REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE BLIDA

INSTITUT D'ELECTRONIQUE

MEMOIRE DE MAGISTER

SPECIALITE ELECTRONIQUE

OPTION : CONTROLE

THEME

IDENTIFICATION ET CONTROLE DES SYSTEMES DYNAMIQUES PAR LES RESEAUX DE NEURONES. APPLICATION A UNE RUCHE APICOLE

Présenté par : **OUBBATI MOHAMED**

Devant le Jury :

Mr : H. MELIANI	Président	(M.C .Université de BLIDA)
Mr : K.AMMOUR	Examinateur	(M.C .Université de BLIDA)
Mr : M. S. BOUCHERIT	Examinateur	(M.C. E.N.P)
Mr : H. SALHI	Rapporteur	(M.C .Université de BLIDA)
Mr : A. GUESSOUM	Rapporteur	(M.C .Université de BLIDA)
Mr : B. KAZED	Invité	(C.C Université de BLIDA)

BLIDA, ALGERIE

Décembre, 1996

بسم الله الرحمن الرحيم

الإهداء

الى من جعلها اللّه عزّ و جلّ لي آيةً،

الى من دافعت عن الحق في عهد أصبح فيه المعروف منكرا و المنكر معروفا،

الى من قُتلت نتيجة تسيُّب و إهمال مجموعة من المسؤولين تسببوا في إنحطاط مستوى الجامعة، فلا هم عملوا و لا هم تركوا غيرهم يعمل و لا هم رضوا بمنزلة القاعدين التي إستحقوها بتراخيهم، بل وقفوا عقبة في وجه كل إصلاح،

الى شهيدة العلم،

الى الشهيدة و الأخت و الزميلة (نجية ترّاس) رحمها اللّه و أسكنها فسيح جنانه،

و الى كل مصلح في هذه الأمّة الجريحة،

أهدي هذا الجهد المتواضع.

محمد أوباتي

AVANT-PROPOS

Ce travail a été effectué au laboratoire des systèmes et de contrôle dans le cadre d'un projet de recherche, *développement de systèmes de contrôle par les réseaux de neurones et la logique floue,* à l'institut d'électronique (UNIVERSITE DE BLIDA).

J'exprime ma profonde reconnaissance à mes promoteurs Mr.H.SALHI et Mr.A.GUESSOUM, qui en dirigeant ce travail m'ont fait profiter de leurs connaissances, de leurs aide précieuse et de leurs conseils d'or.

Je profite par la même occasion, pour remercier mes professeurs Mme.H.BOUGHRIRA et Mrs.A.MENACER et H.BENCHOUBAN, et tout l'ensemble des enseignants de l'institut d'électronique qui ont contribué à ma formation.

J'exprime aussi toute ma gratitude à toutes les personnes qui n'ont cessé de m'apporter aide et soutien lors de l'élaboration de ce travail, en particulier : M^{elles}.F.Z.DOUDOU, S.HARIOUK, D.BOUTOUTA et Mrs T.SEGHIRATE et M.HADJ SADOK.

Mes remerciements vont aussi aux membres de jury, pour l'honneur qu'ils m'ont accordé en acceptant de juger mon travail.

MOHAMED

الملخص

الهدف من هذا البحث، تبيان إمكانية إستعمال الشبكات العصبية الإصطناعية في مجال التمثيل و التحكم في الأنظمة الغير خطية.

قمنا بإستعراض طريقة التحكم التآلفي للأنظمة الغير خطية بواسطة المثال المرجعي و هذا بإستعمال شبكة عصبية ديناميكية.

في نهاية هذا البحث، إستعملنا النتائج النظرية و التجريبية المحصل عليها في التحكم في نظام فيزيائي حقيقي، و هذا النظام يمثل "خلية النحل" و هدفنا من هذه الدراسة، زيادة كمية العسل الصافي و كمية الشمع.

كلمات مفتاحية:

الشبكة العصبية الاصطناعية، الانظمة الغير خطية، التحكم الغير خطي، الانظمة التآلفية.

ABSTRACT

The main objectif of this thesis is to demonstrate that neural networks can be used effectively for the identification and control of nonlinear dynamical systems, when the states are accessible [1],[2],[4],[5],...*etc. A neural network based model reference adaptive controller (NN-MRAC) is proposed for nonlinear time variant systems using a new paradigm called diagonal recurrent neural network «DRNN »* [12]. *Finally the theoretical and simulation results are used to control a real system. This system is a bee hive and our objective is first to improve the level of the pure honey and the second is to improve the level of the polish.*

KEYWORDS *: Neural network, Nonlinear systems, Nonlinear control, Adaptive systems.*

RESUME

Le but de cette thèse est de montrer des techniques neuronales pour l'identification et contrôle des systèmes non linéaires, quand ses états sont accessibles [1],[2],[4],[5],...*etc. La stratégie du contrôle adaptatif avec modèle de référence est proposée pour le contrôle des systèmes non linéaires variants avec le temps en utilisant une nouvelle architecture de réseau appelé « Diagonal recurrent neural network » DRNN* [12]. *Finalement les résultats théoriques et expérimentaux sont utilisés pour contrôler un système réel qui est la* ***RUCHE APICOLE****, où nous sommes intéressé par l'augmentation de la quantité du miel pur et de la quantité de cire.*

MOTS CLES *:Réseaux de Neurones, Systèmes non linéaires, Contrôle non linéaire, systèmes adaptatifs.*

TABLE DES MATIERES

CHAPITRE II *IDENTIFICATION DES SYSTEMES PAR LES RESEAUX DE NEURONES*

CHAPITRE III *CONTROLE DES SYSTEMES PAR LES RESEAUX DE NEURONES*

CHAPITRE IV *CONTROLE D'UNE RUCHE APICOLE PAR LES RESEAUX DE NEURONES*

INTRODUCTION GENERALE

INTRODUCTION GENERALE

La théorie de contrôle est basée sur des méthodes d'analyse qui sont adaptées aux systèmes linéaires. Ces méthode supposent toujours que le domaine d'opération est petit. Quand ce domaine d'opération est large (ex : actionneur dans sa saturation), un contrôleur linéaire va sûrement résulter dans la détérioration des performances, allant parfois à l'instabilité du système en boucle fermée. Ce point peut être illustré très clairement dans le contrôle des robots où une commande linéaire négligeant les non linéarités (qui peuvent être sévères), résulte dans la dégradation des performances. Par contre une loi non-linéaire permet de compenser ces effets.

Une autre supposition des commandes linéaires est que le système physique est linéarisable autour d'un point d'équilibre ou une trajectoire nominale. Cependant, il y a beaucoup de non-linéarités discontinues. De nouvelles méthodes de calcul qui doivent prendre en compte les caractéristiques particulières des systèmes non-linéaires s'avèrent nécessaires.

Dans ce travail, nous sommes intéressés par l'identification et la commande adaptative des systèmes non-linéaires. Le contrôle est effectué en utilisant des approximateurs de fonctions, basés sur l'interconnexion de plusieurs éléments de base qui sont non-linéaires. Ces approximateurs de fonctions sont appelés « Réseaux de Neurones Artificiels». Les résultats théoriques et expérimentaux obtenus seront utilisées pour contrôler un système réel qui est la **RUCHE APICOLE** où nous nous sommes intéressés à l'augmentation de la quantité du miel pur et de la quantité de cire.

Un réseau de neurones artificiels est une modélisation mathématique de la merveilleuse machine que constitue le cerveau. La capacité de ce dernier d'accomplir des opérations très complexes à partir des éléments de base que constituent les neurones a fasciné les chercheurs, qui ont d'abord commencé par modéliser le neurone. Le premier modèle fut réalisé par **McCULLOCH** et **PITTS** en 1943 [8]. Dans ce modèle, un neurone accomplit la somme pondérée des potentiels d'actions qui lui parviennent, puis s'active si celle-ci dépasse un certain seuil, en transmettant une réponse dont la valeur correspond à son activation.

Comme pour le cerveau, un réseau de neurones artificiels est un ensemble de neurones reliés entre eux. La sortie de chaque neurone peut être reliée en entrée à plusieurs

autres neurones. Un réseau de neurones peut être divisé en plusieurs couches. Les neurones d'entrée du réseau sont reliés au monde extérieur; les neurones de sortie donnent le résultat et entre les deux une ou plusieurs couches d'autres neurones appelées couches cachées.

Deux classes de réseaux de neurones ont reçu plus d'attention ces dernières années: les réseaux statiques et les réseaux dynamiques. Les réseaux statiques ont montré un grand succés dans la reconnaissance des données (caractères, images, formes, objets,...). D'autre part les réseaux dynamiques ont été utilisés dans les mémoires associatives aussi bien que pour la solution des problèmes d'optimisation [34], [35].

L'utilisation des réseaux de neurones dans la commande des systèmes non-linéaires peut être vue comme une évolution naturelle des techniques de commande.

Cette évolution est fondée sur plusieurs points :

1. Capacités limitées de la part des régulateurs classiques.
2. Analyse des non-linearités dures.
3. Utilisation d'un nombre minimal d'informations sur le système.

Le comportement dynamique des systèmes physiques étudiés pose un problème pour l'intégration de cette dynamique dans les réseaux statiques. On peut résoudre ce problème soit en utilisant les réseaux récurrents, soit en introduisant un comportement dynamique dans les neurones [8], soit en augmentant le nombre d'entrées du réseau, en considérant les sorties et les entrées précédentes [1],[2],[3],[4],[5],[6]. Dans ce cadre, quatre modèles sont introduits pour la représentation des systèmes non-linéaires monovariables avec la possibilité de généraliser aux systèmes multivariables [1].

Ces modèles ne sont en fait qu'une extension des modèles utilisés dans les systèmes linéaires aux systèmes non-linéaires.

Modèle I : Dans ce modèle, la sortie du système non-linéaire inconnu est supposée dépendante linéairement de ces valeurs précédentes et non linéairement des valeurs passées de l'entrée. L'avantage de ce modèle est qu'il permet de discuter la stabilité en régime libre.

Modèle II : La sortie du système non-linéaire dans ce cas est supposée dépendante linéairement de l'entrée et de ses valeurs passées, et non-linéairement de ses valeurs anciennes. L'avantage de ce modèle est qu'il est directement applicable dans la commande par réseaux de neurones.

Modèle III : Dans ce cas, la sortie du système non-linéaire inconnu dépend non-linéairement à la fois de ses valeurs passées et des valeurs passées de l'entrée.

Modèle IV : Ce modèle est le plus général, puisqu'il englobe les précédents modèles. La sortie à n'importe quel instant dans ce cas est une fonction non-linéaire des valeurs passées de l'entrée et de la sortie à la fois.

Ensuite ces modèles sont utilisés dans le contrôle adaptatif des systèmes non-linéaires. Dans la simulation, ces modèles ont donné de très bons résultats, mais le temps d'apprentissage des réseaux statiques multicouches est très long. Ceci est un inconvénient dans le contrôle des systèmes variant avec le temps.

Une autre méthode de contrôle utilise les réseaux de neurones multicouches pour l'identification de la dynamique inverse des systèmes non-linéaires. Ce modèle inverse sera simplement mis en cascade avec le système afin que le système résultant soit une fonction identité [2].

Cette méthode présente une grande promesse pour la recherche, car les modèles inverses des systèmes dynamiques ont une utilité immédiate dans la commande. Le problème majeur dans l'identification inverse a lieu quand plusieurs entrées produisent la même sortie. Autrement dit, quand l'inverse du système n'est pas bien défini. Dans ce cas, le réseau de neurones tend à faire correspondre les mêmes entrées à différentes sorties.

Le contrôle par le régulateur auto-ajustable a été proposé par [Chen, 1990] [33] sous une version neuronale en utilisant le réseau de neurones multicouches. Le régulateur auto-ajustable est essentiellement un retour classique avec des paramètres ajustés en temps réel. Ce régulateur est composé de deux boucles, la boucle interne est constituée par le système à commander et d'un régulateur ordinaire tandis que la boucle externe contient l'algorithme d'apprentissage permettant d'ajuster les paramètres du régulateur.

Le problème majeur dans tous ces travaux est toujours le temps d'apprentissage des réseaux de neurones multicouches. Une solution proposée pour minimiser le temps d'apprentissage, est d'utiliser le contrôle adaptatif direct où l'ajustement des poids synaptiques du contrôleur neuronal se fait directement sans passer par le modèle d'identification [11]. Dans [11] une structure de contrôle adaptatif direct est proposée. Le problème dans cette structure est la détermination de la sensibilité du système.

La performance de cette technique pour le contrôle des systèmes non-linéaires est démontrée par comparaison avec la méthode de Lyapunov. Dans la simulation, deux études de cas réels ont été présentés dans lesquelles le contrôle adaptatif direct basé sur un réseau de neurone multicouche est utilisé pour améliorer d'une part le contrôle d'un réacteur de char (CSTR) et d'autre part le contrôle d'un missile (BTT-CLOS).

La plupart des études sur le contrôle des systèmes par les réseaux de neurones sont basées sur le réseau multicouche combiné avec des temps de retard et l'algorithme de la rétro-propagation [1],[2],[4],[5],[6],[10],[11],[22]. Malgré la capacité prouvée de ce réseau dans le domaine du contrôle des systèmes non-linéaires, il a l'inconvénient du temps d'apprentissage qui est très long. Puisque en pratique il y a la contrainte du temps réel, alors des recherches se sont poursuivies pour trouver d'autres architectures de réseau de neurones possédant un temps d'apprentissage très court [12],[28],[30],[31],[32]. Ces recherches sont surtout dirigées vers les réseaux récurrents ou dynamiques, car ils possèdent un temps d'apprentissage réduit. Les chercheurs [Chao-Chee Ku and Kwang Y-Lee, 1995] ont développé une nouvelle architecture de réseau de neurones dynamique appelée *(Diagonal Recurrent Neural Network)* DRNN [12]. Ce réseau possède trois couches, dont la couche cachée est entièrement connectée. C'est cette architecture qui sera détaillée dans le chapitre III de ce mémoire.

Ce travail comporte quatre chapitres :

1. Les réseaux de neurones artificiels.
2. Identification des système par les réseaux de neurones.
3. Contrôle des systèmes par les réseaux de neurones.
4. Contrôle d'une RUCHE APICOLE par les réseaux de neurones.

Le premier chapitre concerne l'étude de plusieurs réseaux de neurones où on subdivisera les réseaux de neurones en deux classes, les réseaux statiques et les réseaux dynamiques en présentant les algorithmes d'apprentissage de chaque classe.

Le second chapitre permet d'aborder le problème de l'identification des systèmes par les réseaux de neurones. Le principe du traitement par les réseaux de neurones sera utilisé pour stocker toutes les informations du système dans un réseau de neurones appelé modèle neuronal; ce chapitre se termine par des simulations.

Le troisième chapitre utilise également les caractéristiques d'un réseau de neurones pour le contrôle adaptatif des systèmes non-linéaires. On présente plusieurs méthodes de contrôle basées sur les réseaux de neurones statiques et dynamiques. On termine ce chapitre par des simulations montrant la robustesse du réseau DRNN en temps réel.

Le quatrième chapitre a comme objectif, l'utilisation des réseaux de neurones pour le contrôle d'un système physique qui est une « RUCHE APICOLE ». Le contrôle est sous forme d'un nourrissement artificiel des abeilles pendant l'insuffisance qualitative et quantitative de la nourriture à certaines périodes de l'année afin de préparer les abeilles pour une miellée. Dans ce chapitre, un résumé des travaux antérieurs est présenté dans le contrôle de la RUCHE APICOLE [14],[15],[16]. Ensuite on montre les résultats de l'identification de la RUCHE APICOLE par les réseaux de neurones. Enfin le contrôle est effectué pour atteindre les objectifs souhaités par l'apiculteur est vérifié par la simulation.

CHAPITRE I

LES RESEAUX DE NEURONES ARTIFICIELS

I.1. INTRODUCTION

Depuis son existence, l'homme n'a cessé de poser des questions comme celles concernant les phénomènes qui l'entourent, par exemple, pourquoi une pierre une fois lancée finit-elle par retomber? Si cette question a pu être posée c'est grâce à la merveilleuse machine à penser que constitue le cerveau. Jusqu'au jour où l'homme se posa une question qui est la base de ce travail. Cette question concerne l'analyse et la compréhension du principe de fonctionnement de ce qui lui permettait d'analyser et de comprendre. Cela est la partie philosophique du problème, car analyser ce qui nous permet d'analyser n'est pas si facile qu'on puisse l'imaginer.

En ce qui nous concerne , nous sommes intéressés par le cerveau lui-même et sa modélisation.

Il est estimé que le cerveau humain contient $\mathbf{10^{11}}$ neurones et $\mathbf{10^{14}}$ synapses dans le système nerveux [23]. L'étude de la neuro-anatomie montre qu'il y a plus de **1000** synapses en entrée et en sortie de chaque neurone [23]. Les neurones et les synapses constituent les éléments de base pour le traitement de l'information. La plupart des neurones possèdent les dendrites qui reçoivent des signaux provenant des autres neurones par l'intermédiaire des jonctions synaptiques. Une représentation simplifiée d'un réseau de neurones biologique est illustré par la figure (1.1).

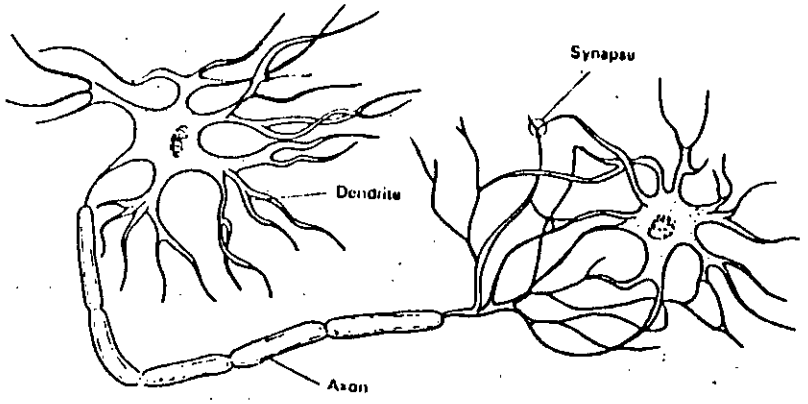


Figure 1.1 Un réseau de neurones biologiques.

Il y a trois parties dans un neurone.

1. Le corps de la cellule.
2. Les branches d'extension appelé dendrites.
3. L'axone qui fait la liaison entre la sortie du neurone avec les dendrites des autres neurones.

Le synapse représente la jonction entre un axone et une dendrite.

Un seul neurone n'est capable de rien faire ou presque. C'est l'interconnexion des neurones qui permet au cerveau de réaliser des tâches telles que reflexion,vision,perception....On appelle l'interconnexion de ces neurones **« un réseau de neurones ».**

Le réseau de neurones fonctionne de la façon suivante: la membrane génère une action potentielle qui se propage vers un axone, qui à son tour amplifie ce signal pour qu'il se propage jusqu'à la jonction synaptique. Le neurone rassemble les signaux à ses synapses en sommant les influences excitatrices et inhibitrices. Si les influences excitatrices dominent alors le neurone envoie un message vers d'autres neurones via les synapses de sorties.

Depuis quelques années, des chercheurs de part le monde relèvent le défi de construire des circuits électroniques imitant la capacité exceptionnelle dont font preuve les neurones organisés en réseaux qui constituent notre cerveau. On assiste à une évolution sur ce qu'il est convenu d'appeler des **réseaux de neurones formels** ou des **machines neuronales.**

Un réseau de neurones fonctionne sans programme, n'exécute pas d'instruction, ne contient pas de mémoire pour y stocker des données, ne manipule pas des nombres. La destruction d'une partie de ces circuits ne l'empêche pas de fonctionner. Le réseau peut même dans certains cas récupérer.

Souvent dans la pratique, les chercheurs ne font pas appel à de véritables neurones électroniques, mais seulement à des programmes sur ordinateur conventionnel. Malgré la limitation technologique de ces recherches qui portent sur des réseaux de quelques dizaines de neurones artificiels, des résultats satisfaisants ont été enregistrés.

I.2. LE NEURONE ARTIFICIEL

Le premier modèle de base d'un réseau de neurones est proposé en 1943 par **Mc-Culloch** et **Pitts** [8].Le neurone de **McCulloch -Pitts** est un mécanisme binaire. Il calcule la somme de ces entrées, la valeur de chaque entrée étant modulée par le poids synaptique w_{ji} correspondant et prend une décision en comparant cette somme à un seuil qui lui est propre.

Si la somme est supérieure au seuil, le neurone se met dans son état actif (+1). Dans le cas contraire il se met dans son état inactif (-1) (figure 1.2). Donc le neurone peut être représenté comme suit :

- les entrées ($x_1, x_2, \ldots, x_n$).
- les poids synaptiques ($w_{i1}, w_{i2}, \ldots, w_{in}$).
- le seuil θ_i
- l'état du neurone $u_i(w,x) = \sum_{j=1}^{n} w_{ij} x_j - \theta_i$
- la fonction d'activation : $f(x) = \begin{cases} -1 & \text{si } x \le 0 \\ +1 & \text{si } x > 0 \end{cases}$
- la sortie du neurone i est : $y_i = \begin{cases} +1 & \text{si } u_i(w,x) > 0 \\ -1 & \text{si } u_i(w,x) < 0 \end{cases}$

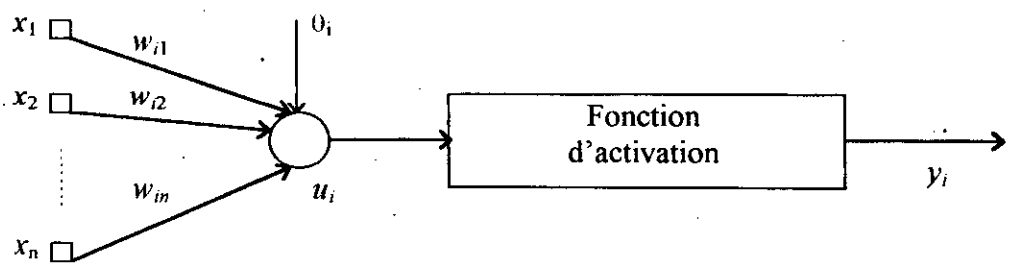


Figure 1.2 Modèle de base d'un neurone artificiel.

En général, un neurone artificiel est caractérisé par la fonction de base et la fonction d'activation.

1.2.1. *LA FONCTION DE BASE*

Pour une étude analytique, les connexions d'un réseau sont représentées mathématiquement par la fonction $u_i(w,x)$ avec w et x représentent respectivement la matrice des poids synaptiques et le vecteur d'entrée. La fonction $u_i(w,x)$ possède généralement deux formes.

1. La fonction (LBF) « *linear-basis function* » qui représente une combinaison linéaire des entrées.

$$u_i(W,X)=\sum_{j=1}^{n} w_{ji}x_j \qquad (1.1)$$

2. La fonction (RBF) « *radial-basis function* »

$$u_i(W,X)=\sqrt{\sum_{j=1}^{n}\left(x_j-w_{ij}\right)^2} \qquad (1.2)$$

Le second ordre de cette fonction peut être étendu pour avoir un cas plus général.

1.2.2. *LA FONCTION D'ACTIVATION*

Les fonctions d'activation les plus utilisées sont les fonctions escalier,rampe sigmoïde et gaussienne.

.Fonction sigmoïde

$$f(u_i)=\frac{1}{1+e^{\frac{-u_i}{\sigma}}} \qquad (1.3)$$

. Fonction gaussienne

$$f(u_i)=c\,e^{\frac{u_i^2}{\sigma^2}} \qquad (1.4)$$

I.3. LES RESEAUX DE NEURONES ARTIFICIELS

Les processeurs neuronaux sont inspirés du modèle du cerveau humain. Comme pour le cerveau, un réseau de neurones artificiels est un ensemble de neurones reliés entre eux. La sortie de chaque neurone peut être reliée en entrée à plusieurs autres neurones. La complexité du réseau n'est définie que par le nombre de neurones et le nombre de connexions. Le réseau peut être divisé en plusieurs couches. Les neurones d'entrée du réseau sont reliés au monde extérieur; les neurones de sortie donnent le résultat et entre les deux une ou plusieurs couches d'autres neurones.

La question la plus intrigante est celle de l'apprentissage: comment un réseau de neurones peut-il apprendre un comportement ? Comment peut-il apprendre, par exemple, à reconnaître des images ou des sons ou une fonction mathématique ?

I.4. TRAITEMENT DE L'INFORMATION PAR LES RESEAUX DE NEURONES

Dans un ordinateur, l'unité mémoire passe par les phases écriture et lecture. Durant la phase écriture, un mécanisme de stockage est utilisé pour spécifier l'information à se rappeler. L'information stockée sera restituée durant la phase de lecture. Par analogie avec les phases d'écriture et de lecture dans l'ordinateur, il existe aussi deux phases dans le traitement de l'information par les réseaux de neurones. Il y a la phase d'apprentissage et la phase de reconnaissance.

I.4.1. *LA PHASE APPRENTISSAGE*

Toute l'information que possède un réseau de neurones est représentée dans les poids synaptiques. Le réseau acquiert cette information pendant la phase d'entraînement ou d'apprentissage.

Au cours de la phase d'apprentissage, les poids synaptiques sont ajustés tel que le réseau remplisse une tâche définie par des exemples. L'apprentissage est défini comme tout changement dans les poids synaptiques.

Les règles d'apprentissage peuvent être divisées en deux catégories:

- l'apprentissage supervisé.
- l'apprentissage non supervisé.

1.4.1.1. *APPRENTISSAGE SUPERVISE*

Dans un apprentissage supervisé, on présente au réseau de neurones les entrées et les sorties désirées correspondantes. Les poids synaptiques sont ajustés de manière à minimiser un certain critère qui est la mesure de la qualité de la réponse du réseau à l'ensemble des couples choisis pour l'apprentissage.

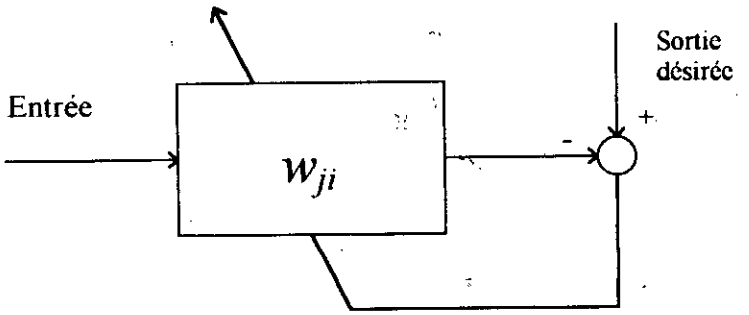


Figure 1.3 Apprentissage supervisé

1.4.1.2. *APPRENTISSAGE NON SUPERVISE*

Pour ce type d'apprentissage, l'adaptation des poids synaptiques n'est pas basée sur la comparaison avec une certaine sortie désirée. Dans ce cas le réseau organise lui même les entrées qui lui sont présentées de façon à optimiser un critère de performance donné.

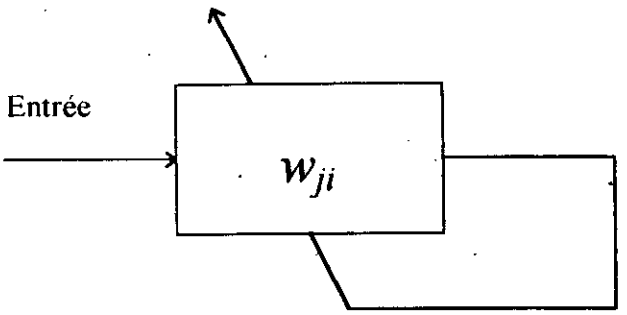


Figure 1.4 Apprentissage non supervisé

1.4.2. *LA PHASE RECONNAISSANCE*

Une fois l'apprentissage effectué le réseau peut être utilisé pour la tâche prévue. En phase d'utilisation, les performances du réseau sont évaluées à l'aide d'un ensemble d'exemples dit ensemble de test.

I.5. ARCHITECTURES DES RESEAUX DE NEURONES

Il existe deux types d'architectures de réseaux de neurones:

- les réseaux statiques.
- les réseaux dynamiques.

1.5.1. *LES RESEAUX DE NEURONES STATIQUES*

Dans un réseau statique, la sortie du neurone ne peut pas être injectée ni directement à son entrée, ni indirectement à travers d'autres neurones, c'est à dire qu'une sortie courante n'a aucune influence sur les sorties futures.

Les réseaux de neurones statiques réalisent une transformation non linéaire de la forme

$$\mathbf{U=G(X)} \text{ où } X \in R^n \text{ et } u \in R^m.$$

Dans ce chapitre , on va présenter deux types de réseaux statiques. Le réseau à multicouches (MLP) « *multilayer perceptron* » et le réseau modulaire.

1.5.1.1. *LE RESEAU MULTICOUCHES (MLP)*

Le réseau MLP est composé d'une couche d'entrée, une couche de sortie et plusieurs couches cachées. Chaque neurone de la couche *l* est connecté à tous les neurones de la couche *l*+1.(fig.1.5)

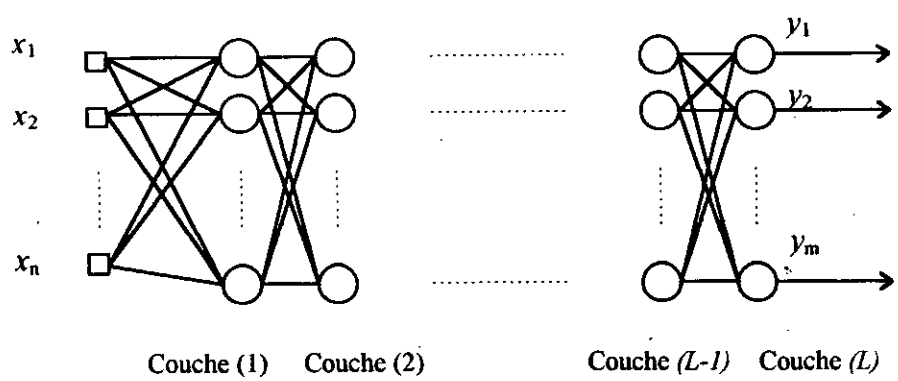


Figure 1.5. Le réseau à multicouches (MLP).

a). *L'ALGORITHME D'APPRENTISSAGE DU RESEAU MLP*

Cet algorithme représente l'algorithme de ***la rétropropagation de l'erreur*** [3].

a.1. *Notations*

Avant de commencer la présentation ,il faut introduire les notations données par le tableau suivant:

Notations	Signification
u_{lj}	La sortie du j ^{eme} neurone de la couche l
$w_{l,j,i}$	Poids entre le i ^{eme} neurone de la couche l-1 et le j ^{eme} neurone de la couche l
$u_{0,i}$	i ^{eme} élément du vecteur d'entrée
$d_j(x_p)$	La sortie désirée du j ^{eme} neurone de la couche de sortie pour la donnée x_p
N_l	Nombre de neurones dans la couche l
L	Nombre de couches
P	Nombre de données d'apprentissage
x_p	Les données d'apprentissage

Tableau 1.1 Notations de l'algorithme de la rétropropagation

a.2. *Les équations du réseau MLP*

La sortie du j^{eme} neurone de la couche l est

$$u_{l,j}=f(\sum_{i=0}^{Nl-1} w_{l,j,i}u_{l-1,i}) \qquad (1.5)$$

où $f(.)$ est la fonction sigmoïde (1.3) ayant pour dérivée

$$\frac{df(\alpha)}{d\alpha} = f(\alpha)(1-f(\alpha)) \qquad (1.6)$$

a.3. *Principe de l'algorithme*

L'objectif de la méthode de la rétropropagation est d'adapter les poids synaptiques $w_{l,j,i}$ de façon à minimiser la valeur moyenne de l'erreur sur l'ensemble d'entraînement.

L'algorithme d'apprentissage est basé sur la méthode du gradient. La fonction à minimisée est le critère de performance suivant:

$$J(w) = \sum_{p=1}^{P} J_p(w) \tag{1.7}$$

où

$$J_p(w) = \frac{1}{2}\sum_{q=1}^{N_L}(u_{Lq}(x_p) - d_q(x_p))^2 \tag{1.8}$$

Les poids seront ajustés récursivement par l'équation suivante:

$$w_{lji}(k+1) = w_{lji}(k) - \eta \frac{\partial J(w)}{\partial w_{lji}} \tag{1.9}$$

$$w_{lji}(k+1) = w_{lji}(k) - \eta \sum_{p=1}^{P} \frac{\partial J_p(w)}{\partial w_{lji}} \tag{1.10}$$

où η est une constante positive appelée « *le taux d'apprentissage* ».

a.4. *Principe de la rétropropagation de l'erreur*

Pour adapter les poids synaptiques, il faut développer une expression de la dérivée partielle de J_p par rapport à ces poids.

$$\frac{\partial J_p(W)}{\partial w_{lji}} = \frac{\partial J_p(w)}{\partial u_{lj}} \frac{\partial u_{lj}}{\partial w_{lji}} \tag{1.11}$$

où

$$\frac{\partial u_{lj}}{\partial w_{lji}} = \frac{\partial}{\partial w_{lji}}\left[f\left(\sum_{m=0}^{N_{l-1}} w_{ljm}\, u_{l-1,m} \right)\right] \tag{1.12}$$

$$= f'\left(\sum_{m=0}^{N_{l-1}} w_{ljm} u_{l-1,m} \right) \frac{\partial}{\partial w_{lji}} \left[\sum_{m=0}^{N_{l-1}} w_{ljm} u_{l-1,m} \right]$$

$$= f'\left(\sum_{m=0}^{N_{l-1}} w_{ljm}\, u_{l-1,m} \right) u_{l-1,i} \tag{1.13}$$

En utilisant les équations (1.13) et (1.6) nous aurons:

$$\frac{\partial u_{lj}}{\partial w_{lji}} = u_{lj}\left(1-u_{lj}\right)u_{l-1,i}$$

Par conséquent

$$\frac{\partial J_p(w)}{\partial w_{lji}} = \frac{\partial J_p(w)}{\partial u_{lj}} u_{lj}(1-u_{lj})u_{l-1,i} \qquad (1.14)$$

Le terme $\partial J_p(w)/\partial u_{l,j}$ représente la sensibilité de $J_P(w)$ à la sortie du neurone $u_{l,j}$. Donc il peut être écrit en fonction des sensibilités par rapport aux sorties des neurones de la couche suivante,

$$\frac{\partial J_p(w)}{\partial u_{lj}} = \sum_{m=1}^{N_{l+1}} \frac{\partial J_p(w)}{\partial u_{l+1,m}} \frac{\partial u_{l+1,m}}{\partial u_{lj}}$$

$$= \sum_{m=1}^{N_{l+1}} \frac{\partial J_p}{\partial u_{l+1,m}} \frac{\partial}{\partial u_{lj}} \left[f\left(\sum_{q=0}^{N_l} w_{l+1,m,q}\, u_{l,q} \right) \right]$$

$$= \sum_{m=1}^{N_{l+1}} \frac{\partial J_p}{\partial u_{l+1,m}} u_{l+1,m}\left(1-u_{l+1,m}\right) w_{l+1,m,j} \qquad (1.15)$$

Ce processus continue avec $\dfrac{\partial J_p(w)}{\partial u_{l+1,m}}$ et ainsi de suite jusqu'à la couche de sortie.

A la couche de sortie, nous avons:

$$\frac{\partial J_p}{\partial u_{L,j}} = u_{L,j}(x_p) - d_j(x_p) \qquad (1.16)$$

a.5. *Résume de l'algorithme de la rétropropagation*

- Première étape : initialiser les poids synaptiques $w_{l,j,i}$ à de petites valeurs aléatoires.
- Deuxième étape : présenter un exemple X=[x_0 ,x_1, ... , x_n] et la sortie désirée
  D=[d_0, d_1, ... , d_m].
- Troisième étape :
  - calculer la sortie actuelle du réseau en utilisant (1.5).
  - pour la couche de sortie, calculer la dérivée en utilisant (1.16).
  - pour la couche cachée, calculer les dérivées en utilisant (1.15)
  - Ajuster les poids $w_{l,j,i}$ en utilisant (1.10).

- Quatrième étape : répéter depuis la deuxième étape jusqu'à la convergence de l'algorithme.

a.6. *RECOMMANDATIONS POUR L'UTILISATION DE L'ALGORITHME*

choix du facteur de taux d'apprentissage η

Le facteur de taux d'apprentissage peut être choisi de différentes manières. Il peut être le même pour chaque poids, différent pour chaque couche, différent pour chaque neurone ou différent pour chaque poids. En général il est difficile de déterminer le meilleur taux d'apprentissage. Une méthode simple est d'ajouter un terme de la forme $\alpha [w_{ji}(k)-w_{ji}(k-1)]$ à chaque ajustement d'un poids, avec $0<\alpha<1$ [3]. Le processus de calcul du gradient et l'ajustement des poids est répété jusqu'à atteindre un minimum.

Test d'arrêt

En pratique il est difficile d'accomplir la terminaison automatique de l'algorithme, mais il y a des méthodes pour cela. La première méthode est basée sur le test de l'amplitude du gradient car, par définition, il sera nul lorsqu'on a un minimum. La seconde consiste à arrêter le calcul quand le critère de performance sera inférieur à un certain seuil. Cette méthode exige une connaissance de la valeur min $J(w)$. La troisième consiste à fixer un certain nombre d'itérations, quoique cette méthode ne garantit pas l'arrêt à un minimum. Finalement la méthode de « *cross-validation* » peut être utilisée pour tester les performances de généralisation durant l'apprentissage. Cette méthode consiste à diviser les données en deux parties, une partie pour l'apprentissage et l'autre partie pour le test de généralisation du réseau. L'inconvénient de cette méthode est qu'elle exige plus de calcul par rapport aux autres méthodes.

Choix de la taille du réseau

En général on ne peut pas savoir la taille exacte du réseau pour un problème donné. Une méthode consiste à démarrer avec un réseau très petit (ex: un seul neurone) puis ajouter les neurones jusqu'à atteindre de bonnes performances. Une autre méthode consiste à démarrer avec un réseau large puis éliminer les neurones [3].

1.5.1.2 *LE RESEAU DE NEURONES MODULAIRE*

Le principe de base d'un réseau de neurones modulaire est comme suit :

Un réseau de neurones est dit ***Réseau Modulaire*** *si le calcul à exécuter par le réseau peut être décomposé en deux ou plusieurs modules. Ces modules fonctionent sans communiquer l'un avec l'autre. Les sorties des modules sont gérées par le réseau de coordination « gating network ». En particulier, le réseau de coordination décide comment les sorties des modules doivent être combinées pour former la sortie finale du réseau modulaire, et décide quel module doit apprendre quelle tâche* [8].

Dans un tel réseau, une tâche est divisée en plusieurs tâches simples, puis chaque module fait l'apprentissage d'une tâche ,et c'est le rôle du réseau de coordination de gérer ces modules pour avoir la sortie finale du réseau. Chaque module est appelé réseau expert (Figure 1.7).

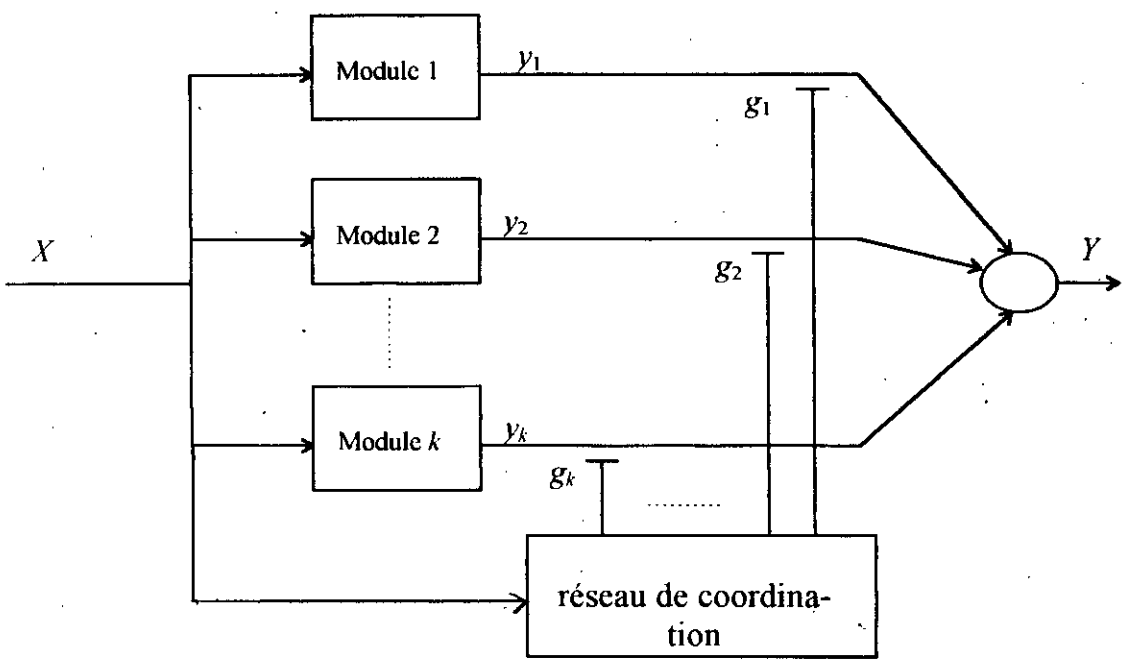


Figure 1.7. Le réseau de neurones modulaire

a). Avantages du réseau modulaire

Le réseau modulaire possède plusieurs avantages par rapport à un seul bloc de réseau de neurones(ex: MLP).

. *La vitesse d'apprentissage*

Si une fonction complexe peut être divisée en plusieurs fonctions simples, alors le réseau modulaire peut facilement détecter cette décomposition et il fait l'apprentissage de plusieurs fonctions simples. Donc il est plus rapide que l'apprentissage d'une fonction complexe par un MLP (fig.1.8).

Supposons que nous voulons approximer la fonction définie par :

$$g(x) = \begin{cases} x & \text{si} \quad x \geq 0 \\ -x & \text{si} \quad x < 0 \end{cases}$$

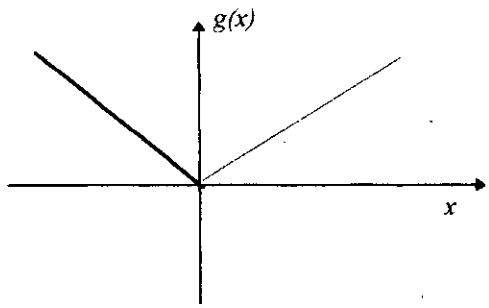


Figure 1.8 La fonction *g(x)*

Si un MLP est utilisé on aura à la limite une couche cachée de neurones. Par contre avec un réseau modulaire on aura besoin seulement de deux neurones, un neurone apprend la fonction $g(x) = x$; $x > 0$,et l'autre neurone apprend la fonction $g(x) = -x$; $x \leq 0$. Le rôle du réseau de coordination est de sélectionner le neurone correspondant à la valeur de x.

• *La représentation des données*

Puisque le réseau modulaire décompose la tâche complexe en plusieurs tâches simples , la représentation sera plus simple.

• *Le hardware*

La structure modulaire du réseau est caractérisée par l'utilisation d'un nombre faible de neurones, d'où l'avantage d'augmenter le taux d'intégration de ces réseaux dans des circuits intégrés.

b). L'algorithme d'apprentissage du réseau modulaire

L'apprentissage des modules du réseau modulaire et le réseau de coordination se font simultanément en suivant les étapes suivantes:

- Choisir le vecteur d'entrée X.
- Le choix du module se fait sous forme de probabilité d'avoir le module i sachant l'entrée X $P(i / X)$.
- La sortie du module i représente la moyenne conditionnelle de la réponse désirée sachant l'entrée et le module.

Soient X l'entrée du réseau , y_i la sortie du i^{eme} module et d la sortie désirée.

Nous avons

$$y_i = \mu_i \quad , \quad i = 1,2,...,k$$

où

$$\mu_i = E(d / X,i) = F_i(X) \tag{1.17}$$

donc la sortie désirée du module i est

$$d = F_i(X) + \varepsilon_i \tag{1.18}$$

ε_i est supposé un vecteur obéissant à une loi gaussienne avec une moyenne nulle et une matrice de covariance identité.

On suppose que ε_i est le même pour chaque module. Soit Λ_i la matrice de covariance de ε_i , alors

$$\Lambda_i = I \qquad i = 1,2,...,k$$

Nous avons la distribution gaussienne multivariable de la réponse désirée d donnée par :

$$f(d / X,i) = \frac{1}{(2\pi \det \Lambda_i)^{\frac{q}{2}}} \exp\left(-\frac{1}{2}(d-y_i)^T \Lambda_i^{-1}(d-y_i)\right) \qquad (1.19)$$

$$f(d / X,i) = \frac{1}{(2\pi)^{\frac{q}{2}}} \exp\left(-\frac{1}{2}\|d-y_i\|^2\right) \qquad (1.20)$$

$$i = 1,2,\ldots,k$$

où q est la dimension de la sortie.

L'équation (1.20) représente la densité de probabilité conditionnelle pour que le i^{eme} module produise la réponse désirée d, sachant le vecteur d'entrée X.

En se basant sur la figure (1.7), nous avons la probabilité d'avoir la réponse désirée d sachant l'entrée X,

$$f(d / X) = \sum_{i=1}^{k} g_i f(d / X,i)$$

$$= \frac{1}{(2\pi)^{\frac{q}{2}}} \sum_{i=1}^{k} g_i \exp\left(-\frac{1}{2}\|d-y_i\|^2\right) \qquad (1.21)$$

Soient $W = [W_1, W_2, \ldots, W_k]^T$ les vecteurs poids de chaque module et $g = [g_1, g_2, \ldots, g_k]^T$ les activations des neurones de sortie du réseau de coordination.

Le principe de l'algorithme est de déterminer les poids d'un module pour maximiser la densité de probabilité $f(d/X)$. On considère cette fonction comme une fonction de vraisemblance où le vecteur des poids synaptiques W et le vecteur d'activation g jouent le rôle des paramètres à identifier. Pour notre cas, il est préférable de travailler avec le logarithme naturel de $f(d/X)$[1].

On définit la fonction de vraisemblance comme suit:

$$L(W,g) = \ln f(d / X) \qquad (1.22)$$

[1] cela est possible car le logarithme est une fonction croissante et monotone.

En utilisant (1.21) et (1.22), on aura: *(on ignore $ln(2\pi)^{q/2}$ car elle est constante)*

$$L(W,g) = \ln \sum_{i=1}^{k} g_i \exp\left(\frac{-1}{2}\|d - y_i\|^2\right) \tag{1.23}$$

donc *L(W,g)* représente la fonction à maximiser en estimant les paramètres W,g.

On considère les sorties du réseau de coordination comme étant des probabilités conditionnelles pour le choix du module pour une entrée/sortie données. Donc les g_i doivent satisfaire les conditions suivantes:

$$\begin{cases} 0 \le g_i \le 1 \\ \sum_{i=1}^{k} g_i = 1 \end{cases} \tag{1.24}$$

Pour satisfaire ces deux conditions, on peut définir l'activation g_i du i^{eme} neurone comme suit:

$$g_i = \frac{\exp(u_i)}{\sum_{j=1}^{k} \exp(u_j)} \tag{1.25}$$

où u_i est la somme pondérée de l'entrée du i^{eme} neurone du réseau de coordination (fig.1.9).

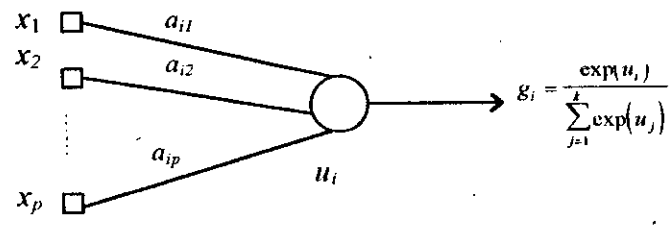


Figure 1.9 Un neurone du réseau de coordination

Avant de développer l'algorithme d'apprentissage, on définit la probabilité postérieure.

$$h_i = \frac{g_i \exp\left(\frac{-1}{2}\|d - y_i\|^2\right)}{\sum_{j=1}^{k} g_i \exp\left(\frac{-1}{2}\|d - y_i\|\right)} \tag{1.26}$$

Il faut noter que h_i satisfait (1.24).

b.1. L'apprentissage des modules (réseaux experts)

Supposons que chaque réseau expert est représenté par la figure (1.10).

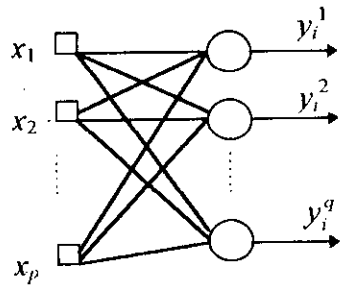


Figure 1.10. Un réseau expert

On suppose que la dimension du vecteur d'entrée X est p, et celle de la sortie est q.

D'après la figure (1.10), la sortie du neurone m est donnée par :

$$y_i^m = X^T W_i^m \tag{1.27}$$

où $$X = \left[x_1, x_2, \ldots, x_p\right]^T \text{ et } W_i^m = \left[w_{i1}^m, w_{i2}^m, \ldots, w_{ip}^m\right]^T$$

Notre but est de déterminer les poids W_i^m qui maximisent la fonction $L(W,g)$.Ces poids seront calculés récursivement selon l'équation suivante:

$$W_i^m(k+1) = W_i^m(k) + \Delta W_i^m(k) \tag{1.28}$$

où

$$\Delta W_i^m(k) = \eta \frac{\partial L(W,g)}{\partial W_i^m(k)} \tag{1.29}$$

η : le pas d'apprentissage.

Pour le calcul de $\frac{\partial L(W,g)}{\partial W_i^m(k)}$, on a :

$$\frac{\partial L}{\partial W_i^m} = \frac{\partial L}{\partial y_i^m} \frac{\partial y_i^m}{\partial W_i^m} \tag{1.30}$$

Après des simplifications, nous obtenons :

$$\frac{\partial L}{\partial y_i^m} = h_i\left(d^m - y_i^m\right)$$

D'après l'équation (1.27), nous avons :

$$\frac{\partial y_i^m}{\partial W_i^m} = X \tag{1.31}$$

En utilisant les équations (1.28) à (1.31) et en posant $e_i^m = d^m - y_i^m$, nous aurons:

$$W_i^m(k+1) = W_i^m(k) + \eta\ h_i e_i^m(k) X \tag{1.32}$$

b.2. L'apprentissage du réseau de coordination

On considère le réseau de coordination représenté par la figure (1.11). Il possède la même forme qu'un réseau expert, sauf que le nombre de sorties est égal au nombre de réseaux experts, et sa fonction d'activation est différente de la fonction d'activation d'un réseau expert.

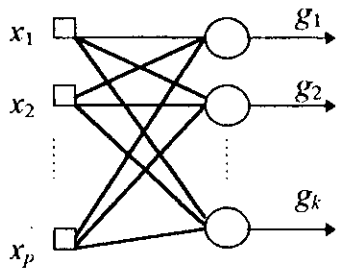


Figure 1.11. Le réseau de coordination

A partir de (1.23) et (1.25) nous obtenons:

$$L(W, gX) = \ln \sum_{i=1}^{k} \exp(u_i) \exp\left(\frac{-1}{2} \left\| d - y_i \right\|^2\right) - \ln \sum_{j=1}^{k} \exp(u_j) \tag{1.33}$$

Soit $a_i = \left[a_{i1}, a_{i2}, \ldots, a_{ip}\right]^T$ le vecteur poids du i^{eme} neurone du réseau de coordination.

D'après la figure (1.11), on a :

$$u_i = X^T a_i \tag{1.34}$$

L'adaptation se fait récursivement selon les équations suivantes :

$$a_i(k+1) = a_i(k) + \Delta a_i(k) \tag{1.35}$$

$$\Delta a_i(k) = \eta \frac{\partial L(W,g)}{\partial a_i(k)} \tag{1.36}$$

où

$$\frac{\partial L}{\partial a_i} = \frac{\partial L}{\partial u_i} \frac{\partial u_i}{\partial a_i} \tag{1.37}$$

Après des simplifications, nous avons :

$$\frac{\partial L}{\partial u_i} = h_i - g_i \tag{1.38}$$

$$\frac{\partial u_i}{\partial a_i} = X \tag{1.39}$$

$$a_i(k+1) = a_i(k) + \eta\left(h_i(k) - g_i(k)\right)X \tag{1.40}$$

b.3. Résumé de l'algorithme d'apprentissage du réseau modulaire

1. Initialiser les poids synaptiques de tout le réseau modulaire.

2. Calculer pour $i = 1,2,\ldots,k$ et $m = 1,2,\ldots,q$

$$u_i(k) = X^T a_i(k)$$

$$g_i(k) = \frac{\exp\left(u_i(k)\right)}{\sum_{j=1}^{k} \exp\left(u_j(k)\right)}$$

$$y_i^m = X^T W_i^m(k)$$

$$y_i(k) = \left[y_i^1, y_i^2, \ldots, y_i^q\right]^T$$

$$h_i(k) = \frac{g_i(n) \exp\left(\frac{-1}{2}\left\|d - y_i\right\|^2\right)}{\sum_{j=1}^{k} g_j \exp\left(\frac{-1}{2}\left\|d - y_i\right\|^2\right)}$$

$$e_i^m(k) = d^m - y_i^m(k)$$
$$W_i^m(k+1) = W_i^m(k) + \eta\, h_i(k) e_i^m(k) X$$
$$a_i(k+1) = a_i(k) + \eta\big(h_i(k) - g_i(k)\big) X$$

3. Répéter l'étape 2 pour toutes les données d'apprentissage.

4. Répéter 2 et 3 jusqu'à la convergence de l'algorithme.

1.5.2. *LES RESEAUX DE NEURONES DYNAMIQUES*

Les réseaux dynamiques, appelés aussi réseaux récurrents sont organisés tel que chaque neurone reçoit sur ses entrées une partie ou la totalité de l'état du réseau (sorties des autres neurones) en plus des informations externes. Pour les réseaux récurrents, l'influence entre les neurones s'exerce dans les deux sens.

L'état global du réseau dépend aussi de ses états précédents. L'équation du neurone dans ce cas est décrite par des équations différentielles ou des équations aux différences. Ce type d'architecture de réseaux est très important, car beaucoup de systèmes que l'on veut modéliser dans la pratique sont des systèmes dynamiques non linéaires (ex: avions, missiles, robots... etc).

Avant de commencer l'étude des réseaux dynamiques, on va montrer comment un réseau MLP peut être utilisé pour le traitement des problèmes dynamiques.

1.5.2.1 *LE RESEAU DE NEURONES AVEC UN TEMPS DE RETARD*

On sait que le facteur temps joue un grand rôle dans le traitement des problèmes dynamiques. Pour cela le réseau MLP utilise des temps de retard à son entrée pour qu'il puisse fonctionner comme un prédicteur non linéaire d'un signal stationnaire [8]. L'architecture illustrée par la figure(1.12) est connue sous le nom de TDNN « *Time Delay Neural Network* » [3]. Ce réseau est capable de modéliser un système dynamique non linéaire dont la sortie est définie comme suit:

$$y(k) = f[u(k), u(k-1), \ldots, u(k-n)].$$

où $f[.]$ est une fonction non-linéaire.

Ce réseau utilise l'algorithme de la rétropropagtion comme algorithme d'apprentissage.

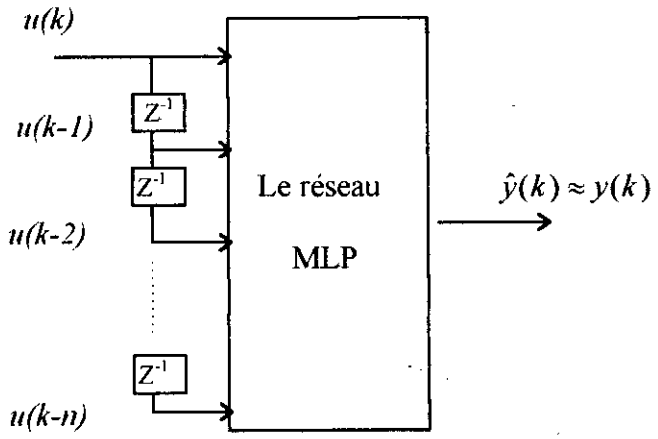


Figure 1.12 Le réseau TDNN

1.5.2.2. *LE RESEAU DE NEURONE AVEC RETOUR DE SORTIE*

Une autre méthode pour présenter la dynamique dans un réseau MLP, c'est de connecter la sortie du réseau avec l'entrée à travers d'autres temps de retard. L'architecture illustrée par la figure (1.13) est donnée par [K.S. Narendra and K.Parthasarathy,1990][1] . Cette architecture est capable de modéliser un système non-linéaire qui est décrit par l'équation suivante:

$$y(k) \quad f[y(k-1), y(k-2),..., y(k-n), u(k), u(k-1),..., u(k-m)]$$

où $f[.]$ est une fonction non-linéaire.

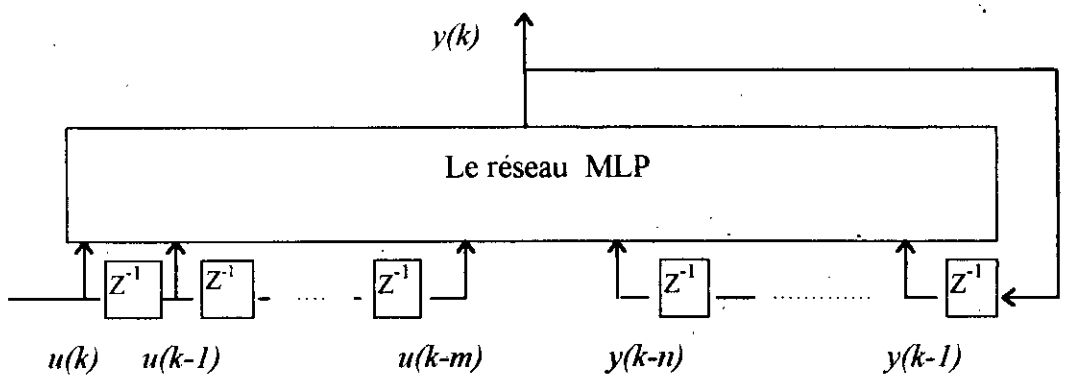


Figure 1.13 Le réseau de neurones dynamique de NARENDRA.

Ce réseau utilise l'algorithme de la rétropropagation comme algorithme d'apprentissage, mais sans utiliser le retour de sortie, car il n'y a aucune garantie que les poids synaptiques vont converger avec la présence de ce retour [1].
Une fois le réseau apprend convenablement la tâche prévue, on peut utiliser le retour de sortie du réseau pour modéliser le système dynamique.

1.5.2.3. *LE RESEAU DE HOPFIELD*

Il faut constater que contrairement aux réseaux statiques, le cerveau humain fonctionne apparemment comme un système dynamique, qui n'atteint pas instantanément un état d'équilibre lorsqu'il est soumis à un stimulant extérieur. C'est justement cette propriété qui a inspiré J.HOPFIELD, de *« California Institut of Technology USA »* dès 1982 [8]. Il a analysé un modèle de réseau dans lequel chaque neurone reçoit des informations de tous les autres neurones du réseau et envoie lui-même des signaux à tous les autres neurones. Il s'agit donc d'un système « coopératif » dont la décision est prise par étapes successives. La caractéristique essentielle de son comportement réside dans l'existence d'états stables dits « attracteurs ».

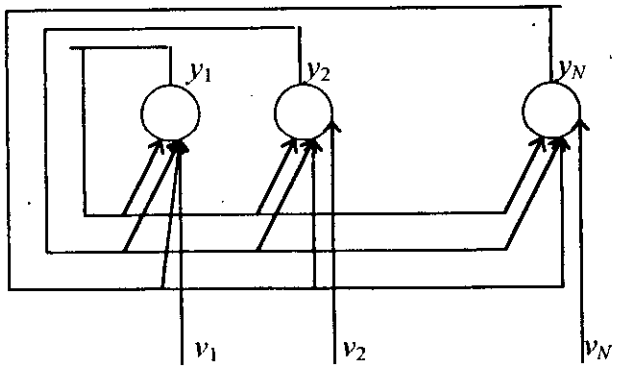


Figure 1.14 Le réseau de HOPFIELD

Le réseau de HOPFIELD est probablement le réseau dynamique le plus connu. Il est sous forme d'une seule couche de neurones entièrement interconnectés.

L'équation d'un neurone est donnée par:

$$\begin{cases} \tau_i \dot{u}_i(t) = -u_i(t) + \sum_{j=1}^{N} w_{ij} y_i(t) + v_i \\ y_i(t) = f[u_i(t)] \end{cases} \quad (1.41)$$

où $u_i(t)$: l'état interne du i^{eme} neurone, $f(.)$: fonction sigmoïde, $y_i(t)$: la sortie du i^{eme} neurone, N : le nombre de neurones, v_i : l'entrée extérieure, τ_i : une constante de temps.

Ce réseau peut être vu comme un système non-linéaire dynamique avec le vecteur v comme entrée, $u(t)$ comme état et $y(t)$ comme sortie. Il peut produire plusieurs comportements suivant les valeurs de ses paramètres. Il peut fonctionner comme un système stable, oscillatoire, ou un système chaotique.

Dans la plupart des cas, le réseau de HOPFIELD est utilisé comme un système stable avec plusieurs points d'équilibre asymptotiquement stables. Pour n'importe qu'elle condition initiale $u(0)$, le réseau converge vers un point d'équilibre. Le nombre exact des points d'équilibre est fixé par W,v,et $f(.)$. Par conséquent, il faut bien choisir ces paramètres pour que ces points correspondent à des solutions du problème.

1.5.2.4. *LE MODELE SPATIO-TEMPOREL D'UN NEURONE*

Dans tous les réseaux de neurones cités, un modèle non-linéaire du neurone est utilisé. Une limitation de ce modèle est qu'il représente seulement le comportement spatial du neurone. Pour qu'il représente aussi un comportement temporel, on modélise chaque poids synaptique par un filtre linéaire invariant dans le temps [8] (fig.1.15).

La réponse du synapse i (i = 1,..,p) à l'entrée $x_i(t)$ est donnée par la convolution suivante :

$$h_{ji}(t) * x_i(t) = \int_{-\infty}^{t} h_{ji}(\lambda) x_i(t-\lambda) d\lambda \quad (1.42)$$

D'après la figure (1.15), l'état du neurone j est :

$$v_j(t) = u_j(t) - \theta_j$$

$$= \sum_{i=1}^{p} h_{ji}(t) * x_i(t) - \theta_j$$

$$v_j(t) = \sum_{i=1}^{p} \int_{-\infty}^{t} h_{ji}(\lambda) x_i(t-\lambda)\, d\lambda \quad - \theta_j \tag{1.43}$$

La sortie du neurone sera :

$$y_j(t) = f\left(v_j(t)\right) = \frac{1}{1+\exp\left(-v_j(t)\right)} \tag{1.44}$$

Les équations (1.43) et (1.44) représentent le comportement spatio-temporel d'un neurone artificiel.

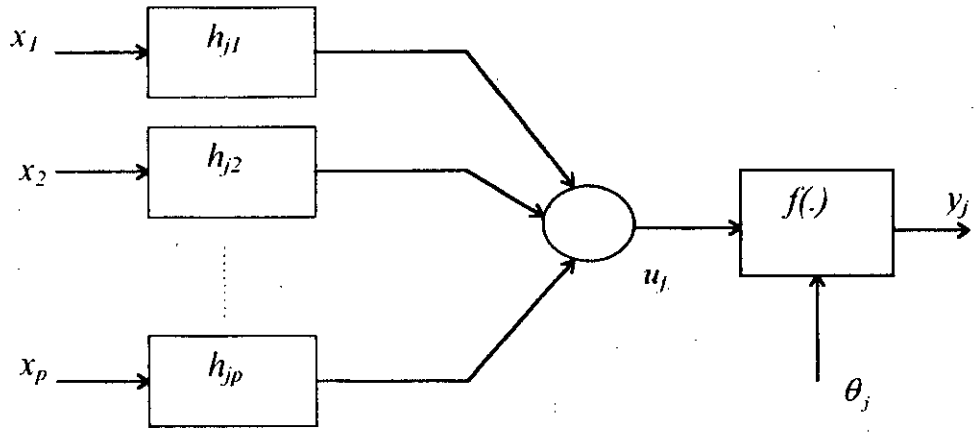


Figure 1.15 Modèle dynamique d'un neurone.

Propriétés

- le filtre synaptique est causal
- le filtre synaptique possède une mémoire finie

  $h_{ji}(t) = 0 \quad , t > T.$

1.5.2.5. *LE RESEAU MLP AVEC LE MODELE FIR«Finite-duration Impulse Response »*

Considérons un réseau MLP dont les neurones cachés sont basés sur le modèle de la figure(1.15). Ce réseau est appelé FIR-MLP.

La sortie de chaque synapse est :

$$s_{ji}(k) = \sum_{L=0}^{M} w_{ji}(L) x_i(k-L) \tag{1.45}$$

Si on pose

$$X_i(k) = \left[x_i(k), x_i(k-1), \ldots, x_i(k-M)\right]^{\mathrm{T}}$$

$$W_{ji} = \left[w_{ji}(0), \ldots, w_{ji}(M)\right]^{\mathrm{T}}$$

$$s_{ji}(k) = W^{T}{}_{ji}(k) X_i(k) \quad ; \quad i \quad 1,2,\ldots,p \tag{1.46}$$

la sortie du neurone j sera :

$$\begin{cases} v_j(k) = \sum_{i=1}^{p} s_{ji}(k) - \theta_j = \sum_{i=1}^{p} w_{ji}{}^{T} X_i(k) - \theta_j \\ y_j(k) = f(v_j(k)) \end{cases} \tag{1.47}$$

L'algorithme d'apprentissage du réseau FIR-MLP

Cet algorithme est basé sur l'apprentissage supervisé.

Soit l'erreur instantanée :

$$e_j(k) = d_j(k) - y_j(k) \tag{1.48}$$

$$\zeta(k) = \sum_{k} e_j^2(k) \tag{1.49}$$

Alors notre objectif est de minimiser le critère de performance :

$$\xi_{total} = \sum_{k} \zeta(k) \tag{1.50}$$

Cet algorithme est basé sur la méthode du gradient.

$$\frac{\partial \xi_{total}}{\partial w_{ji}(k)} = \sum \frac{\partial \xi_{total}}{\partial v_j(k)} \frac{\partial v_j(k)}{\partial w_{ji}(k)} \tag{1.51}$$

$$w_{ji}(k+1) = w_{ji}(k) - \eta \frac{\partial \xi_{total}}{\partial v_j(k)} \frac{\partial v_j(k)}{\partial w_{ji}(k)} \tag{1.52}$$

De (1.47) on a : $$\frac{\partial v_j(k)}{\partial w_{ji}(k)} = X_i(k) \tag{1.53}$$

avec X_i est le vecteur d'entrée du neurone j.

On définit le gradient local par l'équation (1.54).

$$\delta_j(k) = -\frac{\partial \xi_{total}}{\partial v_j(k)} \tag{1.54}$$

Des équations (1.52) à (1.54), nous avons :

$$w_{ji}(k+1) = w_{ji}(k) + \eta \delta_j(k) X_i(k) \tag{1.55}$$

Le calcul de $\delta_j(k)$ varie selon la position du neurone j :

1. Neurone j est de sortie

$$\delta_j(k) = -\frac{\partial \zeta(k)}{\partial v_j(k)}$$

$$= e_j(k) f'\left(v_j(k)\right) \tag{1.56}$$

où $$f'\left(v_j(k)\right) = \frac{\partial y_j(k)}{\partial v_j(k)}$$

2. Neurone j est caché

On définit $\mathcal{A}$ le domaine des neurones dont les entrées sont alimentés par le neurone j, soit $v_m(k)$ l'état interne du neurone m appartenant au domaine $\mathcal{A}$ alors:

$$\delta_j(k) = -\frac{\partial \xi_{total}}{\partial v_j(k)} = -\sum_{m \in A} \sum_{K} \frac{\partial \xi_{total}}{\partial v_m(k)} \frac{\partial v_m(k)}{\partial v_j(k)} \tag{1.57}$$

en utilisant (1.54), nous avons :

$$\delta_j(k) = \sum_{m \in A} \sum_{K} \delta_m(k) \frac{\partial v_m(k)}{\partial v_j(k)}$$

$$\delta_j(k) = \sum_{m \in A} \sum_{K} \delta_m(k) \frac{\partial v_m(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)}$$

$$\delta_j(k) = f'\left(v_j(k)\right) \sum_{m \in A} \sum_{K} \delta_m(k) \frac{\partial v_m(k)}{\partial y_j(k)} \tag{1.58}$$

Puisque $v_m(k)$ est l'activation potentielle du neurone m qui est alimenté par le neurone j alors:

$$v_m(k) = \sum_{j=0}^{p} \sum_{l=0}^{M} w_{mj}(l) y_j(k-l)$$

$$w_{m0}(l) = \theta_m \quad \text{et} \quad y_0(k-l) = -1$$

$$v_m(k) = \sum_{j=0}^{p} \sum_{l=0}^{M} w_{mj}(k-l) y_j(l)$$

$$\frac{\partial v_m(k)}{\partial y_j(k)} = \begin{cases} w_{mj}(k-l) & 0 \le k-l \le M \\ 0 & \text{ailleurs} \end{cases} \tag{1.59}$$

Des équations (1.58) et (1.59), nous avons :

$$\begin{aligned} \delta_j(k) &= f'\left(v_j(k)\right)\sum_{m\in A}\sum_{l=1}^{M+1}\delta_m(k)w_{mj}(k-l) \\ &= f'\left(v_j(k)\right)\sum_{m\in A}\sum_{l=0}^{M}\delta_m(k+l)w_{mj}(k) \end{aligned} \tag{1.60}$$

Soit le vecteur $\Delta_m(k)$ défini par :

$$\Delta_m(k) = \left[\delta_m(k), \delta_m(k+1), \ldots, \delta_m(k+M)\right]^T \tag{1.61}$$

De (1.60) et (1.61), on tire :

$$\delta_j(k) = f'\left(v_j(k)\right)\sum_{m\in A}\Delta_m^T(k)W_{mj} \tag{1.62}$$

L'équation (1.62) est une évaluation complète de $\delta_j(k)$ d'un neurone caché.

Résumons les étapes de l'algorithme:

$$\begin{cases} 1.\ w_{ji}(k+1) = w_{ji}(k) + \eta\delta_j(k)X_i(k) \\ 2.\ \delta_j(k) = \begin{cases} e_j(k)f'\left(v_j(k)\right) & \text{si le neurone } j \text{ est dans la couche de sortie.} \\ f'\left(v_j(k)\right)\sum\Delta_m^T(k)W_{mj} & \text{si le neurone } j \text{ est dans une couche caché e} \end{cases} \end{cases}$$

Il y a un problème avec cet algorithme. Si on examine attentivement l'équation (1.62),on voit que le calcul de $\delta_j(k)$ est non causal parce qu'il a besoin des valeurs futures de δ et W.

Pour résoudre ce problème, on procède de la façon suivante:

Considérons la couche juste avant la couche de sortie,

$$\delta_j(k-M) = f'\left(v_j(k-M)\right)\sum_{m\in A}\Delta_m^T(k-m)W_{mj} \tag{1.63}$$

où

$$\Delta_m^T(k-M) = \left[\delta_m(k-M), \delta_m(k+1-M), \ldots, \delta_m(k)\right]^T \tag{1.64}$$

l'équation (1.64) est obtenue en remplaçant k par k-M dans (1.62). Puisque $\Delta_m^T(k-M)$ contient seulement les valeurs passées du δ_m, alors on peut facilement calculer $\delta_j(k-M)$.

Le même raisonnement pour les autres couches, nous conduit à l'algorithme suivant.

- Pour un neurone j dans la couche de sortie

$$W_{ji}(k+1) = W_{ji}(k) + \eta \delta_j(k) X_i(k)$$

$$\delta_j(k) = e_j(k) f'\left(v_j(k)\right)$$

- Pour un neurone j dans une couche cachée

$$W_{ji}(k+1) = W_{ji}(k) + \eta \delta_j(k-lM) X_i(k-lM)$$

$$\delta_j(k-lM) = f'\left(v_j(k-lM)\right) \sum \Delta_m^T(k-lM) W_{mj}$$

où M est l'ordre du filtre synaptique, l spécifie la couche cachée en question. $l=1$ correspond à la première couche de la couche de sorties, $l=2$ correspond à la deuxième couche de la couche de sortie et ainsi de suite.

I.6. CHOIX DU PAS D'APPRENTISSAGE

Durant l'apprentissage des réseaux de neurones, nous avons fixé un certain critère de performance à minimiser ou à maximiser. Cependant pour atteindre l'extremum, il faut très bien choisir le pas d'apprentissage η. Pour cela, il existe plusieurs méthodes. Par simplicité, le critère de performance choisi est $J(w(k))$ et notre but est de trouver un pas η_k pour que $J(w(k))$ soit minimal.

I.6.1. *METHODE DU PAS CONSTANT*

Cette méthode consiste à fixer le pas d'apprentissage $\eta_k = \eta_0$ pour toutes les itérations. L'inconvénient de cette méthode est qu'il peut y avoir plusieurs points stationnaires et si le pas est mal choisi, il peut y avoir une divergence.

I.6.2. *METHODE DE LA MINIMISATION SUCCESSIVE DU PAS*

Cette méthode propose un pas d'apprentissage variable pour chaque itération. Son principe est le suivant :

Soit $s > 0$ et $0 < \beta < 1$ et m_k représente un nombre naturel.

Le pas d'apprentissage suit la loi suivante : $\eta_k = \beta^{m_k} s$

Pour chaque itération k, il faut trouver le premier scalaire m_k tel que

$$J(w(k+1)) < J(w(k))$$

En d'autre mots pour trouver η_k on commence tout d'abord par $\eta_k = s$ si $J(w(k+1)) \geq J(w(k))$, donc réduire successivement le pas par la multiplication de β jusqu'à atteindre $\eta_k = \beta^{m_k} s$ qui satisfait la condition $J(w(k+1)) < J(w(k))$.

Cette méthode est plus précise que la précédente mais il y a des cas où elle tombe dans des minima locaux. Elle peut aussi engendrer des oscillations autour des points non stationnaires sans converger vers le minimum global [21].

I.6.3. *LA METHODE D'ARMIJO*

Le but de cette méthode est décroitre $J(w(k))$ toujours en évitant tout point stationnaire, c'est-à-dire $[J(w(k+1)-J(w(k))]$ ne doit pas tendre vers zéro si $w(k)$ tend vers un point non stationnaire [21].Le principe de cette méthode est le suivant :

Soit s, β, σ des scalaires tel que : $s > 0, 0 < \beta < 1, 0 < \sigma < \frac{1}{2}$.

Le pas d'apprentissage suit la loi suivante : $\eta_k = \beta^{m_k} s$

avec m_k le premier nombre naturel m pour que :

$$J(w(k)) - J(w(k+1)) \geq -\sigma \beta^m s \left\| \nabla J(w(k)) \right\|^2.$$

I.7. CONCLUSION

Les réseaux de neurones artificiels deviennent un nouveau moyen de traitement de l'information, bien que la plupart de ces réseaux sont souvent exploités à travers des simulations sur des ordinateurs, qui ne sont que des outils d'analyse.

Le principe de traitement dans les réseaux de neurones est tout à fait différent du traitement algorithmique des calculateurs classiques. Un réseau de neurones fonctionne sans programme, n'exécute pas d'instructions de façon séquentiel et ne possède pas une mémoire pour y stocker les codes d'instructions ou les données. Il doit être considéré comme un « Processeur Parallèle » et non pas comme une nouvelle technique de programmation.

A cause de leur complexité, les simulations de réseau sont des méthodes inefficaces d'implantation. L'aspect hardware doit être développé pour que l'implantation soit efficace.

Dans la première partie de ce chapitre nous avons présenté le principe de base d'un réseau de neurone. Rappelons qu'un réseau de neurones n'est rien qu'un approximateur de fonctions basé sur l'interconnexion de plusieurs entités élémentaires appelées neurones. Nous avons présenté deux types de réseaux: les réseaux statiques (réseau MLP et le réseau modulaire) et les réseaux dynamiques (le réseau de hopfield et FIR-MLP).

Après cette étude on remarque que les réseaux de neurones possèdent plusieurs propriétés qui leurs permettent d'être des candidats naturels lors de l'identification et de la commande des systèmes. Toutes les informations du système peuvent être stockées dans un réseau, et ceci grâce aux propriétés suivantes: la nonlinearité, le parallélisme, l'implantation hardware, l'apprentissage et la généralisation, et ils sont naturellement multivariables donc directement applicables au systèmes MIMO « *Multi-input Multi-output* ». Ces propriétés rendent les réseaux de neurones souhaitable en identification et en commande des systèmes non linéaires.

CHAPITRE II

IDENTIFICATION DES SYSTEMES PAR LES RESEAUX DE NEURONES

II.1.INTRODUCTION

Dans la pratique, on est toujours amené à approcher la dynamique du processus à commander par un modèle paramétrique linéaire et stationnaire dans un domaine plus ou moins restreint autour de son point de fonctionnement. Ce type de modèle, désigné dans la littérature par un modèle de représentation ou de commande [29], établit une relation de cause à effet entre les variables auxquelles le fonctionnement est le plus sensible de manière à réaliser un meilleur compromis entre l'erreur de modélisation et la simplicité du système de commande.

Le souci constant d'améliorer les performances des systèmes commandés conduit à des modélisations de plus en plus précises. Mais, si un tel modèle rend compte du comportement d'un système dans une large plage de fonctionnement, il est malheureusement le plus souvent non linéaire. De ce fait il faut utiliser d'autres méthodes qui prennent en compte les caractéristiques des systèmes non linéaires.

Contrairement à l'identification des systèmes linéaires, l'identification des systèmes non lineaires est une tâche difficile, car ces modèles d'identification requièrent un grand nombre de paramètres. On peut citer par exemple le modèle paramétrique basé sur la série de VOLTERRA et le modèle NARMAX « **N**on linear **A**uto-**R**egressive **M**oving **A**verage with e**X**ogenous input ».

A la fin des années **80** la recherche sur les réseaux de neurones artificiels a fait un grand progrès grâce à leur remarquable capacité d'apprentissage. Ils sont utilisés dans plusieurs domaines (ex :traitement d'images, reconnaissance des caractères, approximation des fonctions...).

En **1990** le chercheur **Kumpati.S.Narendra** « *Yale university USA* » a montré que les réseaux de neurones peuvent être utilisés comme un outil dans le domaine de contrôle des systèmes dynamiques [1]. Après cette publication, l'étude des réseaux de neurones dans un contexte dynamique a progressé à l'Université de «*Yale* » où plusieurs modèles mathématiques pour l'identification des systèmes nonlineaires basés sur les réseaux de neurones ont été proposés.

Dans le présent chapitre, nous allons d'abord citer quelques caractéristiques des systèmes non-linéaires. On expose ensuite le principe d'utilisation des réseaux de

neurones pour l'identification des systèmes non-linéaires bruités et non bruités et on termine par présenter des simulations pour valider la théorie, et une conclusion.

II.2. COMPORTEMENTS DES SYSTEMES NON LINEAIRES

Les systèmes non-linéaires sont décrits par des équations différentielles non linéaires. Les non-linéarités peuvent être classées comme naturelles et intentionnelles. Par exemple, des forces de frottement résultent des nonlinéarités naturelles. Par contre, les nonlinéarités artificielles sont induites par le concepteur comme par exemple les lois de commande adaptative. Les systèmes nonlinéaires se caractérisent par :

Points d'équilibres multiples.

Cycles limites: les systèmes nonlinéaires peuvent exhiber des oscillations avec des amplitudes et fréquences fixes sans aucune excitation extérieure appelée cycles limites.

Le chaos: pour les systèmes linéaires stables, de faibles variations dans les conditions initiales ne résultent que par de variations comparables dans les sorties. Pour les systèmes nonlinéaires de telles variations peuvent entraîner un phénomène appelé chaos, c'est-à-dire que la réponse du système est extrêmement sensible aux conditions initiales. Le résultat le plus important du chaos est l'impossibilité de prédire la réponse du système dans ces cas.

II.3.ANALYSE DES SYSTEMES NON LINEAIRES

Cette analyse peut être assez compliquée pour trois raisons principales:

- les systèmes non-linéaires ne peuvent généralement pas être résolus analytiquement.
- les puissantes méthodes mathématiques pour les cas linéaires (transformée de Laplace,

transformée de Fourier,...) ne peuvent être utilisées pour les systèmes non-linéaires.

- pour un système linéaire, une entrée sinusoïdale de n'importe qu'elle amplitude produit une sortie sinusoïdale de même fréquence, alors que pour un système non linéaire la réponse du système peut être sinusoidale,periodique,chaotique ou instable dépendant des conditions initiales et de l'amplitude de l'entrée du système.

Par conséquent, il n'y a pas de méthodes systématiques pour prévoir le comportement des systèmes non-linéaires. Cependant, il existe des méthodes d'analyse et de

conception pour quelques classes de systèmes parmi lesquelles, il y a la théorie de Lyapunov, le critère de Popov et la théorie de l'hyperstabilité.

II.4. MODELISATION DES SYSTEMES

Un système dynamique peut être représenté par plusieurs modèles mathématiques. Ils sont obtenus soit à partir des lois physiques des procédés (modèle de connaissance), soit à partir des suites de mesure des entrées et des sorties (modèle de représentation).

II.4.1. *MODELE DE CONNAISSANCE*

Un modèle de connaissance est un modèle dont la structure a été établie en faisant appel à des modèles plus généraux (lois de la physique, chimie,...).Les paramètres des modèles de connaissance ont alors un sens physique (longueur, résistance électrique, inertie,...). Ce dernier est beaucoup plus riche de signification que le modèle de représentation et contient toutes les informations utiles sur le processus. Il est par contre plus difficile à obtenir.

II.4.2. *MODELE DE REPRESENTATION*

Ce modèle n'a aucun pouvoir explicatif de la structure physique de l'objet. Sa structure n'est qu'une relation mathématique qui relie localement les mesures des différentes variables du processus. Les paramètres n'ont aucun sens physique connu. Ce modèle est de type « boîte noire ». Quoique non indicatif, il est cependant suffisant dans les problèmes de traitement de signal et de commande de processus. Il est d'une utilisation très fréquente.

La première étape de la modélisation consiste à émettre des hypothèses sur la structure du modèle, c'est-à-dire, choisir un type de relation mathématique liant les entrées et les sorties du système.

II.4.3 *LA REPRESENTATION (I S O) (INPUT - STATE - OUTPUT) DES SYSTEMES*

Les systèmes dynamiques sont souvent représentés par des équations différentielles ou aux différences:

$$\begin{cases} \dot{X}(t) = f\big(X(t), u(t), t\big) \\ \dot{Y}(t) = h\big(X(t), u(t), t\big) \end{cases} \tag{2.1}$$

où

$$X(t) = \left[x_1(t), x_2(t), \ldots, x_n(t)\right]^T$$

$$Y(t) = \left[y_1(t), y_2(t), \ldots, y_m(t)\right]^T$$

$$u(t) = \left[u_1(t), u_2(t), \ldots, u_p(t)\right]^T$$

$X(t)$, $Y(t)$, $u(t)$ représentent respectivement l'état, la sortie et la commande du système.

Les fonctions $f(.)$ et $h(.)$ sont des fonctions nonlinéaires.

Le même système peut être représenté par des équations aux différences suivante :

$$\begin{cases} X(k+1) = \phi\big(X(k), u(k), k\big) \\ Y(k) = \psi\big(X(k), u(k), k\big) \end{cases} \tag{2.2}$$

Si ce système est invariant dans le temps et satisfait certaines conditions de continuité autour d'un point d'équilibre stable, alors on peut le linéariser pour obtenir les équations suivantes :

$$\begin{cases} X(k+1) = A\,X(k) + B\,u(k) \\ Y(k) = C\,X(k) + D\,u(k) \end{cases} \tag{2.3}$$

Les paramètres inconnus du modèle mathématique seront déterminés dans l'étape suivante, dite d'identification.

II.5. IDENTIFICATION DES SYSTEMES

Quand les fonctions ϕ et ψ dans (2.2) sont inconnues, alors on utilise l'identification pour les approximer.

Soient $u(k)$ et $y_p(k)$ l'entrée et la sortie d'un système dynamique,invariant et causal.Le système est supposé stable avec des paramètres inconnus regroupés dans un vecteur dénoté par θ. L'objectif est de construire un modèle d'identification tel que quand on

applique la même entrée $u(k)$ au système et au modèle, il produit une sortie $\hat{y}(k)$ qui est proche à $y(k)$.

Le problème à résoudre est le suivant :

Compte tenu des informations d'entrée-sortie que l'on peut recueillir sur le processus, nous estimerons le vecteur de paramètres θ du modèle mathématique. Ce principe est illustré par la figure (2.1). Le vecteur est estimé en minimisant un certain critère d'optimalité.

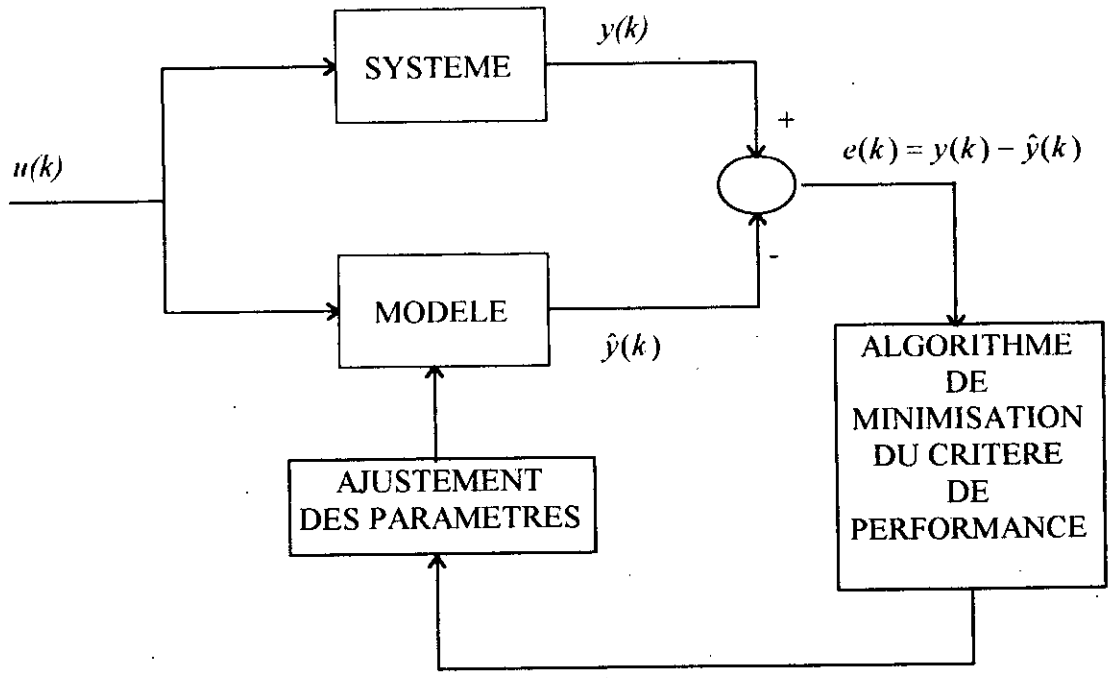


Figure 2.1 Identification des systèmes

II.6. IDENTIFICATION DES SYSTEMES NON LINEAIRES PAR LES RESEAUX DE NEURONES

Le principe d'identification par les réseaux de neurones consiste à remplacer le modèle paramétrique classique par des modèles neuronaux.

Un système dynamique nonlinéaire peut être décrit par l'équation suivante:

$$y(k)=f[y(k-1),y(k-2),\ldots,y(k-n),\ u(k-1),u(k-2),\ldots,u(k-m)] \qquad (2.4)$$

où f [.] est une fonction non-linéaire. L'identification par les réseaux de neurones consiste à trouver un réseau de neurones qui reproduit fidélement la fonction f [.].

En effet, il a été démontré que n'importe quelle fonction $g : R^n \rightarrow R^m$ définit sur un sous ensemble compact de R^n peut être approchée par un réseau de neurones à multicouches avec une seule couche cachée [1],[11].

Pendant l'identification, les poids synaptiques du réseau seront détérminés récursivement pour qu'ils réalisent l'application non linéaire f[.]. Tout d'abord, le système est excité par un signal riche en fréquence et les sorties seront collectées. Ensuite, l'apprentissage supervisé du réseau de neurones est réalisé.

Comme dans l'identification classique, deux structures d'identification peuvent être considérées. Dans la première, on utilise des réseaux de neurones non récurrents (cela correspond au modèle série-parallèle de l'identification classique), dans la deuxième, des réseaux de neurones récurrents (cela correspond au modèle parallèle).

II.6.1. *IDENTIFICATION PAR MODELE NEURONAL RECURRENT (PARALLELE)*

Le modèle neuronal d'identification du système (2.4) est donné par l'équation suivante:

$$\hat{y}(k) = N\left[\hat{y}(k-1), \hat{y}(k-2), \ldots, \hat{y}(k-n), u(k-1), u(k-2), \ldots, u(k-m)\right] \qquad (2.5)$$

où N est réseau de neurones.

La méthode d'identification par le modèle parallèle est basée sur l'utilisation des sorties du modèle lui même pour l'entraînement du réseau de neurones.

Le principe de cette méthode est illustré par la figure (2.2).

Dans cette méthode il y a le problème de la stabilité durant le processus de l'identification, car il n'y a aucune garantie pour que les poids synaptiques convergent vers des valeurs constantes, même dans le cas des modèles mathématiques linéaires. Les conditions de convergence des paramètres du modèle parallèle sont jusqu'à présent inconnues [1].

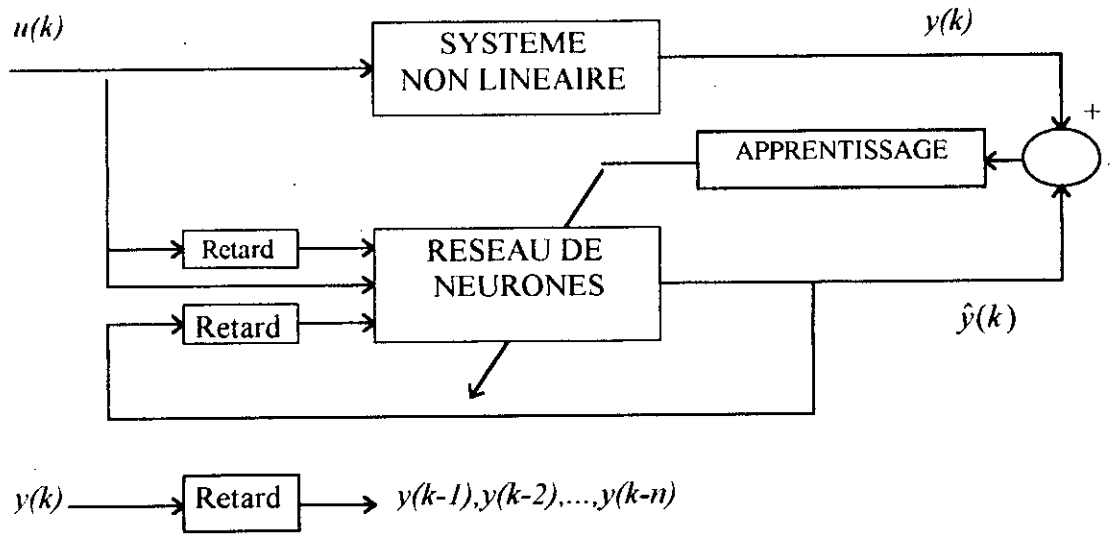


Figure 2.2 Schéma de l'identification en utilisant le modèle parallèle.

II.6.2. *IDENTIFICATION PAR MODELE NEURONAL NON RECURRENT (SERIE-PARALLELE)*

Contrairement au réseau récurrent (parallèle), dans l'identification par un réseau non récurrent (série-parallèle), la sortie du système est réinjectée dans le réseau de neurones à la place de la sortie du réseau.

Dans ce cas, le modèle neuronal sera :

$$\hat{y}(k) = N\left[y(k-1), y(k-2), \ldots, y(k-n), u(k-1), u(k-2), \ldots, u(k-m)\right] \qquad (2.6)$$

La procédure d'identification est illustrée par la figure (2.3). Cette structure sera adoptée durant l'identification des systèmes par la suite.

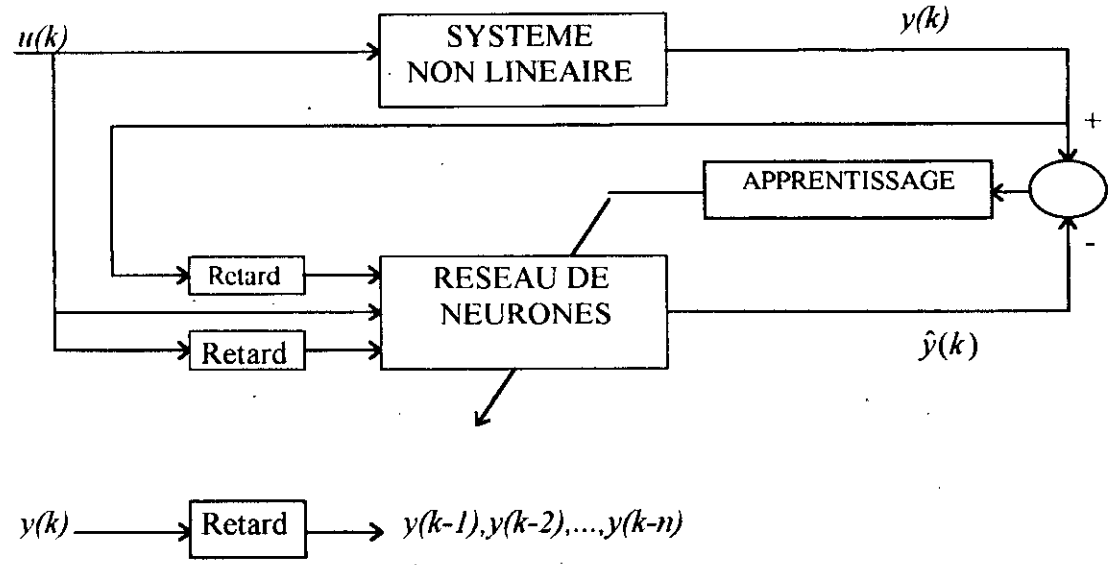


Figure 2.3. Schéma de l'identification en utilisant le modèle (serie-parallel).

II.7. RESULTATS DES SIMULATIONS

Dans ces simulations, on va présenter les résultats de l'identification de plusieurs systèmes en utilisant le réseau modulaire et le réseau à multicouches. On subdivisera la simulation en deux parties. Nous commençons tout d'abord par l'identification des systèmes non linéaires non bruités. Ensuite la robustesse des réseaux de neurones dans l'identification d'un système non-linéaire affecté par des perturbations avec plusieurs valeurs du rapport signal / bruit, sera testée.

Dans chaque cas, le réseau de neurones est supposé contenir suffisamment de paramètres (poids synaptiques), afin d'être apte à représenter fidèlement les caractéristiques d'entrée-sortie du système correspondant. Dans ce cas, la fonction non linéaire dans l'équation aux différences décrivant le système peut être remplacée par un réseau de neurones. En plus, on supposera qu'une solution théorique pour le problème de l'identification existe.

Pour une discussion simple, on notera une classe de fonctions générées par un réseau de neurones multicouches avec L couches et n_l neurones dans la couche l ($l: 1..L$) par le symbole $\eta^{L}_{n_1,n_2,...,n_L}$.

Durant toutes ces simulations, on a utilisé une entrée aléatoire appartenant à l'intervalle [-1,+1].

II.7.1. *IDENTIFICATION DES SYSTEMES NON BRUITES*

a). Identification d'un système monovariable

Le système (SISO) à identifier est représenté par l'équation aux différences du second-ordre:

$$y(k+1) = f\big(y(k), y(k-1)\big) + u(k) \tag{2.7}$$

où $$f\big(y(k), y(k-1)\big) = \frac{y(k)y(k-1)\big[y(k)+2{,}5\big]}{1+y^2(k)+y^2(k-1)}$$

Le modèle d'identification sera :

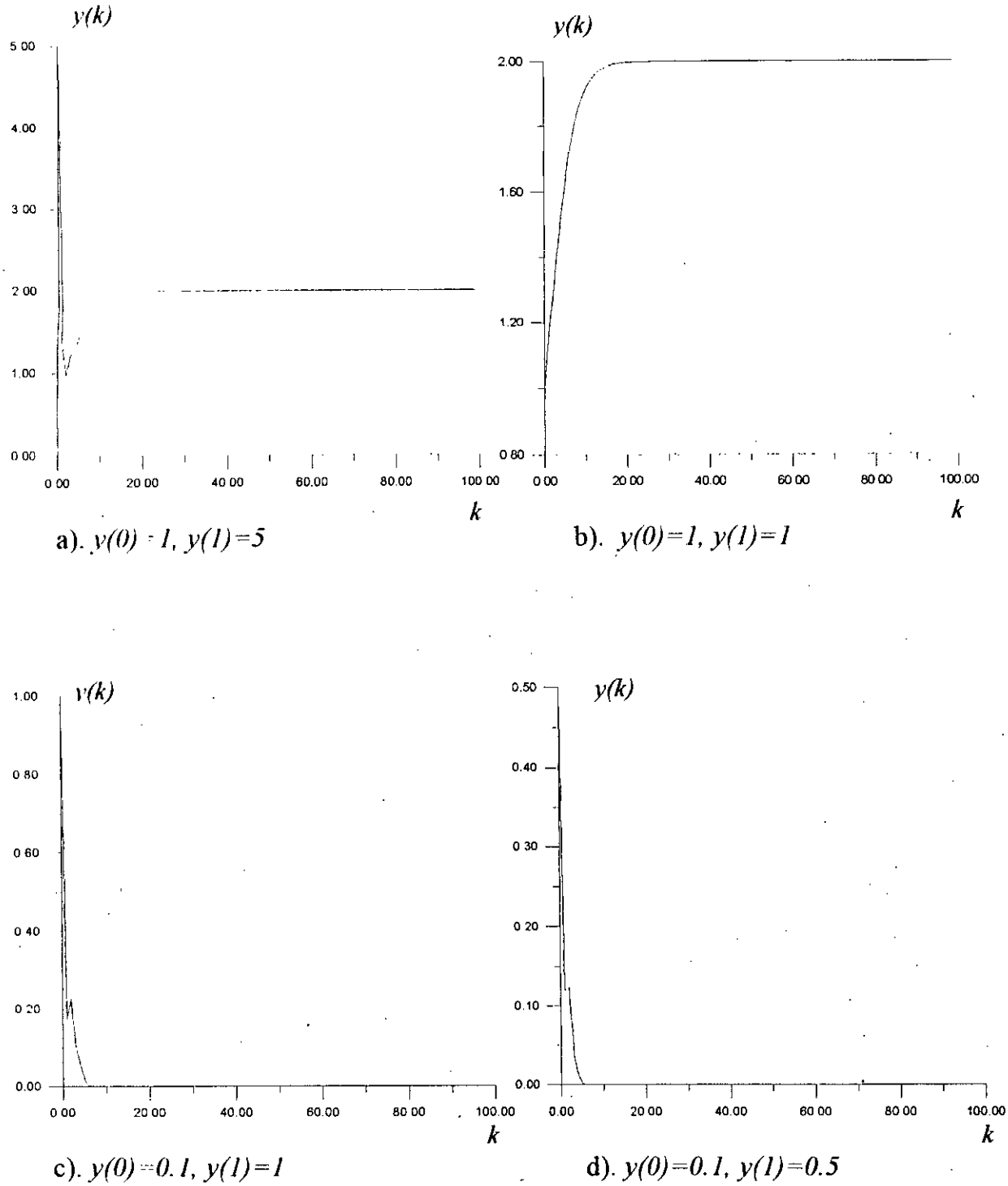
$$\hat{y}(k+1) = N\big(y(k), y(k-1)\big) + u(k) \tag{2.8}$$

avec N représente un réseau de neurone.

Nous avons choisit le réseau de neurones modulaire pour l'identification de ce système. Après plusieurs essais sur le nombre de modules, on a remarqué qu'un réseau de cinq modules est capable d'identifier le système.

Avant de commencer l'identification du système, il faut avoir des informations à priori concernant son comportement tels que le nombre de points d'équilibre du système libre avec ses propriétés de stabilité et le domaine Ω ($u \in \Omega$) produisant une sortie bornée.D'après la figure (2.4), les états d'équilibre du système libre sont les séquences {0} et {2}. En plus, comme l'indique la figure (2.5) on remarque que la sortie du système est bornée avec les conditions initiales {0} et {2}. L'adaptation des poids synaptiques du réseau modulaire est réalisée au bout de 10000 itérations.La figure (2.6) montre le résultat du test de validité du réseau modulaire obtenu en superposant la sortie du système et celle du modèle neuronal obtenu après la phase apprentissage.

D'après la figure (2.6), on remarque que les deux sorties sont pratiquement superposées. En plus du nombre de neurones qui est faible (11 neurones), l'apprentissage s'est fait rapidement par rapport à un réseau multicouches appartenant à la classe $\eta^{4}_{2,20,10,1}$ qui, pour les mêmes performances a requis 100000 itérations [1].



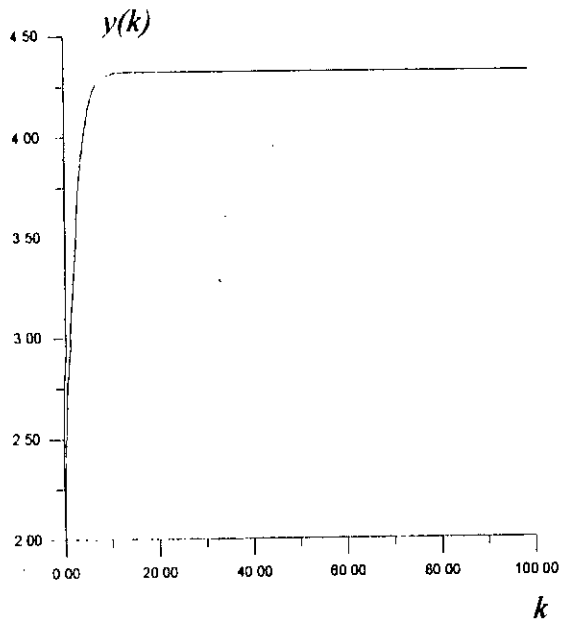
a). $y(0)=1, y(1)=5$

b). $y(0)=1, y(1)=1$

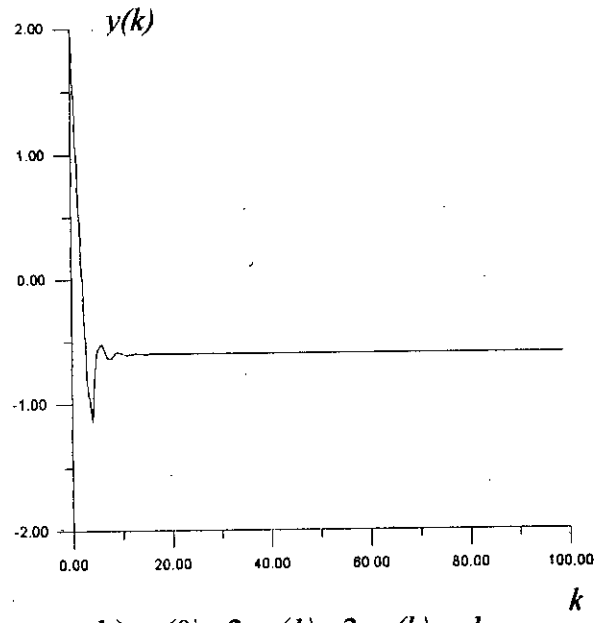
c). $y(0)=0.1, y(1)=1$

d). $y(0)=0.1, y(1)=0.5$

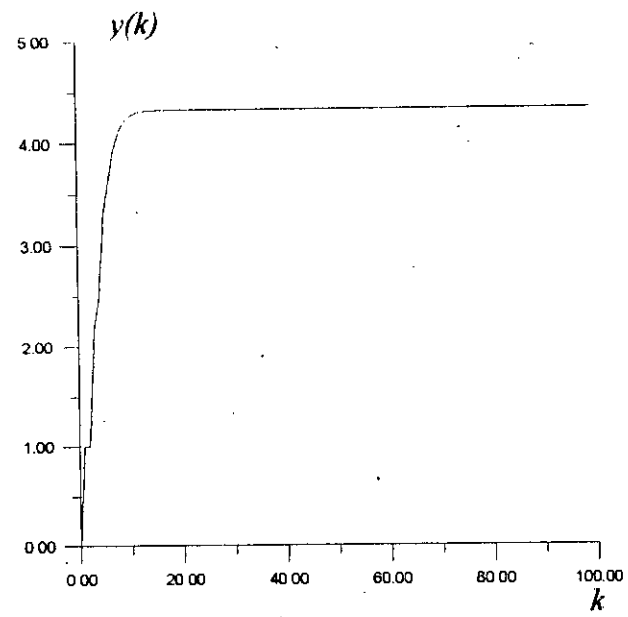
Figure 2.4. Test de stabilité du système autonome ($u(k)=0$).



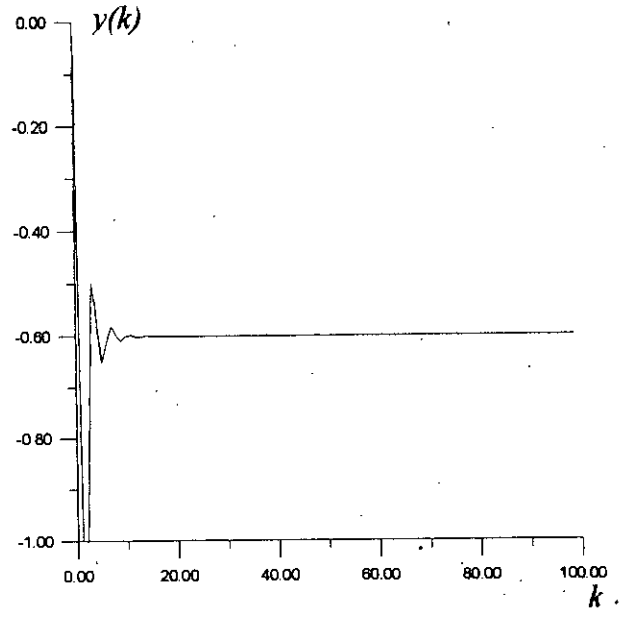
a). $y(0)=1$, $y(1)=5$, $u(k)=1$



b). $y(0)=2$, $y(1)=2$, $u(k)=-1$



c). $y(0)=0$, $y(1)=0$, $u(k)=1$



d). $y(0)=0$, $y(1)=0$, $u(k)=-1$

Figure 2.5 Test de stabilité en présence d'une entrée bornée.

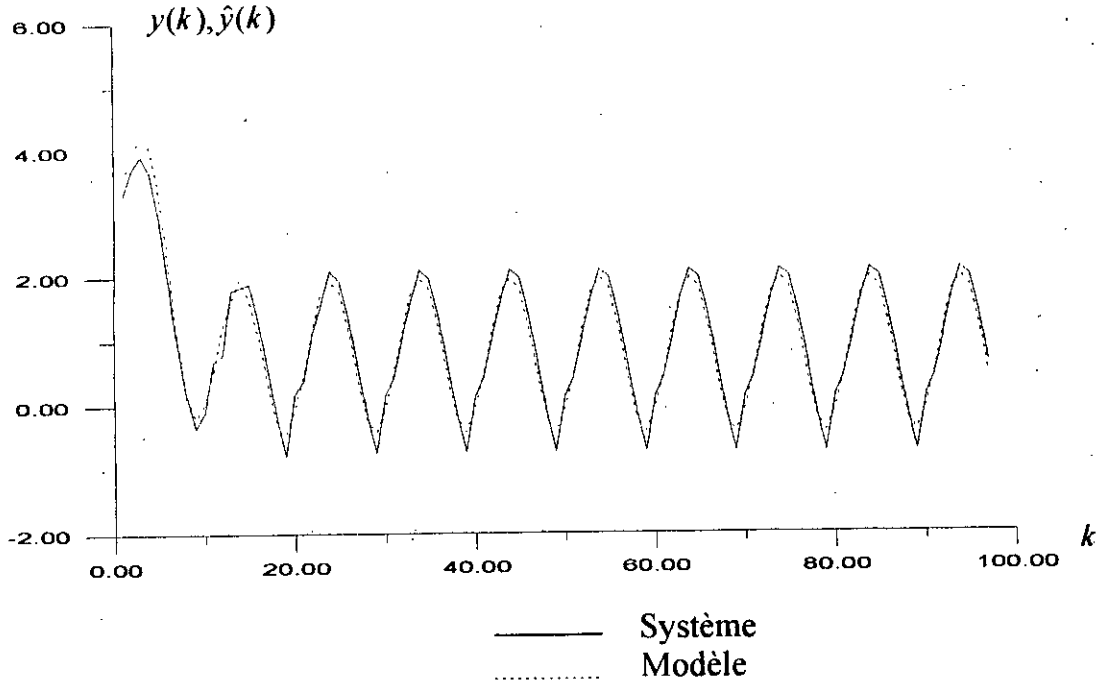


Figure 2.6. Sortie du système et sortie du réseau modulaire pour une entrée :

$$u(k) = \sin(2\pi k / 450)$$

Rapport Signal / Bruit	Pas de bruit
Nombre de données d'apprentissage	100
Le pas d'apprentissage	0.001
Le nombre d'iterations	10000
temps d'apprentissage	1 minutes, 29 secondes, 28 centième
Matériel utilisé	Micro-Ordinateur DX2 66 MHZ

Tableau 2.1. Caractéristiques de la simulation.

b). Identification d'un système multivariable

Dans cette partie, on montre que les réseaux de neurones peuvent être utilisés pour identifier les systèmes multivariables .

Le système à identifier est donné par l'équation aux différences suivante :

$$\begin{cases} x_1(k+1) = \cos(x_2(k)) + u_1(k) \\ x_2(k+1) = \sin(2x_1(k)) + u_2(k) \end{cases} \tag{2.9}$$

Le modèle sera comme suit :

$$\begin{cases} \hat{x}_1(k+1) = N_1(x_1(k), x_2(k)) + u_1(k) \\ \hat{x}_2(k+1) = N_2(x_1(k), x_2(k)) + u_2(k) \end{cases} \tag{2.10}$$

avec N_1 , $N_2 \in \eta^4_{2,10,10,1}$.

D'après la figure (2.7), on remarque que le système autonome possède les valeurs x_1 = 0.59 et x_2 = 0.93 comme point d'équilibre stable. En plus, comme indiqué dans la figure (2.8), on voit que le système est stable pour un vecteur d'entrée dont les éléments sont : $|u_1(k)| < 1$ et $|u_2(k)| < 1$.

Pour un vecteur d'entrée $u(k) = [+1,+1]^T$ le système commence à produire des oscillations puis se stabilise. Dès que les entrées seront supérieures à 1 ($|u_1(k)| > 1$, $|u_2(k)| > 1$) le système devient marginalement stable.

Au vu de ces constatations, ce système sera identifié avec des entrées u_1 et u_2 qui représentent un bruit blanc dans l'intervalle [-1,+1].

L'adaptation est accomplie au bout de 20000 itérations avec un pas d'apprentissage égal à 0.1. Après la phase apprentissage des réseaux multicouches, les résultats sont illustrés par la figure (2.9) . Elle montre les réponses du modèle et du système au vecteur d'entrée $u(k) = \left[\frac{1}{3}(1-\exp(-k)), \exp(-k)\right]^T$. On remarque que les deux réponses sont pratiquement identiques. Ceci montre que la méthode d'identification appliquée aux systèmes monovariables peut être généralisée aux systèmes multivariables.

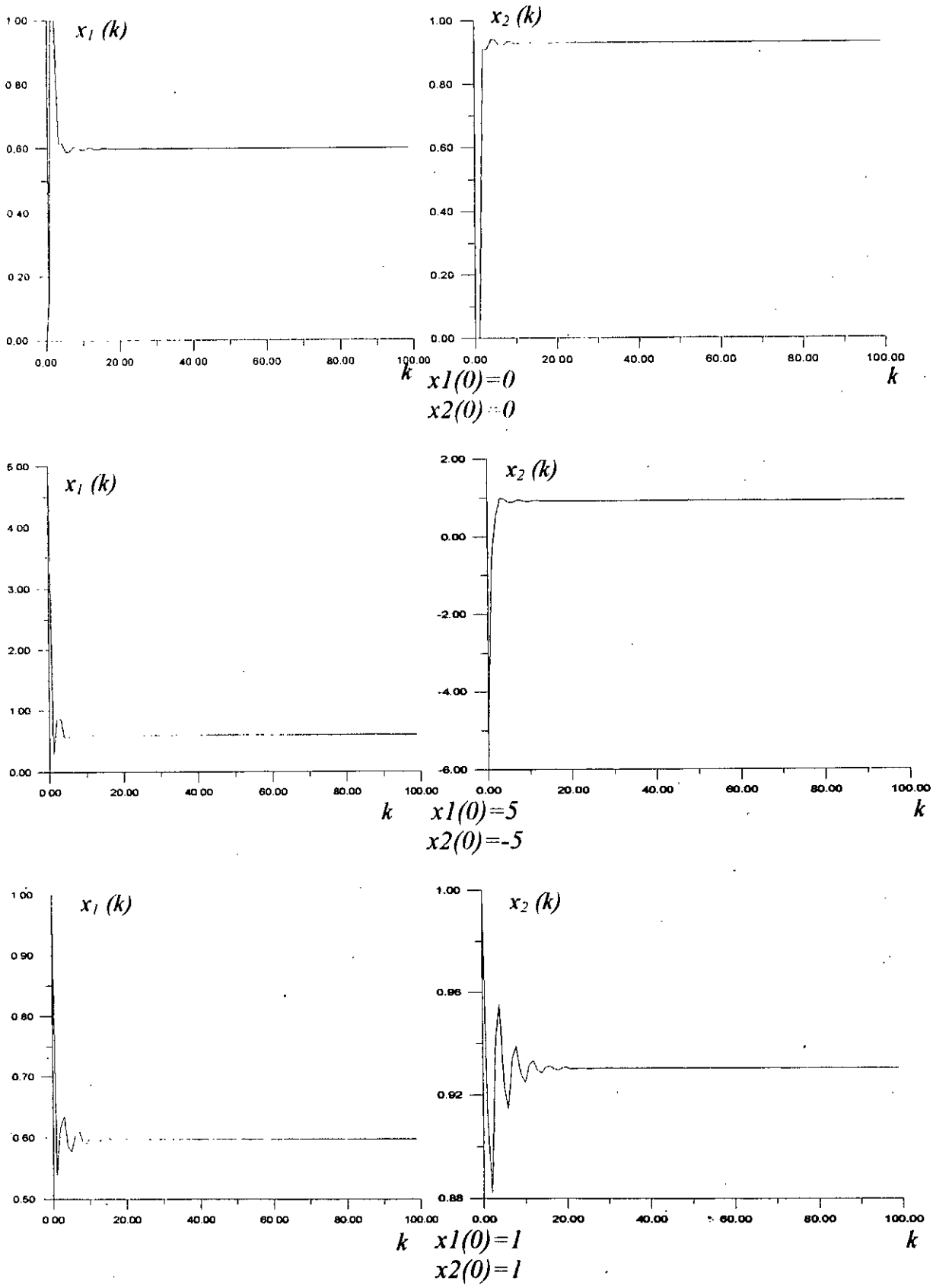


Figure 2.7 Test de stabilité du système autonome.

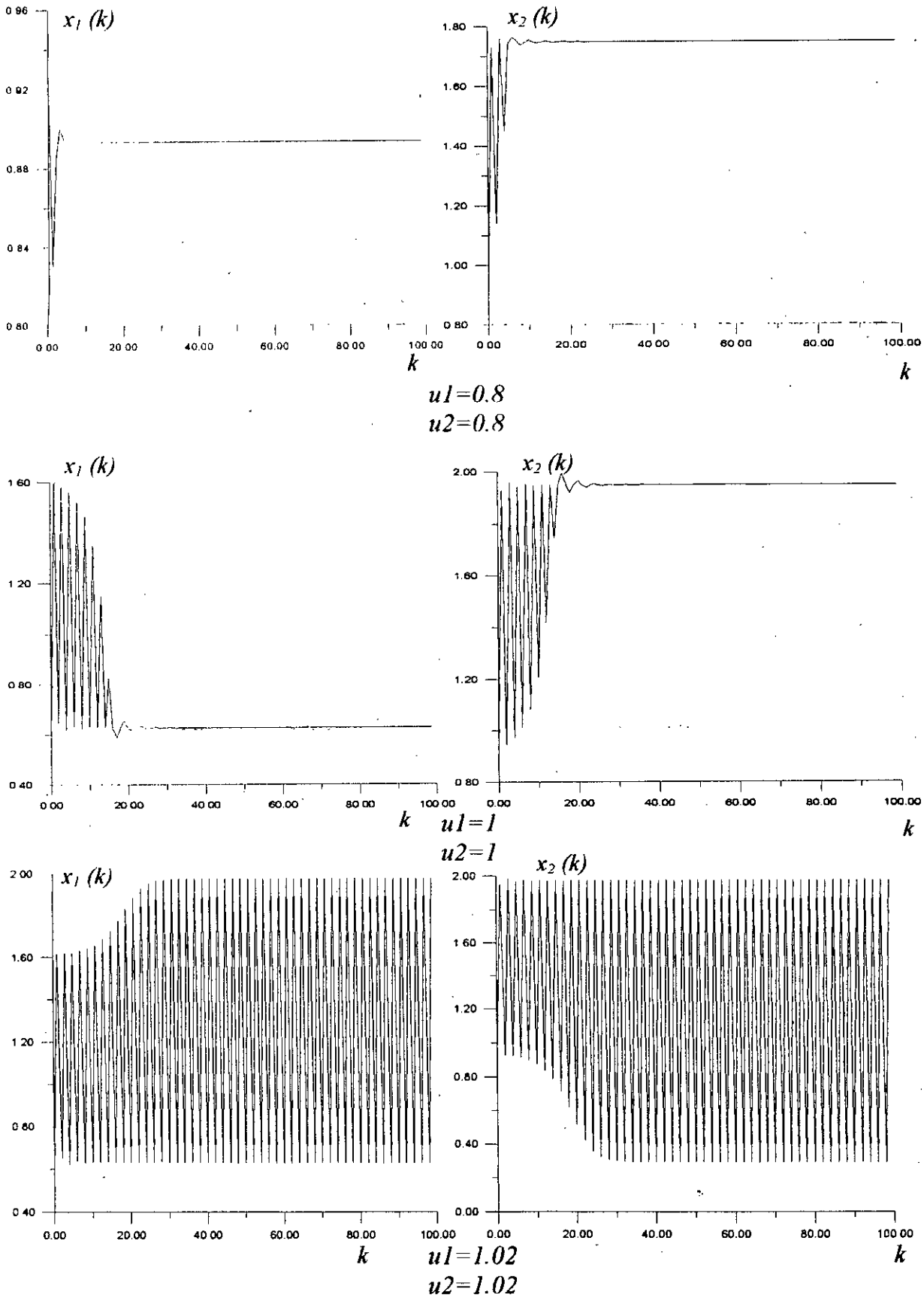


Figure 2.8 Test de stabilité en présence d'une entrée bornée.

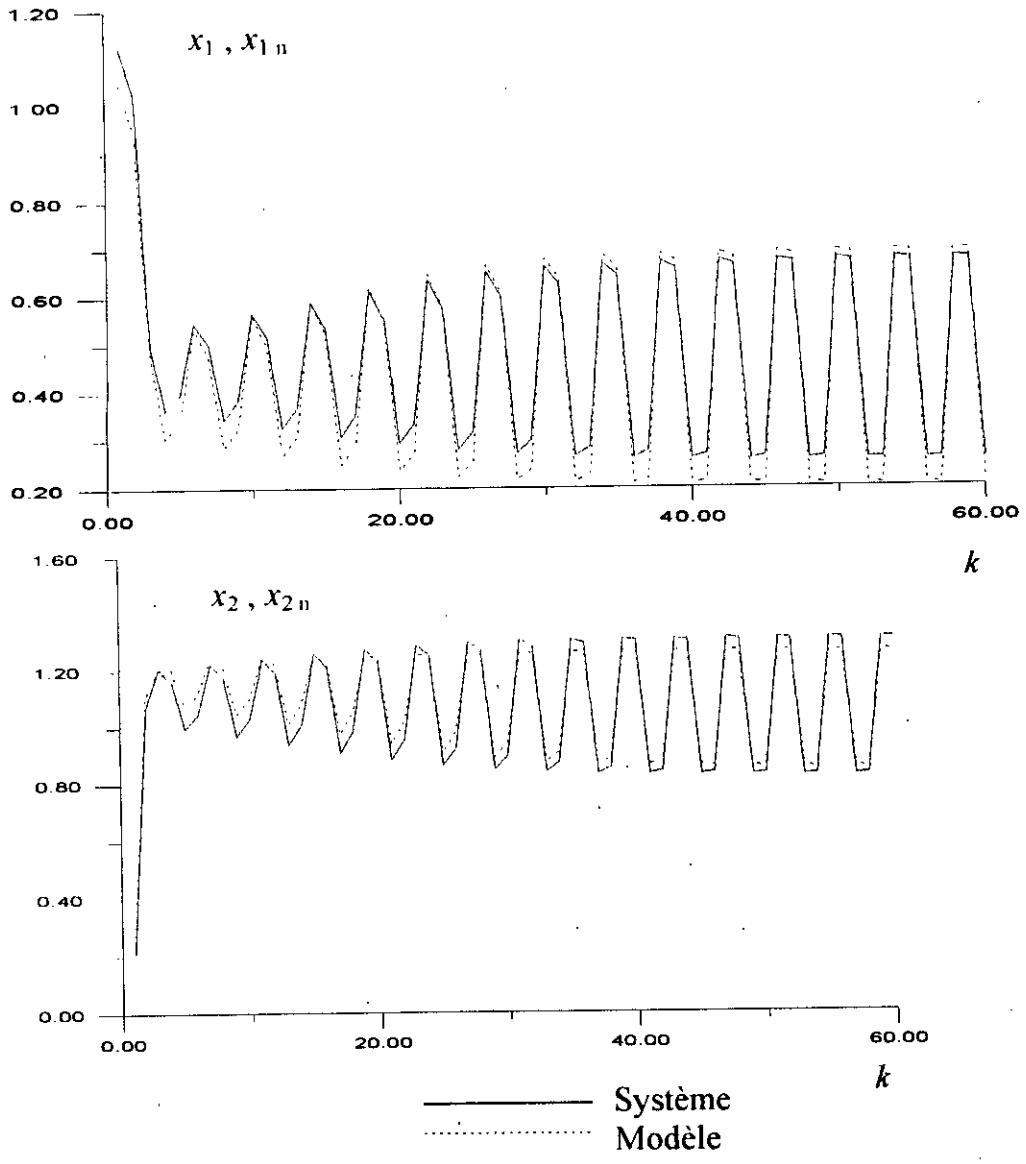


Figure 2.9. Etats du système et sorties des réseaux multicouches .

Rapport Signal / Bruit	Pas de bruit
Nombre de données d'apprentissage	100
Le pas d'apprentissage	0.1
Le nombre d'iterations	20000
temps d'apprentissage	8 minutes,34 secondes,24 centième
Matériel utilisé	Micro-Ordinateur DX2 66 MHZ

Tableau 2.2. Caractéristiques de la simulation.

II.7.2. *IDENTIFICATION DU SYSTEME ET DE SON ENVIRONNEMENT*

Comme il a été mentionné dans l'introduction de ce chapitre, les processus industriels sont trop complexes pour que l'on puisse établir rigoureusement un modèle mathématique de leurs comportements. Par ailleurs, l'influence des perturbations n'est pas toujours facile à estimer et varie éventuellement avec le temps.

Le terme *v(t)* de la figure (2.10) peut englober généralement plusieurs composantes :

- les perturbations mesurables qui représentent l'effet des variables du processus qui ont été jugée secondaires lors de l'élaboration du modèle.
- les perturbations aléatoires qui permettent de tenir compte de l'influence de l'environnement, des bruits de mesure et de quantification, etc.
- les perturbations de charge dont l'amplitude est généralement inconnue et lentement variable dans le temps. Celles-ci peuvent changer brusquement si le point de fonctionnement change.
- une constante non nulle qui permet de prendre en considération que les perturbations ne sont pas nécessairement de moyenne nulle. En effet, dans le cas des processus réels, une commande nulle ne conduit pas généralement à une sortie nulle. Cette constante est appelée *« la composante continue »* du processus [29].

Dans cette partie on étudiera la robustesse des réseaux de neurones dans l'identification des systèmes à sorties bruitées.

Avant d'entamer les simulations, quelques notions sur le bruit seront brièvement exposées.

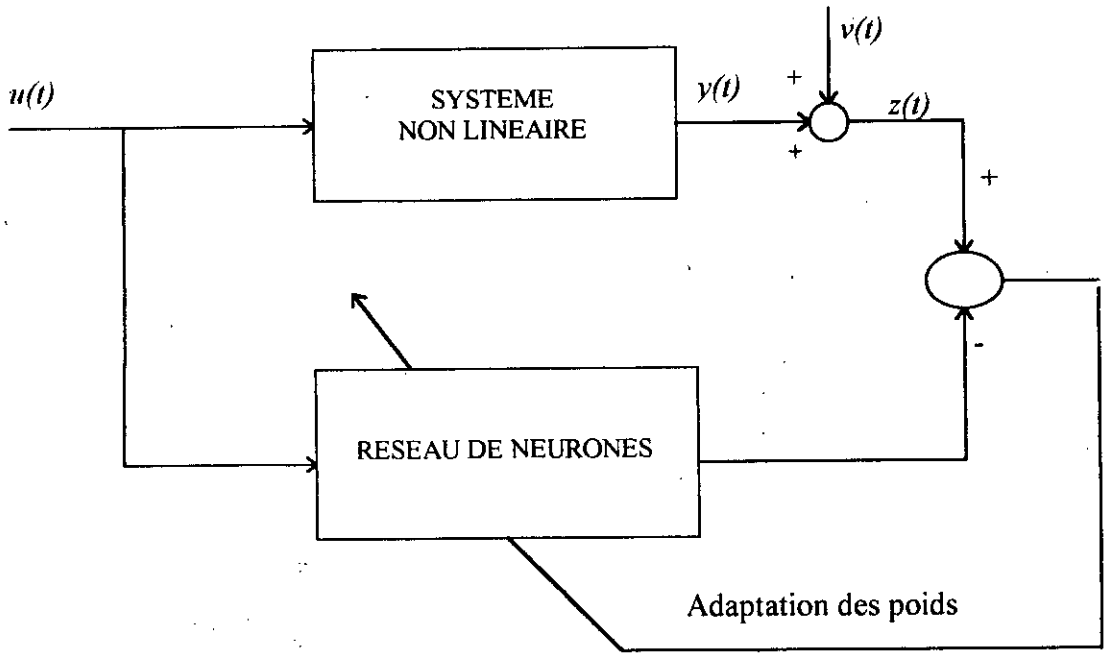


Figure 2.10. Identification d'un système, dont la sortie est entachée par des perturbations.

II.7.2.1. *TOPOLOGIE DU BRUIT*

D'une manière générale, nous considérons le bruit comme étant tout écart entre un signal utile déterministe ou aléatoire transportant une information qui intéresse le destinataire et le signal effectivement reçu. Ceci découle du fait que le bruit est inhérent à l'environnement naturel et à l'équipement électronique caractérisant ainsi le bruit d'origine externe ou interne.

II.7.2.2. *DIFFERENTS TYPES DE BRUITS*

a). BRUIT GAUSSIEN

Un bruit est dit gaussien si sa densité de probabilité suit une distribution gaussienne

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-m)^2/2\sigma^2}$$

où $f(x)$ est la densité de probabilité, m et σ^2 sont respectivement la moyenne et la variance.

b). BRUIT BLANC

Un bruit blanc est un signal aléatoire dont la connaissance à un instant donnée ne fournit aucun renseignement sur sa valeur à l'instant suivant.

Cette propriété se traduit mathématiquement par une fonction d'auto-corrélation équivalente à une impulsion de DIRAC. C'est un signal totalement incorrélé ou encore ayant un spectre de fréquence uniforme et de largeur de bande infinie.

Notons que c'est par analogie avec la lumière blanche qui occupe tout le spectre visible que ce type de bruit a été dénommé.

c). BRUIT COLORE

C'est par analogie avec les autres couleurs de la lumière (présence de la fréquence correspondante à la couleur voulue et absence de toutes les autres fréquences) qu'on a ainsi appelé ce type de bruit. Par exemple, le bruit rose est un bruit qui décroît avec la fréquence de 6 db par octave d'où son abondance dans le bases fréquences.

Ainsi la fonction d'auto-corrélation d'un tel bruit ne sera pas une impulsion de DIRAC mais plutôt une courbe étroite.

II.7.2.3. RESULTATS DES SIMULATIONS

Dans les simulations qui suivent , on va présenter les résultats de l'identification d'un système non linéaire affecté par des perturbations avec plusieurs valeurs de rapport signal/bruit (S/N). Le système à identifier possède l'équation aux différences suivante :

$$y(k) = \sin\left(\cos(y(k))\right) + u(k) \qquad (2.11)$$

Un réseau de neurones multicouches appartenant à la classe $\eta^4_{1,10,10,1}$ sera utilisé . Cette architecture est la même pour tous les cas traités.

D'après la figure (2.11), on remarque que le système autonome est stable avec le point $y=0.69$ comme point d'équilibre stable. De plus, comme sur la figure (2.12) le système est stable pour certaines entrées bornées.

On a testé la robustesse des réseaux de neurones multicouches en introduisant des perturbations à la sortie du système. Le système est affecté par un bruit blanc avec

des rapports S/N de 39 db et de 23 db. Les figures (2.14), (2.15) et (2.16) montrent les résultats de l'identification. Enfin on a utilisé un bruit qui est donné par $v(k) = \sin\left(\frac{2\pi k}{10}\right)$ avec un rapport S/N de 39 db , et l'apprentissage a été effectué après 1000 itérations. Le résultat de test de validité est illustré par la figure (2.18).

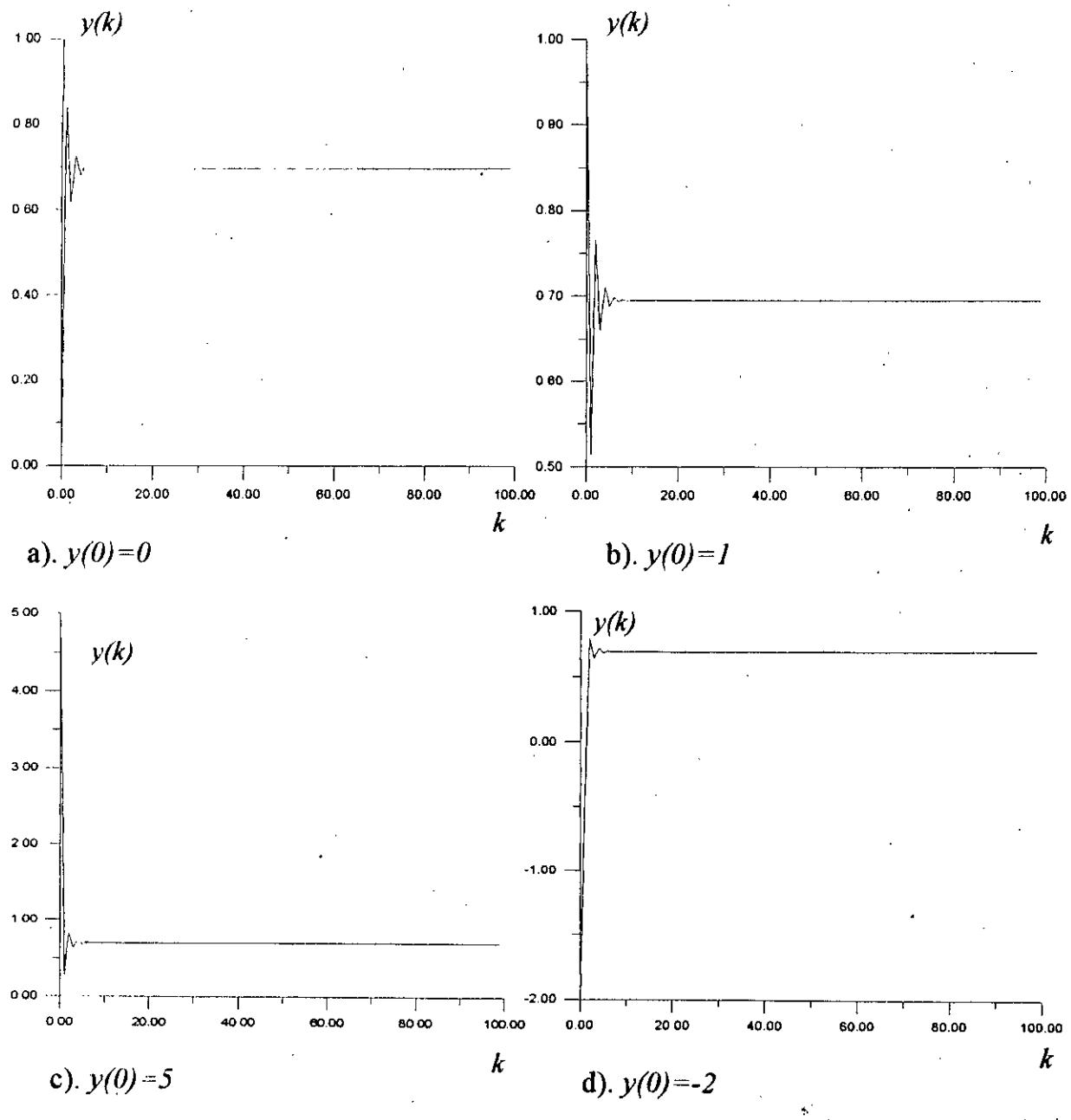
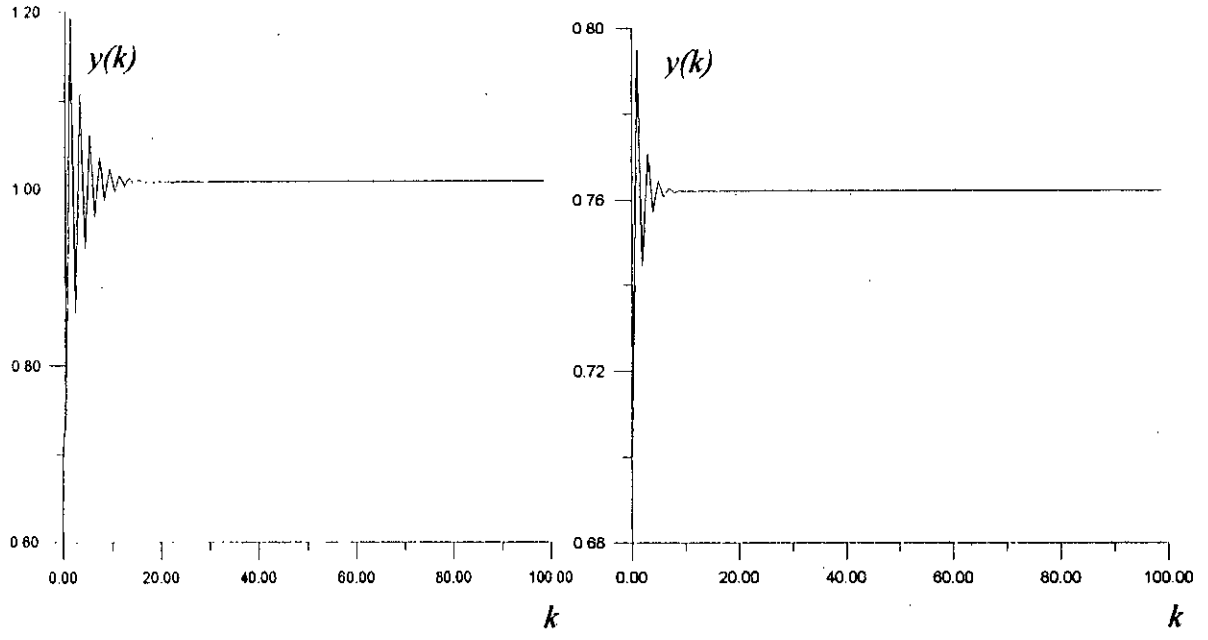
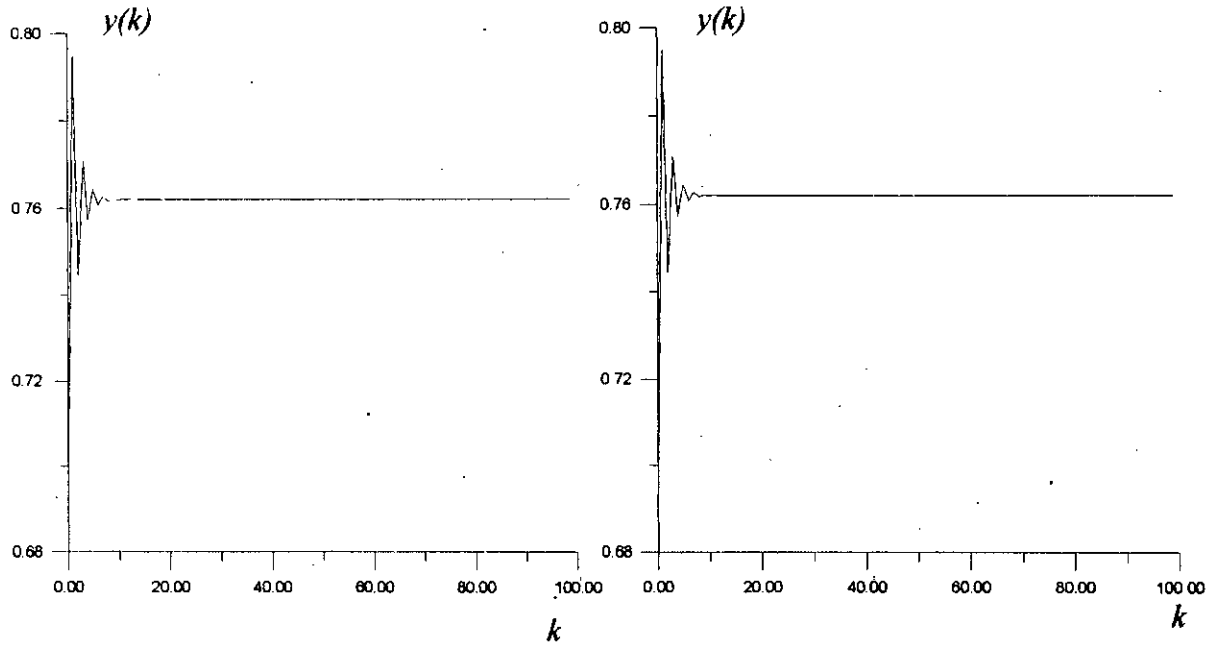


Figure 2.11 Test de stabilité du système autonome.



a). $u(k)$ 0.5 b). $u(k)$ =0.1



c). $u(k) = 1$ d). $u(k) = 2$

Figure 2.12 Test de stabilité en présence d'une entrée bornée.

Simulation 1

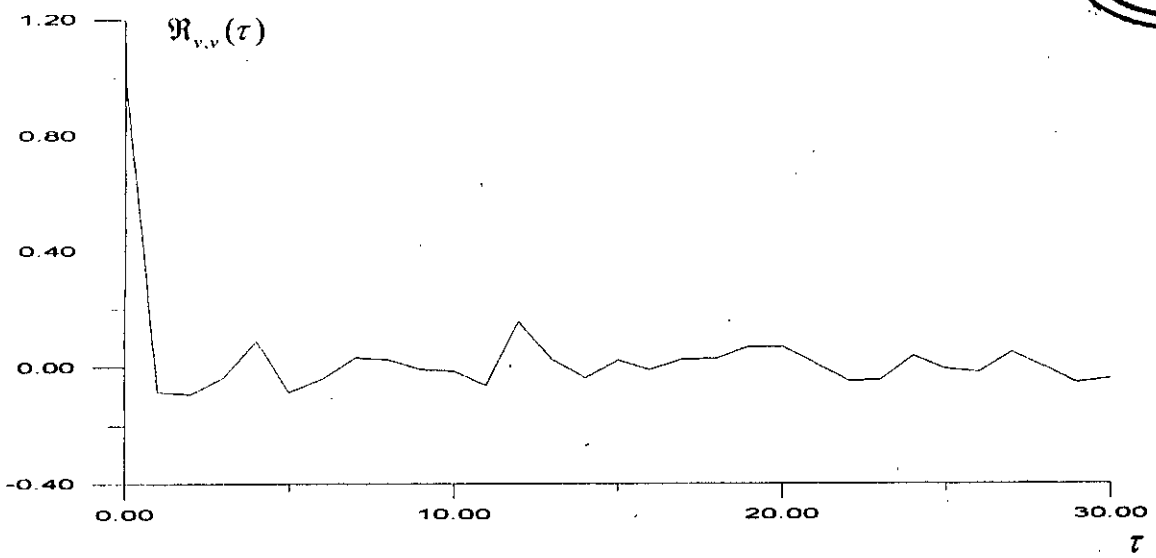


Figure 2.13 Autocorrelation du bruit blanc additif

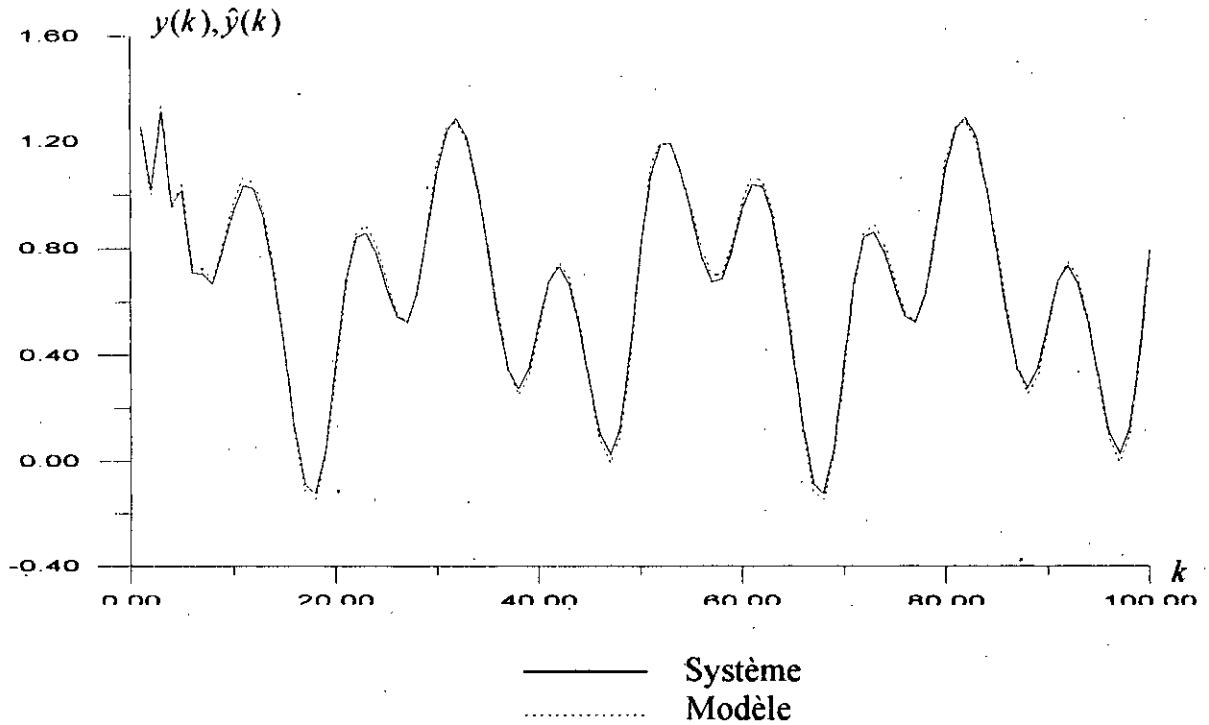


Figure 2.14. Sortie du système et sortie du réseau pour une entrée :

$$u(k) = 0.5\sin\left(\frac{2\pi k}{25}\right) + 0.5\sin\left(\frac{2\pi k}{10}\right)$$

Rapport Signal / Bruit	39 db
Nombre de données d'apprentissage	100
Le pas d'apprentissage	0.1
Le nombre d'iterations	768
temps d'apprentissage	6 minutes, 54 secondes, 26 centième
Matériel utilisé	Micro-Ordinateur DX2 66 MHZ

Tableau 2.3. Caractéristiques de la simulation 1.

Dans cette simulation nous avons affecté le système par un bruit blanc avec un rapport S/N de 39 db . D'après la figure (2.14), on remarque que les deux sorties sont indiscernables. Le réseau de neurones a réussi à apprendre la dynamique du système malgré l'existence du bruit.

Simulation 2

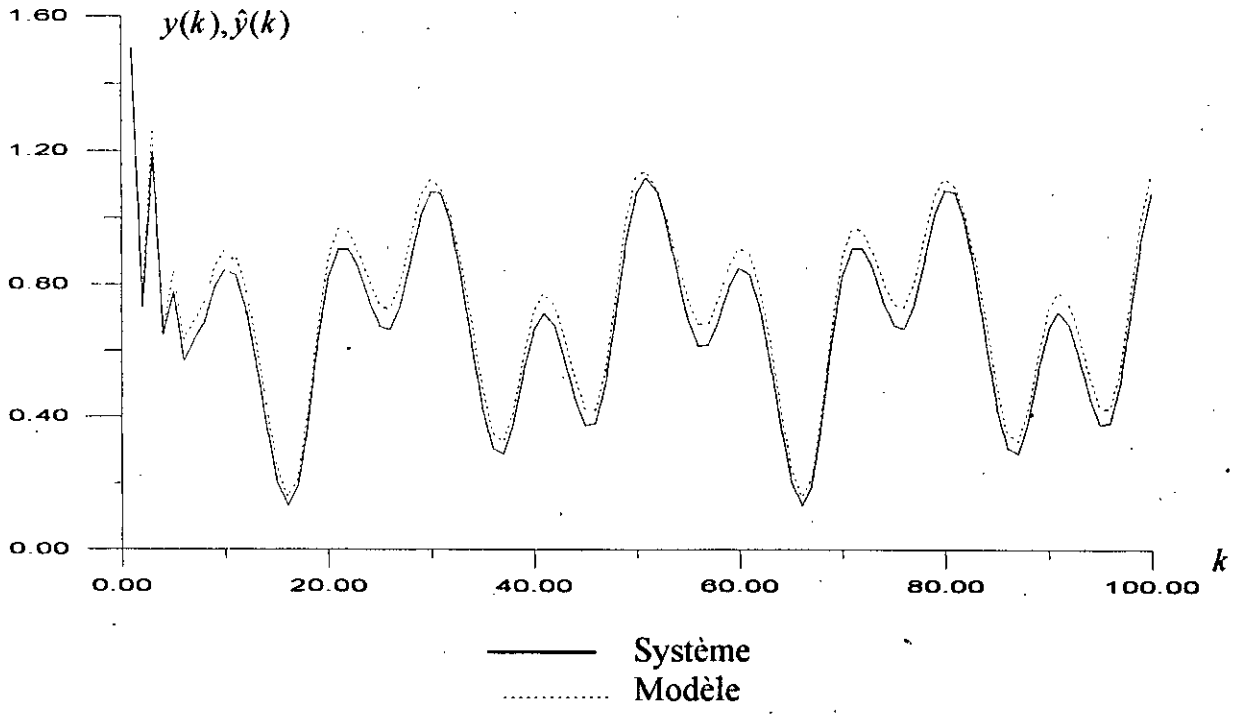


Figure 2.15 . Sortie du système et sortie du réseau pour une entrée :

$$u(k)=\left[\sin\left(\frac{2\pi k}{25}\right)+\sin\left(\frac{2\pi k}{10}\right)+\cos\left(\frac{2\pi k}{10}\right)+\cos\left(\frac{2\pi k}{25}\right)\right]\Big/4$$

Rapport Signal / Bruit	23 db
Nombre de données d'apprentissage	60
Le pas d'apprentissage	0.1
Le nombre d'iterations	4140
temps d'apprentissage	9 minutes,12 secondes,28 centième
Matériel utilisé	Micro-Ordinateur DX2 66 MHZ

Tableau 2.4. Caractéristiques de la simulation 2.

Simulation 3

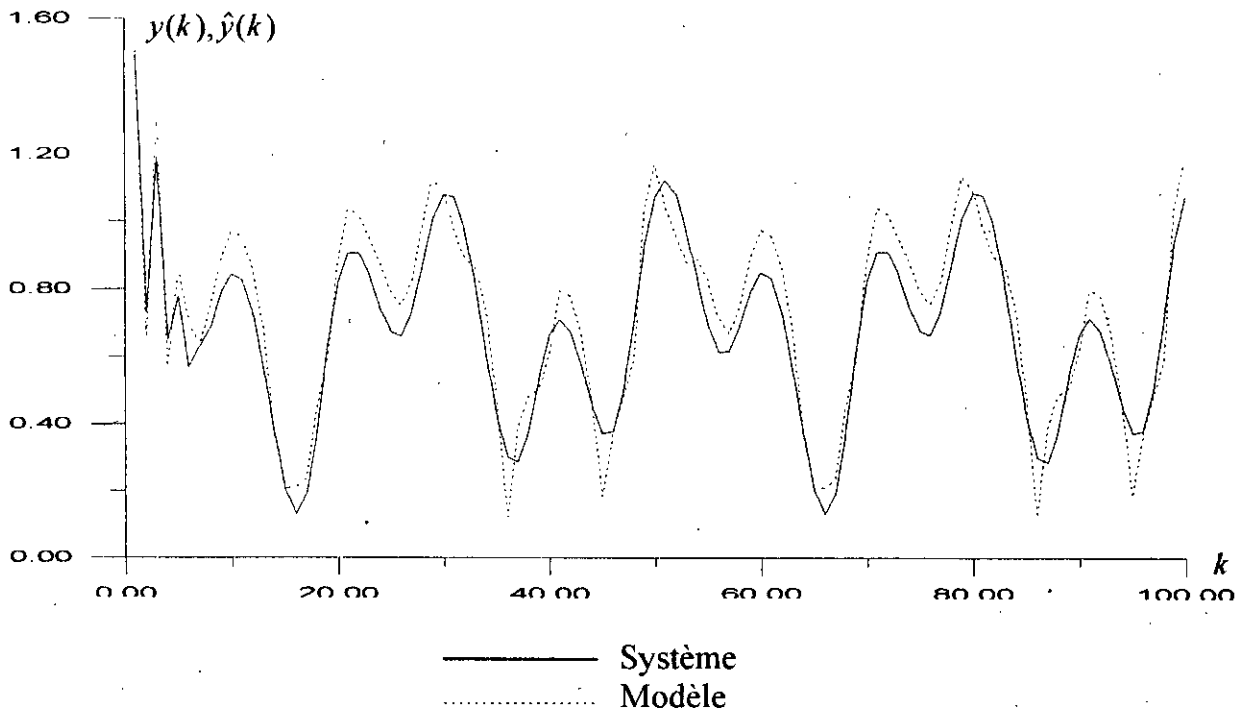


Figure 2.16. Sortie du système et sortie du réseau pour une entrée :

$$u(k) = \left[\sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{10}\right) + \cos\left(\frac{2\pi k}{10}\right) + \cos\left(\frac{2\pi k}{25}\right)\right] \Big/ 4$$

Rapport Signal / Bruit	23 db
Nombre de données d'apprentissage	60
Le pas d'apprentissage	0.1
Le nombre d'iterations	100000
temps d'apprentissage	47 minutes,20 secondes,20 centième
Matériel utilisé	Micro-Ordinateur DX2 66 MHZ

Tableau 2.5. Caractéristiques de la simulation 3.

Le but des simulations (2) et (3) est de montrer que l'apprentissage des réseaux de neurones peut engendrer quelques inconvénients. La figure (2.15) montre le résultat de l'identification après 4140 itérations. On remarque que les deux sorties sont très proches l'une de l'autre malgré que l'amplitude du bruit est très élevée. Après l'obtention de ce résultat, nous avons continué la procédure d'identification jusqu'à 100000 itérations et on a obtenu le résultat illustré par la figure (2.16). On remarque que la qualité de l'identification commence à se dégrader.

Simulation 4

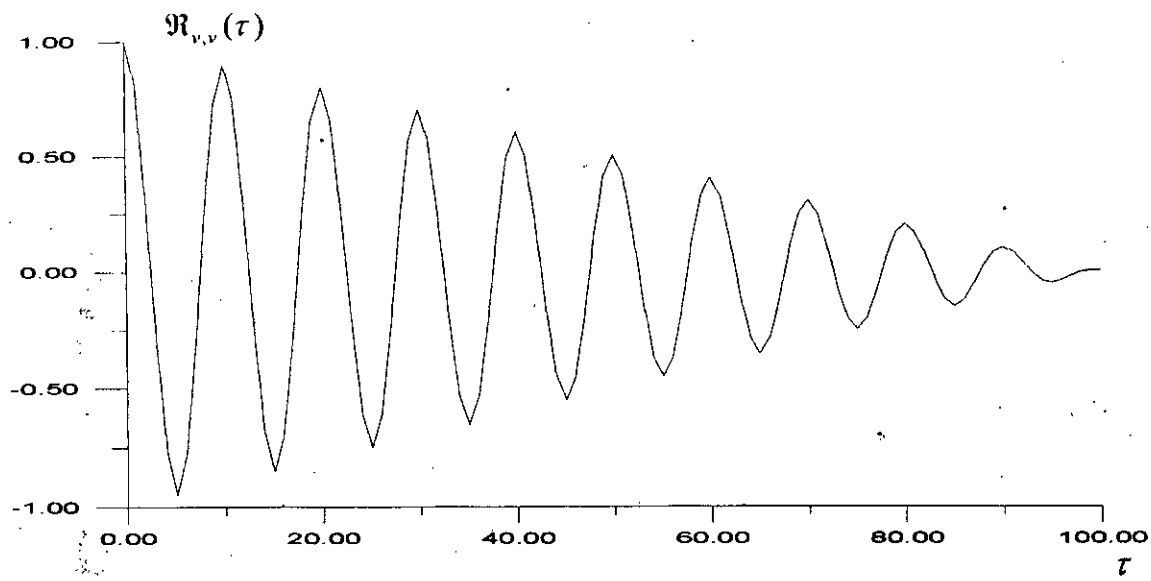


Figure 2.17. Autocorrelation du bruit additif.

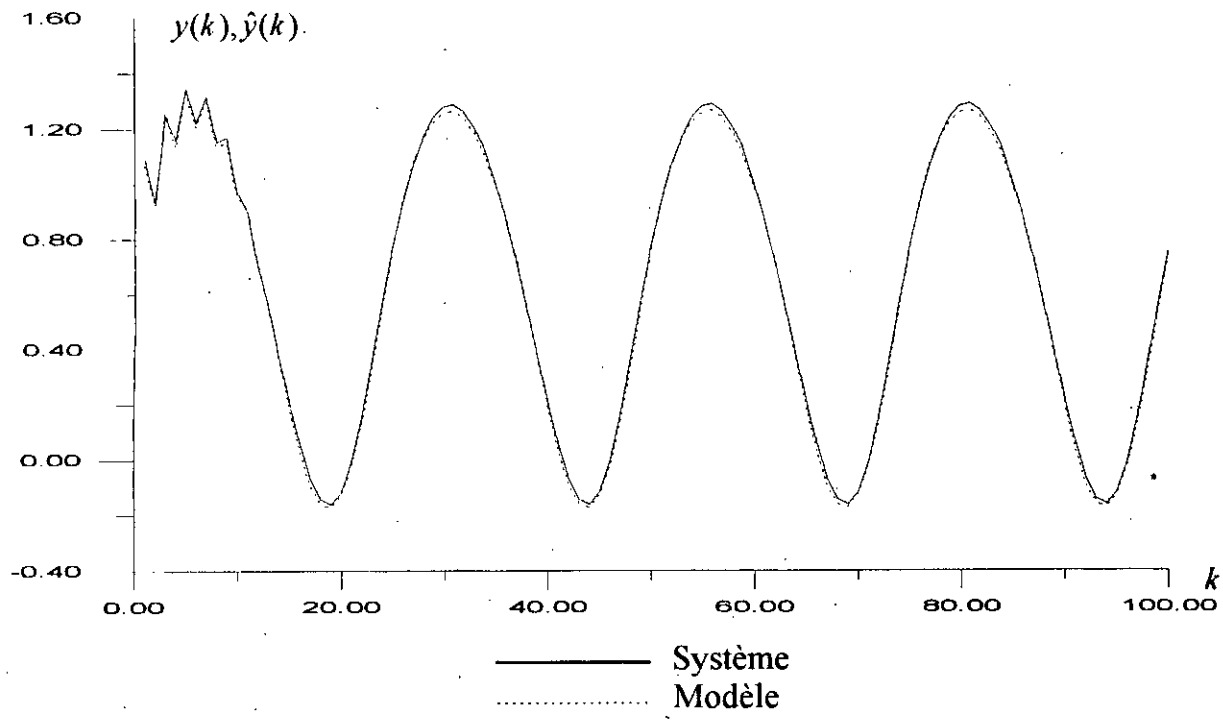


Figure 2.18. Sortie du système et sortie du réseau pour une entrée $u(k) = \sin\left(\frac{2\pi k}{25}\right)$.

Rapport Signal / Bruit	39 db
Nombre de données d'apprentissage	100
Le pas d'apprentissage	0.1
Le nombre d'iterations	1000
temps d'apprentissage	8 minutes,58 secondes, 0 centième
Matériel utilisé	Micro-ordinateur DX2 66 MHZ

Tableau 2.6. Caractéristiques de la simulation 4.

Dans cette simulation on a affecté le système par un bruit corrélé, car notre but est de montrer que les réseaux de neurones peuvent faire l'identification d'un système affecté par un bruit corrélé et cela contrairement à la méthode des moindres carrés qui

donne un estimateur biaisé si le bruit additif est corrélé [20]. Comme indiqué par la figure (2.18) , les deux sorties sont superposées. D'où la capacité des réseaux de neurones dans l'identification des systèmes quelque soit la nature du bruit additif.

Le problème qui se pose maintenant c'est quand faudrait-il arrêter l'apprentissage des données bruitées? Cette question est très importante, car si on arrête l'apprentissage trop tôt le réseau n'aura rien appris. Dans le cas échéant, le réseau risque d'apprendre le bruit qui entache les données et il y aura une « *sur-spécialisation* » du réseau (on voit clairement ce phénomène dans les simulations (2) et (3)). Pour éviter ce phénomène, on propose deux solutions :

- La première solution consiste à ne pas trop pousser la procédure d'identification, et à chaque fois, les performances du réseau sont testées jusqu'à atteindre une bonne identification du système.
- La deuxième solution consiste à faire le filtrage du bruit additif. Pour cela, on propose le schéma d'identification illustré par la figure (2.19).

Pour mieux comprendre cette idée, on suppose que le système est régi par l'équation

$$y_s(k) = f_s(u(k)) \tag{2.12}$$

où f_s est une fonction non-linéaire, y_s et u sont respectivement la sortie et l'entrée du système.

La sortie bruitée du système est :

$$y(k) = y_s(k) + v(k) \tag{2.13}$$

Si on suppose que le bruit $v(k)$ est un bruit corrélé, alors il peut être considéré comme résultat de filtrage d'un bruit blanc. Donc notre but est d'identifier le système et le filtre. Le système global sera régi par le système d'équations

$$\begin{cases} v(k) = f_v(e(k)) \\ y(k) = f_s(u(k)) + v(k) \end{cases} \tag{2.14}$$

où $e(k)$ est bruit blanc.

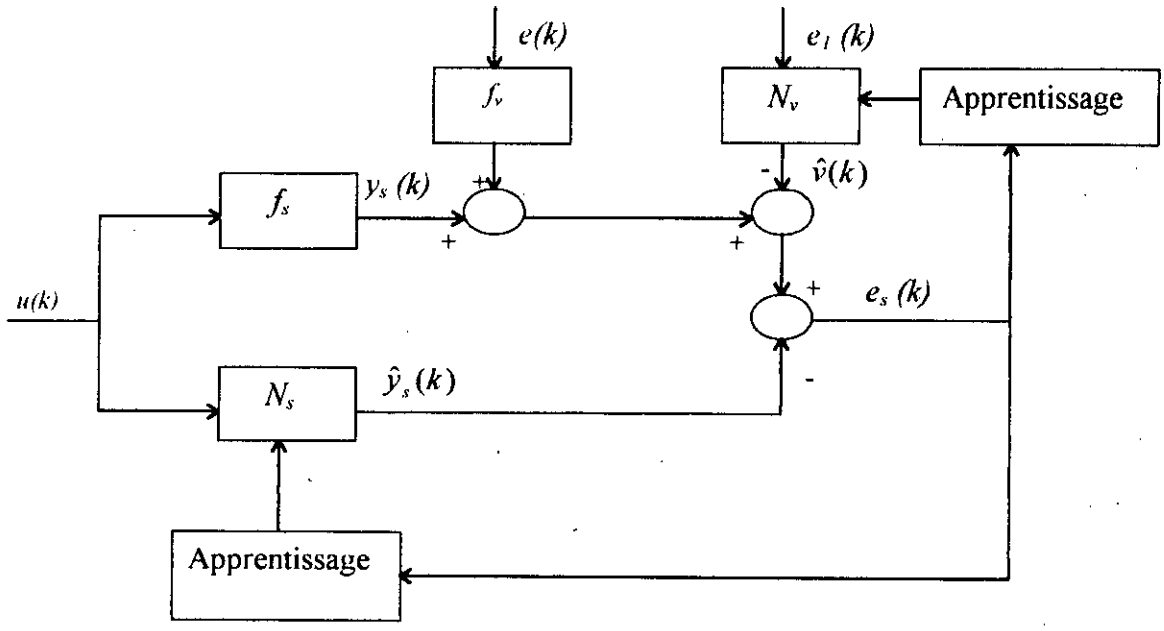


Figure 2.19. Identification avec filtrage du bruit additif.

Pour le modèle d'identification, on propose un réseau de neurones qui identifie le filtre, et un autre réseau pour l'identification du système. Le modèle d'identification sera régi par le système d'équations

$$\begin{cases} \hat{v}(k) = N_v\big(e(k)\big) \\ \hat{y}_s(k) = N_s\big(u(k)\big) \end{cases} \tag{2.15}$$

où N_s et N_v sont des réseaux de neurones.

L'erreur qu'on doit utiliser pour l'apprentissage du réseau N_s est :

$$\begin{aligned} e_s(k) &= y_s(k) - \hat{y}_s(k) = \big[y(k) - v(k)\big] - \hat{y}_s(k) \\ &= y(k) - f_v\big(e(k)\big) - \hat{y}_s(k) \end{aligned} \tag{2.16}$$

Puisque la fonction f_v est inconnue , elle sera remplacée par le réseau de neurones N_v qui est une estimation de f_v. Alors l'erreur sera :

$$e_s(k) = y(k) - N_v\big(e(k)\big) - \hat{y}_s(k) \tag{2.17}$$

L'erreur utilisée pour l'apprentissage du réseau N_v est donnée par l'équation

$$e_v(k) = y_s(k) - \hat{y}_s(k) \tag{2.18}$$

Il reste maintenant à développer l'algorithme d'apprentissage des deux réseaux pour assurer la stabilité globale des deux boucles (apprentissage du N_s et N_v).

II.8. CONCLUSION

Dans ce chapitre, nous avons présenté des schémas d'identification basés sur les réseaux de neurones pour la représentation des systèmes qui peuvent être dans un environnement déterministe ou stochastique. Rappelons que l'identification par les réseaux de neurones consiste à remplacer le modèle paramétrique classique par un modèle neuronal, puis on adapte les poids synaptiques du réseau pour qu'il représente avec fidélité le système physique.

On a considéré le cas des systèmes monovariables et multivariables. On a ensuite testé la robustesse des réseaux de neurones dans l'identification des systèmes affectés par plusieurs types de perturbations. On a remarqué que lorsqu'on pousse la procédure d'identification, le réseau va apprendre le bruit qui entache les données. Pour résoudre ce problème, deux solutions sont proposées. La première consiste à ne pas trop pousser la procédure d'identification. La deuxième consiste à filtrer le bruit additif en utilisant une architecture d'identification qui, à notre sens, peut donner de bon résultat.

Enfin, on peut conclure que les capacités d'apprentissage et de généralisation des réseaux de neurones donnent des issues nouvelles pour le problème d'identification des systèmes non-linéaires.

CHAPITRE III

CONTROLE DES SYSTEMES PAR LES RESEAUX DE NEURONES

III.1. INTRODUCTION

Durant ces dernières décennies, on note une grande activité dans le domaine de la commande adaptative, dans le but d'améliorer les performances des réglages des procédés à paramètres variables dans le temps. Dans les processus industriels, le comportement dynamique du système est en général variable, et peut être partiellement ou totalement inconnu. Ainsi, les méthodes classiques de commande présentent parfois des performances insuffisantes. Par exemple, l'hypothèse d'invariance est souvent inadéquate car les procédés industriels évoluent parfois dans le temps, les points de fonctionnement dérivent et les valeurs des paramètres changent.

La théorie de contrôle fournit des outils d'analyse et de synthèse qui sont parfaitement adaptés aux systèmes linéaires et qui ont fait leurs preuves dans l'industrie. Cependant, les chercheurs et scientifiques dans les domaines incluant la commande d'avions, engins spatiaux, robotique,... etc., ont récemment montré un grand intérêt au contrôle non-linéaire. Car, en pratique il n'est pas toujours suffisant de linéariser le système à commander. Même lorsque la construction d'un tel modèle est réalisable, celui-ci ne permet que d'obtenir une estimation qualitative du système dans des limites définies, souvent très petites.

L'utilisation des réseaux de neurones offre un grand potentiel pour résoudre le problème du contrôle des systèmes non-linéaires. Dans la structure de contrôle contenant des réseaux de neurones, les signaux de commande sont générés par un réseau de neurones. Cependant le problème qui se pose est comment entraîner le contrôleur neuronal. Plusieurs approches ont été développées pour résoudre ce problème [1],[2],[11],[12],...etc.

Dans le présent chapitre on montre le principe d'utilisation des réseaux de neurones pour le contrôle des systèmes non-linéaires en utilisant la stratégie MRAC (**M**odel **R**eference **A**daptive **C**ontrol) contrôle adaptatif avec modèle de référence. On commence par une introduction aux systèmes adaptatifs, puis on présente les différentes architectures de contrôle adaptatif basées sur les réseaux de neurones. Dans la partie simulation, on présente les résultats du contrôle d'un système multivariable après avoir fait son identification «OFF-LINE». Ensuite on passe au contrôle d'un système non- linéaire où l'apprentissage du réseau de neurones se fait

d'une façon continue pour qu'il s'adapte aux différentes variations de la référence et aussi aux effets des perturbations qui affectent le système. On utilise pour cette application un réseau de neurones dynamique appelé DRNN (**D**iagonal **R**ecurrent **N**eural **N**etwork) [12].

III.2. PRINCIPE DE LA COMMANDE ADAPTATIVE

La commande adaptative est un ensemble de techniques utilisées pour l'ajustement automatique en temps réel des régulateurs des boucles de commande afin de réaliser ou maintenir un certain niveau de performance quand les paramètres du système à commander sont inconnus ou/et sont variables dans le temps.

Quelques tâches typiques pouvant être effectuées par un système de commande adaptative sont indiquées ci-dessous :

1. Ajustement automatique des régulateurs à la mise en oeuvre (effet : réduction du temps d'ajustement et amélioration des performances).
2. Détermination automatique des paramètres optimaux des régulateurs dans les divers points de fonctionnement du procédé.
3. Maintien des performances du système de commande quand les caractéristiques du procédé changent ou en présence de perturbations aléatoires.
4. Possibilité de mise en oeuvre de régulateurs plus complexes et plus performants que les P.I.D.
5. Détection des variations anormales des caractéristiques des procédés (ces variations se reflètent dans les valeurs des paramètres fournis par les algorithmes d'adaptation).
6. Conception de nouveaux procédés technologiques utilisant des systèmes de commande adaptative (pour assurer le fonctionnement correct du procédé).

Les techniques de commande adaptative ont été utilisées avec succès pour un grand nombre d'applications: asservissement de moteurs électriques, systèmes d'armes, robots manipulateurs, commande de procédés industriels,pilotage automatique, ... etc.

L'utilisation des systèmes de commande adaptative connaît aujourd'hui un essor certain, d'une part à cause de leurs complexités raisonnables et d'autre part à cause de l'utilisation de microprocesseurs pouvant servir de support pour leur mise en oeuvre.

Dans les systèmes de régulation conventionnels, la contre-réaction est essentiellement utilisée pour réduire (ou éliminer) l'effet des perturbations agissant sur les variables à réguler. Pour réaliser ceci, on mesure les variables, on les compare aux valeurs désirées et les différences obtenues sont appliquées à l'entrée du régulateur qui engendre la commande appropriée. Une approche conceptuellement similaire peut être considérée pour le problème du maintien des performances désirées d'un système de commande en présence de perturbations paramétriques. Il faut définir d'abord un indice de performance (I.P) du système. Il faut, après, calculer cet (I.P.) et le comparer avec l' I.P. désiré. L'écart obtenu va être traité par un « mécanisme d'adaptation ». La sortie de ce dernier agira sur les paramètres du régulateur ou directement sur le signal de commande afin de modifier d'une manière appropriée les performances du système. Ce principe est illustré dans la figure(3.1). Les termes « Systèmes Adaptatifs » est introduit pour la première fois par les chercheurs **DERNIK ET SHAHBENDER** en 1957 [17].

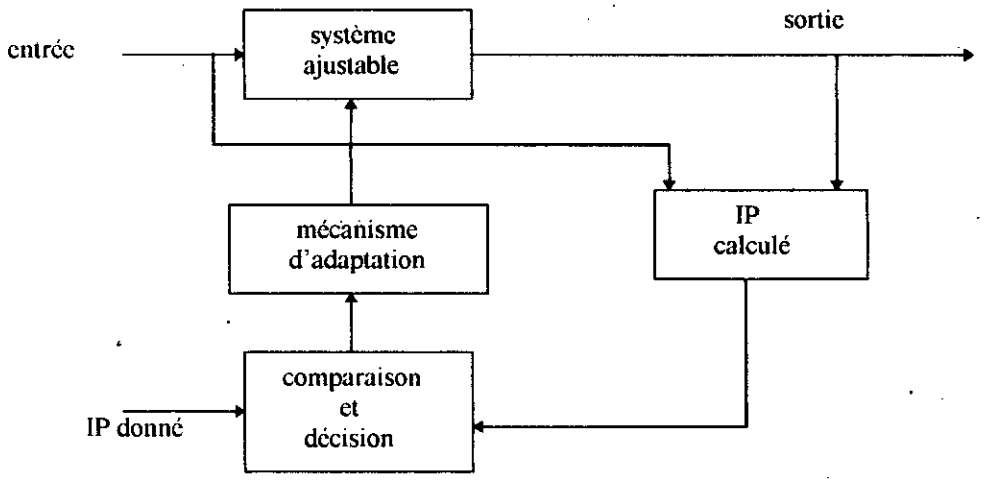


Figure 3.1. Système adaptatif

On considère la définition suivante pour un système de commande adaptative :
Un système de commande adaptative mesure un certain indice de performance (I.P) du système de commande et à partir de l'écart entre l'indice de performance désiré et l'indice de performance mesuré, le mécanisme d'adaptation modifie les paramètres du régulateur ajustable ou les signaux de commande afin de maintenir l'I.P. du système dans le voisinage des valeurs désirées.

Un système de commande à contre-réaction conventionnelle va réduire l'effet des perturbations agissant sur les variables à réguler, mais ses performances dynamiques vont varier sous l'effet des perturbations paramétriques. Un système de commande adaptative contient en plus d'une boucle de commande à contre- réaction ayant un régulateur à paramètres ajustables, une boucle supplémentaire qui agit sur les paramètres du régulateur afin de maintenir les performances du système en présence des variations des paramètres du procédé. Cette boucle supplémentaire a aussi une structure à contre-réaction [29].

III.3. COMMANDE ADAPTATIVE AVEC MODELE DE REFERENCE

La commande adaptative est relativement simple à mettre en oeuvre. Le schéma de commande adaptative avec modèle de référence a été originalement proposé par [WHITAKER,1958] [29]. Le développement de cette commande repose sur l'hypothèse fondamentale suivante:
Pour toutes les valeurs possibles des paramètres du procédé, on suppose qu'il existe un régulateur de structure donnée qui peut assurer la réalisation des performances désirées. Le rôle de la boucle d'adaptation est uniquement limité à trouver les bonnes valeurs des paramètres de ce régulateur dans chaque cas [29].

Le schéma illustré dans la figure (3.2) est appelé « Commande Adaptative avec Modèle de Référence Explicite » [29]. Le modèle de référence n'est autre qu'une réalisation de la fonction de transfert désirée du système de commande en boucle fermée.

Dans ce cas, le calcul est fait afin que :

1. L'erreur entre la sortie du procédé et la sortie du modèle soit identiquement nulle pour des conditions initiales identiques.
2. L'erreur initiale s'annule avec une dynamique préspécifiée (c'est la dynamique de régulation).

La différence entre la sortie du procédé et la sortie du modèle de référence est une mesure de la différence entre la performance réelle et la performance désirée. Cette information est utilisée par le mécanisme d'adaptation (qui reçoit aussi d'autres informations) pour ajuster automatiquement les paramètres du régulateur.

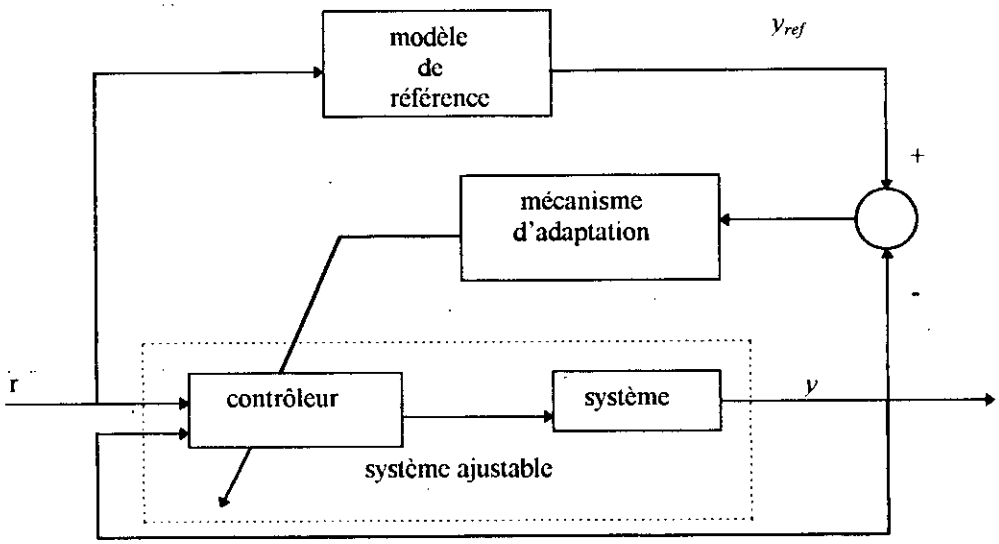


Figure 3.2 Commande adaptative avec modèle de référence.

III.3.1. *COMMANDE ADAPTATIVE DIRECTE ET INDIRECTE*

Deux approches sont fondamentales dans la théorie de la commande adaptative. Elles sont par ailleurs équivalentes dans la plupart des cas et on distingue généralement deux types de schémas de commande adaptative.

a). *Les schémas indirects*

Ils sont développés à partir d'une approche naturelle du concept de commande adaptative. Elle consiste à identifier en temps réel les paramètres du modèle du système et à les utiliser pour le calcul de la loi de commande comme s'ils étaient les paramètres réels du processus(fig 3.3). Toutes les stratégies de commande linéaire et

les méthodes d'identification peuvent être combinées pour la synthèse d'un schéma de commande adaptative de ce type. Cependant, le choix de cette combinaison doit conduire à la stabilité du système de commande adaptative.

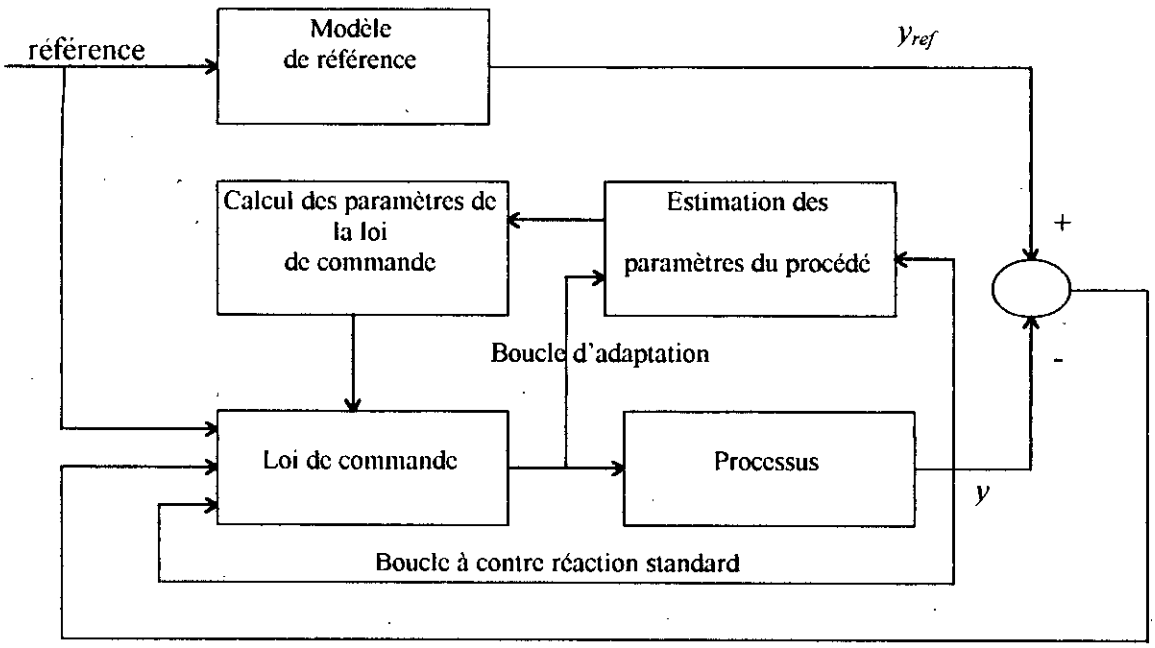


Figure 3.3 . Schéma de commande adaptative indirect

b). *Les schéma directs*

Ces schémas conduisent directement à l'estimation des paramètres de la loi de commande; la phase de calcul des paramètres du système est ainsi éliminée(fig3.4). Le problème principale dans un système MRAC est de déterminer le mécanisme d'adaptation pour non seulement minimiser l'erreur, mais aussi obtenir un système stable.

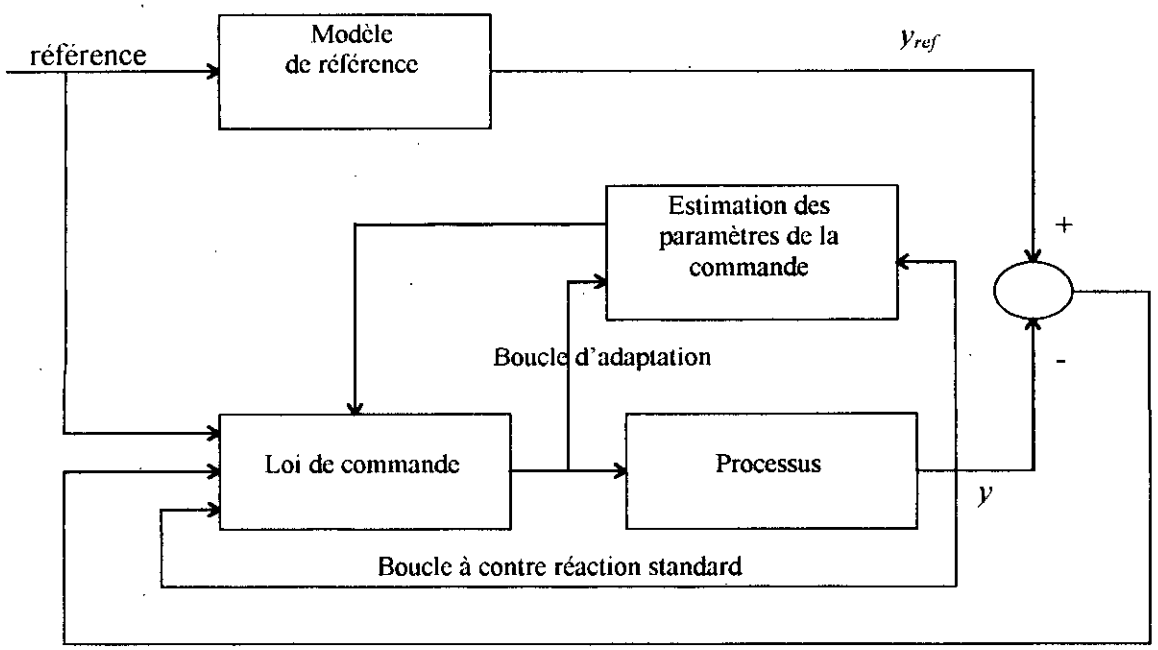


Figure 3.4. Schéma de commande adaptative direct

III.3.2. *DIFFERENTES CONFIGURATIONS DES SYSTEMES MRAC*

1.SYSTEMES LMFC

Le système LMFC « *linear model following control* » est très utile quand les paramètres du système et le modèle de référence sont connus (fig 3.5).

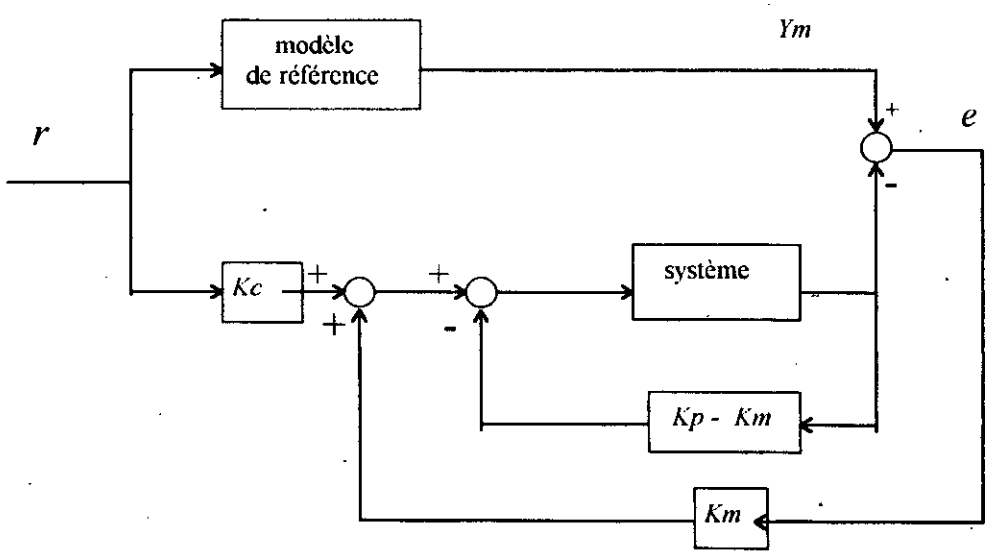


Figure 3.5. Système LMFC « *Linear model following control* ».

2.SYSTEMES AMFC

Le système AMFC « *adaptive model following control* » est utilisé quand les paramètres du système sont inconnus ou lorsque d'importantes variations se produisent. Le mécanisme d'adaptation dans la figure(3.6) modifie les paramètres K_C et K_P pour assurer une bonne poursuite du modèle de référence.

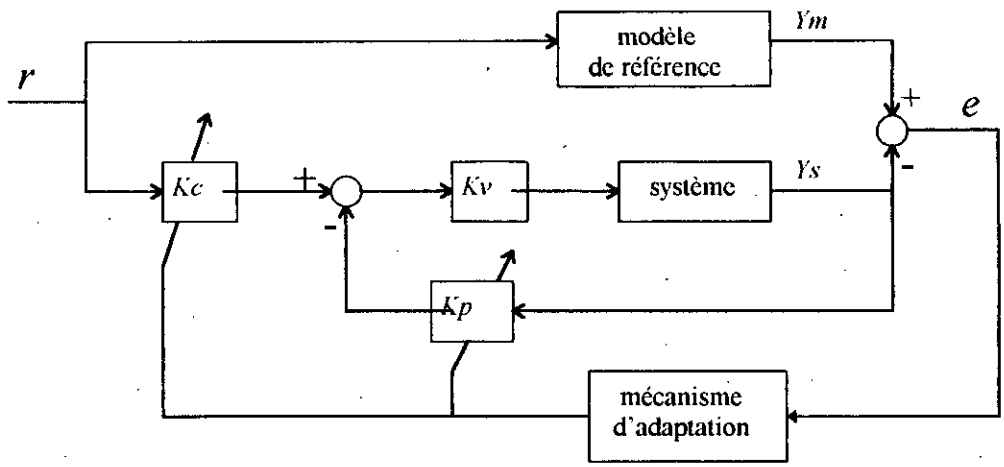


Figure 3.6. Système AMFC « *adaptive model following control* ».

III.4. LE CONTROLE ADAPTATIF DES SYSTEMES NON LINEAIRES PAR LES RESEAUX DE NEURONES

La plupart des travaux effectués dans les systèmes MRAC sont dirigés surtout vers le contrôle des systèmes linéaires invariants à paramètres inconnus [18],[19]. Le but est le développement des lois pour l'ajustement des paramètres du contrôleur pour assurer la stabilité du système global. Dans ce paragraphe, par contre, nous sommes intéressés par le contrôle adaptatif des systèmes non linéaires.

Le contrôle adaptatif des systèmes non-linéaires est basé sur le même principe que celui des systèmes linéaires, mais à la place des gains linéaires, des réseaux de neurones sont utilisés.

La capacité d'approximation non-linéaire des réseaux de neurones offre un grand potentiel pour le contrôle non linéaire. C'est, le réseau de neurones multicouches qui est le plus utilisé grâce à sa capacité d'approximation de n'importe quelle fonction non linéaire [1].

Le professeur **K.S. NARENDRA** de l'université de YALE (USA) [1],[9],..etc, a contrôlé des systèmes non linéaires en passant tout d'abord par l'identification « OFF-LINE » du système par les réseaux de neurones. Puis à l'aide des réseaux obtenus, il a obtenu les commandes nécessaires pour que le système suit la trajectoire du modèle de référence.

III.4.1 *CONTROLE ADAPTATIF DIRECTE PAR LES RESEAUX DE NEURONES*

Pour un contrôle satisfaisant d'un système non linéaire, il est préférable d'utiliser un contrôle adaptatif où les poids synaptiques du contrôleur seront ajustés d'une façon continue « ON-LINE » (fig.3.7).

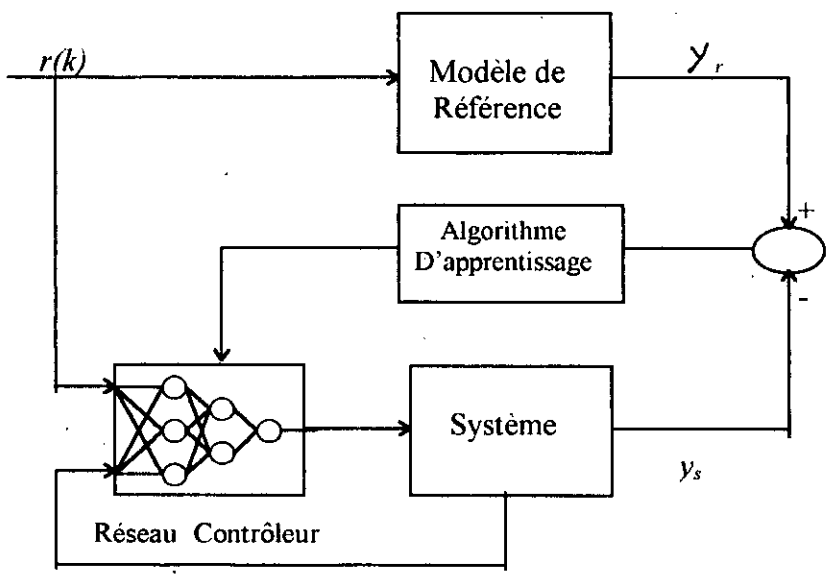


Figure 3.7. Contrôle adaptatif direct par les réseaux de neurones

Une autre configuration consiste à utiliser des gains fixes en parallèle avec le réseau de neurones contrôleur,comme sur la figure(3.8) [11]. Les gains du contrôleur sont utilisés pour la stabilisation du système et génèrent une commande approximative. Ensuite, le système peut être commandé dans son domaine d'opération par le réseau de neurones qui sera soumis à un apprentissage «ON-LINE».

La loi de contrôle délivré par le réseau et les gains est donnée par les équations suivantes :

$$\begin{cases} u(t) = k_r\, r(t) - k^T x_p(t) + u_{net}(t) \\ u_{net}(t) = f\left(x_p(t), x_m(t), r(t)\right) \end{cases} \tag{3.1}$$

où *f(.)* est une fonction non linéaire.

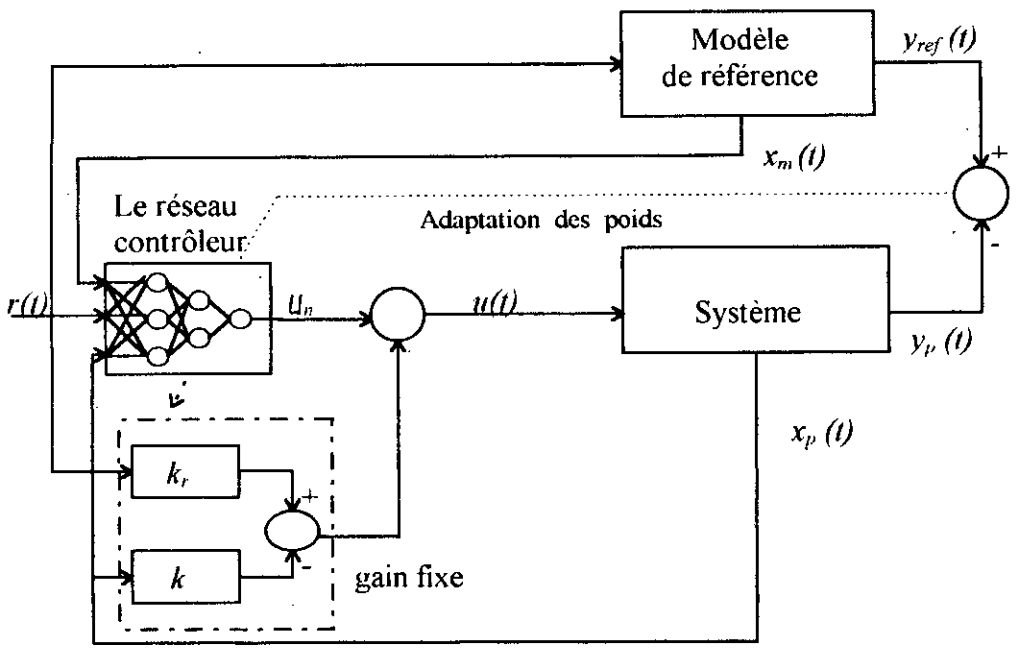


Figure 3.8. Contrôle adaptatif direct par les réseaux de neurones et un gain fixe.

Dûe à la non-linéarité du système et du réseau de neurones, trouver la méthode d'apprentissage du réseau pour garantir la stabilité globale du système en boucle fermée est encore un problème ouvert [11].

Dans ce travail, on va montrer comment utiliser l'algorithme de la rétropropagation pour l'apprentissage du réseau contrôleur.

Le critère de performance utilisé est :

$$J\left(k, w(k)\right) = \frac{1}{2}\left(y_s(k) - y_{ref}(k)\right)^2 = \frac{1}{2}e^2(k) \tag{3.2}$$

Les poids synaptiques peuvent être adaptés en utilisant la méthode du gradient.

Si le Jacobien $\frac{\partial y_p}{\partial u_{net}}$ du système est connu, alors on peut ajuster les poids par l'équation (3.3).

$$w(k+1) = w(k) - \eta \frac{\partial J(k, w(k))}{\partial w(k)} = w(k) - \eta \frac{\partial y_p}{\partial u_{net}} e(k) \frac{\partial u_{net}}{\partial w(k)} \tag{3.3}$$

En général nous n'avons pas une bonne connaissance du système à contrôler. Par conséquent, il est impossible d'avoir une formule analytique du Jacobien.

Une autre méthode utilise une technique stochastique pour l'ajustement des poids [11]. Cette méthode consiste à additionner aux poids un vecteur de perturbations qui suit une distribution gaussienne. Uniquement les poids qui donnent une décroissance de *J(k)* seront retenus, les autres seront ignorés. La valeur de *J(k)* est calculée après un certain nombre *N* de mesures en utilisant (3.4).

$$J(k) = \frac{1}{2N} \sum_{i=1}^{N} \left(y_s(k-i) - y_{ref}(k-i) \right)^2 \tag{3.4}$$

A la fin de chaque *N* mesures, *J(k)* est comparé avec sa valeur précédente pour les mêmes valeurs du signal de référence, ensuite les poids seront retenus ou ignorés. En utilisant cette méthode, la convergence peut être généralement lente.

III.4.2. *CONTROLE ADAPTATIF INDIRECT PAR LES RESEAUX DE NEURONES*

III.4.2.1. *CONTROLE PAR IDENTIFICATION INVERSE*

L'apprentissage du réseau de neurones contrôleur passe par deux étapes :

1. Etape de modélisation de la dynamique inverse du système.
2. Etape pour la génération de la commande.

a). *IDENTIFICATION INVERSE DU SYSTEME*

Soit un système monovariable non-linéaire invariant dans le temps décrit par l'équation suivante :

$$\dot{x} = f(x, u)$$

x représente l'état et u représente la commande du système (pour faciliter la discussion, on suppose que u est un scalaire). Notre but est de trouver un contrôleur discret à ce système. On va considérer les hypothèses suivantes :

Hypothèse 1

L'ordre du système est connu.

Hypothèse 2

L' état du système est mesurable.

Soit $x(k)$ et $u(k)$ l'état et la commande à l' instant k. Alors par définition, l'état du système à l'instant $k+1$ est en fonction de l'état et la commande à l'instant k. On obtient l'équation suivante :

$$x(k+1)=f_1(x(k),u(k)) \tag{3.5}$$

De la même façon l'état à l'instant $k+2$ est en fonction de la commande entre les instants $k+1$ et $k+2$, et de $x(k+1)$.

$$x(k+2)=f_1(x(k+1),u(k))=f_1[f_1(x(k),u(k),u(k+1))]$$

En suivant le même raisonnement jusqu'à l'instant $k+n$.

$$x(k+n)=f_n(x(k),U) \tag{3.6}$$

où $$U=[u(k),u(k+1),\ldots,u(k+n-1)]^T$$

Hypothèse 3

L'équation (3.6) est inversible de manière unique pour le vecteur U. Donc U peut être en fonction de $x(k+n)$ et $x(k)$.

$$U=g[x(k),x(k+n)] \tag{3.7}$$

L'équation (3.7) représente la relation fondamentale de dynamique inverse du système. Ceci implique que les valeurs de la commande qui vont amener le système de l'état initial $x(k)$ à l'état final $x(k+n)$ se trouvent dans le vecteur U.

Il suffit donc faire l'approximation de la fonction g par un réseau de neurones.

$$\hat{u}=N_g(x(k),x(k+n),W) \tag{3.8}$$

N_g :réseau de neurones.

W: Les poids synaptiques.

La procédure d'identification est illustrée par la figure (3.9.a).

b). LA PROCEDURE D'APPRENTISSAGE DU CONTROLEUR NEURONAL

Une méthode pour contrôler le système d'une façon «ON-LINE» est d'utiliser l'équation (3.8) pour générer le signal de contrôle chaque n itérations, tandis que l'apprentissage du réseau de neurones se fait à chaque itération.

La procédure d'apprentissage (identification inverse) et le contrôle est résumé comme suit :

1. La mesure de l'état : Mesurer et stocker l'état $x(k)$ à l'instant k .

2.L'apprentissage «ON-LINE» du réseau à la k^{eme} periode d'echantillonage :

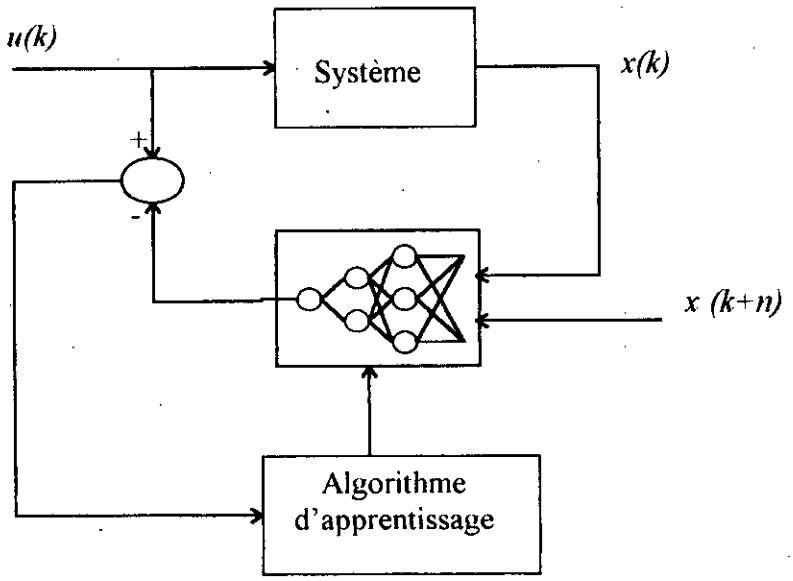
Utiliser le vecteur de commande $u(k-n), u(k-n+1),..., u(k-1)$ qui a été utilisé pour amener le système de l'état $x(k-n)$ jusqu'à $x(k)$ pour l'apprentissage du réseau.

3.Le calcul de la commande : en présence de l'état actuel $x(k)$ et l'état désiré $x_d\ (k+n)$, le réseau de neurones délivre à sa sortie un vecteur de commande pour les prochaines n périodes d'échantillonnage.

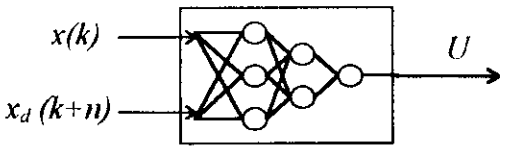
4. L'apprentissage «ON-LINE» du réseau à la $(k+1)^{eme}$,$(k+2)^{eme}$,...,$(k+n-1)^{eme}$ période d'échantillonnage: Quand les états $x(k+1),...,x(k+n-1)$ sont mesurés , faire l'apprentissage du réseau de la même manière que l'étape 2.

5. Répétition

Quand l'état $x(k+n)$ est obtenu, répéter les étapes 1-4.



a). Identification inverse par les réseaux de neurones



b). Génération du vecteur de commande.

Figure 3.9. Contrôle par identification inverse.

C). RECOMMENDATIONS

La procédure de contrôle présentée par les étapes 1-5 a une tendance à s'améliorer au fur et à mesure que l'apprentissage du réseau commence à se stabiliser. Cependant il se peut que les valeurs initiales des poids synaptiques ne donnent pas une bonne performance initiale du contrôleur; d'où la possibilité de divergence du système par rapport à l'état désiré. Dans ce cas, il faut stopper puis redémarrer l'opération sans réinitialiser les poids ajustés dans l'opération précédente qui donnent quand même une information sur le système.

La procédure de contrôle présentée dans les étapes 1-5 suppose qu'il n' y a aucune information à priori sur le système à contrôler. Mais, en pratique, il se peut que le système a été déjà contrôlé par un contrôleur classique (ex : P.I.D) et il se peut aussi que ce contrôleur a provoqué l'instabilité du système. Dans ce cas, on peut utiliser les données acquises sur le système pour faire l'identification de sa dynamique inverse d'une façon «OFF-LINE» par un réseau de neurones avant de commencer la procédure de contrôle. Le réseau contrôlleur donnera de meilleurs résultats [2].

Plusieurs systèmes non linéaires possèdent un bon modèle mathématique sans une loi de contrôle précise pour le contrôler et cela est dû à l'inexistence d'une méthode générale systématique pour le contrôle des systèmes non-linéaires [18],[19]. Un réseau de neurones peut être utilisé pour générer la commande nécessaire, après un apprentissage «OFF-LINE», utilisant les données générées par les modèles mathématiques donnés.

III.4.2.2. *CONTROLE PAR LA COMMANDE PREDICTIVE PAR LES RESEAUX DE NEURONES*

Le principe de cette méthode est de faire tout d'abord l'identification du système non-linéaire par un réseau de neurones. Le réseau obtenu va délivrer la prédiction de la sortie du système en présence de la commande $\hat{u}$ donnée par la routine d'optimisation. Cette routine ajuste la commande $\hat{u}$ pour que la sortie du système soit très proche de celle du modèle de référence.(fig.3.10)

La commande $\hat{u}$ est déterminée en minimisant le critère de performance suivant :

$$J = \sum_{j=1}^{N} \left(y_{ref}(k+j) - \hat{y}(k+j) \right)^2 \tag{3.9}$$

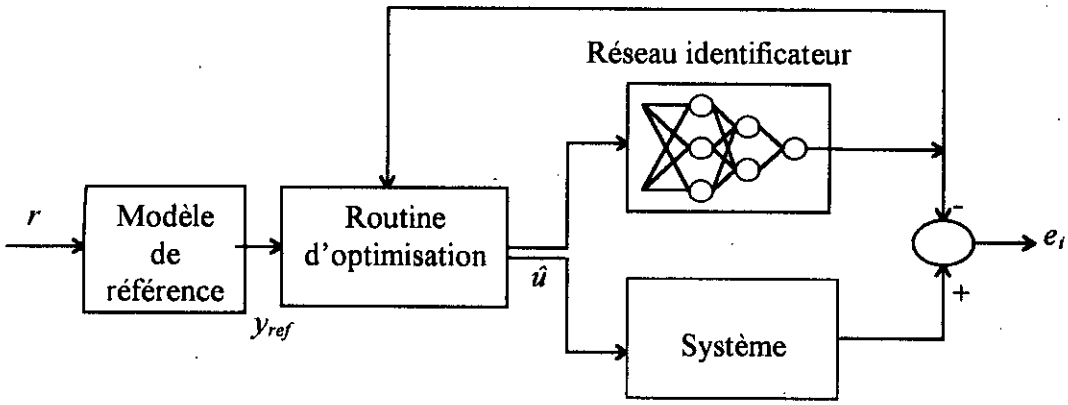


Figure 3.10. Structure de commande prédictive par les réseaux de neurones

Comme l'indique la figure (3.11), Une autre possibilité est d'utiliser un autre réseau de neurones qui va jouer le rôle de contrôleur. L'apprentissage de ce réseau se fait en utilisant la commande prédictive délivrée par la routine d'optimisation (boucle d'optimisation). L'avantage de cette configuration est que la routine d'optimisation peut être éliminée lorsque l'apprentissage du réseau contrôleur sera terminé.

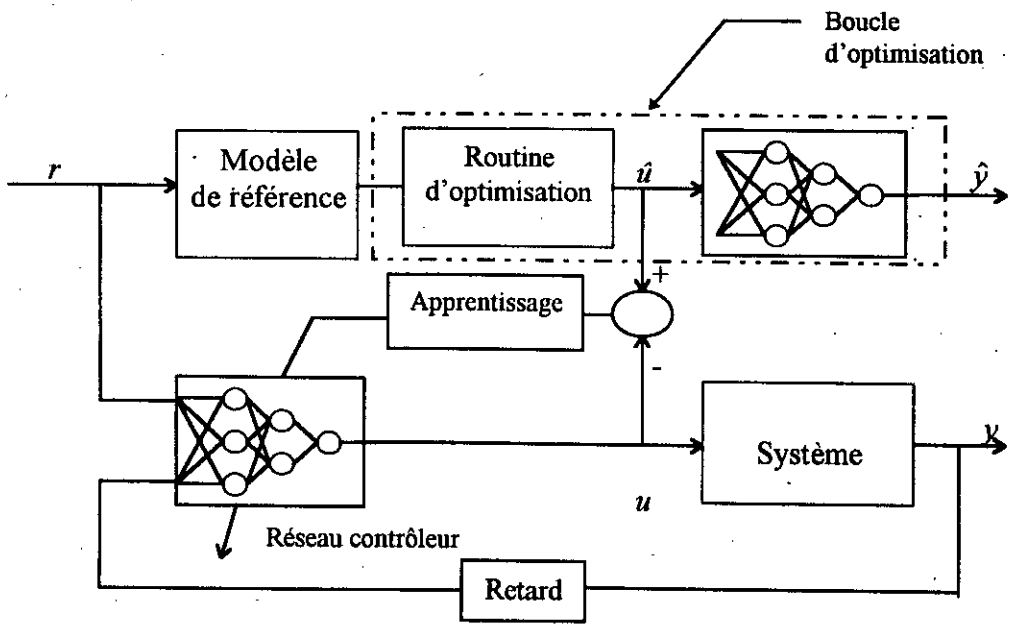


Figure 3.11. Apprentissage du réseau contrôleur par la commande prédictive.

III.4.2.2. *CONTROLE PAR IDENTIFICATION DIRECTE*

Dans le paragraphe (III.4.1) nous avons évoqué le problème de l'apprentissage du réseau contrôleur qui est basé sur le Jacobien du système et on a dit que ce Jacobien est en général inconnu. Une méthode plus exacte pour l'estimation du Jacobien est détaillée dans ce paragraphe.

Soit $y_r(k)$ et $y(k)$ les réponses désirées et actuelles du système. L'erreur utilisée pour l'apprentissage du réseau de neurones contrôleur (RNC) est $e_c(k)=y_r(k)-y(k)$ car notre but est de tendre cette erreur vers zero, $\|y_r(k)-y(k)\| \to 0$.

Nous savons que l'apprentissage supervisé d'un réseau de neurones est basé sur l'erreur à la sortie de ce réseau. Mais dans notre cas l'erreur disponible est entre le système et le modèle de référence. La technique proposée est d'utiliser un réseau de neurones identificateur (RNI) du système comme un canal de propagation de cette erreur jusqu'à la sortie du (RNC). Le réseau (MLP) convient très bien à cette utilisation, car il est lui même basé sur la rétropropagation de l'erreur. Malgré cette capacité du (MLP), il a l'inconvénient du temps d'apprentissage qui est très long, et puisque en pratique il y a la contrainte du temps réel, alors il faut trouver un réseau de neurones qui sert de canal de propagation de l'erreur et possédant un temps d'apprentissage très court.

Les chercheurs **CHAO-CHEE KU** et **KWANG Y-LEE** [12] ont développé une nouvelle architecture de réseau de neurones appelée *(Diagonal Recurrent Neural Network)* DRNN qui satisfait aux deux conditions. On commence par une description du réseau DRNN , puis on montre l'utilisation de ce réseau dans le contrôle adaptatif des systèmes non linéaires.

a). *LE RESEAU DRNN*

Ce réseau est un réseau dynamique dont la caractéristique est qu'il possède une couche cachée dont chaque neurone est un neurone récurrent.

Ce réseau est illustré par la figure (3.12).

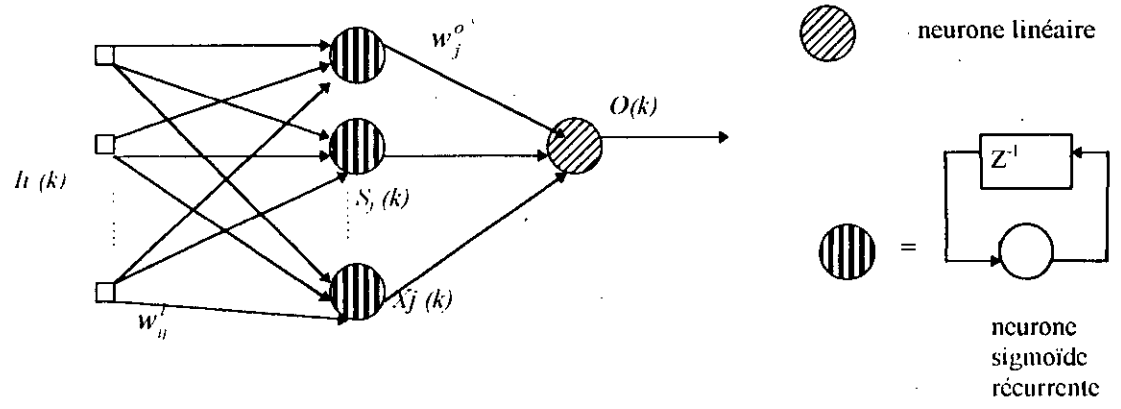


Figure 3.12. Le réseau DRNN.

A chaque instant k, $I_i(k)$ est la i^{eme} entrée, $s_j(k)$ est l'état du j^{eme} neurone récurrent « *caché ou diagonal* » ; $X_j(k)$ représente la sortie du jème neurone et $O(k)$ représente la sortie du réseau globale. W^I, W^D, W^o représentent respectivement les vecteurs des poids synaptiques de la couche d'entrée, de la diagonale (cachée) et de la sortie.
Le modèle mathématique est le suivant:

$$\begin{cases} O(k) = \sum_j w_j^o X_j(k), & X_j(k) = f\left(s_j(k)\right) \\ s_j(k) = w_j^D X_j(k-1) + \sum_i w_{ij}^I I_i(k) \end{cases} \tag{3.10}$$

La plupart des travaux utilisent le réseau MLP avec temps de retard pour résoudre un problème dynamique [1],[3],[4],[5],[10]. D'autre part, le réseau récurrent possède plus de capacité par rapport au réseau statique, tel que l'habilité de stocker les informations pour une utilisation ultérieure et il peut fonctionner avec des données variant dans le temps. Donc le réseau récurrent est très adapté aux systèmes dynamiques. L'algorithme d'apprentissage de ce type de réseau est appelé *(dynamic backpropagation algorithm)* (DPB) l'algorithme de la rétro-propagation dynamique. Cet algorithme sera développé dans le paragraphe suivant.

b).LE RESEAU DRNN POUR LE CONTROLE ADAPTATIF DES SYSTEMES

En se basant sur la stratégie MRAC, on va utiliser le DRNN pour contrôler un système dynamique inconnu. Le système sera identifié par un système d'identification appelé *(diagonal recurent neuroidentifier)*, le DRNI, qui produit des informations concernant le système au contrôleur *(diagonal recurent neurocontroler)*, le DRNC.

Ensuite le neurocontrôleur ajuste le système inconnu de sorte que l'erreur entre le système et le modèle de référence est minimisée. Le schéma bloc du contrôle par le réseau DRNN est montré dans la figure (3.13).

Durant l'apprentissage du réseau contrôleur DRNC, on a besoin de la sensibilité du système, mais puisque le système est normalement inconnu, le réseau identificateur DRNI est utilisé pour estimer la sensibilité y_u du système.

L'apprentissage des réseaux DRNC et DRNI se fait simultanément d'une façon «ON-LINE». Après le développement de l'algorithme (DBP), on montre à travers des simulations les capacités d'adaptation du réseau DRNN aux différentes variations du modèle de référence. On montre aussi sa capacité de rejection des bruits affectant le système. Le réseau DRNI et DRNC utilisent la même architecture illustré dans la figure (3.12) .

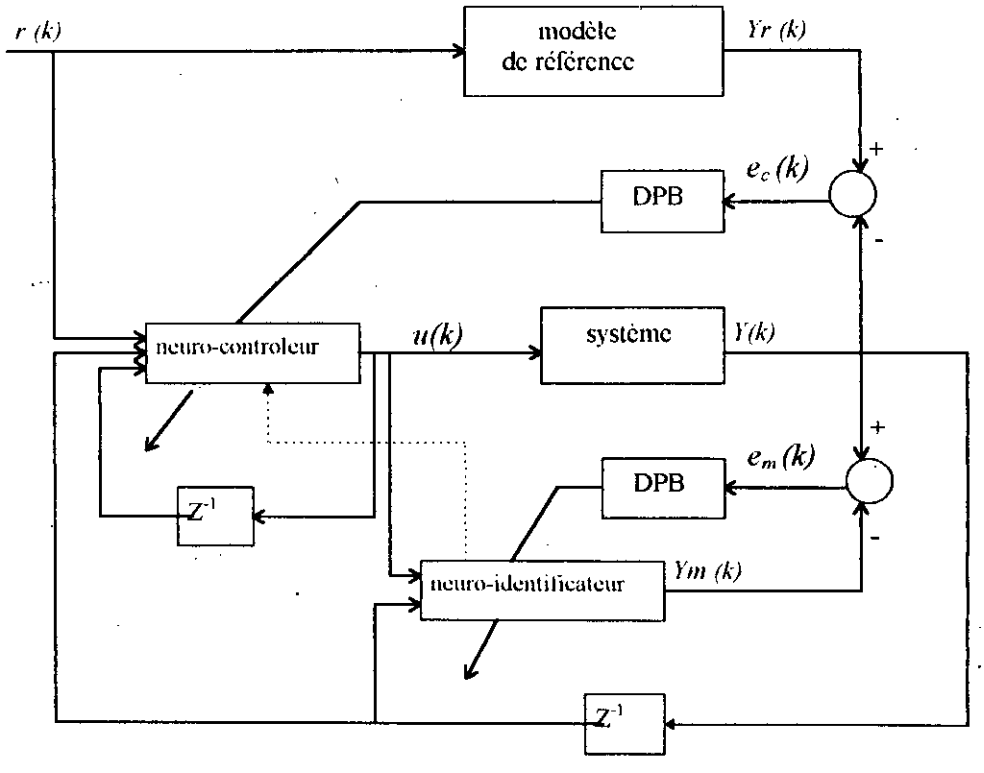


Figure 3.13. Le contrôle adaptatif par les réseaux DRNN.

b.1). ***L'apprentissage des réseaux contrôleur et identificateur DRNC et DRNI***

L'apprentissage du DRNC utilise la fonction d'erreur suivante :

$$E_c = \frac{1}{2}\left(y_r(k) - y(k)\right)^2 = \frac{1}{2}e_c^2(k) \qquad (3.11)$$

u(k) représente la sortie du DRNC et l'entrée du DRNI et le système.

La fonction d'erreur utilisée pour l'apprentissage du DRNI est :

$$E_m = \frac{1}{2}\left(y(k) - y_m(k)\right) = \frac{1}{2} e_m^2(k) \tag{3.12}$$

Les lois d'adaptation des poids synaptiques des réseaux DRNC et DRNI sont respectivement (3.13) et (3.14).

$$w_c(k+1) = w_c(k) + \eta_c\left(-\frac{\partial E_c}{\partial w}\right) \tag{3.13}$$

$$w_i(k+1) = w_i(k) + \eta_i\left(-\frac{\partial E_m}{\partial w}\right) \tag{3.14}$$

D'après (3.13) et (3.14), il faut trouver des expressions mathématiques des gradients des fonctions d'erreur par rapport aux poids synaptiques.

1. *Détermination des gradients* $\frac{\partial E_c}{\partial w}$ et $\frac{\partial E_m}{\partial w}$

Le gradient de la fonction d'erreur (3.11) par rapport au vecteur poids du contrôleur est:

$$\frac{\partial E_c}{\partial w} = -e_c(k)\frac{\partial y(k)}{\partial w} = -e_c(k) y_u(k)\frac{\partial u}{\partial w}$$

$$\frac{\partial E_c}{\partial w} = -e_c(k) y_u(k)\frac{\partial O(k)}{\partial w} \tag{3.15}$$

où $y_u(k) = \frac{\partial y(k)}{\partial u(k)}$ représente la sensibilité du système par rapport à son entrée. Puisque le système est inconnu, alors cette sensibilité doit être estimée.

Pour le réseau DRNI le gradient de la fonction d'erreur (3.12) est :

$$\frac{\partial E_m}{\partial w} = -e_m(k)\frac{\partial y_m(k)}{\partial w} = -e_m(k)\frac{\partial O(k)}{\partial w} \tag{3.16}$$

D'après les équations (3.15) et (3.16) on remarque que le terme $\frac{\partial O(k)}{\partial w}$ qui est le gradient de la sortie du réseau par rapport aux différents poids synaptiques est commun pour les réseaux DRNC et DRNI.

2. *Le calcul de* $\frac{\partial O(k)}{\partial w}$

En utilisant le système d'équations (3.10) les valeurs du gradient de la sortie par rapport aux poids synaptiques des couches de sortie, de la diagonale et de l'entrée sont respectivement données par:

$$\begin{cases} \dfrac{\partial O(k)}{\partial w_j^o} = X_j(k) \\ \dfrac{\partial O(k)}{\partial w_j^D} = w_j^o p_j(k) \\ \dfrac{\partial O(k)}{\partial w_{ij}^I} = w_j^o Q_{ij}(k) \end{cases} \tag{3.17}$$

où $p_j(k) = \dfrac{\partial X_j(k)}{\partial w_j^D}$ et $Q_{ij}(k) = \dfrac{\partial X_j(k)}{\partial w_{ij}^I}$

$P_j(k)$ et $Q_{ij}(k)$ sont calculés en utilisant (3.10) et (3.17) pour obtenir les équations suivantes :

$$\begin{cases} p_j(k) = f'(s_j)\left[X_j(k-1) + w_j^D p_j(k-1)\right] \\ p_j(0) = 0 \\ Q_{ij}(k) = f'(s_j)\left[I_i(k) + w_j^D Q_{ij}(k-1)\right] \\ Q_{ij}(0) = 0 \end{cases} \tag{3.18}$$

On peut dire maintenant que les différentes valeurs du gradient des sorties par rapport aux différents poids sont complètement déterminées.

3.*Determination de* $\dfrac{\partial E_m}{\partial w}$

D'après l'équation (3.16), on remarque que $\dfrac{\partial E_m}{\partial w}$ est fonction seulement de $\dfrac{\partial O(k)}{\partial w}$.

Donc il suffit de combiner le les équations (3.16), (3.17) et (3.18) pour avoir la valeur $\dfrac{\partial E_m}{\partial w}$

4.*determination de* $\dfrac{\partial E_c}{\partial w}$

Puisque nous avons déterminé $\dfrac{\partial O(k)}{\partial w}$, alors d'après l'équation (3.15), il reste à déterminer la sensibilité $y_u(k)$ du système. Puisque le système est inconnu, alors le terme $y_u(k)$ est inconnu. Cette valeur doit être estimée à partir de DRNI, car lorsqu'on fait l'apprentissage de ce dernier nous aurons $y(k) \approx y_m(k)$. ,et donc on peut dire que :

$$y_u(k) = \frac{\partial y(k)}{\partial u(k)} \approx \frac{\partial y_m(k)}{\partial u(k)} \tag{3.19}$$

nous avons :

$$\frac{\partial y_m(k)}{\partial u(k)} = \sum_j \frac{\partial o(k)}{\partial X_j(k)} \frac{\partial X_j(k)}{\partial u(k)} = \sum_j w_j^o \frac{\partial X_j(k)}{\partial u(k)}$$

De (3.10) on tire :

$$\frac{\partial X_j(k)}{\partial u(k)} = f'(s_j(k)) \frac{\partial s_j(k)}{\partial u(k)} \tag{3.20}$$

(*f est la fonction sigmoïde du neurone*).

Si on considère que le réseau DRNI possède seulement trois entrées (*u(k),y(k-1,X(k-1))* (fig.3.13), alors on aura l'équation suivante :

$$s_j(k) = w_j^D X_j(k-1) + w_{1,j}^I u(k) + w_{2,j}^I y(k-1) + w_{3,j}^I b_I \tag{3.21}$$

(b_I *représente le seuil d'activation*).

Donc $$\frac{\partial s_j(k)}{\partial u(k)} = w_{1j}^I$$

En utilisant les équations (3.19), (3.20) et (3.21), on tire la formule de la sensibilité:

$$y_u(k) \approx \frac{\partial y_m(k)}{\partial u(k)} = \sum_j w_j^o f'\left(s_j(k)\right) w_{1j}^I \tag{3.22}$$

En remplaçant (3.22) dans (3.15), on trouve la formule de $\frac{\partial E_c}{\partial w}$.

5. *Résumé de la procédure de contrôle adaptatif par le réseau* DRNN

1. Initialiser les poids synaptiques des réseaux DRNC et DRNI avec des valeurs arbitraires.
2. Calculer la sortie du modèle de référence y_r.
3. Utiliser le système d'équations (3.10) pour calculer la sortie du DRNC.
4. Utiliser la sortie du DRNC, « la commande », pour calculer la sortie y du système.
5. Utiliser la sortie du DRNC pour calculer la sortie y_m du DRNI.
6. En utilisant les équations (3.17) et (3.18) calculer $\frac{\partial O}{\partial w_i}$ du réseau DRNI.

7. Ajuster les poids du réseau DRNI par l'équation suivante :

$$w_I(k+1) = w_I(k) - \eta_I (y - y_m) \frac{\partial O}{\partial w_I}$$

8. Utiliser l'équation (3.22) pour calculer la sensibilité y_u *(k)* du système.

9. En utilisant les systèmes équations (3.17) et (3.18), calculer le terme $\frac{\partial O}{\partial w_c}$.

10. Ajuster les poids du réseau DRNC pour l'équation suivante :

$$w_c(k+1) = w_c(k) + \eta_c (y_r - y) y_u \frac{\partial O}{\partial w_c}$$

11. Répéter les étapes 2-10.

b.2). *Convergence et stabilité de la procédure de contrôle*

Dans les équations (3.13) et (3.14), il faut très bien choisir le coefficient d'apprentissage η pour garantir une stabilité et une bonne convergence. Pour une petite valeur de η, la convergence est garantie mais avec une faible vitesse; d'autre part si η est très large l'algorithme devient instable.

Dans ce paragraphe, on va citer deux théorèmes pour la détermination des pas d'apprentissage adaptatifs qui assurent la stabilité de la procédure de contrôle [12].

Théorème 3.1

Soit η_I le pas d'apprentissage pour le réseau DRNI et $g_{I,max}$ défini par :

$g_{I,\max} = \max_k \|g_I(k)\|$, où $g_I(k) = \frac{\partial O}{\partial w_I}$, $\|\|$ représente la norme euclidienne dans $\Re^n$.

Alors la convergence est garantie si η_I est choisi tel que: $0 < \eta_I < \frac{2}{g^2_{I,\max}}$

Théorème 3.2

Soit η_c le pas d'apprentissage du réseau DRNC et soit $g_{c,\max}$ défini par :

$$g_{c,\max} = \max_k \|g_c(k)\|, \text{ où } g_c(k) = \frac{\partial O}{\partial w_c}.$$

Soit $S_{\max} = h_I w^o_{I,\max} w^I_{I,\max} / 2$. Alors la convergence est garantie si η_c satisfait l'inégalité suivante :

$$0 < \eta_c < \frac{2}{S^2_{\max} g^2_{c,\max}}$$

Noter que h_I représente le nombre des neurones cachés.

III.5. RESULTATS DES SIMULATIONS

Dans cette section on commencera par le contrôle d'un système multivariable après son identification «OFF-LINE». Ensuite nous présenterons des simulations pour tester la capacité du réseau DRNN dans le contrôle adaptatif avec modèle de référence.

On va commencer par le contrôle d'un système stable où on test la capacité d'adaptation des réseaux DRNC et DRNI aux différentes variations de la référence. Ensuite nous allons effectuer des tests de robustesse de ces réseaux vis-à-vis des effets de perturbations affectant le système.

Après le contrôle du système stable, on passe au contrôle d'un système instable. L'instabilité du système est perçue dans le sens que si la commande dépasse un certain seuil, la sortie du système diverge.

III.5.1. *CONTROLE D'UN SYSTEME MULTIVARIABLE PAR LE RESEAU MLP*

Soit le système MIMO « *multi-input multi-output* » suivant:

$$\begin{cases} x_1(k+1) = \cos(x_2(k)) + u_1(k) \\ x_2(k+1) = \sin(2x_1(k)) + u_2(k) \end{cases} \tag{3.23}$$

Ce système a été identifié dans le chapitre précédent par le modèle suivant:

$$\begin{cases} \hat{x}_1(k+1) = N_1(x_1(k), x_2(k)) + u_1(k) \\ \hat{x}_2(k+1) = N_2(x_1(k), x_2(k)) + u_2(k) \end{cases} \tag{3.24}$$

avec N_1 , $N_2 \in \eta^3_{2,10,10,1}$.

On veut que le système suive le modèle de référence suivant:

$$\begin{bmatrix} x_{1r}(k+1) \\ x_{2r}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{1r}(k) \\ x_{2r}(k) \end{bmatrix} + \begin{bmatrix} 1 - e^{-0.01k} \\ e^{-0.01k} \end{bmatrix} \tag{3.25}$$

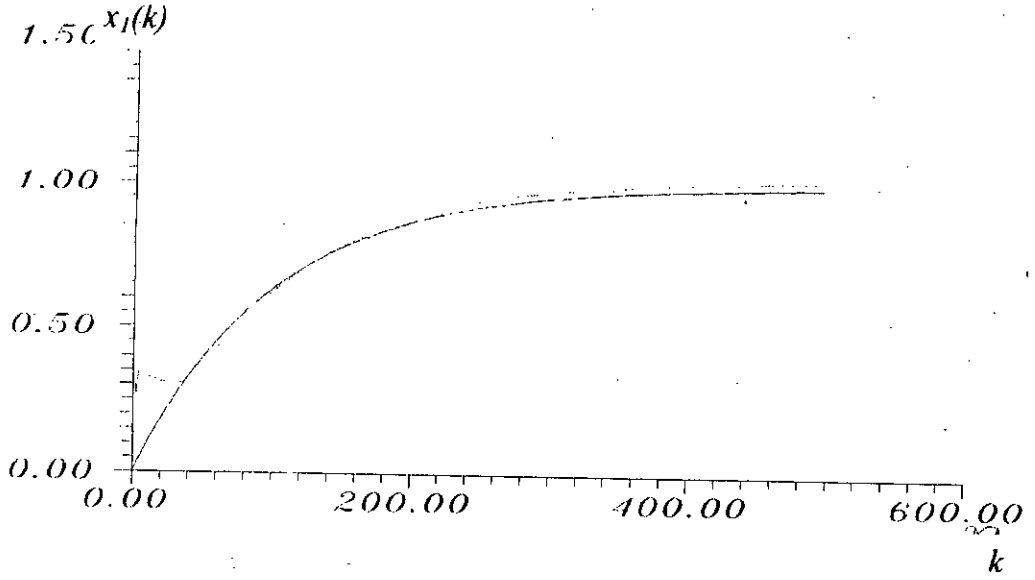
En utilisant le modèle d'identification on peut tirer la commande qui correspond à la trajectoire désirée.

$$\begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} = \begin{bmatrix} x_{1r}(k+1) \\ x_{2r}(k+1) \end{bmatrix} - \begin{bmatrix} N_1(x_1, x_2) \\ N_2(x_1, x_2) \end{bmatrix} \tag{3.26}$$

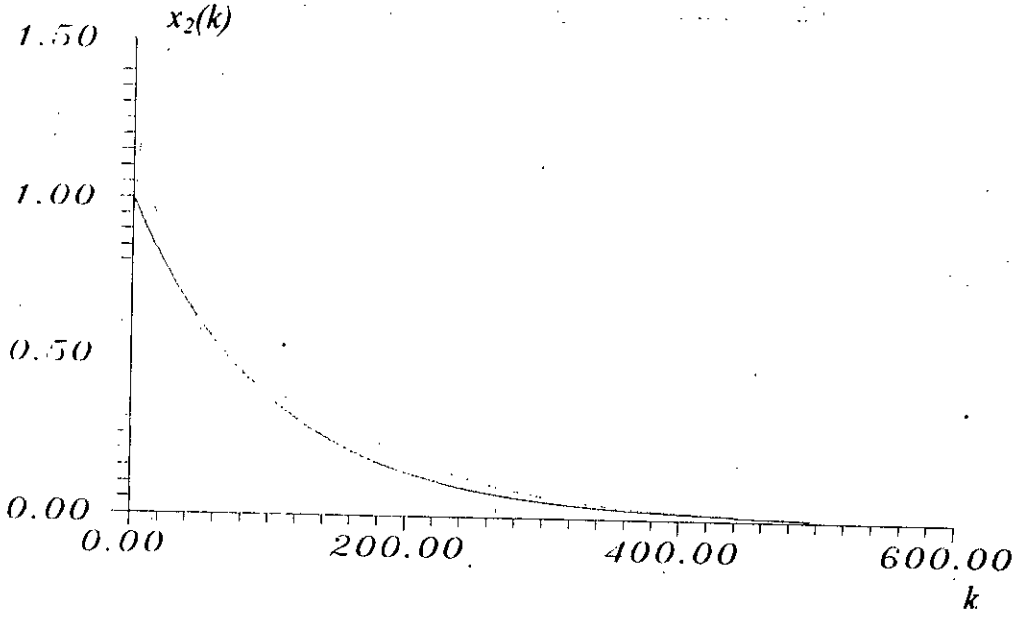
Dans cet exemple on va montré que les réseaux de neurones peuvent être facilement étendus aux systèmes MIMO. D'après la figure (3.14) on voit que le système suit bien le modèle de référence.

La technique utilisée est basée sur l'identification «OFF-LINE» du système, mais en général les systèmes réels sont soumis à des perturbations internes ou externes, d'où la

nécessité de faire l'identification du système d'une façon «ON-LINE» et par la suite ajuster les paramètres du contrôleur.



a) le contrôle de $x_1(k)$



b). contrôle de $x_2(k)$

——— Modèle de référence
..... Système

Figure 3.14. Contrôle d'un système multivariable.

III.5.2. *CONTROLE DES SYSTEMES PAR LE RESEAU* « DRNN »

Soit n_c et n_I le nombre des entrées du DRNC et DRNI respectivement, et soit h_c et h_I le nombre de neurones dans la couche cachée pour le DRNC et le DRNI respectivement. Le choix du nombre de neurones cachées est donné par les équations suivantes :

$$h_c = 2n_c + 1, \quad h_I = 2n_I + 1$$

a). *CONTROLE D'UN SYSTEME STABLE*

Le système à contrôler est donné par l'équation aux différences suivante :

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{3.27}$$

Le modèle de référence est décrit par :

$$y_r(k+1) = 0.6y_r(k) + r(k) \tag{3.28}$$

Avant de commencer la simulation on va tester la stabilité du système comme indiquée dans la figure (3.15). D'après ces résultats on remarque que tant que *u(k)* est bornée le système converge vers un état stable; alors le principe de stabilité « Entrée bornée / Sortie bornée » est vérifié pour une entrée $u(k) \in \{0,1,2\}$.

Dans la partie contrôle, notre objectif est de déterminer une commande *u(k)* pour que $\lim_{k\to\infty} |y(k) - y_r(k)| \to 0$, et ceci en utilisant la stratégie MRAC qui est basée sur les réseaux DRNI et DRNC

Caractéristiques des réseaux

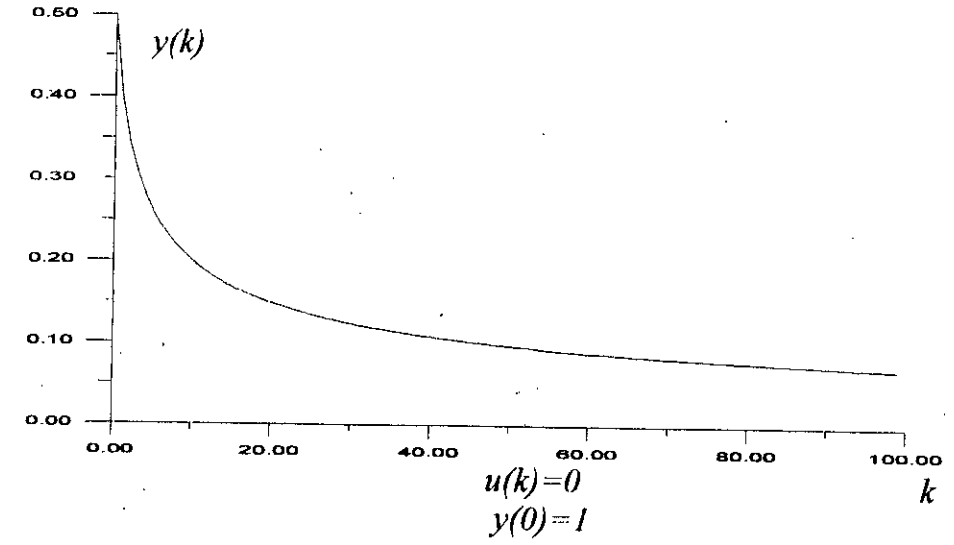
1. Le réseau DRNI

- 2 entrées.
- 5 neurones diagonales « cachées ».
- 1 neurone de sortie.

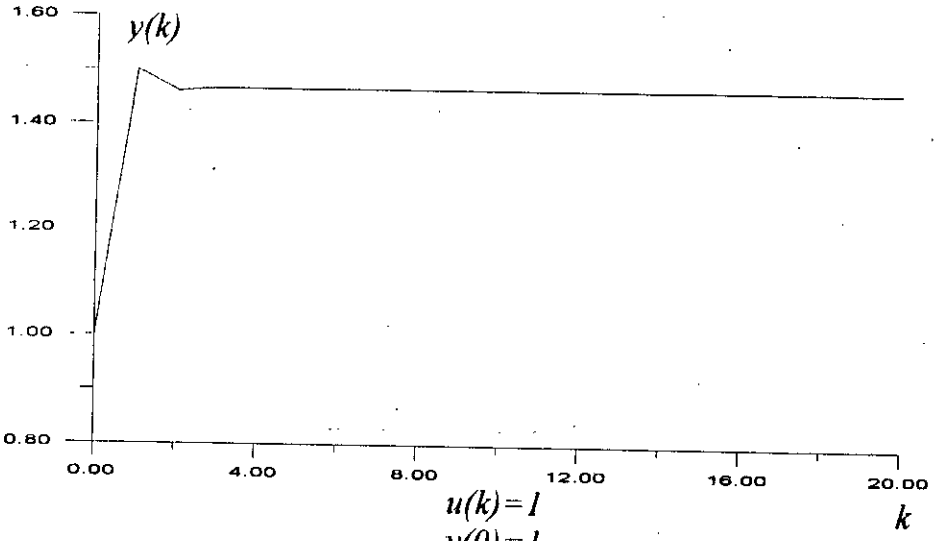
2. Le réseau DRNC

- 3 entrées.*(r(k),u(k-1),y(k-1)).*
- 7 neurones diagonales.
- 1 neurone de sortie.

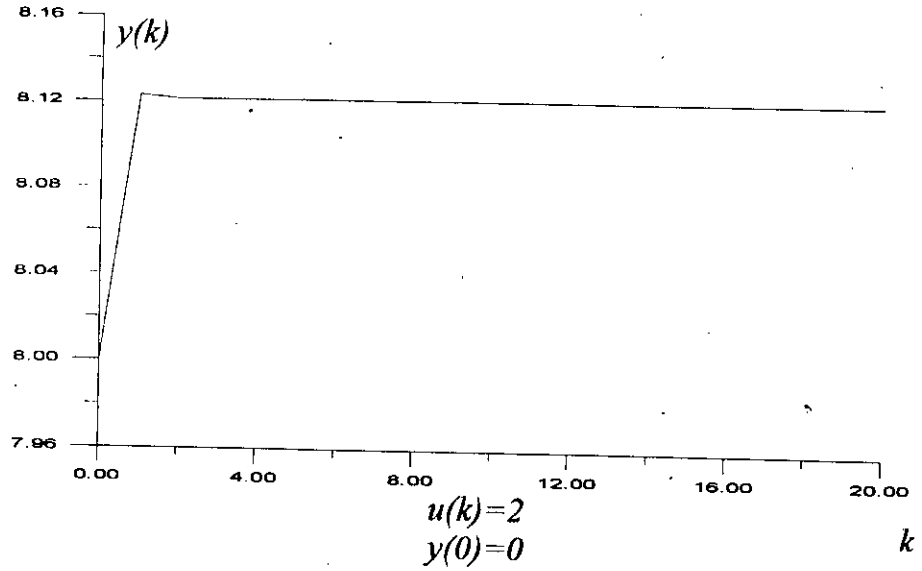
Les résultats obtenus son sur la figure (3.16).



$u(k)=0$
$y(0)=1$



$u(k)=1$
$y(0)=1$



$u(k)=2$
$y(0)=0$

Figure 3.15. Test de stabilité

SIMULATION 1

Dans cette simulation on traite le problème de la poursuite.

On a utilisé la référence $r(k) = \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{10}\right)$.

Après 30000 itérations d'apprentissage les réseaux DRNI et DRNC s'adaptent à cette référence et le système suit exactement le modèle de référence comme indiqué dans la figure (3.16).

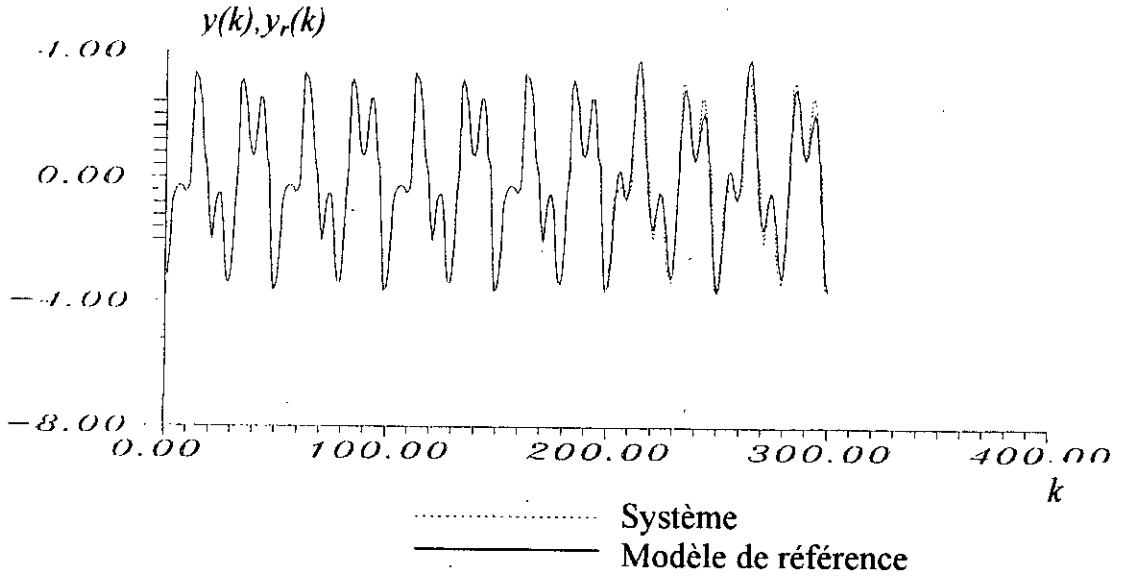
SIMULATION 2

Après l'apprentissage effectué dans la simulation 1, on s'intéresse dans cette partie à faire des tests de robustesse des réseaux DRNC et DRNI vis-à-vis aux variations de la référence. Ces tests sont basés sur deux changements successifs de la référence. Dans le premier changement la référence devient $r(k) = \sin\left(2\pi k/25\right)$, dans le second elle devient un signal carré. D'après les figures (3.17) et (3.18) on remarque que le système suit rapidement les deux changements de la sortie du modèle de référence, et ceci montre la capacité d'adaptation des réseaux DRNI et DRNC dans le contrôle adaptatif avec modèle de référence.

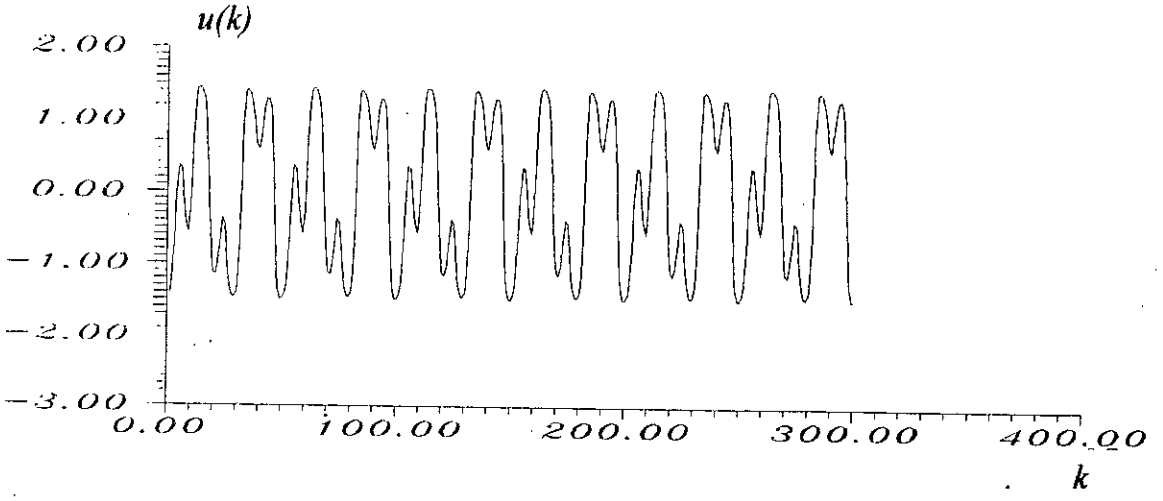
SIMULATION 3

Le but de cette simulation est de tester la capacité de rejection du réseau DRNN des perturbations qui affectent le système. Nous avons commencé par une perturbation constante $v(k)=2$. Sur la figure(3.19) on remarque que les réseaux DRNC et DRNI identifient rapidement cette perturbation et le système rattrape la sortie du modèle de référence. Dans la deuxième partie de cette simulation, nous avons affecté le système par une perturbation $v(k)$ qui suit une loi gaussienne dans l'intervalle [-0.5,+0.5]. Les figures (3.19) et (3.20) montrent le résultat du contrôle. Sur la figure (3.19) on remarque que le contrôleur DRNC arrive à faire tendre le système vers le modèle de référence malgré l'existence de la perturbation constante, et on voit que l'erreur s'annule au bout de 300 itérations.

La figure (3.20) montre le résultat du contrôle du système affecté par une perturbation aléatoire. On constate que le système reste stable et proche de la référence. On remarque aussi que l'erreur ne s'annule pas rapidement et ceci est dû à la nature du bruit additif qui est difficile à apprendre par les réseaux DRNC et DRNI.



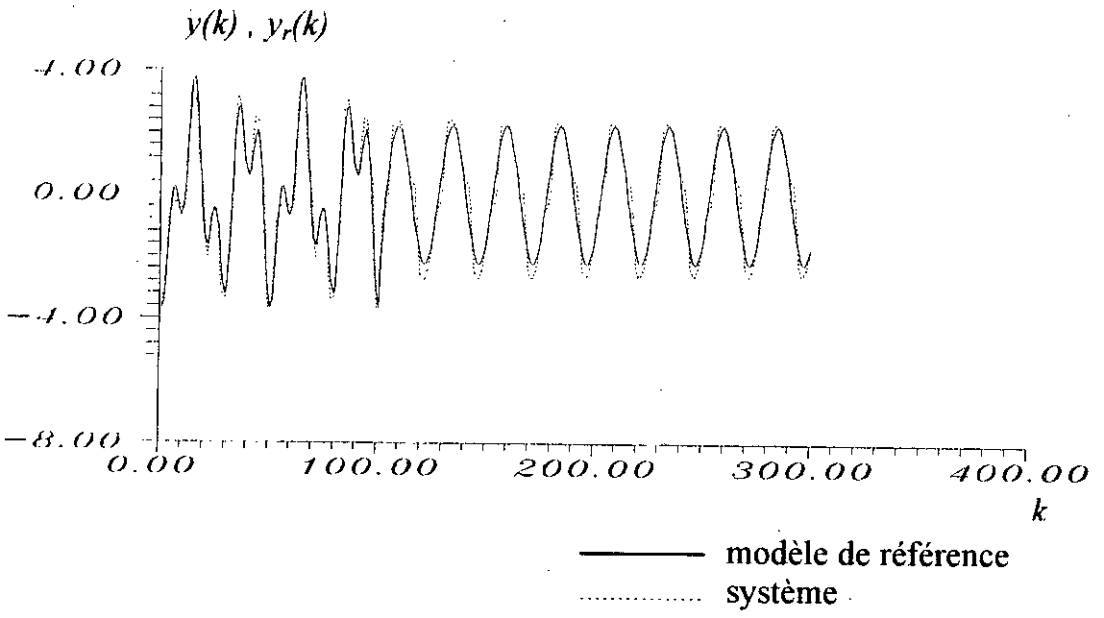
a). La sortie du système et la sortie du modèle de référence



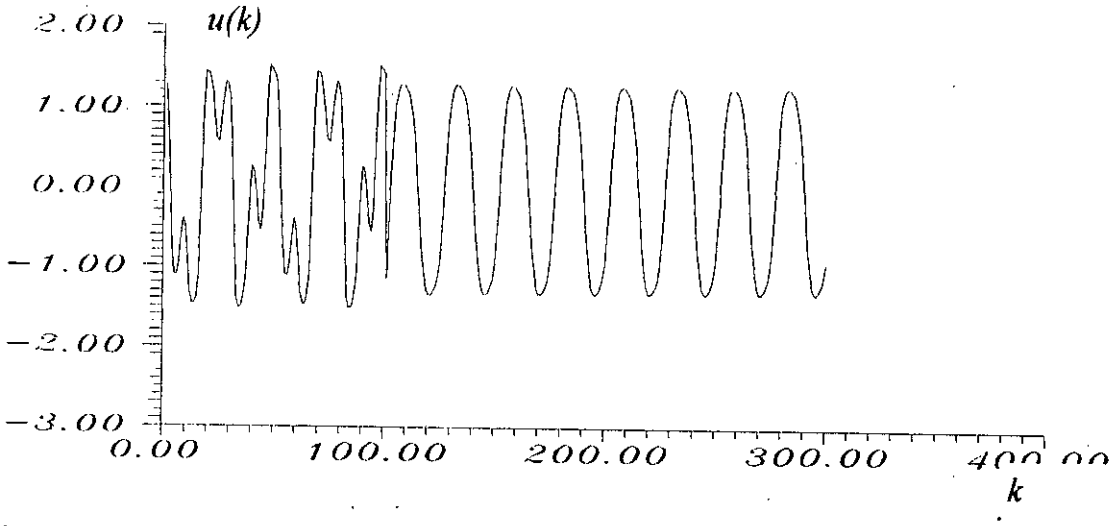
b). La sortie du DRNC, « la commande ».

Figure 3.16. Contrôle du système avec la référence

$$r(k) = \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{10}\right)$$

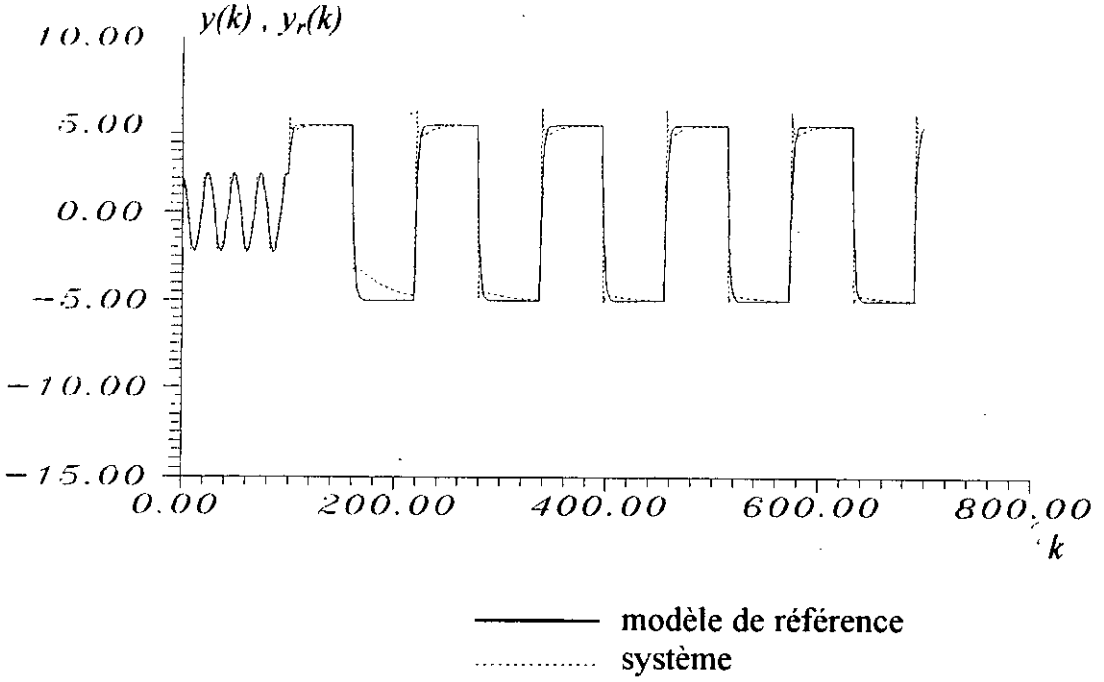


a). La sortie du système et la sortie du modèle de référence

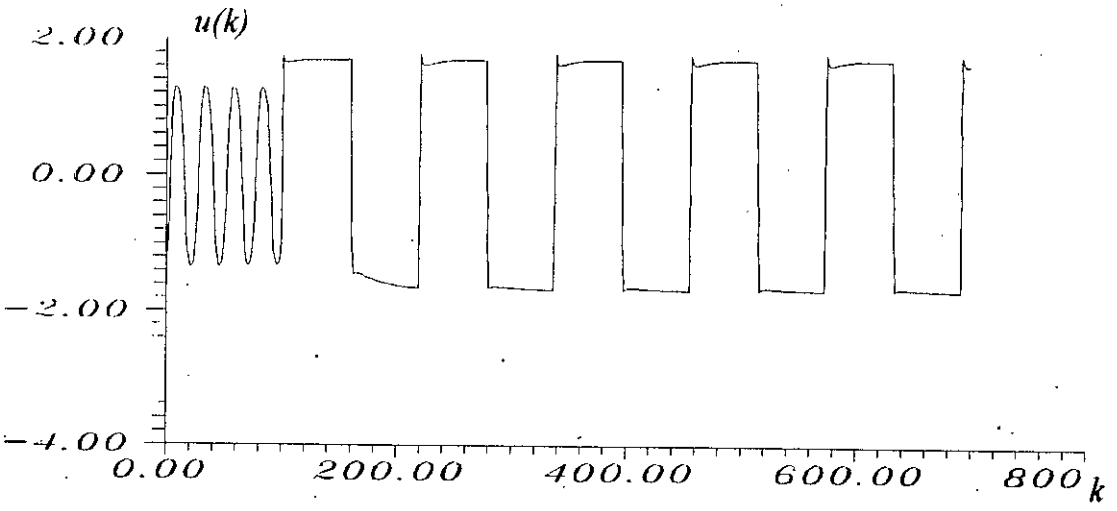


b). *La sortie du DRNC, « la commande ».*

Figure 3.17. Test de robustesse des réseaux DRNC et DRNI face au premier changement de la référence $r(k) = \sin\left(2\pi k / 25\right)$

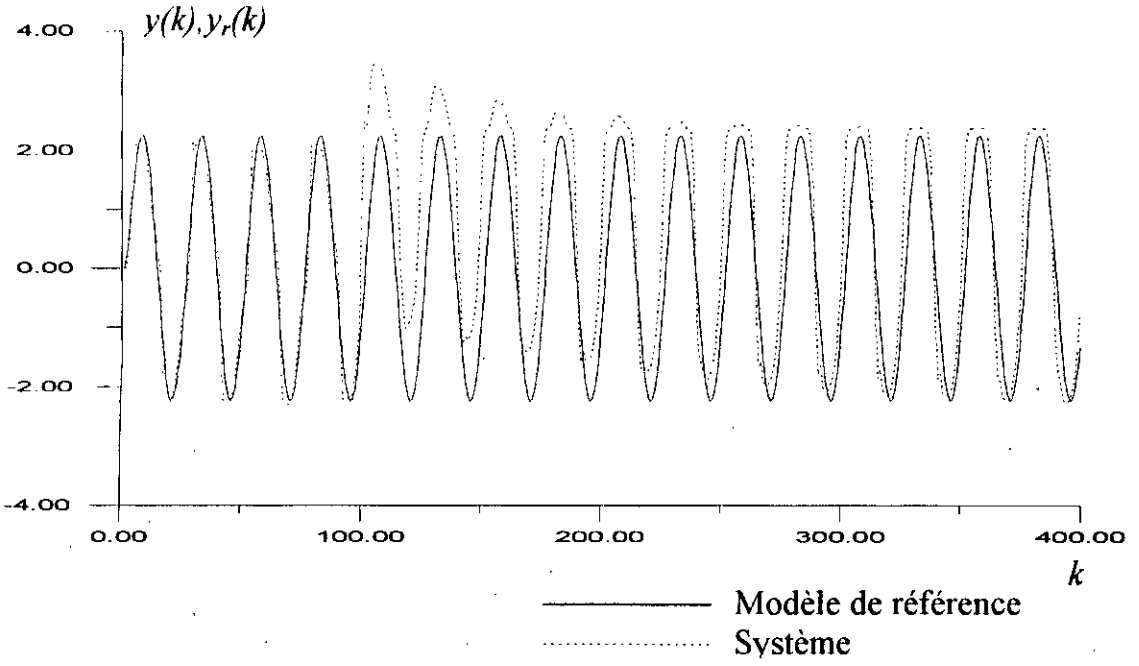


a) La sortie du système contrôlé et le modèle de référence

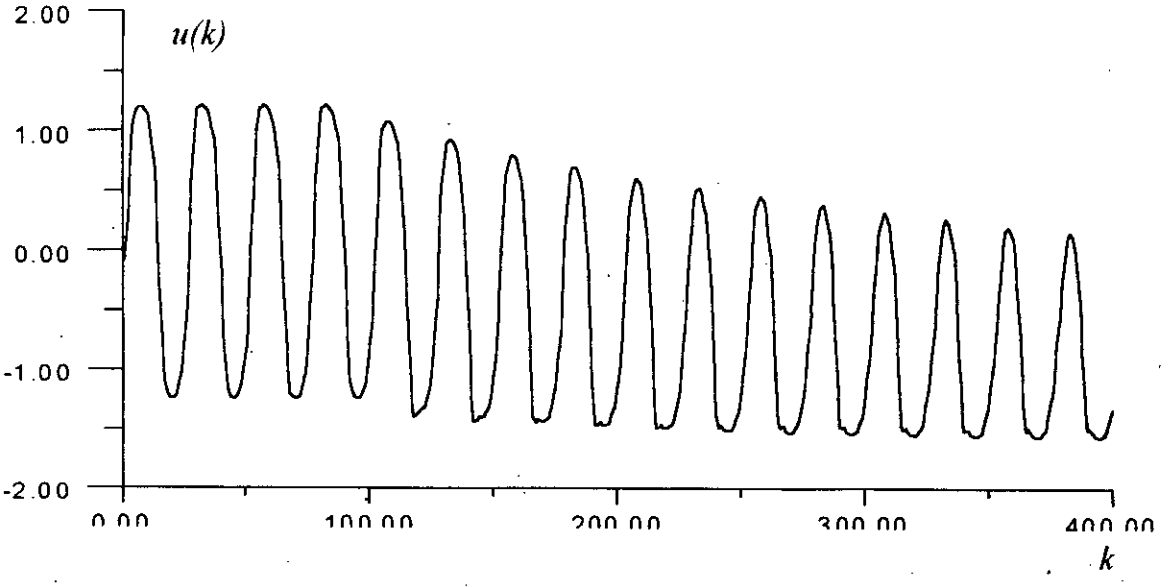


b). *La sortie du DRNC, « la commande ».*

Figure 3.18. Test de robustesse des réseaux DRNC et DRNI face au deuxième changement de la référence $r(k) \in [-2,+2]$.

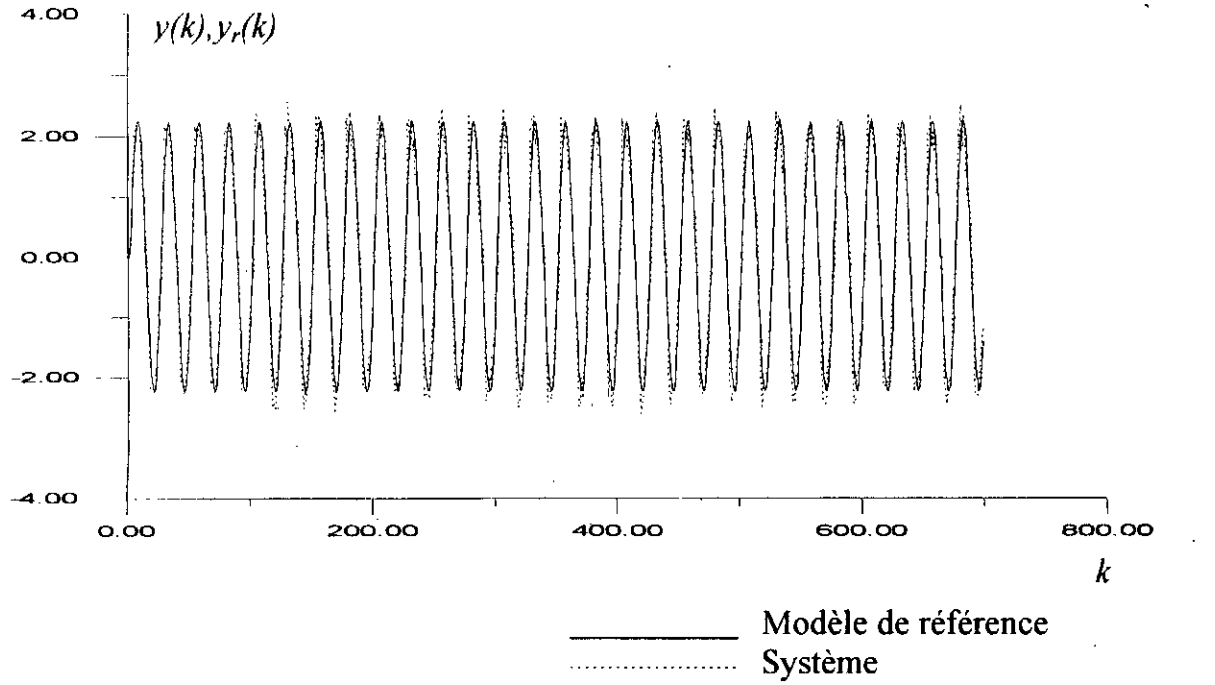


a). La sortie du système et le modèle de référence.

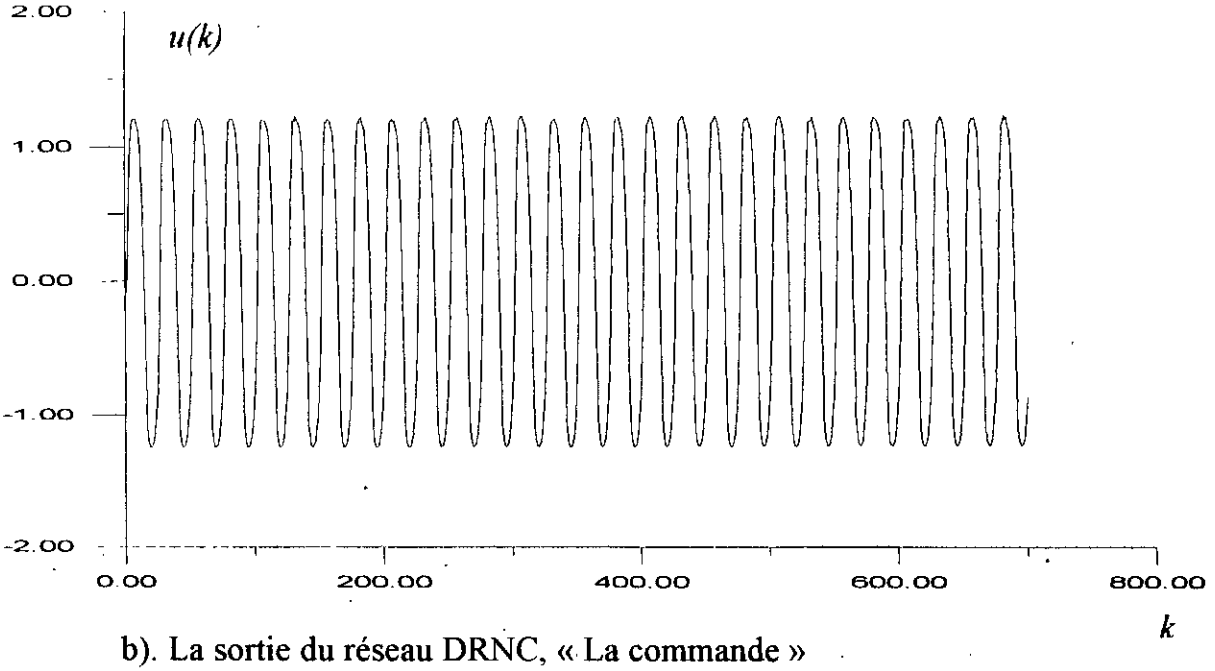


b). La sortie du réseau DRNC, « La commande »

Figure 3.19 Résultat de la commande adaptative avec une perturbation constante.



a). La sortie du système et le modèle de référence.



b). La sortie du réseau DRNC, « La commande »

Figure 3.20 Résultat de la commande adaptative avec une perturbation aléatoire

b).*CONTROLE D'UN SYSTEME INSTABLE*

Le système à contrôler est donné par l'équation suivante :

$$y(k+1) = 0.2y^2(k) + 0.2y(k-1) + 0.2\sin[y(k) + y(k-1)] + 1.2u(k) \qquad (3.29)$$

Le système est instable dans le sens que pour une certaine entrée *u(k)* le système peut diverger. D'après la figure (3.21), on remarque que le système diverge si *u(k)* dépasse la valeur 0.822. Donc pour assurer la stabilité du système durant la procédure de contrôle, il faut que la commande *u(k)* soit inférieur à 0.822 (*u(k)<0.822*). On remarque aussi que la sortie maximale du système que peut délivrer le système dans la région de stabilité est de 2.26.

Dans la procédure de contrôle, notre but est que le système suive le modèle de référence suivant :

$$y_r(k+1) = 0.6y_r(k) + r(k)$$

où

$$r(k) = \beta \sin\left(\frac{2\pi k}{10}\right).$$

Il faut choisir la valeur de β pour que le modèle de référence ne dépasse pas $y_{\max}$.

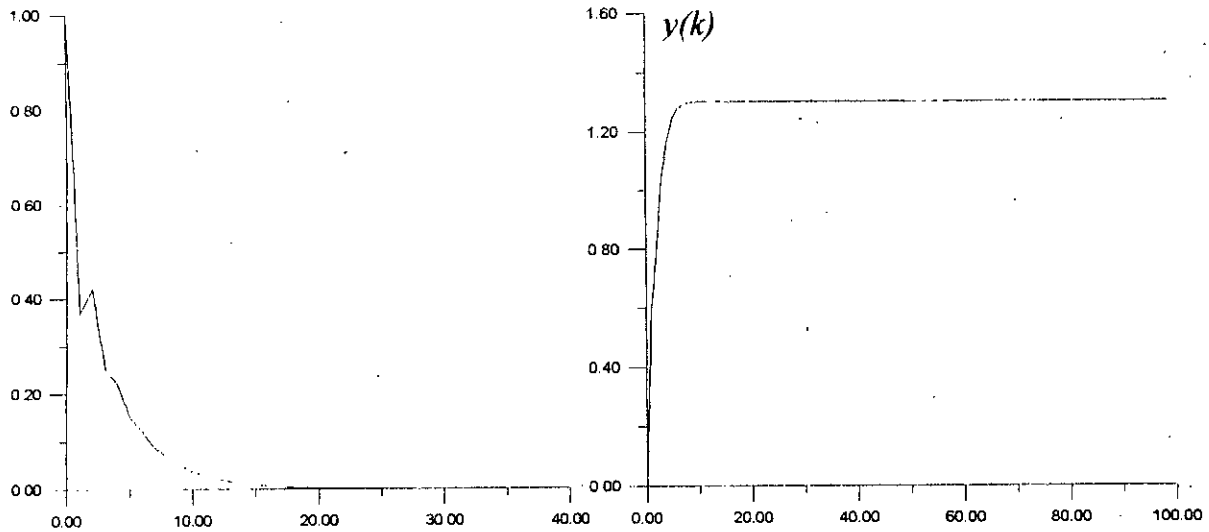
La valeur maximale que peut délivrer le modèle de référence est $y_{r,\max} = \frac{5}{2}\beta$ [12].

Donc pour que $y_{r.max}$ ne dépasse pas y_{max} il faut que $\beta \leq 0.9$.

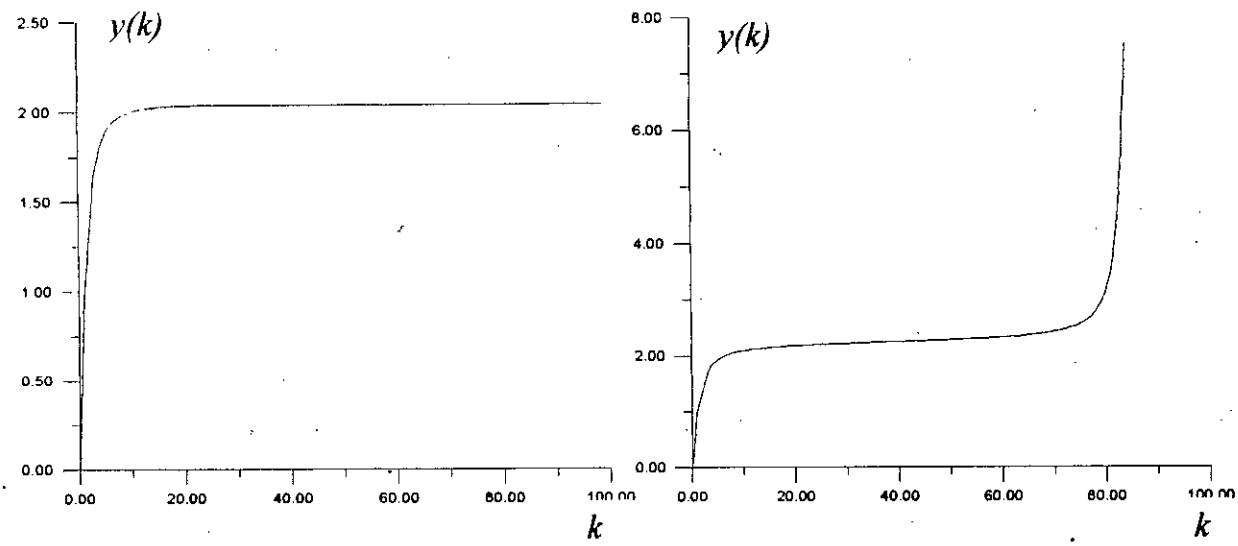
Pour tester la robustesse du réseau DRNN, le système sera contrôlé dans des régions d'instabilité. Pour cela, on posera $\beta = 1$.

D'après la figure (3.22) on remarque que le système suit le modèle de référence avec stabilité malgré l'existence des intervalles où la commande dépasse le seuil de la stabilité (0.822) (figure 3.21. a). On remarque aussi que durant ces intervalles il y a une petite erreur entre la sortie du système et celle du modèle de référence car $y_{r,\max} > y_{\max}$.

Ce résultat a été obtenu après plusieurs tests sur la référence *r(k)*, durant lesquels nous avons essayé d'augmenter la région d'instabilité, par exemple avec une référence $r(k) = \sin\left(\frac{2\pi k}{100}\right)$, une divergence du système a eu lieu.

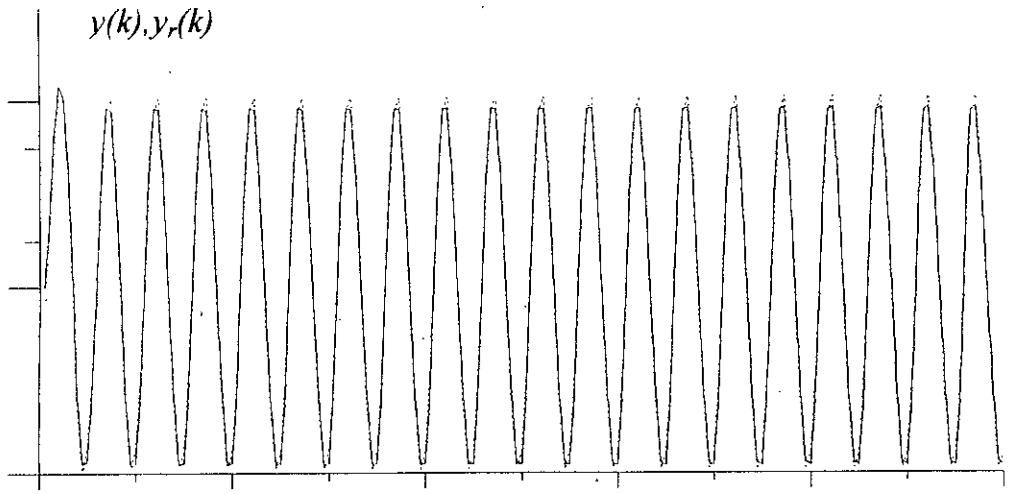


a). $y(0)=0$, $y(1)=1$, $u(k)=0$ b). $y(0)=0$, $y(1)=0$, $u(k)=0.5$

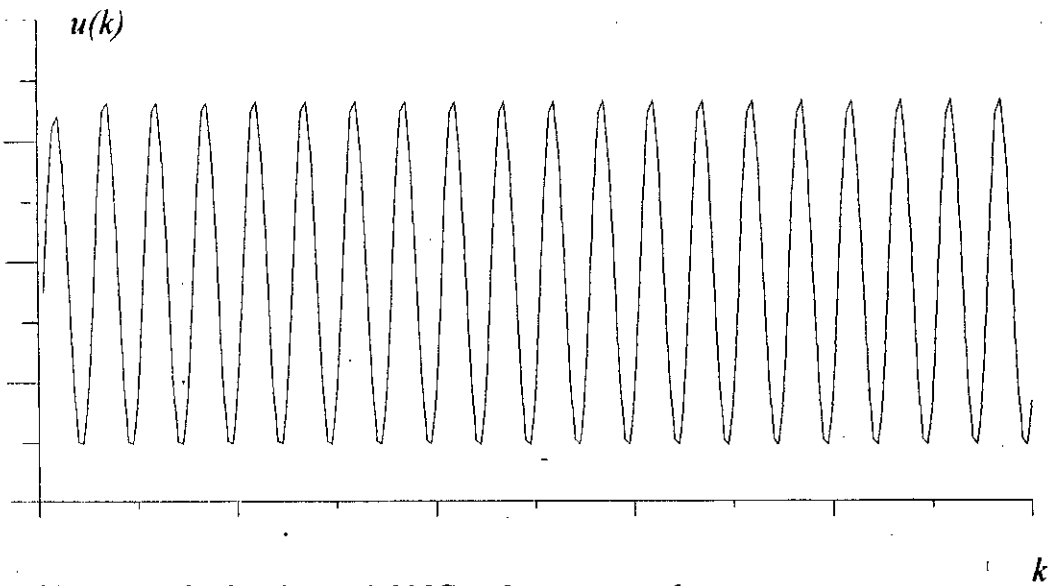


c). $y(0)=0$, $y(1)=0$, $u(k)=0.8$ d). $y(0)=0$, $y(1)=0$, $u(k)=0.822$

Figure 3.21 Test de stabilité



a). La sortie du système et le modèle de référence



b). La sortie du réseau DRNC, « La commande »

——— Modèle de référence
..... Système

Figure 3.22. Contrôle du système instable.

III.6. CONCLUSION

Dans ce chapitre nous avons montré le principe de contrôle adaptatif par les réseaux de neurones. On a présenté quelques schémas de commande basés sur les réseaux de neurones qui s'appliquent à des systèmes dans un environnement déterministe ou stochastique.

Des schémas directs et indirects on été présentés.

- L'approche directe qui consiste, rappelons le, à ajuster directement les poids du contrôleur neuronal. Cette approche, qui permet de réduire le temps de calcul souffre du fait qu'en général la sensibilité du système est inconnue et que celle-ci est utilisée dans la phase apprentissage du réseau de neurones contrôleur. Dans plusieurs travaux, cette sensibilité est tout simplement ignorée, dans d'autres cas, on utilise seulement le changement de signe dans la réponse du système comme sensibilité [11].
- L'approche indirecte consiste, rappelons-le, à ajuster les poids d'un réseau de neurones identificateur qui joue le rôle d'un prédicteur adaptatif de la sortie du système, puis à ajuster les poids du contrôleur. Dans cette approche, nous avons étudié une architecture de réseau de neurones récurrents appelé DRNN, « diagonal recurrent neural network » [12].

Dans la partie simulation, on a commencé par le cas où on a identifié «OFF-LINE» le système à contrôler ensuite on a tiré la commande qui permet au système de suivre le modèle de référence. Cette méthode est valable seulement si le système est invariant dans le temps. Après cette simulation on a testé la robustesse du réseau DRNN aux variations de la référence et aux perturbations qui affectent le système. Enfin nous avons utilisé le réseau DRNN dans le contrôle d'un système instable. Les résultats obtenus sont encourageants et illustre la robustesse du réseau DRNN dans le contrôle adaptatif.

En conclusion, on peut dire que les réseaux de neurones permettent de commander des systèmes non linéaires complexes dont le traitement se heurte encore aujourd'hui à des difficultés d'ordre pratique et théorique.

CHAPITRE IV

CONTROLE D'UNE RUCHE APICOLE PAR LES RESEAUX DE NEURONES

IV.1. INTRODUCTION

Historiquement,l'abeille été domestiquée dès l'antiquité (BABYLONIEN, EGYPTIENS,...etc) à partir des essaims sauvages pour la production du miel. Le miel a été utilisé dans la médecine traditionnelle comme remède contre plusieurs maladies. Ceci a donné naissance à l'APICULTURE.

La ruche apicole est considérée comme un système de la nature des plus complexes attirant ainsi l'attention des chercheurs, non seulement les biologistes mais aussi les mathématiciens. Elle est l'une des ressources naturelles qu'il convient d'exploiter afin d'optimiser son rendement.

Un des problèmes majeurs ayant un effet important sur le rendement de la ruche est la diminution des entrées en pollen et en nectar dans les régions à miellée courte, et espacées. Cela entraîne souvent un manque en provisions si des précautions ne sont pas prisent à temps. Pour pallier à ce problème, les apiculteurs s'en remettent à leurs expériences pour faire un nourissement artificiel contenant des produits nutritifs qui sont importants pour la survie de la colonie d'abeilles. Seulement ce nourissement engendre quelques inconvénients :

- le premier danger réside dans l'apparition de maladies du couvain telles que:
  les loques européenne et américaine et par l'épuisement des provisions de pollen.
- le deuxième danger réside dans la diminution de la taille de la population par l'essaimage naturel.

Donc il est nécessaire de contrôler la productivité de la ruche apicole sur une base scientifique. Pour accomplir cet objectif, il est impératif de déterminer un modèle mathématique. La modélisation et le contrôle de la ruche apicole ont été déjà entamé dans [14],[15],[16].

Dans ce travail nous avons un cahier des charges qui contient deux objectifs.

Objectif 1:

La ruche N°1,est nourrie artificiellement « Contrôle par les réseau de neurones » durant les périodes où il y a un manque en provisions, Pour augmenter le nombre d'abeilles afin de les préparer à une miellée (augmentation de la production du miel pur). L'apiculteur nous a confié un modèle de référence durant les périodes [0,60] (Janvier,Fevrier) et [150,200] (juin, juillet) selon son expérience.

Objectif 2:

Cet objectif consiste à augmenter la production de la cire dans la ruche N°2. Pour atteindre cet objectif, le nourrissement est administré durant toute l'année.

- durant [0,60] jours, on veut que la cire soit de 3 kg.
- à la fin de l'année, on veut une quantité de 10 kg de cire.

Dans le présent chapitre on commence par traiter la biologie de l'abeille et une description générale de la ruche apicole, puis on présente un résume des travaux réalisés dans [15] et [16]. Ensuite nous aborderons notre travail qui consiste à faire l'identification de la ruche en utilisant les réseaux de neurones et enfin le contrôle est effectué pour aboutir aux objectifs désirés. On termine ce chapitre par la discussion des résultats obtenus et une conclusion.

IV.2. BIOLOGIE DE L'ABEILLE

Une colonie d'abeilles est constituée d'une reine, quelques dizaines de milliers d'ouvrières et de quelques milliers de faux bourdons.

- la reine peut vivre de 5 à 7 ans. Elle peut pondre jusqu'à 3000 oeufs par jour. Elle contrôle minutieusement toutes les activités de la ruche par la sécrétion de produits chimiques (hormones).
- les faux bourdons ne jouent essentiellement aucun rôle déterminant autre que la fécondation d'une jeune reine.
- les ouvrières exercent toutes les activités au sein de la ruche sauf la reproduction.

Dés sa sortie de l'alvéole, l'ouvrière est:

- nettoyeuse pendant 02 jours.
- nourrice(production de la gelée royale) pendant 10 jours.
- cirière(production de cire et la construction des alvéoles) pendant 06 jours.
- gardienne et ventileuse pendant 03 jours.
- butineuse(récolte le nectar et le pollen) jusqu'à sa mort.

La vie d'une ouvrière est de 30 à 45 jours(prolongée jusqu'à 06 mois pendant l'inactivité ou durant l'orphelinage).

IV.3. DESCRIPTION GENERALE DE LA RUCHE APICOLE

Il existe plusieurs types de ruches apicoles portant le nom de leurs inventeurs. Nous citons quelques unes :

-Ruche traditionnelle en terre cuite, en paille, ou dans des morceaux de tronc d'arbre.

-Ruche LONGSTROTH.

-Ruche DADANT.

-Ruche PROKOPOVITCH.

Les trois dernières sont en bois avec des caractéristiques bien spécifiques. Leur propriété commune est qu'elles sont à cadres mobiles; ceci est d'une importance primordiale, car l'intervention de l'homme est facilitée d'une manière considérable.

Une ruche est composée du corps de la ruche *« élevage de couvain, stockage des réserves de miel et de pollen »* et d'une ou plusieurs hausses pour la récolte du miel avec un nourrisseur au dessus.

Il faut remarquer que la ruche apicole la plus répandue en Algérie est celle de LONGSTROTH.

IV.4. DESCRIPTION DE L'AUTOMATICIEN

La ruche apicole est un système de la nature les plus automatisés car la température, le taux d'humidité, ainsi que le cycle biologique sont minutieusement contrôlés.

Dans notre étude, la ruche apicole est considéré comme une boîte noire contenant une colonie d'abeilles placée dans un environnement donné (fig.4.1).

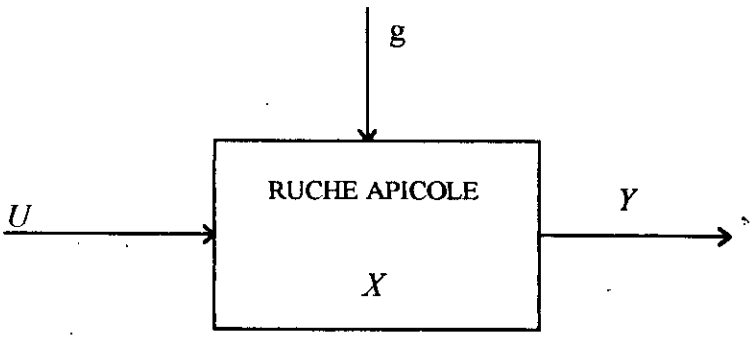


Figure 4.1 Schéma d'une RUCHE APICOLE comme système

U : Le vecteur de commande à travers le nourrisseur.

Le système peut être commandé en ajoutant une quantité de sirop (sucre dilué dans l'eau) auquel des matières protéiniques (pollen, farine de seigle, ou maïs ...etc) sont ajoutées, du candi (sucre cristallisé mélangé à du miel et de l'eau).

X : Le vecteur des variables d'état du système, qui est constitué du nombre d'abeilles, de la quantité du miel, de la quantité de pollen (sous forme de pain d'abeilles) et de la quantité de cire (sous forme de cadres bâtis).

Y : Le vecteur des sorties qui peut être constitué du miel naturel, du pollen, de la propolis, de la cire ... etc.

g : Le vecteur des perturbations du système qui est constitué du nectar et du pollen récolté par des abeilles butineuses. Ce vecteur dépend de la population d'abeilles, du temps et de nombreux paramètres caractérisant l'environnement de la ruche.

IV.5. MODELISATION DE LA RUCHE APICOLE [14]

IV.5.1. *HYPOTHESES DE TRAVAIL*

Des hypothèses simplificatrices réalistes ont été émises pour obtenir un modèle mathématique simple et utile.

-Jeune reine sélectionnée.

-Traitement systématique de toutes les maladies.

-Lutte contre l'essaimage naturel.

-Abondance de l'eau au voisinage de la ruche.

-L'eau utilisée par les abeilles n'est pas modélisée.

-La production de la gelée royale n'est pas modélisée.

-Production du miel naturel, de la cire et du pollen uniquement.

-Corps de la ruche bâti.

-Abondance de matières premières « *nectar et pollen* » au voisinage de la ruche.

- climat tempéré.

- Existence de miellées multiples.

- Récolte du miel non entreprise.

Ces hypothèses permettent d'obtenir le plus simple modèle mathématique. En pratique elles peuvent être réalisées avec des interventions ponctuelles de l'apiculteur.

La collecte des données a été effectuée durant l'années 1992 en mesurant les quatre variables d'états x_i ainsi que les deux entrées g_2 *et* g_3 à savoir :

x_1(k): Le nombre d'abeilles

x_2(k): La quantité de miel.

x_3(k): La quantité du pollen.

x_4(k): La quantité de cire.

g_2(k): entrée en nectar

g_3(k): entrée en pollen.

Il faut remarqué que ces données sont très approximatives vue la procédure suivie, à savoir:

-Le nombre d'abeilles est estimé en comptant le nombre de cadres couverts d'abeilles multiplié par 3000.

-La quantité du miel est estimée en comptant le nombre de cadres sans couvains multiplié par 02 kg et le nombre cadres avec couvain multiplié par 01 kg.

-La quantité de pollen est estimée en comptant le nombre de cadres contenant du pain d'abeilles multiplié par 0,4 kg.

-Les mesures ont été prise tout les 10 jours afin de ne pas refroidir le couvain subitement et ainsi compromettre le développement de la colonie d'abeilles.

Les données expérimentales sont résumées dans le tableau 4.1. et ses variations durant toute l'année sont illustrées dans la figure (4.2).

t	$x_1(t)$	$x_2(t)$	$x_3(t)$	$x_4(t)$	$g_2(t)$	$g_3(t)$
0	10	10	0.25	1	0.01	0.01
10	10	10	0.25	1	0.05	0.02
20	10	10	0.25	1	0.10	0.04
30	13	10	0.25	1	0.15	0.06
40	20	10	0.5	1	0.25	0.09
50	26	10	0.75	1	0.35	0.14
60	36	10	1.25	1	0.50	0.20
70	46	13	2.25	1.25	0.70	0.30
80	56	17	3.25	1.50	1.05	0.30
90	73	20	5	2.25	1.50	0.30
100	100	25	5	3	1.50	0.30
110	100	32	5	3	1.50	0.30
120	100	50	5	3	1.50	0.30
130	100	50	5	3	1.05	0.20
140	100	40	4	3	0.65	0.12
150	93	30	3	3	0.20	0.02
160	75	26.5	2.75	3	0.10	0.01
170	60	25	2.60	3	0.40	0.06
180	50	25	2.50	3	0.85	0.15
190	55	33.5	2.80	3.08	1.55	0.28
200	67	47.5	3.20	3.15	2.00	0.19
210	80	60	3.50	3.25	1.15	0.05
220	80	60	3.50	3.25	0.13	0.02
230	70	53.5	2.85	3.25	0.07	0.02
240	59	46	2.25	3.25	0.05	0.01
250	47	37	1.85	3.25	0.04	0.01
260	42	28.5	1.50	3.25	0.04	0.01
270	37	21.5	1.12	3.25	0.03	0.01
280	31	15	0.80	3.25	0.03	0.01
290	26	11	0.55	3.25	0.03	0.01
300	20	7	0.45	3.25	0.03	0.01
310	18.3	7.5	0.40	3.25	0.03	0.01
320	16.7	6.5	0.38	3.25	0.03	0.01
330	15	5.5	0.35	3.25	0.03	0.01
340	13.3	5.3	0.30	3.25	0.03	0.01
350	11.7	5	0.28	3.25	0.03	0.01
360	10	5	0.25	3.25	0.03	0.01
Jours	Milliers	kg	kg	kg	kg/jour	kg / jour

Tableau 4.1 : données expérimentales sur une année (1992).

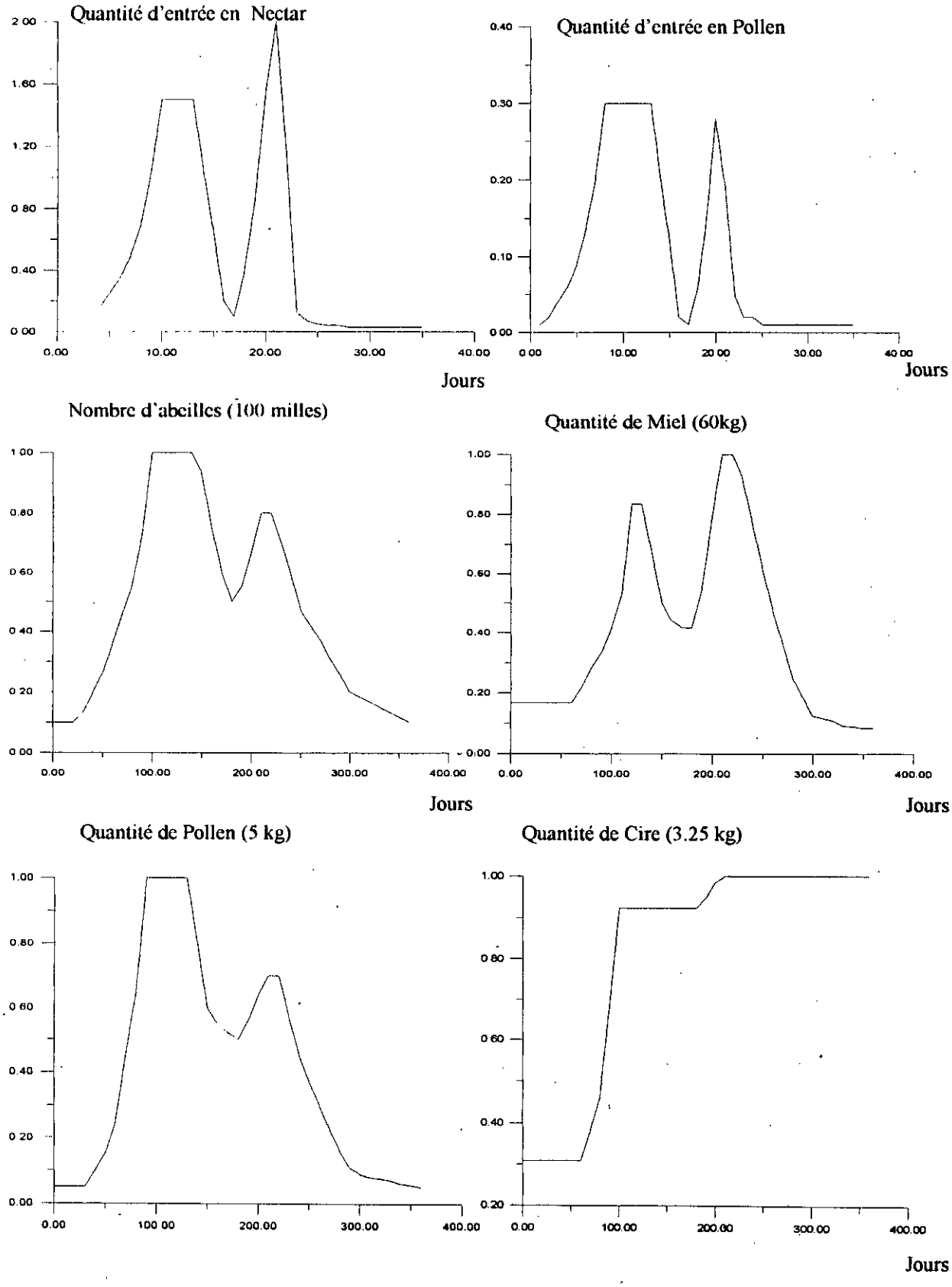


Figure 4.2 . Les variations des données expérimentales de la RUCHE APICOLE sur une année (l'année 1992).

IV.6.IDENTIFICATION DE LA RUCHE APICOLE PAR MODELES MATHEMATIQUES

Les travaux réalisés dans [14],[15],[16] utilisent des modèles paramétriques pour faire l'identification de la ruche. Ces modèles sont les modèles linéaires, le modèle général bilinéaire,et le modèle LOTKA- VOLTERRA.

IV.6.1. *ADOPTION DU MODELE GENERAL BILINEAIRE*

La formulation du modèle général bilinéaire dans le cas continu sera :

$$\begin{cases} \dot{x}_1(t) = \sum_{j=1}^{4} x_j(t)\left[a_{1j} + \sum_{L=j}^{4} b_{jL}x_L(t)\right] \\ \dot{x}_2(t) = \sum_{j=1}^{4} x_j(t)\left[a_{2j} + \sum_{L=j}^{4} c_{jL}x_L(t)\right] + \beta_2 g_2(t) \\ \dot{x}_3(t) = \sum_{j=1}^{4} x_j(t)\left[a_{3j} + \sum_{L=j}^{4} d_{jL}x_L(t)\right] + \beta_3 g_3(t) \\ \dot{x}_4(t) = \sum_{j=1}^{4} x_j(t)\left[a_{4j} + \sum_{L=j}^{4} e_{jL}x_L(t)\right] \end{cases} \quad (4.1)$$

Les paramètres de ce modèle ont été estimés par la méthode des moindres carrées.
L'annexe (I) montre la superposition des résultats du modèle général bilinéaire et les données expérimentales.

IV.6.2. *ADOPTION DU MODELE LOTKA -VOLTERRA*

Le modèle LOTKA-VOLTERRA est un cas particulier du modèle précédent où quelques coefficients sont supposés nuls.

$$\dot{x}_i(t) = \sum_{j=1}^{4} x_j(t)\left[a_{ij} + \sum_{L=j}^{4} b_{jL}x_L(t)\right] \quad (4.2)$$

où $b_{ii}=0$, $i=1,2,3,4$

L'annexe (II) montre la superposition des résultats du modèle LOTKA-VOLTERRA et les données expérimentales.

IV.6.3. *ADOPTION DU MODELE LINEAIRE*

Ce modèle est aussi un cas particulier du modèle bilinéaire car tous les coefficients des termes bilinéaires sont supposés nuls. Le modèle s'écrit comme suit :

$$\dot{X}(t) = AX(t) + \beta\, g(t) \qquad (4.3)$$

Les résultats de ce modèle sont illustrés par L'annexe (III)

IV.7. CONTROLE DE LA RUCHE APICOLE PAR LA COMMANDE OPTIMALE [16]

Le but était le contrôle de la ruche durant approximativement deux mois, du premier septembre (correspondant à l'instant initial t_0) au premier novembre (correspondant à l'instant final t_1).

L'objectif adopté est d'avoir un nombre d'abeilles avoisinant un nombre ν à l'instant t_1 en minimisant l'énergie de commande. Après application de différents algorithmes pour satisfaire les conditions d'optimalité de « PONTRIAGUIN », les résultats obtenus sont présentés dans l'annexe (IV).

IV.8. CONTROLE DE LA RUCHE APICOLE PAR LA LOGIQUE FLOUE

Dans ce travail, le problème de nourissement artificiel a été interprété comme un problème de commande floue dont les règles sont établies à partir d'un expert bien qualifié dans son domaine qui est l'apiculteur, consistant à déterminer un nourrissement approprié pour atteindre l'objectif désiré dans le but d'augmenter la production du
miel [15].

La simulation a été faite sur trois périodes de l'année :

- La première est de [0,60] jours.
- La deuxième période de [120,180] jours.
- La troisième période de [240,360] jours.

Les autres périodes de l'année le système est autonome.

Les résultats du contrôle flou sont illustréspar L'annexe (V).

IV.9. DISCUSSION DES RESULTATS OBTENUS

Après l'étude des travaux effectués dans [15] et [16], nous avons remarqué les inconvénients suivants :

- Ils ont utilisé les modèles mathématiques pour identifier la ruche apicole. Les paramètres de ces modèles sont estimés par la méthode des moindres carrées, qui donne un estimateur non biaisé à condition que le bruit additif en sortie du système soit un bruit blanc [19]. Mais en réalité le bruit blanc n'existe pas (énergie infinie), donc cette méthode n'est pas toujours bonne en pratique. Puisque notre système est affecté par un bruit qui est corrélé, alors on peut conclure que les modèles mathématiques obtenus sont imprécis, à cause de l'insuffisance du nombre des données utilisées durant l'identification (36 données seulement) et la nature du bruit affectant le système.
- Nous savons qu'en pratique, il peut y avoir des variations internes ou externes affectant le système, donc les paramètres de ces modèles peuvent changer dans le temps d' où le danger d'utilisation des lois de commande qui sont tirées directement à partir du modèle ayant des paramètres différent de ceux du système après la variation.
- Dans la partie contrôle de ces travaux, on remarque qu'ils ne peuvent pas contrôler la trajectoire du comportement de la ruche durant la période de nourrissement mais ils fixent seulement l'état initial de la ruche et son état désirée à la fin de la période.

Après la discussion de ces résultats, on peut dire que les capacités des réseaux de neurones présentés dans ce mémoire peuvent donner des issues nouvelles au problèmes rencontrés dans [15] et [16] car d'une part on peut identifier la ruche sans prendre en compte la nature du bruit qui l'affecte, d'autre part on peut la contrôler en utilisant un modèle de référence (ruche apicole idéale) durant la période de nourrissement.

IV.10. IDENTIFICATION DE LA RUCHE APICOLE PAR LES RESEAUX DE NEURONES

On suppose que notre système est régi par les équations discrètes suivantes :

$$\begin{cases} x_1(k+1) = f_1(X(k)) \\ x_2(k+1) = f_2\big(X(k)\big) + g_2(k) \\ x_3(k+1) = f_3\big(X(k)\big) + g_3(k) \\ x_4(k+1) = f_4(X(k)) \end{cases} \tag{4.4}$$

Le modèle d'identification est:

$$\begin{cases} \hat{x}_1(k+1) = N_1(X(k)) \\ \hat{x}_2(k+1) = N_2\big(X(k)\big) + g_2(k) \\ \hat{x}_3(k+1) = N_3\big(X(k)\big) + g_3(k) \\ \hat{x}_4(k+1) = N_4(X(k)) \end{cases} \tag{4.5}$$

où les N_i sont des réseaux de neurones à multicouches appartenant à la classe $\eta^4_{4,10,10,1}$.

En utilisant le principe de l'identification des systèmes, on va faire l'apprentissage des réseaux de neurones N_i (fig.4.3).

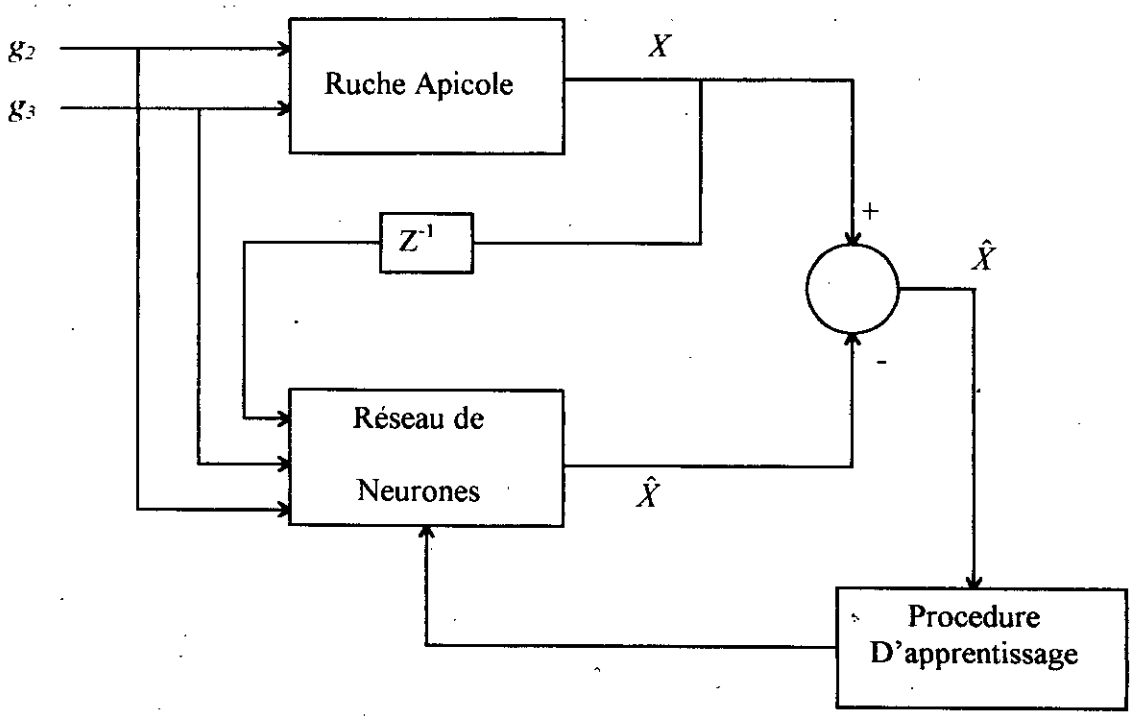


Figure 4.3. Identification d'une ruche apicole par les réseaux de neurones

Après apprentissage des réseaux de neurones nous avons obtenu les résultats illustrés par la figure (4.4) .On remarque une nette amélioration de l'identification de la ruche apicole par rapport aux résultats illustrés dans les annexes I,II et III, car les sorties des réseaux de neurones présentent moins d'écart avec les données empiriques. Il faut noter que si les données expérimentales sont recueillies d'une façon continue à l'aide des capteurs électroniques, le modèle neuronal donnerait des résultats meilleurs.

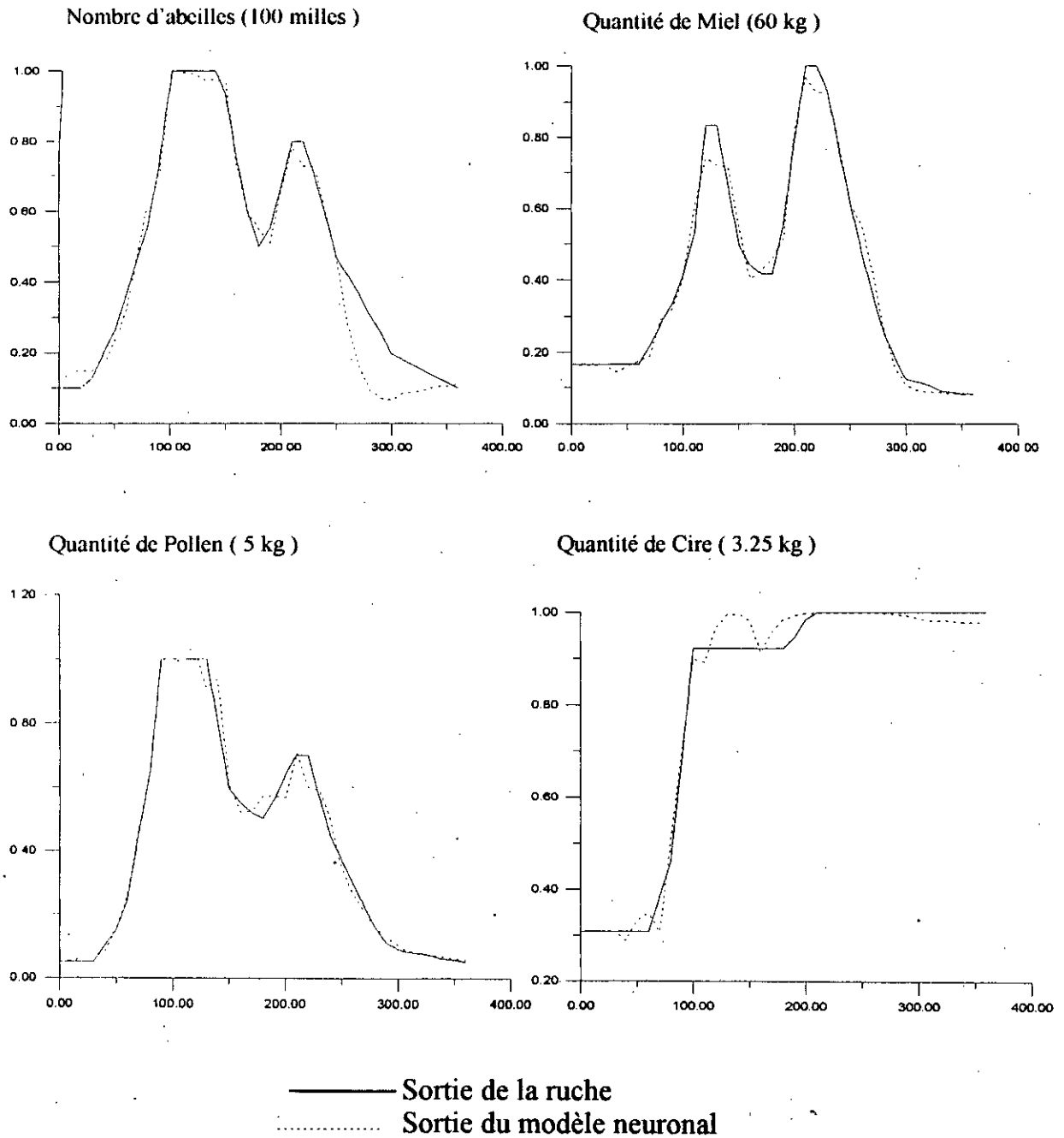


Figure 4.4. Superposition des données expérimentales avec les sorties des réseaux de neurones.

Rapport Signal / Bruit	-
Nombre de données d'apprentissage	36
Le pas d'apprentissage	0.1
Le nombre d'itérations	36200
temps d'apprentissage	2 minutes,8 secondes, 64 entième
Matériel utilisé	micro-ordinateur DX2 66 MHZ

Tableau 4.4. Caractéristique de la simulation.

IV.11. LE CONTROLE DE LA RUCHE AVEC MODELE DE REFERENCE

IV.11.1. *CONTROLE DE LA RUCHE N°1*

Cette ruche est destinée à la production du miel pur. Notre but est d'augmenter la quantité du miel pur en jouant sur le nombre d'abeilles durant les périodes ou il y a le manque en provisions. Pour cela, le système dispose d'un nourrisseur à travers lequel des produits nutritifs sont dispensés pour la survie de la colonie d'abeilles. Ce nourrissement est nécessaire pour que cette colonie soit assez peuplée avant le début d'une miellée pour la production du miel pur.

Le problème de nourrissement peut être interprété comme un problème de contrôle d'un système qui consiste à remplacer les entrées naturelles $g_2(k)$ *et* $g_3(k)$ *« nectar et pollen »* par des entrées $u_2(k)$ et $u_3(k)$ *« sirops et farine de mais »* dans les situations défavorables pour que la ruche suit le comportement d'une autre ruche *« Modèle de référence »* qui est supposé idéale.

Avant le début d'une miellée on arrête le contrôle (le nourrissement) et on laisse la ruche fonctionner d'une façon autonome pour que les abeilles produisent du miel pur sans qu'il soit mélangé avec les produits de nourissement.

La stratégie du contrôle avec modèle de référence est illustré par la figure 4.5. Le nourrissement est administré durant [0,60] jours et [150,200] jours où il y'a le manque des entrées g_2,,g_3 (fig.4.2).

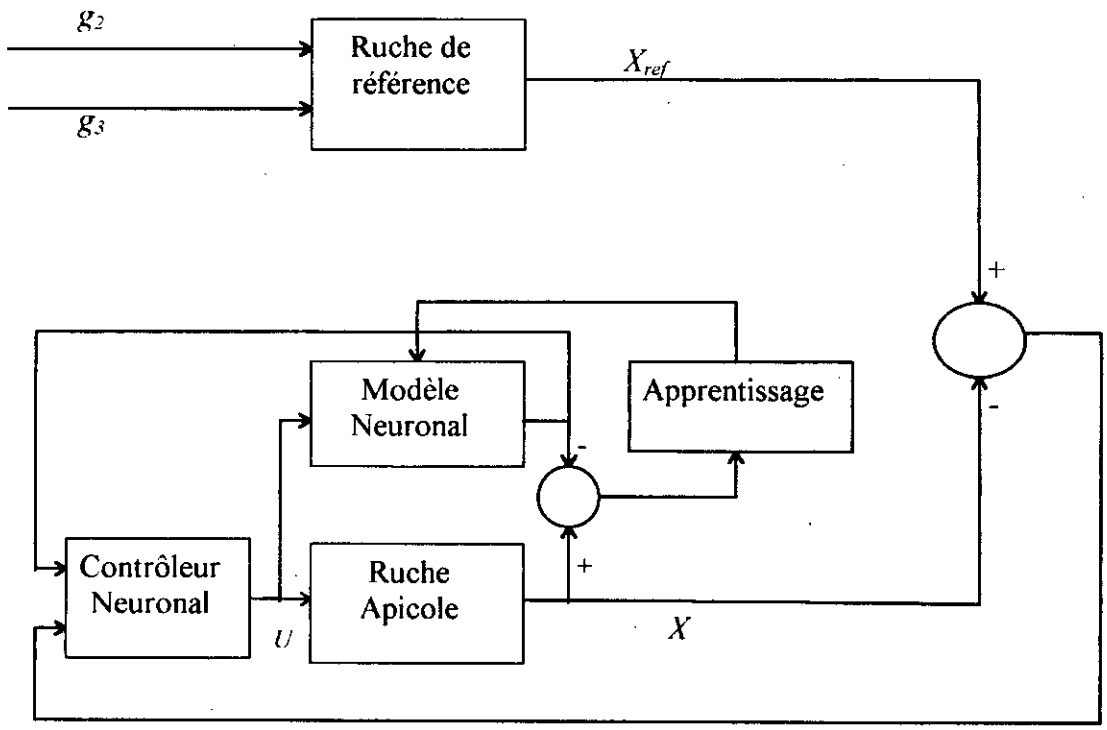


Figure 4.5. Contrôle adaptatif d'une ruche apicole par les réseaux de neurones.

D'après la figure (4.2), on voit que les quantités de nectar et de pollen récoltés sont très faibles dans les intervalles [0,60] et [150, 200] jours. Donc on remplacera ce manque en provision par un nourrissement artificiel (contrôle par les réseaux de neurones). Dans ce travail , on va essayer de contrôler la ruche de sorte que le nombre d'abeilles suit une certaine trajectoire donnée par une ruche de référence.

En se basant sur le systèmes d'équation (4.4) on peut tirer la commande nécessaire.

Le résultat de contrôle de la RUCHE APICOLE durant la première et la deuxième période est illustré par les figures (4.6) et (4.7) respectivement. Dans la figure (4.8) on montre le résultat du contrôle durant toute l'année.

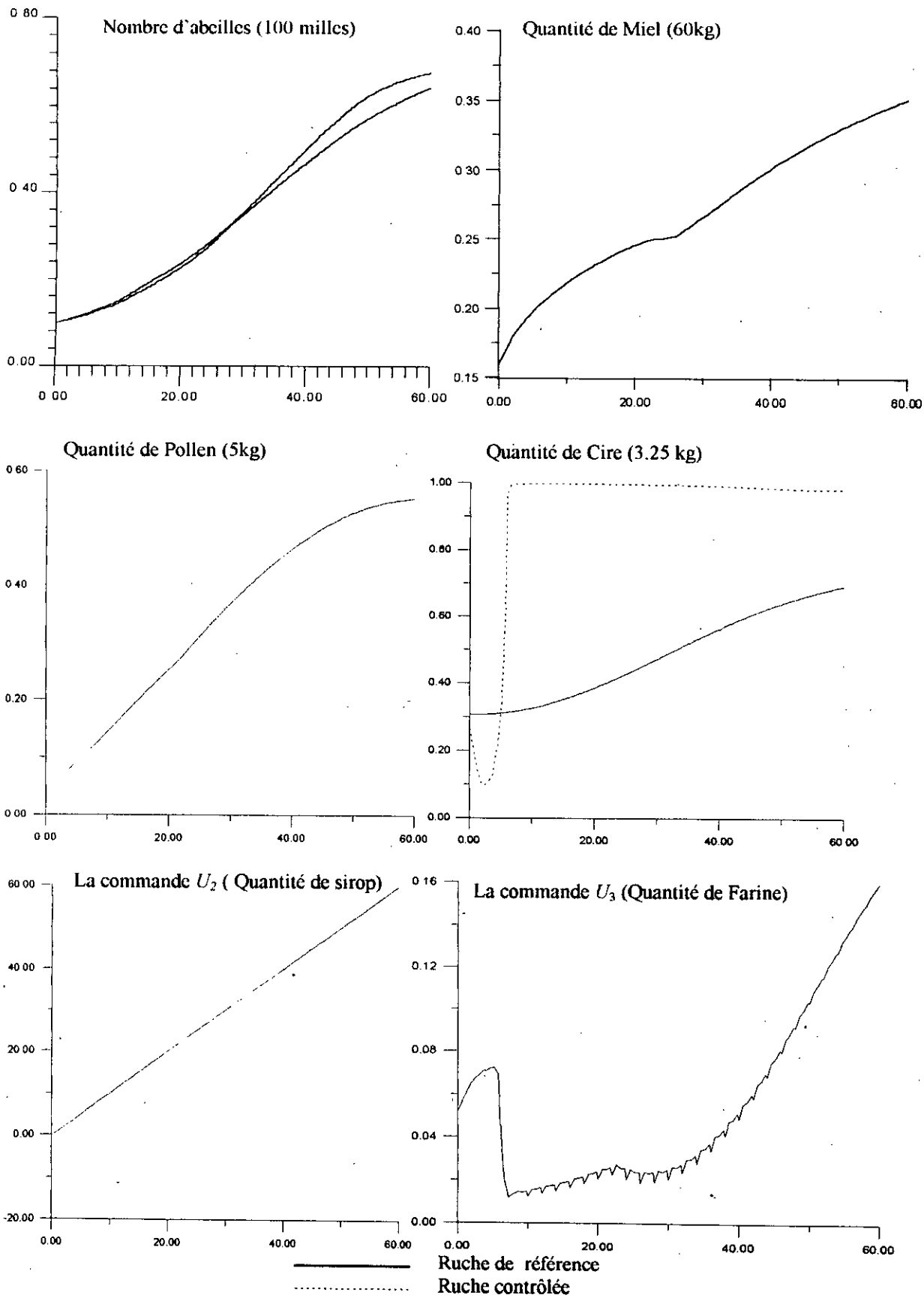


Figure 4.6. Résultats du contrôle de la Ruche dans la période (Janvier,Fevrier).

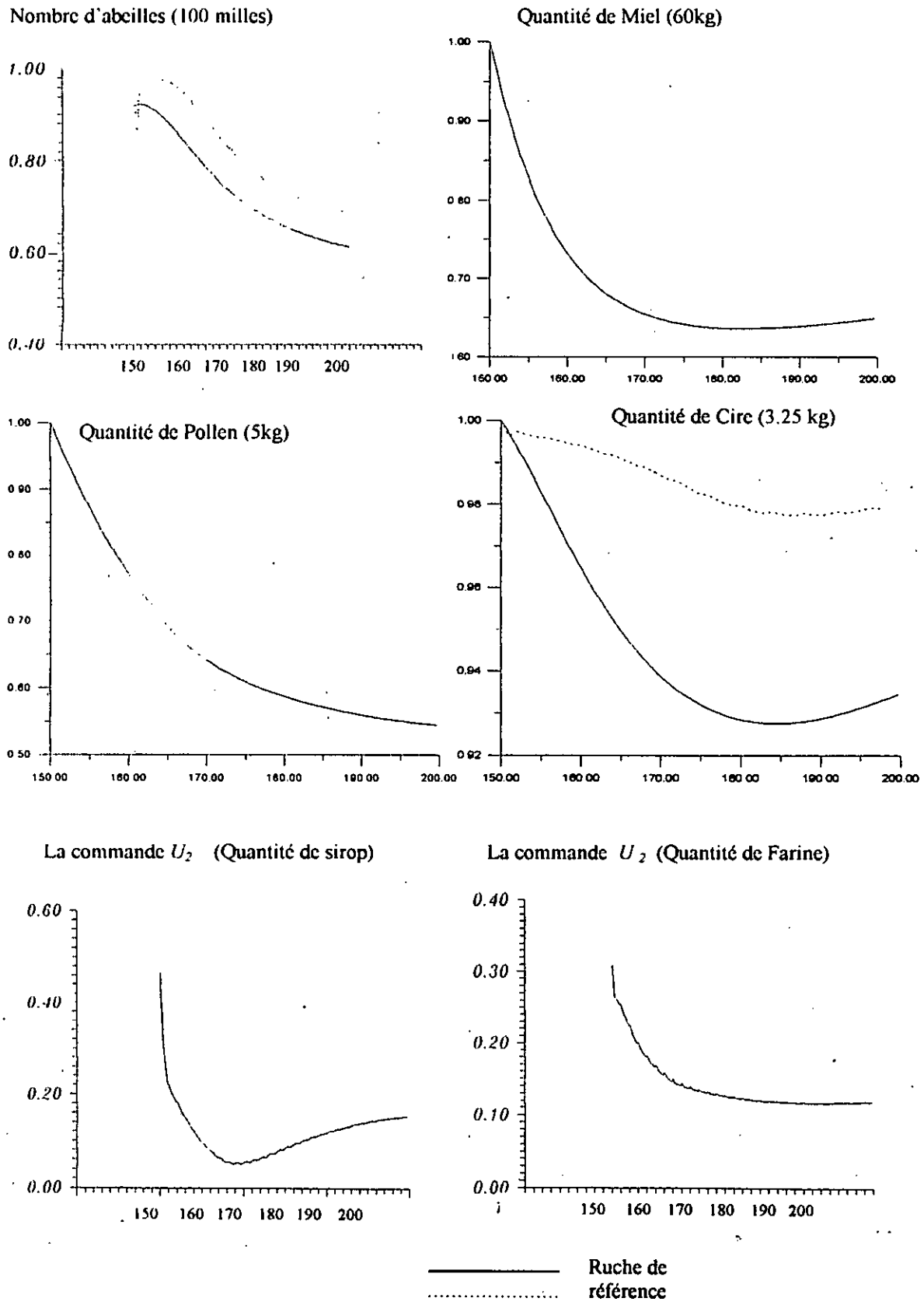


Figure 4.7. Controle de la RUCHE dans la période (Juin, Juillet).

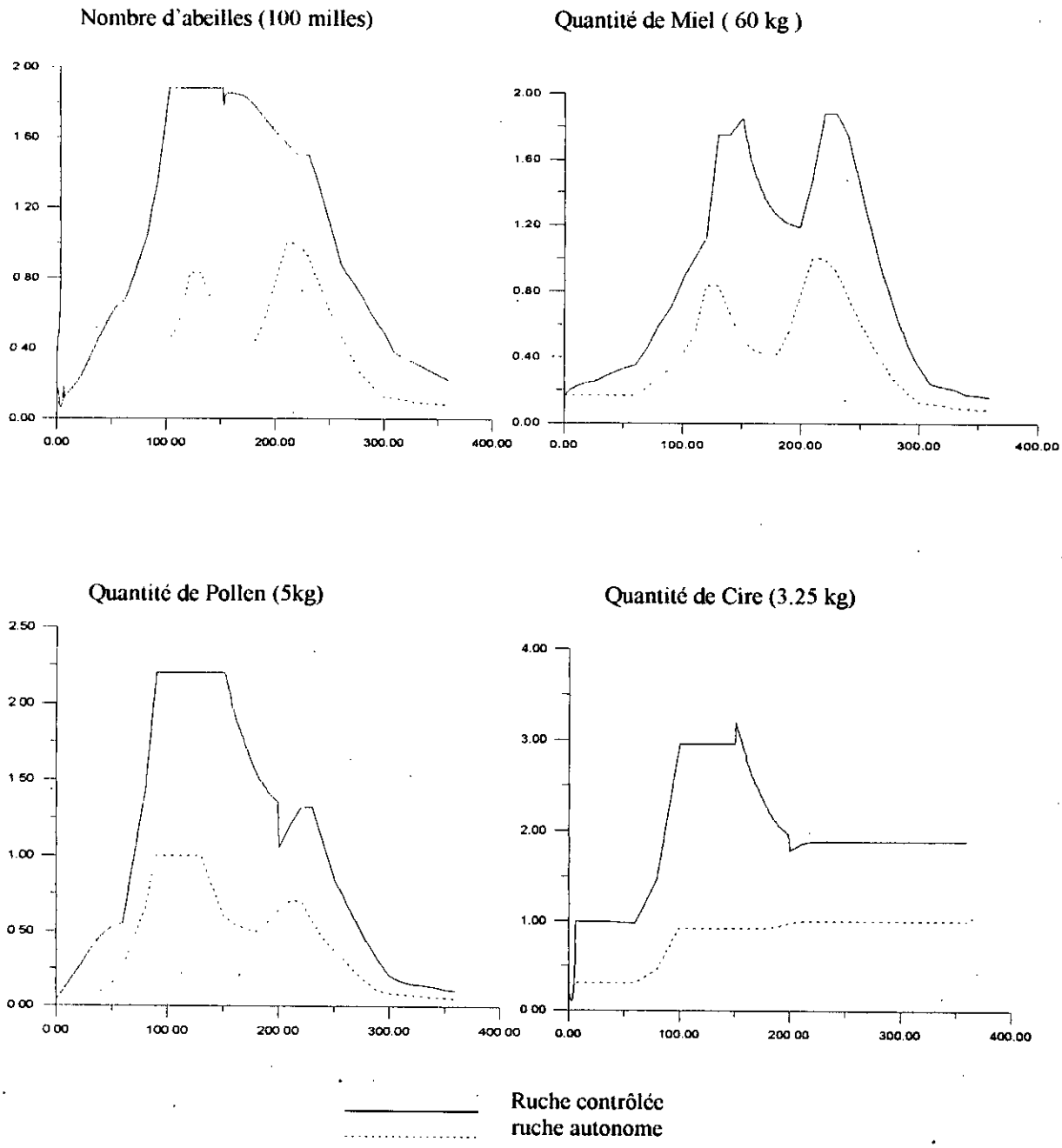


Figure 4.8. Résultat du contrôle de la ruche N°1 durant toute l'année.

IV.11.2. *LE CONTROLE DE LA RUCHE N°2*

Cette ruche est destinée à la production de la cire, le nourrissement se fait durant toute l'année. En utilisant le même principe de contrôle de la ruche N°1, nous avons eu les résultats illustrées par La figure (4.9).

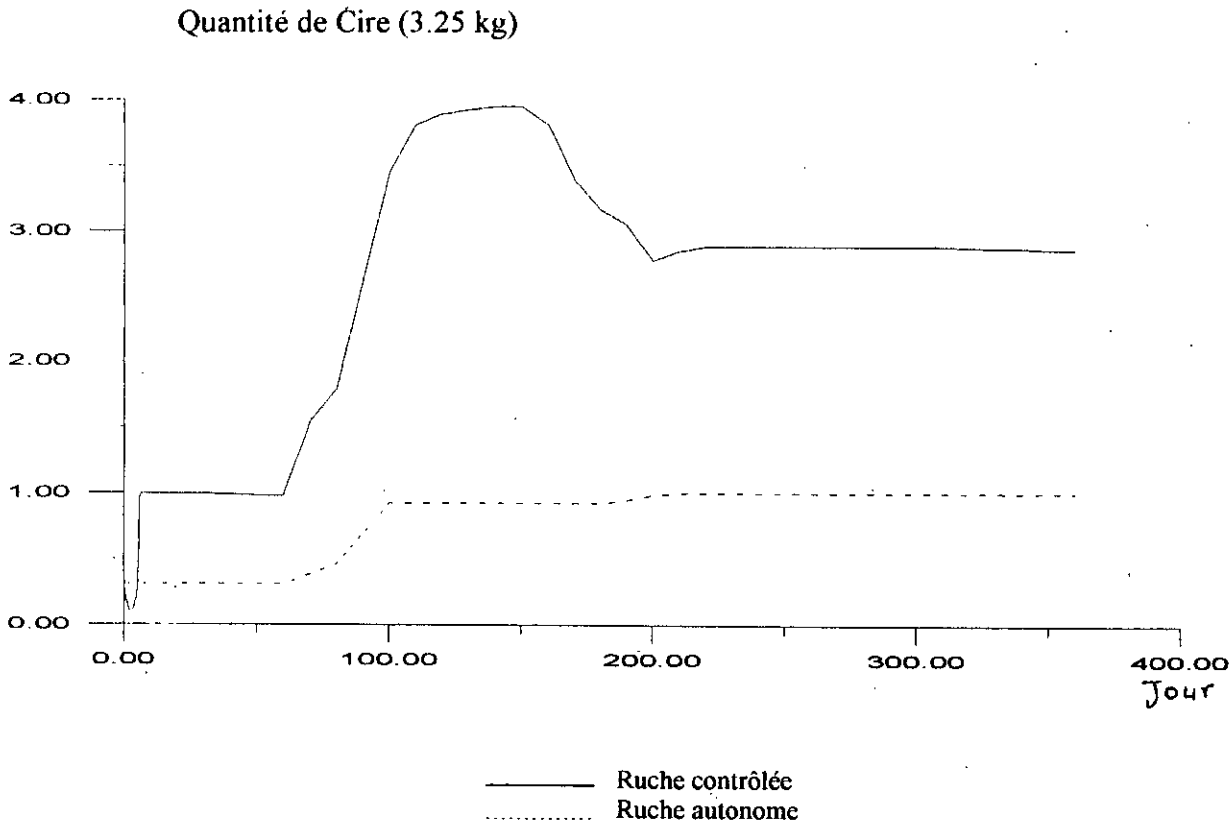


Figure 4.9. Résultat du contrôle de la ruche N°2 durant toute l'année.

IV.12. DISCUSSIONS

Pour la ruche N° 1, nous avons pu augmenter le nombre d'abeilles durant la période [0,60] jours (Janvier,février) jusqu'à 67000 abeilles avec une quantité de 21 kg de miel qui est quantité nécessaire pour approvisionner les abeilles durant 10 jours. Après cette période, il y aura la période [60,150] jours (Mars,Avril) (periode de miellée). Donc on arrête le nourrissement pour que les abeilles produisent du miel pur. On remarque que dans cette période la quantité du miel a augmenté considérablement. Durant la période [150,200] jours (Mai, Juin) il y a un manque en provisions. D'où la nécessité de nourrissement. Mais cette fois le nourrissement se fait avec du miel pur mélangé avec de l'eau pour conserver la pureté du miel stocké.

Il faut noter que les commandes sont appliquées au processus seulement dans les périodes de nourrissement (Janvier,Février) et (Mai,Juin). Durant les autres périodes de l'année, les commandes sont remplacées par les entrées de nectar et de pollen ramenées par les butineuses.

En ce qui concerne la ruche N° 2 qui est destinée à la production de la cire sans prendre en compte la qualité du miel, le nourrissement se fait durant toute l'année. Durant la première période (Janvier,Février) nous avons eu une quantité de 3,18 kg de cire. Dès le début de la miellée (Mars), les abeilles récoltent du nectar et du pollen. Avec ces récoltes, la quantité de cire sera de 9.4 kg. Il suffit donc de faire le nourrissement nécessaire à la production de 2,6 kg pour que la quantité totale de la cire sera de12 kg.

D'après ces résultats, on peut conclure que nous avons atteint les objectifs fixés pour les deux ruches. Il reste seulement l'implantation pratique en réalisant les capteurs nécessaires à l'acquisition des données. L'apprentissage des réseaux de neurones identificateurs des ruches se fait continuellement pour suivre les variations de la ruche apicole durant toute l'année.

IV.13. CONCLUSION

Dans ce chapitre, nous avons utilisé l'étude présentée dans les chapitres précédents pour contrôler deux ruches apicole. L'objectif postulé est l'augmentation de la production du miel pur pour la Ruche N°1 et l'augmentation de la quantité de la cire pour la Ruche N°2 dans les conditions défavorables du système, à savoir, un nombre d'abeilles réduit, les provisions atteignant leurs seuils inférieurs et les conditions climatiques contraignantes. Pour accomplir cet objectif, un nourrissement en matières nutritives est nécessaire. Nous avons tout d'abord identifié la ruche apicole par des réseaux de neurones multicouches d'une façon off-line en utilisant les données mesurées durant l'année 1992. Puis à l'aide du modèle neuronal obtenu, on a tiré la commande nécessaire pour que les deux ruches suivent le comportement d'autres ruches de référence.

Les résultats des simulations obtenus démontrent avec certitude qu'une récolte assez conséquente de miel pur et de la cire peut être obtenue, si la colonie d'abeilles est nourrie convenablement.

CONCLUSION GENERALE

CONCLUSION GENERALE

Le but de cette thèse est le contrôle adaptatif des systèmes non-linéaires avec modèle de référence en utilisant les réseaux de neurones comme identificateurs et contrôleurs.

Nous avons commencé notre travail par une étude détaillée de plusieurs réseaux de neurones.

Le réseau de neurones le plus populaire est le réseau à multicouches, qui est utilisé dans plusieurs articles pour le contrôle des systèmes non-linéaires [1],[3],[4],...etc. Apres des simulations nous avons vu que ce réseau peut approximer n'importe qu'elle fonction non linéaire, mais il possède un temps d'apprentissage très long, d'où l'impossibilité de l'utiliser dans le contrôle adaptatif où il y a des perturbations externes et des variations rapides du système. Il fallait donc trouver une architecture de réseau de neurones qui possède un temps d'apprentissage rapide et fonctionnant d'une façon «ON-LINE» pour avoir une meilleur robustesse.

Le réseau DRNN « *Diagonal recurrent neural network* » est un réseau dynamique qui a donné une solution au problème d'adaptation et la robustesse dans le contrôle adaptatif avec modèle de référence [12]. Dans les simulations nous avons tout d'abord commencé par l'identification «OFF-LINE» d'un système multivariable par les réseaux multicouches, puis à l'aide de ce modèle nous avons tiré le vecteur de commande nécessaire pour que le système suit un certain modèle de référence.

Dans la deuxième simulation nous avons utilisé le réseau DRNN pour le contrôle adaptatif d'un système SISO avec modèle de référence. Dans cette simulation, un système d'identification appelé « Diagonal recurrent neuroidentifier » DRNI qui produit des informations concernant le système au contrôleur « Diagonal recurrent neurocontroler » DRNC.

Pour tester la capacité d'adaptation du réseau DRNN, nous avons changé deux fois successivement le signale de référence, et nous avons remarqué une grande capacité

d'adaptation aux différentes situations. Enfin nous avons perturbé le système à sa sortie et on a vu que le réseau DRNN s'adapte rapidement à cette situation et oblige le système à suivre le modèle de référence.

Enfin on a utilisé l'étude théorique de contrôle adaptatif par les réseaux de neurones pour contrôler un système physique qui est la ruche apicole. D'après les résultats obtenus, on peut conclure que:

1. Pour le cas de la ruche N°1, l'objectif est atteint pour chaque période de nourrissement, et on a remarqué une augmentation considérable de la quantité de miel pur.
2. Pour le cas de la ruche N°2, nous avons fait le nourrissement durant toute l'année et nous avons pu augmenté la quantité de cire.

Nous espérons élargir ce travail par son implémentation pratique, en réalisant des capteurs à savoir:

- Compteur de butineuses de nectar et de pollen en utilisant les techniques de l'imagerie.
- Compteur d'abeilles au sein d'une ruche.
- Capteur pour estimer le quantité de miel, de pollen, de cire et de couvain.

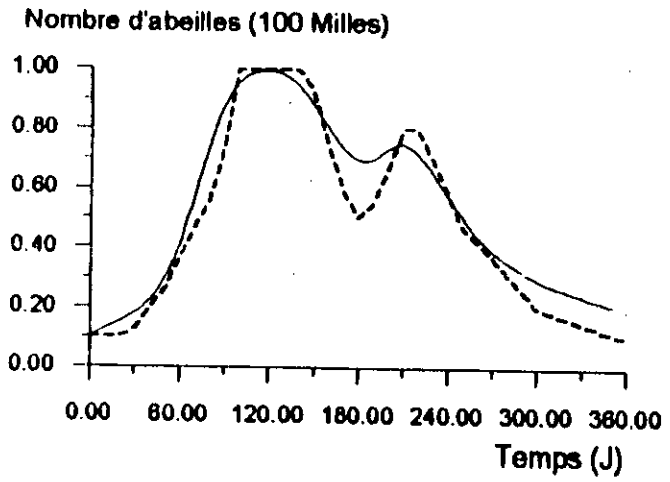
Un système incorporant ces différents capteurs autour d'une ruche une fois réalisé, nous permettra d'obtenir des informations très utiles sur les paramètres d'une miellée dans une région donnée pour l'utilisation des ruches témoins.

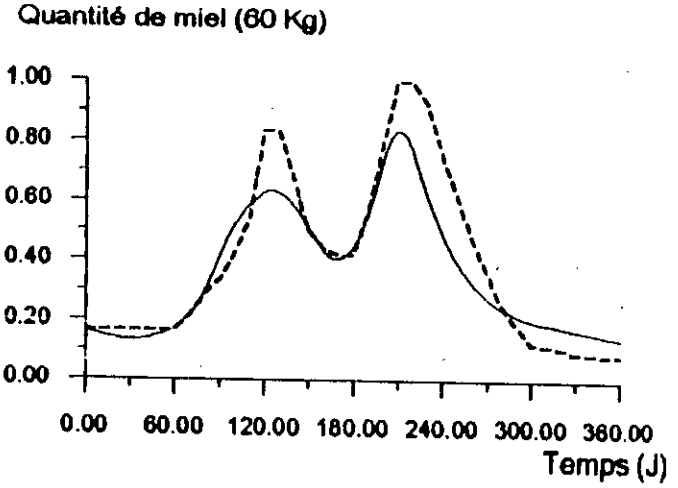
En conclusion, on peut dire que les capacités d'identification, d'apprentissage et de généralisation des réseaux de neurones donnent des issues nouvelles pour les problèmes d'identification et de commande des systèmes non linéaires, dont le traitement se heurte encore aujourd'hui à des difficultés d'ordre pratique et théorique.

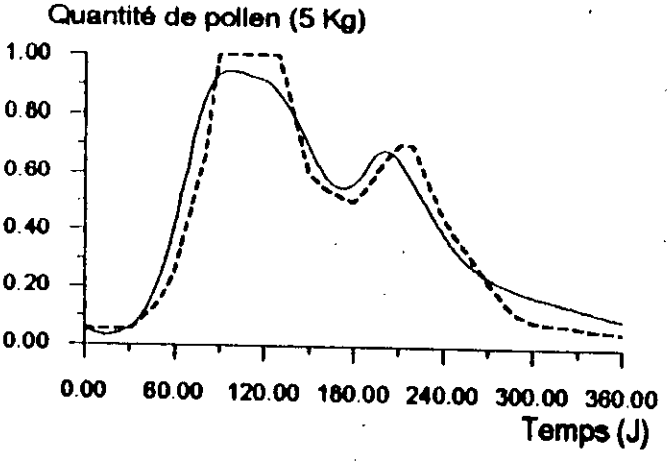
ANNEXES

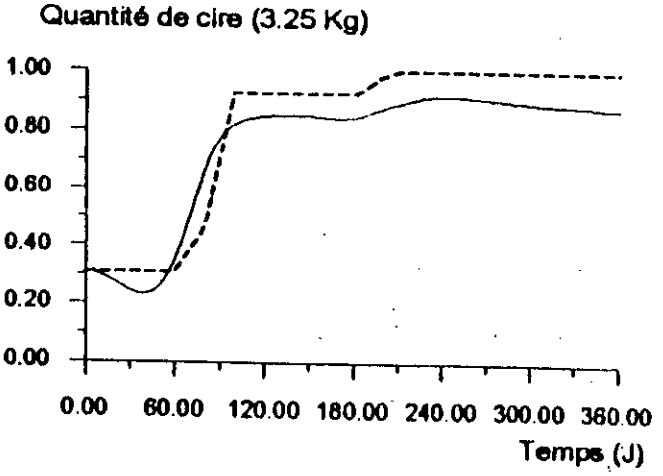
ANNEXE I

Superposition des résultats du modèle général bilinéaire et les données expérimentales.







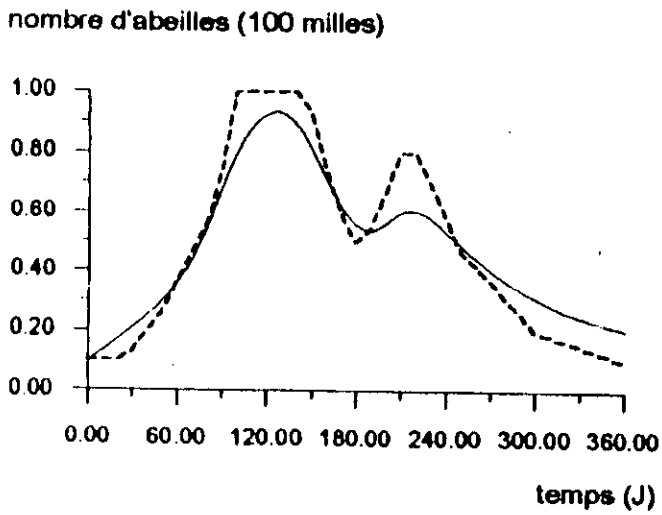


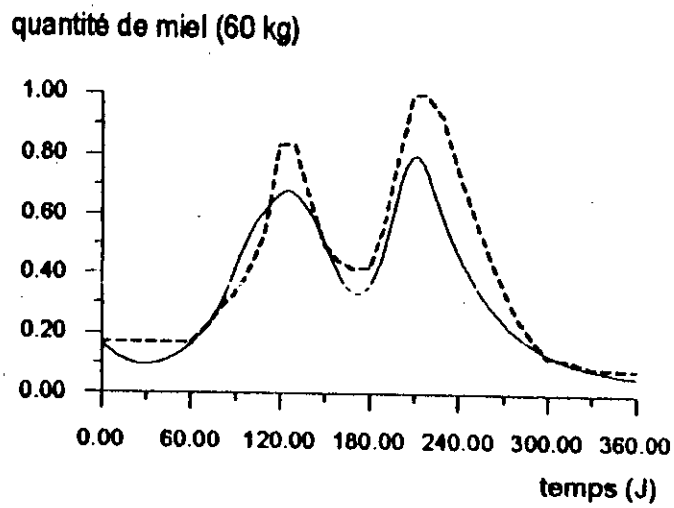
——— Modèle

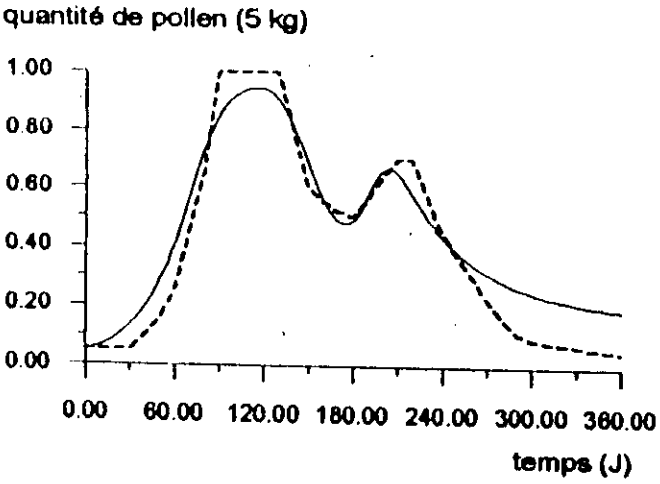
..... Les données expérimentales

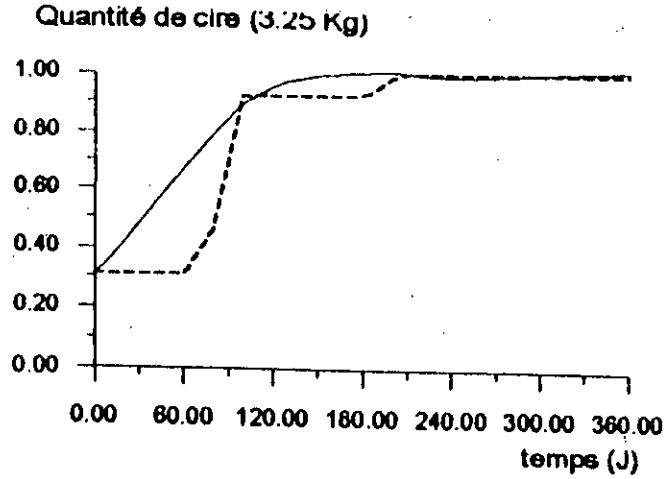
ANNEXE II

Superposition des résultats du modèle LOTKA - VOLTERRA et les données expérimentales.







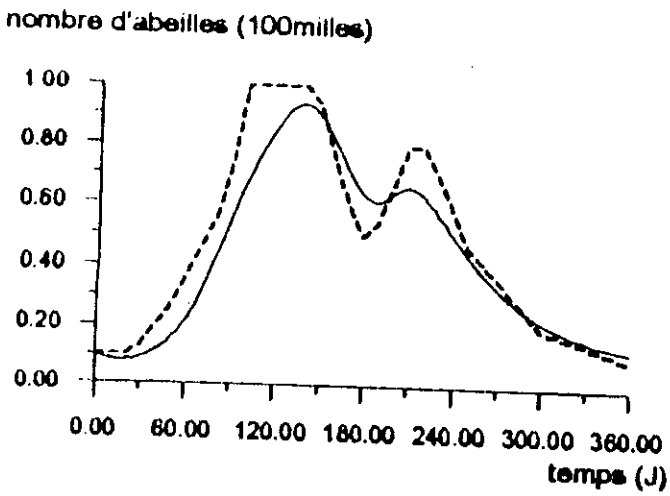


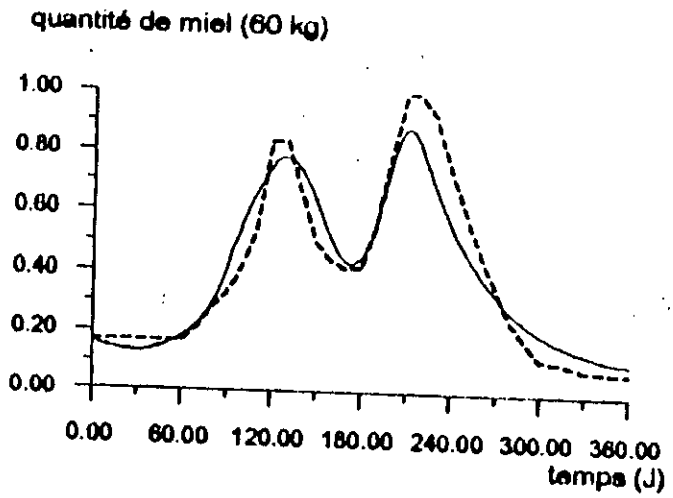
——— Modèle

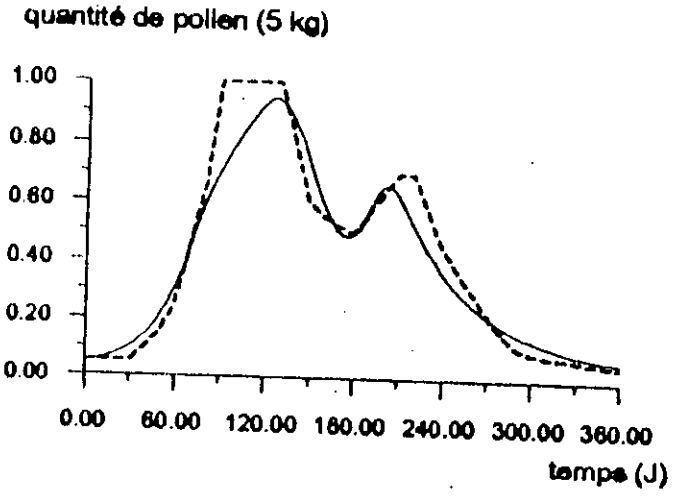
..... Les données expérimentales

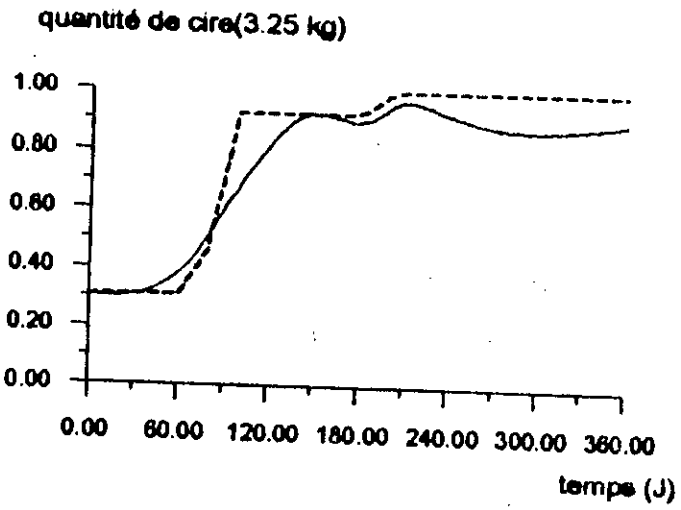
ANNEXE III

Superposition des résultats du modèle linéaire et les données expérimentales.









——— Modèle

..... Les données expérimentales

ANNEXE IV

Résultats du contrôle optimal de la RUCHE APICOLE.

	MODELE UTILISE	OBSERVATIONS
$\nu = 30$ $X_1(t_0) = 50$	LINEAIRE	- $X_1(t_1) = 29$ (objectif atteint avec erreur 3,33%) - Variations brusques de U_2.
$\nu = 50$ $X_1(t_0) = 50$	LINEAIRE	- $X_1(t_1) = 48$ (objectif atteint avec erreur 4%) - Variations brusques de U_2.
$\nu = 50$ $X_1(t_0) = 30$	LINEAIRE	- $X_1(t_1) = 47$ (objectif atteint avec erreur 6 %) - Variations brusques de U_2. - Valeurs négatives de X_2.
$\nu = 50$ $X_1(t_0) = 10$	LINEAIRE	- $X_1(t_1) = 46$ (objectif atteint avec erreur 8 %) - Variations brusques de U_2. - Valeurs négatives de X_2.
$\nu = 100$ $X_1(t_0) = 50$	LINEAIRE	- $X_1(t_1) = 94$ (objectif atteint avec erreur 6 %) - Variations brusques de U_2. - Valeurs négatives de X_2.

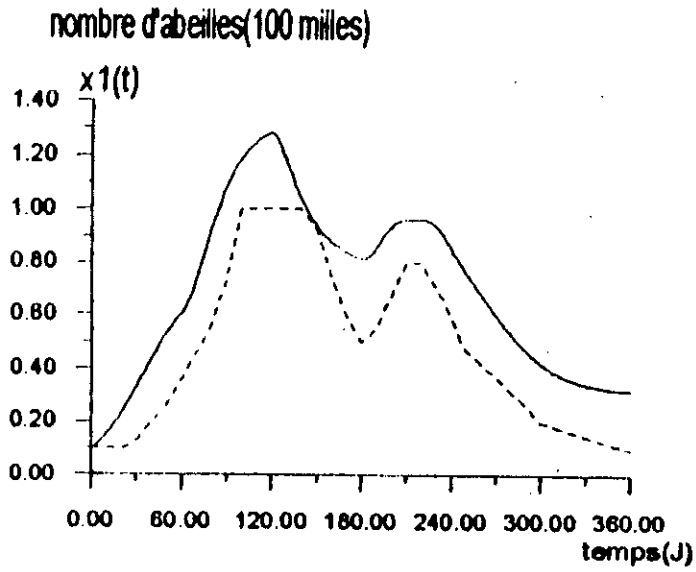
Tableau 4.1

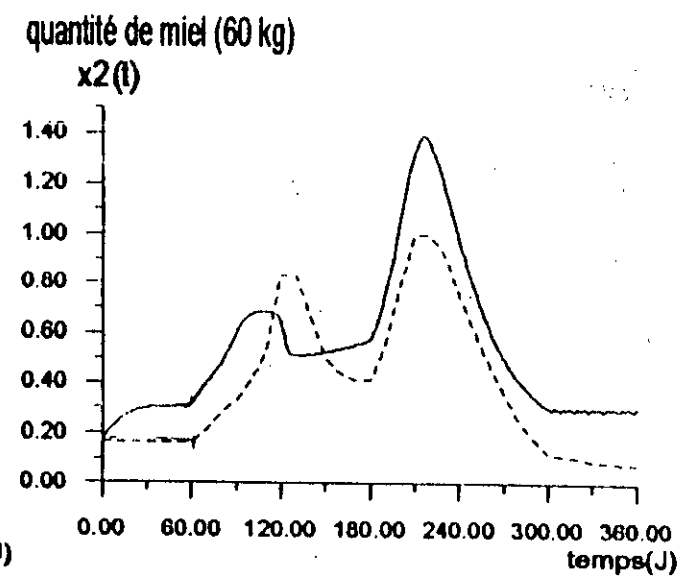
	MODELE UTILISE	OBSERVATIONS
$\nu = 50$ $X_1(t_0) = 50$	NON LINEAIRE	- $X_1(t_1) = 64$ (objectif atteint avec erreur 28 %)
$\nu = 50$ $X_1(t_0) = 30$	NON LINEAIRE	- $X_1(t_1) = 69$ (objectif atteint avec erreur 38 %)
$\nu = 100$ $X_1(t_0) = 50$	NON LINEAIRE	- $X_1(t_1) = 46$ (objectif atteint avec erreur 8 %)

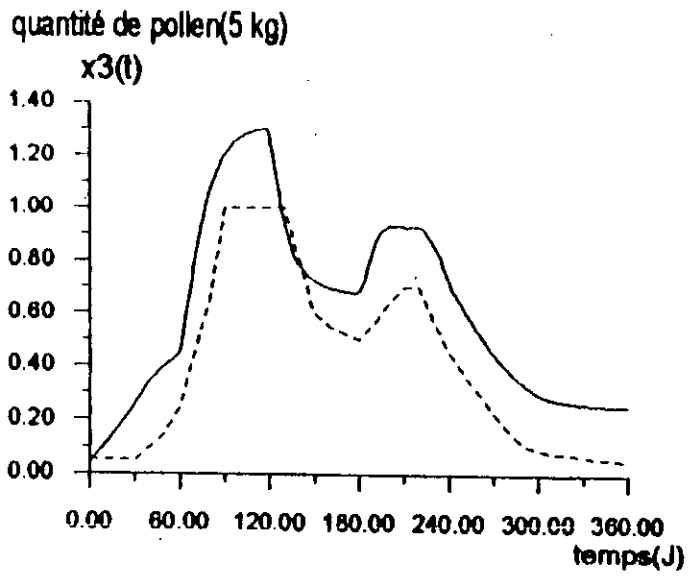
Tableau 4.2

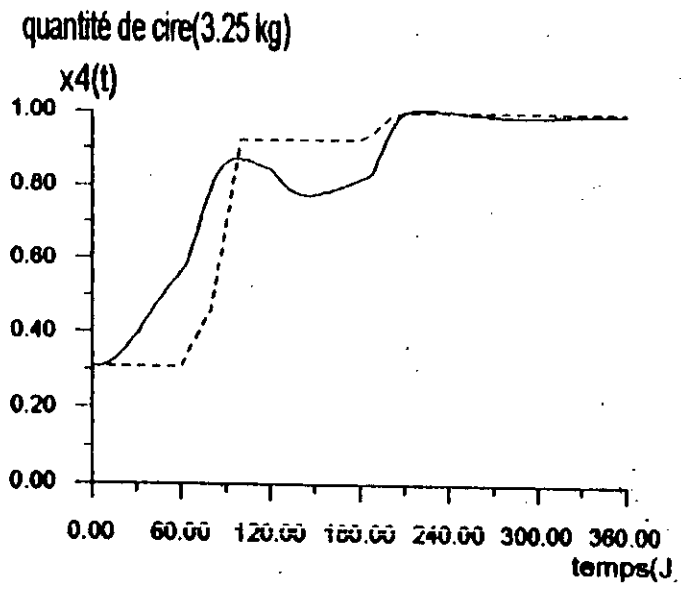
ANNEXE V

Résultats du contrôle Flou de la RUCHE APICOLE.









——— Ruche autonome

..... Ruche contrôler

REFERENCES BIBLIOGRAPHIQUES

BIBLIOGRAPHIE

[1] K.S. Narendra and K.Parthasarathy, « Identification and control of dynamic systems using neural networks », *IEEE Trans.Neural Networks*,Vol.1, pp. 4 - 27, Mar.1990.

[2] Li,W, and Slotine, J.E.,« Neural network control of a unknown nonlinear systems », *Proc. 1989 American Control Conference, Pittsburgh*, Vol.2, PP. 1136-1141 1989.

[3] Don. R. Hush and Bill G. Horne « Progress in supervised Neural Network ». *IEEE Signal processing magazine.* Jan. 1993.

[4] Maarouf Saad, Pascal Bigras, Louis-A, and Kamal Al-Haddad, « Adaptive Robot Control Using Neural Networks »,*IEEE Transactions on industrial electronics,* Vol 41, N° .2, April 1994.

[5] Levin. A. U. and Narendra. K. S, « Control of nonlinear dynamical systems using neural networks: controllability and Stabilisation », *IEEE Trans.N.N.4.192 - 206.*

[6] Si . Zhao Quin, H. Te .Su, and T.J.McAvoy, « Comparison of four neural network learning methods for dynamic system identification », *IEEE Trans.N.N,* Vol 3, N° 1. pp.122 - 130. Jan 1992.

[7] S.Y.Kung , *Digital neural networks* , PTR Prentice Hall. Englewood Cliffs, New Jersey 07632, 1993.

[8] S.Haykin, *Neural Networks A Comprehensive Foundation* , Copyright by Macmillan college publishing company, INC, 1994.

[10] M.SAAD, L. A. DESSAINT, P.BIGRAS AND K. AL- HADDAD, « Adaptive versus neural adaptive control : Application to ROBOTICS ». *International Jounal Of Control and Signal Processing,* Vol. 8, 223-236 1994.

[11] G.Lightbody and Prof. G.W. Irwin,« Direct Neural Model reference adaptive control », *IEE Proc-Control Theory Appl,* Vol. 142, N° 1, January 1995.

[12] Chao- Chee Ku and Kwang Y.lee,« Diagonal recurrent neural networks for dynamic systems control »,*IEEE Transaction on neural networks,* Vol.6, N° 1, January 1995.

[13] F. BOUDJEMA, D. BOUKHETALA, M.C.SOUAMI et B. HAMZI,S. LABIOD, *Identification et commande des sytèmes Dynamiques par les réseaux de neurones,* Thèse de projet de fin d'études Dpt. Génie Electrique (E.N.P), Juillet 1995.

[14] Dr. H. SALHI,« Modélisation d'une Ruche Apicole » *S.S.A 1992 Blida*

[15] A. GUESSOUM, H. SALHI et I. RIZOUG, A. DJELLAL, *Commande d'une Ruche Apicole par la logique floue* ,Thèse de projet de fin d'études -Institut d'électronique- UNIVERSITE DE BLIDA Octobre 1995.

[16] H. SALHI et DRA- EL-MIZAN BACHIRA, *Commande d'une Ruche Apicole* , Thèse de projet de fin d'études - Institut de mathématique - UNIVERSITE DE BLIDA 1994.

[17] BOUMEHRAZ. M, KHIER. B, BOUKEZZOULA.N, et ARRAS.B, *Identification et controle avec les réseaux de neurones , S.S.A 1992 Blida.*

[18] Narendra.K, ANURARDHA M.A, *Stable adapive systems* , by Prentice-Hall, Inc 1989.

[19] V.V.Chalam, *Adaptive Control Systems* , by MARCEL DEKKER, INC 1987.

[20] L.Ljung and T.Sǫderstrǫm, *Theory and practice of recursive identification* , Cambridge, MA : MIT Press, 1983.

[21] D.P. BERTSEKAS, Lectures notes of EE410 University of Illinois, USA, 1976.

[22] D.A. Hoskins, G.N. Hwang, and J.vagners, « Iterative inversion of neural networks and its application to adaptative control », *IEEE Transaction of neural networks,* VOL.3, N° 2, Marsh 1992.

[23] SUN-YUANKUNG,Fellow,IEEE, and JENQ-NENG HWANG, member, IEEE, « Neural networks architectures for robotic applications »,*IEEE Transactions in Robotics and Automation,* VOL.5, N° 5, October 1989.

[24] D. Sbarbaro, K.J. Hunt, and P.J. Gawthrop, « Designing nonlinear controllers using connectionist networks », *13th IMACS World Congress on Computation and Applied Mathematics,* Trinity College Dublin, Irland, JULY 22-26 , 1991.

[25] Toshio Fukuda, menber, IEEE, and Tkanori Shibata, « Theory and application of neural networks for industrial control systems »,*IEEE Transactions On Industrial Electronics,* VOL.39, N° 6, DECEMBER 1992.

[26] Maarouf Saad, Pascal Bigras, Louis-A. Dessaint, Senior member, IEEE, and Kamel Al-Hadad, Senior member, IEEE, « Adaptive robot control using neural networks », *IEEE Transactions On Industrial Electronics,* VOL.41, N° 2, APRIL 1994.

[28] Ahmed Karakasoglu, Subramania I. Sudharsanan, Member, IEEE , and Malur K. Sundareshan, Senior Member, IEEE, « Identification and decentralized adaptive control using dynamical neural networks with application to robotic manipulators », *IEEE Transactions on neural networks,* VOL.4, N° 6, NOVEMBER 1993.

[29] I.D. Landau and L. Dugard, *Commande adaptative aspects pratiques et théoriques* , Edition Masson 1986.

[30] A. Müller, Anton Gunzinger, and Walter Guggenbühl, « Fast Neural Net Simulation with a DSP Processor Array », *IEEE Transactions on neural networks,* VOL.6, N° 1, January 1995.

[31] Rangachari Anand, Kshan Mehtora, Chilukuri K. Mohan, and Sanjay Ranka, « Efficient Classification for Multiclass Problems Using Modular Neural Networks », *IEEE Transactions on neural networks,* VOL.6, N° 1, January 1995.

[32] S. Ergezinger and E. Thomsen, « An Accelerated Learning Algorithm for Multilayer Perceptrons : Optimization Layer by Layer », *IEEE Transactions on neural networks,* VOL.6, N° 1, January 1995.

[33] Fu-Chung Chen, « Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control », *IEEE Control Systems Magazine, 1990.*

[34] Tjaart van der Walt, Etienne Barnard, and jannie van deventer, « Process Modeling with the Regression Network », *IEEE Transactions on neural networks,* VOL.6, N° 1, January 1995.

[35] Stanislaw H. Zak, Viriya Upatising, and Stefen Hui, « Solving Linear Programming Problems with Neural Networks : A Comparative Study », *IEEE Transactions on neural networks,* VOL.6, N° 1, January 1995.