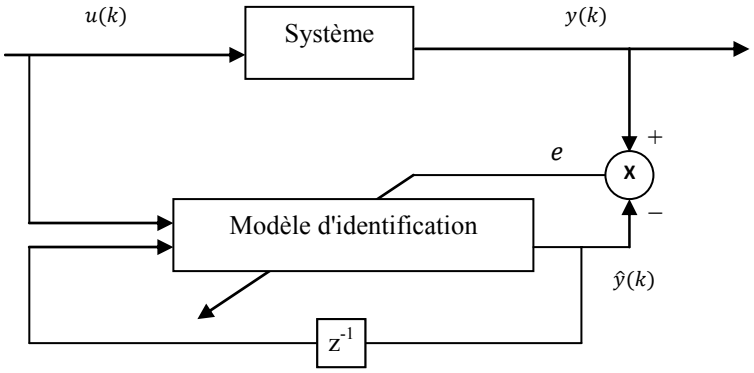


Figure II.4 : identification de processus par la méthode parallèle

II.7 Modélisation neuronale des systèmes non linéaires

L'utilisation des réseaux de neurones pour la modélisation des systèmes non linéaires découle naturellement des aptitudes de ces derniers à l'approximation et la généralisation. La détermination du modèle d'identification neuronale d'un système comporte en général les étapes suivantes:

- acquisition des données d'apprentissage et de test,
- choix de la structure du modèle,
- estimation des paramètres du modèle,

- validation du modèle identifié.

La première étape fournit les données entrées/sorties susceptibles de permettre l'extraction d'un modèle de procédé significatif, la deuxième étape consiste à choisir la structure du modèle susceptible de représenter la dynamique du système, l'architecture du réseau de neurones et ses entrées. Les réseaux multicouches statiques sont les plus utilisés à cause de la simplicité de leurs algorithmes d'apprentissage et leurs aptitudes à l'approximation et à la généralisation. Il n'existe pas de méthodes générales pour le choix du nombre de neurones sur chaque couche cachée ainsi que le nombre de ces dernières. Cependant, un réseau à une seule couche cachée est dans la majorité des cas suffisant [16][29]. En référence à la théorie des systèmes linéaires, plusieurs modèles non linéaires ont été proposés [16][30][31]:

1. **Le Modèle NFIR**: la régression est composée uniquement des entrées passées.
   $\hat{y}(k) = f[u(k-1), \dots, u(k-n)]$ (II.16)
2. **Le Modèle NARX** : dans ce cas la régression est composée de sorties et entrées passées.
   $\hat{y}(k) = f[u(k-1), \dots, u(k-n), y(k-1), y(k-m)]$ (II.17)
3. **Le Modèle NOE**: la régression est composée d'entrées et sorties estimées passées.
   $\hat{y}(k) = f[u(k-1), \dots, u(k-n), \hat{y}(k-1), \hat{y}(k-m)$ (II.18)
4. **Le Modèle NARMAX** : la régression est composée de sorties et entrées passées ainsi que d'erreurs d'estimation.
   $\hat{y}(k) = f[u(k-1), \dots, u(k-n), y(k-1), \dots$
   $, y(k-m), e(k-1), \dots, e(k-l)]$ (II.19)

Le choix d'un tel modèle dépend de l'application, des informations disponibles et de la complexité du modèle; le modèle NARX est le plus utilisé vu sa simplicité et sa structure non récursive. Le choix des régresseurs est un problème combinatoire complexe. En fait, s'il existe N régresseurs possibles, alors 2^N hypothèses de modèles doivent être considérées. Plusieurs algorithmes de sélection des régresseurs sont décrits dans les références [16][32][33].

Après avoir choisi la régression et le modèle, il faut estimer les paramètres de ce dernier. Ces paramètres sont les poids de connexions entre les neurones qui sont adaptés de telle sorte à minimiser un critère de performance; ceci est appelé dans la littérature des réseaux de neurones « Apprentissage ». Les algorithmes d'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 de la fonction coût, tels que la méthode de la rétro-propagation et la méthode utilisant l'algorithme de Levenberg-Marquardt. Les méthodes d'apprentissage utilisant les algorithmes d'optimisation qui ne nécessitent pas le calcul de la dérivée constituent la deuxième classe [16][33] L'algorithme de Levenberg-Marquardt même s'il ne fournit aucune garantie d'atteindre le minimum global, est cependant largement recommandé dans le cas d'une identification non récursive. Enfin, la dernière étape doit permettre de mettre en évidence si le modèle identifié est représentatif des comportements entrées/sorties du système. Plusieurs méthodes de validation sont données dans la référence [16][31].

II.8 Etudes et simulations

II.8.1 Cas des systèmes non linéaires SISO

Exemple (II.8.1.1)

On a un système d'ordre 2 représenté par l'équation aux différences suivantes:

$$y(k+1) = f[y(k)] + u(k) \qquad \text{(II.19)}$$

Où la fonction inconnue f est donnée par:

$$f[y(k)] = \frac{y(k)}{1+y^2(k)} \qquad \text{(II.20)}$$

Le modèle d'identification neuronal est décrit par :

$$\hat{y}(k+1) = \hat{f}[y(k)] + u(k) \qquad \text{(II.21)}$$

Où l'entrée du système et du modèle est donné par :

$$u(k) = \sin(2\pi k/25) + \sin(2\pi k/50) \qquad \text{(II.22)}$$

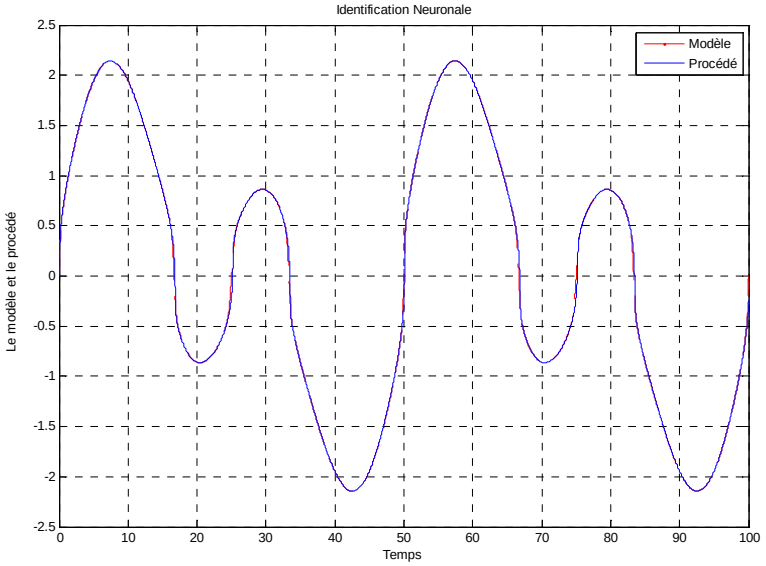


Figure II.5 : Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle

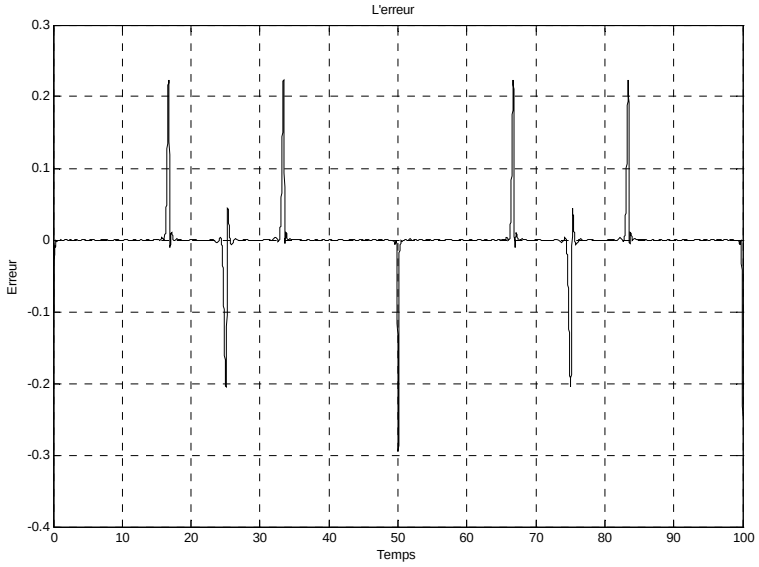


Figure II.6: Erreur d'identification

Exemple (II.8.1.2)

Soit un système d'ordre 2 décrit par l'équation aux différences suivante

$$y(k+1) = f[y(k), y(k-1)] + u(k) \qquad \text{(II.23)}$$

Où f est donnée par :

$$f[y(k), y(k-1)] = \frac{y(k)y(k-1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)} \qquad \text{(II.24)}$$

Le modèle d'identification neuronal est ainsi donné décrit par :

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1)] + u(k) \qquad \text{(II.24)}$$

L'entrée du système est donné par :

$$u(k) = \sin(2\pi k/25) \qquad \text{(II.24)}$$

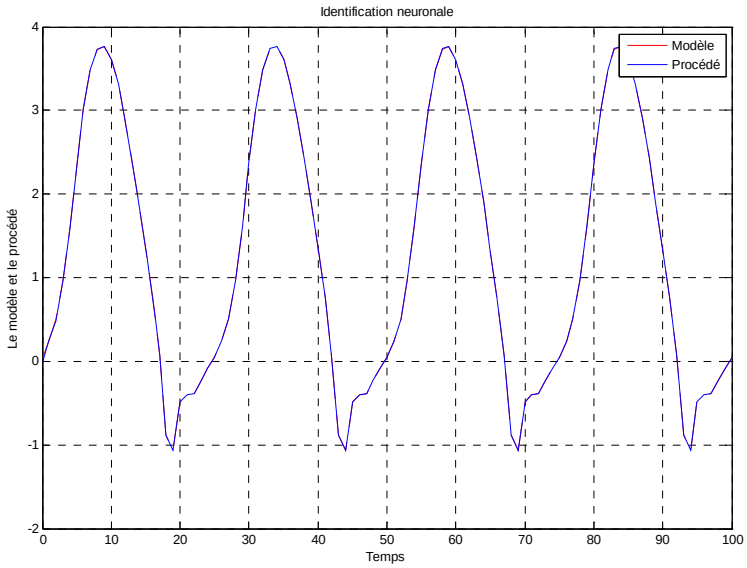


Figure II.7: Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle

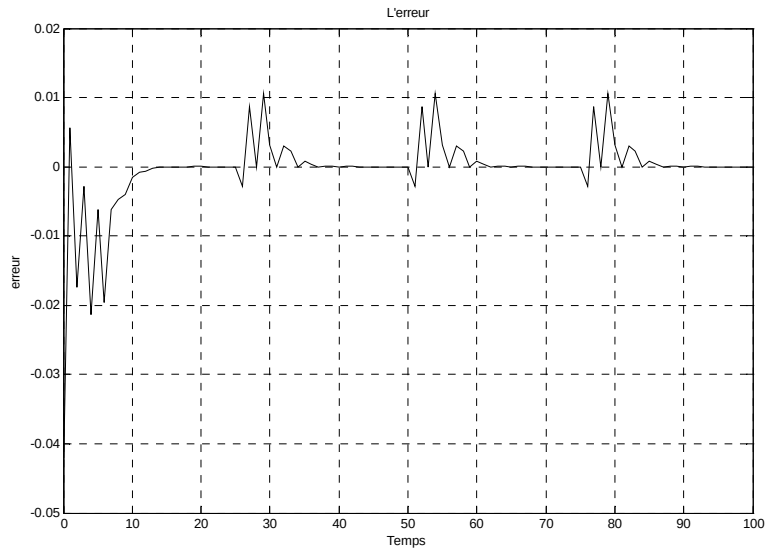


Figure II.8: Erreur d'identification

II.8.2 Cas des systèmes non linéaires MIMO

La plupart des systèmes pratiques sont de nature multivariable dont la représentation, l'identification et le contrôle sont plus complexes que ceux du type SISO.

Dans le cas de la représentation d'état des systèmes dynamiques non linéaires MIMO le u et y dans l'équation (2.7) qui étaient des scalaires deviennent des vecteurs. Ainsi un système non linéaire MIMO est donnée par:

$$y(k+1) = f[y(k), \dots, y(k-n+1); u(k), \dots, u(k-n+1)] \quad \text{(II.25)}$$

Avec :

$$y(k) = [y_1(k), y_2(k), \dots y_m(k)]^T$$
$$u(k) = [u_1(k), u_2(k), \dots u_p(k)]^T$$

Exemple (II.8.2)

On a choisi système MIMO décrit par l'équation aux différences :

$$\begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} \frac{y_1(k)}{1+{y_2}^2(k)} \\ \frac{y_1(k)y_2(k)}{1+{y_2}^2(k)} \end{pmatrix} + \begin{pmatrix} u_1(k) \\ u_2(k) \end{pmatrix} \quad \text{(II.26)}$$

Le modèle d'identification neuronal est ainsi donné décrit par :

$$\begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} \hat{f}^1[y_1(k), y_2(k)] \\ \hat{f}^2[y_1(k), y_2(k)] \end{pmatrix} + \begin{pmatrix} u_1(k) \\ u_2(k) \end{pmatrix} \quad \text{(II.27)}$$

L'entrée du système est donné par :

$$[u_1(k), u_2(k)] = [\sin(2\pi k/25), \cos(2\pi k/25)] \quad \text{(II.28)}$$

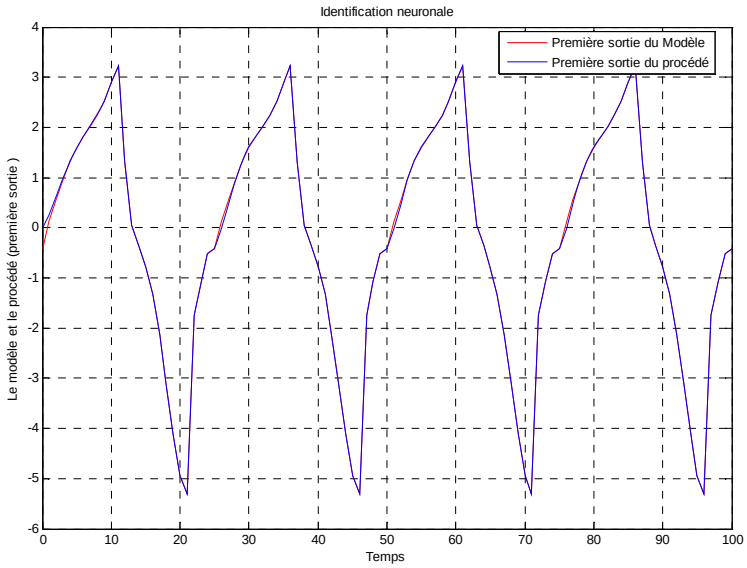


Figure II.9: Sorties, $y_1(-)$ du procédé et $\hat{y}_1(--)$ du modèle

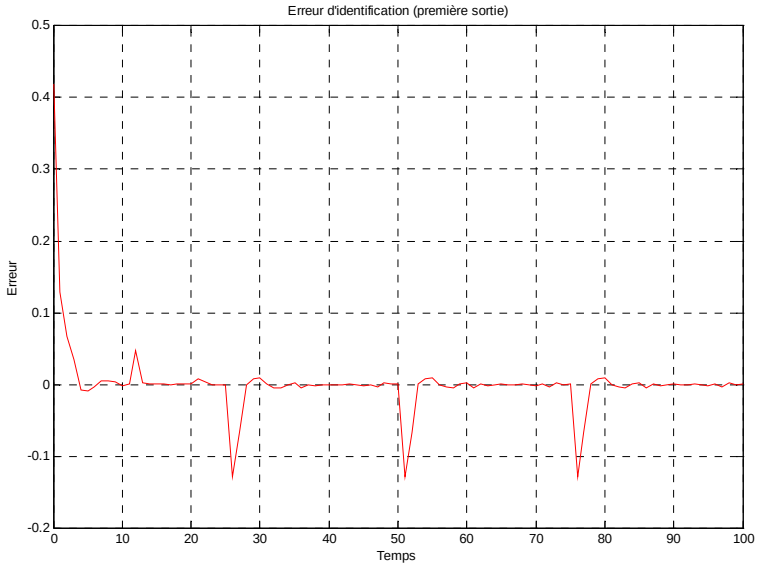


Figure II.10: Erreur d'identification de y_1

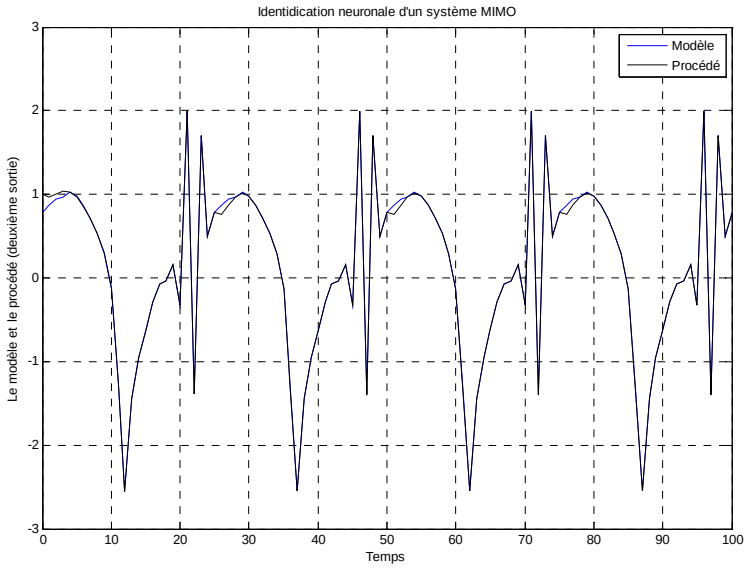


Figure II.11: Sorties, $y_2(-)$ du procédé et $\hat{y}_2(--)$ du modèle

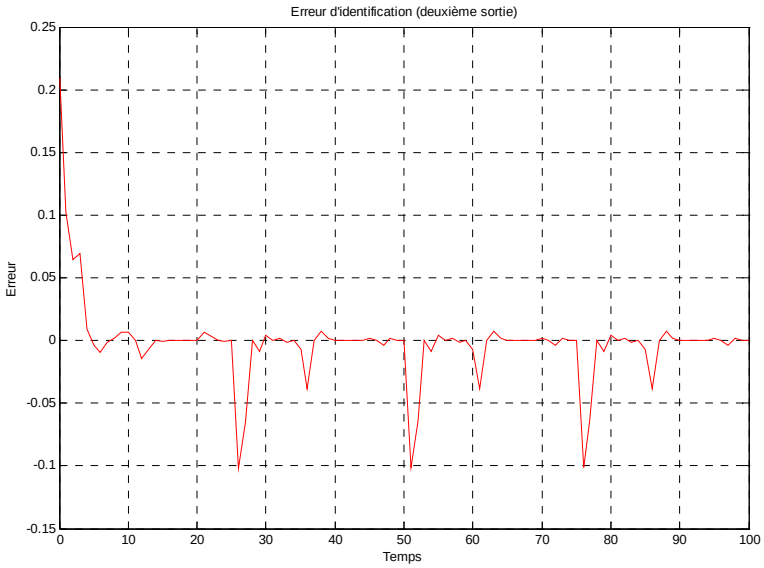


Figure II.12: Erreur d'identification de y_2

II.9 Modélisation en présence de perturbations

Dans la très grande majorité des applications, les réseaux de neurones sont utilisés pour réaliser une modélisation ou régression non linéaire. A cet effet, on réalise des mesures, en nombre fini, des entrées et des sorties du système que l'on cherche à modéliser. Ces mesures sont évidemment entachées de bruit de mesure, ou affectées par des perturbations

non mesurées. 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. Nous pouvons distinguer deux cas différents:

Dans le premier cas, nous supposons que la perturbation peut être représentée par le modèle linéaire et dans le deuxième cas, nous avons supposé que les différentes perturbations sont modélisées par un modèle non linéaire

II.10 Rejection de la perturbation (Problématique)

Soit un système dans l'absence d'une perturbation externe :

$$x_p(k+1) = f_1\big[x_p(k), u(k)\big]$$

$$y(k) = h_1\big[x_p(k)\big] \qquad \text{(II.29)}$$

Où $x_p(k) = \in \Re^n$ est l'état du système à l'instant k, $u(k) = \in \Re$ est l'entrée appliquée à l'instant k et y(k) est la sortie observée à l'instant *k*. on peut décrire le système par l'équation aux différences suivante :

$$y(k+1) = F_1[Y(k), U(k)] \qquad \text{(II.30)}$$

Où $Y(k) \in \Re^n = [y(k), y(k-1), \dots, y(k-n+1)]^T$

Et $U(k) \in \Re^n = [u(k), u(k-1), \dots, u(k-n+1)]^T$

La perturbation externe est représentée par l'équation suivante :

$$x_v(k+1) = f_2[x_v(k)]$$

$$v(k) = h_2[x_v(k)] \qquad \text{(II.31)}$$

Où $x_v(k) \in \Re^p$ est l'état du système générateur de perturbations et $v(k) \in \Re$ est la sortie correspondante. Désormais *v*(*k*) c'est la perturbation considérée comme la sortie du système dynamique libre, cette perturbation donnée par l'équation (II.31) est représentée par une forme autorégressive décrite par :

$$v(k+1) = F_2[\bar{V}(k)]$$

Où

$$\bar{V}(k) = F_2[v(k), v(k-1), \dots, v(k-p+1)]^T$$

Si v(k) est considéré aussi comme entrée externe du système donnée par l'équation (II.29) alors le système global peut être représenté par :

$$\begin{cases} x_p(k+1) = f_3\left[x_p(k), u(k), v(k)\right] \\ x_v(k+1) = f_2[x_v(k)] \\ v(k) = h_2[x_v(k)] \\ y(k) = h_1\left[x_p(k)\right] \end{cases} \quad \text{(II.32)}$$

Ou bien sous une forme équivalente:

$$x(k+1) = f_4[x(k), u(k)]$$
$$y(k) = h_1\left[x_p(k)\right] = h_4[x(k)] \quad \text{(II.33)}$$

Où $x = [x_p^T, x_v^T] \in R^{n+p}$ est l'état du système global. Nous remarquons alors que l'entrée et la sortie du système global sont les mêmes que ceux du système en absence de perturbations, l'espace d'état a été élargi pour tenir compte au système générateur des perturbations, à chaque instant k, les seules quantités mesurables sont l'entrée et la sortie du système global.

Le modèle non linéaire donné par (II.32) et (II.33) présente des difficultés mathématiques dans son analyse théorique, d'où la nécessité de recourir à plusieurs représentations, utilisées pour résoudre le problème de rejection des perturbations. Ces représentations sont données par [7] [12][19].

Représentation I :

$$y(k+1) = \sum_{i=0}^{N} \alpha_i f_i[y(k), \dots, y(k-n+1)] + \sum_{j=0}^{n-1} \beta_j [u(k-j) + v(k-j)]$$
$$v(k+1) = \sum_{i=0}^{p-1} \lambda_i v(k-i) \quad \text{(II.34)}$$

Représentation II :

$$y(k+1) = f[y(k), \dots, y(k-n+1)] + \sum_{j=0}^{n-1} \beta_j [u(k-j) + v(k-j)]$$
$$v(k+1) = \sum_{i=0}^{p-1} \lambda_i v(k-i) \quad \text{(II.35)}$$

Représentation III :

$$y(k+1) = f[y(k), \dots, y(k-n+1)] + \beta_0[u(k) + v(k)]$$
$$v(k+1) = g[v(k), \dots, v(k-p+1)] \quad \text{(II.36)}$$

Représentation IV :

$$y(k+1) = f\big[y(k), \dots, y(k-n+1), [u(k) + v(k)], \dots, [u(k-n+1) + v(k-n+1)]\big]$$
$$v(k+1) = g[v(k), \dots, v(k-p+1)] \quad \text{(II.37)}$$

Pour **la représentation l,** il s'agit seulement d'estimer les coefficients $\propto_i$ $(i = 1, \dots, N)$ et β_j $(j = 0, \dots, n-1)$, puisque les fonctions $f_i[.]$ $(i = 1, \dots, N)$sont connues. La perturbation dans ce cas est la sortie d'une équation aux différences d'ordre p et affecte le système additivement sur l'entrée, La **représentation II** est une généralisation de la représentation I, mais dans ce cas la fonction f [.] est inconnue et doit être estimée.

Dans **la représentation III**, le système aussi bien que la perturbation sont décrits par des équations non linéaires. Il est à noter que la sortie y (k+ 1) dépend seulement de u(k) et non pas de ses valeurs passées.

La **représentation IV** représente le cas le plus général du problème de rejection des perturbations, la sortie y (k+1) dépend non linéairement de l'entrée.

Le principe de l'identification neuronale des systèmes non linéaires avec rejection des perturbations est toujours le même. La seule différence réside dans l'extension de l'espace d'état du modèle d'identification neuronale.

II.11 Etudes et simulation

Dans ce qui suit, nous allons effectuer l'identification des systèmes non linéaires affectés par des perturbations additives en entrée en étudiant les deux représentations II et IV. Nous verrons dans les résultats des simulations la différence entre le cas de rejection et le cas de non rejection des perturbations.

II.11.1 Etude de la présentation II

Soit la représentation II donnée par l'équation (II.35) :

$$y(k+1) = f[y(k), \dots, y(k-n+1)] + \sum_{j=0}^{n-1} \beta_j[u(k-j) + v(k-j)]$$
$$v(k+1) = \textstyle\sum_{i=0}^{p-1} \lambda_i v(k-i)$$

Soit les polynômes D(z) et R(z), D(z) est stable, alors que R(z) est stable, ils sont définis comme suit :

$$D(z) = \beta_0 + \beta_1 z^{-1} + \cdots + \beta_{n-1} z^{-(n-1)}$$
$$R(z) = \lambda_0 + \lambda_1 z^{-1} + \cdots + \lambda_{n-1} z^{-(p-1)} \quad \text{(II.38)}$$

Un problème plus général est considéré ici, avec la sortie de système à l'instant *k+1* dépend linéairement de l'entrée *u*(*k*). En particulier, le système avec la perturbation additive, décrit par l'équation différentielle suivante, est considéré:

$$y(k+1) = f[Y(k)] + D(z)[u(k) + v(k)]$$
$$v(k+1) = R(z)v(k) \quad \text{(II.39)}$$

Où *u*(*k*) et *y*(*k*) sont, respectivement, l'entrée et la sortie du système à l'instant *k.*

Y(*k*), *D*(*z*) et *R*(*z*) sont définis précédemment. La non linéarité *f* du système est supposée inconnue. Cependant, tant que les nombres entiers *n* et *p* et les coefficients $\{\beta_i\}_{i=\overline{0,n-1}}$ et $\{\lambda_i\}_{i=\overline{0,n-1}}$ sont supposés connus.

$$D(z)v(k-i) = y(k-i+1) - f[Y(k-i)] - D(z)u(k-i) \quad \text{(II.40)}$$

(II.39) peut être écrit comme :

$$y(k+1) = f[Y(k)] + D(z)u(k) + \sum_{i=1}^{p} \lambda_{i-1} D(z)v(k-i) \quad \text{(II.41)}$$

D'où la représentation entrée-sortie du système dans la présence de la perturbation devient :

$$y(k+1) = \bar{f}[\bar{Y}(k)] + \bar{D}(z)u(k) \quad \text{(II.42)}$$

Où :

$$\bar{Y}(k) = [y(k), y(k-1), \ldots, y(k-(p+n)+1)]^T$$
$$\bar{f}[\bar{Y}(k)] = f[Y(k)] + \sum_{i=1}^{p} \lambda_{i-1} \left(y(k-i+1) - f[Y(k-i)]\right) \quad \text{(II.43)}$$
$$\bar{D}(z) = D(z)R_1(z) = D(z)[1 - z^{-1}R(z)] = \sum_{i=0}^{n+p-1} \bar{\beta}_i z^{-i} \quad \text{(II.44)}$$

$\bar{D}(z)$ Est un polynôme stable. Un modèle neuronal peut être utilisé pour approximer la non

Linéarité f. [7][19].

Comme le montre l'équation (II.42), la représentation entrée-sortie donnée par reste linéaire par rapport à l'entrée. Cependant le nombre des valeurs passées des entrées et des sorties nécessaires pour représenter le système a augmenté de p valeurs en comparaison avec la représentation entrée-sortie en absence des perturbations.

Exemple (II.11.1) : Cas d'une perturbation linéaire (additiveen entrée)

Soit un système du premier ordre en présence d'une perturbation externe v(k), le système est décrit par l'équation aux différences suivante :

$$f[y(k)] = \frac{y(k)}{1+y^2(k)} + u(k) + v(k) \qquad \text{(II.45)}$$

La perturbation v (k) est considéré comme étant la réponse libre du système linéaire suivant :

$$v_1(k+1) = \cos(\psi)\, v_1(k) + \sin(\psi)\, v_2(k)$$

$$v_2(k+1) = -\sin(\psi)\, v_1(k) + \cos(\psi)\, v_2(k)$$

$$v(k) = v_1(k) \qquad \text{(II.46)}$$

$$\psi = \frac{\pi}{6}\ , \qquad v_1(0) = 1\ et v_2(0) = 0$$

L'entrée du système est donnée par :

$$u(k) = 0.25\sin(2\pi k/60) + 0.5\ \sin(2\pi k/60)$$

Donc la représentation entrée sortie du système dans la présence de la perturbation est :

$$\bar{y}(k+1) = \bar{f}[y(k), y(k-1), y(k-2), u(k-1), u(k— 2) + u(k)]$$

Les figures ci-dessus montrent les résultats de simulations pour deux cas, sans rejection et avec rejection des perturbations, pour le cas sans rejection de perturbations , Figure (II.13), nous remarquons que l'erreur entre la sortie du système et celle du modèle neuronale est importante, par contre, cette erreur démunie dans le cas de rejection de perturbation, Figure(II.15)

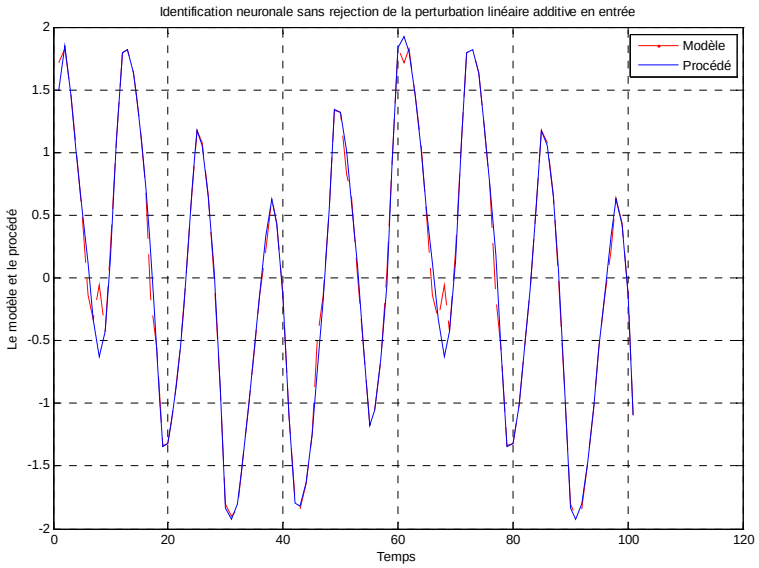


Figure II.13 : Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle
Sans rejection de la perturbation

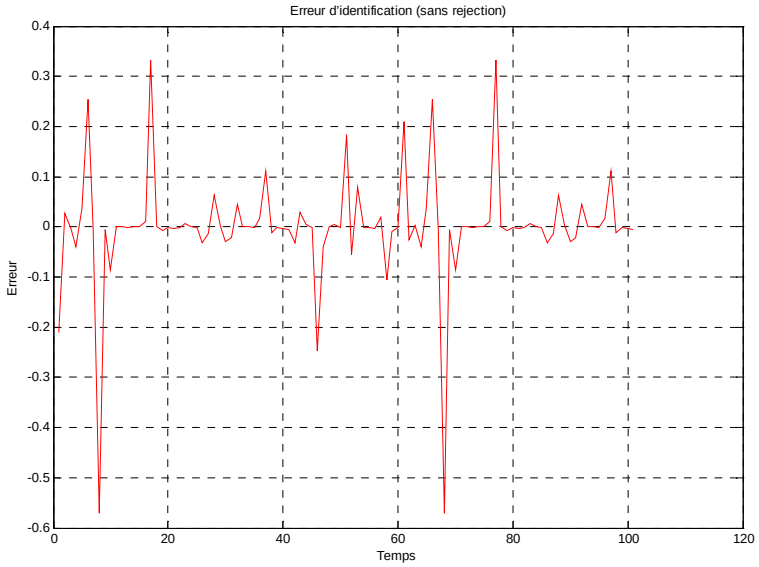


Figure II.14 : Erreur d'identification
Sans rejection de la perturbation

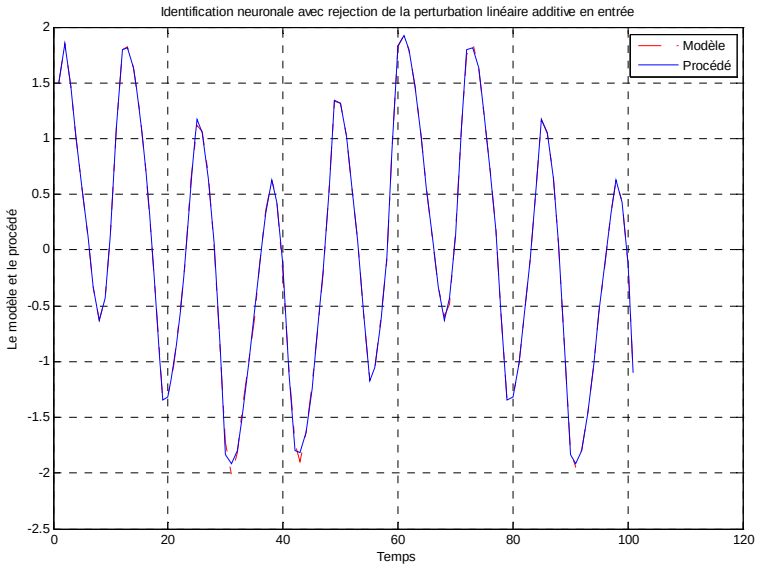


Figure II.15 : Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle
Avec rejection de la perturbation

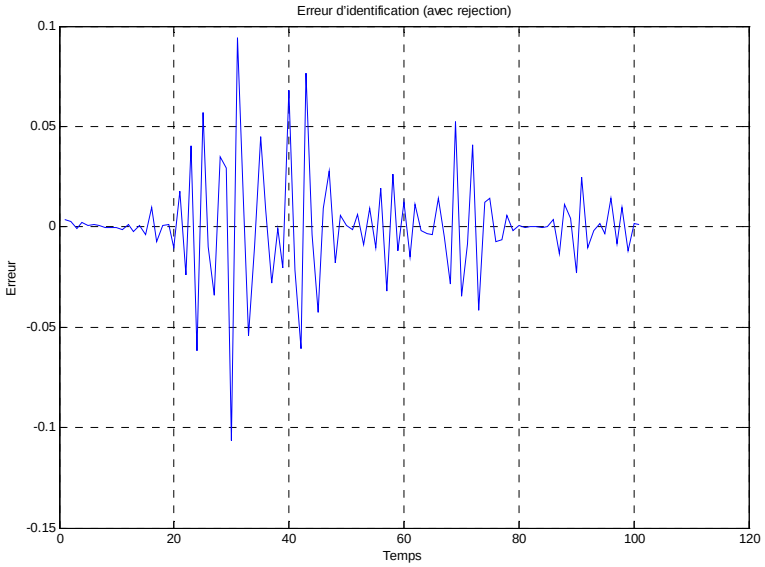


Figure II.16 : Erreur d'identification
Avec rejection de la perturbation

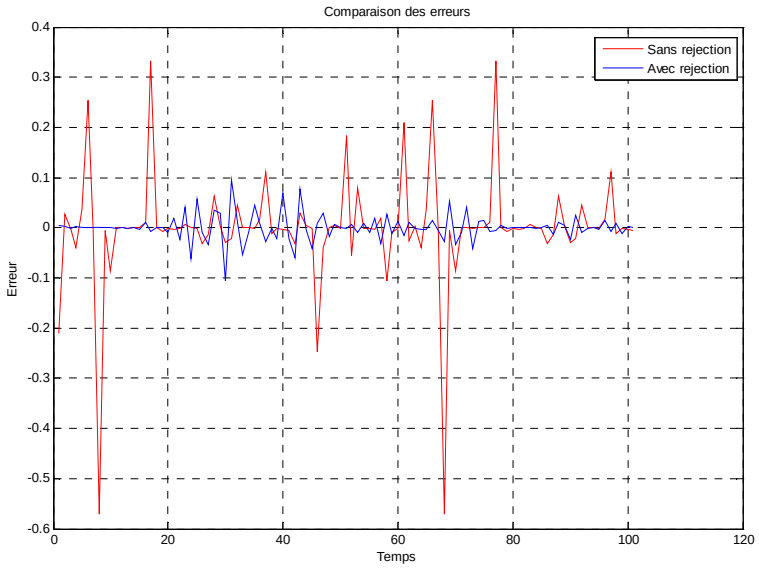


Figure II.17 : Comparaison de l'erreur
(Avec et sans rejection de la perturbation)

II.11.2 Etude de la présentation IV

Dans ce cas, nous considérons le problème général dans lequel la commande apparait comme non linéaire dans la dynamique du système et la perturbation est considérée comme étant la réponse libre d'un système non linéaire. A l'évidence, l'existence d'une représentation entrée sortie pour le système en présence d'une perturbation externe, est difficile à résoudre, puisque la sortie y(k + 1) dépend non linéairement de l'entrée [7] [12][19] . Si le système en présence d'une perturbation est représenté par:

$$y(k+1) = f[Y(k), U(k) + V(k)] \qquad \text{(II.47)}$$
$$v(k+1) = g[\bar{V}(k)]$$

Où

$Y(k) = [y(k), \dots, y(k-n+1)]^T$, $U(k) = [u(k), \dots, u(k-n+1)]^T$

$V(k) = [v(k), \dots, v(k-n+1)]^T$, $\bar{V}(k) = [v(k), \dots, v(k-p+1)]^T$

Donc le probleme c'est de detérminer si le système global admet une répresentation entrée sortie de la forme :

$$y(k+1) = \bar{f}[\bar{Y}(k), \bar{U}(k)] \qquad \text{(II.48)}$$

Où

$\bar{Y}(k) = [y(k), \dots, y(k-q)]^T$, $\bar{U}(k) = [u(k), \dots, u(k-q)]^T$ pour un entier positif fini q

Exemple (II.11.2): Cas d'une perturbation non linéaire (additive en entrée)

Notre objectif est l'identification d'un système de troisième ordre en présence d'une perturbation externe v(k) non linéaire, le système est décrit par l'équation aux différences suivante :

$$y(k+1) = \frac{0.6[\sin[y(k)y(k-1)y(k-2)[y(k-2)-1][u(k-1)+v(k-1)]]+[u(k)+v(k)]}{1+y^2(k-2)+y^2(k-1)} \quad \text{(II.49)}$$

La perturbation v(k) est considérée comme étant la réponse libre d'un système non linéaire du deuxième ordre représentée par l'équation aux différences suivante:

$$v(k+1) = \gamma_1 v(k) + \gamma_2 v(k-1) + \gamma_3 (v^2(k-1)-1)(v(k)-v(k-1)) \quad \text{(II.50)}$$

Avec: $\gamma_1 = 0.02,\ \gamma_2 = 0.0154 \ \ et \ \ \gamma_3 = 1.0154;\ \ v(1) = 0.1736\ et\ \ v(0) = 0$

L'entrée du système est donnée par :

$$u(k) = 0.75\sin(2\pi k/25) + 0.75\ \sin(2\pi k/10) \quad \text{(II.51)}$$

Dans le cas sans rejection de perturbations la figure (II.18) montre que l'erreur entre la sortie du système et celle du modèle neuronale est importante, par contre la figure(II.20) le cas de rejection de perturbation nous remarquons que cette erreur diminue.

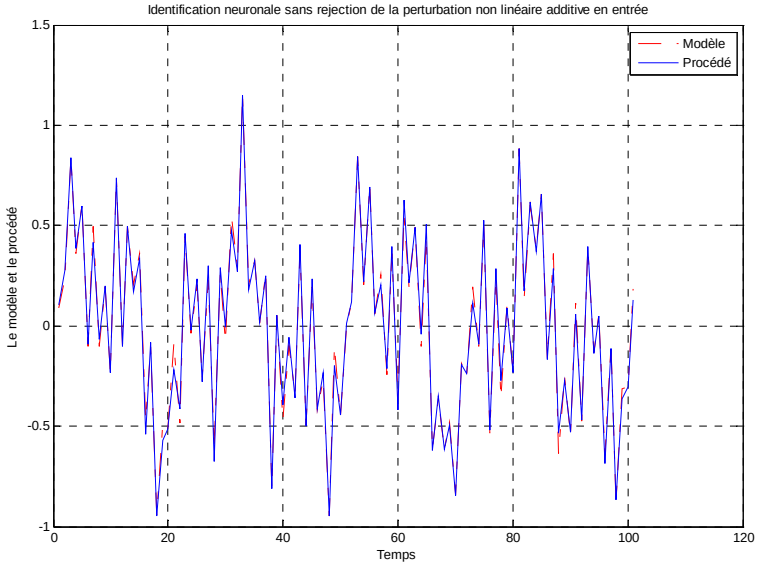


Figure II.18 : Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle
Sans rejection de la perturbation

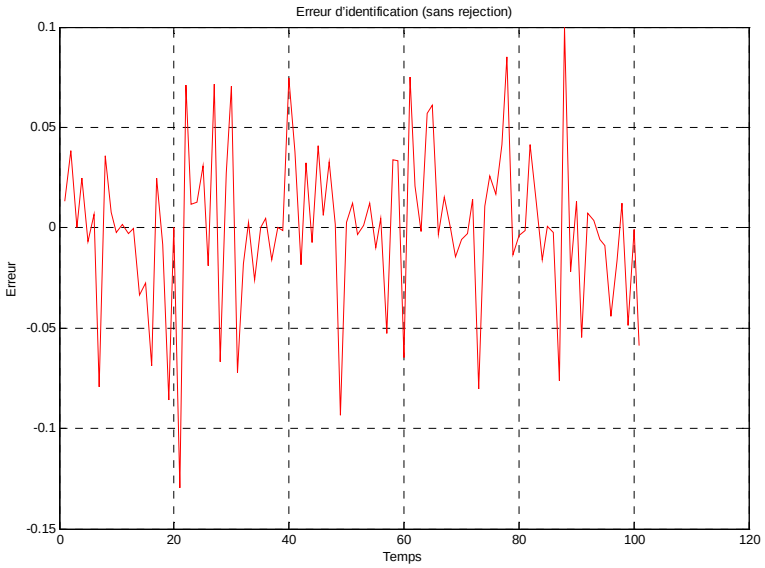


Figure II.19 : Erreur d'identification
Sans rejection de la perturbation

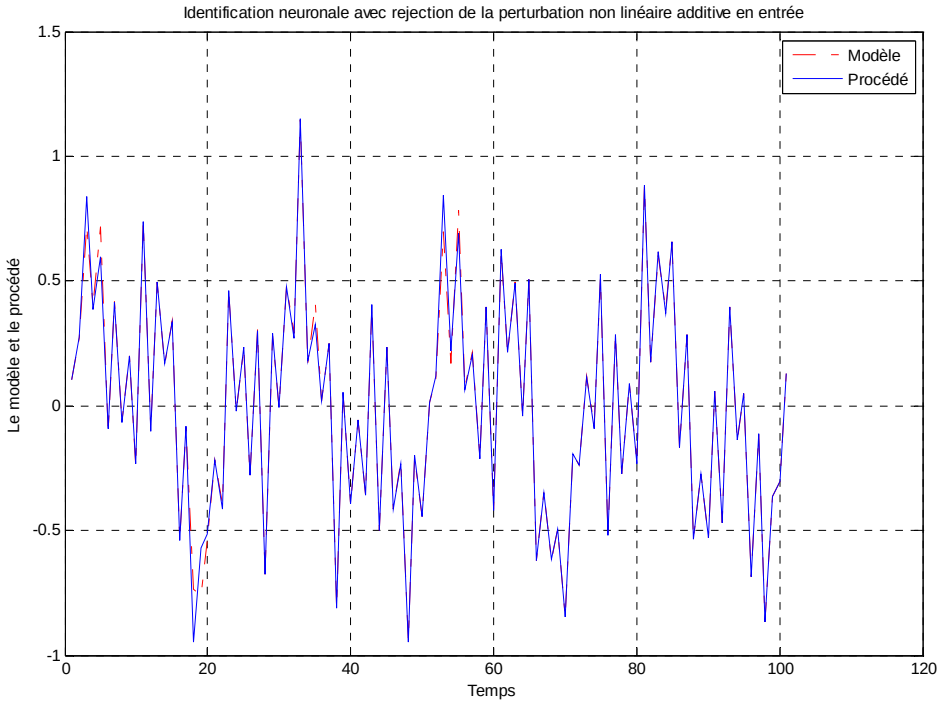


Figure II.20 : Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle
Avec rejection de la perturbation

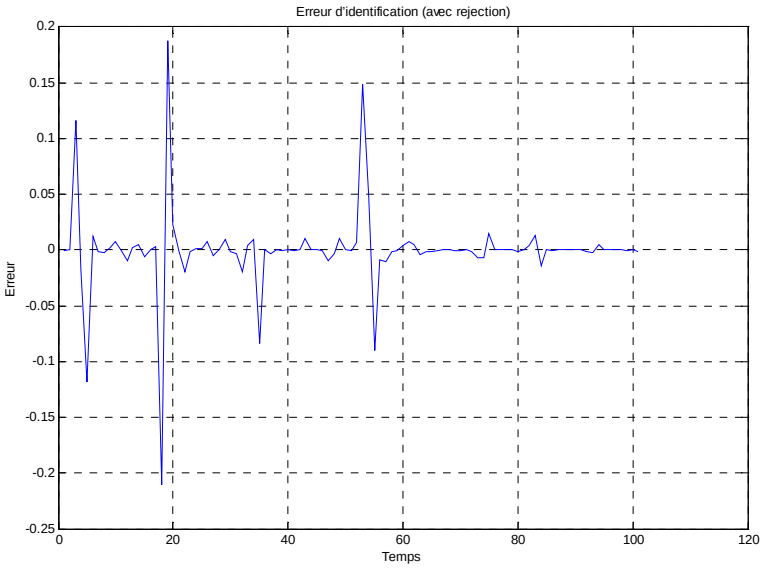


Figure II.21 : Erreur d'identification
Avec rejection de la perturbation

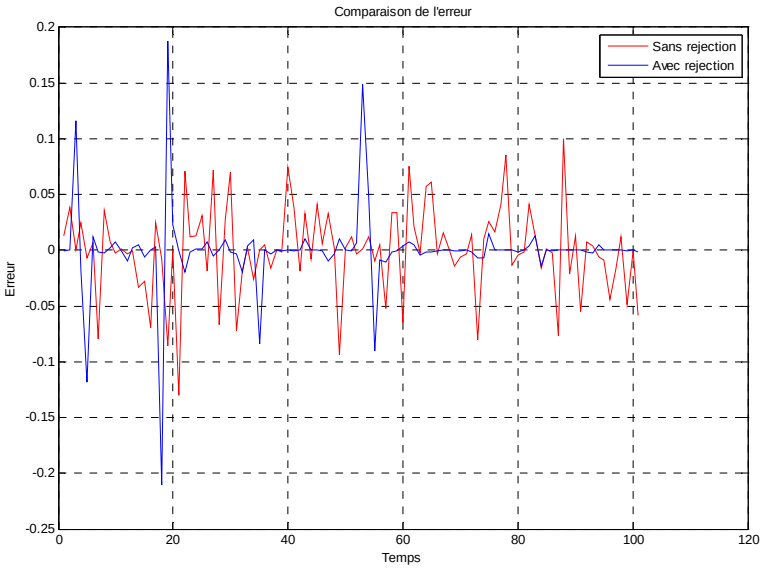


Figure II.22 : Comparaison de l'erreur
(Avec et sans rejection de la perturbation)

II.12 Rejection des perturbations additives en sortie

Nous allons considérer maintenant le cas des perturbations additives en sorties, deux cas sont considérés : Perturbation linéaire et perturbation non linéaire.

II.12.1 Perturbation linéaire

Soit le système linéaire SISO en présence d'une perturbation externe linéaire :

$$y(k+1) = f[y(k), y(k-1), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k—m+1)] + v(k) \quad \text{(II.52)}$$

La perturbation externe linéaire est donnée par :

$$v(k) = \sum_{i=0}^{p-1} \lambda_i v(k-i) \quad \text{(II.53)}$$

De (II.52) et (II.53) nous obtenons :

$$v(k) = \sum_{i=0}^{p-1} \lambda_i [y(k-i) - f[y(k-i-1), \dots, y(k-i-n), u(k-i-1), \dots, u\big(k—i-m\big)]] \quad \text{(II.54)}$$

L'équation(II.54) permet d'écrire (II.52) sous la forme :

$$y(k+1) = f[y(k), \dots, y(k-n+1), u(k), \dots, u(k—m+1)] + \sum_{i=0}^{p-1} \lambda_i f[y(k-i-1), \dots, y(k-i-n), u(k-i-1), \dots, u(k—i-m)] \quad \text{(II.55)}$$

La forme équivalente est :

$$y(k+1) = \bar{f}[\bar{Y}(k), \bar{U}(k)] \quad \text{(II.56)}$$

Où

$$\bar{Y}(k) = [y(k-1), \dots, y(k-(n+p)+1)\,]^T \text{et} \bar{U}(k) = [u(k-1), \dots, u(k-(m+p)+1)\,]^T$$

Exemple (II.12.1): Cas d'une perturbation linéaire (additive en sortie)

Cet exemple concerne le cas où la perturbation externe est décrite par un modèle linéaire.

La dynamique du système non linéaire du premier ordre est donnée par :

$$y(k+1) = \frac{0.9\, y(k)+u(k)}{1+y^2(k)} + v(k) \qquad \text{(II.57)}$$

Où v(k) est donnée par le modèle linéaire du deuxième ordre :

$$\begin{cases} v_1(k+1) = \cos(\psi)\, v_1(k) + \sin(\psi)\, v_2(k) \\ v_2(k+1) = -\sin(\psi)\, v_1(k) + \cos(\psi)\, v_2(k) \\ v(k) = 0.5\, v_1(k) \end{cases} \qquad \text{(II.58)}$$

Avec : $\psi = \frac{\pi}{6}$, $v_1(0) = 0.45$ et $v_2(0) = 0$

La sortie du modèle obtenu et celle du système pour l'entrée par :

$$u(k) = 0.25\sin(2\pi k/60) + 0.5\ \sin(2\pi k/60) \qquad \text{(II.59)}$$

Pour le premier cas (sans rejection), nous avons utilisé le modèle d'identification :

$$\hat{y}(k+1) = \hat{f}[y(k), u(k)] \qquad \text{(II.60)}$$

La sortie du modèle obtenu et celle du système pour l'entrée u(k) sont représenté dans la figure II.23, on constate que l'écart entre les deux sorties est important.

Ensuite, dans un deuxième cas, nous avons effectué un apprentissage sur le modèle neuronal :

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1)y(k-2), u(k), u(k-1), u(k-2)] \qquad \text{(II.61)}$$

La figure II.25représente la réponse de ce modèle et celle du système à l'entrée donnée dans le cas précédent. On remarque que l'écart entre les deux réponses est négligeable et que la perturbation v(k) ne présente, dans ce cas, aucun effet sur les performances du modèle obtenu.

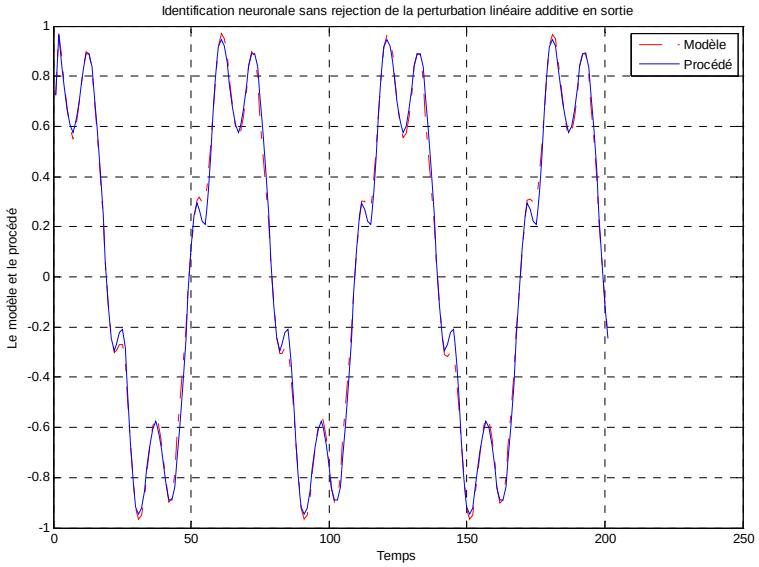


Figure II.23 : Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle
Sans rejection de la perturbation

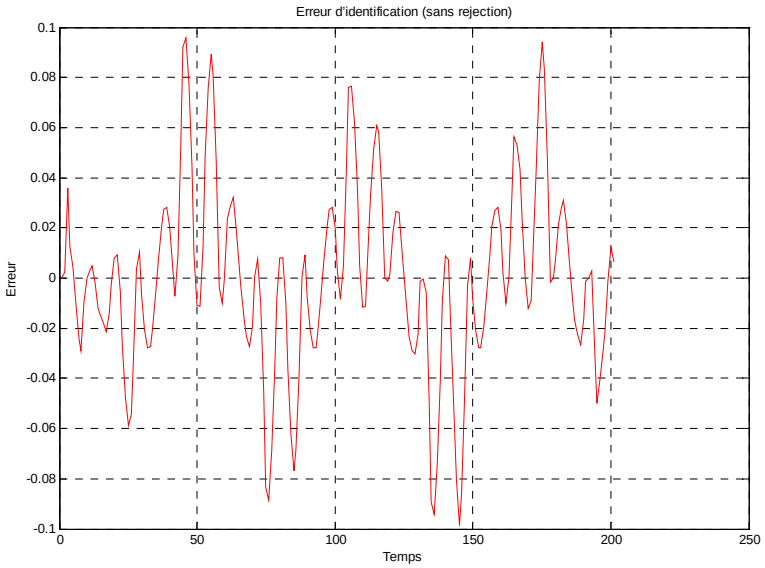


Figure II.24 : Erreur d'identification
Sans rejection de la perturbation

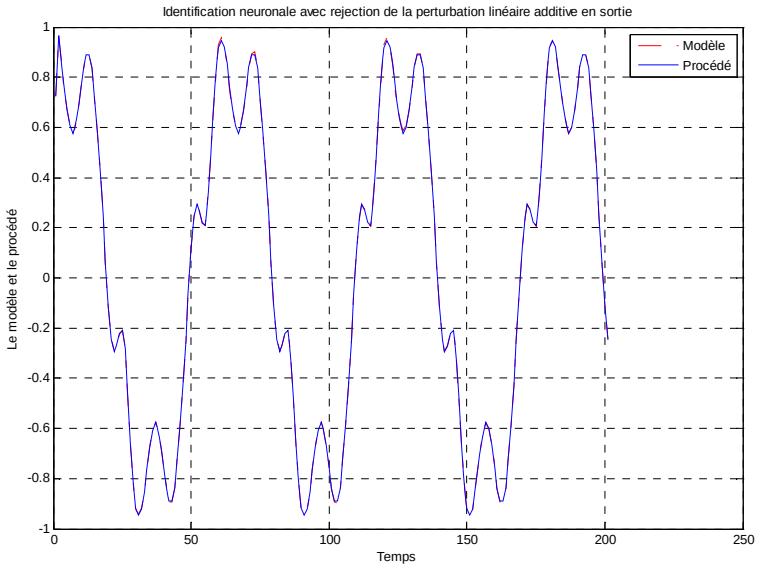


Figure II.25 : Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle
Avec rejection de la perturbation

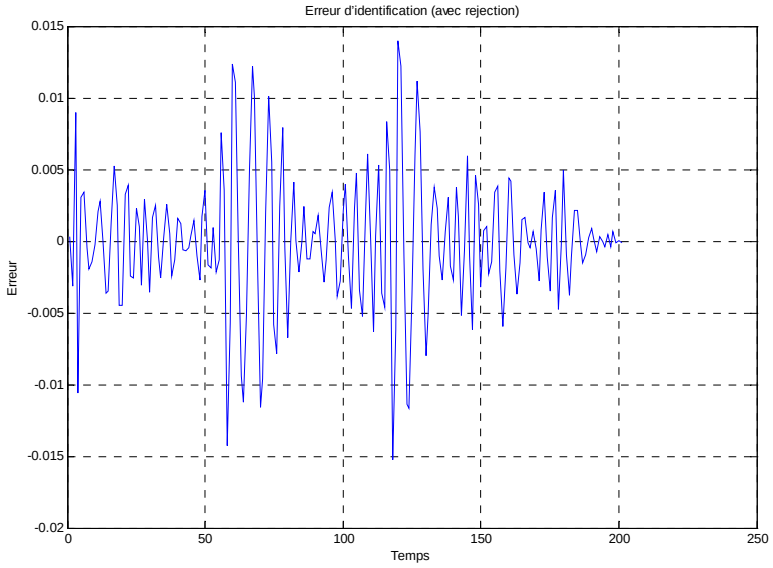


Figure II.26 : Erreur d'identification
Avec rejection de la perturbation

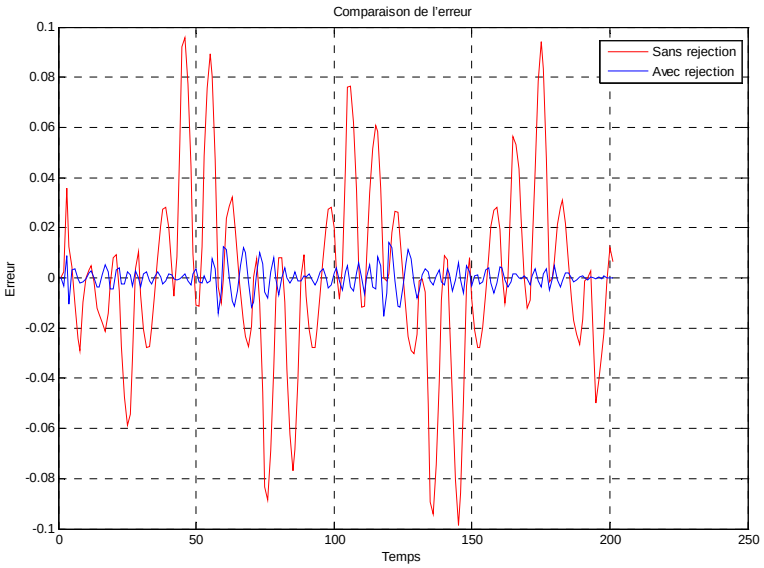


Figure II.27 : Comparaison de l'erreur
(Avec et sans rejection de la perturbation)

II.12.2 Perturbation non linéaire

Dans ce cas la perturbation est prise comme la sortie libre d'un système non linéaire. Un système non linéaire soumis à cette perturbation sur sa sortie est représenté par les équations suivantes:

$$y(k+1) = f[Y(k), U(k)] + v(k) \qquad \text{(II.62)}$$

$$v(k+1) = g[V(k)]$$

Où :

$$Y(k) = [y(k), \dots, y(k-n+1)\,]^T, U(k) = [u(k), \dots, u(k-m+1)\,]^T$$

$$V(k) = [v(k), \dots, v(k-p+1)\,]^T$$

v(k) Peut aussi s'écrire sous la forme suivante :

$$v(k) = y(k+1) - f[Y(k), U(k)] \qquad \text{(II.63)}$$

$$v(k+1) = g[V(k)]$$

Où

$$V(k) = [y(k+1) - f[Y(k), U(k)], y(k) - f[Y(k-1), U(k-1)], \dots$$

$$, y(k-p+2) - f[Y(k-p+1), U(k-p+1)\,]]^T \qquad \text{(II.64)}$$

En utilisant les équations (II.63) et (II.64) l'expression (II.62) devient :

$$y(k+1) = f[Y(k), U(k)] + g[V(k-1)]) \quad \text{(II.65)}$$

Et sous une autre forme :

$$y(k+1) = \bar{f}[\bar{Y}(k), \bar{U}(k)] \quad \text{(II.66)}$$

Où

$\bar{Y}(k) = [y(k), y(k-1), \dots, y(k-(n+p)+1)]^T$

$\bar{U}(k) = [u(k), u(k-1), \dots, u(k-(m+p)+1)\ ]^T$

En conclusion, nous constatons que quel que soit la nature de la perturbation additive en sortie (linéaire ou non linéaire), le problème de rejection des perturbations consiste toujours à utiliser un modèle d'identification d'ordre (n+p).

Exemple (II.12.2): Cas d'une perturbation non linéaire (additive en sortie)

Dans cet exemple, il s'agit d'identifier la dynamique directe du système non linéaire donné dans l'exemple précédent.

$$y(k+1) = \frac{0.9\, y(k) + u(k)}{1 + y^2(k)} + v(k)$$

Cette fois la perturbation v(k) est décrite par le modèle non linéaire. Soit un système non linéaire du premier ordre donnée par :

$$\begin{cases} v_1(k+1) = v_1(k) + 0.2\, v_2(k) \\ v_2(k+1) = -0.2\, v_1(k) + v_2(k) - 0.1(v_1^2(k) - 1)v_2(k) \\ v(k) = 0.4\ v_1(k) \end{cases} \quad \text{(II.67)}$$

Avec : $v_1(0) = 0.45\ \ et\ v_2(0) = 0$

Le modèle d'identification neuronal utilisé est de la forme :

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1), \dots, y(k-q), u(k), u(k-1), \dots, u(k-q)] \quad \text{(II.68)}$$

La sortie du modèle obtenu et celle du système pour l'entrée par :

$$u(k) = 0.25\sin(2\pi k/60) + 0.5\ \sin(2\pi k/60) \qquad \text{(II.69)}$$

Les deux modèles correspondants ont été entrainés. Ensuite l'entrée donnée dans l'exemple précédent a été utilisée pour comparer les sorties des modèles obtenus dans les deux cas (sans et avec rejection) avec celle du système considéré.

La Figure II.28 montre que l'écart entre la Sortie du système et celle du modèle obtenu pour le cas de sans rejection est important. Cet écart est faible pour le cas avec rejection comme le montre la Figure II.30.

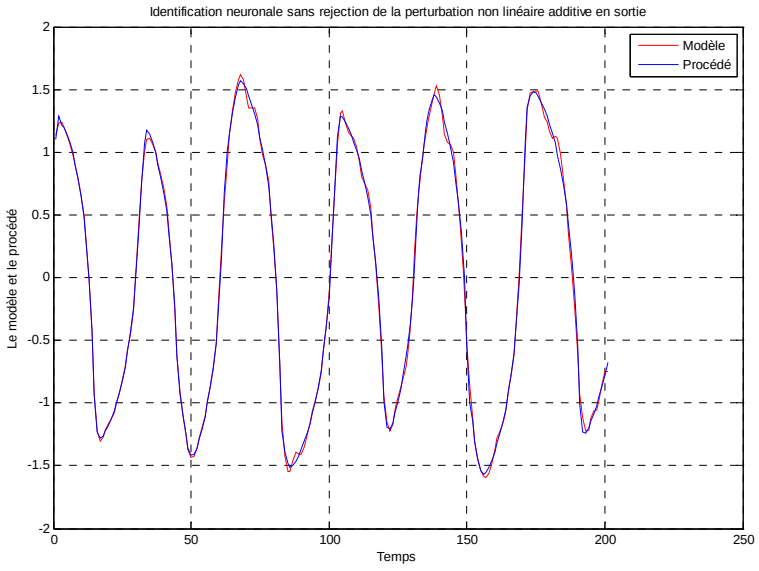


Figure II.28 : Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle
Sans rejection de la perturbation

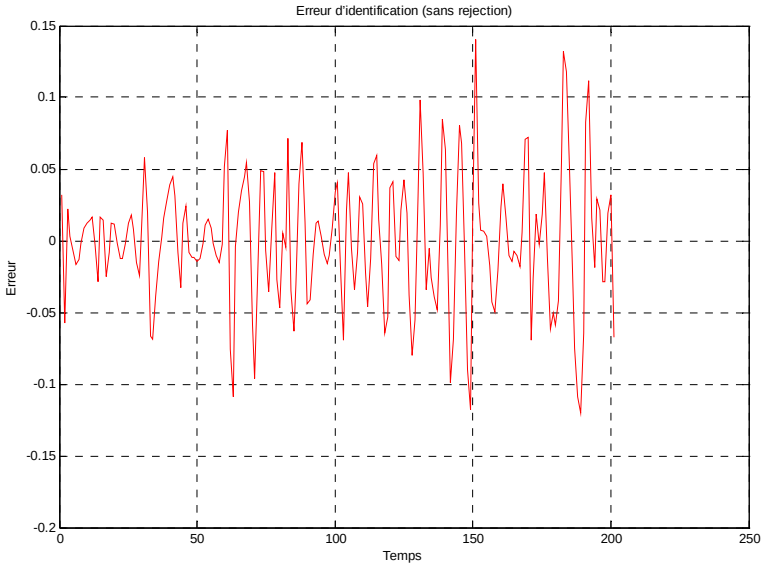


Figure II.29 : Erreur d'identification
Sans rejection de la perturbation

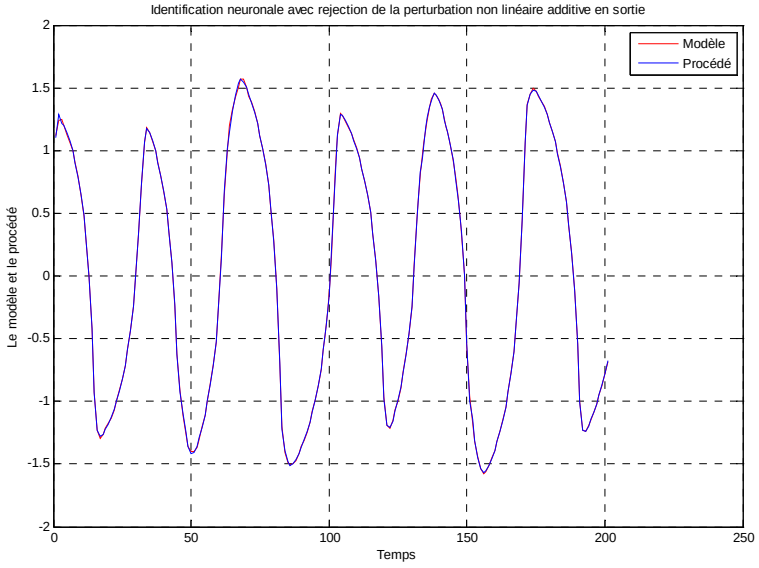


Figure II.30 : Sorties, $y(-)$ du procédé et $\hat{y}(--)$ du modèle
Avec rejection de la perturbation

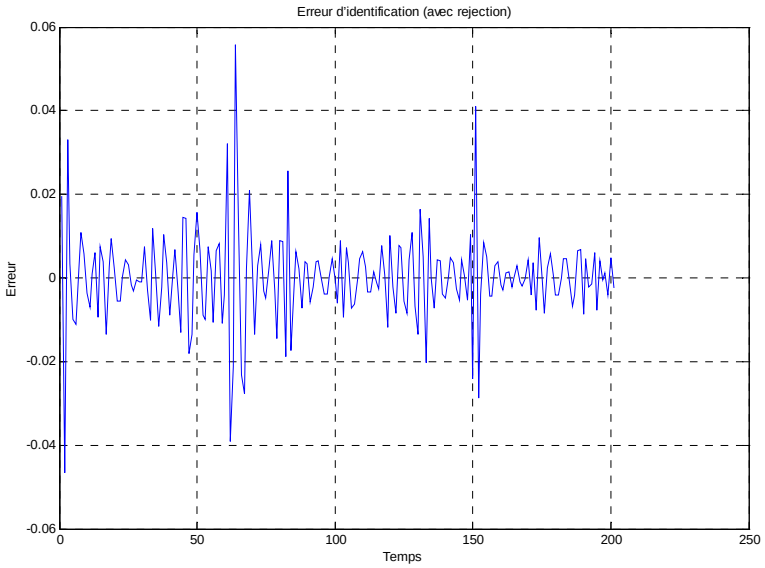


Figure II.31 : Erreur d'identification
Avec rejection de la perturbation

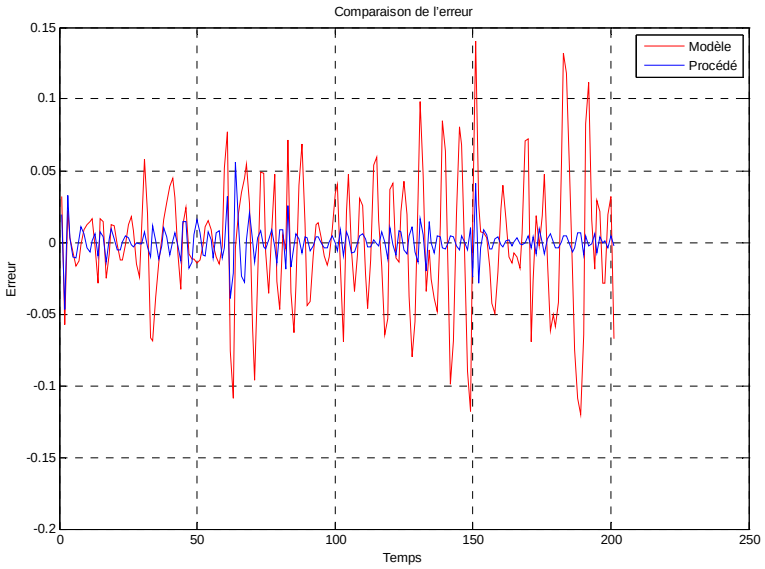


Figure II.32 : Comparaison de l'erreur
(Avec et sans rejection de la perturbation)

II.13 Conclusion

Dans ce chapitre, nous avons étudié le problème de l'identification des systèmes non linéaires et la théorie de la rejection de la perturbation.

- Nous avons vu qu'il existe plusieurs modèles d'identification pouvant être utilisés pour l'identification de ce type de systèmes.
- L'identification des systèmes non linéaires en présence des perturbations additives en entrée et en sortie.

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.

Ce 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.

Les deux cas de représentation des perturbations, par un modèle linéaire ou 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 régresseur.

Tous les exemples considérés dans ce chapitre ont été pris de [6].En effet, pour chaque exemple traité, nous avons constaté que l'approche neuronale donne de bons résultats.

CHAPITRE III

Contrôle neuronal des systèmes non linéaires

III.1 Introduction

Les principales difficultés dans la théorie de la commande des systèmes dynamiques réels sont les non linéarités et les incertitudes. Or la commande passe par l'élaboration d'un modèle mathématique du système en trouvant une relation entre les entrées et les sorties, ce qui suppose une bonne connaissance de la dynamique du système et ses propriétés.

Dans le cas des systèmes non linéaires, les techniques conventionnelles ont montré souvent leur insuffisance surtout quand les systèmes à étudier présentent de fortes non linéarités.

Le manque de connaissances à priori nécessaires pour l'élaboration du modèle mathématique était en quelque sorte dans cet échec.

Face à ce problème, le recours aux méthodes de commandes par apprentissage est devenu une nécessité car les systèmes de commande obtenus ainsi procèdent par collecter des données empiriques, stocker et extraire les connaissances contenues dans celle-ci et utiliser ces connaissances pour réagir à de nouvelles situations : on est passé à la commande intelligente (intelligent control) [34][35].

Parmi les techniques d'apprentissage, les réseaux de neurones, par le faite qu'ils sont des approximateurs universels et parcimonieux [16][35][36] et par leur capacité de s'adapter à une dynamique évoluant au cours du temps, sont de bons candidats, de plus, en tant que systèmes multi entrées multi sorties, ils peuvent être utilisés dans le cadre de la commande des systèmes multi variables.

Un réseau de neurones non bouclé réalise une ou plusieurs fonctions algébriques de ses entrées, par composition des fonctions réalisées par chacun de ses neurones. [8][21]. Ceux-ci sont organisés en couches et interconnectés par des connexions synaptiques pondérées. L'apprentissage supervisé d'un réseau à N sortie, consiste à modifier les poids w pour avoir un comportement donné en minimisant une fonction de coût représentée souvent par l'erreur quadratique [8][37].

$$E(W) = \sum_{i=1}^{N} \frac{1}{2}(y_{di} - y_i)^2 \qquad \text{(III.1)}$$

Où y_d est la sortie désirée du réseau, y_i est la sortie calculée.

Plusieurs chercheurs ont essayé d'exploiter les avantages des réseaux de neurones pour commander un système dynamique, et plus précisément, dans le domaine de la robotique [38][39] et pour la commande d'un moteur asynchrone [16][40][41]. On peut

trouver plus de détails sur les structures de commandes avec modèle neuronal dans [42][43].

III.2 Le modèle neuronal inverse

Bien que le modèle inverse de système joue un rôle important dans la théorie de la commande, l'accomplissement de sa forme analytique est assez laborieuse. Plusieurs méthodes de modélisation des systèmes ont été présentées dans la littérature [8][15].Un système dynamique peut être décrit par l'équation (III .2) reliant ces entrées aux sorties :

$$y(k+1) = f(\, y(k), \dots y(k-n+1), u(k), \dots, u(k-m+1)) \qquad \text{(III.2)}$$

Où la sortie du système $y(k+1)$ dépend des n valeurs précédentes de la sortie et les m valeurs passées de l'entrée. En général, le modèle inverse de ce système peut être présenté sous la forme suivante (III.3):

$$u(k) = f^{-1}(y(k+1), y(k), \dots, y(k-n+1), u(k-1), \dots, u(k-m)) \quad \text{(III.3)}$$

Les réseaux de perceptrons multicouches peuvent être utilisés pour élaborer le modèle neuronal inverse du système.

Cependant, d'autres types de réseaux ont été utilisés [44], les réseaux MLP statiques présentent la solution la plus simple, toutefois la représentation de l'aspect dynamique du système reste une problématique. L'application des retards vers la couche d'entrée à ce type de réseau peut présenter la solution pour palier ce manque et remédier à l'aspect statique des MLP.

Cette solution présente l'avantage de permettre l'application de l'algorithme traditionnel de rétro-propagation du gradient pour l'apprentissage des réseaux multicouches.

Le réseau correspondant à (III.3) est :

$$\hat{u}(k) = g(x(k), w) \qquad \text{(III.4)}$$

La fonction g va être approximée par un réseau de neurones en modifiant ses poids w. Le réseau a un vecteur d'entrée x(k) composé de la sortie k+1 et des valeurs des sorties et entrées passées et ayant pour sortie le signal û qui va servir à commander le système figure(III.1).

L'identification du modèle neuronal inverse commence par la détermination du vecteur d'entrée, à savoir le nombre des retards en sorties et entrées, ceci est lié à l'ordre du système.

La deuxième étape est l'architecture du réseau (nombre de couches cachées et nombre de neurones). La détermination des paramètres du modèle est effectuée par un apprentissage du réseau selon trois architectures.

III.2.1 Architecture générale d'apprentissage

Dans la première architecture figure (III.2), le signal u est appliqué à l'entrée de système, produisant une sortie y qui est fournie au réseau. La différence entre le signal d'entrée u et la sortie du réseau de neurone û est en, l'erreur rétropropagée à travers le réseau et qui va servir pour l'apprentissage hors ligne du réseau [45]. On tente de minimiser le critère *E(w)*:

$$E(w) = \sum_{i=1}^{N} \frac{1}{2}(u(k) - \hat{u}(k,w))^2 \qquad \text{(III.5)}$$

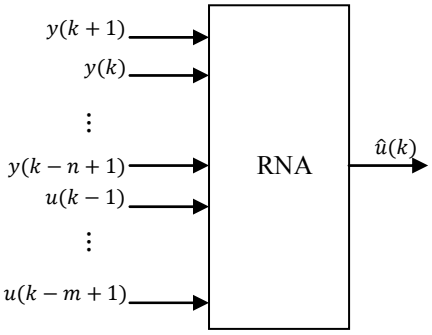


Figure (III.1) :Réseau de neurone pour identifier le modèle inverse

Où *w* est un vecteur contenant les poids du réseau inverse.

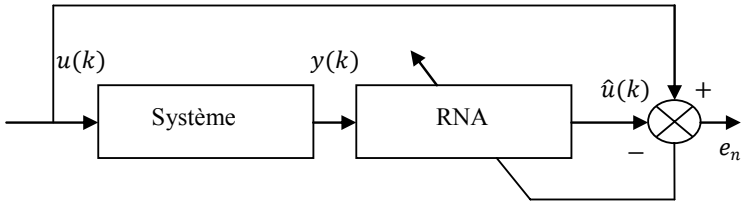


Figure (III.2) : Architecture générale d'apprentissage

Cette méthode présente plusieurs inconvénients [46][47] : Les sorties y du système utilisées dans l'apprentissage ne garantissent pas que les sorties du modèle neuronal vont être dans des régions voulues pour le succès de son utilisation dans la commande.

Si le système à commander est multi-variable, le modèle ainsi retenu peut ne pas imiter le système réel.

III.2.2 Architecture indirecte d'apprentissage

Cette approche est une mise en œuvre particulière de l'architecture précédente dans laquelle, le modèle inverse en cours d'apprentissage sert aussi à commander le système. La consigne r est fournie au premier réseau qui produit une commande û au système, la sortie de celui-ci est passée comme consigne à la seconde copie du modèle inverse, qui produit alors une commande û. La différence entre u et û sert de signal d'erreur afin d'effectuer l'apprentissage des paramètres du modèle par rétro-propagation. Cette architecture, proposée par [48] est présentée sur la Figure (III.3).

L'idée derrière cette architecture est que la minimisation de l'erreur commise sur la commande entraînera une minimisation de l'erreur en sortie du système. Cependant, une erreur nulle sur la commande ne provoque pas nécessairement l'annulation de l'erreur totale en sortie du système.

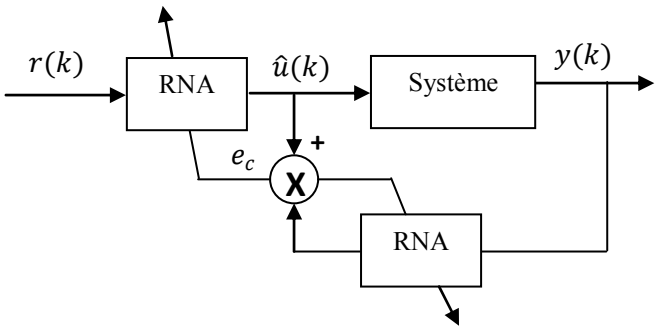


Figure (III.3) : Architecture indirecte d'apprentissage

III.2.3 Architecture spécialisé d'apprentissage

Contrairement aux deux précédentes architectures, l'architecture spécialisée, utilisée par [38], procède par l'utilisation de l'erreur e_c « équation (III.6) », la différence entre la sortie désirée r et la sortie réelle du système, pour apprendre au modèle neuronal à suivre la dynamique inverse du système figure (III.4).

$$e_c = r(k) - y(k) \qquad \text{(III.6)}$$

On peut remarquer que la valeur utilisée pour l'apprentissage des paramètres du modèle inverse est l'erreur en sortie du procédé. Une utilisation adéquate de la rétro-propagation imposerait pourtant que l'on rétropropage l'erreur en sortie du modèle, qui est naturellement inconnue. Quoiqu'il en soit, l'expérience montre qu'en général, cette approche est inefficace. De plus, le modèle connexionniste étant utilisé dés l'initialisation pour commander le procédé, il est conseillé de disposer au départ d'un modèle relativement cohérent pour que cette architecture puisse fonctionner [49].

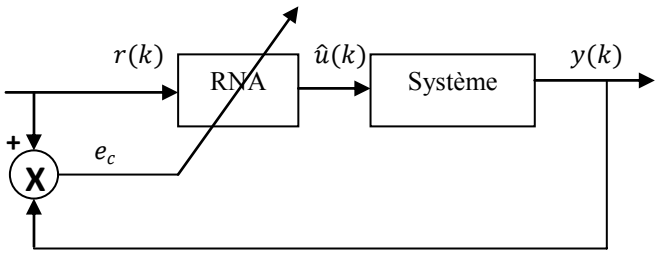


Figure (III.4) : Architecture spécialisée d'apprentissage

III.3 Commande neuronale directe par modèle inverse

Comme son nom l'indique, le modèle neuronal inverse placé en amant du système, est utilisé comme contrôleur pour commander le système en boucle ouverte figure (III.5)

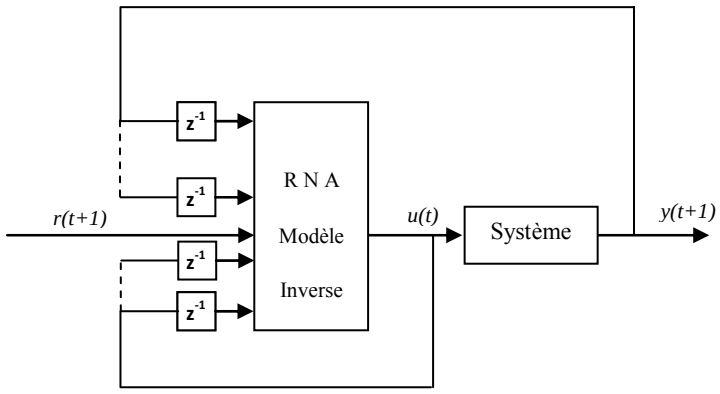


Figure (III.5) : Commande directe par modèle inverse

La valeur de y(k+1) Dans (III.3) est substituée par la sortie désirée r(k+1) et on alimente le réseau par les valeurs retardées de u(k) et y(k). Si le modèle neuronal est exactement l'inverse du système alors il conduit la sortie à suivre la consigne. Cette commande a été utilisée par [48][50][51].

III.5 Contrôle adaptatif

Dans la littérature du contrôle adaptatif, il existe deux approches différentes, utilisées pour contrôler des systèmes linéaires et non linéaires variant lentement. Il s'agit du contrôle adaptatif direct et du contrôle adaptatif indirect. Dans ce cas les structures du contrôleur et du modèle d'identification sont linéaires. Cependant àla place des modèles linéaires utilisés dans le contrôle adaptatif conventionnel, des systèmes neuronaux sont utilisés pour l'identification et le contrôle d'une plus large classe de systèmes non linéaires, dans des situations de larges incertitudes et avec des variations inconnues concernant les structures et les paramètres du système [17][19][52].

III.5.1 Position du problème du contrôle adaptatif

Soit un système inconnu avec une paire entrée-sortie $\{r(k), y(k)\}$, Un modèle de référence avec une paire entrée-sortie $\{r(k), y_m(k)\}$ représente le comportement désiré de la sortie. Que le système soit linéaire ou non linéaire, l'objectif est de déterminer une entrée de contrôle u(k) tel que $e_c(k) = y(k) - y_m(k)$ tende vers zéro [6] [12][19].

III.5.2 Contrôle adaptatif des systèmes

Il existe deux approches de contrôle adaptatif: l'approche directe et l'approche indirecte [53]. Dans le contrôle adaptatif direct, montré en figure (III.6), e, est l'erreur de sortie (erreur de suivi) entre la sortie du système $y(k)$et la sortie du modèle de référence $y_m(k)$: Les paramètres du contrôleur sont directement ajustés pour réduire une norme de l'erreur de sortie

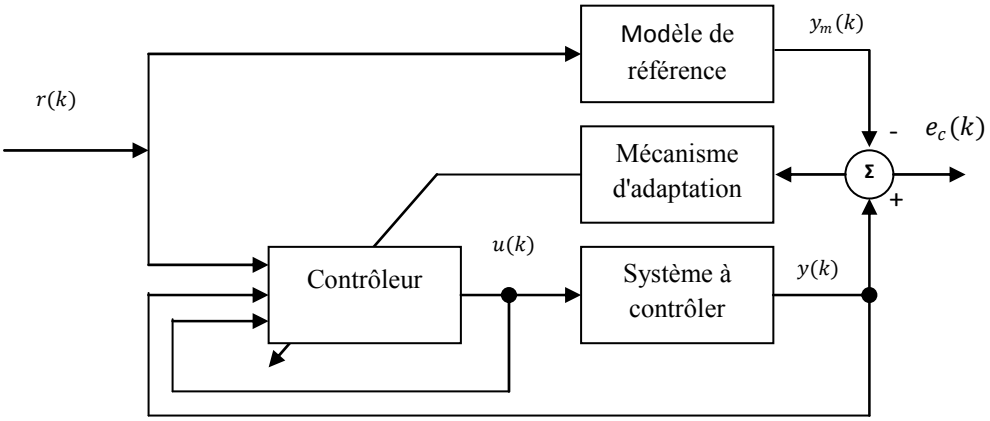


Figure (III.6): Contrôle adaptatif direct

Dans l'approche indirecte, les paramètres du système à contrôler sont estimés et utilisés ensuite pour déterminer les paramètres du contrôleur ce qui est montré en figure(III.7).

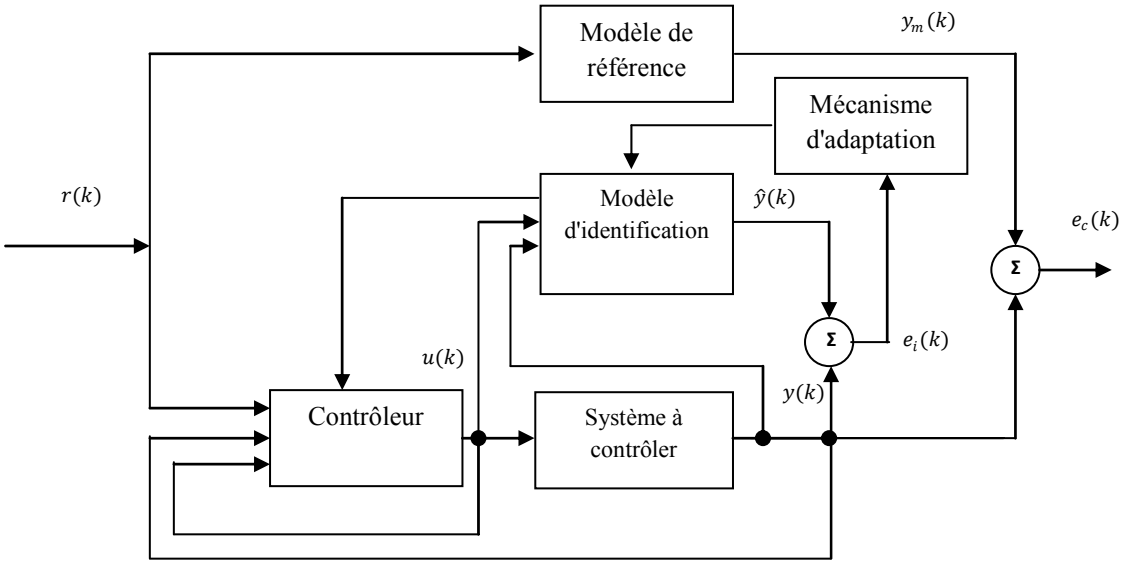


Figure (III.7). Contrôle adaptatif indirect

Le contrôle adaptatif des systèmes non linéaires inconnus nécessite l'utilisation d'un autre appui tel que les réseaux neurones afin d'appliquer les deux techniques conventionnelles du contrôle adaptatif, nous parlons alors du contrôle adaptatif neuronal.

III.6 Contrôle neuronaladaptatif

Lorsqu'un système est construit à partir d'un système neuronal équipé d'un algorithme d'adaptation, il est appelé contrôleur adaptatif neuronal [18][19][52].

Dans les structures de contrôle adaptatif neuronal, les réseaux de neurones sont utilisés pour remplacer les transformations linéaires utilisées par les techniques standards de contrôle adaptatif [3][54].

Dans le cas de contrôle adaptatif à modèle de référence, les performances désirées du système en boucle fermée sont spécifiées par un modèle de référence stable, défini par ses entrées sorties$\{r(k), y_m(k)\}$. Le rôle de contrôleur est de commander la sortie du système de façon à minimiser l'écart entre la sortie du système et la sortie du modèle de référence.

Cette erreur est utilisée pour l'entraînement du contrôleur. Donc cette approche est étroitement liée à l'entraînement du modèle inverse [55].

La procédure d'entraînement force le contrôleur à se comporter comme un modèle inverse réglé dans le sens défini par le modèle de référence.

Deux structures de contrôle adaptatif neuronal à modèle de référence ont été proposées par Narendra et Parthasarathy[54]; la structure de contrôle direct et la structure de contrôle indirect.

Des méthodes permettant l'adaptation des poids du contrôleur, Dans le cas du contrôle indirect, les méthodes d'entraînement ressemblent, en principe, à la méthode d'apprentissage spécialisé [55].

III.6.1 Contrôle neuronal adaptatif direct

Lorsqu'une structure de contrôle neuronal adaptatif utilise un système neuronal comme contrôleur, nous parlons de contrôle neuronal **adaptatif** direct.

Par exemple en prenant, comme structure pour le système inconnu à contrôler:

$$y(k+1) = f[y(k), u(k)] \quad \text{(III.7)}$$

Pour le modèle de référence:

$$y_m(k+1) = f[y_m(k), r(k)] \quad \text{(III.8)}$$

La structure du contrôleur neuronal est un système neuronal à 3 entrées et une sortie.

La sortie globale du système neuronal représente l'action de contrôle à appliquer, celle-ci est donnée par:

$$u(k+1) = NN[y_m(k), y(k), u(k)] \quad \text{(III.9)}$$

L'ajustement de ces paramètres se fait directement en se basant sur l'utilisation de l'erreur de sortie :

$$e_c(k+1) = y(k+1) - y_m(k+1) \quad \text{(III.10)}$$

Cependant, ce type de contrôleur est difficile à utiliser lorsque le système est inconnu, puisque l'effet du changement des paramètres du contrôleur sure_c, ne peut pas être calculé [6][19].

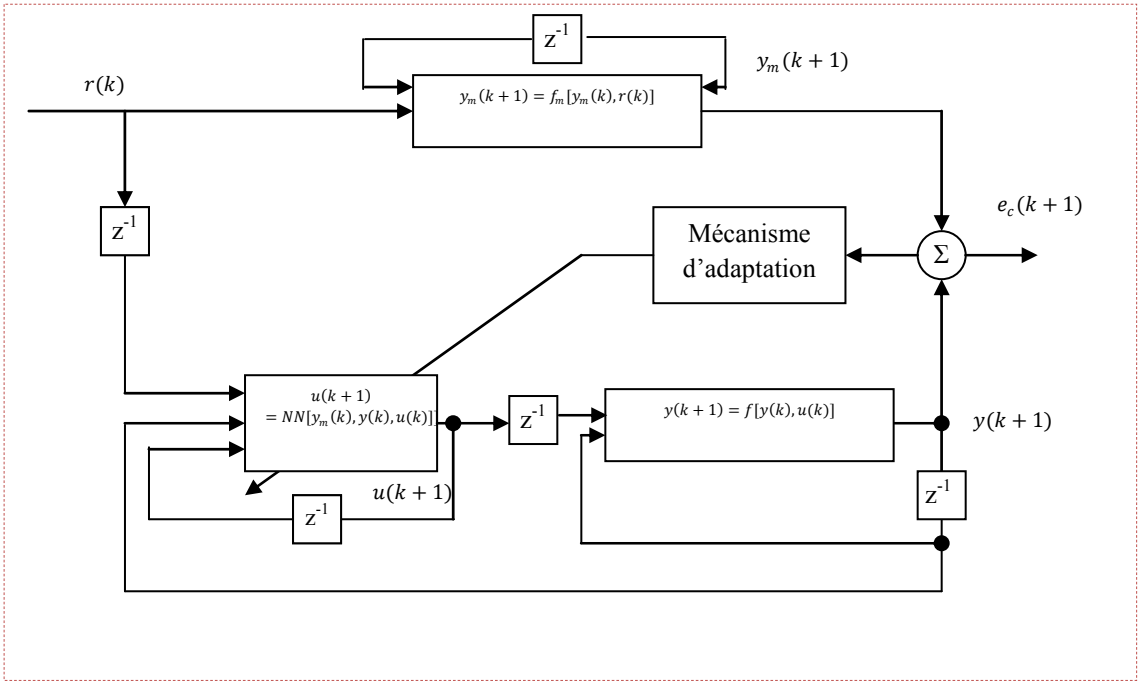


Figure (III.8): Contrôle neuronal adaptatif direct

III.6.2 Contrôle neuronaladaptatif indirect

Dans le cas de ce contrôleur, le calcul se base sur le modèle d'identification neuronal du système à contrôler.

Le principe du contrôle neuronaladaptatif indirect se résume en deux étapes:

- Préparation du modèle d'identification neuronal représentant le système à contrôler hors ligne dans le but d'accélérer la convergence dans la boucle du contrôle.
- Identification du système non linéaire et calcul de l'action de contrôle adaptatif à appliquer au système pour que sa sortie suive celle du modèle de référence.

Comme nous l'avons dit le contrôleur est calculé en se basant sur le modèle

d'identification neuronal, en d'autres termes la structure du contrôleur dépend de la structure choisie pour la représentation du système non linéaire.

Pour la structure de contrôle indirect, les paramètres du contrôleur sont ajustés de façon à minimiser l'écart entre la sortie du système et celle du modèle de référence. Le signal de commande généré par le contrôleur est utilisé comme une entrée pour le modèle neuronal et le système commandé, d'une part, l'erreur entre la sortie du système et celle du modèle sera utilisée pour ajuster les paramètres du modèle neuronal. D'autre part, l'erreur entre la sortie du modèle de référence et la sortie du système commandé sera utilisée pour adapter les paramètres du contrôleur neuronal.

L'objectif du contrôleur et de générer les commandes nécessaires pour deux buts: le premier est d'adapter les paramètres du modèle de façon à avoir une bonne identification, le second est que le système réel suit le comportement du modèle de référence.

Notons enfin que les performances obtenues par le contrôle indirect sont plus intéressantes que celles obtenues par le contrôle direct [3][56][57].

- **Cas d'un modèle d'identification du type IV**

Si nous considérons maintenant un autre cas, en prenant pour modèle d'identification la structure IV la plus générale. Alors pour un système SISO d'ordre 2 quelconque donné par:

$$y(k+1) = f[y(k), y(k-1), u(k), u(k-1)] \qquad \text{(III.11)}$$

Le modèle d'identification neuronal choisi est donné par:

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1), u(k), u(k-1)] \qquad \text{(III.12)}$$

Où$\hat{f}[.]$ Est un système neuronal avec quatre entrées et une sortie, Les paramètres du modèle sont ajustés en se basant sur l'erreur d'identification.

$$\hat{y}(k+1) = NN[y(k), y(k-1), u(k), u(k-1)] \qquad \text{(III.13)}$$

Pour un modèle de référence donné par exemple par:

$$y_m(k+1) = f_m[y_m(k), r(k)] \quad \text{(III.14)}$$

La structure du contrôleur neuronal est donnée par:

$$u(k) = f_c[y_m(k+1), y(k), y(k-1), u(k-1)] \quad \text{(III.15)}$$

Où $f_c[.]$ est aussi un système neuronal avec quatre entrées et une sortie, Les paramètres du contrôleur sont ajustés en se basant sur l'erreur de contrôle tel que :

$e_c(k+1) = y(k+1) - y_m(k+1)$Dans cette équation d'erreur, le modèle d'identification neuronal est utilisé à la place du système puisqu'il est inconnu

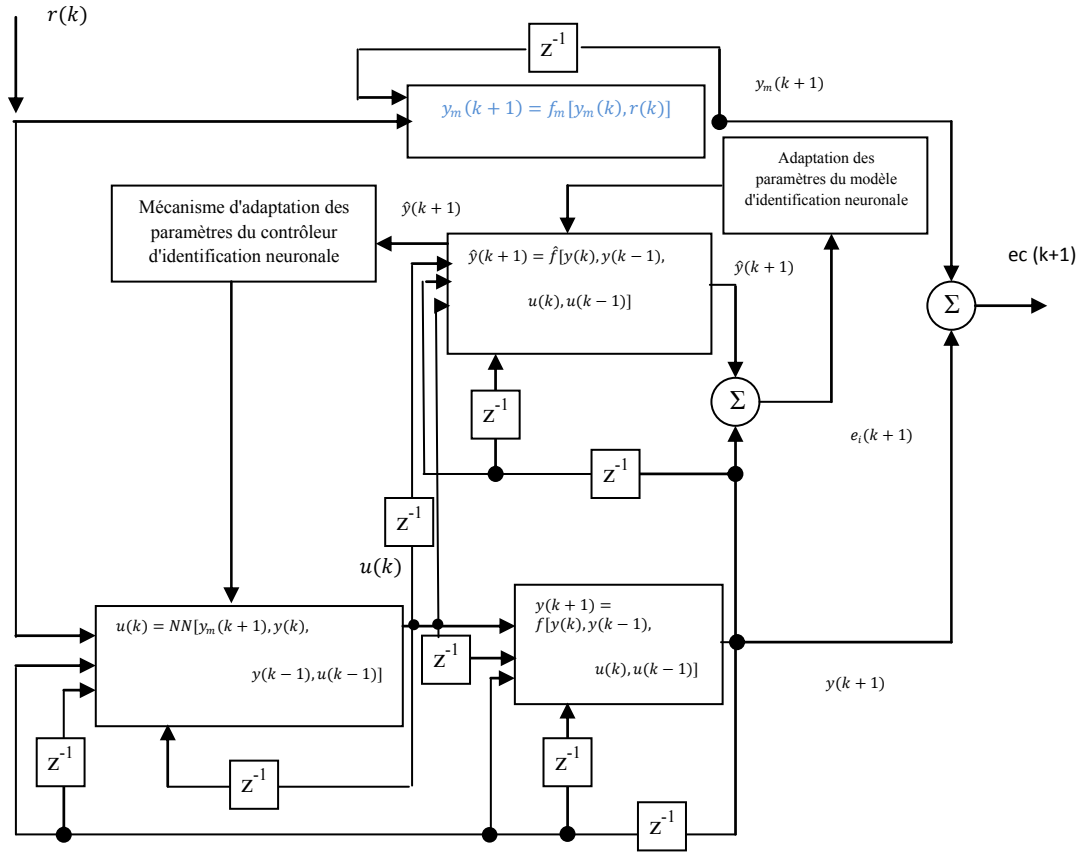


Figure (III.9) : Contrôle neuronal adaptatif indirect pour
Le cas d'un modèle d'identification du type IV

III.7 Conclusion

Dans ce chapitre, nous avons présenté les deux stratégies de la synthèse d'une commande neuronale des systèmes non linéaires en utilisant les modèles ainsi obtenus par la modélisation directe et inverse. D'abord, nous avons étudié le contrôle par modèle inverse, la méthode la plus simple de contrôle, ensuite nous avons donné un aperçu sur le contrôle adaptatif conventionnel, en citant les deux approches directe et indirecte. Puis nous sommes passés au problème du contrôle adaptatif des systèmes non linéaires, en introduisant le contrôle neuronal adaptatif. Pour ce dernier, l'approche indirecte est utilisée puisque l'approche directe pose un problème lorsque nous avons un système inconnu ou mal défini [6][19].

À partir des résultats de simulations nous allons juger les capacités de ces structures neuronales d'approximation et de génération des commandes nécessaires pour la réalisation des tâches désirées.

CHAPITRE IV

Contrôle neuronal adaptatif d’un bras manipulateur à deux degrés de libertés

IV.1 Introduction

La commande des robots manipulateurs est l'une des préoccupations majeures des recherches en robotique. En effet, un robot manipulateur est caractérisé par un comportement purement non linéaire, de plus, Certaines tâches qui lui sont confiées sont délicates et exigent une très grande précision sous des trajectoires rapides.

Dans le cas où le modèle exact du robot est parfaitement connu, plusieurs stratégies de commande peuvent être appliquées. Cependant, en pratique, cette condition idéale n'est jamais tout à fait remplie vu les différentes perturbations agissant sur le robot manipulateur, et les incertitudes du modèle, d'où la nécessité d'adapter la commande.

Durant ces trois dernières décennies, en vue d'améliorer les performances des robots manipulateurs, des recherches avancées ont permis de développer de nouvelles techniques de commande non linéaire telle que la commande neuronale adaptative pour les applications aux robots manipulateurs.

La problématique de la commande neuronale adaptative est basée sur la propriété d'approximation universelle des réseaux de neurone, En effet, ceux-ci sont capables d'approximer, avec un degré de précision arbitraire fixé, n'importe quelle dynamique non linéaire [6].

D'une manière générale, les contrôleurs neuronaux adaptatifs sont divisés en deux grandes classes : une s'appelle la commande neuronale adaptative directe et l'autre la commande neuronale adaptative indirecte. Dans la commande neuronale adaptative directe, le système neuronal est vu comme un contrôleur adaptatif. Cependant, dans la commande neuronale adaptative indirecte, les réseaux de neurones sont utilisés pour modéliser le système réel et le contrôleur est construit en supposant que les modèles neuronales représentent approximativement le vrai système, ces deux tâches se font séquentiellement.

Dans ce chapitre, nous appliquons la technique de commande neuronale adaptative pour la commande de la position d'un bras manipulateur à deux degrés de libertés.

IV.2 Modélisation

Pour développer une stratégie de commande performante pour un robot, il est impératif de connaître la cinématique et la dynamique du manipulateur considéré. Pour cela on est souvent amené à décrire les différentes relations mathématiques qui permettent de définir les mouvements de ce dernier dans l'espace.

Dans la pratique courante de robotique, la description du mouvement d'un robot manipulateur dans l'espace est réalisée en fonction du modèle géométrique, cinématique et dynamique.

IV.3 Structure mécanique des robots

La structure mécanique du robot manipulateur peut être divisée en deux parties distinctes comme le montre la figure (IV.1) [58] [59] :

- **Organe terminal** : les tâches qui sont dévolues aux robots sont très variées. Pour chaque opération ou travail spécifique, l'organe terminal prend un aspect particulier.
- **Elément porteur** : il est composé d'un ensemble de corps souples ou rigides liés par des articulations, servant à déplacer l'organe terminal d'une configuration à une autre.

Avant de décrire les relations géométriques entre les différents corps du robot, on définit les notions suivantes [60]:

- **Degré de liberté** : nombre de paramètres utilisés pour spécifier la configuration d'un élément de la chaîne cinématique par rapport à un autre.
- **Degré de mobilité** : toute articulation est caractérisée par son degré de mobilité m, c'est-à-dire le nombre de degrés de liberté entre deux corps successifs de la chaîne cinématique ($0 \leq m \leq 6$)
- **Espace articulaire** : représente l'état des corps composant le robot en fonction des variables articulaires, sa dimension n est égale au nombre de degrés de liberté du robot.
- **Espace opérationnel** : décrit la position et l'orientation de l'organe terminal du robot par rapport à un repère de référence, sa dimension $m \leq 6$.

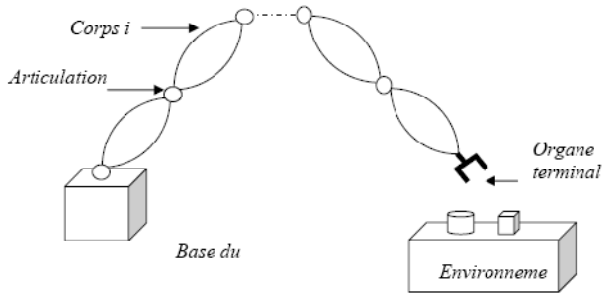


Figure IV.1 : Robot à chaine ouvert simple [58]

IV.4 Structure géométrique des robots

Parmi les diverses méthodes utilisées pour déterminer la position et l'orientation de l'organe terminal par rapport au repère de référence, la plus répandue est celle de Denavit-Hartenberg [59] **(**figure IV.2).

Pour exprimer le passage du repère R_{i-1} au repère R_i, on définit les paramètres géométriques suivants [3] :

- Translation selon X_{i-1}d'une distance a_i
- Rotation autour de X_{i-1}d'un angle α_i
- Translation selon Z_{i-1}d'une distance d_i
- Rotation autour de Z_{i-1}d'un angle $\bar{\omega}_i$

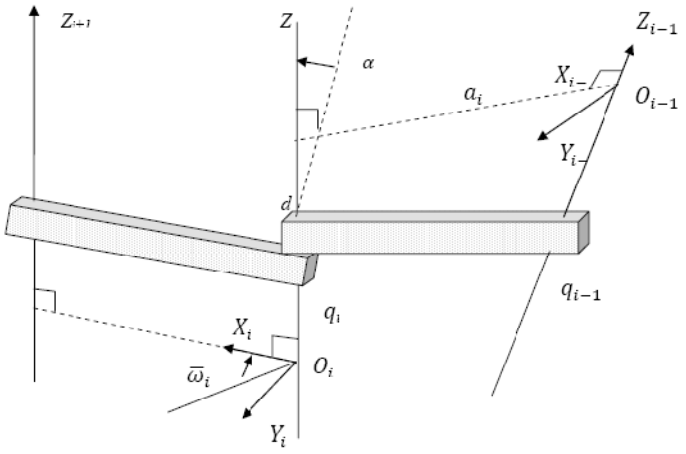


Figure IV.2 : Représentation Denavit-Hartenberg [58]

La matrice de transformation homogène est donnée comme suit [58][59]:

$$T_i^{i-1} = Trans(X_{i-1}, a_{i-1})Rot(X_{i-1}, \propto_{i-1})Trans(Z_i, d_i)Rot(Z_i, \bar{\omega}_i)$$

$$T_i^{i-1} = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & -c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\bar{\omega}_i & -s\bar{\omega}_i & 0 & 0 \\ c\bar{\omega}_i & c\bar{\omega}_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_i^{i-1} = \begin{bmatrix} c\bar{\omega}_i & -s\bar{\omega}_i & 0 & d_{i-1} \\ c\alpha_{i-1}s\bar{\omega}_i & c\alpha_{i-1}c\bar{\omega}_i & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\alpha_{i-1}s\bar{\omega}_i & s\alpha_{i-1}c\bar{\omega}_i & c\alpha_{i-1} & -c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_i^{i-1} = \begin{bmatrix} A_i^{i-1} & P_i^{i-1} \\ 0_3 & 1 \end{bmatrix} \quad \text{(IV.1)}$$

Avec :

$$A_i^{i-1} = \begin{bmatrix} c\bar{\omega}_i & -s\bar{\omega}_i & 0 \\ c\alpha_{i-1}s\bar{\omega}_i & c\alpha_{i-1}c\bar{\omega}_i & -s\alpha_{i-1} \\ s\alpha_{i-1}s\bar{\omega}_i & s\alpha_{i-1}c\bar{\omega}_i & c\alpha_{i-1} \end{bmatrix} \quad et \quad P_i^{i-1} = \begin{bmatrix} d_{i-1} \\ -s\alpha_{i-1}d_i \\ -c\alpha_{i-1}d_i \end{bmatrix}$$

Si on désigne par: T_n^0 la matrice de transformation reliant le repère R_n au repère R_0 alors :

$$T_n^0 = T_1^0 T_2^1 \ldots . T_n^{n-1} \quad \text{(IV.2)}$$

IV.5 Modèle géométrique

IV.5.1 Modèle géométrique direct

Le modèle géométrique direct permet de déterminer la position et l'orientation de l'organe terminal du manipulateur par rapport à un repère de référence en fonction des variables articulaires, le modèle s'écrit :

$$X = f(q) \quad \text{(IV.3)}$$

Où

$q = [q_1, q_2, \ldots q_n]^T \in R^n$: Vecteur des variables articulaires.

$X = [x_1, x_2, \ldots x_n]^T \in R^m$: Vecteur des variables opérationnelles, et m ≤ n.

n : le nombre de degrés de libertés du robot.

m : le nombre de degrés de libertés de l'espace opérationnel.

Par exemple, si le manipulateur se déplace dans l'espace on pose m=6 (trois coordonnées pour la position et trois coordonnées pour la rotation). S'il se déplace dans un plan on pose m=2 et si en plus on est concerné par la rotation on pose m=3.

La position de l'organe terminal peut être définie par des cordonnées cartésiennes, cylindriques ou sphériques. Le choix d'une structure particulière est guidé par les caractéristiques du robot, ainsi que par celle de la tâche à réaliser [58][59].

IV.5.2 Modèle géométrique inverse

Le modèle géométrique inverse permet de déterminer le vecteur des variables articulaires à partir du vecteur de coordonnées opérationnelles, le modèle s'écrit :

$$q = f^{-1}(X) \qquad \text{(IV.4)}$$

Parmi les méthodes utilisées pour déterminer le modèle géométrique inverse on cite :

- **Les méthodes géométriques** : permettent de déterminer le vecteur q par utilisation des transformations géométriques en prenant avantage de la structure particulière du manipulateur considéré.
- **Les méthodes algébriques :** permettent de déterminer le vecteur q en effectuant des transformations algébriques sur l'équation (IV.3). Parmi les méthodes utilisées on cite la méthode de Paul [58][59] qui consiste à multiplier successivement les deux membres du modèle géométrique direct par les matrices homogènes T_i^{i-1} avec $(i = 1, \dots., n-1)$ permettant ainsi d'isoler et d'identifier les variables articulaires l'une après l'autre.

Lors de la résolution du problème géométrique inverse on rencontre pratiquement les situations suivantes [58][59]

- ✓ Solutions en nombre fini, lorsqu'elles peuvent être calculées sans ambiguïté.
- ✓ Aucune solution possible lorsque la position désirée ne peut être atteinte par le manipulateur.
- ✓ Plusieurs solutions possibles lorsque le manipulateur est redondant ou lorsqu'il passe par une configuration singulière.

IV.6 Modèle cinématique

IV.6.1 Modèle cinématique direct

Le modèle cinématique direct permet de déterminer la vitesse de l'organe terminal dans l'espace opérationnel en fonction de la vitesse des variables articulaires.
Le modèle est décrit par l'équation :

$$\dot{X} = J(q)\dot{q} \qquad \text{(IV.5)}$$

$J(q) \in R^{m*n}$ est la matrice jacobienne.

L'une des méthodes utilisées pour le calcul de la matrice jacobéenne est la dérivation du modèle géométrique direct :

$$J(q) = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial q_n} & \cdots & \frac{\partial f_m}{\partial q_n} \end{bmatrix} \quad \text{(IV.6)}$$

IV.6.2 Modèle cinématique inverse

Le modèle cinématique inverse permet de déterminer la vitesse des variables articulaires en fonction de la vitesse des variables opérationnelles. Pour les manipulateurs non redondants *(n=m)*, le modèle s'écrit :

$$\dot{q} = J^{-1}(q)\dot{X} \quad \text{(IV.7)}$$

La solution de l'équation (IV.7) existe si *J* est de rang plein, cela est valable tant que le manipulateur ne passe pas par une configuration singulière. Pour les manipulateurs redondants, le modèle cinématique inverse admet plusieurs solutions possibles.

IV.7 Modélisation dynamique

IV.7.1 Modèle dynamique inverse

Le modèle dynamique inverse exprime les couples exercés par les actionneurs en fonction des positions, vitesses et accélérations des articulations. Les formalismes les plus utilisés pour le calcul du modèle dynamique inverse sont [58][59] :

- Formalisme de Lagrange.
- Formalisme de Newton-Euler.

Dans le présent travail, on s'intéresse au formalisme de Lagrange.

IV.7.1.1 Formalisme de Lagrange

Afin de pouvoir commander ou simuler les robots, il est nécessaire d'établir le modèle dynamique du robot, c'est à dire les équations liant les couples et les forces exercées par les actionneurs et ceux dues aux interactions avec l'environnement aux déplacements des axes.
On obtient ainsi un système d'équations différentielles non linéaires de la variable q.

Les équations de Lagrange opèrent à partir de l'énergie cinétique et l'énergie potentielle d'un système.

- **Cas général**

Soit l'équation l'Euler-Lagrange :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial q}\right) - \frac{\partial L}{\partial q} = \tau_i \qquad \text{(IV.18)}$$

Avec :

$L = K - V$ qui est appelé Lagrangien du système,

K est l'énergie cinétique du système,

V est l'énergie potentielle du système.

q_i : est la $i^{ème}$ coordonnée généralisée du système,

τ_i : est la force généralisée appliquée au $i^{ème}$ élément du système.

- **Application à la robotique**

Les variables articulaires q_i sont les coordonnées généralisées. La force généralisée τ_i est, soit un couple si l'articulation est rotoïde soit une force si l'articulation est prismatique.

- **Expression de l'énergie cinétique**

$$K = \sum_{i=1}^{n} \frac{1}{2} m_i {v_i}^T v_i + \frac{1}{2} {\omega_i}^T I_i \omega_i \qquad \text{(IV.19)}$$

Avec :

I_i : Matrice d'inertie du corps i,

m_i : Masse du corps i,

v_i : Vitesse linéaire du centre de gravité du corps i,

ω_i : Vitesse angulaire du corps i.

La matrice d'inertie I_i tout comme la vitesse ω_i sont exprimées dans un repère lié au corps i et dont l'origine est au centre de gravité du corps i.

$$I = \begin{bmatrix} \int (y^2 + z^2) dm & -\int xy dm & -\int xz dm \\ -\int xy dm & \int (x^2 + z^2) dm & -\int ya dm \\ -\int xz dm & -\int yz dm & \int (x^2 + y^2) dm \end{bmatrix} \qquad \text{(IV.20)}$$

Soient J_{vi} et $J_{\omega i}$ deux Jacobiens tels que :

$$v_i = J_{vi}(q)\dot{q} \quad Et \quad \omega_i = {R_{Oi}}^T J_{\omega i}(q)\dot{q}$$

Alors l'énergie cinétique du système peut s'écrire sous la forme :

$$K = \frac{1}{2}\dot{q}^T \sum_{i=1}^{n} [m_i J_{vi}(q)^T J_{vi}(q) + J_{\omega i}(q)^T R_{Oi}(q) I_i {R_{Oi}}^T J_{\omega i}(q)]\dot{q} \qquad \text{(IV.21)}$$

Ce qu'on écrit sous forme condensée :

$$K = \frac{1}{2}\dot{q}^T M(q)\dot{q} \qquad \text{(IV.22)}$$

La matrice *M(q)* est symétrique définie positive de dimension n*n, elle dépend de la configuration q du manipulateur.

Cette matrice est appelée matrice d'inertie du robot.

- **Expression de l'énergie potentielle**

La seule source d'énergie potentielle est la gravité. Soit g, le vecteur de gravité exprimé dans le repère de base.

On a :

$$V = g^T \sum_{i=1}^{n} O_{gi}\, m_i \qquad \text{(IV.23)}$$

O_{gi} : Coordonnées du centre de gravité du corps i dans le repère de base ;

m_i : Masse du corps *i*

- **Equation du mouvement (Application des équations d'Euler-Lagrange)**

On peut remarquer que l'énergie cinétique K peut se mettre sous la forme :

$$K = \frac{1}{2}\dot{q}^T M(q)\dot{q} = \frac{1}{2}\sum_{\mathrm{i,j}} [d_{ij}(q)\, \dot{q}_i \dot{q}_j \qquad \text{(IV.24)}$$

D'autre part, l'énergie potentielle *V* ne dépend que de la position q du robot :

$$V = V(q)$$

Le Lagrangien s'écrit donc :

$$L = K - V = \frac{1}{2}\sum_{\mathrm{i,j}} d_{ij}(q)\, \dot{q}_i \dot{q}_j - V(q) \qquad \text{(IV.25)}$$

On a :

$$\frac{\partial L}{\partial \dot{q}_k} = \frac{1}{2}\sum_{\mathrm{j}} d_{kj}(q)\,\dot{q}_j \qquad \text{(IV.26)}$$

Et donc

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) = \sum_j d_{kj}(q)\,\ddot{q}_j + \sum_j \frac{d}{dt} d_{kj}(q)\,\dot{q}_j = \sum_j d_{kj}(q)\,\ddot{q}_j + \sum_{i,j} \dot{q}_i\,\dot{q}_j$$

(IV.27)

De même :

$$\frac{\partial L}{\partial q_k} = \frac{1}{2}\sum_j \frac{\partial d_{ij}}{\partial q_k}\dot{q}_i\,\dot{q}_j - \frac{\partial V}{\partial q_k} \qquad \text{(IV.28)}$$

Ainsi, les équations d'Euler - Lagrange deviennent:

$$\sum_j d_{kj}(q)\,\ddot{q}_j = \sum_j [\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}]\,\dot{q}_i q_j + \frac{\partial V}{\partial q_k} = \tau_k \qquad k = 1, \dots, n$$

(IV.29)

En utilisant le fait que :

$$\sum_{i,j} \frac{\partial d_{kj}}{\partial q_i}\dot{q}_i\dot{q}_j = \frac{1}{2}\sum[\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j}]\,\dot{q}_i\dot{q}_j \qquad \text{(IV.30)}$$

On obtient :

$$\sum_{i,j}[\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}]\dot{q}_i\dot{q}_j = \sum[\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}]\,\dot{q}_i\dot{q}_j \qquad \text{(IV.32)}$$

On note :

$c_{ijk} = \frac{1}{2}[\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}]$ et $\phi_k = \frac{\partial V}{\partial q_k}$

N.B : $c_{ijk} = c_{jik}$

Finalement, les équations d'Euler-Lagrange deviennent :

$$\sum_j d_{kj}(q)\dot{q}_j + \sum c_{ijk}(q)\dot{q}_i\dot{q}_j + \phi_k(q) = \tau_k \qquad k = 1, \dots, n \qquad \text{(IV.33)}$$

Ce qui peut s'écrire sous forme matricielle :

$$M(q)\ddot{q} + N(q,\dot{q})\dot{q} + G(q) = \tau \qquad \text{(IV.34)}$$

Où

x_{kj}, le $(k,j)^{ème}$ Élément de *N*, est défini positif par : $x_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i$

Dans $N(q,\dot{q})$ les termes impliquant un produit. ${\dot{q}_1}^2$ Sont appelés centrifuges, et ceux impliquant un produit $\dot{q}_i\dot{q}_j$ avec $i \neq j$ sont les termes de Coriolis [9].

IV.7.2 Modèle dynamique direct

Le modèle dynamique direct exprime la position, la vitesse et l'accélération des articulations en fonction du couple appliqué [58][59]. Il est obtenu par inversion du modèle précédant, il s'écrit :

$$\ddot{q} = M^{-1}(q)(\tau - N(q,\dot{q}) - G(q) - \tau_f) \qquad \text{(IV.35)}$$

IV.8 Commande neuronale adaptative d'un bras manipulateur à 2DDL [21] [61][62][63]

Il est difficile d'obtenir les performances désirées avec des techniques où l'algorithme de commande est basé seulement sur le modèle dynamique explicite du bras manipulateur.

Une solution pour résoudre ce problème consiste à modéliser le bras manipulateur par un modèle neuronal fixé a priori, auquel on attache quelque fois une erreur d'approximation. Cependant, le modèle neuronal construit off line avec des paramètres fixes ne peut pas faire face au changement des paramètres. Par conséquent, les paramètres doivent être adaptés sans interruption lors du fonctionnement pour compenser cet effet indésirable. D'où l'appellation commande neuronale adaptative.

Une commande est dite adaptative si elle comporte des paramètres non fixés à l'avance mais modifiés en ligne.

Dans la technique adaptative neuronale indirecte, le modèle neuronal est utilisé pour estimer en premier lieu la dynamique du robot puis l'élaboration de la loi de commande est déduite en considérant les paramètres estimés du modèle neuronal comme de vrais paramètres, ces deux étapes se font séquentiellement.

La résolution du problème de la commande des robots manipulateurs nécessite la détermination d'un ensemble d'entrées articulaires (les couples τ) qui résulte par le suivi de l'organe effecteur d'une trajectoire désirée, spécifiée typiquement par des séquences de positions et de vecteurs d'orientation de l'organe effecteur *x* ou par une trajectoire continue.

IV.9 Etude et simulation (bras manipulateur à 2DDL) [9]

La simulation et l'implantation d'une commande sur un robot exige la connaissance des valeurs des paramètres géométriques et inertiels des corps du bras manipulateur, Le bras manipulateur est généralement considéré comme un ensemble de corps rigides connectés en série par des articulations, avec une extrémité au sol, et l'autre libre (effecteur ou élément terminal).

Soit le robot plan montré dans la figure (IV.3). Il est constitué de deux corps parallélépipédiques de longueur l_1 et l_2 dont la section est un carré de côté $d * d$ La masse volumique du matériau utilisé est ρ.

On note m_1 et m_2 les masses respectives des corps 1 et 2. [9]

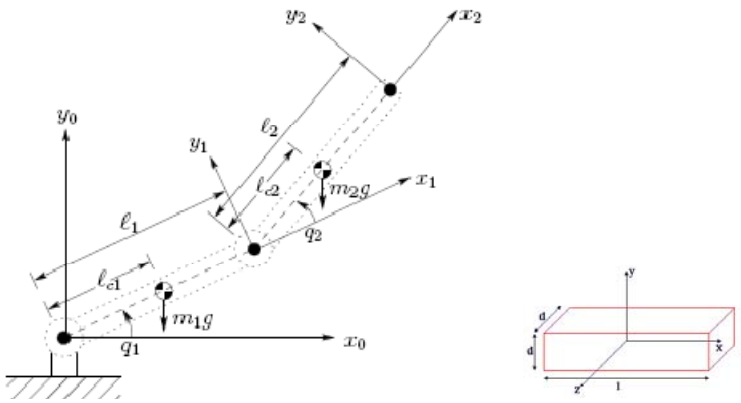


Figure (IV.3) : Structure d'un robot manipulateur à deux degrés de liberté [9]

Détermination de la matrice d'inertie d'un corps parallélépipédique :

$$I = \begin{bmatrix} \int (y^2 + z^2)dm & -\int xydm & -\int xzdm \\ -\int xydm & \int (x^2 + z^{2)}dm & -\int yadm \\ -\int xzdm & -\int yzdm & \int (x^2 + y^2)dm \end{bmatrix} \quad \text{(IV.36)}$$

On a :

$$\int (y^2 + z^2)dm = \int_{-d/2}^{d/2} \int_{-d/2}^{d/2} \int_{-d/2}^{d/2} (y^2 + z^2)\rho dxdydz$$

$$= \rho l \int_{-d/2}^{d/2} \int_{-d/2}^{d/2} (y^2 + z^2)dydz = \rho l \int_{-d/2}^{d/2} (\frac{d^3}{12} + dz^2)\, dz = \frac{\rho l d^4}{4} = md^2/6$$

(IV.37)

$$\int xydm = \int_{-d/2}^{d/2} \int_{-d/2}^{d/2} \int_{-d/2}^{d/2} xy\rho dxdydz$$

$$= \rho \int_{-d/2}^{d/2} \int_{-d/2}^{d/2} \left[\frac{x^2}{2} * y\right]_{-l/2}^{l/2} dydz = \rho \int_{-d/2}^{d/2} \int_{-d/2}^{d/2} y(\frac{l^2}{2} - \frac{l^2}{2})dydz = 0$$

(IV.38)

On procède de même pour tous les termes et on obtient :

$$I = \begin{bmatrix} \frac{md^2}{6} & 0 & 0 \\ 0 & m(d^2 + l^2)/12 & 0 \\ 0 & 0 & m(d^2 + l^2)/12 \end{bmatrix} \quad \text{(IV.39)}$$

On en déduit l_1 et l_2, les matrices d'inertie des corps 1 et 2 :

$$I_1 = \begin{bmatrix} \frac{m_1 d^2}{6} & 0 & 0 \\ 0 & m_1(d^2 + l^2)/12 & 0 \\ 0 & 0 & m_1(d^2 + l^2)/12 \end{bmatrix} \quad \text{(IV.40)}$$

$$I_2 = \begin{bmatrix} \frac{m_2 d^2}{6} & 0 & 0 \\ 0 & m_2(d^2 + l^2)/12 & 0 \\ 0 & 0 & m_2(d^2 + l^2)/12 \end{bmatrix} \quad \text{(IV.41)}$$

❖ **Energie cinétique**

$$K = 1/2\dot{q^T}\sum_{i=1}^{n}[m_i J_{vi}(q)^T J_{vi}(q) + J_{wi}(q)^T R_{0i}(q) I_i R_{0i}^T J_{wi}(q)]\dot{q} \qquad \text{(IV.42)}$$

▪ **Partie translation**

On a de manière évidente :

$$v_1 = \begin{bmatrix} -\frac{l_1}{2} sinq_1 & 0 \\ \frac{l_1}{2} cosq_1 & 0 \\ 0 & 0 \end{bmatrix} \dot{q} = J_{v1}\dot{q} \qquad \text{(IV.43)}$$

D'autre part, les coordonnées x_2 et y_2 du centre de gravité du corps 2 sont données par :

$$\begin{cases} x_2 = l_1 cosq_1 + \frac{l_2}{2}\cos(q_1 + q_2) \\ \dot{x}_2 = -l_1 sinq_1\dot{q}_1 - \frac{l_2}{2}\sin(q_1 + q_2)\,(\dot{q}_1 + \dot{q}_2) \end{cases} \qquad \text{(IV.44)}$$

D'où :

$$\begin{cases} y_2 = l_1 sinq_1 + \frac{l_2}{2}\sin(q_1 + q_2) \\ \dot{y}_2 = l_1 cosq_1\dot{q}_1 + \frac{l_2}{2} cos(q_1 + q_2)(\dot{q}_1 + \dot{q}_2) \end{cases} \qquad \text{(IV.45)}$$

On en déduit que :

$$J_{v2} = \begin{bmatrix} -l_1 sinq_1 - \frac{l_2}{2}\sin(q_1 + q_2) & -\frac{l_2}{2}\sin(q_1 + q_2) \\ l_1 cosq_1 + \frac{l_2}{2}\cos(q_1 + q_2) & \frac{l_2}{2}\cos(q_1 + q_2) \\ 0 & 0 \end{bmatrix} \qquad \text{(IV.46)}$$

▪ **Partie rotation**

Rappelons que les vitesses de rotation doivent être exprimées dans le même repère que celui qui a servi à la détermination de I_1 et I_2 c'est à dire un repère dont l'origine est au centre de gravité du corps considéré et dont les axes sont parallèles aux repères de Denavit-Hartenberg. En utilisant le fait que l'axe z de tous les repères est dans la même direction :

$$w_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \dot{q} = R_{01}^T J_{w1} \dot{q} \quad et \quad w_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \dot{q} = R_{02}^T J_{w2} \dot{q}$$

On en déduit que :

$$J_{w1}(q)^T R_{01}(q) I_1 R_{01}^T J_{w1}(q) = \begin{bmatrix} \frac{m_1(d^2 + {l_1}^2)}{12} & 0 \\ 0 & 0 \end{bmatrix} \quad \text{(IV.47)}$$

De même :

$$J_{w2}(q)^T R_{02}(q) I_2 R_{02}^T J_{w2}(q) = \begin{bmatrix} m_2(d^2 + {l_2}^2)/12 & m_2(d^2 + {l_2}^2)/12 \\ m_2(d^2 + {l_2}^2)/12 & m_2(d^2 + {l_2}^2)/12 \end{bmatrix} \quad \text{(IV.48)}$$

Si on note : $\begin{cases} i_1 = I_1(3,3) = \frac{m_1(d^2 + {l_1}^2)}{12} \\ \qquad\qquad et \\ i_2 = I_2(3,3) = \frac{m_2(d^2 + {l_2}^2)}{12} \end{cases}$ (IV.49)

L'énèrgie cinétique provenant de la rotation s'écrit :

$$\frac{1}{2} \dot{q}^T \begin{bmatrix} i_1 + i_2 & i_2 \\ i_2 & i_2 \end{bmatrix} \dot{q}$$

Finalement, la matrice d'inertie du robot est donnée par :

$$M(q) = m_1 J_{v1}^T J_{v1} + m_2 J_{v2}^T J_{v2} + \begin{bmatrix} i_1 + i_2 & i_2 \\ i_2 & i_2 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{m_1 l_1^2}{4} + m_2 \left(l_1^2 + \frac{l_2^2}{4} + l_1 l_2 cosq_2 \right) + i_1 + i_2 & m_2 \left(\frac{l_2^2}{4} + \frac{l_1 l_2}{2} cosq_2 \right) + i_2 \\ m_2 \left(\frac{l_2^2}{4} + \frac{l_1 l_2}{2} cosq_2 \right) + i_2 & \frac{m_2 l_2^2}{4} + i_2 \end{bmatrix}$$

(IV.50)

D'où :

$$\begin{cases} c_{111} = \frac{1}{2}\frac{\partial d_{11}}{\partial q_1} = 0 \\ c_{121} = c_{211} = \frac{1}{2}\frac{\delta d_{11}}{\delta q_2} = -\frac{1}{2}m_2 l_1 l_2 sinq_2 = h \\ c_{221} = \frac{\partial d_{12}}{\partial q_2} - \frac{1}{2}\frac{\partial d_{22}}{\partial q_1} = -\frac{1}{2}m_2 l_1 l_2 sinq_2 = h \\ c_{112} = \frac{\partial d_{21}}{\partial q_1} - \frac{1}{2}\frac{\partial d_{11}}{\partial q_2} = \frac{1}{2}m_2 l_1 l_2 sinq_2 = -h \\ c_{122} = c_{212} = \frac{1}{2}\frac{\partial d_{22}}{\partial q_1} = 0 \\ c_{222} = \frac{1}{2}\frac{\partial d_{22}}{\partial q_2} = 0 \end{cases} \quad \text{(IV.51)}$$

- **Energie potentielle**

$$\begin{cases} v_1 = \frac{m_1 g l_1}{2} sinq_1 \\ v_2 = m_2 g(l_1 sinq_1 + \frac{l_2}{2}\sin(q_1 + q_2)) \\ v = v_1 + v_2 \end{cases} \quad \text{(IV.52)}$$

D'où :

$$\begin{cases} \phi_1 = \frac{\partial v}{\partial q_1} = \left(\frac{m_1 l_1}{2} + m_2 l_1\right) gcosq_1 + \frac{m_2 l_2}{2} g\cos(q_1 + q_2) \\ \phi_2 = \frac{\partial v}{\partial q_2} = \frac{m_2 l_2}{2}\cos(q_1 + q_2) \end{cases} \quad \text{(IV.53)}$$

Finalement, on obtient les équations dynamiques suivantes :

$$\begin{cases} d_{11}\ddot{q_1} + d_{12}\ddot{q_2} + c_{121}\dot{q_1}\dot{q_2} + c_{211}\dot{q_2}\dot{q_1} + c_{221}\dot{q_2}^2 + \phi_1 = \tau_1 \\ d_{21}\ddot{q_1} + d_{22}\ddot{q_2} + c_{112}\dot{q_1}^2 + \phi_2 = \tau_2 \end{cases}$$

Donc le modèle dynamique de ce robot est donné par l'équation matricielle suivante :

$$\tau = M(q)\ddot{q} + N(q,\dot{q})\dot{q} + G(q) \quad \text{(IV.54)}$$

$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$: Vecteur des variables articulaires généralisées.

$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$: Vecteur des couples généralisés.

Matrice d'inertie :

$$M(q) = \begin{bmatrix} \frac{m_1 l_1^2}{4} + m_2\left(l_1^2 + \frac{l_2^2}{4} + l_1 l_2 cosq_2\right) + i_1 + i_2 & m_2\left(\frac{l_2^2}{4} + \frac{l_1 l_2}{2} cosq_2\right) + i_2 \\ m_2\left(\frac{l_2^2}{4} + \frac{l_1 l_2}{2} cosq_2\right) + i_2 & \frac{m_2 l_2^2}{4} + i_2 \end{bmatrix}$$

Matrice de Coriolis et centrifuge :

$$N(q,\dot{q}) = \begin{bmatrix} -\frac{1}{2} m_2 l_1 l_2 sinq_2 \dot{q}_2 & -\frac{1}{2} m_2 l_1 l_2 sinq_2 (\dot{q}_2 + \dot{q_1}) \\ +\frac{1}{2} m_2 l_1 l_2 sinq_2 \dot{q}_1 & 0 \end{bmatrix}$$

Vecteur des forces de gravité :

$$G(q) = \begin{bmatrix} \left(\frac{m_1 l_1}{2} + m_2 l_1\right) g cosq_1 + \frac{m_2 l_2}{2} g \cos(q_1 + q_2) \\ \frac{m_2 l_2}{2} \cos(q_1 + q_2) \end{bmatrix}$$

Avec les paramètres du modèle sont :

m_1 =15.91Kg, m_2=11.36 Kg, l_1 =0.432 m, l_2=0.432 m.

Les positions désirées sont donnée par :

$$\begin{cases} q_{d1}(k) = 2\cos\left(\frac{4\pi k}{3}\right) + \sin\left(\frac{2\pi k}{3}\right); \quad 0 \le k \le 10 \\ q_{d2}(k) = 1 - 2\cos\left(\frac{4\pi k}{3}\right) - \sin\left(\frac{2\pi k}{3}\right); \quad 0 \le k \le 10 \end{cases} \qquad \text{(IV.55)}$$

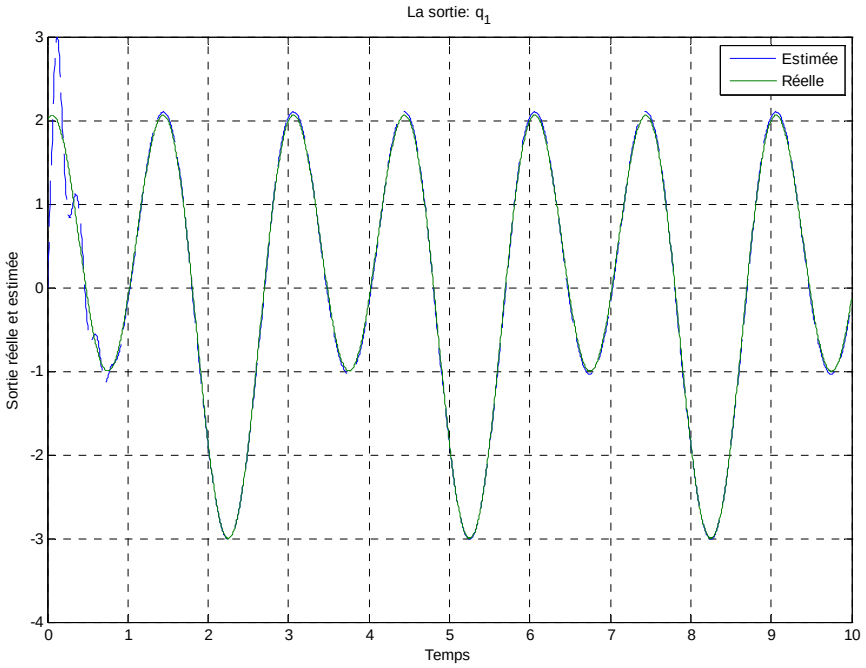


Figure (IV.4) : La première sortie, $q_1(-)$de référence et $\hat{q}_1(--)$ du modèle

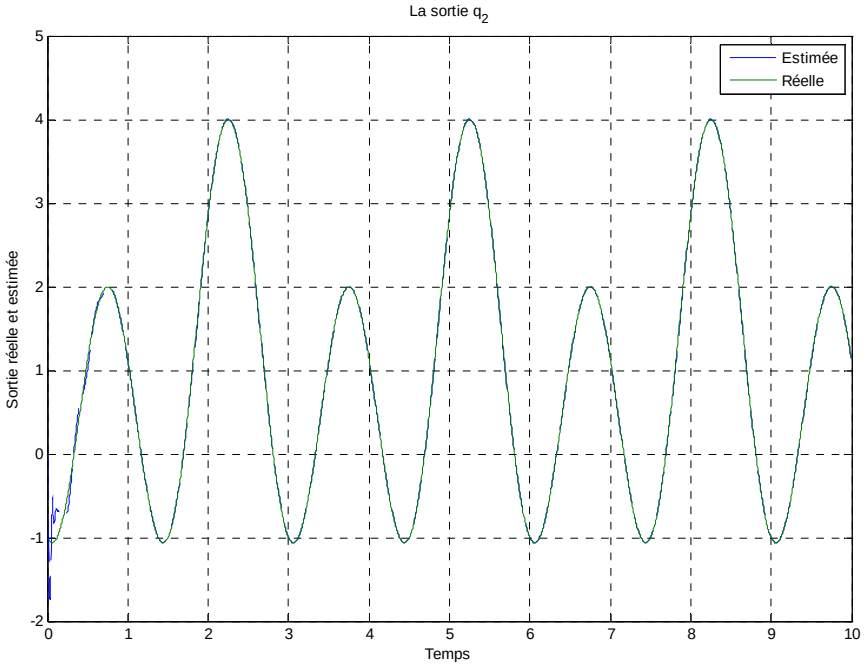


Figure (IV.5) : La première sortie, $q_2(-)$de référence et $\hat{q}_2(--)$ du modèle

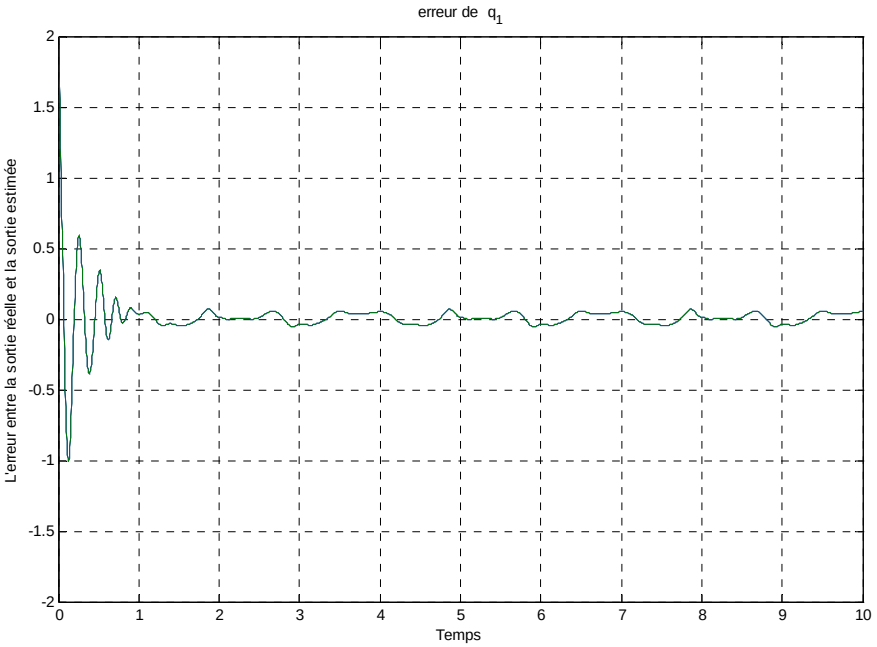


Figure (IV.6) : l'erreur entre q_1 et $\hat{q}_1$

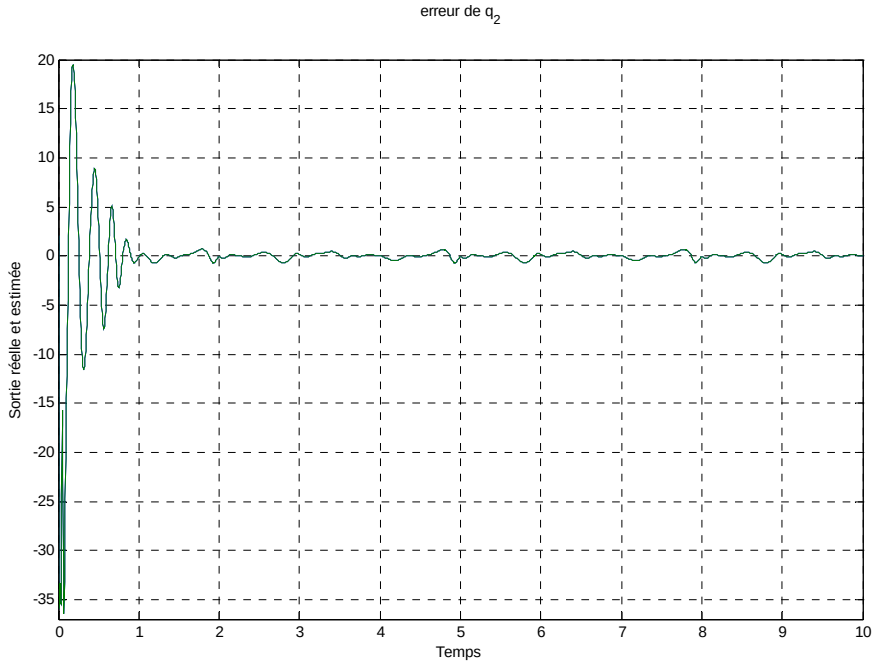


Figure (IV.7) : l'erreur entre q_2 et $\hat{q}_2$

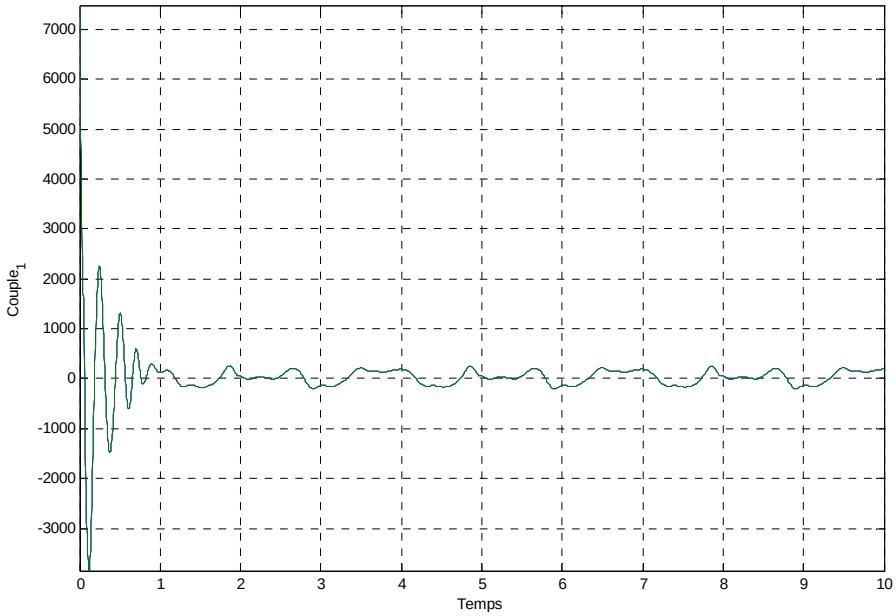


Figure (IV.8) : le couple τ_1

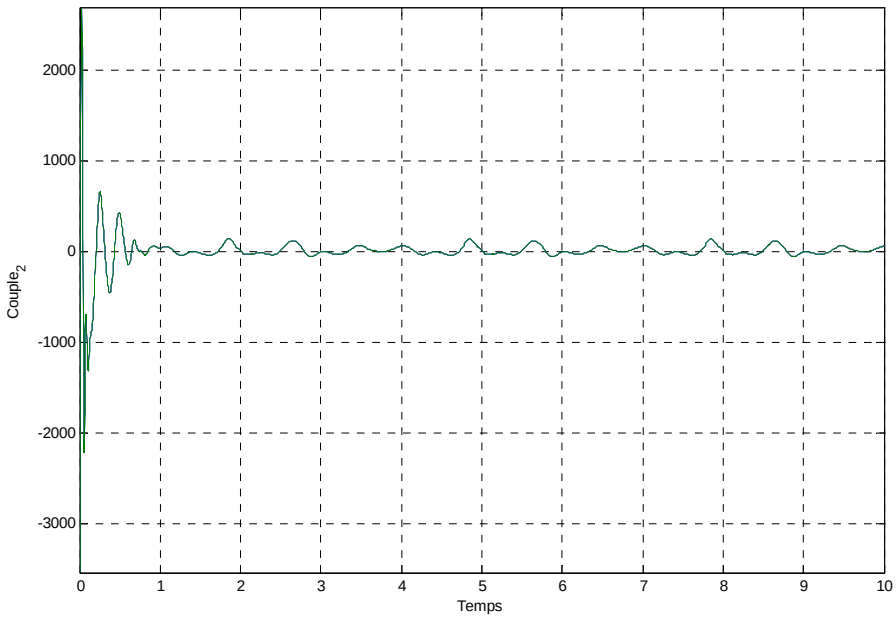


Figure (IV.9) : le couple τ_2

IV.10 Conclusion

Dans ce chapitre, on a appliqué la commande neuronale adaptative à un bras manipulateur à deux degrés de liberté. Les résultats de simulation montrent bien l'efficacité de modèle neuronal dans la commande de la position des robots. Nous avons aussi constaté que cette stratégie de commande ne présente pas uniquement une robustesse vis-à-vis des incertitudes, mais aussi une robustesse en présence d'un changement dans les paramètres du système.

Conclusion générale

Conclusion générale

Les propriétés d'apprentissage et de généralisation des réseaux de neurones artificiels, nous conduisent à étudier l'apport des techniques neuronales dans les problèmes d'identification et de commande des systèmes dynamiques non linéaires pour lesquels les techniques classiques se heurtent à des difficultés d'ordre pratique et théorique, et 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.

Arrivé au terme de ce mémoire et avant d'en évoquer les perspectives, nous résumons le contenu de notre travail établi comme suit :

En premier lieu la démarche suivie a conduit à un aperçu sur les notions de base de la théorie des réseaux de neurones.

Ensuite notre première application de l'approche neuronale était l'identification des systèmes non linéaires inconnus SISO et MIMO, les résultats de simulations obtenus ont démontrés que les modèles neuronaux sont de bons approximateurs universels pour les fonctions non linéaire inconnus, après cette étape nous avons traité le problème de l'identification neuronale avec rejection des perturbations, où les perturbations sont additives en entrée ou additives en sortie, les deux cas de représentation des perturbations, par un modèle linéaire ou par un modèle non linéaire, ont été considérés.

Le principe s'agit toujours d'augmenter l'ordre du système en question, 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.

Les résultats obtenus montrent qu'un modèle neuronal avec un bon apprentissage, s'adapte parfaitement à la dynamique d'un système avec rejet des perturbations.

Le contrôle neuronal était l'étape qui suit l'identification neuronale dans le contenu du travail, les propriétés d'apprentissage ont été utilisées pour réaliser des structures de contrôle dans lesquelles le contrôleur neuronal est capable de générer, d'une façon immédiate, les

commandes nécessaires pour la réalisation de la tâche désirée. Le contrôleur neuronal acquiert cette habilité d'association pendant la phase d'apprentissage.

Nous avons présenté deux structures de commande par réseaux de neurones, la première est basée sur le modèle neuronal inverse, la deuxième c'est le contrôle neuronal adaptatif avec modèle de référence, Dans ce type de contrôle, nous avons utilisé l'approche indirecte, où le calcul du contrôleur neuronal se base sur le modèle d'identification neuronale du système inconnu à contrôlé.

Le travail effectué présentés dans la dernière partie a pour objectif d'appliquer la technique de commande neuronale adaptative indirecte avec modèle de référence pour la poursuite de la trajectoire en position d'un bras manipulateur à deux degrés de libertés avec la génération des trajectoires dans l'espace de configuration. Cette stratégie devrait faire face d'une manière efficace aux variations des paramètres du système contrôlé.

Finalement, dans ce mémoire, on a supposé, pour accomplir les tâches d'identification et de contrôle, que les systèmes étudiés sont stables. Tous les systèmes considérés dans ce travail ont été pris de [6].

Nous pouvons affirmer que l'objectif, fixé au départ, a été atteint et que les réseaux de neurone constituent vraiment un moyen crédible pour le contrôle des systèmes dynamiques non linéaires (dans notre cas un bras manipulateur).

Et comme perspective, il serait intéressant d'appliquer d'autres techniques telles que la commande adaptative neuro-floue, les systèmes utilisent à la fois une base de règles pour les connaissances à priori et un algorithme d'adaptation, pour le réajustement des sous-ensembles flous. Cette combinaison permettra l'amélioration de performances obtenues.

Il est noter que les systèmes que nous avons pris sont considérés comme étant stables, une extension évidente et importante de l'approche neuronale est actuellement ressentie comme une nécessité pour le contrôle des systèmes instables, l'étude de la stabilité des structures des contrôleurs neuronaux font partie des perspectives d'avenir.

Comme la simulation ne peut en aucun cas remplacer une application réelle. Les systèmes de commande appliqués dans ce mémoire ne pourront être validé qu'une fois testés sur des robots réels, ainsi que la commande des robots manipulateurs en présence de contact et les paramètres de l'environnement.

Bibliographie

[1] S.A. Billings and I.J. Leontaritis, "Identification of nonlinear systems using parameter estimation techniques", Research report no. 117, Sept. 1980.

[2] S.A. Billings, "Identification of nonlinear systems-a survey", IEE Proc., vol. 127, Pt. D, no. 6, pp. 272-285, Nov. 1980.

[3] M. Boumahraz, "Identification et contrôle avec réseaux de neurones", Thèse de Magister, Université de Sétif 1993.

[4] C. Touzet, ‘’Les réseaux de neurones artificiels introduction au connexionnisme‘’, Juillet 1992.

[5] M.Parizeau, ‘’Réseaux de neurones‘’, Université de Laval, Automne 2004.

[6] K S. Narendra and K Parthasarathy, "Identification and control of dynamical systems using neural networks", IEEE Trans. On Neural Networks., vol. 1, no. 1, pp.4-26, Mar 1990.

[7] S. Mukhopadhyay and KS. Narendra, "Disturbance rejection in nonlinear systems using neural networks", IEEE Trans. On neural networks. vol. 4, no. 1, pp. 6372, Jan. 1993.

[8] S.Mohammed, DJ.Chaouch1, M.F Khelfi, "Commande neuronale inverse des systèmes non linéaires ", 4th International Conférence on Computer Integrated Manufacturing CIP’2007, 03-04 November 2007.

[9] M. W. Spong, Seth Hutchinson, and M. Vidyasagar, "Robot Dynamics and Control", Second Edition, January 28, 2004

[10] F. Moutarde, ‘’Introduction aux réseaux de neurones‘’, Ecole des Mines de Paris, Avril 2007.

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

[12] KS. Narendra, "Neural networks for identification and control", Center for Systems Science, Yale University, Dec. 1998.

[13] S. Chen and S. A. Billings, "Neural Networks for nonlinear dynamic system modeling and Identification", lnt. J. Control, vol. 56, no. 2, pp 319-346, Aug. 1992.

[14] Ioan D. Landau, "Identification des systèmes", Hermes, Paris, 1998.

[15] L. Ljung, "System identification: Theory for the user", Prentice Hall, Englewood Cliffs, New Jersey 1987.

[16] 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, 28 et 29 Septembre 2003.

[17] 1. Zhao, "System modeling, identification and control using fuzzy logic", Thèse de Docteur en sciences appliquées, UCL, CESAME, Belgique 1995.

[18] N. Golea, "Commande par logique floue " Thèse de Magister, Institut. D’Electronique, Université de Sétif, 1994.

[19] I. Abdelmalek, "Identification et commande floues des systèmes non linéaires", Thèse de Magister, Université de Batna 1999.

[20] C. Foulard, S. Gentil et G.P. Sandaraz, "Commande et régulation par calculateur numérique", 3e édition Eyrolles, 1982.

[21] Chang-Woo Park et Young-Wan Cho, "T–S Model based indirect adaptive fuzzy control using online parameter estimation", IEEE transactions on systems, man, and cybernetics—part b: cybernetics, vol. 34, no. 6, décembre 2004.

[22] I. Rivals, "Modélisation et commande de processus par réseaux de neurones"; application au pilotage d'un véhicule autonome », Thèse de Doctorat de l'Université Pierre et Marie Curie - Paris VI, Janvier 1995.

[23] L. Ljung, "System identification: Theory for the user", Englewood Cliffs, Prentice Hall, 1986.

[24] L. Ljung and T. Soderstrorn, "Theory and practice of recursive identification "MIT Press. Cambridge, Massachusetts, England, 1983.

[25] P. Eykhoff, "System identification: Parameter and state estimation", Eds John Wiley & Sons, USA, 1979.

[26] S. Rebouli et S. Hamoudi, "Identification et contrôle de la machine asynchrone par réseaux de neurones flous », mémoire d'ingéniorat, Institut d'électronique université Mohamed Boudiaf De M'Sila.

[27] A. Aouich, "Rejection des perturbations dans les systèmes non linéaires : étude comparative", Thèse de Magister, Université de M'Sila 2006.

[28] K. Benharira et N. Hadj Brahim, "Etude comparative des méthodes d'identification des systèmes non linéaires logique floue et réseaux de neurones ", mémoire d'ingéniorat, Institut d'électronique université Mohamed Boudiaf De M'Sila.

[29] K. Hornik, M. Stinchcombe, H. White, "Multilayer feedforward networks are universal approximators", Neural Networks vol. 2, pp. 359-366,1989.

[30] J. Sjëberg, et al. "Nonlinear black-box modelling in system identification: a unified overview", Automatica, 1995, 31(12), 16911724.

[31] M. Norgaard, O. Ra Vn, N. K. Poulsen and L. K. Hansen, "Neural networks for modeling and control of dynarnic systems", Spriger-Verlag, London 2000.

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

[33] Yao, X., "Evolvin Artificial Neural Networks", Proeedings of the IEEE 1999, 87(9) ,1423-1447.

[34] J.Panos, K.Antsaklis and M.Passino, ‘’Introduction to intelligent and autonomous control‘’, Kluwer Academic Publishers, ISBN: 0-7923-9267- 1, 1993.

[35] I. Rivals, L. Personnaz and G. Dreyfus, ‘’Modélisation, classification et commande par réseaux de neurones : principes fondamentaux, Méthodologie de conception et illustrations industrielles’’, 1995.

[36] G. Cybenko, ‘’Approximation by superposition of a sigmoidal function’’, Math. In: Control Signals System 2nd ed, 1989, pp.303-314.

[37] M.T.Hagan and H.B.Demuth, ‘’ Neural network design‘’, Thomson Asia Pte Ltd, 2nd ed 1996.

[38] T. Yabuta and T. Yamada, ‘’Possibility of neural networks controller for robot manipulators, Robotics and Automation’’, Proceedings, IEEE International Conference, vol.3, 1990, pp. 1686-1691.

[39] S. Yildirim, ‘’New neural networks for adaptive control of robot manipulators’’, Neural Networks, International Conference, vol.3, 1997, pp.1727 – 1731.

[40] P. Branštetter and M. Skotnica, ‘’Application of artificial neural network for speed control of asynchronous motor with vector control’’, Proceedings of international conference of Košice, EPE-PEMC, (2000, pp. 6-157-6-159).

[41] M.T. Wishart and R.G. Harley, ‘’Identification and control of induction machines using artificial neural networks’’. IEEE Transaction on Industry Applications, Vol. 31, No 3, 1995.

[42] E. Ronco and, P.J. Gawthrop, ‘’Neural networks for modelling and control’’, Techncal Report CSC-97008, Center for Systems and Control, Glasgow, 1997.

[43] P. Tai, H.A. Ryaciotaki-Boussalis and T. Kim, ‘’the application of neural networks to control systems: a survey’’, Signals, systems and computers, Record Twenty-Fourth Asilomar Conference on Vol.1, 1990.

[44] L. Yan and C.J. Li, ‘’Robot Learning control based on recurrent neural network inverse model’’, Journal. of Robotic Systems, Vol. 14, 1997, pp.199-212.

[45] G.W. Ng, ‘’Application of neural networks to adaptive control of nonlinear systems’’, Research Studies Press, Taunton, Somerset, England, 1997.

[46] G. Karsai, ‘’learning to control, some practical experiments with neural networks’’, In International Joint Conference On Neural Networks, vol. II, Seattle, 1991, pp. 701-707

[47] B. Widrow, ‘’Adaptive inverse control’’, In Proceedings of the IFAC Adaptive Systems in Control and Signal Processing Conference, Lund, 1988, pp.1-5.

[48] D. Psaltis, A.Sideris and A. Yamamura, ‘’Neural controllers’’, In International Neural Networks Conference, vol.IV, 1987, pp.551-558.

[49] C. Goutte and C. Ledoux, ‘’Synthèse des techniques de commande connexionniste’’, IBP-Laforia 1995/02: Rapport de Recherche Laforia, 1995.

[50] G.A. Pugh, ‘’Ship directional control by synthetic neural network’’, Proc of the American Control Conference ACC90, San Diego, 1990, pp.3028- 3029.

[51] Y. Ichikawa and T.Sawa, ‘’Neural network application for direct feedforward controller’’, IEEE Trans On Neural Networks, vol.3 No2, 1992, pp.224-231.

[52] L.X Wang, "Adaptive fuzzy systems and control: Design and stability analysis", Englewood Cliffs, NJ: Prentice Hail, 1994.

[53] K.J. Âstrôrn and B. Wittenmark, "Adaptive control", Addison-Wesley, 1989.

[54] 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. C. W. Anderson.

[55] K.J.Hunt, D. Sbarbaro, R. Zbikowski and P. J. Gawthrop, "Neural network for control systems - A survey," Automatica, vol. 28, no. 6, pp 1083-1112, Dec 1992.

[56] G.Dreyfus, "Réseaux de neurones, méthodologie et application", 2ème édition Eyrolles, 416p, 2004.

[57] D.Arar, A.Ghadhbane et A.Bensaadi, "Identification et contrôle des systèmes non linéaires par les réseaux de neurones ", mémoire d’ingéniorat, université de Batna, 1995.

[58] N.Azoui, "Commande non linéaire d’un bras manipulateur", Thèse de Magister, université de Batna 2009.

[59] Y. BAAZI, "Etude d'un manipulateur à 3 degrés de liberté application a la fonction de préhenseur", Thèse de magister, Institut d'électronique, Université de Batna, 2000.

[60] C. VIBET, "Robots Principes et Contrôle", Editions Ellipses, 1988.

[61] H. Chekireb, M. Tadjine et D. Bouchaffra, "Direct adaptative fuzzy control of nonlinear system class with applications", Control and intelligent systems, vol.31, No.2, 2003

[62] Jeffrey T. Spooner, Manfredi Maggiore, Raul Ordonez, Kevin M. Passino "Stable adaptive control and estimation for nonlinear systems: neural and fuzzy approximator techniques".2002.

[63] Y.Oulmas, "Commande Adaptative Floue de la Position et de l’Orientation des Robots Manipulateurs", Thèse de Magister, Laboratoire de Commande des Processus, ENP ElHarrach, Alger, 2006.