République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Abderrahmane MIRA de Bejaia

Faculté de Technologie

Département Génie Electrique

Mémoire de fin cycle

En vue de l'obtention du diplôme de Master

En électronique

Option : Automatique

Identification et Commande des Systèmes non Linéaires par les Techniques Neuronales

Réalisé par :

- Mr HABCHI Fares

Encadré par :

- M[lle] MEZZAH. S

Devant le jury composé de :

Présidente : M[me]. BELLAHSENE

Examinateur : M[r]. LEHOUCHE

Promotion 2012/2013

Remerciement

Je remercie Dieu de m'avoir donné tout le courage pour mener ce projet à terme.

Je tiens à remercie tous ceux qui ont contribué de prés ou de loin, à la réalisation de ce travail, en particulier M^{lle} MEZZAH Samia pour avoir dirigé ce travail et dont les critiques et les conseils m'ont été très précieux.

Mes sincères remerciements s'adressent aussi aux membres de jury d'avoir accepté de juger mon travail.

Je remercie aussi tous les enseignants, sans exception, car c'est grâce à eux que je suis arrivé à faire ce travail.

Sans oublies tous mes amis.

Dédicace

***J*e dédié ce travail,**

À ma mère et mon père

À mes deux frères, Abdelhalim,
Abdeslem et sa femme Vanessa

À ma tante O.Salima

À mes grands parents

À toute ma famille

Et à tous mes amis en particulier M.Aicha, et ses deux sœurs Wassila et Numidia, N.A.Hassiba, F.Youva, R.Sofiane, B.Amirouche, A.Lakhdar, et Zeddam.

Table des matières

Chapitre III : Contrôle neuronal des systèmes non linéaires

Chapitre IV : Contrôle neuronale adaptatif d'un bras manipulateur à 2 degrés de liberté

Table des figures

Tables des figures

Introduction générale

Introduction générale

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 la 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.

Les sciences de l'ingénieur font largement appel aux modèles non linéaires pour décrire les comportements dynamiques des systèmes physiques réels. Si les modèles non linéaires sont en mesure de décrire correctement les comportements non linéaires d'un système, ils peuvent néanmoins s'avérer, en fonction de leur complexité mathématique, difficilement exploitables dans un contexte de synthèse d'une loi de commande et/ou de mise en place d'une stratégie de diagnostic du système.

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

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.

En effet, les capacités d'identification des réseaux de neurones, en particulier ceux de type multicouche, 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 ces paramètres de telle sorte que sa sortie s'approche le plus possible de celle du système inconnu.

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, principe et fonctionnement.

Le deuxième chapitre détaille l'identification des systèmes non linéaire avec les réseaux de neurones.

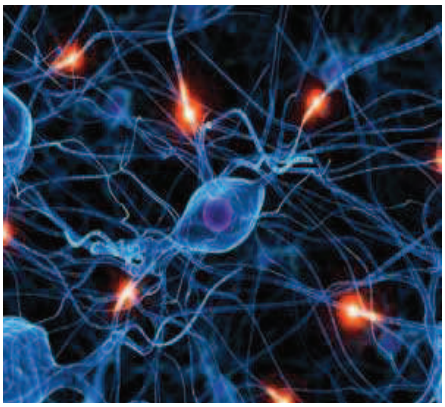
Le troisième chapitre traite la commande des systèmes non linéaires avec réseaux de neurones, commande avec modèle inverse, et la commande adaptative.

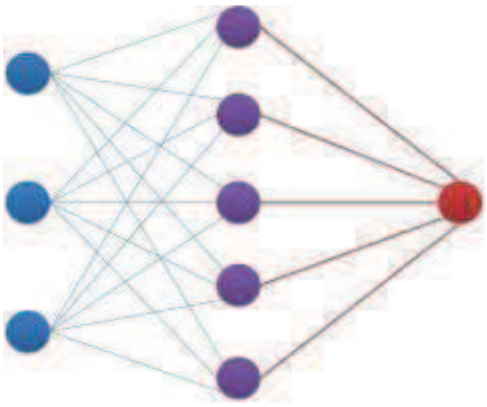
Chapitre quatre, et le dernier chapitre de notre travail, nous présentons l'identification, et le contrôle neuronal adaptatif d'un bras manipulateur 2 degrés de liberté

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

Chapitre I :

Les réseaux de neurones artificiels





I.1 Introduction

Le terme de réseaux de neurones formels (ou artificiels) fait rêver certains, et fait peur à d'autres ; la réalité est à la fois plus prosaïque et rassurante. Les réseaux de neurones sont 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.

De nombreuses applications sont opérationnelles, que ce soit dans le milieu de la recherche ou dans l'industrie.

Les réseaux de neurones artificiels ont des racines en neurobiologie. La structure et la fonctionnalité des réseaux neurones motivée par l'architecture du cerveau humain. Un réseau de neurones se compose des couches d'unités de traitement simples couplées par intercommunication pesée. Avec le développement de l'informatique, le progrès significatif dans la recherche de réseau de neurones a été accompli. Ces dernières années un certain nombre de réseaux neurone a été proposé. [1]

I.2 Définition d'un neurone

I.2.1 Neurone biologique

Réseaux de neurones artificiels sont inspirés sur les processus biologiques pour le traitement de l'information, y compris en particulier le système nerveux et de son unité de base, le neurone. Les signaux sont reproduits sous la forme de différences de potentiel entre l'intérieur et l'extérieur des cellules. Les composants de cellules neuronales sont représentés par la figure I.1. Dendrites introduisent des signaux en provenance d'autres neurones dans le corps cellulaire ou soma, éventuellement en multipliant chaque signal d'entrée par un coefficient de pondération. Dans le soma, la capacité de la cellule intègre les signaux qui s'accumulent dans l'axone monticule (axon hillock). Une fois que le signal combiné dépasse un certain seuil de la cellule, un signal potentiel d'action est transmis à travers l'axone. L'axone se connecte via les synapses avec les dendrites des neurones suivants. Les synapses fonctionnent grâce à la décharge de neurotransmetteurs dans les lacunes intercellulaires, et peuvent être soit excitateurs (tendant à tirer le neurone) ou inhibiteurs (tendant à empêcher l'activation du prochain neurone). [2]

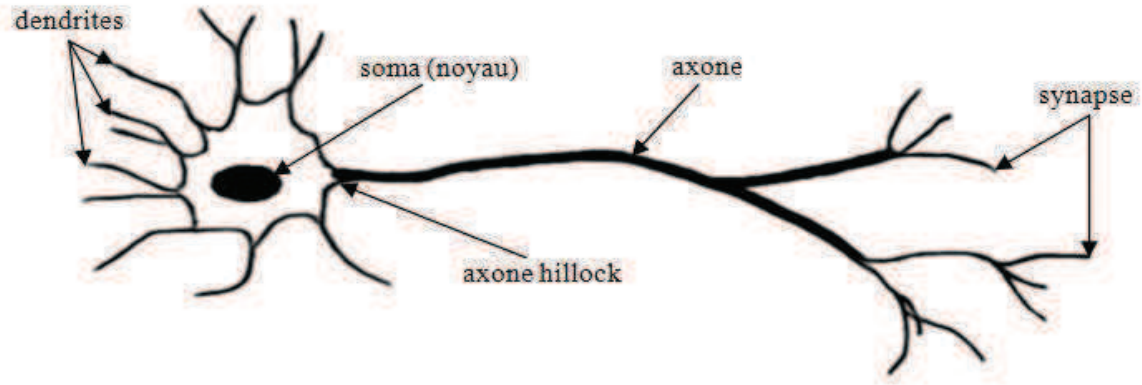


Figure I.1 Neurone biologique [2]

I.2.2 Neurone formel

Un neurone est une unité de traitement de l'information, est la base fondamentale du fonctionnement d'un réseau neurones. Le modèle du neurone est illustré dans figure I.2. Il y a trois éléments de base dans le modèle de neurone : les liens de connexion, un additionneur, et une fonction d'activation [1]

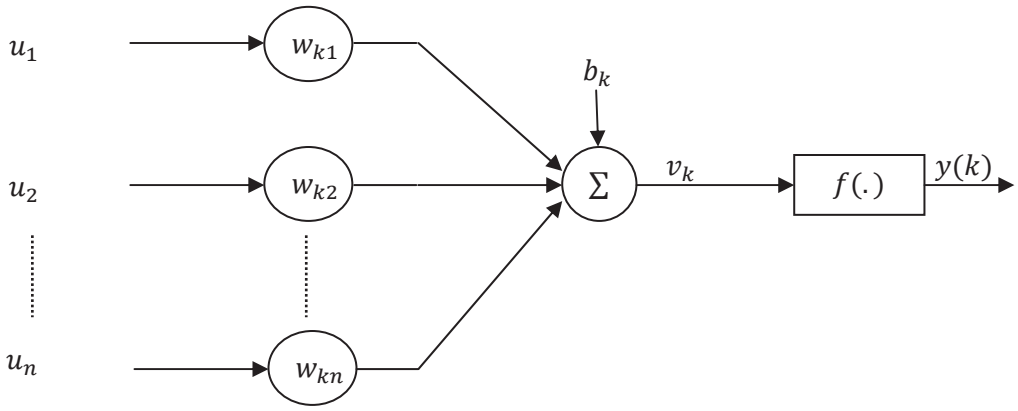


Figure I.2 Modèle d'un neurone artificiel [1]

Chaque lien de connexion est caractérisé par un poids. Particulièrement, un signal « uj » à la « j-éme » entré connecté au « k-éme » neurone, est multiplié par le poids « wkj ». Pour les indices du poids « wkj », le premier indice « k » se rapporte au neurone et le deuxième indice « j » se rapporte à l'entrée à laquelle le poids se réfère, pour le biais b s'appel le seuil d'activation.

L'additionneur, il fait la somme dans l'ordre des signaux d'entrée qui ont été multipliés par les poids qui lui correspondent

La fonction d'activation limite l'amplitude de la sortie du neurone à une certaine valeur finie. Et les fonctions de transfert couramment utilisé sont représenté dans le tableauI.1, et les plus utilisés sont : fonction seuil figure I.3, fonction linéaire figure I.4, et la fonction sigmoïdale figure I.5.

En mathématique on peut résumer ce modèle par deux équations I.1 et I.2

$$v_k = \sum_{j=1}^{n} w_{kj} u_j + b_k \qquad \text{I.1}$$

$$y_k = f(v_k) \qquad \text{I.2}$$

I.3 fonction de transfert (d'activation)

I.3.1 fonction de transfert couramment utilisé :

La fonction	L'équation mathématique	Fonction Matlab
Seuil	$f(n) = \begin{cases} 1 \; si \; n \geq 0 \\ 0 \; si \; n < 0 \end{cases}$	hardlim
Seuil symétrique	$f(n) = \begin{cases} 1 \; si \; n \geq 0 \\ -1 \; si \; n < 0 \end{cases}$	hardlims
Linéaire	$f(n) = n$	purelin
Linéaire saturée	$f(n) = \begin{cases} 0 \; n < 0 \\ n \; si \; 0 \leq n \leq 1 \\ 1 \; si \; n > 1 \end{cases}$	satlin
Linéaire saturée symétrique	$f(n) = \begin{cases} -1 \; n < -1 \\ n \; si \; -1 \leq n \leq 1 \\ 1 \; si \; n > 1 \end{cases}$	satlins
Linéaire positive	$f(n) = \begin{cases} 0 \; si \; n < 0 \\ n \; si \; n \geq 0 \end{cases}$	poslin
Sigmoïde	$f(n) = \dfrac{1}{1 + e^{-n}}$	logsig
Sigmoïde symétrique	$f(n) = \dfrac{1 - e^{-n}}{1 + e^{-n}}$	tansig
Tangente hyperbolique	$f(n) = \dfrac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$	tanh

Tableau I.1 Fonction de transfert $y = f(n)$ [1] [2]

I.3.2 Représentation des fonctions de transfert les plus utilisé : [2]

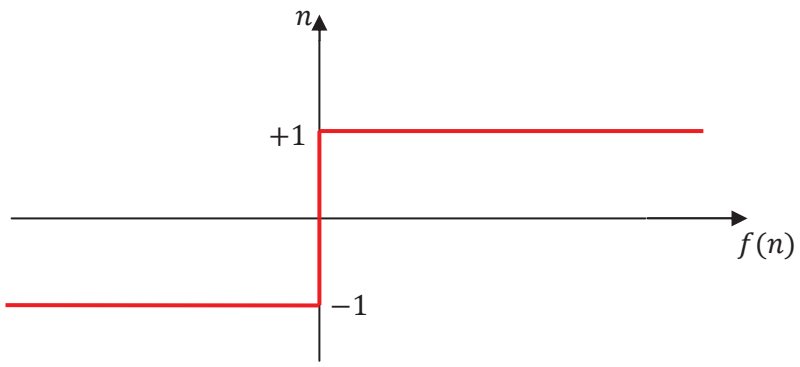


Figure I.3 fonction de transfert « seuil symétrique »

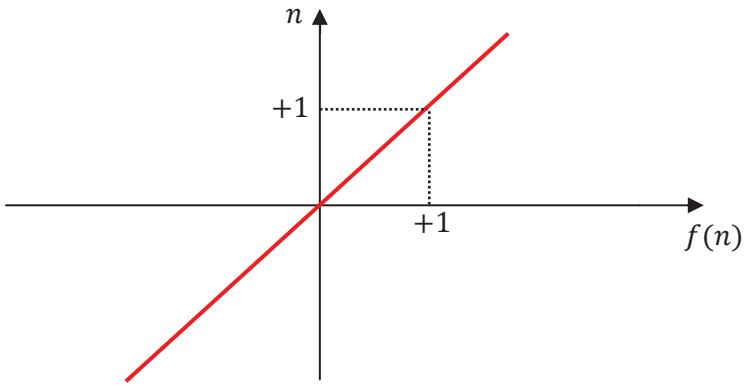


Figure I.4 fonction de transfert « linéaire »

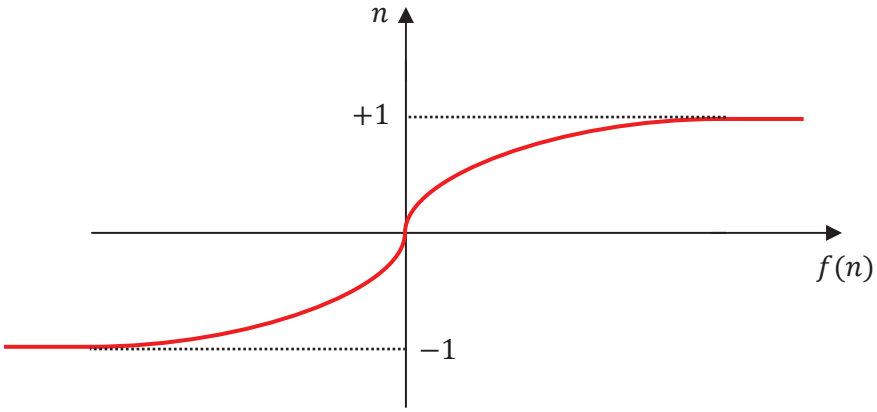


Figure I.5 fonction de transfert « sigmoïde symétrique »

I.4. Architecture de réseau de neurones artificiel

Un réseau de neurones artificiel est organisé sous forme de couche, chaque couche est composée des neurones (nœuds), chaque couche est connectée à la couche suivante.

Ces dernières années un certain nombre d'architectures de réseau de neurones artificiel a été proposé. Dans [1], trois classes différentes des architectures de réseau (ou structures) sont proposées : réseaux à une seule couche, réseaux récurrents, et réseaux multi couches

I.4.1. Réseaux de neurones récurrents :

Un réseau de neurones récurrent a au moins une boucle de retour, le réseau récurrent peut se composer d'une seule ou plusieurs couches de neurones, et de chaque neurone de sortie peuvent introduire son signal aux entrées du réseau. Une classe des réseaux récurrents avec des neurones cachés est illustrée dans figure I.6. Dans la structure des réseaux de neurones récurrents, la boucle de retour provient des neurones de sortie, et même des

neurones cachés. La présence des boucles de retour dans un réseau récurrent a un impact profond sur les performances du réseau. [1]

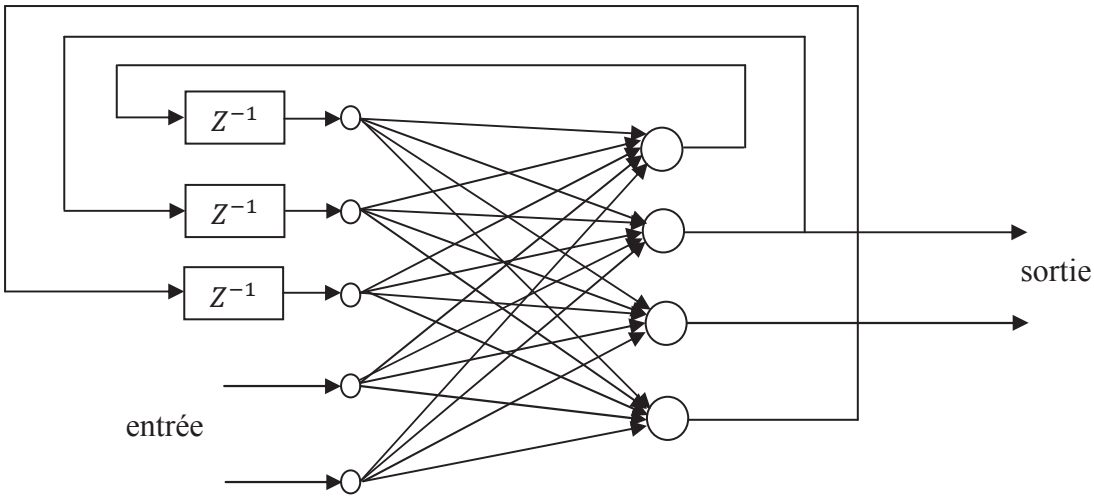


Figure I.6 Architecture d'un réseau de neurones récurrent

I.4.2. Réseaux de neurones à une seule couche

Un réseau qui a une couche d'entrée et une couche de sortie s'appelle un réseau à une seule couche, comme montre la figure I.7, la couche d'entrée est composée de cinq nœuds et la couche de sortie de quatre nœuds [1]

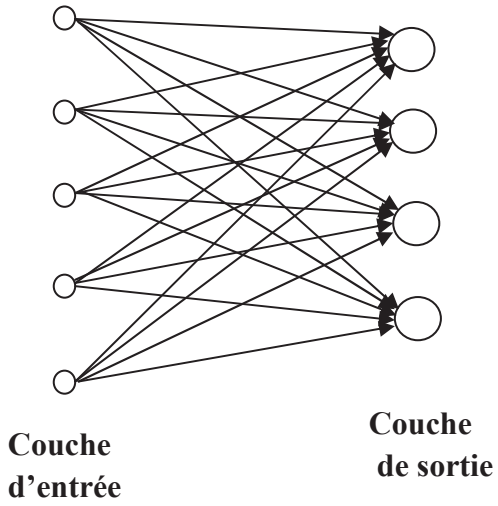


Figure I.7 Architecture d'un réseau de neurones à une seule couche

I.4.3. Réseaux de neurones multi couches

Un réseau multicouche, a une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie, chaque entrée est connectée a tous les neurones de la première couche, les sorties de neurones de cette couche est connecté aux neurones de la couche suivante, et ainsi de suite, la dernière couche est appelée couche de sortie comme montre la figure I.8 la couche d'entrée est composée de cinq nœuds, la couche cachée de quatre nœuds, et la couche de sortie de deux nœuds [1]

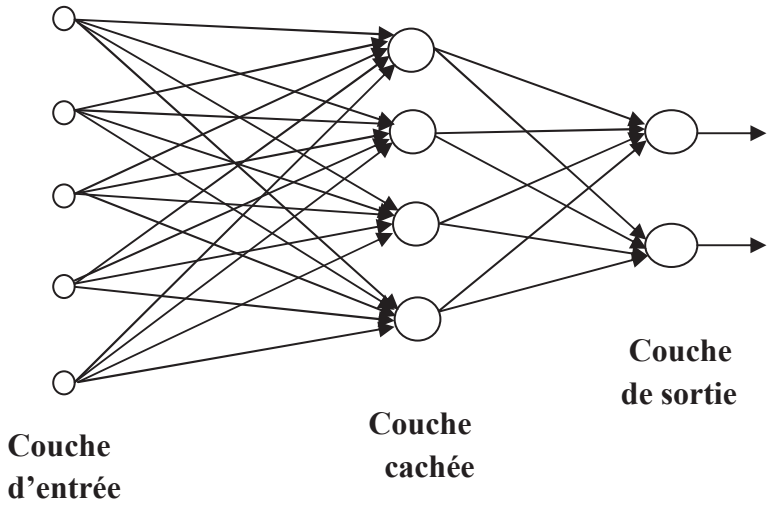


Figure I.8 Architecture d'un réseau de neurones multi couche

I.5. Apprentissage

L'apprentissage d'un réseau de neurones artificiel est induit par une procédure itérative d'ajustement ou d'adaptation de ses paramètres internes au moyen d'un processus de stimulation par l'environnement, cette procédure d'adaptation est décrite par un algorithme d'apprentissage. Il en existe de multiples formes d'adaptation qui se distinguent par la nature de la connaissance de l'environnement acquise par le réseau. Ainsi, le comportement d'un même réseau diffère selon l'algorithme d'apprentissage utilisé pour ses paramètres. [3]

I.5.1. Apprentissage supervisé

L'apprentissage supervisé représenté dans la figure I.9 suppose l'existence d'un expert (ou éducateur qui possède une connaissance innée de l'environnement. Le rôle de l'expert est de fournir les informations relatives à l'environnement nécessaire à l'apprentissage du réseau, sous forme d'un ensemble d'exemples composé de stimuli (entrée) auxquels sont associées des réponses désirées (ou comportements souhaités) [3]

En pratique, les connaissances de ce professeur prennent la forme d'un ensemble de m couple de vecteurs d'entrée et de sortie que nous noterons$\{(p_1, d_1), (p_2, d_2), \dots, (p_m, d_m)\}$, p_i désigne une entrée et d_i la cible pour cette entrée, c'est-à-dire les sorties désirées du

réseau. Chaque couple (p_i, d_i) correspond donc à un cas d'espèce de ce que le réseau devrait produire (la cible) pour une entrée donnée. Pour cette raison, l'apprentissage supervisé est aussi qualifié d'apprentissage par exemple [3]

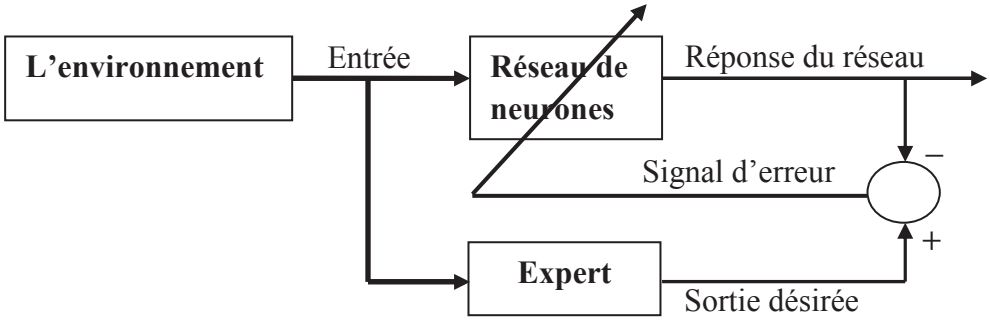


Figure I.9 Architecture d'apprentissage supervisé

I.5.2. Apprentissage par renforcement

L'apprentissage par renforcement est une forme d'apprentissage supervisé moins contraignante consiste à instruire ou entraîner le réseau par tâtonnement en procédant par essais et erreurs. Le réseau est alors stimuli par l'environnement et ses réponses sont sanctionnées ou récompensées afin de l'inciter à adopter le bon comportement [3]

Lorsqu'une action (décision) prise par le réseau engendre un indice de satisfaction positif, alors la tendance du réseau à prendre cette action doit être renforcée. Autrement, la tendance à prendre cette action doit être diminuée.

I.5.3. Apprentissage non supervisé

Contrairement à l'apprentissage supervisé effectué sous contrôle d'un expert, l'apprentissage non supervisé (figure I.9) est autodidacte. L'ensemble des exemples d'apprentissage ne comprend que des entrées. Aucune réponse désirée n'est associée [3]

C'est-à-dire qu'on ne dispose ni d'un signal d'erreur ni d'un indice de satisfaction, comme dans le cas par renforcement. Nous ne disposons donc que d'un environnement qui fournit des entrées, et d'un réseau qui doit apprendre sans intervention externe. En assimilant les entrées de l'environnement à une description de son état interne. La tache du réseau est alors de modéliser cet état le mieux possible. Pour y arriver, il importe d'abord de définir une mesure de la qualité pour ce modèle, et de s'en servir par la suite pour optimiser les paramètres libres du réseau, c'est-à-dire ses poids synaptiques. La figure I.10 illustre ce type d'apprentissage

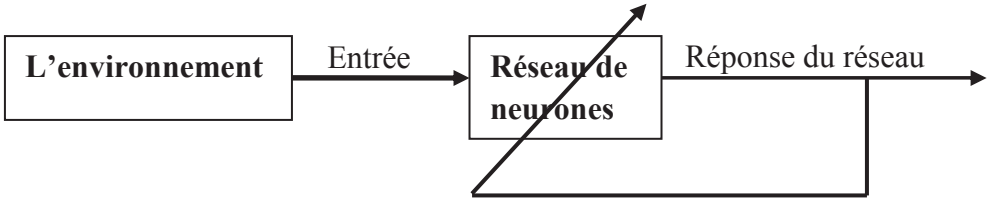


Figure I.10 Architecture d'apprentissage non supervisé

I.6. Algorithme d'apprentissage de "rétropropagation du gradient"

C'est un algorithme itératif, car le processus est répété jusqu'à ce que le réseau atteigne un état stable. Il est aussi un algorithme d'apprentissage par essai et erreur, c-à-dire il utilise un critère d'erreur et un algorithme de la rétropropagation [3]

On ajuste les différents poids de connexion par le calcule d'erreur à chaque instant pour les sorties de réseau avec la sortie désirée, pour l'erreur on utilise le critère de moindre carrée (équation I.3) [11]

$$e(k) = \frac{1}{2}\sum_{j=1}^{NL}\left(S_j^L(k) - y_j(k)\right)^2 \qquad I.3$$

Avec

NL : Nombre de neurones de la couche de sortie

$y_j(k)$: Est la sortie j désirer à l'instant k

S_j : La sortie j du réseau de neurones

Pour l'adaptation des poids

$$w_j^l(k+1) = w_j^l(k) - \mu g_j^l(k) \qquad I.4$$

Avec

μ : Le taux d'apprentissage

$g^l(k)$: Le gradient correspond à la matrice w^l (l correspond à la couche)

En effet, il considère un gain constant lors de la variation des paramètres selon le gradient.

I.7. 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.

Un type de réseau neuronal est défini par sa topologie, sa structure (nombre de couches et la nature de connexion entre les couches) et son algorithme d'apprentissage.

Chapitre II :

Identification neuronal des systèmes non linéaires

II.1. Introduction

Les systèmes réels sont difficile a étudie, donc on est amener à les représenter mathématiquement pour pouvoir les commander.

Il y a un intérêt considérable pour ces dernières années en explorant l'application des réseaux de neurones artificiels pour l'identification. L'identification des systèmes exige la sélection d'une classe de la fonction (ou modèle) pour approximer le comportement entrée-sortie des systèmes de la meilleure façon possible. Dans beaucoup de situations, telles que l'identification des systèmes dynamiques, le rendement du système physique à modeler est une fonction des entrées et des sorties antérieures. [4]

On va décrire brièvement les différentes méthodes, et les étapes de la conception d'un modèle on utilisant les réseaux de neurones artificiels MLP

II.2. Différent type de modèle

II.2.1. Modèle « boite noire» :

Ce modèle est celui où l'on ignore tout ou une grande partie des phénomènes mis en jeu. Dans ce cas-là, on se contente d'une description mathématique sans lien apparent avec la réalité physique [5]

II.2.2. Modèle de « connaissance »

Pour définir la structure du modèle, on peut s'aider des lois de la physique (mécanique, thermodynamique, électricité, …). Remarquons au passage que ces ''lois'' sont en fait des modèles essayant de décrire mathématiquement la nature. L'ensemble des équations ainsi établies constitue ce qu'on appelle un modèle de connaissance. Ceci est bien sûr le cas idéal rarement rencontré en pratique [5]

II.3. Représentation des systèmes non linéaires

Les systèmes dynamiques non linéaires peuvent être décrits par les équations suivantes :

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

$$y(k+1) = h[x(k)] \qquad \text{II.2}$$

Où $u(k), y(k) \in \mathbb{R}^m$ et $x(k) \in \mathbb{R}^n$ représentent l'entrée, la sortie, et le vecteur d'état, respectivement, à l'instant k. La fonction $f: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ et $h: \mathbb{R}^n \rightarrow \mathbb{R}^m$

Si m = 1, le système dispose d'une entrée unique et une sortie unique (SISO). Si m >1, est un système multi entrée, multi sortie (MIMO). [6]

Bien que le modèle état-sortie est assez général, tous les états du système ne sont pas disponibles généralement, les systèmes non linéaires ARMA (ou NARMA modèle) prédit la nouvelle sortie comme une fonction non linéaire des entrées et des sorties passées comme montre l'équation suivante: [7]

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

II.4. Identification des systèmes

L'identification est la détermination, sur la base de la connaissance d'un nombre fini d'entrées-sorties du système, d'un modèle appelé modèle d'identification, qui soumit aux mêmes entrées que le système fournit des sorties suffisamment proches de celui-ci [8]

L'identification consiste à déterminer un ensemble d'équations –un modèle- décrivant le mieux possible le procédé. Il y a deux étapes dans le travail la première consiste à fixer la forme des équations, c'est l'étape **qualitative,** ou caractérisation, la seconde consiste à trouver les valeurs numériques des coefficients qui interviennent dans ces équations, c'est l'étape **quantitative,** ou estimation des paramètres. Ces valeurs numériques sont déterminées pour que le comportement de modèle soit le plus proche de celui du système[5]

II.5. Les structures d'identification

La sortie d'un système dynamique dépend de son entrée et de son état ultérieur, c'est pour cela qu'il existe deux classes de modèle d'identification [8]

II.5.1. Identification parallèle

Dans le cas d'un système dynamique à temps discret, la sortie du modèle est calculée à partir de ses entrées et sorties passées

$$\hat{y}(k+1) = NN[\hat{y}(k), \hat{y}(k-1), \dots, \hat{y}(k-n+1); u(k), u(k-1), \dots, u(k-m+1)] \quad \text{II.4}$$

Certaines précautions doivent être prises lors de l'identification, la principale étant d'utiliser des entrées bornées et sorties bornées, le système reste stable, il est dit BIBO (Bonded Input Bonded Output).

L'inconvénient de l'identification parallèle est que même avec un système, rien ne garantie que les paramètres vont converger et que $e_i \rightarrow 0$ [8]

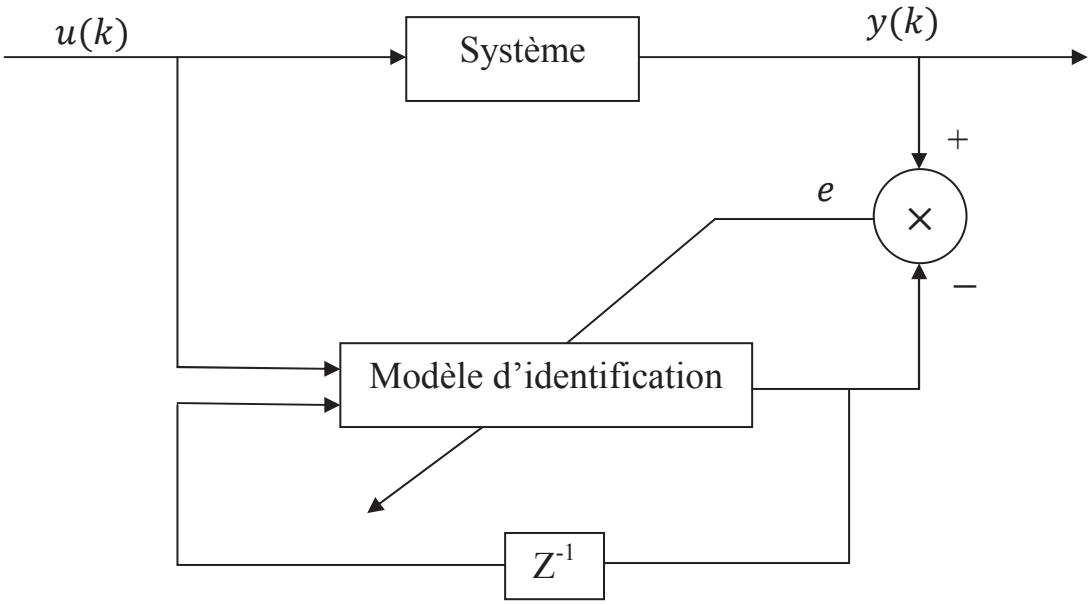


Figure II.1 Structure d'identification parallèle

II.5.2. Identification série-parallèle

La sortie du modèle est calculée à partir de ses entrées et la sortie du système à identifier

$$\hat{y}(k+1) = f[y(k), y(k-1), \dots, y(k-n+1); u(k), u(k-1), \dots, u(k-m+1)] \qquad \text{II.5}$$

Ce modèle a plus de chances de converger, car tous les signaux utilisés lors de l'identification sont bornés [8]

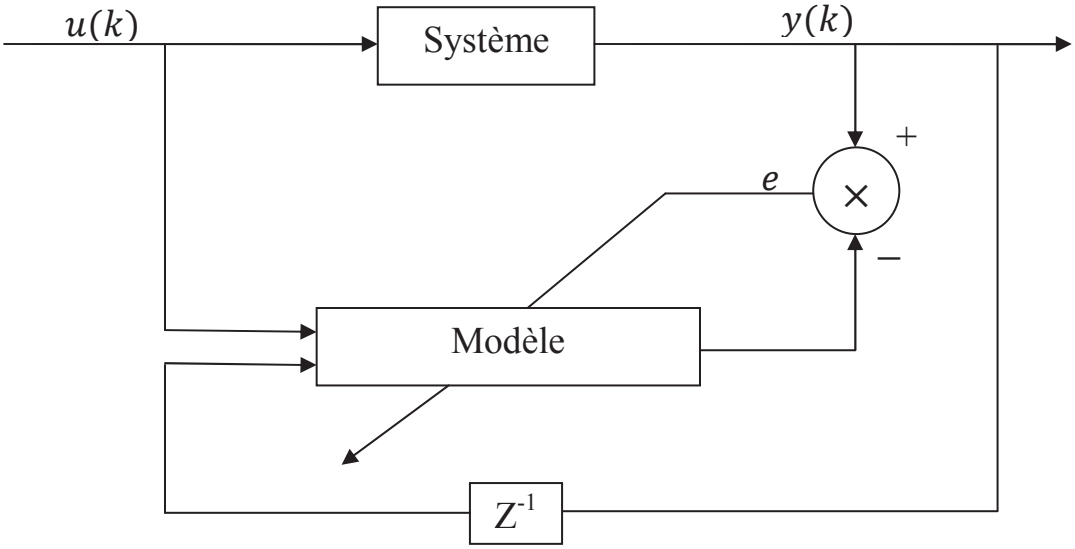


Figure II.2 Structure d'identification série-parallèle

II.6. Modélisation neuronale des systèmes non linéaires

- 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

II.7. Étude et simulation

II.7.1. Cas des systèmes SISO

II.7.1.1. Exemple :

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

$$y(k+1) = f[y(k)] + u(k) \qquad \text{II.6}$$

Ou la fonction inconnue f est donnée par :

$$f[y(k)] = \frac{y(k)}{1+y^2(k)} \qquad \text{II.7}$$

Le modèle d'identification neuronal est décrit par :

$$\hat{y}(k+1) = \hat{f}[y(k)] + u(k) \qquad \text{II.8}$$

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

$$u(k) = \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{50}\right) \qquad \text{II.9}$$

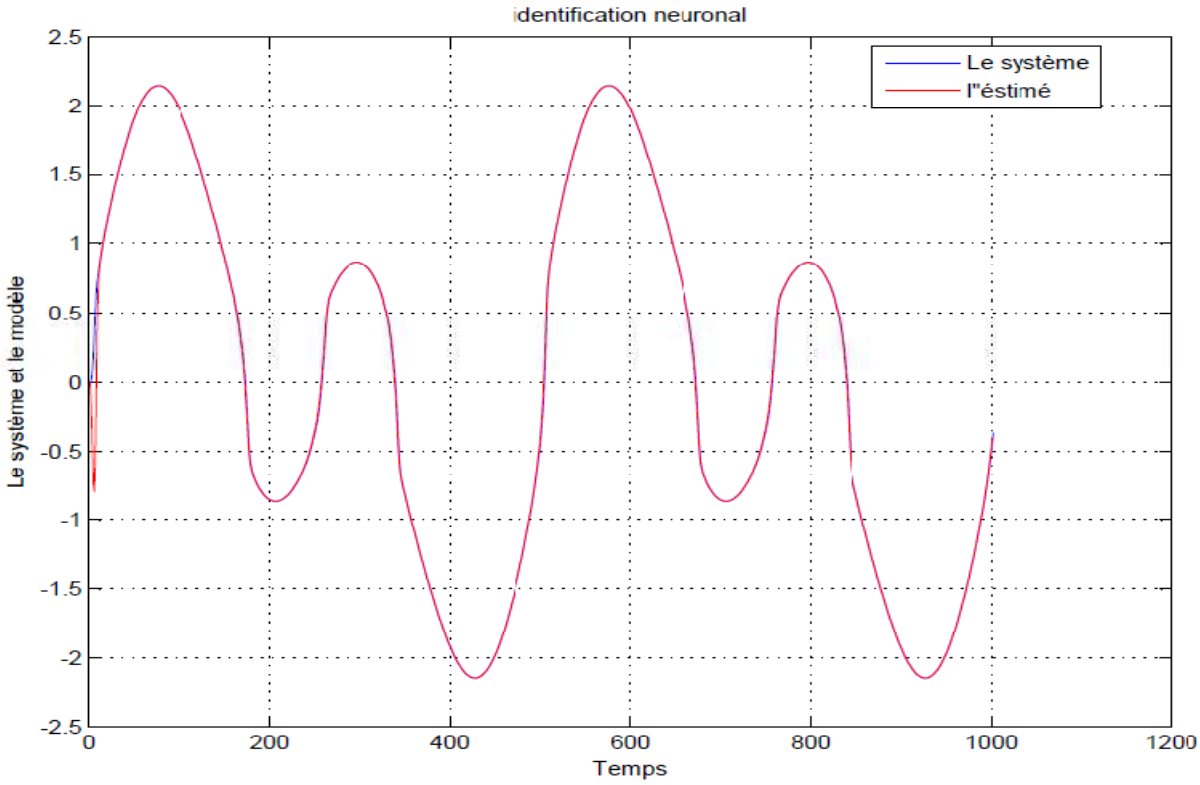


Figure II.3 Sorties, y(-) du système, $\hat{y}$(-) du modèle

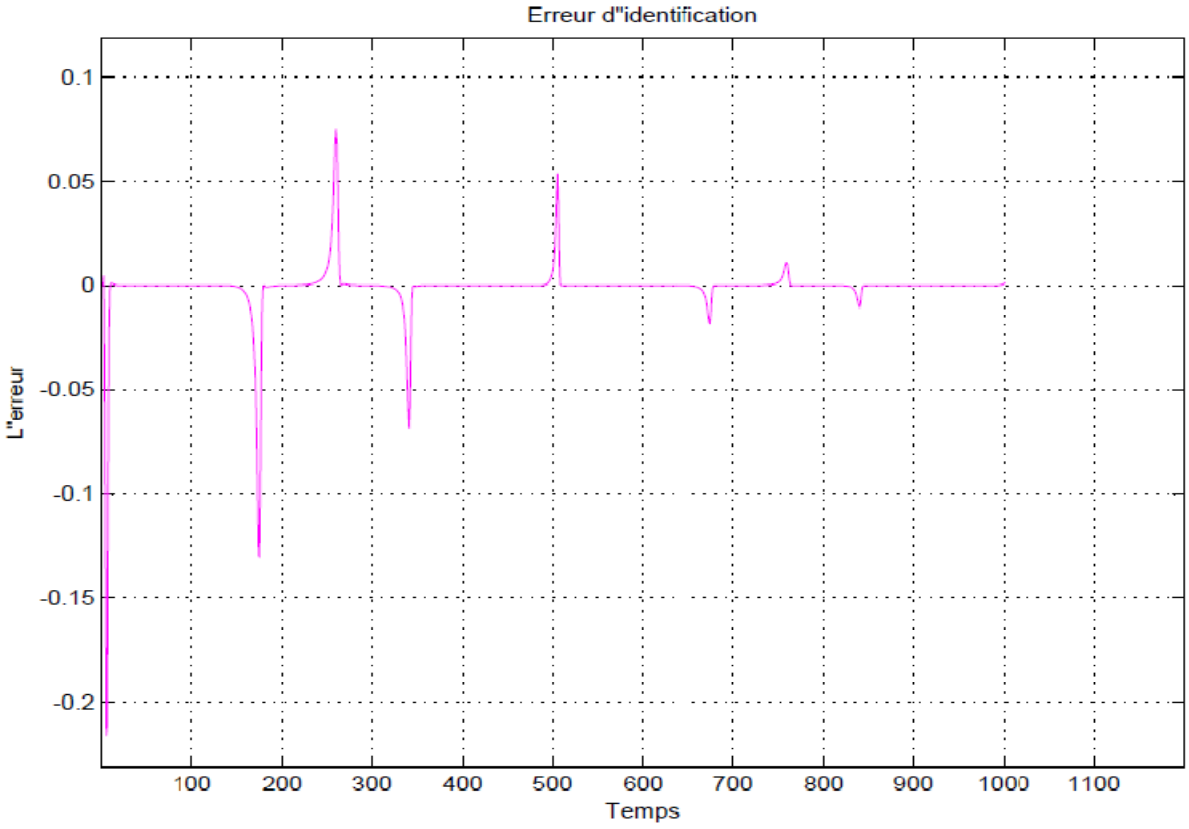


Figure II.4 Erreur d'identification

II.7.1.2. Exemple :

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

$$y(k+1) = f[y(k), y(k-1)] + u(k) \qquad \text{II.10}$$

Où f est donnée par :

$$f[y(k), y(k-1)] = \frac{y(k)y(k-1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)} \qquad \text{II.11}$$

Le modèle d'identification neuronal est décrit par :

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

L'entrée du système et du modèle est donnée par

$$u(k) = \sin\left(\frac{2\pi k}{25}\right) \qquad \text{II.13}$$

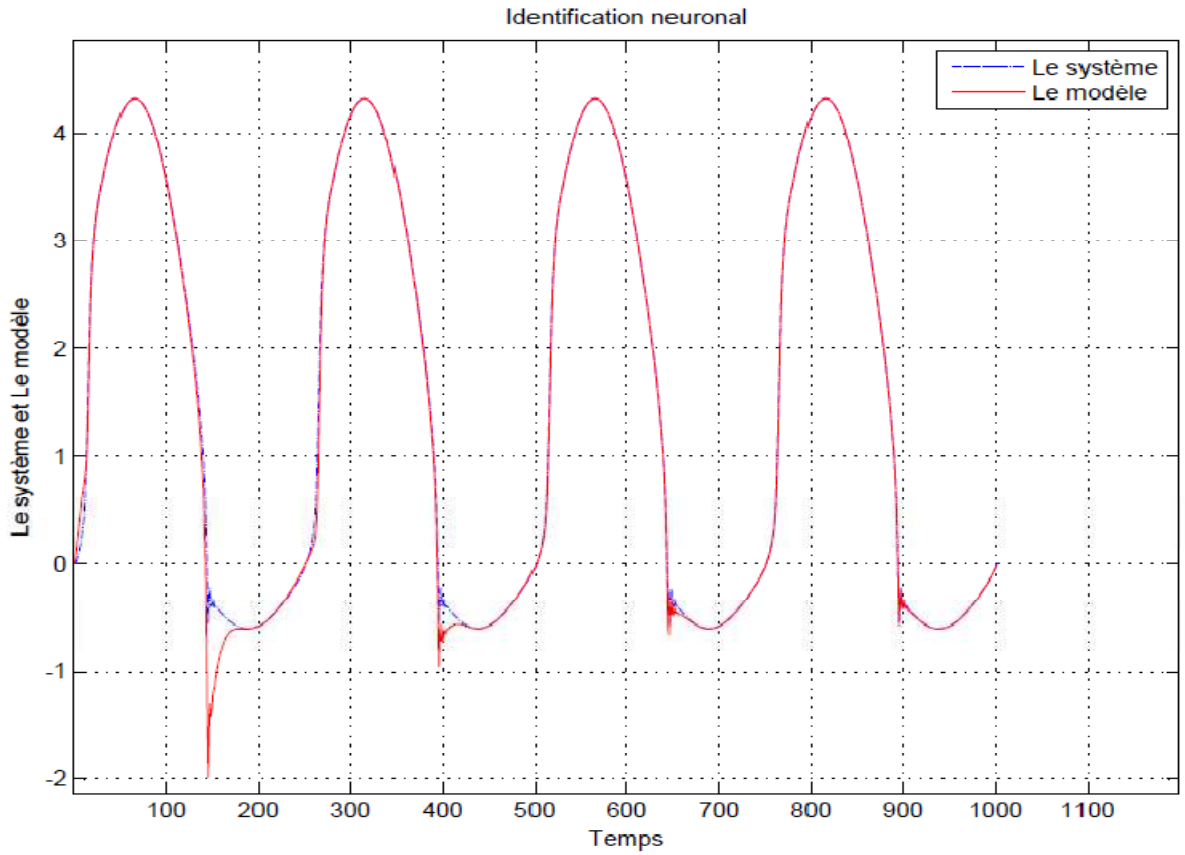


Figure II.5 Sorties, y(-) du système, $\hat{y}$(-) du modèle

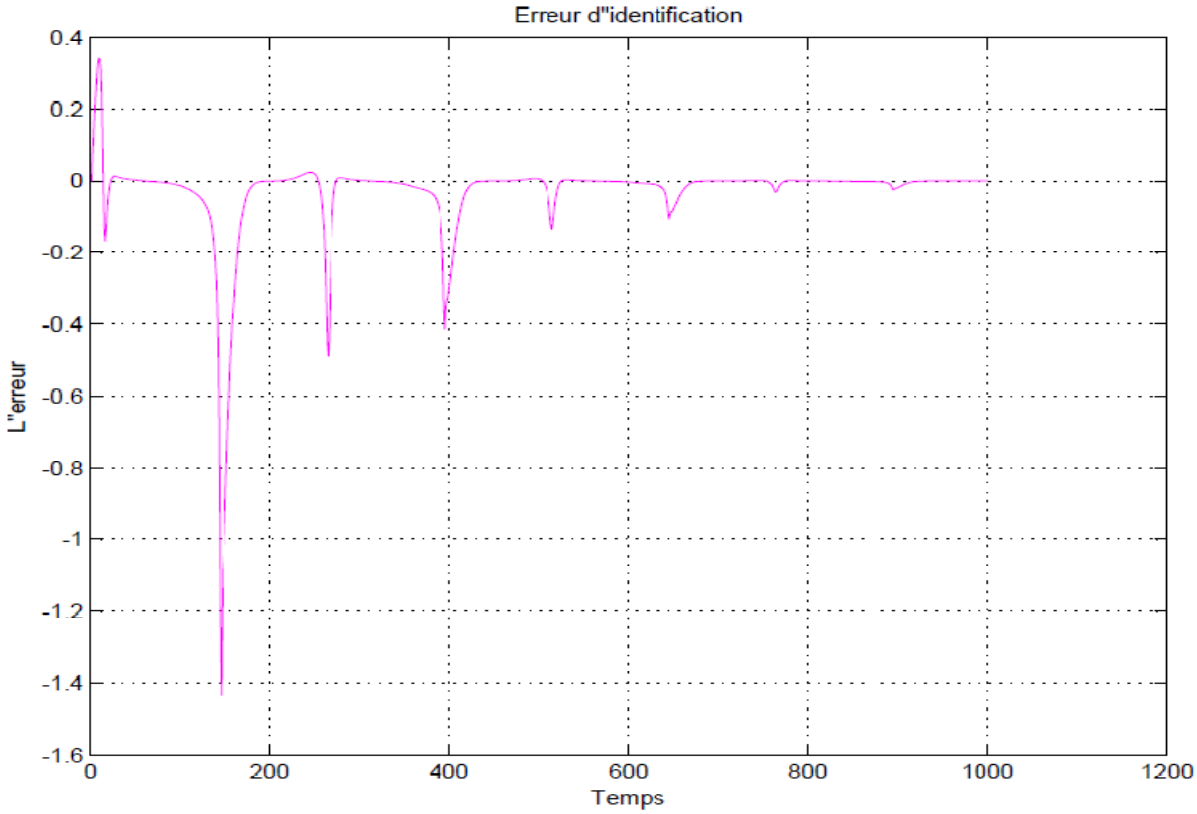


Figure II.6 Erreur d'identification

II.7.2. Cas des systèmes 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 linéaire MIMO le **u** et **y** qui étaient des scalaires dans l'équation II.3 deviennent des vecteurs ainsi un système non linéaire MIMO est donnée par :

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

Avec :

$$y(k) = \begin{bmatrix} y1(k) \\ y2(k) \\ . \\ . \\ . \\ y_m(k) \end{bmatrix} \quad \text{II.15}$$

$$u(k) = \begin{bmatrix} u1(k) \\ u2(k) \\ . \\ . \\ . \\ u_p(k) \end{bmatrix} \quad \text{II.16}$$

II.7.2.1. Exemple :

Soit le système MIMO décrit par l'équation aux différences suivante :

$$\begin{pmatrix} y_1(k+1) \\ y_2(k+1) \end{pmatrix} = \begin{pmatrix} \frac{y_1(k)}{1+y_2^2(k)} \\ \frac{y_1(k)y_2(k)}{1+y_2^2(k)} \end{pmatrix} + \begin{pmatrix} u_1(k) \\ u_2(k) \end{pmatrix} \qquad \text{II.17}$$

Le modèle d'identification neuronal est ainsi décrit par :

$$\begin{pmatrix} \hat{y}_1(k+1) \\ \hat{y}_2(k+1) \end{pmatrix} = \begin{pmatrix} \hat{f}_1[y_1(k), y_2(k)] \\ \hat{f}_2[y_1(k), y_2(k)] \end{pmatrix} + \begin{pmatrix} u_1(k) \\ u_2(k) \end{pmatrix} \qquad \text{II.18}$$

L'entrée du système est donnée par

$$\begin{pmatrix} u_1(k) \\ u_2(k) \end{pmatrix} = \begin{pmatrix} \sin\left(\frac{2\pi k}{25}\right) \\ \cos\left(\frac{2\pi k}{25}\right) \end{pmatrix} \qquad \text{II.19}$$

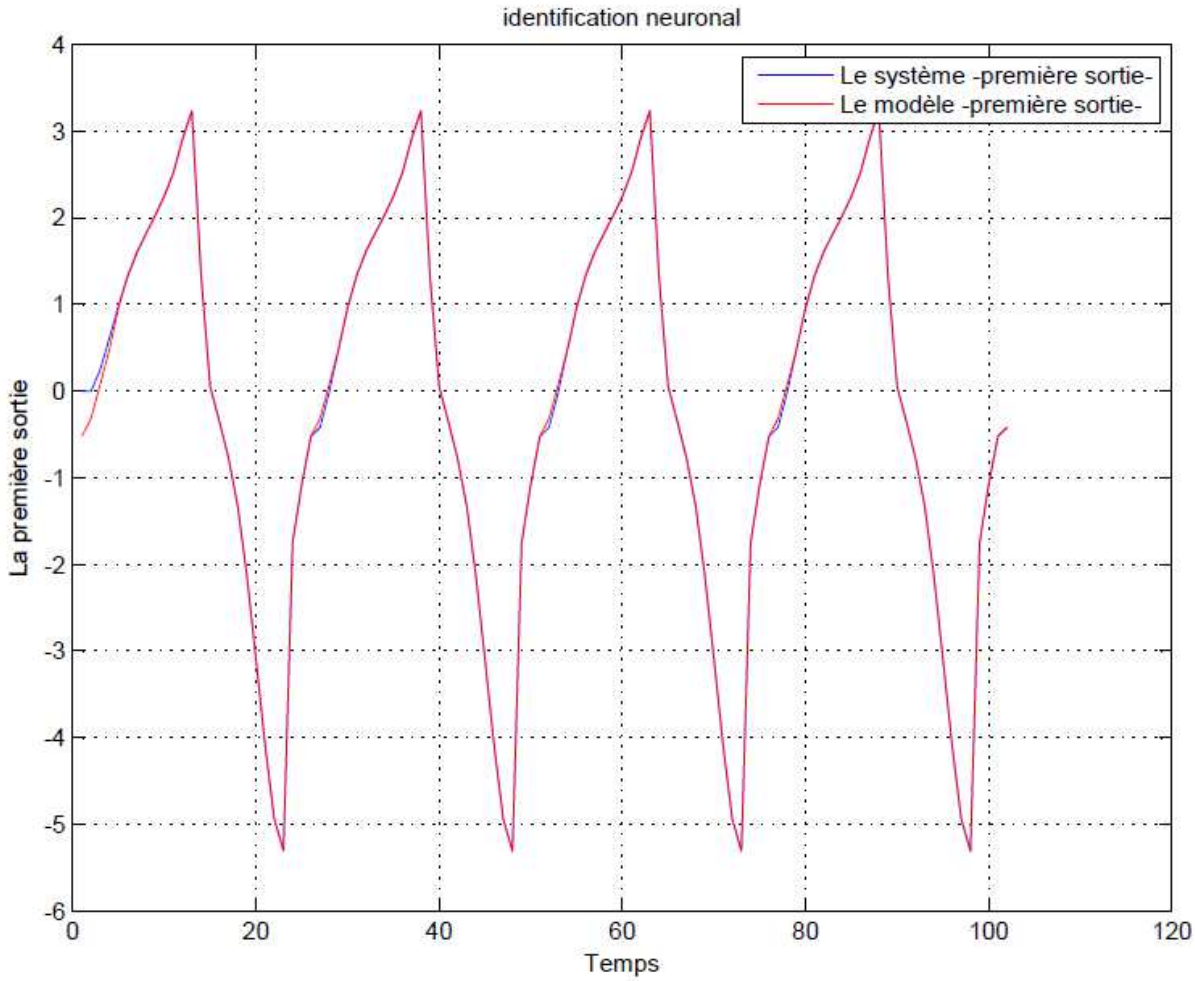


Figure II.6 Sorties, $y1$(-) du système, $\hat{y}1$(-) du modèle

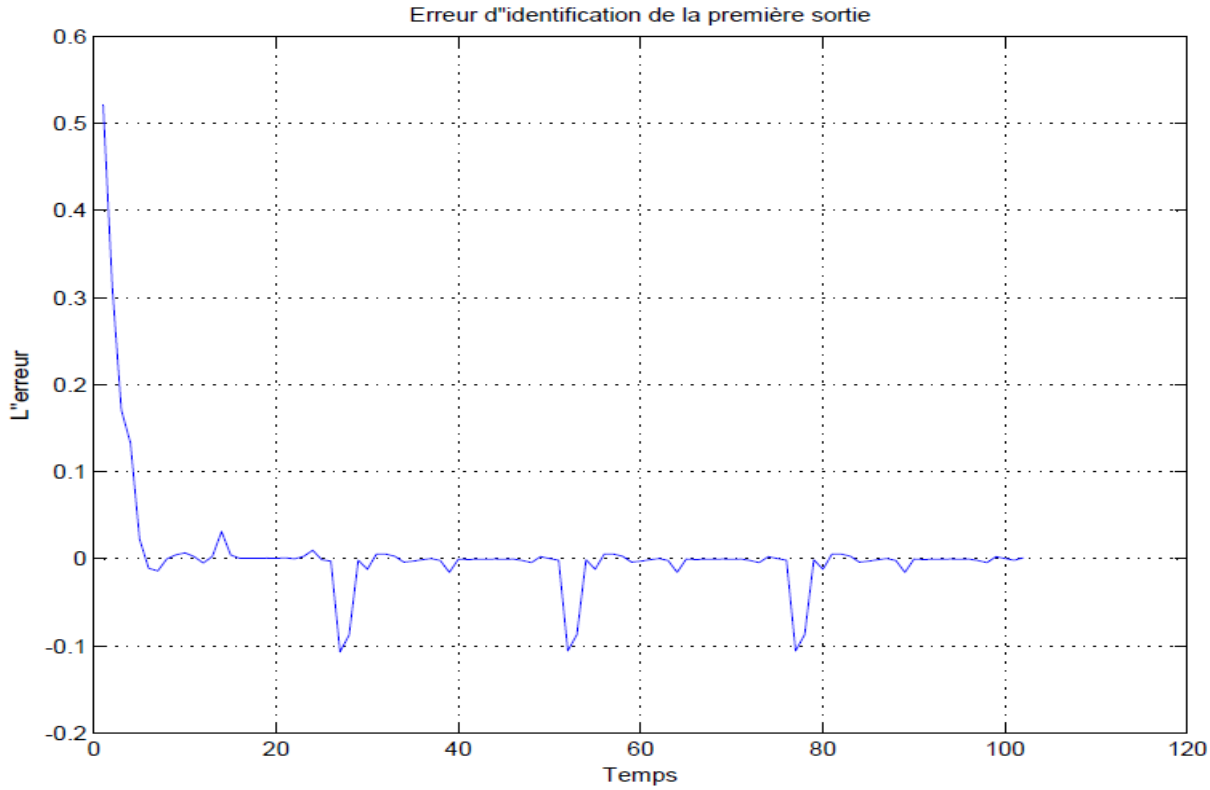


Figure II.7 Erreur d'identification de $y1$

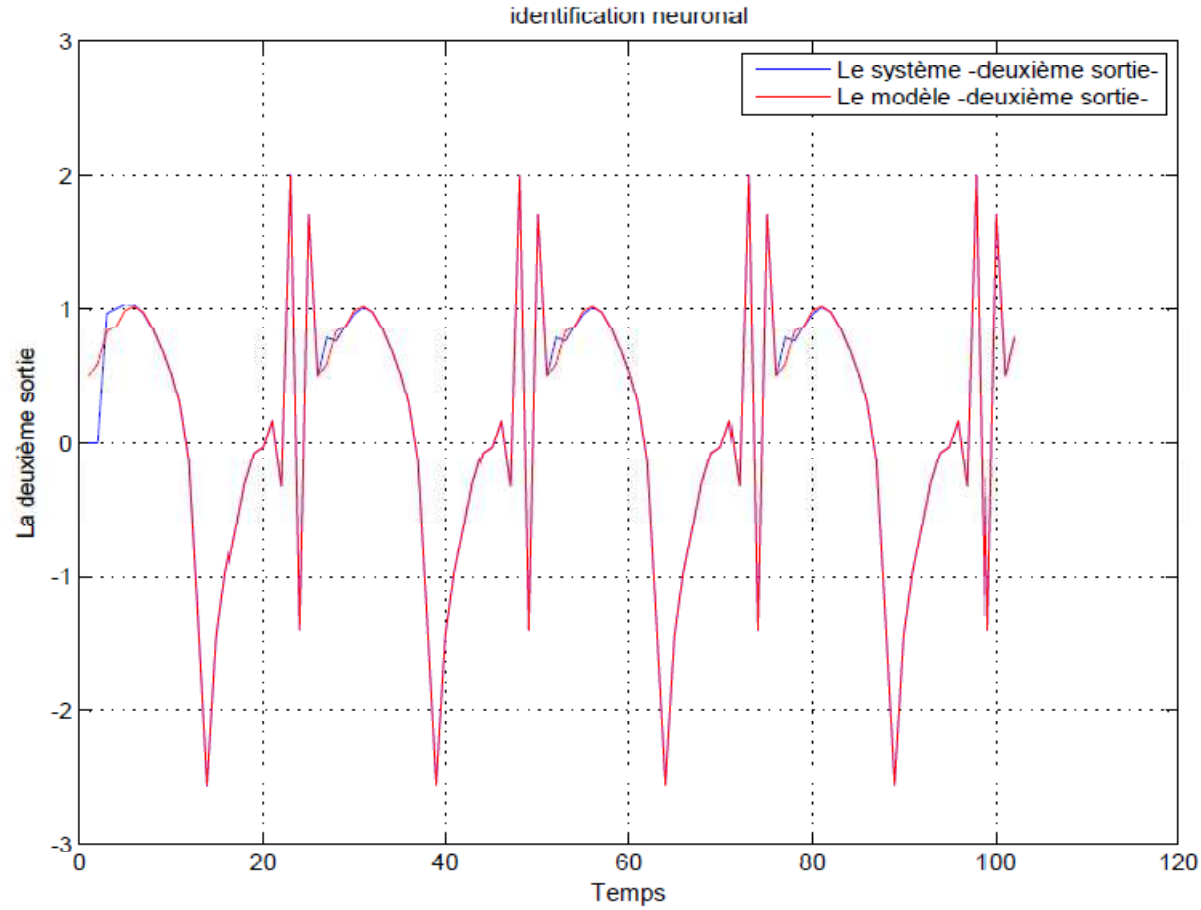


Figure II.8 Sorties, $y2$(-) du système, $\hat{y}2$(-) du modèle

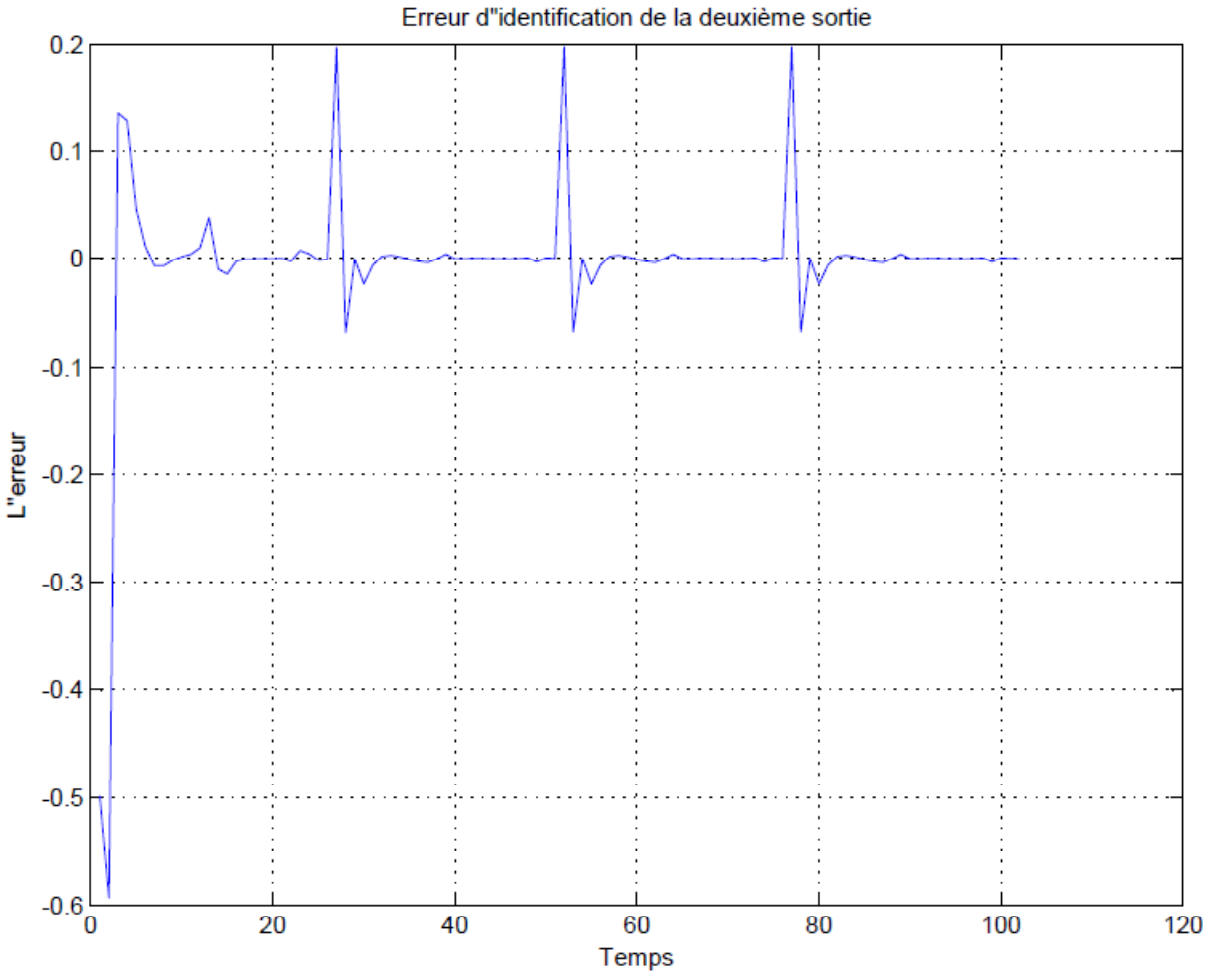


Figure II.9 Erreur d'identification de $y2$

II.8. Conclusion

Dans ce chapitre, nous avons étudié le problème de l'identification des systèmes non linéaires

Nous avons vu qu'il existe plusieurs modèles d'identification pouvant être utilisés pour l'identification de ce type de systèmes.

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 qui sont multiples. Ils permettent de réaliser, de manière simple et efficace, des modèles précis.

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). [9]

III.2. Le modèle neuronal inverse [10]

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 laborieux. Plusieurs méthodes de modélisation des systèmes ont été présentées dans la littérature. Un système dynamique peut être décrit par l'équation (III.1) reliant ces entrées aux sorties :

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

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 :

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

Les réseaux de perceptrons multicouches peuvent être utilisés pour élaborer le modèle neuronal inverse d'un système. Cependant, d'autre types de réseaux peuvent être utilisés, mais **l**es réseaux MLP statiques présentent la solution la plus simple, toutefois la représentation de l'aspect dynamique du système reste une problématiques. 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 rétropropagation du gradient pour l'apprentissage des réseaux multicouche.

Le réseau correspond à III.2 est :

$$\hat{u}(k) = g(x(k), w) \qquad \text{III.3}$$

La fonction $\boldsymbol{g}$ 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 $\hat{u}$ qui va servir à commander le système Figure III.1

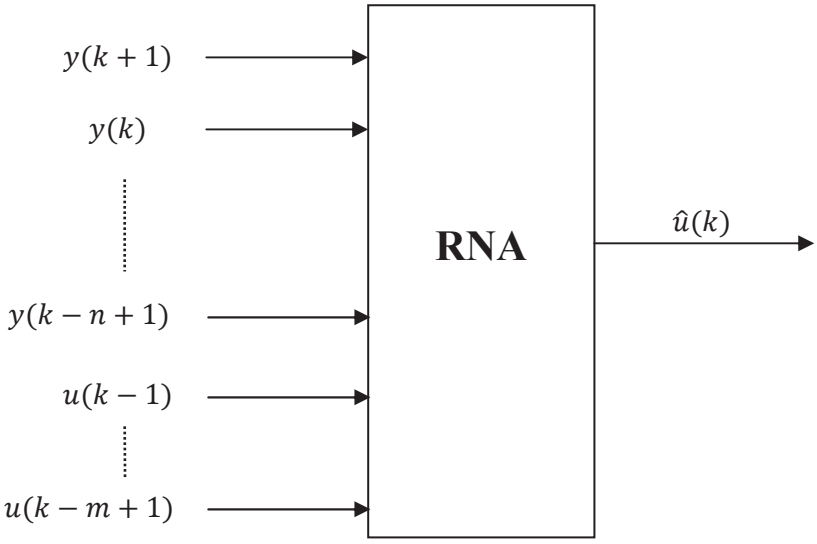


Figure III.1 Réseau de neurones pour identifier le modèle inverse

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 [10]

Dans la l'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 neurones û est$\boldsymbol{e_n}$, l'erreur rétropropagée à travers le réseau et qui va servir pour l'apprentissage hors ligne du réseau. On tente de minimiser le critère E(w):

$$E(w) = \sum_{k-1}^{N} \frac{1}{2}(u(k) - \hat{u}(k,w))^2 \qquad \text{III.4}$$

Où w est un vecteur contenant les poids du réseau inverse

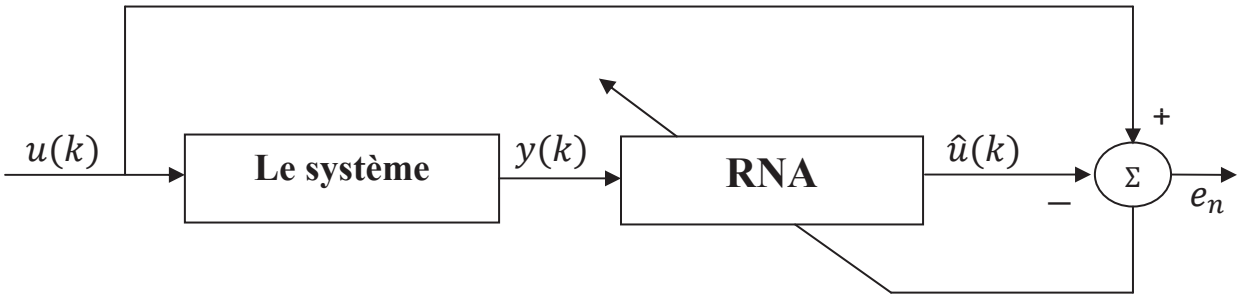


Figure III.2 Architecture générale d'apprentissage

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 multivariable, le modèle ainsi retenu peut ne pas imiter le système réel.

III.2.2. Architecture indirecte d'apprentissage [10]

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 rétropropagation. Cette architecture 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.

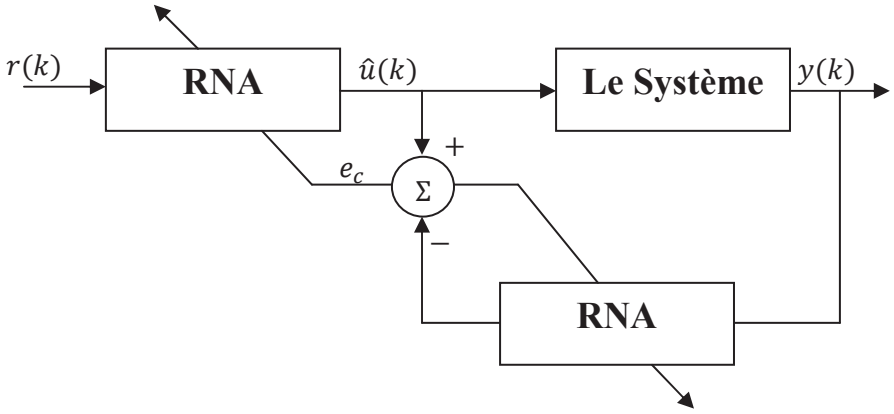


Figure III.3 Architecture indirect d'apprentissage

III.2.3. Architecture spécialisée d'apprentissage [10]

Contrairement aux deux précédentes architectures, l'architecture spécialisée procède par l'utilisation de l'erreur e_c « équation (III.4) », 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.

$$e_c = r(k) - y(k) \qquad \text{III.4}$$

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 rétropropagation imposerait pourtant que l'on rétropropage l'erreur en sortie du modèle, qui est naturellement inconnue. Quoi qu'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.

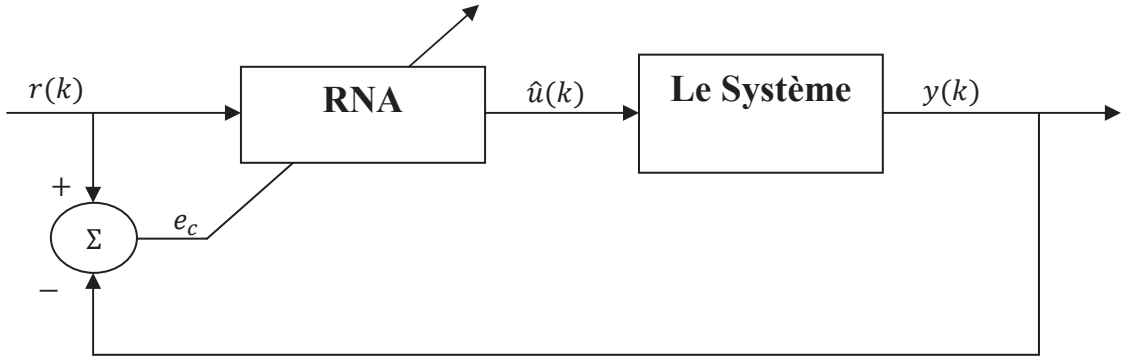


Figure III.4 Architecture spécialisée d'apprentissage

III.3. Commande neuronale directe par modèle inverse [10]

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**.**

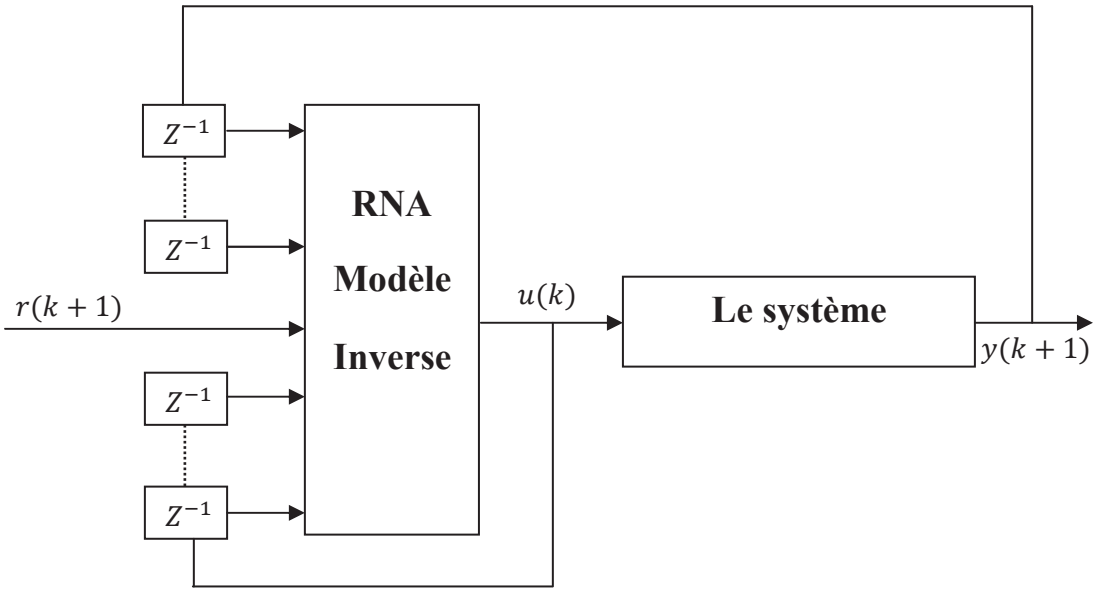


Figure III.5 commande directe par modèle inverse

La valeur de $y(k+1)$Dans (III.2) 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.

III.4. Contrôle adaptatif [8] [11] [12]

L'objectif de la commande adaptative c'est d'appliquer une entrée $u(k)$ à l'entrée du système de façon à ce que l'erreur de commande e_c soit nulle, ce qui veut dire appliquer une entrée pour que le système suive la sortie désirée

Qu'on peut aussi exprimer avec une autre façon, 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. L'objectif est de déterminer une entrée du système $u(k)$ tel que $e_c = y(k) - y_m(k)$ tend vers zéro

Parmi les techniques de commande adaptative les plus vues dans le domaine de la recherche et précisément le domaine de l'automatique est la commande adaptative par un modèle de référence (**M**odel **R**eference **A**daptive **C**ontrol)

Il existe deux approches de contrôle adaptatif: l'approche directe et l'approche indirecte. Dans le contrôle adaptatif direct, montré en figure III.6,e_c 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

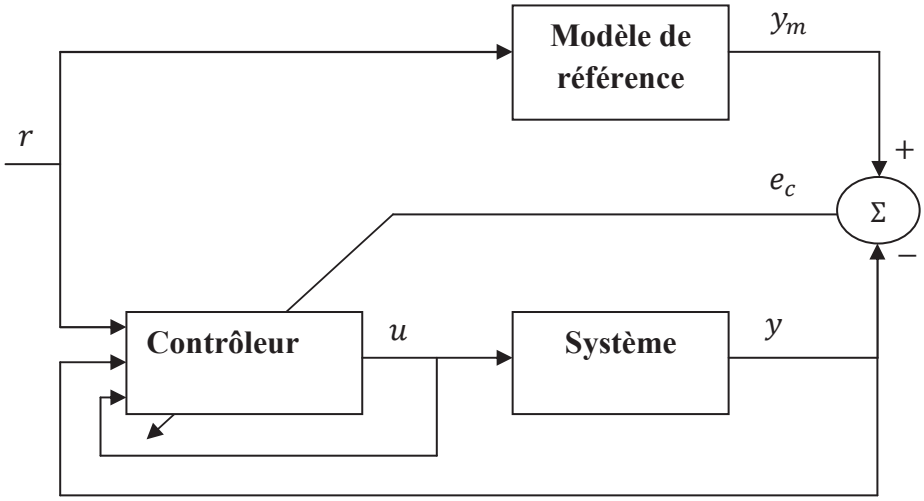


Figure III.6 Contrôle adaptatif direct

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.

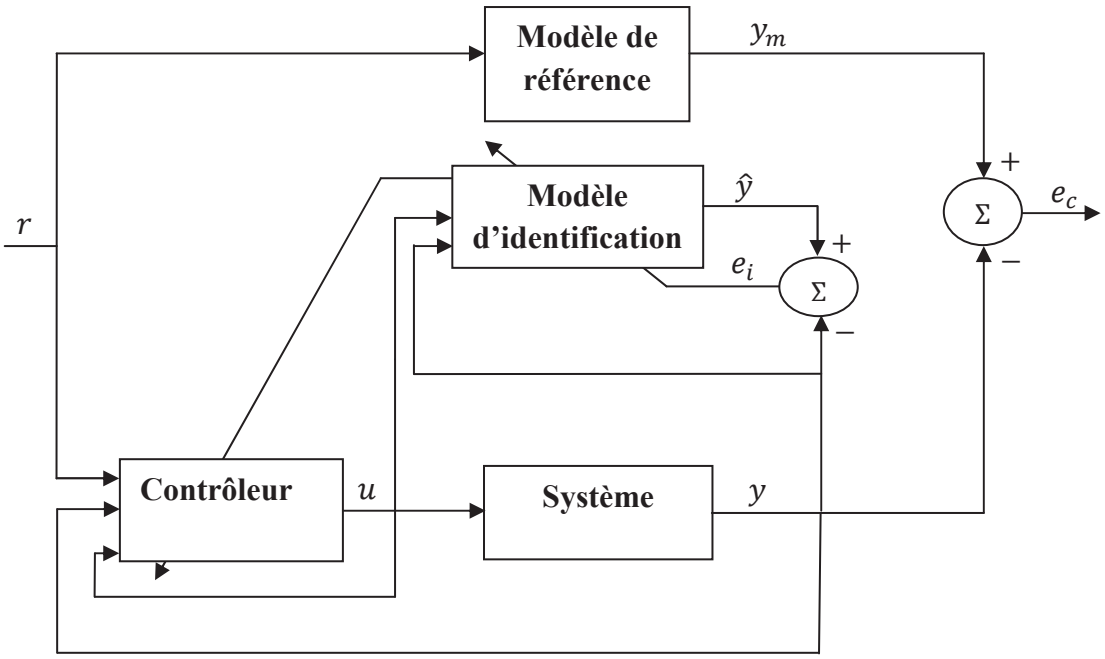


Figure III.7 Contrôle adaptatif indirect

III.5. Contrôle neuronal adaptatif

Lorsqu'un système est construit à partir d'un système neuronal équipé d'un algorithme d'adaptation, il est appelé contrôleur adaptatif neuronal.

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.

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.

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 dans [8]; 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.

III.5.1. 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:

$$y(k+1) = f[y(k), u(k)] \qquad \text{III.5}$$

Pour le modèle de référence

$$y_m(k+1) = f[y_m(k), r(k)] \qquad \text{III.6}$$

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:

$$u(k+1) = RN[y_m(k), y(k), u(k)] \qquad \text{III.7}$$

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

$$e_c(k+1) = y(k+1) - y_m(k+1) \qquad \text{III.8}$$

Cependant, ce type de contrôleur est difficile à utiliser lorsque le système est inconnu, puisque l'effet du changement des paramètres du contrôleur sur e_c, ne peut pas être calculé[8].

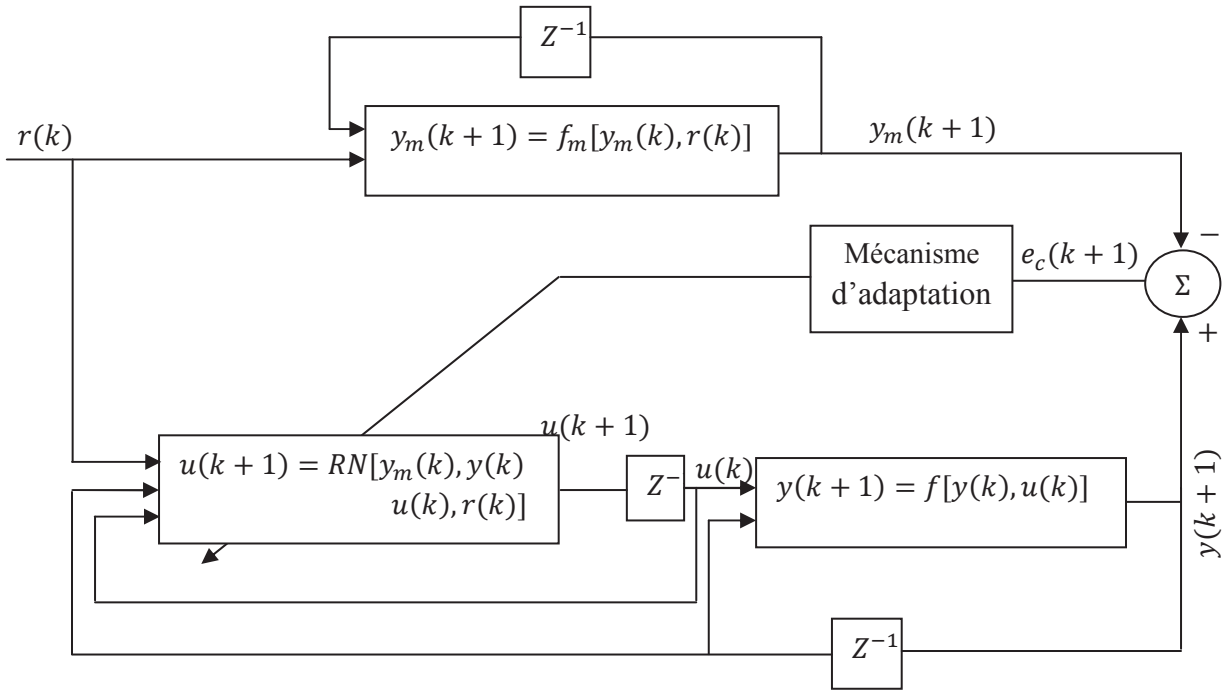


Figure III.8 Contrôle neuronal adaptatif direct

III.5.2. Contrôle neuronal adaptatif 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 neuronal adaptatif 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.

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. [8][13]

Cas d'un système SISO **(II.7.1.2. Exemple) :**

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

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

Le modèle d'identification neuronal est décrit par :

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1), u(k)] \quad \text{III.10}$$

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

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

Pour un modèle de référence donnée par exemple par :

$$y_m(k+1) = f_m[y_m(k), r(k)] \quad \text{III.12}$$

La structure du contrôleur neuronal est donnée par :

$$u(k) = f_c[y_m(k+1), y(k), y(k-1), u(k-1)] \quad \text{III.13}$$

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

$$e_c(k+1) = y(k+1) - y_m(k+1) \quad \text{III.14}$$

Dans cette équation d'erreur, le modèle d'identification neuronal est utilisé à la place du système puisqu'il est inconnu

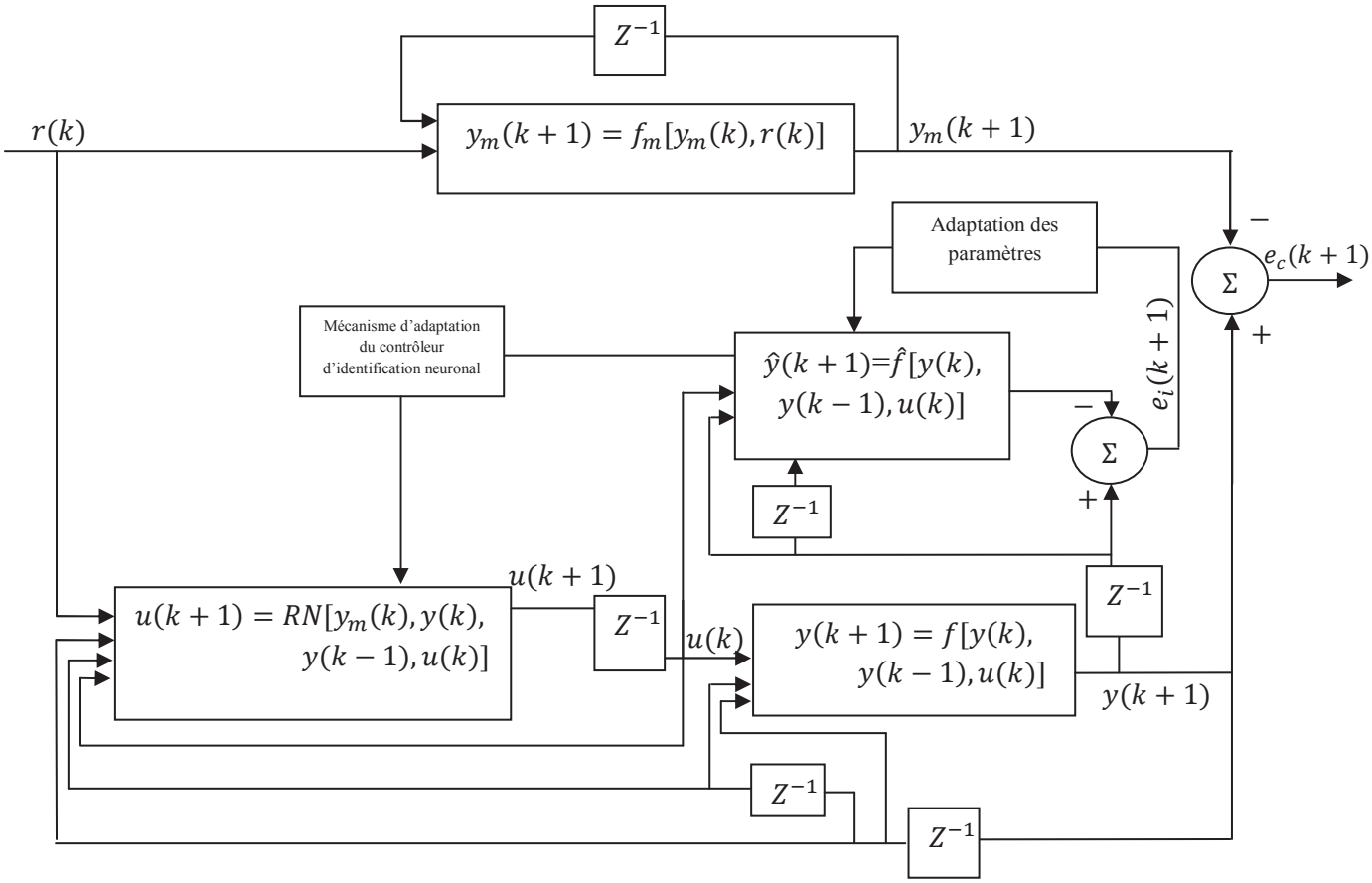


Figure III.9 Contrôle neuronal adaptatif indirect

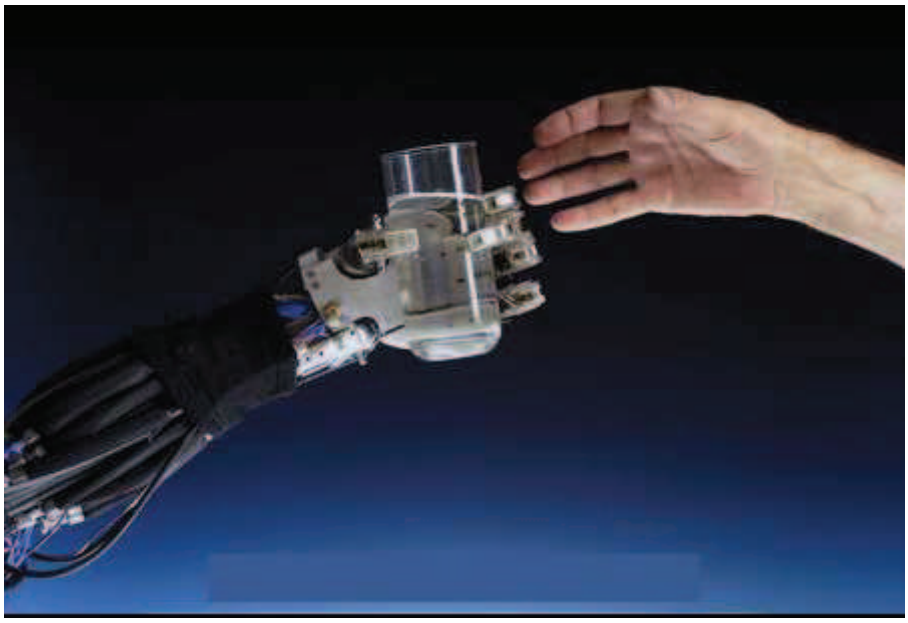
III.6 Conclusion

Dans ce chapitre, on a 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 on a expliqué le principe de la commande adaptatif des systèmes non linéaire on introduisant le contrôle neuronal adaptatif.

Pour le contrôle neuronal adaptatif des systèmes non linéaire, inconnu ou mal définie l'approche directe posent un problème, alors on utilise l'approche indirecte

Chapitre IV :

Contrôle neuronale adaptatif d'un bras manipulateur à deux degrés de liberté



IV.1. Introduction

La commande des robots manipulateurs est l'une des préoccupations majeures des recherches en robotique. En effet, un robot manipulateur est caractérisées 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 [8].

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. Structure mécanique des robots [14]

Un robot manipulateur est constitue de deux parties distinctes, un (ou plusieurs) organe terminal et une structure mécanique articulé [15], comme montre la (figure IV.1)

- **Organe terminal :**

C'est dispositif destiné à manipuler des objets ou à les transformer. Il s'agit donc d'une interface permettant au robot d'interagir avec son environnement.

- **Structure mécanique articulée :**

Son rôle est d'amener l'organe terminal dans une situation (position et orientation) donnée, selon des caractéristiques de vitesse et d'accélération données.

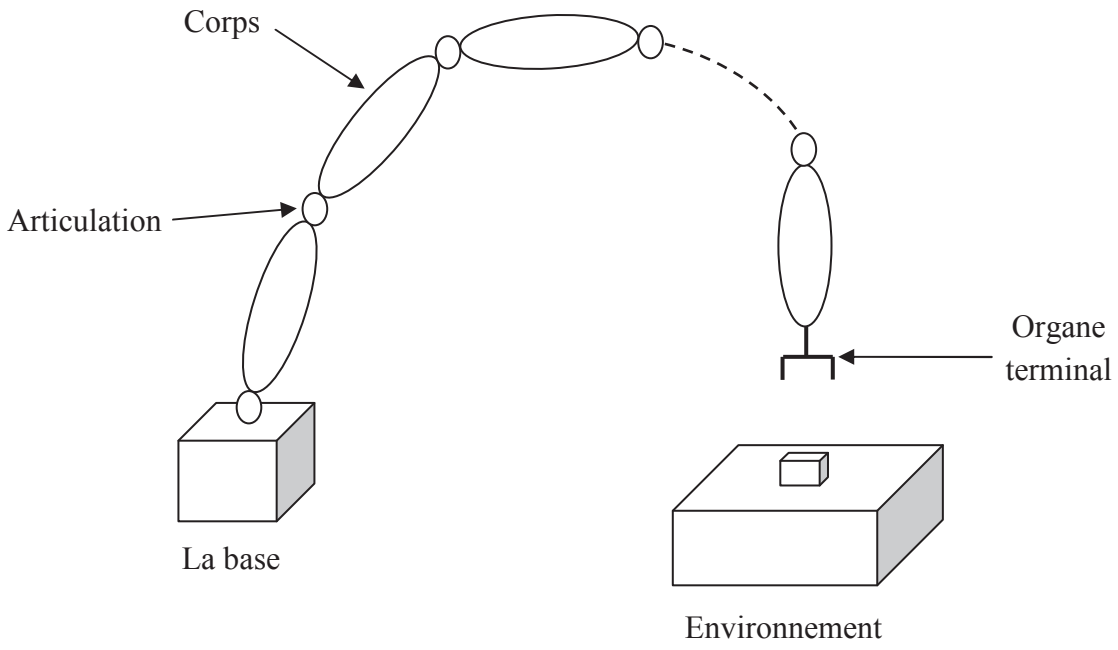


Figure IV.1 Structure ouverte simple [16]

Avant de décrire les relations géométriques entre les différents corps du robot, on définit les notions suivantes: [14] [15] [17]

IV.2.1. Espace articulaire :

Représente l'état des corps composants le robot en fonction des variables articulaires, sa dimension n est égale au nombre de degrés de liberté du robot.

IV.2.2. Espace opérationnelle :

Décris la position et l'orientation de l'organe terminal du robot par rapport à un repère de référence.

IV.2.3. Configuration singulière :

Pour tous les robots, il se peut que dans certaines configurations dites singulières, le nombre de degrés de liberté de l'organe terminal soit inférieur à la dimension de l'espace opérationnel.

Les configurations singulières ou singularités se traduisent, physiquement, par un comportement anormal du mécanisme et, analytiquement, par la nullité du déterminant de la matrice jacobienne *J*.

IV.2.4. Degré de liberté :

Nombre de paramètres utilisés pour spécifier la configuration d'un élément de la chaîne cinématique par rapport à un autre.

IV.3. Modélisation :

Pour la conception, et le développement d'une commande d'un robot nécessitent le calcul de certains modèles mathématiques, tel que :

- Le modèle géométrique direct et inverse qui exprime la situation de l'organe terminal en fonction des variables articulaires du mécanisme et inversement
- le modèle cinématique direct et inverse qui exprime la vitesse de l'organe terminal en fonction des vitesses articulaires et inversement
- le modèle dynamique définissant les équations du mouvement du robot, qui permet d'établir la relation entre les couples ou les forces exercées par les actionneurs et les positions, vitesses, et accélérations des articulations.

IV.4. Modèle géométrique [15]

Dans la modélisation géométrique, on s'intéresse au mouvement sans tenir compte des forces qui le provoquent. Elle s'intéresse à l'étude de la géométrie du robot en vue de décrire ses paramètres géométriques : position et orientation

IV.4.1 Modèle géométrique direct [15]

Le modèle géométrique direct (MGD) est l'ensemble des relations qui permettent d'exprimer la situation de l'organe terminal, c'est-à-dire les coordonnées opérationnelles du robot, en fonction de ses coordonnées articulaires. [15]

La technique la plus répandue pour décrire la géométrie d'un bras manipulateur consiste à utiliser "les paramètres de Denavit-Hartenberg modifiés ".

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

α_i : Angle entre l'axe z_{i-1} et z_i correspond à une rotation autour de x_{i-1}

a_i : Distance entre z_{i-1}et z_i le long de x_{i-1}

θ_i : Angle entre x_{i-1}et x_i correspond à une rotation autour de z_i

d_i : Distance entre x_{i-1}et x_i le long de z_i

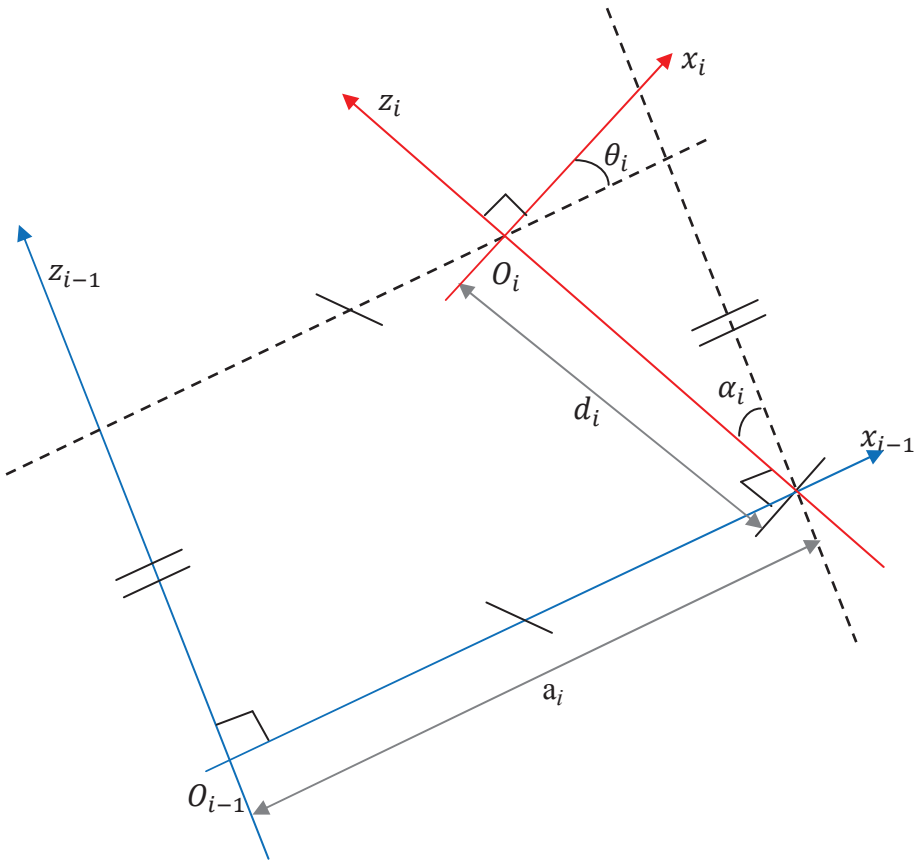


Figure IV.2 Paramètres géométriques

La matrice de transformation homogène est donnée comme suite :

$$T_i^{i-1} = Rot(x_{i-1}, \alpha_{i-1}) Tran(x_{i-1}, a_{i-1}) Rot(z_i, \theta_i) Tran(z_i, d_i) \qquad \text{IV.1}$$

$$T_i^{i-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & -c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \alpha_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ c\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

$$T_i^{i-1} = \begin{bmatrix} A_i^{i-1} & P_i^{i-1} \\ 0_3 & 1 \end{bmatrix} \qquad \text{IV.2}$$

Avec

$$A_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 \\ c\alpha_{i-1}s\theta_i & c\alpha_{i-1}c\theta_i & -s\alpha_{i-1} \\ s\alpha_{i-1}s\theta_i & s\alpha_{i-1}c\theta_i & c\alpha_{i-1} \end{bmatrix} \quad \text{Et} \quad P_i^{i-1} = \begin{bmatrix} a_{i-1} \\ -d_i.s\alpha_{i-1} \\ d_i.c\alpha_{i-1} \end{bmatrix}$$

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

$$T_n^0 = T_1^0 T_2^1 \dots \dots T_n^{n-1} \qquad \text{IV.3}$$

Le modèle géométrique peut être représenté par la relation :

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

Avec

X: Le vecteur des coordonnées opérationnelles ;
q: Le vecteur des coordonnées articulaires ;

IV.4.2 Modèle géométrique inverse [15]

Le modèle géométrique direct d'un robot permet de calculer les coordonnées opérationnelles donnant la situation de l'organe terminal en fonction des coordonnées articulaire. Le problème inverse consiste à calculer les coordonnées articulaires correspondant à une situation donnée de l'organe terminal.

Lorsqu'elle existe, la forme explicite qui donne toutes les solutions possibles (il y a rarement unicité de solution) constitue ce que l'on appelle le modèle géométrique inverse (MGI).

Le MGI s'écrit

$$q = f^{-1}(X) \qquad \text{IV.5}$$

Plusieurs méthodes existent pour résoudre l'équation IV.5 dont on cite la méthode de Paul, qui traite chaque cas particulier et convient pour la plupart des robots industriels

Le principe de la méthode consiste à prémultiplié successivement les deux membres de l'équation IV.6 par les matrices de T_{i-1}^i pour i variant de 1 à $n-1$

$$U_0 = T_n^0 \qquad \text{IV.6}$$

Avec

U_0 : La situation désirée

T_n^0 : La matrice de transformation homogène telle que

$$T_n^0 = T_1^0 T_2^1 \dots T_n^{n-1} \quad \text{IV.7}$$

Donc

$$U_0 = T_1^0 T_2^1 T_3^2 \dots T_n^{n-1}$$

$$T_1^0 U_0 = T_2^1 T_3^2 \dots T_n^{n-1}$$

$$T_2^1 U_1 = T_3^2 \dots T_n^{n-1}$$

$$\vdots$$

$$T_{n-1}^{n-2} U_{n-2} = T_n^{n-1} \quad \text{IV.8}$$

Lors de la résolution du problème géométrique inverse, on rencontre pratiquement les situations suivantes [15]

- Absence de solution lorsque la situation désirée est en dehors de la zone accessible du robot
- Infinité de solutions, lorsque le robot se trouve dans certaines configurations singulières.
- solution en nombre fini, lorsqu'elles peuvent être calculées sans ambiguïté.

IV.5. Modèle cinématique : [15] [17]

IV.5.1. Modèle cinématique direct

Le modèle cinématique direct décrit les vitesses de coordonnées opérationnelles (la vitesse de l'organe terminal) en fonction des vitesses articulaire

Le MCD est décrit par l'équation IV.9

$$\dot{X} = J(q)\dot{q} \quad \text{IV.9}$$

Avec

$J(q)$ Est la matrice jacobienne de dimension $m \times n$

L'une des méthodes de calcul de la matrice jacobienne est la dérivation du modèle géométrique direct :

$$J_{ij} = \frac{\partial f_i(q)}{\partial q_j} \quad \text{IV.10}$$

$pour \quad i = 1, \dots, m;\ j = 1, \dots, n$

Où J_{ij} est l'élément (i, j) de la matrice jacobienne J

IV.5.2. Modèle cinématique inverse

L'objectif du modèle cinématique inverse (MCI) est de calculer les vitesses articulaires $\dot{q}$ qui assurent au repère terminal une vitesse opérationnelle $\dot{X}$ imposée.

Pour obtenir le MCI, on inverse le modèle cinématique direct en résolvant un système d'équations linéaires

Dans le cas régulier la matrice jacobienne est carrée et son déterminant non nul, alors le MCI est : équation IV.11

$$\dot{q} = J^{-1}(q)\dot{X} \qquad \text{IV.11}$$

IV.6. Modèle dynamique [15][16][17]

Le modèle dynamique est la relation entre les couples (et/ou forces) appliqués aux actionneurs et les positions, vitesses, et accélérations articulaires.

IV.6.1 Modèle dynamique inverse

Le modèle dynamique inverse exprime les couples exercés par les actionneurs en fonction des positions, vitesses et accélérations des articulations. Il est alors représenté par la relation IV.12

$$\tau_i = f(q, \dot{q}, \ddot{q}, f_e) \qquad \text{IV.12}$$

Avec

τ : Vecteur des couples/forces des actionneurs, selon l'articulation

q: Vecteur des positions articulaires

$\dot{q}$: Vecteurs des vitesses articulaires

$\ddot{q}$: Vecteurs des accélérations articulaires

f_e : Vecteurs représentant l'effort extérieur (forces et moments) qu'exerce le robot sur l'environnement.

Plusieurs formalismes ont été utilisés pour obtenir le modèle dynamique des robots. Les formalismes les plus souvent utilisés sont :

- Le formalisme de Lagrange
- Le formalisme de Newton-Euler

Dans ce travail, on s'intéresse au formalisme de Lagrange

Formalisme de Lagrange

Le formalisme d'Euler Lagrange décrit les équations du mouvement en termes de travail et d'énergie du système, ce qui se traduit, lorsque l'effort extérieur sur l'organe terminal est supposé nul, par l'équation IV .13

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i \qquad \text{IV.13}$$

Avec

L : Le Lagrangien du système est égale à $E - U$

E : L'énergie cinétique totale du système

U : L'énergie potentielle totale du système

q_i : $i^{éme}$ Cordonnée généralisé du système

τ_i : Le couple appliqué au $i^{éme}$ élément du système

Expression de l'énergie cinétique :

$$E = \sum_{i=1}^{n} \frac{1}{2} m_i v_i^T v_i + \frac{1}{2} \omega_i^T I_i \omega_i \qquad \text{IV. 14}$$

Avec

I_i : Matrice d'inertie du corps *i*

m_i : Masse du corps *i*

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

ω_i : Vitesse angulaire du corps *i*

La matrice d'inertie I_i tous 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*.

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \qquad \text{IV.15}$$

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

Soit J_{vi} et J_{wi} deux jacobiens tels que :

$$v_i = J_{vi}(q)\dot{q} \quad \text{Et} \quad \omega_i = R_i(q)J_{\omega i}(q)\dot{q}$$

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

$$E = \frac{1}{2}\dot{q}^T \sum_{i=1}^{n} [m_i J_{vi}(q)^T J_{vi}(q) + J_{\omega i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega i}(q)]\dot{q} \qquad IV.16$$

En d'autre terme, l'énergie cinétique du manipulateur s'écrit :

$$E = \frac{1}{2}\dot{q}^T M(q)\dot{q} \qquad \text{IV.17}$$

La matrice *M(q)* est symétrique définie positive de dimension $n \times n$, elle dépend de la configuration q du manipulateur. Cette matrice est appelée matrice d'inertie du robot.

Expression de l'énergie potentielle

L'énergie potentielle d'un corps *i* à pour seule source la pesanteur, elle peut être calculé en supposant que la masse de l'objet entier est concentrée à son centre de gravité, est donnée par l'équation IV.18

$$U_i = g^T O_{gi} m_i \qquad \text{IV.18}$$

Avec

g : Vecteur de gravité exprimé dans le repère de base ;

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

m_i : La masse du corps *i*.

L'énergie potentielle totale du robot de n corps est donc

$$U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} g^T O_{gi} m_i \qquad \text{IV.19}$$

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

On peut remarquer que l'énergie cinétique E (équation IV.17) peut se mettre sous la forme :

$$E = \frac{1}{2}\sum_{i,j}^{n} d_{ij}(q)\dot{q}_i\dot{q}_j \qquad \text{IV. 20}$$

Le Lagrangien s'écrit donc :

$$L = E - U = \frac{1}{2}\sum_{i,j}^{n} d_{ij}(q)\dot{q}_i\dot{q}_j - U(q) \qquad \text{IV. 21}$$

On a

$$\frac{\partial L}{\partial \dot{q}_k} = \sum_{j} d_{kj}\dot{q}_j \qquad \text{IV. 22}$$

Et

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} = \sum_{i} d_{kj}\ddot{q}_j + \sum_{j}\frac{d}{dt}d_{kj}\dot{q}_j \qquad \text{IV. 23}$$

$$= \sum_{j} d_{kj}\ddot{q}_j + \sum_{i,j}\frac{\partial d_{kj}}{\partial q_i}\dot{q}_i\dot{q}_j$$

Et aussi

$$\frac{\partial L}{\partial q_k} = \frac{1}{2}\sum_{i,j}\frac{\partial d_{ij}}{\partial q_k}\dot{q}_i\dot{q}_j - \frac{\partial U}{\partial q_k} \qquad \text{IV. 24}$$

Ainsi, les équations d'Euler - Lagrange deviennent:

$$\sum_{j} d_{kj}\ddot{q}_j + \sum_{i,j}\left[\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right]\dot{q}_i\dot{q}_j - \frac{\partial U}{\partial q_k} = \tau_k \quad k = 1, \dots, n \qquad \text{IV. 25}$$

En utilisant le fait que :

$$\sum_{i,j}\left[\frac{\partial d_{kj}}{\partial q_i}\right]\dot{q}_i\dot{q}_j = \frac{1}{2}\sum_{i,j}\left[\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{kj}}{\partial q_j}\right]\dot{q}_i\dot{q}_j \qquad \text{IV. 26}$$

On obtient

$$\sum_{i,j}\left[\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right]\dot{q}_i\dot{q}_j = \sum_{i,j}\frac{1}{2}\left[\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right]\dot{q}_i\dot{q}_j \qquad \text{IV. 27}$$

On note

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

Et

$$\phi_k = \frac{\partial U}{\partial q_k}$$

Finalement

$$\sum_i d_{kj}(q)\ddot{q}_j + \sum_{i,j} c_{ijk}(q)\,\dot{q}_i\dot{q}_j + \phi_k(q) = \tau_k; \quad k = 1, \dots, n \qquad \text{IV.28}$$

Ce qui peut s'écrire sous forme matricielle :

$$M(q)\ddot{q} + N(q,\dot{q})\dot{q} + G(q) = \tau \qquad \text{IV.29}$$

Où x_{kj}, le $(k,j)^{éme}$ 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. q_i^2 Sont appelé centrifuge, et ceux impliquant un produit $\dot{q}_i\dot{q}_j$ avec $i \neq j$ sont les termes de Coriolis

IV.6.2 Modèle dynamique direct [16]

Le modèle dynamique direct est celui qui exprime les accélérations articulaires en fonction des positions, vitesses et couples des articulations. Il est alors représenté par la relation IV.30

$$\ddot{q} = f(q,\dot{q},\tau,f_e) \qquad \text{IV.30}$$

Pour obtenir l'équation du modèle dynamique direct il suffit d'inverser l'équation du modèle inverse (équation IV.29) comme montre l'équation IV.31

$$\ddot{q} = M^{-1}(q)(\tau - N(q,\dot{q})\dot{q} - G(q)) \qquad \text{IV.31}$$

IV.7 Commande neuronale adaptative d'un bras manipulateur à 2 ddl

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 hors-ligne avec des paramètres fixes ne peut pas faire face au changement des paramètres. Par conséquent.

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, ou par une trajectoire continue.

IV.8. **Etude et simulation d'un bras à 2 ddl**

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 en figure IV.3 [17]

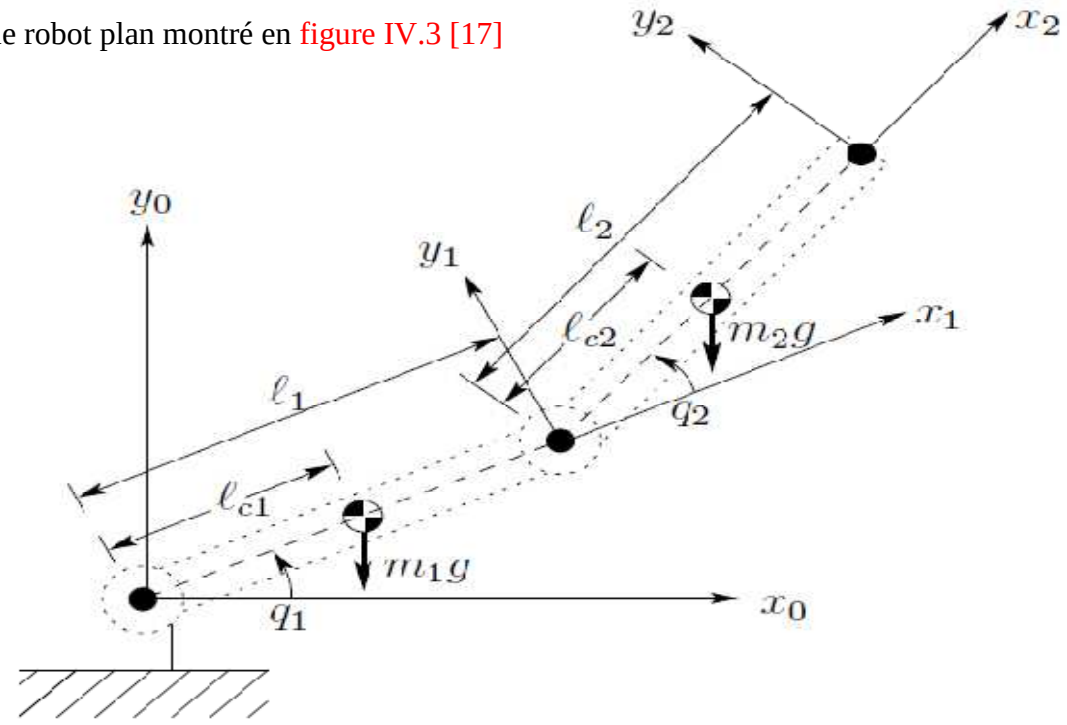


Figure IV.3 Structure d'un robot manipulateur à deux degrés de liberté

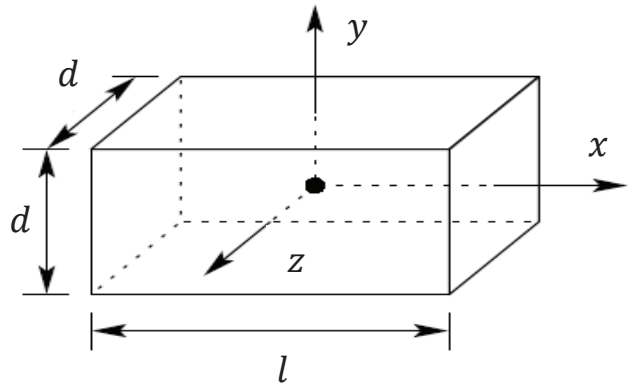


Figure IV.4 Le corps parallélépipède

Ce bras manipulateur est constitué de deux corps parallélépipédique de longueur l_1 et l_2 dont la section est un carré de côté $d \times d$, la masse volumique du matériau utilisé est ρ, et de masses m_1 et m_2 respectives des corps 1 et 2.

Calcul de la matrice d'inertie d'un corps parallélépipédique figure IV.4

$$I = \begin{bmatrix} \int (y^2+z^2)dm & -\int xydm & -\int xzdm \\ -\int xydm & \int (x^2+z^2)dm & -\int yzdm \\ -\int xzdm & -\int yzdm & \int (x^2+y^2)dm \end{bmatrix} \quad \text{IV.32}$$

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

$$= \rho\frac{d^2 l}{12}(d^2+d^2) = \frac{md^2}{6} \quad \text{IV.33}$$

- $\int (x^2+z^2)dm = \frac{m(d^2+l^2)}{12}$ IV.34
- $\int (x^2+y^2)dm = \frac{m(d^2+l^2)}{12}$ IV.35
- $-\int xydm = -\int xzdm = -\int yzdm = 0$ IV.36

Donc

$$I = \begin{bmatrix} \frac{md^2}{6} & 0 & 0 \\ 0 & \frac{m(d^2+l^2)}{12} & 0 \\ 0 & 0 & \frac{m(d^2+l^2)}{12} \end{bmatrix} \quad \text{IV.37}$$

On déduit I_1 et I_2, les matrices d'inertie des corps 1 et 2

$$I_1 = \begin{bmatrix} \frac{md^2}{6} & 0 & 0 \\ 0 & \frac{m(d^2+l_1^2)}{12} & 0 \\ 0 & 0 & \frac{m(d^2+l_1^2)}{12} \end{bmatrix} \quad \text{IV.38}$$

$$I_2 = \begin{bmatrix} \frac{md^2}{6} & 0 & 0 \\ 0 & \frac{m(d^2+l_2^2)}{12} & 0 \\ 0 & 0 & \frac{m(d^2+l_2^2)}{12} \end{bmatrix} \quad \text{IV.39}$$

Energie cinétique :

$$E = \frac{1}{2}\dot{q}^T \sum_{i=1}^{n} [m_i J_{vi}(q)^T J_{vi}(q) + J_{\omega i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega i}(q)]\dot{q} \quad IV.40$$

Partie translation

$$v_1 = J_{v_1}\dot{q} \quad \text{IV.41}$$

$$J_{v_1} = \begin{bmatrix} -\frac{l_1}{2}\sin(q_1) & 0 \\ \frac{l_1}{2}\cos(q_1) & 0 \\ 0 & 0 \end{bmatrix} \quad \text{IV.42}$$

$$v_2 = J_{v_2}\dot{q} \quad \text{IV.43}$$

$$J_{v_2} = \begin{bmatrix} -l_1\sin(q_1) - \frac{l_2}{2}\sin(q_1+q_2) & -\frac{l_2}{2}\sin(q_1+q_2) \\ l_1\cos(q_1) + \frac{l_2}{2}\cos(q_1+q_2) & \frac{l_2}{2}\cos(q_1+q_2) \\ 0 & 0 \end{bmatrix} \quad \text{IV.44}$$

Partie rotation

$$\omega_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}\dot{q} = R_1^T J_{\omega 1}\dot{q} \quad \text{IV.45}$$

$$J_{\omega 1}(q)^T R_1(q) I_1 R_1(q)^T J_{\omega 1}(q) = \begin{bmatrix} \frac{m_1(d^2+l_1^2)}{12} & 0 \\ 0 & 0 \end{bmatrix} \quad \text{IV.46}$$

$$\omega_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \dot{q} = R_2^T J_{\omega 2} \dot{q} \quad \text{IV.47}$$

$$J_{\omega 2}(q)^T R_2(q) I_2 R_2(q)^T J_{\omega 2}(q) = \begin{bmatrix} \frac{m_2(d^2+l_1^2)}{12} & \frac{m_2(d^2+l_1^2)}{12} \\ \frac{m_2(d^2+l_1^2)}{12} & \frac{m_2(d^2+l_1^2)}{12} \end{bmatrix} \quad \text{IV.48}$$

Si on note

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

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

$$\frac{1}{2} \dot{q}^T \begin{bmatrix} i_1 + i_2 & i_2 \\ i_2 & i_2 \end{bmatrix} \dot{q} \quad \text{IV.50}$$

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

$$M(q) = m_1 {J_{v1}}^T J_{v1} + m_2 {J_{v2}}^T J_{v2} + \begin{bmatrix} i_1 + i_2 & i_2 \\ i_2 & i_2 \end{bmatrix} \quad \text{IV.51}$$

$$M(q) = \begin{bmatrix} \frac{m_1 l_1^2}{4} + m_2\left(l_1^2 + \frac{l_2^2}{4} + l_1 l_2 \cos(q_2)\right) + i_1 + i_2 & m_2\left(\frac{l_2^2}{4} + \frac{l_1 l_2}{2}\cos(q_2)\right) + i_2 \\ m_2\left(\frac{l_2^2}{4} + \frac{l_1 l_2}{2}\cos(q_2)\right) + i_2 & \frac{m_2 l_2^2}{4} + i_2 \end{bmatrix}$$

$$\begin{cases} c_{111} = \frac{1}{2}\frac{\partial M_{11}}{\partial q_1} = 0 \\ c_{121} = c_{211} = \frac{1}{2}\frac{\partial M_{11}}{\partial q_2} = -\frac{1}{2}m_2 l_1 l_2 \sin(q_2) = h \\ c_{221} = \frac{\partial M_{12}}{\partial q_2} - \frac{1}{2}\frac{\partial M_{22}}{\partial q_1} = -\frac{1}{2}m_2 l_1 l_2 \sin(q_2) = h \\ c_{112} = \frac{\partial M_{21}}{\partial q_1} - \frac{1}{2}\frac{\partial M_{11}}{\partial q_2} = \frac{1}{2}m_2 l_1 l_2 \sin(q_2) = -h \\ c_{122} = c_{212} = \frac{1}{2}\frac{\partial M_{22}}{\partial q_1} = 0 \\ c_{222} = \frac{1}{2}\frac{\partial M_{22}}{\partial q_2} = 0 \end{cases} \quad \text{IV. 52}$$

Energie potentielle

$$\begin{cases} U_2 = m_2 g(l_1 \sin(q_1) + \frac{l_2}{2}\sin(q_1 + q_2)\,) \\ U_2 = m_2 g(l_1 \sin(q_1) + \frac{l_2}{2}\sin(q_1 + q_2)) \\ U = U_1 + U_2 \end{cases} \quad \text{IV.53}$$

D'où

$$\begin{cases} \phi_1 = \frac{\partial U}{\partial q_1} = \left(\frac{m_1 l_1}{2} + m_2 l_1\right) g cos(q_1) + \frac{m_2 l_2}{2} g cos(q_1 + q_2) \\ \phi_2 = \frac{\partial U}{\partial q_2} = \frac{m_2 l_2}{2} g cos(q_1 + q_2) \end{cases} \quad \text{IV.54}$$

Finalement, on obtient les équations dynamiques suivantes :

$$\begin{cases} M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_{121}\dot{q}_1\dot{q}_2 + c_{211}\dot{q}_2\dot{q}_1 + c_{221}\dot{q}_2^2 + \phi_1 = \tau_1 \\ M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + c_{112}\dot{q}_1^2 + \phi_2 = \tau_1 \end{cases} \quad \text{IV.55}$$

Dans ce cas la matrice $N(q, \dot{q})$ (de coriolis et centrifuge) est donnée par :

$$N = \begin{bmatrix} h\dot{q}_2 & h\dot{q}_2 + h\dot{q}_1 \\ h\dot{q}_1 & 0 \end{bmatrix} \quad \text{IV.56}$$

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

$$\tau = M(q)\ddot{q} + N(q, \dot{q})\dot{q} + G(q) \quad \text{IV.57}$$

Avec

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

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

Matrice d'inertie

$$M(q) = \begin{bmatrix} \frac{m_1 l_1^2}{4} + m_2\left(l_1^2 + \frac{l_2^2}{4} + l_1 l_2 \cos(q_2)\right) + i_1 + i_2 & m_2\left(\frac{l_2^2}{4} + \frac{l_1 l_2}{2}\cos(q_2)\right) + i_2 \\ m_2\left(\frac{l_2^2}{4} + \frac{l_1 l_2}{2}\cos(q_2)\right) + i_2 & \frac{m_2 l_2^2}{4} + i_2 \end{bmatrix}$$

Matrice de coriolis et centrifuge

$$N(q,\dot{q}) = \begin{bmatrix} -\frac{1}{2} m_2 l_1 l_2 \sin(q_2)\dot{q}_2 & -\frac{1}{2} m_2 l_1 l_2 \sin(q_2)(\dot{q}_2 + \dot{q}_1) \\ +\frac{1}{2} m_2 l_1 l_2 \sin(q_2)\dot{q}_1 & 0 \end{bmatrix}$$

Vecteur des forces et de gravité

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

Avec les paramètres du modèle sont

m_1 =15.91 Kg, $m_2 = 11.36$ Kg

$l_1 = 0.432$ m, $l_2 = 0.432$ m

Les positions désirées sont donnée par :

$$\begin{cases} q_{d1} = 2\cos\left(\frac{4\pi k}{3}\right) + \sin\left(\frac{2\pi k}{3}\right); & 0 \le k \le 10 \\ q_{d2} = 1 - 2\cos\left(\frac{4\pi k}{3}\right) - \sin\left(\frac{2\pi k}{3}\right); & 0 \le k \le 10 \end{cases} \quad \text{IV.58}$$

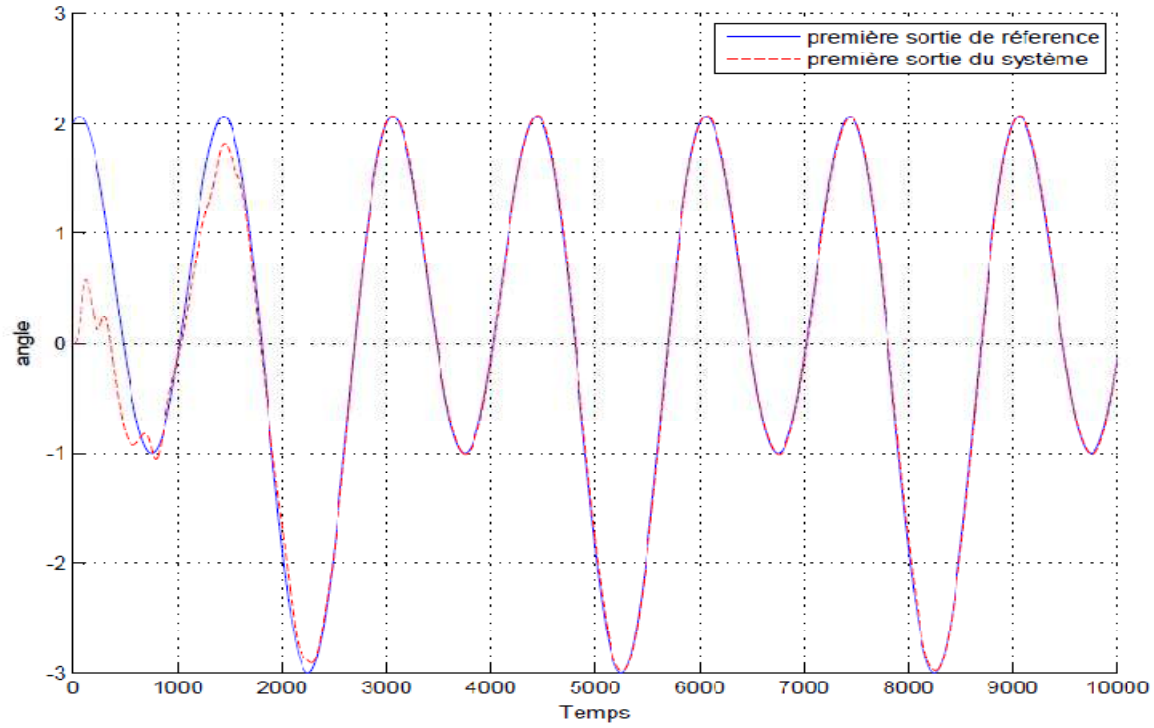


Figure IV.5 : Sorties, $q1$(--) du système, $q1d$(-) désirée

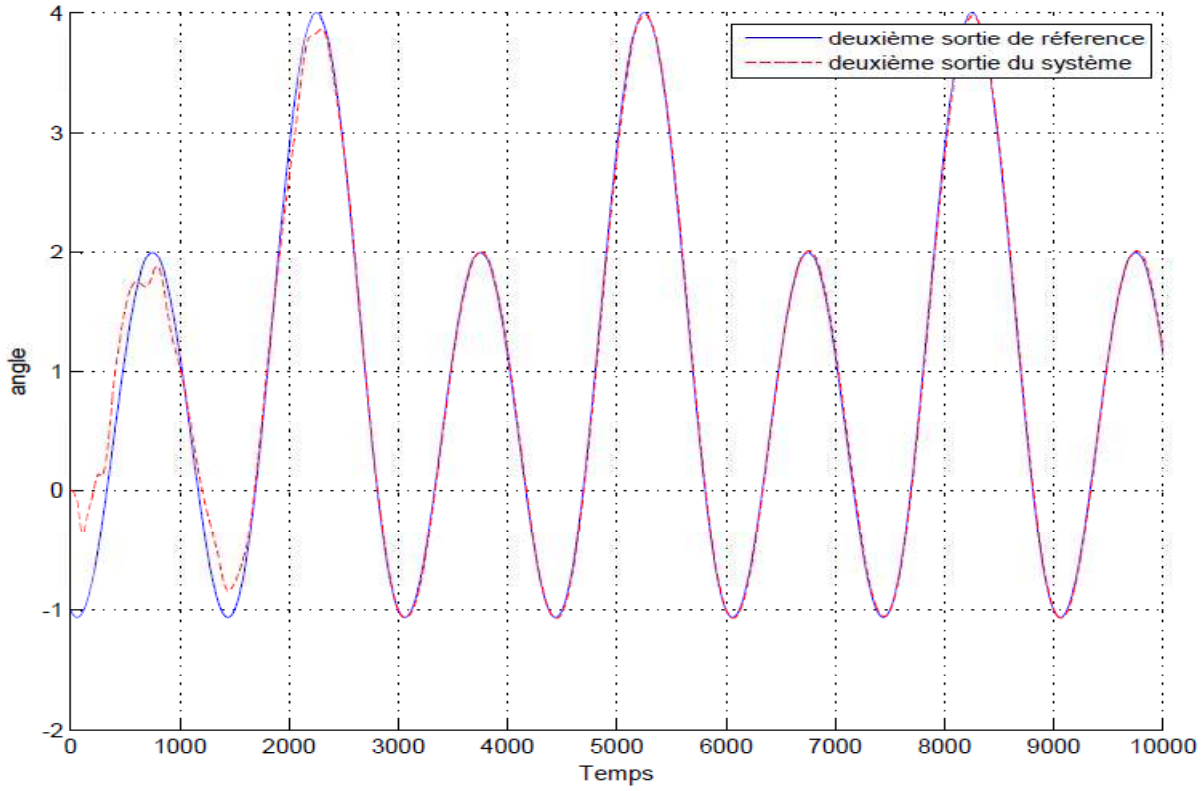


Figure IV.6: Sorties, $q2$(--) du système, $q2d$(-) désirée

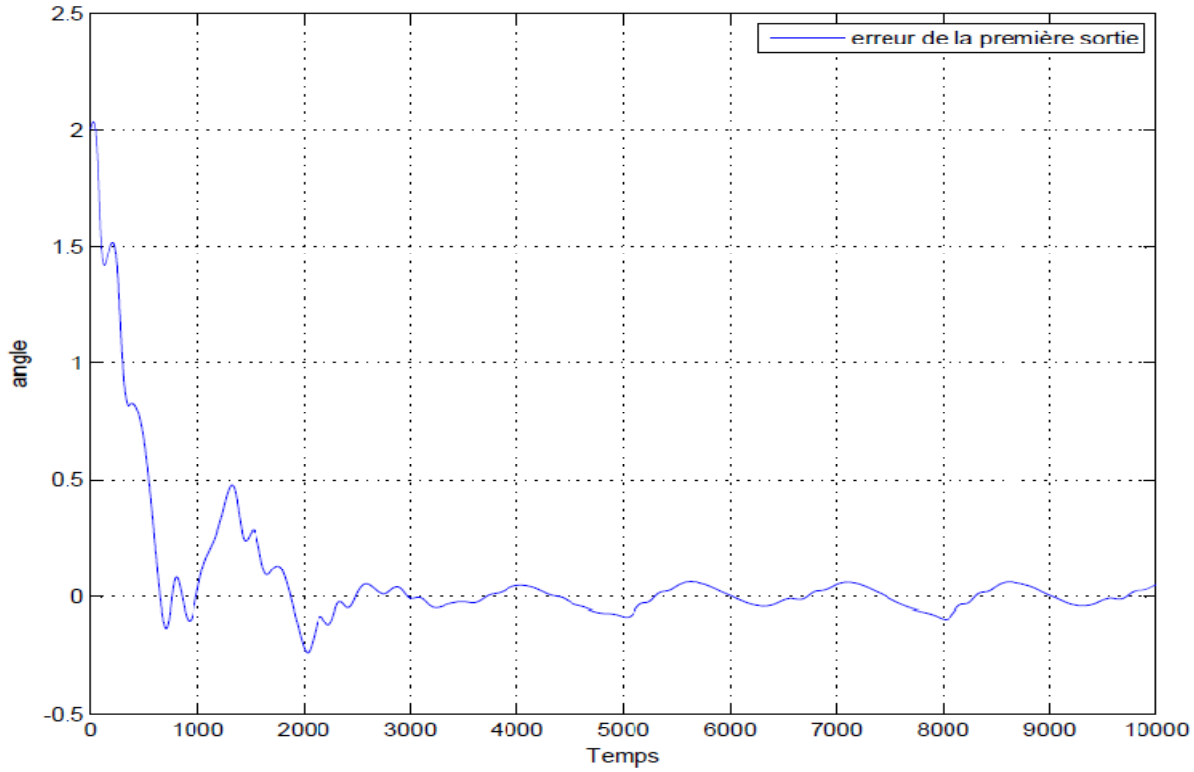


Figure IV.7 : l'erreur entre la première sortie du système et la sortie désirée

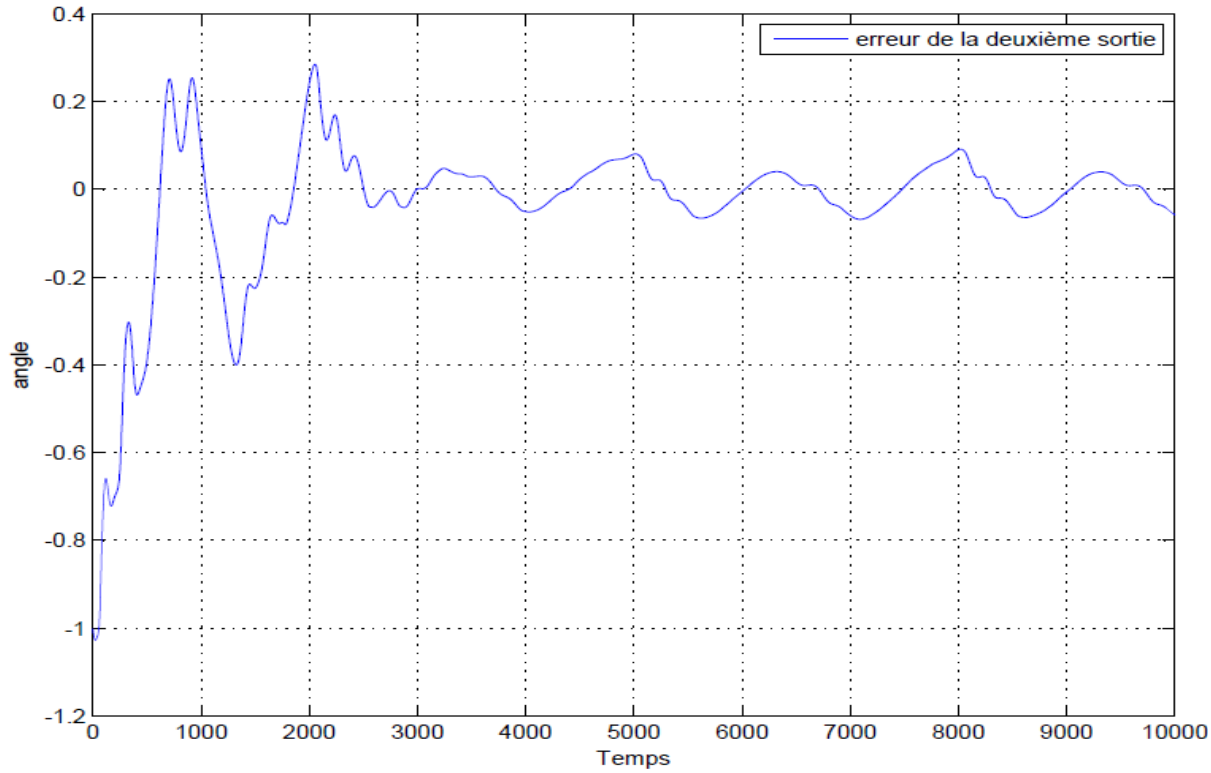


Figure IV.8 : l'erreur entre la deuxième sortie du système et la sortie désirée

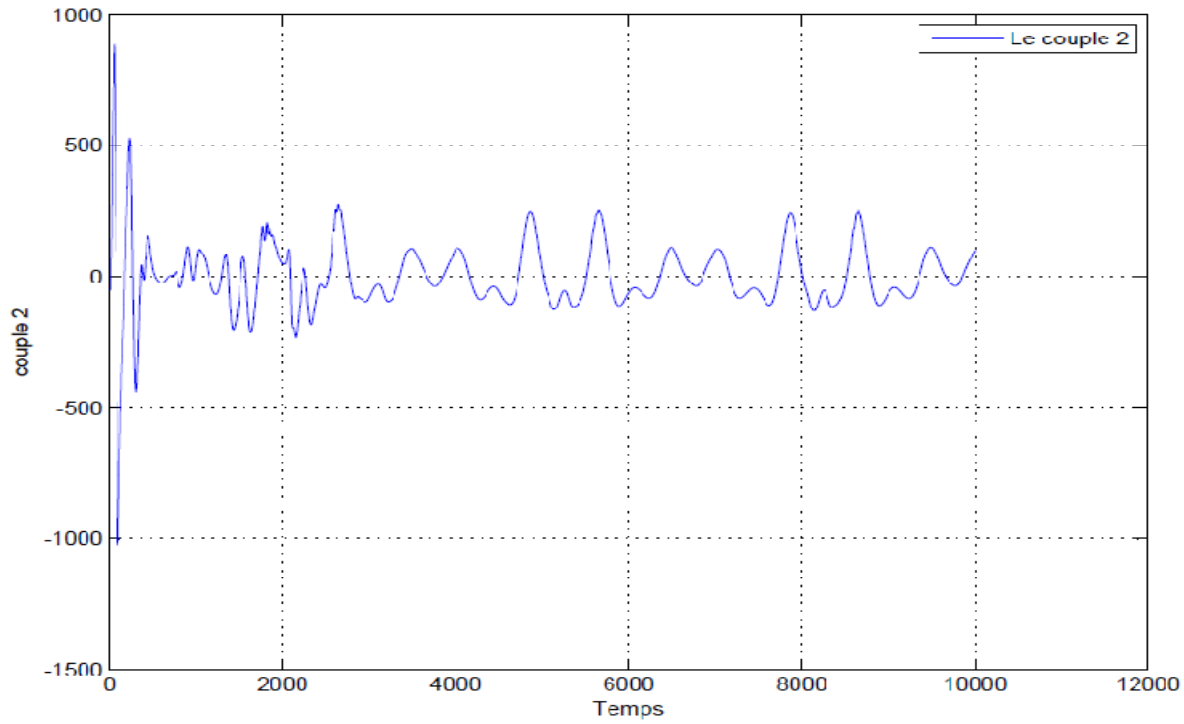


Figure IV.9 : couple 1

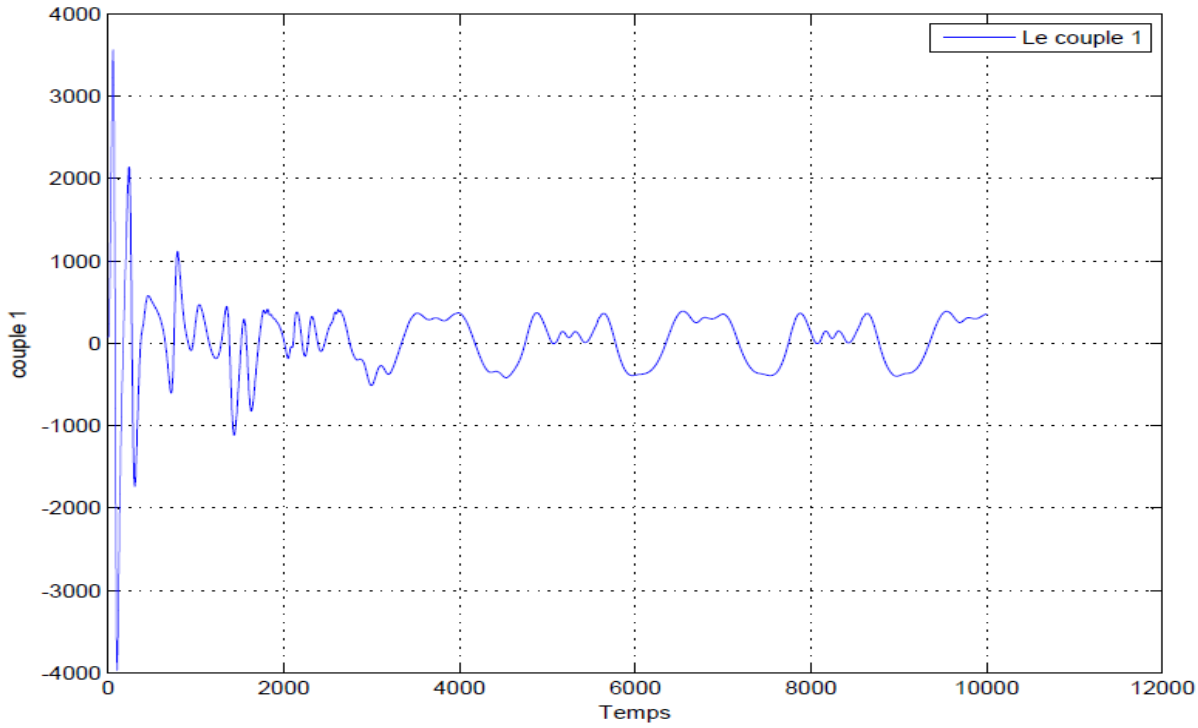


Figure IV.10 : couple 2

IV.9. **Conclusion**

Dans ce chapitre on a fait une étude sur un bras de robot manipulateur, on lui a appliquée une commande neuronal adaptative, les résultats de simulation montre bien l'efficacité de cette méthode dans la commande des bras manipulateur

Conclusion générale

Conclusion général

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

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.

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 (sous l'environnement Matlab) ne peut en aucun cas remplacer une application réelle. Les systèmes de commande appliqués dans ce mémoire ne pourront être validés 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.

Bibliographies

[1] G.P.Liu "Nonlinear Identification and Control, With Neural Networks" Springer 2001

[2] Jagannathan Sarangapani "Neural Network, Control of Nonlinear Discrete Time Systems" Taylor & Francis 2006

[3] BEDDORE Mourad "Contribution d'un réseau de neurones dans une commande d'une chaine de conversion éolienne, mémoire d'ingéniorat université de Bejaia 2011

[4] P.S. Sastry, "Memory Neuron Networks for Identification and Control of Dynamical Systems" IEEE Trans. On Neural Networks. Vol. 5, no.2, MARCH 1994

[5] C. Foulard, S. Gentil et G.P. Sandaraz," Commande et régulation par Calculateur" 3eme édition Eyrolles, 1982.

[6] KUMPATI S. NARENDRA "Neural Networks for Control Theory and Practice" Proceedings of the IEEE Vol. 84, NO. 10, OCTOBER 1996

[7] B. Srinivasan, U. Prasard " Back Propagation Through Adjoints for the Identification of Nonlinear Dynamic Systems Using Recurrent Neural Models", IEEE Trans. On Neural Networks. Vol. 5, no. 2, MARCH 1994

[8] K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Trans. On Neural Networks. Vol. 1, No. 1, MARCH 1990

[9] I. Rivals, L. Personnaz and G. Dreyfus, "Modélisation, classification et commande par réseaux de neurones, principes fondamentaux, Méthodologie de conception et illustrations", 1995

[10] S.Mohamed, M.K.Khelfi, DJ.Chaouch, "Commande neuronale inverse des systèmes non linéaire", 4th International Conférence on Computer Integrated Manufacturing CIP' 2007, 03-04-November 2007

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

[12] SAOUD Mouloud "Commande adaptative à base de réseau de neurones récurrent appliqué sur un système de chauffage bâtiment, mémoire de Master, université de bejaia 2011

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

[14] Claude VIBET " Robots Principes et contrôle" ellipse 1987

[15] W. Khalil, E. Dombre "Modèlisation Identification, et Commande des Robots" 2éme Edition Hermes Paris 1999

[16] R. Kelly, V. Santibáñez and A. Loría "Control of Robot Manipulators in Joint Space" Springer 2005

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

```
% Exemple 01 système SISO


 close all
 clear all
 clc

 % === valeurs initiales ========
 y(1)=0;                        %==
 yd(1)=0;                       %==
 %==============================

 %==== initialisations des poids ======
 w1=rands(5,2);                      %==
 w2=rands(2,5);                      %==
 w3=rands(1,2);                      %==
 %====================================

 j=0:0.1:100;

 %=== signal d'entrer==================
 u=sin(2*pi.*j/25)+sin(2*pi.*j/50); %==
 %====================================


 for k=1:length(u);
     for i=1:2;

 y(k+1)=(y(k)/(1+(y(k))^2))+u(k);% le système

 %==== Propagation direcrt ==========
         inp=[y(k)  u(k)];         %==
         s1=w1*inp';               %==
         u1=purelin(s1);           %==
         s2=w2*u1;                 %==
         u2=purelin(s2);           %==
         s3=w3*u2;                 %==
         u3=purelin(s3);           %==
 %==================================

 %===== sortie du réseau de neurones ===
         yd(k)=2.5*u3;                %==
 %======================================


 %===== calcul des erreurs %==========
         e3=yd(k)-y(k);             %==
         ereur(k)=e3;               %==
         h3=e3.*1/2*(1-u3.^2);      %==
         e2=w3'*h3;                 %==
         h2=e2.*1/2.*(1-u2.^2);     %==
         e1=w2'*h2;                 %==
         h1=e1.*1/2.*(1-u1.^2);     %==
 %==================================
```

```
%=== taux d'apprentissage ========
        mu=0.3;                 %==
%=================================

%==== calcul des gradients =====
        g3=h3*u2';            %==
        g2=h2*u1';            %==
        g1=h1*inp;            %==
%================================

% === adaptation des poids =======
        w1=w1-mu*g1;            %==
        w2=w2-mu*g2;            %==
        w3=w3-mu*g3;            %==
%=================================

    end
end


figure(01)
plot(y)
hold on
plot(yd,'r')

legend('Le système','l"éstimé')
title('identification neuronal')
xlabel('Temps')
ylabel('Le système et le modèle')
grid

figure(02)
plot(ereur,'m')
grid
```

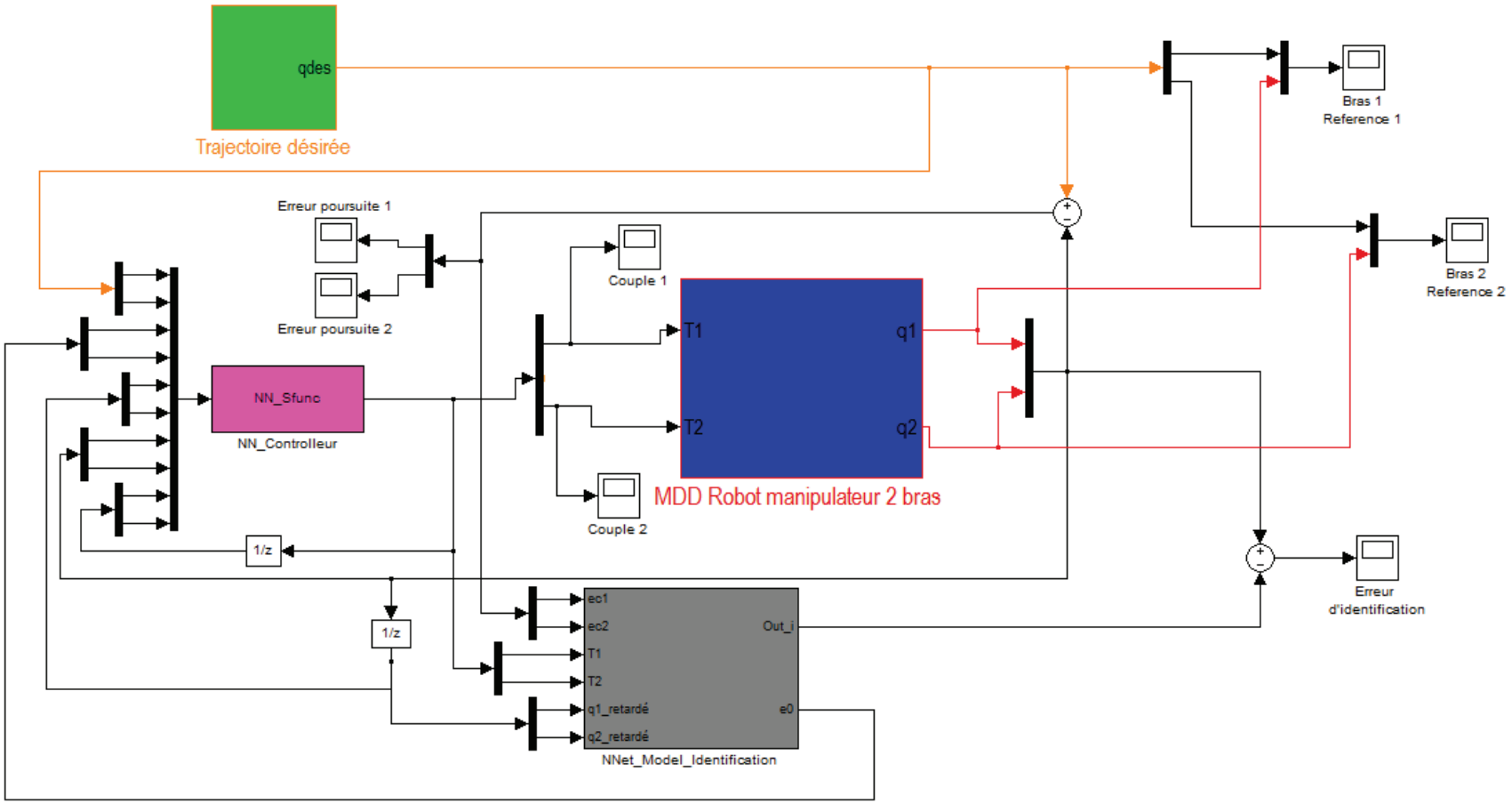
```
% Exemple 02 système SISO

close all
clear all
clc


% === valeurs initiales ========
y(1)=0;                         %==
y(2)=0;                         %==
yd(1)=0;                        %==
%================================

%==== initialisations des poids ======
w1=rands(5,3);                       %==
w2=rands(2,5);                       %==
w3=rands(1,2);                       %==
%======================================


j=0:0.1:100;

%=== signal d'entrer=============
u=sin(2*pi.*j/25);              %==
%================================

for k=2:length(u);

    for i=1:10;

y(k+1)=((y(k)*y(k-1)*(y(k)+2.5))/(1+y(k)^2+y(k-1)^2))+u(k); % le système

%==== Propagation direcrt ===========
        inp=[y(k) y(k-1)  u(k)];  %==
        s1=w1*inp';               %==
        u1=purelin(s1);           %==
        s2=w2*u1;                 %==
        u2=purelin(s2);           %==
        s3=w3*u2;                 %==
        u3=purelin(s3);           %==
%====================================

%===== sortie du réseau de neurones ===
       yd(k)=5*u3;                  %==
%======================================

%===== calcul des erreurs %==========
        e3=yd(k)-y(k);            %==
        ereur(k)=e3;              %==
        h3=e3.*1/2*(1-u3.^2);     %==
        e2=w3'*h3;                %==
        h2=e2.*1/2.*(1-u2.^2);    %==
        e1=w2'*h2;                %==
        h1=e1.*1/2.*(1-u1.^2);    %==
```

```
%====================================

%=== taux d'apprentissage ========
        mu=0.01;                %==
%==================================

%==== calcul des gradients =====
        g3=h3*u2';             %==
        g2=h2*u1';             %==
        g1=h1*inp;             %==
%================================

% === adaptation des poids =======
        w1=w1-mu*g1;           %==
        w2=w2-mu*g2;           %==
        w3=w3-mu*g3;           %==
%================================

    end
end



figure(01)
plot(y)
hold on
plot(yd,'r')

legend('Le système','l"éstimé')
grid

figure(02)
plot(ereur,'m')
grid
```

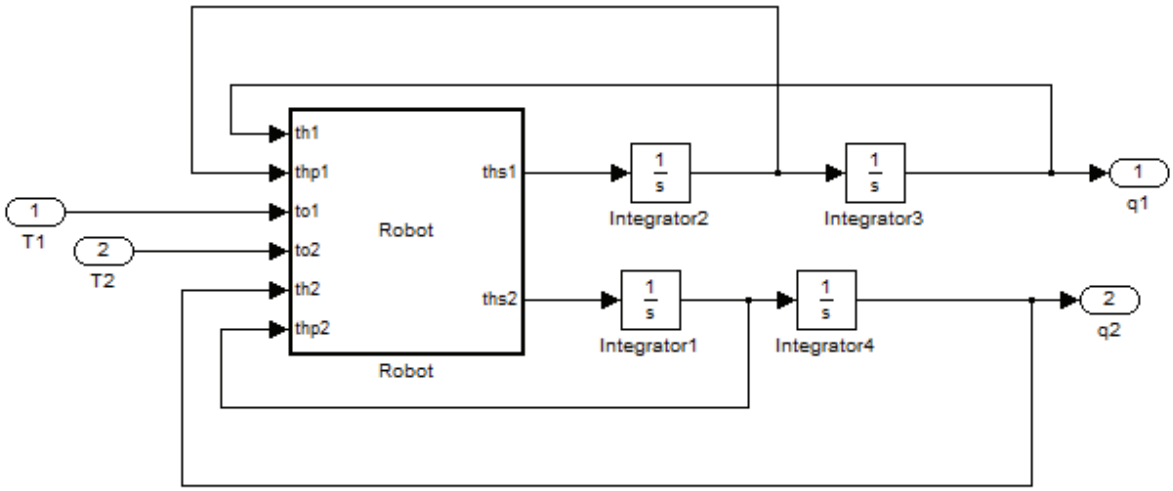
```
% exemple_03 système MIMO
close all
clear all
clc

y1(1)=0;
y2(1)=0;

y1(2)=0;
y2(2)=0;



k=0:1:100;

r1=sin(2*pi.*k/25);
r2=cos(2*pi.*k/25);



for i=2:length(k);


    y1(i+1)=(y1(i)/(1+y2(i)^2))+r1(i);
    y2(i+1)=((y1(i)*y2(i))/(1+y2(i)^2))+r2(i);


end
%=== entrer et sortie de système =====
inputs=[r1;r2];                      %==
targets=[y1(1:end-1);y2(1:end-1)]; %==
%=====================================

 % Create Network
numHiddenNeurons = 25;  % nombres de neurons caché
net = newfit(inputs,targets,numHiddenNeurons);
net.divideParam.trainRatio = 70/100;  % Adjust as desired
net.divideParam.valRatio = 15/100;  % Adjust as desired
net.divideParam.testRatio = 15/100; % Adjust as desired


% Train and Apply Network
[net,tr] = train(net,inputs,targets);
outputs = sim(net,inputs);

in=outputs;

figure(01)
plot(targets(1,:),'b')
hold on
plot(outputs(1,:),'r')
legend('targets1','outputs1')
hold off
```

```
figure(02)
plot(targets(1,:)-outputs(1,:),'k')
legend('error1')

figure(03)
plot(targets(2,:),'b')
hold on
plot(outputs(2,:),'r')
legend('targets2','outputs2')
hold off

figure(04)
plot(targets(2,:)-outputs(2,:),'k')
legend('error2')
```

❖ **Le schéma principal de la commande**



❖ Schéma principale du MDD du Bras manipulateur



❖ Fonction Robot

```
function [ths1,ths2]=Robot(th1,thp1,to1,to2,th2,thp2)

a1=3.82;  % Les
a2=2.12;  % paramètres
a3=0.71;  % du
a4=81.82; % robot
a5=24.06; % manipulateur

% Matrice d'inertie
m11=a1+a2*cos(th2);
m12=a3+(a2/2)*cos(th2);
m21=a3+(a2/2)*cos(th2);
m22=a3;

%Matrice de Coriolis
cq1=-(a2*sin(th2))*(thp1*thp2+((((thp2)^2))/2));
cq2=(a2*sin(th2))*(((thp1).^2)/2);

% vecteur de gravitation
g1=a4*cos(th1)+a5*cos(th1+th2);
g2=a5*cos(th1+th2);

% Inverse de la Matrice d'inertie
detm=m11*m12-m21*m22;
h1=cq1+g1;
h2=cq2+g2;

% Calcule de la sortie
ths1=(m22/detm)*(to1-h1)-(m12/detm)*(to2-h2);
ths2=-(m21/detm)*(to1-h1)-(m11/detm)*(to2-h2);
```

Résumé - Dans la dernière décennie, plusieurs travaux de recherche ont montré l'efficacité de l'application des réseaux de neurones à l'identification et au contrôle des systèmes non linéaires. Dans ce travail nous considérons le problème de l'identification et du contrôle des systèmes non linéaires. Et on présentera l'identification et le contrôle neuronal adaptatif indirect avec modèle de référence d'un bras manipulateur à deux degrés de libertés.

Abstract - In the last decade, several research tasks showed the effectiveness of the application of the neural networks to the identification and the control of the nonlinear systems. In this work we consider the problem of the identification and the control of the nonlinear systems. And one will present the identification and indirect adaptive neural control with reference model of an arm manipulator to two degrees of freedom.