Electronique

THESE

PRESENTEE

A L'institut D'electronique
de L'universite de Constantine

En vue de l'obtention du titre de Magister en Electronique
OPTION : Decision et controle

Par
KAMEL KARA

THEME

APPLICATION DES RESEAUX DE NEURONES A L'IDENTIFICATION DES SYSTEMES NON LINEAIRES

Soutenue le : 30/01/1995

Devant le jury d'examen :

K . BELARBI	(M.C Universite de Constantine)	President
K . BENMAHAMMED	(M.C Universite de Setif)	Rapporteur
M. BARKAT	(M.C Universite de Constantine)	Examinateur
A . KHELLAF	(M.C Universite de Setif)	Examinateur

A tous ceux qui me sont chers.

REMERCIEMENTS

Je tiens à remercier très chaleureusement Monsieur K. BELARBI, maître de conférences à l'université de Constantine, de l'honneur qu'il m'a fait en présidant le jury de cette thèse.

Je tiens à exprimer toute ma gratitude à Monsieur K. BENMAHAMMED, maître de conférences à l'université de Sétif, pour m'avoir orienter et pour la confiance et l'intérêt qu'il a témoigné tout au long de l'élaboration de ce travail.

Que Monsieur M. BARKAT, maître de conférences à l'université de Constantine, trouve ici l'expression de mes remerciements pour avoir accepter d'être l'un des examinateurs de ce travail.

Je suis également très sensible à l'intérêt que Monsieur A.KHELLAF, maître de conférences à l'université de Sétif, a accordé à mon travail en acceptant d'examiner cette thèse.

Je suis fort reconnaissant à Messieurs T. ABED, étudiant en première post-graduation à l'université de Sétif, et N. BENOUADJIT, maître assistant à l'université de Batna, qui m'ont grandement aidé dans la réalisation de ce travail.

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

RESUME

Dans ce travail, on a mis en évidence les propriétés et avantages des réseaux de neurones, et envisagé leurs exploitation dans l'identification des systèmes dynamiques non linéaires.

Après avoir donné les définitions et les concepts de base des réseaux de neurones, on a présenté une étude détaillée sur les architectures statiques MLP et RBF. Ensuite, le problème d'identification des systèmes non linéaires en utilisant ces architectures a été considéré.

Ce problème a été aussi étudié en utilisant les réseaux récurrents pour lesquels on a introduit une nouvelle structure. Enfin, la solution du problème de réjection des perturbations additives en entrée a été etendue au cas des perturbations additives en sortie.

Plusieurs exemples de simulation ont été utilisés pour montrer l'apport des techniques connexionistes dans le problème d'identification des systèmes non linéaires.

SOMMAIRE

INTRODUCTION

INTRODUCTION

La recherche d'un modèle, au sens large du terme, c'est finalement l'objet de la physique toute entière. Si l'on tient un discours moins général, le besoin de prédire le comportement dynamique d'un objet, d'une entité naturel ou artificiel, s'est accru brutalement avec les impératifs stratégiques nés de la seconde guerre mondiale.

L'identification est devenue un atout maître dans la recherche de modèles dits de comportement, qu'il faut fondamentalement distinguer des modèles de connaissance. Ces derniers représentent le plus souvent le savoir-faire de spécialistes d'un domaine particulier et sont directement élaborés à partir des équations décrivant les phénomènes fondamentaux de façon précise et détaillée.

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

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

Deux raisons importantes motivent les automaticiens :

- prédire le comportement d'un système pour différentes conditions de fonctionnement (analyse, simulation).
- élaborer une loi de commande à appliquer au processus, de façon qu'il réalise " au mieux " l'objectif assigné (synthèse des lois de commande). Cette loi peut être établie selon deux grands types de méthodes :

- celles ne nécessitant pas la connaissance d'un modèle

mathématique; c'est par exemple le cas du classique bouclage P.I.D.

- celles nécessitant un modèle mathématique.

La théorie de contrôle des processus fournit des techniques d'identification parfaitement adaptées aux systèmes linéaires. Cependant, en pratique ces méthodes ne s'avèrent pas toujours applicables à cause des non linéarités des systèmes réels et parce qu'il n'est pas toujours possible de linéariser le système à commander. Bien que, plusieurs techniques classiques pour l'identification de certaines classes des systèmes non linéaires sont disponibles dans la littérature, des méthodes générales et efficaces qui peuvent être utilisées pour l'identification des systèmes non linéaires à structures inconnues ne sont pas encore disponibles.

L'application des techniques neuronales pour l'identification des systèmes non linéaires peut fournir des nouvelles solutions pour ce problème. En effet, les capacités d'identification des réseaux de neurones, en particulier ceux de type multi-couches, leur aptitude à la généralisation et leur adaptativité, nous conduisent aujourd'hui à étudier et développer des architectures modulaires à base de réseaux de neurones en identification.

En général, les réseaux de neurones ont des applications puissantes dans tous les niveaux d'une structure de commande hiérarchisée qui permet au système d'avoir un degré élevé d'autonomie.

L'objectif du présent travail est de mettre en évidence les propriétés et avantages des réseaux de neurones, et d'envisager leur exploitation dans l'identification des systèmes dynamiques non linéaires.

Le premier chapitre donne un aperçu général sur les réseaux de neurones: les définitions et les concepts de base.

Le deuxième chapitre est consacré aux architectures des réseaux de neurones. On présente une étude détaillée sur les architectures statiques MLP et RBF, et on donne un bref aperçu sur les réseaux dynamiques.

Le chapitre trois détaille le problème d'identification des systèmes non linéaires en utilisant les réseaux statiques MLP et RBF, et donne une solution pour le problème de rejection des perturbations additives en entrée.

Dans le chapitre quatre, on considère l'identification des systèmes non linéaires avec les réseaux dynamiques et on introduit une solution pour le problème de rejection des perturbations additives en sortie.

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

CHAPITRE I

Les réseaux de neurones

LES RESEAUX DE NEURONES

La recherche sur les réseaux de neurones a connu un développement important au cours de la dernière décennie et a déjà donné lieu à des applications intéressantes en plusieurs domaines.

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

1-1 HISTORIQUE :

L'idée des réseaux de neurones vient originalement de la modélisation biophysique du cerveau [1-5], cette modélisation tente d'expliquer la biophysiologie du cerveau et ses fonctionnalités.

Le but de la recherche sur les réseaux de neurones, n'est pas de créer des machines qui traitent l'information plus rapidement que les calculateurs traditionnels, mais c'est de créer des machines qui se montrent supérieures dans les domaines où le cerveau humain dépasse ces calculateurs [4].

Ces recherches débutèrent en 1890, quand le psychologiste William James étudia les activités de l'esprit [1,5]. En 1943, Mc Culloch et Pitts adoptèrent les affirmations de James [1] et formalisèrent une description du neurone. Puis dans les années soixantes, Frank Rosenblatt et Bernard Widrow, réalisèrent des "neurones adaptatifs" et des réseaux simples capables d'apprentissage.

Le neurone de Mc Culloch et Pitts est l'élément fondamentale de tous les réseaux, la recherche dans le domaine des réseaux de neurones est centrée sur les architectures selon lesquelles, les neurones sont combinés et les méthodologies par lesquelles les poids des interconnexions sont calculés ou ajustés.

Aujourd'hui, il y a deux groupes de chercheurs, le premier est constitué des biologistes, physiciens et des psychologistes. Ce groupe tente de développer un modèle neuronal capable d'imiter, avec une précision donnée, le comportement du cerveau. Le second groupe se compose des ingénieurs qui sont concernés par la façon avec laquelle les neurones artificiels sont interconnectés pour former des réseaux possédant des capacités de calcul puissantes.

En 1969, Minsky et Papert démontrèrent, dans leur ouvrage intitulé "Perceptrons", un certain nombre de théorèmes sur les limitations d'un réseau mono-couche, malheureusement ils ont conclus que ces limitations se généralisent pour les réseaux multi-couches [1,3]. Ceci a poussé de nombreux chercheurs à abandonner cette voie pour se diriger vers l'intelligence artificielle qui semblait un domaine plus prometteur.

Les jugements de Minsky et Papert ont été contredits; tous les problèmes non linéairement séparables peuvent être résolus en utilisant les réseaux multi-couches, les recherches recommencèrent en 1974, quand Werbos développa un algorithme, dit algorithme de la rétropropagation, pour entraîner les réseaux multi-couches.

D'autres chercheurs, comme Shunichi Amari, Stephen Grossberg, Kohonen et Bart Kosko ont tentés de mieux modéliser le fonctionnement des neurones; ils ont étudié des modèles mathématiques et des architectures de circuits afin de classer des formes, d'une part et de constituer des "mémoires associatives", d'une autre part.

Actuellement les études des circuits neuronaux sont en pleine expansion. Les neurobiologistes ont affinés leur compréhension des mécanismes cérébraux de traitement de l'information et, parallèlement, les ordinateurs étant plus puissants, les analystes ont pu analyser plus en détail les

modèles théoriques. Un nombre croissant de chercheurs s'intéressent au calcul parallèle et aux circuits VLSI, qui permettent de réaliser certains types de réseaux de neurones formels, et de nombreux résultats mathématiques ont amélioré les modélisations.

1-2 ELEMENTS DE BASE ET TERMINOLOGIE :

1-2-1 MODELISATION BIOLOGIQUE :

Le cerveau humain, est le meilleur modèle de machine, polyvalente, incroyablement rapide et surtout douée d'une incomparable capacité d'auto-organisation, son comportement est beaucoup plus mystérieux que le comportement de ses cellules de base. Il est constitué d'un grand nombre d'unités biologiques élémentaires (environ 100 milliards unités) qui sont les neurones, unités qui, de plus sont extrêmement interconnectées: chacune reçoit et envoit des informations à plusieurs milliers de ses congénères.

Les cellules nerveuses, appelées neurones, sont les éléments de base du système nerveux centrale. Elles sont constituées de trois parties essentielles [1,3,4] : le corps cellulaire, les dendrites et l'axone (fig.(1-1)).

1-2-1-1 LE CORPS CELLULAIRE :

Parfois appelé soma, il contient le noyau du neurone et effectue les transformations biochimiques nécessaires à la synthèse des enzymes et d'autres molécules qui assurent la vie du neurone.

1-2-1-2 LES DENDRITES :

Chaque neurone possède une "chevelure" de dendrites qui entourent le corps cellulaire. Celles-ci se ramifient, ce qui les amène à former une espèce d'arboressance autour du corps cellulaire. Elles sont les récepteurs principaux du neurone pour capter les signaux qui lui parviennent.

1-2-1-3 L'AXONE :

L'axone qui est à proprement parler la fibre nerveuse, sert de moyen de transport pour les signaux émis par le neurone. Les neurones sont connectés les uns aux autres suivant des répartitions spatiales complexes [3]. Les connexions entre deux neurones se font en des endroits appelés synapses où ils sont séparés par un petit espace synaptique de l'ordre du centième de micron.

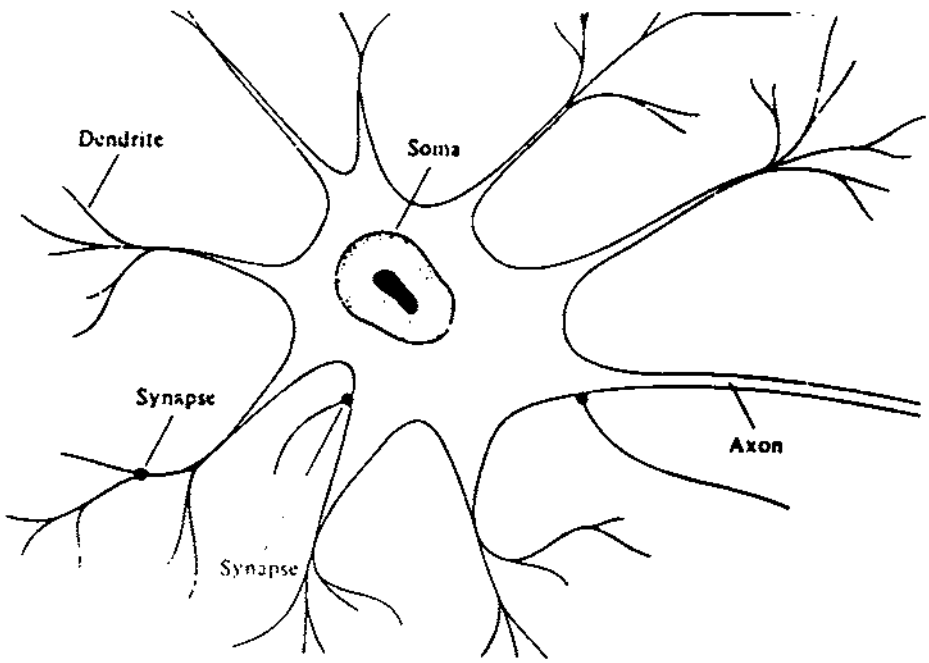


fig.(1-1): Anatomie d'un neurone

1-2-2 FONCTIONNEMENT DES NEURONES :

D'une façon générale, on peut dire que le soma du neurone traite les courants électriques qui lui proviennent de ses dendrites, et qu'il transmet le courant électrique résultant de ce traitement aux neurones auxquelles il est connecté par l'intermédiaire de son axone [3,4].

Le modèle classique présenté par les biologistes est celui d'un soma effectuant une sommation des influx nerveux transmis par ses dendrites. Si la sommation dépasse un seuil, le neurone répond par un influx nerveux ou potentiel d'action qui se propage le long de son axone. Si la sommation est inférieure à ce seuil, le neurone reste inactif [4,6].

Les neurones peuvent engendre les potentiels d'action dans une large gamme de fréquences, mais tous ont la même amplitude. L'information transmise est donc représentée par le nombre de potentiels d'action produits par unité de temps. Ceci correspond à un codage en fréquence des signaux transmis [3].

lorsqu'un potentiel d'action est parvenu à l'extrémité d'un axone relié à une dendrite par un synapse, il provoque à travers la membrane la libération d'un médiateur chimiques (des ions chimiques) au niveau de cette synapse. Ce médiateur se diffuse jusqu'à la membrane postsynaptique de la dendrite où, il provoque la naissance d'un potentiel appelé potentiel postsynaptique. Ce potentiel se propage ensuite le long de la dendrite vers le soma du neurone.

1-2-3 LE NEURONE FORMEL :

Le premier modèle du neurone formel date des années quarante. Il a été présenté par Mc Culloch et Pitts. S'inspirant de leurs travaux sur les neurones biologiques ils ont proposé le modèle suivant [1,3] :
Un neurone formel fait une somme pondérée des potentiels d'action qui lui parviennent. Puis s'active suivant la valeur de cette sommation pondérée. Si cette somme dépasse un certain seuil, le neurone est activé et transmet une réponse dont la valeur est celle de son activation. Si le neurone n'est pas activé, il ne transmet rien.

D'une façon plus générale, un neurone formel est un élément de traitement possédant n entrées $X_1, X_2, \ldots, X_n$ (qui sont les entrées externes ou les sorties des autres neurones),et une seule sortie. Son traitement consiste à affecter à sa sortie Y_j le résultat d'une fonction de seuillage f (dite aussi fonction d'activation) de la somme pondérée.

$$Y_j = f\left(\sum_{i=1}^{n} W_{ij} X_i\right) \qquad (1.1)$$

Où W_{ij} est la pondération (coefficient synaptique ou poids)

Les neurones peuvent engendre les potentiels d'action dans une large gamme de fréquences, mais tous ont la même amplitude. L'information transmise est donc représentée par le nombre de potentiels d'action produits par unité de temps. Ceci correspond à un codage en fréquence des signaux transmis [3].

lorsqu'un potentiel d'action est parvenu à l'extrémité d'un axone relié à une dendrite par un synapse, il provoque à travers la membrane la libération d'un médiateur chimiques (des ions chimiques) au niveau de cette synapse. Ce médiateur se diffuse jusqu'à la membrane postsynaptique de la dendrite où, il provoque la naissance d'un potentiel appelé potentiel postsynaptique. Ce potentiel se propage ensuite le long de la dendrite vers le soma du neurone.

1-2-3 LE NEURONE FORMEL :

Le premier modèle du neurone formel date des années quarante. Il a été présenté par Mc Culloch et Pitts. S'inspirant de leurs travaux sur les neurones biologiques ils ont proposé le modèle suivant [1,3] :
Un neurone formel fait une somme pondérée des potentiels d'action qui lui parviennent. Puis s'active suivant la valeur de cette sommation pondérée. Si cette somme dépasse un certain seuil, le neurone est activé et transmet une réponse dont la valeur est celle de son activation. Si le neurone n'est pas activé, il ne transmet rien.

D'une façon plus générale, un neurone formel est un élément de traitement possédant n entrées $X_1, X_2, \ldots, X_n$ (qui sont les entrées externes ou les sorties des autres neurones),et une seule sortie. Son traitement consiste à affecter à sa sortie Y_j le résultat d'une fonction de seuillage f (dite aussi fonction d'activation) de la somme pondérée.

$$Y_j = f\left(\sum_{i=1}^{n} W_{ij} X_i\right) \qquad (1.1)$$

Où W_{ij} est la pondération (coefficient synaptique ou poids)

associée à la $i^{ème}$ entrée du neurone j.

Parfois, il y a un terme additionnel θ_j représentant le seuil interne du neurone, ce terme est considéré comme un poids w_{0j} associé à une entrée constante (fig.(1-2)). L'expression (1.1) devient donc :

$$y_j = f(\sum_{i=1}^{n} W_{ij}X_j - \theta_j) \qquad (1.2)$$

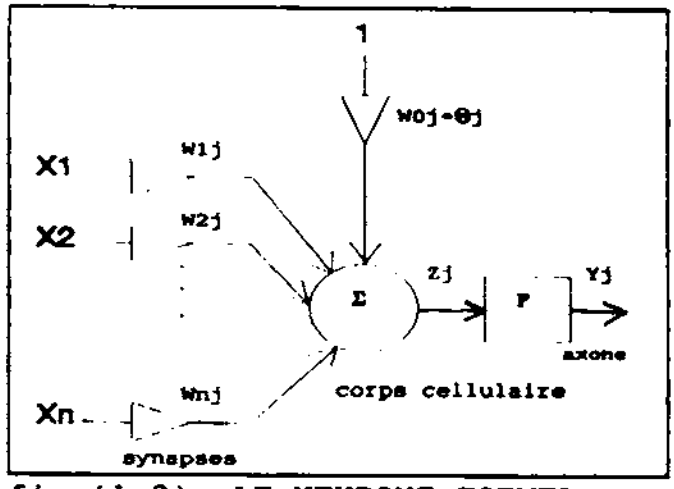


fig.(1-2): LE NEURONE FORMEL

Les fonctions d'activation représentent généralement certaines formes de non linéarité. L'une des formes de non linéarité la plus simple, et qui est appropriée aux réseaux discrets, est la fonction signe (fig.(I-3-a)).

$$F(z_j) = \begin{cases} 1 & si \quad z_j > 0 \\ -1 & si \quad z_j < 0 \end{cases} \qquad (1\text{-}3)$$

Une autre variante de ce type des non linéarités est la fonction de Heaviside (fig.(1-3-b)):

$$F(z_j) = \begin{cases} 1 & si \quad z_j > 0 \\ 0 & si \quad z_j < 0 \end{cases} \qquad (1\text{-}4)$$

La classe, la plus utilisée des fonctions d'activation, et qui est plus appropriée aux réseaux analogiques, est celle des fonctions sigmoides. Un exemple de ces fonctions est illustré par la (fig.(1-3-c)):

$$F(z_j) = \frac{(1-\exp(-\beta z_j))}{(1+\exp(-\beta z_j))} \qquad (1\text{-}5)$$

Cette fonction varie dans l'intervalle [-1 1], lorsque z_j varie entre ($-\infty$ et $+\infty$), de plus cette fonction est différentiable. La constante β détermine la raideur de la région de transition.

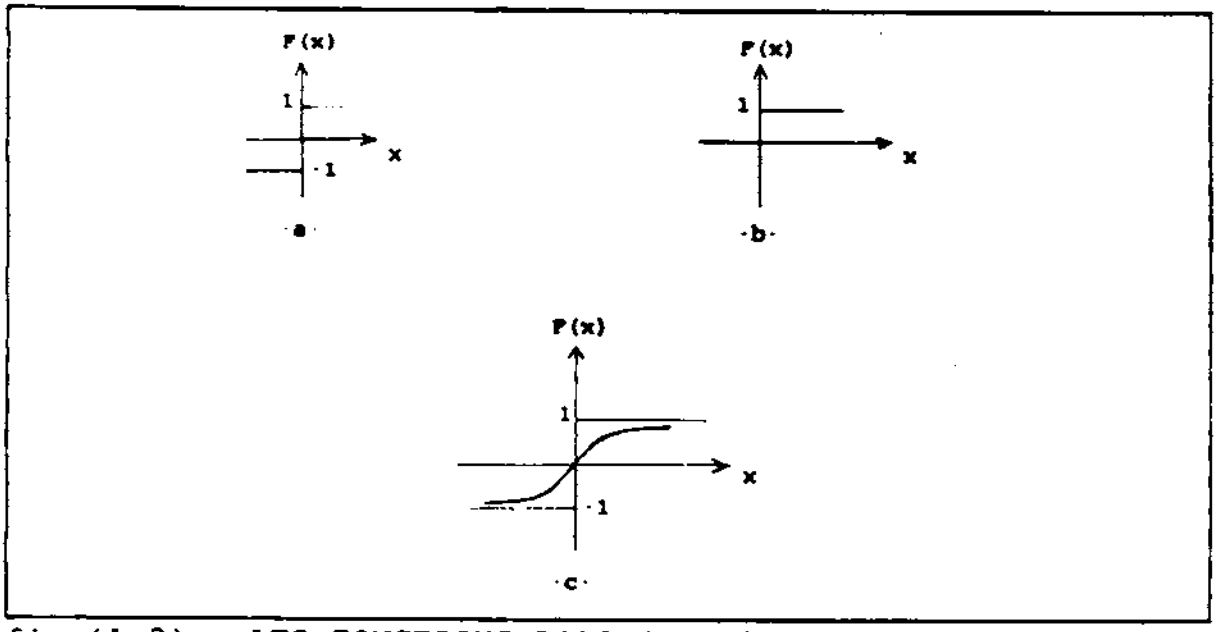


fig.(1-3) : LES FONCTIONS D'ACTIVATION LES PLUS UTILISEES
a) fonction signe, b) fonction de Heaviside
c)fonction sigmoide.

D'autres types de modèles des neurones sont étudiés dans le connexionisme, cependant, comme le neurone formel de Mc Culloch et Pitts est celui sur lequel reposent la plupart des réseaux actuels, nous nous limiterons à celui-ci.

1-3 DEFINITION ET PROPRIETES DES RESEAUX DE NEURONES :

Un réseau de neurones est un ensemble d'éléments de traitement de l'information, avec une topologie spécifique

d'interconnexions entre ces éléments et une loi d'apprentissage pour adapter les poids de connexions, il est caractérisé par un parallélisme à grain très fin et à forte connectivité. Nous entendons par là que dans un réseau de neurones donné, l'information est traitée par un grand nombre de processeurs élémentaires très simples, chacun étant relié à un nombre très important d'autres processeurs. En général le processeur très simple est un neurone formel désigné ainsi car son fonctionnement s'inspire d'une modélisation des cellules neuronales biologiques[1,2,4]

D'une manière général un réseau de neurones possède les propriétés suivantes [3,8]:

Le parallélisme : Cette notion se situe à la base de l'architecture des réseaux de neurones considérés comme ensembles d'entités élémentaires qui travaillent simultanément.

La capacité d'adaptation : Celle-ci se manifeste tout d'abord dans les réseaux de neuronnes par la capacité d'apprentissage qui permet au réseau de tenir compte des nouvelles contraintes ou de nouvelles données du monde extérieur. De plus elle se caractérise dans certains réseaux par leur capacité d'auto-organisation qui assure leur stabilité en tant que systèmes dynamiques.

La mémoire distribuée : Dans les réseaux de neurones, la mémoire d'un fait correspond à une carte d'activation des neurones. Cette carte est en quelque sorte un codage du fait mémorise.

La résistance aux pannes : A cause de l'abondance des entrées et la structure du réseau, les données bruitées ou les pannes locales dans certain nombre de ses éléments n'affectent pas ses fonctionnalités. Cette propriété résulte essentiellement du fonctionnement collectif et simultané des neurones qui le composent.

La généralisation : La capacité de généralisation d'un réseau de neurones est son aptitude de donner une réponse satisfaisante à une entrée qui ne fait pas partie des exemples à partir desquels il a appris.

1-4 STRUCTURES DE CONNEXIONS :

1-4-1 CAS GENERAL :

Les structures qui peuvent être utilisées sont très variées. Si l'on se réfère aux études biologiques du cerveau, on constate que le nombre de connexions est énorme. Par exemple, des chercheurs ont montré que le cortex était divisé en différents couches. A l'intérieur d'une même couche les interactions entre neurones sont très grands, les neurones d'une couche sont aussi reliés aux neurones d'autres couches, le tout formant un système d'une complexité gigantesque.

d'une manière générale, l'architecture des réseaux de neurones formels peut aller d'une connectivité totale (tous les neurones sont reliés les uns aux autres), à une connectivité locale où les neurones ne sont reliés qu'à leurs proches voisins.

1-4-2 LES RESEAUX A COUCHES :

Dans les réseaux à couches, les neurones qui appartiennent à une même couche ne sont pas connectés entre eux, chacune des couches recevant des signaux de la couche précédente, et transmettant le résultat de ses traitements à la couche suivante [3,4]. Les couches extrêmes correspondent à la couche qui reçoit ses entrées du milieu extérieur, cette couche est appelée couche d'entrée, d'une part, et la couche qui fournit les résultats des traitements effectués, on appelle cette couche, couche de sortie, d'autre part. Les couches internes sont appelées couches cachées (c'est-à-dire non reliées au monde extérieur), leur nombre est variable. En général la couche d'entrée est une couche passive, leur neurones n'effectuent aucun traitement, leur rôle étant simplement, de recevoir et transmettre les configurations à mémoriser.

1-4-3 LES RESEAUX ENTIEREMENT CONNECTES :

Dans ce type de réseaux, chaque neurone est reliée à tous les autres et passe de même un retour sur lui [7]. L'exemple typique de cette structure est le réseaux de kohonen [1,10]. Ce réseau est structuré sous forme d'une matrice de neurones

complètement connectés. Chaque neurone possède deux ensembles de poids: un ensemble pour calculer la somme des entrées externes pondérées, et l'autre pour contrôler les interactions entre les neurones du réseau.

1-5 APPRENTISSAGE :

Le besoin d'apprentissage se manifeste lorsque l'information a priori est incomplète, et son type dépend du degré de plénitude de cette information [7]. Comme l'information que peut acquérir un réseau de neurones est représentée dans les poids des connexions entre les neurones, l'apprentissage consiste donc à ajuster ces poids de telle façon que le réseau présente certains comportements désirés. Mathématiquement l'apprentissage est défini par [4]:

$$\frac{\partial W}{\partial t} \neq 0 \qquad (1\text{-}6)$$

où W est la matrice des poids .

Du point de vue source d'inspiration, on peut considérer qu'il existe deux grandes familles de règles d'apprentissage [3]: La première est de sources biologiques et elle correspond à la règle de Hebb et aux modèles développés par des chercheurs, qui veulent modéliser les systèmes visuels animaux. La deuxième famille est de sources mathématiques et elle est basée sur des minimisations de fonctions de coûts ou sur des techniques d'algèbre linéaire. Cette démarche a abouti à des algorithmes comme celui du perceptron ou celui de la rétropropagation.

D'une manière plus générale, on peut distinguer trois types d'apprentissage [4,5]: apprentissage supervisé, apprentissage non supervisé et apprentissage par renforcement.

1-5-1 APPRENTISSAGE SUPERVISE :

Ce type d'apprentissage nécessite que la réponse désirée du système a entraîner soit connue a priori (présence d'un maître qui fournit la réponse désirée), et il est effectué de la façon suivante [5] : on présente au réseau les valeurs d'entrée et on calcule sa sortie actuelle correspondante, ensuite les poids sont

ajustés de façons à réduire l'écart entre la réponse désirée et celle du réseau en utilisant l'erreur de sortie (la différence entre la réponse du réseau et celle désirée). Cette procédure est répétée jusqu'à ce qu'un critère de performance soit satisfait. Une fois la procédure d'apprentissage est achevée, les coefficients synaptiques prennent des valeurs optimales au regard des configurations mémorisées et le réseau peut être opérationnel.

1-5-2 APPRENTISSAGE NON SUPERVISE :

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

1-5-3 APPRENTISSAGE PAR RENFORCEMENT :

L'idée de base de l'apprentissage par renforcement est inspirée des mécanismes d'apprentissage chez les animaux. Dans ce type d'apprentissage on suppose qu'il n'existe pas de maître (superviseur) qui peut fournir la réponse correcte, mais le système à entraîner est informé, d'une manière indirecte, sur l'effet de son action choisie. Cette action est renforcée si elle conduit à une amélioration des performances du système entraîné, et les éléments qui contribuent dans la génération de cette action sont soit récompensés ou punis [12,13].

1-6 CONCLUSION :

Les résultats récents sur les réseaux de neurones couronnent une longue série de modélisations mathématiques des systèmes biologiques de traitement des données. Nous avons vu, dans ce chapitre, que selon la nature des connexions, plusieurs architectures des réseaux de neurones peuvent être obtenues et différents types d'apprentissage peuvent être utilisés. Le prochain chapitre traite, d'une façon particulière, deux architectures des réseaux de neurones, à savoir les réseaux MLP et les réseaux RBF.

CHAPITRE II

Architectures des réseaux de neurones

ARCHITECTURES DES RESEAUX DE NEURONES

2-1 INTRODUCTION :

L'évolution d'un réseau de neurones dépend des interactions entre ces constituants; ces interactions sont définies par les poids des connexions entre les divers neurones. Selon la nature des ces connexions on distingue deux groupes des réseaux de neurones: les réseaux statiques et les réseaux dynamiques. Dans ce chapitre deux classes des réseaux statiques à savoir les réseaux MLP (multilayered perceptron) et les réseaux RBF (radial basis functions) seront étudiés, ainsi que leurs algorithmes d'apprentissage, et une brève description des réseaux dynamiques sera donnée.

2-2 RESEAUX STATIQUES :

Dans les réseaux statiques, dits aussi réseaux non récurrents, les sorties dépendent des entrées actuelles, et non des entrées ou sorties passées. La sortie de n'importe quel neurone ne peut pas être appliquée directement sur son entrée ni indirectement à travers d'autres neurones. Dans ce type de réseaux, lorsqu'un stimulus est présenté en entrée du réseau, la réponse de ce dernier est calculée instantanément, après la traversée de ses couches, de l'entrée vers la sortie (dans un sens unique).

Un réseau statique réalise une transformation non linéaire de la forme [8] :

$$Y=F(X) \qquad (2\text{-}1)$$

Où $X \in R^n$, $Y \in R^m$ sont les vecteurs d'entrée et de sortie.

Le premier modèle des réseaux statiques mono-couche fut présenté par Rosenblatt en 1958, c'était le perceptron [3,8,9].

Ce modèle utilise comme élément de base le neurone de Mc-Culloch dont la fonction d'activation est celle de Heaviside [1].

Le deuxième modèle est l'Adaline (Adaptive Linear Element) de B.Widrow et M.Hoff [1]. ce modèle est composé d'un seul neurone de type Mc-Culloch dont la fonction d'activation est linéaire. L'association de plusieurs perceptrons (une couche des perceptrons) avec des portes logiques forme ce qu'on appelle Madaline (Many Adalines) [1,6].

Deux règles d'apprentissage adaptées aux deux modèles précédents ont été développées. L'une d'elles, due à B.widrow et M.Hoff est appelée aujourd'hui règle de Widrow-Hoff (ou règle Delta). Cette règle réalise une approximation de la descente du gradient [6]. L'autre, due à F.Rosenblatt est appelée règle du perceptron [14]. Ces deux règles permettent de soumettre les réseaux mono-couche à ce qu'on appelle un apprentissage supervisé.

Quoi qu'ils en soient, ces réseaux ne pouvaient résoudre que des problèmes simples de classification. Pour des problèmes complexes. une solution consiste à organiser le réseau en plusieurs couches.

2-2-1 RESEAUX MULTI-COUCHES:

Les réseaux multi-couches, appelés aussi réseaux MLP (une abréviation de multilayered perceptron, c'est-à-dire perceptron multi-couches), consistent à cascader un certain nombre de perceptrons en plusieurs couches [4,8,14]. Le perceptron utilisé dans ces réseaux est aussi appelé neurone ou noeud, et il diffère de celui de Rosenblatt par la fonction d'activation qui est, dans ce cas, une fonction sigmoide au lieu de celle de Heaviside.

Les réseaux MLP sont organisés de la façon suivante (fig.(2-1)): une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Les neurones des deux couches consécutives sont entièrement connectés. Comme la couche d'entrée est passive(elle reçoit seulement les entrées), Le nombre total des couches d'un réseau est en général donné par celui des couches cachées et de la couche de sortie. La figure (2-1) représente un réseau MLP à trois couches: deux couches cachées et une couche de sortie.

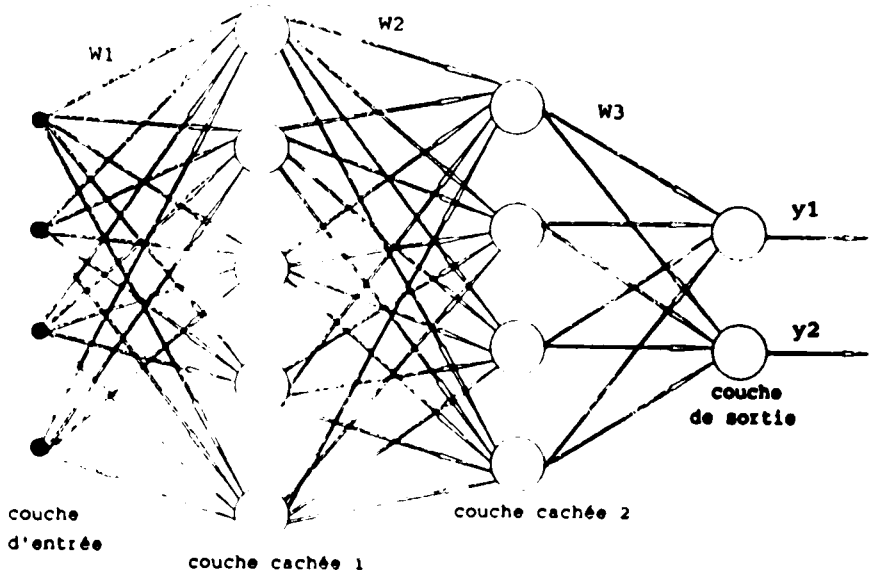


fig.(2-1): Structure d'un réseau MLP

Même si les chercheurs avaient songé à des telles architectures "multi-couches" celles-ci étaient restées inutilisables car on ne connaissait pas de règle d'apprentissage adaptée. En effet, l'utilisation directe des règles d'apprentissage du perceptron ou de Widrow-Hoff, pour ajuster les poids était impossible à cause de l'existence des neurones cachés. Puis une généralisation de la règle de Widrow-Hoff, pour ce type de réseaux, connue sous le nom de l'algorithme de la rétropropagation a permis de surmonter cet obstacle.

2-2-1-1 EQUATIONS DU RESEAU :

Un réseau MLP non récurrent à n entrées et m sorties, réalise une application de R^n dans R^m. Cette application résulte de la composition des applications locales réalisées par les couches du réseau [16]. Supposons que le réseau est composé de L couches (L-1 couches cachées plus une couche de sortie), les sorties des neurones de la couche K sont données par les équations suivantes:

$$y_j^k(t) = F(z_j^k(t)) \quad j=1,..,n_k \qquad (2\text{-}2)$$

où $k=1,2,..,L$ est l'indice des couches, n_k le nombre des neurones correspondant et F est la fonction d'activation choisie, cette

fonction est généralement, celle donnée par l'équation (1-5).

$$z_j^k(t) = \sum_{i=0}^{n_{k-1}} W_{ij}^k \, y_i^{k-1}(t) \qquad (2-3)$$

$y_0^k(t) = 1$ $\quad k = 1,2,..,L$

$y_i^0(t) = x_i(t)$ $\quad i = 1,2,..,n$ sont les composantes du vecteur d'entrée.

$y_i^L(t) = y_i(t)$ $\quad i = 1,2,..,m$ sont les composantes du vecteur de sortie.

$y_j^k(t)$ est la sortie du neurone j de la couche k et w_{j0} son seuil interne. w_{ji}^k est le poids de la connexion entre le neurone j de la couche k et le neurone i de la couche k-1.

2-2-1-2 METHODE DE RETROPROPAGATION :

La rétropropagation est l'algorithme d'apprentissage supervisé le plus utilisé pour ajuster les poids d'un réseau MLP. L'idée simple qui est à la base de cette méthode, est l'utilisation d'une fonction dérivable (fonction sigmoide) en remplacement de la fonction de seuil utilisée dans le modèle original de Rosenblatt.

Mathématiquement, cette méthode est basée sur l'algorithme de la descente du gradient, et utilise simplement les règles de dérivation composée. Dans cette méthode, de même que l'on est capable de propager un signal provenant des cellules d'entrée vers la couche de sortie, on peut, en suivant le chemin inverse, rétropropager l'erreur commise en sortie vers les couches cachées d'où le nom rétropropagation.

PRINCIPE DE LA METHODE :

L'apprentissage utilise le même principe que la règle de Widrow-Hoff. On dispose d'un ensemble d'exemples qui sont des couples (entrées-sorties). A chaque étape, un exemple est présenté en entrée du réseau. Une sortie réelle Y(t) est calculée en utilisant les équations (2-2),(2-3). Ensuite l'erreur de sortie e(t) est rétropropagée dans le réseau. Ceci permet de calculer les erreurs sur chaque neurone dans les couches cachées.

Les poids des connexions sont ajustés de telle sorte à minimiser la fonction de coût donnée par:

$$J(W) = \frac{1}{2}\sum_{p=1}^{N} J_p(W) \qquad (2\text{-}4)$$

Où N est le nombre des exemples d'entraînement et $J_p(w)$ est le carré de l'erreur associée au p^{ème} exemple, et qui est donnée par la relation suivante:

$$J_p(W) = [\,Y(X_p) - Y^d(X_p)\,]\,[\,Y(X_p) - Y^d(X_p)\,]^T \qquad (2\text{-}5)$$

Où $Y^d(X_p)$ est le vecteur de sortie désiré, $Y(X_p)$ celui du réseau et X_p le p^{ème} échantillon d'entraînement. La loi d'adaptation des poids est donnée par [1]:

$$W_{ij}^k(t) = W_{ij}^k(t-1) - \mu\sum_{p=1}^{N} \frac{\partial J_p(W)}{\partial W_{ij}^k(t)}$$

Dans le cas d'un apprentissage récursif on utilise la loi d'adaptation suivante:

$$W_{ij}^k(t) = W_{ij}^k(t-1) - \mu \frac{\partial J_p(W)}{\partial W_{ij}^k(t)} \qquad (2\text{-}6)$$

Où μ est une constante positive appelée taux ou pas d'apprentissage, et t est l'indice des itérations. La dérivée partielle de $J_p(w)$ par rapport à chaque poids est donnée par:

$$\frac{\partial J_p(W)}{\partial W_{ji}^k} = \frac{\partial J_p(W)}{\partial y_j^k(t)} \frac{\partial y_j^k(t)}{\partial W_{ji}^k} \qquad (2\text{-}7)$$

Où:

$$\frac{\partial y_j^k(t)}{\partial W_{ji}^k} = F'\left(\sum_{m=0}^{n_{k-1}} W_{jm}^k y_m^{k-1}\right) y_i^{k-1} \qquad (2\text{-}8)$$

Le terme $\frac{\partial J_p(W)}{\partial y_j^k(t)}$ représente la sensibilité de $J_p(W)$ par rapport à la sortie $y_j^k(t)$, et elle est donnée par l'expression [8]:

$$\frac{\partial J_p(W)}{\partial y_j^k} = \sum_{m=1}^{n_{k+1}} \frac{\partial J_p(W)}{\partial y_m^{k+1}} F'\left(\sum_{q=0}^{n_k} W_{mq}^{k+1} y_q^k\right) W_{mj}^{k+1} \qquad (2\text{-}9)$$

Ce processus peut être continu jusqu'à ce qu'on arrive à la couche de sortie. Pour cette couche la sensibilité est donnée par:

$$\frac{\partial J_p(W)}{\partial y_j^k} = y_j^L - y_j^d \qquad (2\text{-}10)$$

l'expression (2-10) est appelée erreur de sortie, et l'expression (2-9) est souvent appelée erreur des couches cachées ou erreur équivalente.

Ce processus est répété, en présentant successivement chaque exemple d'entraînement. Si pour tous les exemples l'erreur de sortie est inférieure à un seuil choisi, on dit alors que le réseau a convergé [15].

2-2-1-3 LES DIFFICULTES DES MLP :

Bien que les réseaux MLP aient prouvé leur efficacité pratique dans de nombreux problèmes, ils présentent un certain nombre des difficultés encore non résolues:

Architecture du réseau :

Il n'existe pas de résultat théorique, ni même de règle empirique satisfaisante, qui permet de dimensionner correctement un réseau en fonction du problème à résoudre [3]. Faut-il utiliser une ou deux couches cachées? combien doit-il y avoir de neurones sur la ou les couches cachées?

Convergence de l'algorithme de la rétropropagation :

Le problème de minimisation que la méthode de la rétropropagation tente de résoudre n'est pas simple [3,20]. En

effet, la surface de la fonction d'erreur présente des caractéristiques peu satisfaisantes pour effectuer une descente de gradient; minima locaux, qui empêchent la convergence de l'algorithme, plateau où les pentes sont très faibles,etc...

Dans sa version complète, l'algorithme comporte un certain nombre de paramètres, qui sont difficile à régler, comme par exemple le pas du gradient (taux d'apprentissage). Il est clair que ce dernier paramètre a une importance comme dans toute descente de gradient. S'il est très faible, la convergence risque d'être très lente. S'il est trop élevé, un problème d'oscillation peut survenir.

Pour remédier à ces problèmes, plusieurs techniques ont été proposées. Yamada et Yabuta [18] ont proposé une méthode pour ajuster la pente de la fonction sigmoide utilisée. Ceci permet d'améliorer les performances et d'accélérer la convergence de l'algorithme de la rétropropagation. Afin de dimensionner correctement le réseau, R.Azimi. Sadjadi et al [19] ont développé la technique de création dynamique des neurones sur la ou les couches cachées. Dans le but d'éviter les problèmes d'oscillation, certains auteurs ont proposé de modifier la loi d'apprentissage donnée par l'expression (2-6) en lui ajoutant un autre terme appelé moment. La loi (2-6) devient alors :

$$W_{ij}^{k}(t) = W_{ij}^{k}(t-1) - \mu \frac{\partial J_p(W)}{\partial W_{ij}^{k}(t)} + \alpha \left[W_{ij}^{k}(t-1) - W_{ij}^{k}(t-2) \right] \qquad (2\text{-}11)$$

Où $0 \leq \alpha < 1$.

2-2-2 LES RESEAUX RBF :

Contrairement aux réseaux MLP (qui sont non linéaires par rapport aux poids des connexions), les réseaux RBF (une abrévation de radial basis functions; c'est-à-dire fonctions radiales de base) sont linéaires par rapport aux paramètres (poids des connexions) à estimer. Par conséquent ces paramètres sont ajustés en utilisant les techniques d'optimisation linéaire, ceci constitue l'avantage essentiel de ces réseaux.

2-2-2-1 STRUCTURE D'UN RESEAU RBF :

Un réseau RBF (fig.(2-2)) comporte deux couches de neurones [8,21,22]; une couche cachée dont les neurones sont

connectés à ceux de la couche d'entrée par des connexions non pondérées, et une couche de sortie dont les neurones sont connectés à ceux de la couche cachée par des connexions pondérées. Les sorties z_j des neurones cachés sont données par l'expression suivante:

$$z_j = \phi(\|X-c_j\|, \rho_j) \qquad j=1,\ldots,n_h \qquad (2\text{-}12)$$

où ϕ est une fonction de base non linéaire et c_j son centre. ρ_j est une constante positive associée à ϕ, et n_h est le nombre des neurones (ou des RBFs) sur la couche cachée.

La réponse y_i de chaque neurone de la couche de sortie est une combinaison linéaire des z_j. En effet y_i est donnée par:

$$\begin{aligned} y_i &= \sum_{j=1}^{n_h} \eta_{ij} z_j \\ &= \sum_{j=1}^{n_h} \eta_{ij} \phi(\|X-c_j\|, \rho_j) \qquad i=1,\ldots,m \end{aligned} \qquad (2\text{-}13)$$

où m est le nombre des neurones sur la couche de sortie et η_{ij} est le poids de la connexion entre le $i^{ème}$ neurone de la couche de sortie et le $j^{ème}$ neurone de la couche cachée.

certains choix typiques des fonctions de base sont donnés par les expressions suivantes:

$$\phi(z,1) = z^2 \log(z) \qquad (2\text{-}14)$$

$$\phi(z,1) = (z^2+\rho^2)^{\frac{1}{2}} \qquad (2\text{-}15)$$

$$\phi(z,\rho) = \frac{1}{(z^2+\rho^2)^{\frac{1}{2}}} \qquad (2\text{-}16)$$

$$\phi(z,\rho) = \exp(-z^2/\rho^2) \qquad (2\text{-}17)$$

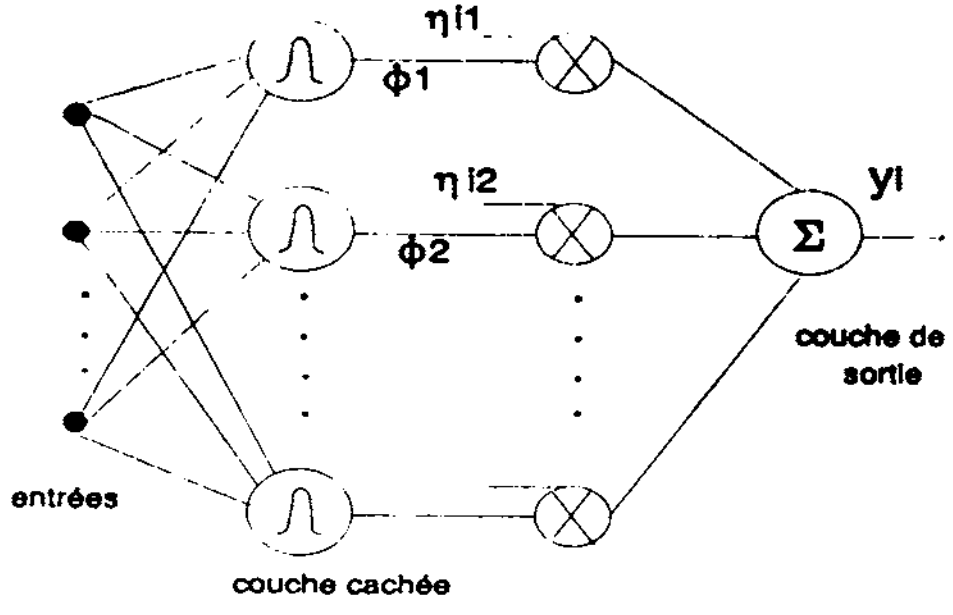


fig.(2-3): Structure d'un réseau RBF

2-2-2-2 L'APPRENTISSAGE DANS LES RESEAUX RBF :

La plupart des algorithmes d'apprentissage des réseaux RBF procèdent à une sélection aléatoire d'un certain nombre d'échantillons de l'espace d'entrée comme étant les centres des fonctions de base. Ensuite une méthode d'optimisation linéaire est utilisée pour déterminer les poids des connexions. Cependant, dans la plupart des cas, il est évident qu'une sélection aléatoire des centres n'est pas satisfaisante. Pour remédier à cet inconvénient, S.Chen et al [22] ont proposé un autre

algorithme basé sur la méthode des moindres carrés orthogonaux. Cet algorithme effectue une sélection un par un des centres à partir de l'ensemble des données, jusqu'à ce qu'un réseau adéquat soit construit.

Bien que cet algorithme permet de déterminer, d'une manière automatique, le nombre des RBFs (nombre des neurones sur la couche cachée) nécessaires au problème considéré, son utilisation dans des applications en temps réel ou dans une identification récursive est impossible, car il nécessite la disposition a priori de toutes les données entrées-sorties.

Dans d'autres algorithmes, la procédure d'apprentissage est divisée en deux étapes: un apprentissage non supervisé dans la couche cachée suivi par un apprentissage supervisé dans la couche de sortie [8]. L'algorithme des **K-means** et la méthode des moindres carrés sont souvent utilisées pour la sélection des centres et l'adaptation des poids des connexions, respectivement. C'est cette approche que nous avons adopté pour entraîner le modèle neuronal lorsque une structure RBF est utilisée.

A: ALGORITHME DES K-MEANS :

L'algorithme des K-means est une procédure de classification (Clustring) non supervisée qui consiste à partitionner l'espace d'entrée en K classes [8,23]. Chaque classe est représentée par un vecteur de paramètres appelé **centre.**

Cet algorithme est basé sur la minimisation d'un critère de performances défini comme étant la somme des distances quadratiques entre les échantillons d'une classe et son centre:

$$J_i = \sum_{X \in S_i} \|X - C_i\|^2 \qquad i = 1, \ldots, K \tag{2-18}$$

où S_i représente l'ensemble des échantillons appartenant à la classe i et C_i son centre. Le centre C_i qui minimise ce critère est tout simplement l'échantillon moyen de S_i , il est donné par:

$$c_i = \frac{1}{N_i} \sum_{X \in s_i} X \qquad i=1,2,\ldots,K \tag{2-19}$$

où N_i est le nombre d'échantillons de la classe i.

Une version de cet algorithme permettant d'ajuster, d'une manière récursive, les centres des fonctions de base d'un réseau RBF est donnée comme suit [33]:
Pour une itération t donnée, les distances Euclidéennes entre les centres et le vecteur d'entrée appliqué sont calculées:

$$d_j(t) = \| X(t) - c_j(t-1) \| \qquad 1 \leq j \leq n_h \tag{2-20}$$

Ensuite le centre correspondant à la distance minimale $d_k(t)$ avec:

$$K = arg[\min \{ d_j(t), 1 \leq j \leq n_h \}] \tag{2-21}$$

est ajusté en utilisant la loi d'adaptation suivante:

$$c_K(t) = c_K(t-1) + \alpha_c(t)\,[X(t) - c_k(t-1)] \tag{2-22}$$

où $\alpha_c(t)$ est le taux d'apprentissage pour les centres.

Les valeurs initiales des centres sont souvent choisies d'une façon aléatoire. Le taux d'apprentissage $\alpha_c(t)$ est inférieur à 1 et doit tendre en fonction des itérations, vers zéro. Une règle pour adapter α_c est donnée par [33]:

$$\alpha_c(t) = \frac{\alpha_c(t-1)}{(1 + inte[t/n_h])^{\frac{1}{2}}} \tag{2-23}$$

où inte[.] est la partie entière de l'argument.

B: METHODE DES MOINDRES CARRES RECURSIFS :

La méthode des moindres carrés récursifs traite une seule paire d'entrées-sorties à la fois. La structure de cet

algorithme est comme suit [24]:

$$\begin{bmatrix} nouvel\ estimé \\ des\ paramètres \end{bmatrix} = \begin{bmatrix} estimé\ précédent \\ des\ paramètres \end{bmatrix} + [gain\ d'adaptation]\ x$$
$$[fonction\ des\ mesures]\ x \begin{bmatrix} fonction\ de\ l'erreur \\ de\ prédiction \end{bmatrix}$$

B-1 MOINDRES CARRES RECURSIFS SIMPLES:

Le vecteur de sortie du réseau à un instant k est donnée par:

$$\hat{y}(k) = \phi^T(t)\theta(k-1) \qquad (2\text{-}24)$$

où $\phi=[\phi_1.....\phi_{n_h}]^T$ est le vecteur de sortie de la couche cachée.

$\theta(k)=[\theta_1(k).....\theta_m(k)]$ est la matrice des poids des connexions, avec $\theta_i(k)=[\eta_{1i}(k),.....,\eta_{1n_h}(k)]^T$ $1 \le i \le m$.

$\hat{Y}(k)$ est le vecteur de sortie du réseau de dimension m.

L'erreur de sortie est donné par :

$e(k)=Y(k)-\hat{Y}(k)$ où $Y(k)$ est le vecteur de sortie désiré.

Quant au critère à minimiser est donnée par:

$$J(\theta(k)) = \frac{1}{2}\sum_{i=0}^{k}(Y(i)-\hat{Y}(i))^2 \qquad (2\text{-}25)$$

Les équations obtenues par minimisation de $J(\theta(k))$ sont les suivantes [24]:

$$\theta(k)=\theta(k-1)+F(k)\phi(k)e(k) \qquad (2\text{-}26)$$

$$F(k) = F(k-1) - \frac{F(k-1)\phi(k)\phi^T(k)F(k-1)}{1+\phi^T(k)F(k-1)\phi(k)} \qquad (2\text{-}27)$$

avec $F(k)$ représente le gain d'adaptation.

La convergence de l'algorithme est garantie si la séquence $\phi(k)$ vérifie la condition d'excitation persistante, cela est vrai si pour une constante entière d donnée et pour tout k, il existe deux constantes a > 0 et b > 0 tel que [24]:

$$aI \geq \sum_{j=k}^{k+d} \phi(j)\phi^T(j) \geq bI \qquad (2\text{-}28)$$

Du fait que le gain d'adaptation $F(k)$ décroit et donne de moins en moins de faibles poids aux nouvelles erreurs donc aux nouvelles mesures, l'algorithme ne peut convenir que pour l'identification des systèmes stationnaires, et non pas pour ceux à paramètres variables.

B-2 MOINDRES CARRES RECURSIFS PONDERES :

Cet algorithme est adapté à l'identification des systèmes à paramètres variables en accordant plus de poids aux dernières mesures.

Le facteur d'oublie à ajouter est de type:

$$a(i,k) = \lambda^{k-i} \qquad 0<\lambda<1 \qquad (2\text{-}29)$$

et le critère à minimiser aura pour équation:

$$J(\theta(k)) = \frac{1}{2}\sum_{i=0}^{k} \lambda^{k-i}[Y(i) - \hat{Y}(i)]^2 \qquad (2\text{-}30)$$

La minimisation de J nous permet d'écrire l'algorithme suivant [24]:

$$\theta(k) = \theta(k-1) + F(k)\phi(k)e(k) \qquad (2\text{-}31)$$

$$F(k) = \frac{1}{\lambda(k)} \left[F(k-1) - \frac{F(k-1)\phi(k)\phi^T(k)F(k-1)}{\lambda + \phi^T(k)F(k-1)\phi(k)} \right] \qquad (2\text{-}32)$$

$$e(k) = Y(k) - \hat{Y}(k) \qquad (2\text{-}33)$$

$$\lambda(k) = \lambda_0 \lambda(k-1) + 1 - \lambda_0 \qquad (2\text{-}34)$$

En absence d'informations initiales sur les paramètres à estimer, on choisi un gain d'adaptation initial grand (la valeur typique est de 1000I), par contre si on dispose d'une estimation initiale des paramètres on choisit un gain initial faible (≤1).

2-3 RESEAUX DYNAMIQUES :

En plus de son organisation en couches, le cerveau humain possède une autre caractéristique essentielle: sa structure permet une recirculation de l'information, de telle sorte que les décisions ne sont pas prises instantanément, mais par étapes successives [25]. Il fonctionne donc comme un système dynamique, qui n'atteint pas instantanément un état d'équilibre lorsqu'il est soumis à un stimulus extérieur. C'est justement cette propriété qui a été utilisé pour concevoir des réseaux dynamiques.

Dans un réseau dynamique, appelé aussi réseau récurrent, chaque neurone peut recevoir des informations de tous les autres neurones et leur envoit lui même des signaux. Plusieurs architectures ont été proposées dans la littérature des réseaux de neurones, cependant on se limite dans ce paragraphe aux architectures les plus utilisées dans le domaine de la modélisation et de contrôle des systèmes non linéaires.

2-3-1 RESEAUX AVEC RETOUR DE SORTIE :

Une manière simple, pour obtenir un réseau dynamique, consiste à réinjecter la sortie du réseau statique, à travers une ligne de retard, à son entrée. Cette structure particulière (fig.(2-3)) a été introduite et utilisée par Narendra [26], dans les problèmes d'identification et de contrôle des systèmes non linéaires. En effet, en utilisant cette architecture, il est possible de modéliser tous les systèmes décrit par une équation de la forme:

$$y(k)=F[u(k),u(k-1),..u(k-m);y(k-1),..y(k-n)] \quad (2\text{-}29)$$

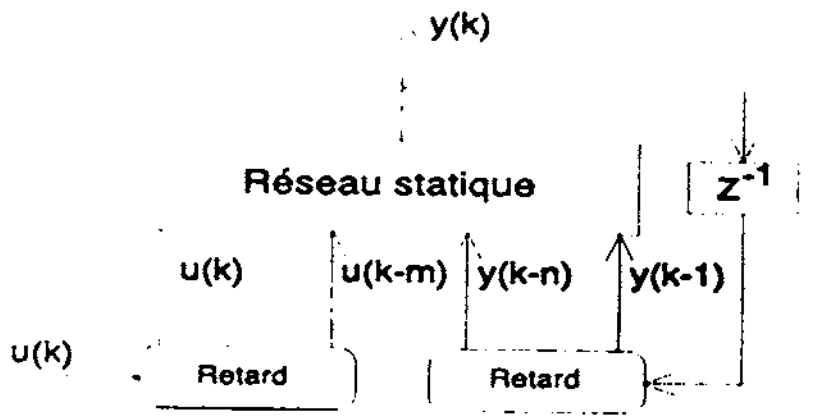


fig.(2-3): Reseau avec retour de sortie

2-3-2 RESEAUX AVEC RETOUR D'ETAT :

Ce type de réseaux possède une seule couche de neurones connectés entre eux, dans le cas le plus général les neurones sont entièrement interconnectés (c'est à dire chaque neurone est connecté à tous les autres neurones et possède un retour sur lui même). En plus chaque neurone peut recevoir une entrée externe. Cette archittecture est illustrée par la figure (2-4) [8]:

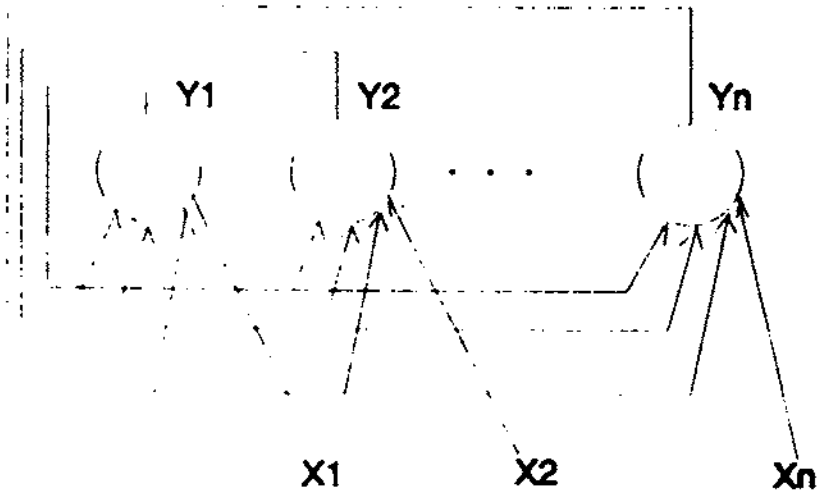


fig.(2-4): Réseau avec retour d'état

Le réseau de Hopfield a été aussi utilisé par plusieurs auteurs dans la modélisation et le contrôle des systèmes non linéaires. Pour plus de détails se référer à [8,9,29].

2-4 CONCLUSION :

Nous avons dans ce chapitre présenté deux structures des réseaux de neurones: les réseaux MLP et les réseaux RBF. Les algorithmes d'apprentissages adaptés à ces structures ont été étudiés en détail.

Les réseaux MLP présentent une réponse non linéaire par rapport aux poids des connexions, par conséquent l'apprentissage dans ce type de réseaux est basé sur des algorithmes d'optimisation non linéaires (les méthodes du gradient), dans ce cas, la convergence de l'algorithme de la rétropropagation, qui réalise une approximation de la descente du gradient vers le minimum global de la surface d'erreur, n'est pas garantie.

Du fait que les réseaux RBF présentent une réponse linéaire par rapport aux poids des connexions, ils peuvent fournir une solution au problème de convergence de l'algorithme de la rétropropagation. Dans ce cas, il existe des théorèmes qui garantissent la convergence de l'algorithme des moindres carrés.

Enfin, nous avons introduit, d'une manière brève la notion de réseaux dynamiques. Le chapitre prochain traite le problème d'identification des systèmes non linéaires en utilisant les deux architectures (MLP et RBF) déjà présentées.

CHAPITRE III

Identification avec les réseaux statiques

IDENTIFICATION AVEC LES RESEAUX STATIQUES

3-1 INTRODUCTION :

Bien que plusieurs techniques d'identification ont été établies pour les systèmes linéaires, des méthodes générales et efficaces qui peuvent être utilisées pour identifier un système non linéaire ne sont pas encore disponibles. Ceci est probablement dû à la complexité inhérente de ces systèmes et à la difficulté de trouver des algorithmes qui peuvent être appliqués sur des classes suffisamment larges des systèmes non linéaires.

Plusieurs techniques classiques pour l'identification de certaines classes de systèmes non linéaires sont disponibles dans la littérature [28]. Ces techniques peuvent être regroupées de la façon suivante:

- Méthodes d'estimation des paramètres pour des structures particulières [29];
- Méthodes heuristiques (GMDH : Group Method of Data Handling) [30];
- Méthodes basées sur les séries de Volterra et Wienner [32].

L'application des techniques neuronales pour l'identification des systèmes non linéaires peut fournir des nouvelles solutions pour ce problème. En effet, les capacités d'identification des réseaux de neurones, en particulier ceux de type multi-couches, leur aptitude à la généralisation et leur adaptativité, nous conduisent aujourd'hui à étudier et développer des architectures modulaires à base de réseaux de neurones en identification.

Le but de ce chapitre est de présenter la résolution des problèmes d'identification par les techniques neuronales en utilisant deux architectures différentes des réseaux de neurones, et de montrer l'apport de cette approche dans les problèmes d'identification des systèmes non linéaires dynamiques.

3-2 CONCEPTS DE BASE :

L'identification d'un processus consiste à déterminer un modèle capable, lorsqu'il est soumis aux mêmes entrées que le processus de restituer une sortie suffisamment proche du celle du processus, au sens d'un certain critère [16]. Souvent, la construction de ce modèle est effectuée en deux étapes:

- La première étape est qualitative (caractérisation), et elle consiste à fixer les formes des équations qui décrivent le processus.

- La deuxième étape est quantitative (estimation des paramètres) et elle consiste à déterminer les valeurs numériques des coefficients qui interviennent dans ces équations en minimisant un critère de mesure.

3-2-1 ETAPE QUALITATIVE :

Pour définir la structure du modèle on peut s'aider des lois de la physique. Les équations ainsi établies constituent ce qu'on appelle le "modèle de connaissance" (modélisation) [24]. Ce cas est rarement rencontré en pratique.

L'autre cas extrême est celui où on ignore tout ou une grande partie des phénomènes mis en jeu. Dans ce cas on se contente d'une description mathématique sans lien apparent avec la réalité physique; c'est ce qu'on appelle modèle de "représentation" (identification). La structure de ce dernier modèle est fixée à priori.

En référence à la théorie des systèmes linéaires, Narendra et al [26] ont proposé les modèles suivants:

Modèle I:

$$y_s(k+1) = \sum_{i=0}^{n-1} \alpha_i y_s(k-i) + g[u(k), u(k-1), .. u(k-m+1)] \quad (3\text{-}1)$$

Modèle II:

$$y_s(k+1) = f[y_s(k), y_s(k-1), .. y_s(k-n+1)] + \sum_{i=0}^{m-1} \beta_i u(k-i) \quad (3\text{-}2)$$

Modèle III:

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

Modèle IV:

$$y_s(k+1) = f[y_s(k), y_s(k-1), .. y_s(k-n+1); u(k), u(k-1), .. u(k-m+1)] \quad (3\text{-}4)$$

Où u(k) est l'entrée du système à identifier et $y_s(k)$ sa sortie.

Les fonctions f et g sont supposées dérivables par rapport à leurs arguments.

Dans tous les modèles, la sortie du système à l'instant k+1 dépend des n valeurs passées de la sortie ainsi que des m valeurs passées de la commande, simultanément. Par la suite on ne considère que le modèle IV dans lequel $y_s(k+1)$ est une fonction non linéaire de $y_s(k-i)$ et $u(k-j)$, il est évident que les autres modèles sont des cas particuliers de ce dernier.

3-2-2 ETAPE QUANTITATIVE :

Pour déterminer les valeurs numériques des paramètres qui interviennent dans le modèle d'identification on doit définir un critère qui exprime quantitativement l'écart entre système et modèle; ce critère devra être minimisé.

L'idée de base de l'approche neuronale est de substituer aux modèles paramétriques classiques des modèles neuronaux dont les

poids des connexions (qui constituent dans ce cas, les paramètres à estimer) sont ajustés en utilisant une loi d'apprentissage appropriée [16].

Si on suppose que le système à identifier est représenté par le modèle IV, la procédure d'identification consiste alors à entraîner un réseau de neurones pour qu'il puisse approximer la fonction non linéaire f(.) représentant la dynamique directe du système sous identification.

En s'inspirant des techniques d'identification des systèmes linéaires, deux structures d'identification en utilisant l'approche neuronale ont été proposées [26,32].

3-2-2-1 MODELE SERIE-PARALLELE :

La structure générale d'identification série-parallèle d'un système non linéaire décrit par le modèle IV est donnée par la fig.(3-1) [16,26,32]. Le modèle neuronale placé en parallèle avec le système est un réseau statique dont le comportement entrée-sortie est décrit par l'expression suivante :

$$y_m(k+1) = \hat{f}[x(k)] \qquad (3\text{-}5)$$

Où x(k) est le vecteur d'entrée du réseau, $y_m(k+1)$ sa sortie et $\hat{f}(.)$ est une approximation de $f(.)$.

L'entrée du réseau (modèle) est donné par :

$$x(k) = [y_s(k), y_s(k-1), .. y_s(k-n+1); u(k), u(k-1), .. u(k-m+1)] \qquad (3\text{-}6)$$

La prise en compte de l'ordre du système s'effectue par l'intermédiaire de lignes de retard mémorisant les valeurs passées de la sortie et de la commande dont dépendra la sortie courante du système.

La différence entre la sortie y_s et la sortie y_m que le modèle neuronal prédit est utilisé à travers un algorithme d'apprentissage pour ajuster les poids des connexions du modèle.

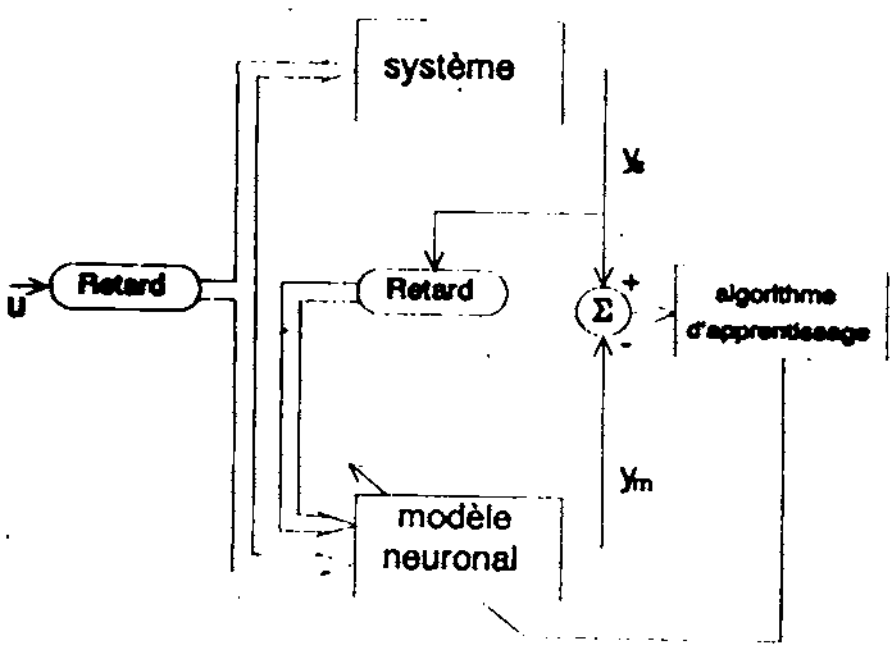


fig.(3-1): Structure d'identification série-parallèle

3-2-2-2 MODELE PARALLELE :

La fig.(3-2) représente l'architecture adaptée à l'identification parallèle d'un système décrit par le modèle IV [32]. Le modèle placé en parallèle avec le système est un réseau de neurones dont la relation entrée-sortie est donnée par l'expression suivante:

$$y_m(k+1) = \hat{f}[x(k)] \qquad (3\text{-}7)$$

Où:

$$x(k) = [y_m(k), y_m(k-1), .. y_m(k-n+1); u(k), u(k-1), .. u(k-m+1)] \qquad (3\text{-}8)$$

est le vecteur d'entrée du modèle neuronal.

Dans ce cas, la sortie courante $y_m(k+1)$ du modèle neuronal dépend des n valeurs passées de sa sortie, au lieu des valeurs passées de la sortie du système dans le cas de la structure précédente.

La différence entre la sortie du système et celle du modèle est utilisée à travers un algorithme d'apprentissage pour ajuster les paramètres du modèle.

L'utilisation de cette structure d'identification souffre des problèmes de convergence et de stabilité, par conséquent l'utilisation de la structure série-parallèle est préférable [26].

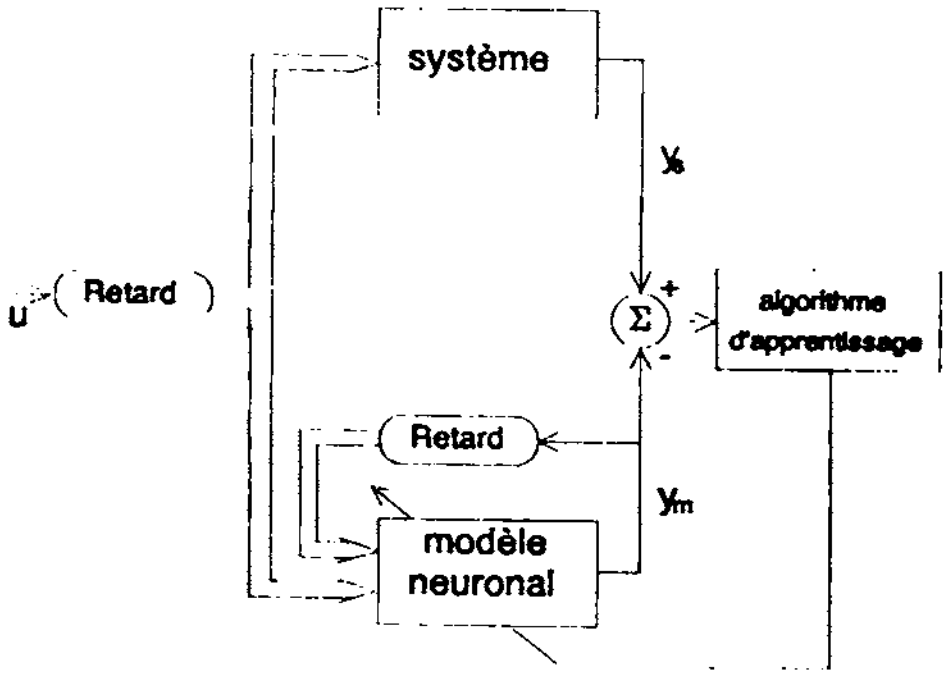


fig.(3): Structure d'identification parallèle

3-3 IDENTIFICATION PAR UN RESEAU MLP :

L'entraînement d'un réseau MLP en utilisant les entrées-sorties d'un système non linéaire peut être considéré comme un problème d'approximation de fonctions non linéaires [34]. Récemment, l'étude des capacités d'approximation des réseaux de neurones était l'objet de plusieurs travaux. Les résultats de ces travaux montrent que les réseaux MLP peuvent approcher n'importe quelle fonction continue, et qu'un réseau MLP à une seule couche cachée dont la fonction d'activation est une fonction sigmoïde est capable de réaliser cette approximation, à condition que ce réseau possède un nombre suffisant de neurones et que les poids des connexions sont correctement choisis [33].

L'identification par un réseau MLP est un problème d'apprentissage supervisé qui consiste à modifier les coefficients synaptiques des connexions du modèle neuronal en

utilisant l'algorithme de la rétropropagation, de telle sorte que la sortie du modèle approche mieux celle du système au sens d'un certain critère de mesure. Dans tous les exemples de simulation nous avons adopté la structure d'identification série-parallèle.

3-3-1 ALGORITHME D'APPRENTISSAGE :

1ère étape :

Initialiser les poids et les seuils internes des neurones par des valeurs aléatoires faibles.

2ème étape :

Présenter le vecteur d'entrée donné par l'expression (3-6) et spécifier la sortie désirée correspondante au vecteur d'entrée appliquée.

3ème étape :

Calculer la sortie courante du réseau en utilisant les expressions (2-2) et (2-3).

4ème étape :

Calculer l'erreur de sortie en utilisant l'équation (2-10), ensuite déterminer les erreurs équivalentes des neurones cachés en utilisant l'équation (2-9).

5ème étape :

Calculer les gradients du critère de performance par rapport aux poids du réseau en utilisant l'expression (2-7), ensuite ajuster les poids selon la loi donnée par l'équation (2-6).

6ème étape :

Présenter un autre vecteur d'entrée et aller à l'étape 3.

Les vecteurs d'entrée sont présentés d'une manière récursive jusqu'à ce que l'erreur de sortie se stabilise à un niveau acceptable.

3-3-2 SIMULATION :

Le modèle neuronal utilisé dans les exemples de simulation est un réseau statique à deux couches cachées. La fonction d'activation choisie pour les neurones cachés est celle donnée par l'équation (1-5) avec β=1. Les neurones de la couche de sortie possèdent une fonction d'activation linéaire. L'entraînement du modèle est réalisé en utilisant l'algorithme décrit dans le paragraphe 3-1.

Exemple 3-1:

Le système à identifier est décrit par une équation aux différence du premier ordre de la forme:

$$y_s(k+1)=f[y_s(k)]+u(k)$$

où la fonction inconnue f est donnée par:

$$f[y_s(k)]=\frac{y_s(k)}{1+y_s^2(k)} \qquad (3\text{-}8)$$

Le modèle d'identification série-parallèle utilisé pour identifier le système est décrit par:

$$y_m(k+1)=N[y_s(k)]+u(k)$$

où N[.] est un réseau MLP à une seule entrée comportant, 12 neurones sur sa première couche cachée, 8 neurones sur la deuxième et un neurone sur la couche de sortie.

L'entraînement du modèle est réalisé en utilisant les mesures entrées-sorties du système. Les commandes appliquées sont uniformément réparties dans l'intervalle [-2 2]. En considérant que $y_s(0)=0.25$, à chaque itération une commande est appliquée simultanément sur le modèle et le système, ensuite la différence entre la sortie du système et celle du modèle est utilisée pour ajuster les paramètres du réseau avec un pas d'apprentissage $\mu = 0.25$.

La sortie du modèle obtenu après 30000 itérations et celle du système pour une entrée teste donnée par:

$$u(k)=\sin(2\pi k/250) \quad \text{pour } k \le 500 \text{ et}$$
$$u(k)=0.8\sin(2\pi k/250)+0.2\sin(2\pi k/25) \quad \text{pour } k > 500 \qquad (3\text{-}9)$$

sont représentées dans la fig.(3-3).

On remarque que les deux courbes sont superposées et que l'erreur d'identification est très faible, malgré qu'on a changé la commande lorsque k > 500. La plage de variation de cette erreur est donnée par la fig.(3-4).

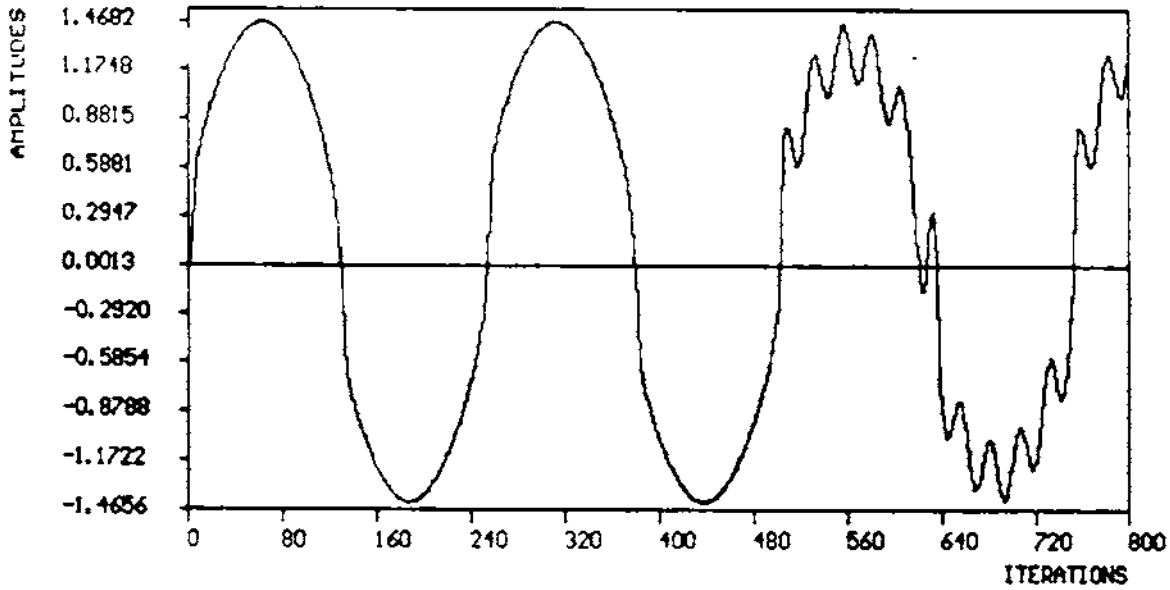


fig.(3-3): Sorties du système et celle du modèle superposées pour l'exemple 3-1.

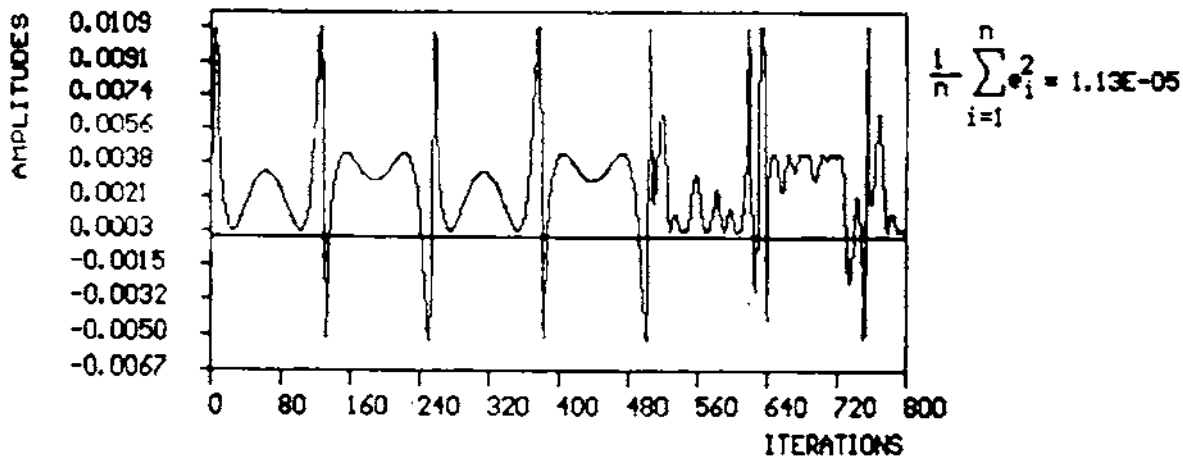


fig.(3-4): Erreur d'identification pour l'exemple 3-1.

Exemple 3-2:

Dans ce cas le système à identifier est décrit par une équation aux différences du premier ordre comportant deux linéarités séparables de la forme:

$$y_s(k+1) = \frac{y_s(k)}{1+y_s^2(k)} + u^3(k) \qquad (3\text{-}10)$$

Ceci correspond à $f[y_s(k)] = \frac{y_s(k)}{1+y_s^2(k)}$ et $g[u(k)] = u^3(k)$ dans le modèle III.

Le modèle d'identification série parallèle est constitué, cette fois, par deux réseaux MLP: N_f et N_g possédant une seule entrée et, comportant 12 neurones sur ses premières couches cachées et 8 neurones sur les deuxièmes. Ce modèle est donné par:

$$y_m(k+1) = N_f[y_s(k)] + N_g[u(k)]$$

Les deux réseaux ont été entraînés avec un pas d'apprentissage μ = 0.1 et une commande aléatoire u(k) uniformément répartie dans l'intervalle [-2 2]. Puisque N_g approxime la fonction $g[u(k)]$ seulement dans ce dernier intervalle, par conséquent N_f approxime la fonction $f[y_s(k)]$ dans l'intervalle [-10 10].

Après 10000 itérations, nous avons utilisé une entrée de test:

$$u(k) = \sin(2\pi k/25) + \sin(2\pi k/10) \qquad (3\text{-}11)$$

La réponse du système et celle du modèle sont représentées dans la fig.(3-5), l'erreur d'identification est très faible et elle est illustrée par la fig.(3-6).

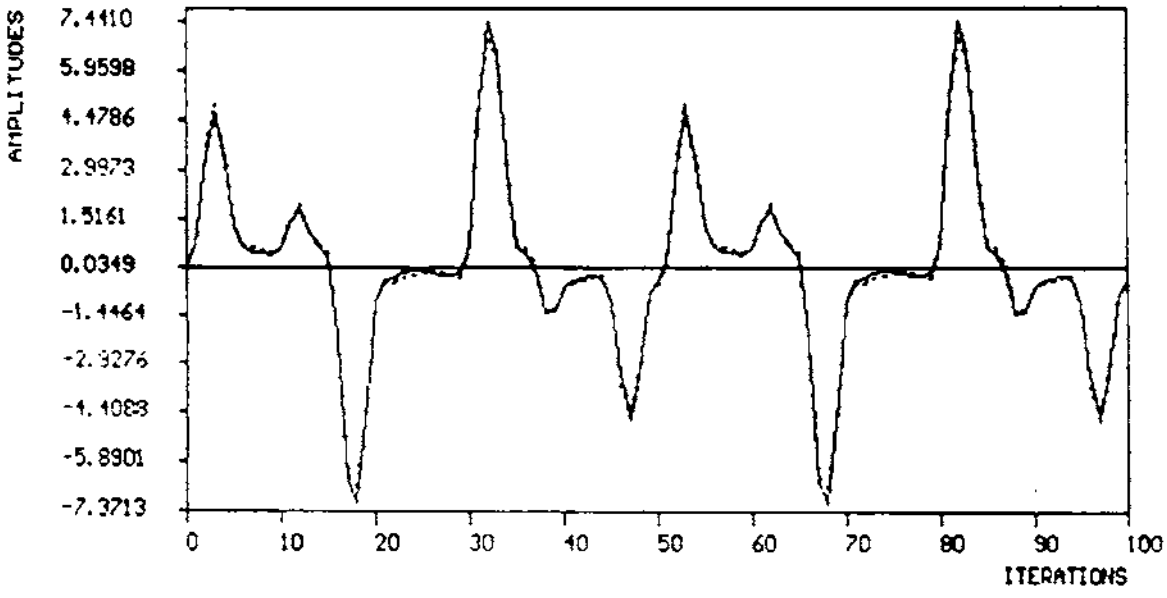


fig.(3-5): Sortie du système (en traits pleins) et celle du modèle (en pointillés) pour l'exemple 3-2.

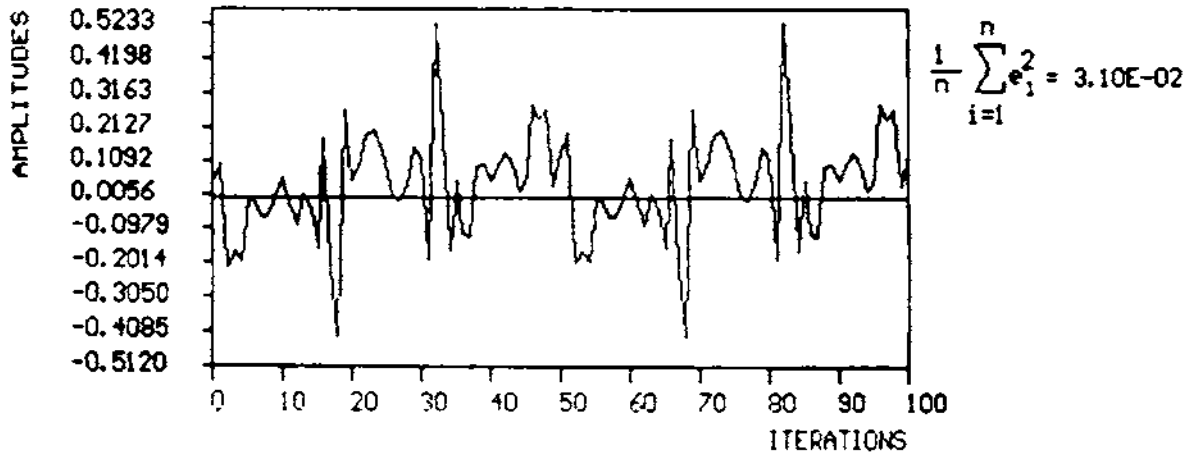


fig.(3-6): Erreur d'identification pour l'exemple 3-2.

Exemple 3-3:

Dans cet exemple, on considère l'identification d'un système du deuxième ordre, décrit par une équation aux différences de la forme:

La réponse du système et celle du modèle sont représentées dans la fig.(3-5), l'erreur d'identification est très faible et elle est illustrée par la fig.(3-6).

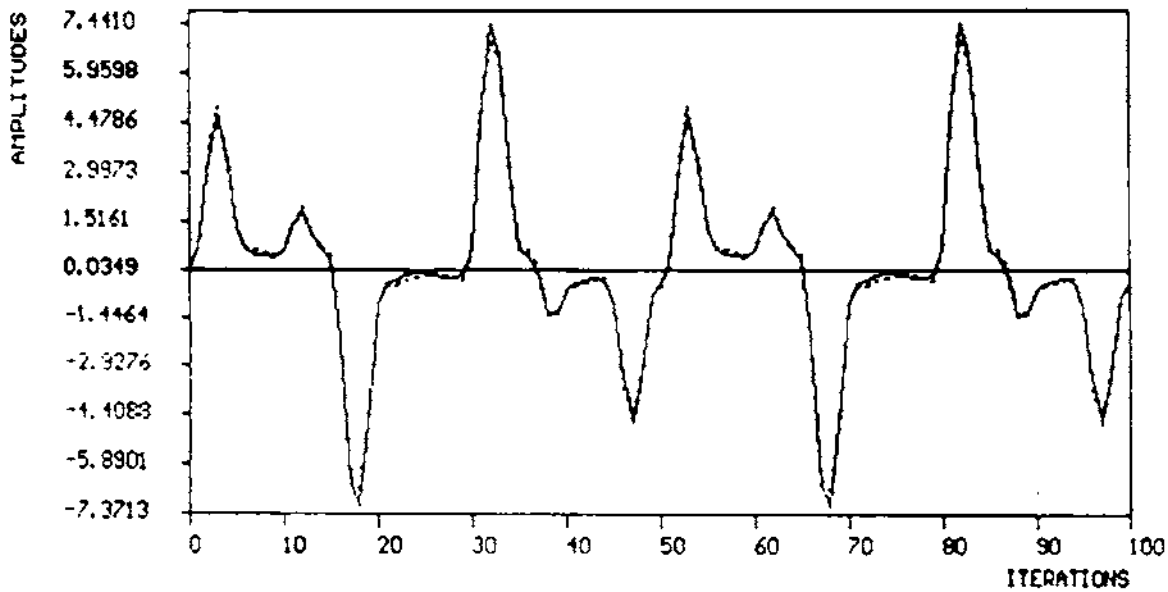


fig.(3-5): Sortie du système (en traits pleins) et celle du modèle (en pointillés) pour l'exemple 3-2.

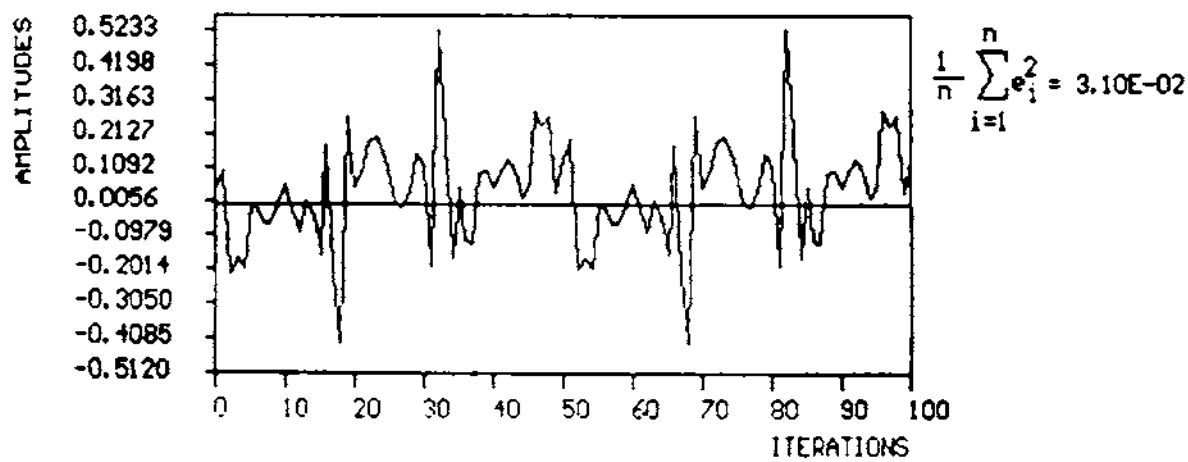


fig.(3-6): Erreur d'identification pour l'exemple 3-2.

Exemple 3-3:

Dans cet exemple, on considère l'identification d'un système du deuxième ordre, décrit par une équation aux différences de la forme:

$$y_s(k+1)=(0.8-0.5EXP(-y_s^2(k))y_s(k)-(0.3+0.9EXP(-y_s^2(k))y_s(k-1)+0.1\sin(\pi y_s(k))+u(k) \quad (3\text{-}12)$$

Le modèle d'identification série-parallèle utilisé est décrit, dans ce cas, par l'équation suivante:

$$y_m(k+1)=N[y_s(k),y_s(k-1)]+u(k)$$

où N est un réseau MLP, à deux entrées et une seule sortie, comportant 15 neurones sur la première couche cachée et 10 neurones sur la deuxième couche cachée.

Les conditions initiales du système sont considérées nulles, et le modèle est entraîné en utilisant une commande aléatoire uniformément répartie dans l'intervalle [-1 1] et un pas d'apprentissage variable.

Afin de tester les performances du modèle obtenu on a utilisé une entrée sinusoidale donnée par:

$$u(k)=1.5\sin(2\pi k/40) \quad (3\text{-}13)$$

l'allure de la réponse du système et celle du modèle sont illustrées par la fig.(3-7). La fig.(3-8) montre l'ordre de grandeur de l'erreur d'identification, bien qu'on remarque une légère différence entre les deux réponses, le modèle présent une réponse de même allure que celle du système.

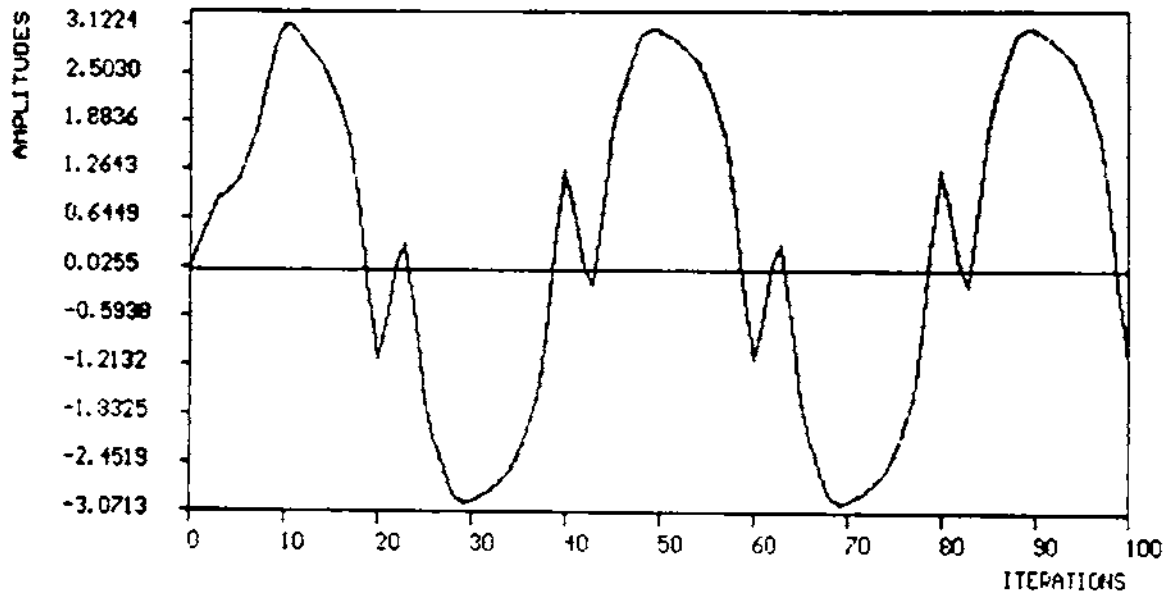


fig.(3-7): Sorties du système et du modèle superposées pour l'exemple 3-3.

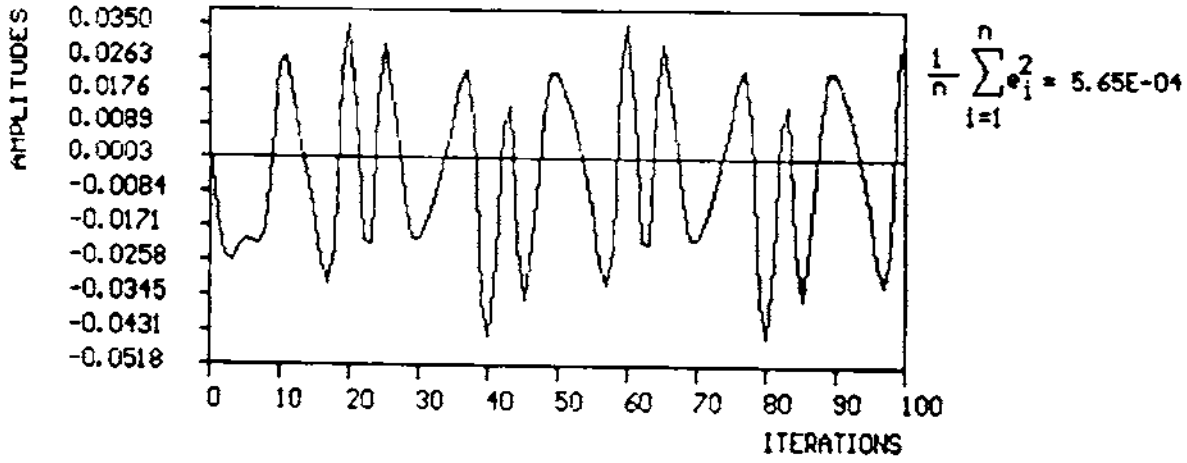


fig.(3-8): Erreur d'identification pour l'exemple 3-3.

Exemple 3-4:

Cette fois, on considère un cas plus général où le comportement dynamique du système à identifier est décrit par le modèle IV. Il s'agit de déterminer un modèle qui peut présenter le même comportement dynamique que celui du système donné par l'équation aux différences du troisième ordre suivante:

$$y_s(k+1) = \frac{0.6\,[\sin[y_s(k)\,y_s(k-1)\,y_s(k-2)\,(y_s(k-2)-1)\,u(k-1)]+u(k)]}{1+y_s^2(k-2)+y_s^2(k-1)} \qquad (3\text{-}14)$$

Le modèle d'identification est donné par:

$$y_m(k) = N[y_s(k), y_s(k-1), y_s(k-2); u(k), u(k-1)]$$

Dans ce cas, nous avons utilisé un réseau MLP, à cinq entrées et une sortie, comportant 25 neurones et 12 neurones sur la première et la deuxième couche cachée, respectivement. Ce réseau a été entraîné avec une commande aléatoire uniformément répartie dans l'intervalle [-1 1] et un pas d'apprentissage μ = 0.1.

La fig.(3-9) montre les formes de la sortie du système et celle du modèle, obtenu après 50000 itérations d'entraînement, lorsqu'une entrée teste de la forme:

$$u(k) = 0.75[\sin(2\pi k/25) + \sin(2\pi k/10)] \qquad (3\text{-}15)$$

est utilisée. On remarque que l'écart entre les deux réponses fig.(3-10) est faible.

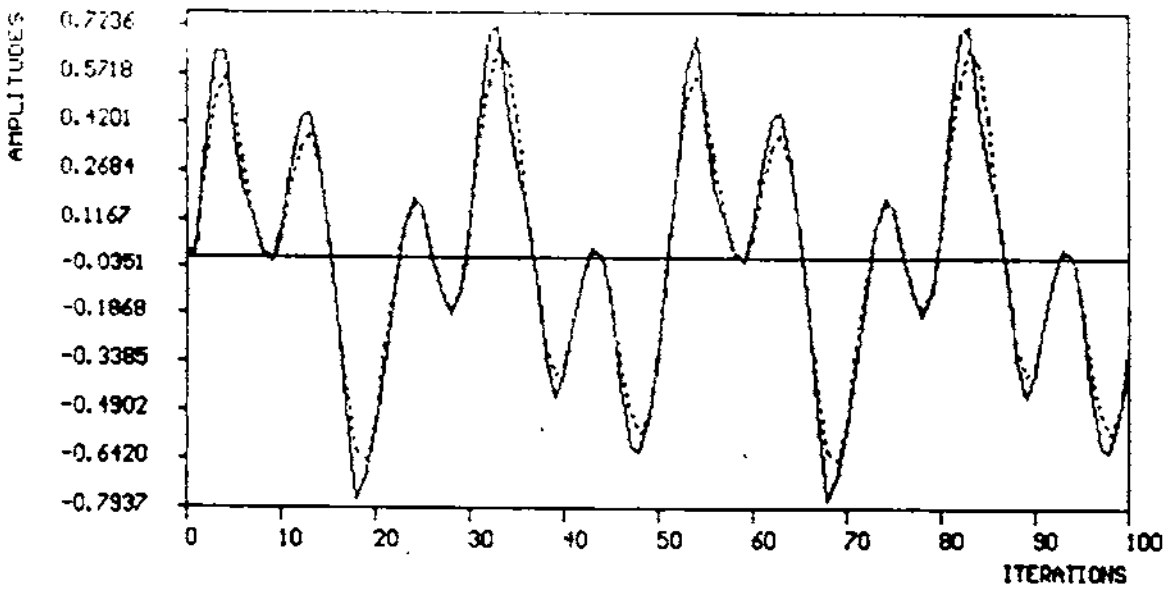


fig.(3-9): Sortie du système (en traits pleins) et celle du modèle (en pointillés) pour l'exemple 3-4.

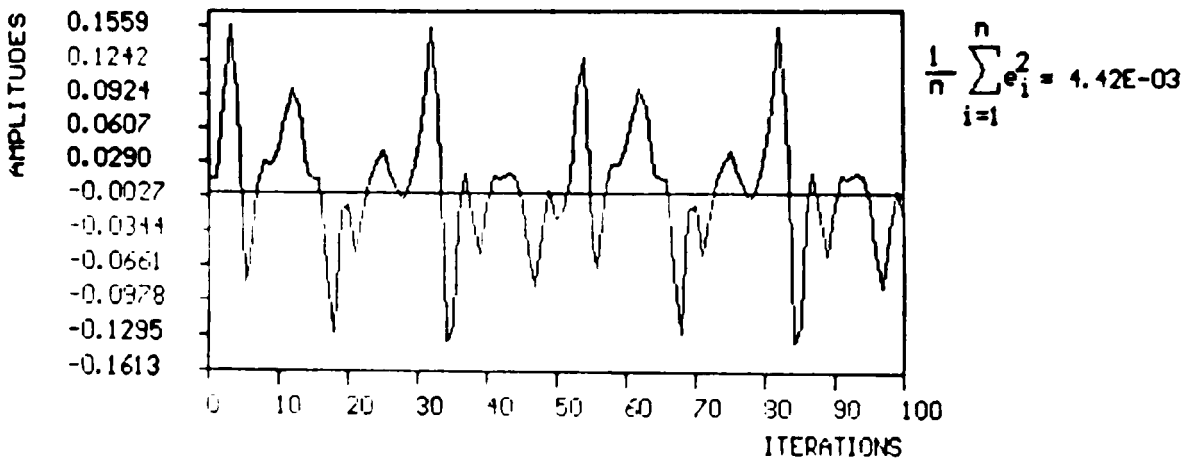


fig.(III-10): Erreur d'identification pour l'exemple 3-4.

3-4 IDENTIFICATION PAR UN RESEAU RBF :

Vu les similitudes existant entre un réseau MLP et un réseau RBF, on s'attend que ce dernier présente les mêmes capacités d'approximation et de généralisation que le premier. En effet ceci a été prouvé par plusieurs auteurs [33], et il a été démontré qu'un réseau RBF dont les centres et les poids des connexions sont correctement choisis est capable d'approximer avec une précision arbitraire n'importe quelle fonction continue.

En plus de leurs capacités d'approximation et de généralisation, un réseau RBF présente à sa sortie une réponse linéaire par rapport aux poids des connexions. Cette dernière propriété des réseaux RBF est d'une importance capitale, car c'est elle qui nous permet d'utiliser un algorithme d'optimisation linéaire (méthode des moindres carrés) pour ajuster les poids des connexions. L'utilisation de cette architecture peut donc fournir une solution aux problèmes soulevés par l'algorithme de la rétropropagation.

Récemment, l'utilisation des réseaux RBF dans les problèmes d'identification et de contrôle des systèmes non linéaires a été envisagée [33,35-37]. A partir de deux types d'apprentissage distincts, le modèle neuronal (dans ce cas un réseau RBF) est entraîné afin d'approcher le comportement dynamique du système sous identification.

3-4-1 ALGORITHME D'APPRENTISSAGE :

L'entraînement du modèle neuronal est effectué en utilisant un algorithme d'apprentissage non supervisé (K-means) pour ajuster les centres des fonctions de base, et un algorithme d'apprentissage supervisé (moindres carrés récursifs pondérés) pour adapter les poids des connexions du modèle neuronal. La version récursive de cet algorithme d'apprentissage hybride est donnée par les étapes suivantes:

1ère étape:
Initialiser les centres et les poids des connexions par des faibles valeurs aléatoires, choisir les valeurs initiales de $\alpha_c(0)$, λ_0 , $\lambda(0)$ et $F(0)$.

2ème étape:
Présenter le vecteur d'entrée donné par (3-6).

3ème étape:
Ajuster les centres des fonctions de base en utilisant les équations (2-20)-(2-22), ensuite recalculer la k ème distance $d_k(i)=\|X(i)-C_x(i)\|$ et adapter le taux d'apprentissage $\alpha_c(i)$ en utilisant l'équation (2-23).

4ème étape:
Spécifier la sortie désirée correspondant au vecteur d'entrée appliqué et calculer l'erreur de sortie en utilisant les équations: (2-12), (2-24) et (2- 33).

5ème étape:
Ajuster les poids des connexions en utilisant les équations (2-31) et (2-32), ensuite adapter le facteur d'oublie $\lambda(i)$ en utilisant la règle (2-34).

6ème étape:
Présenter un autre vecteur d'entrée et aller à l'étape 3.

Le processus d'apprentissage est arrêté lorsque l'erreur de sortie se stabilise à un niveau acceptable.

3-4-2 SIMULATION :

Le modèle neuronal utilisé, dans ce cas, est un réseau RBF statique dont la structure est illustrée par la fig.(2-2). La fonction de base choisie est celle donnée par l'équation (2-14), avec $\rho = 1$. Dans tous les exemples, l'algorithme d'apprentissage décrit dans le paragraphe précédent est utilisé pour entraîner le modèle neuronal, les paramètres de ce dernier sont initialisés comme suit:
$F(0) = 1000I$ où I est la matrice identité d'ordre n_h, $\lambda_0 = 0.99$, $\lambda(0) = 0.95$ et $\alpha_c(0) = 0.9$.

Exemple 3-5 :

Le système à identifier est décrit par l'équation (3-8). La structure du réseau RBF utilisé pour approximer la fonction f est définie par:
un neurone sur la couche d'entrée, 21 neurones sur la couche cachée et un neurone de sortie.

Le modèle neuronal est entraîner pendant 200 itérations, pour un état initial du système $y_s(0) = 0.25$ et une commande aléatoire uniformément repartie sur l'intervalle [-2 2].

Les allures de la réponse du système et celle du modèle neuronal, pour une entrée de commande exprimée par l'équation (3-9), sont données par la fig.(3-11). On constate que les deux courbes sont superposées, l'erreur d'identification est très faible. La fig.(3-12) montre les variations de cette erreur.

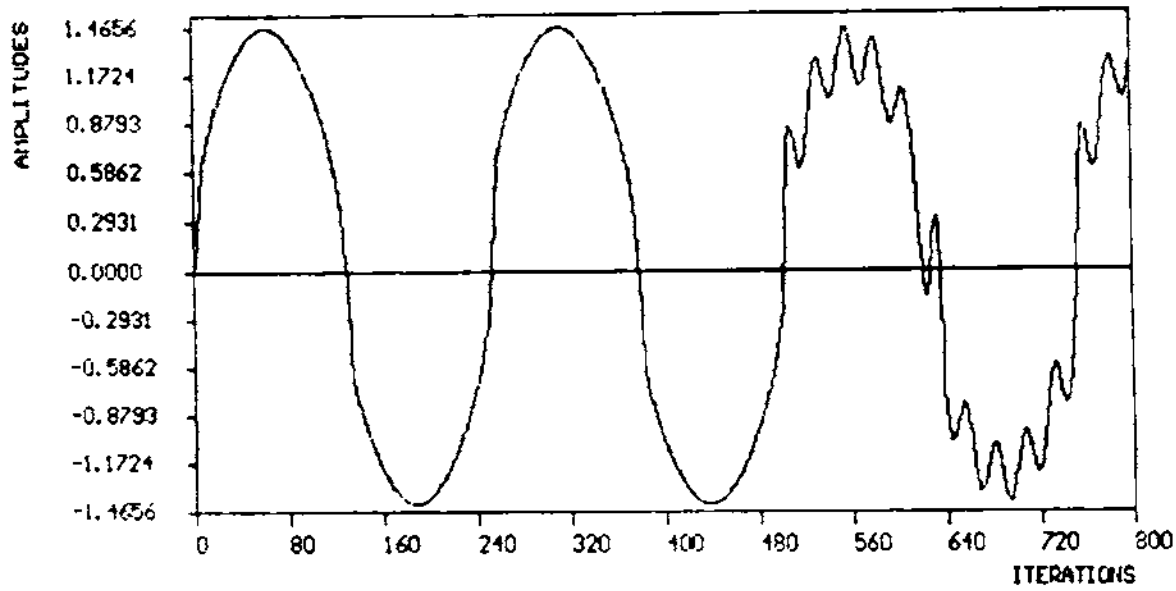


fig.(3-11): Sorties du système et du modèle superposées pour l'exemple 3-5.

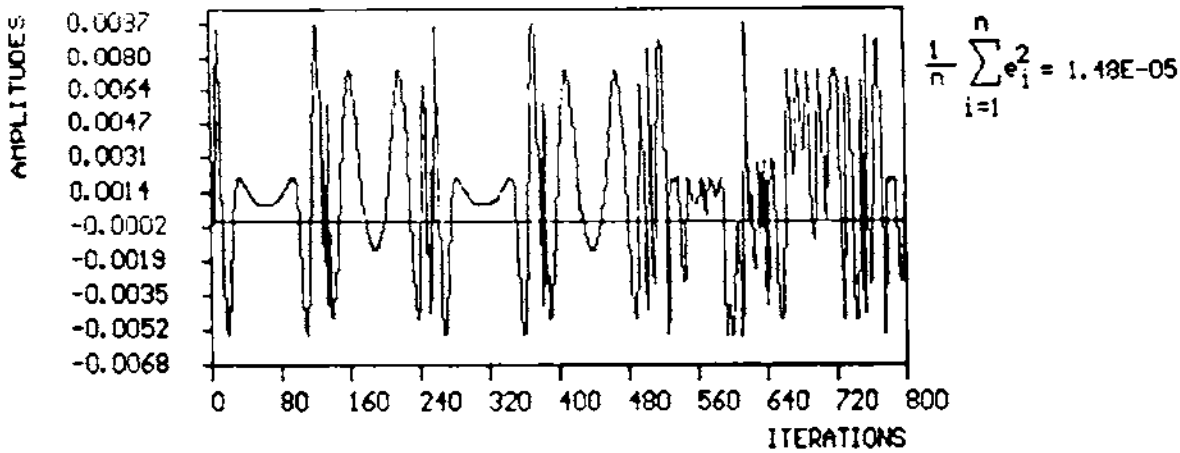


fig.(3-12): Erreur d'identification pour l'exemple 3-5.

Exemple 3-6 :

Dans cet exemple, l'objectif est d'entraîner un modèle neuronal pour apprendre la dynamique directe du système décrit par l'équation (3-10). La structure de ce modèle est donnée par l'équation suivante:

$$y_m(k+1) = R_f[y_s(k)] + R_g[u(k)]$$

Où $R_f[.]$ et $R_g[.]$
sont deux réseaux RBF à une seule entrée. Chacun de ces réseaux comporte 21 neurones sur la couche cachée et un neurone sur la couche de sortie.

Pour un état initial du système donné par $y_p(0) = 0.25$ et une commande aléatoire uniformément répartie sur l'intervalle [-2 2], un nombre total de 1000 paires de mesures entrée-sortie a été utilisé pour entraîner les deux réseaux R_f et R_g.

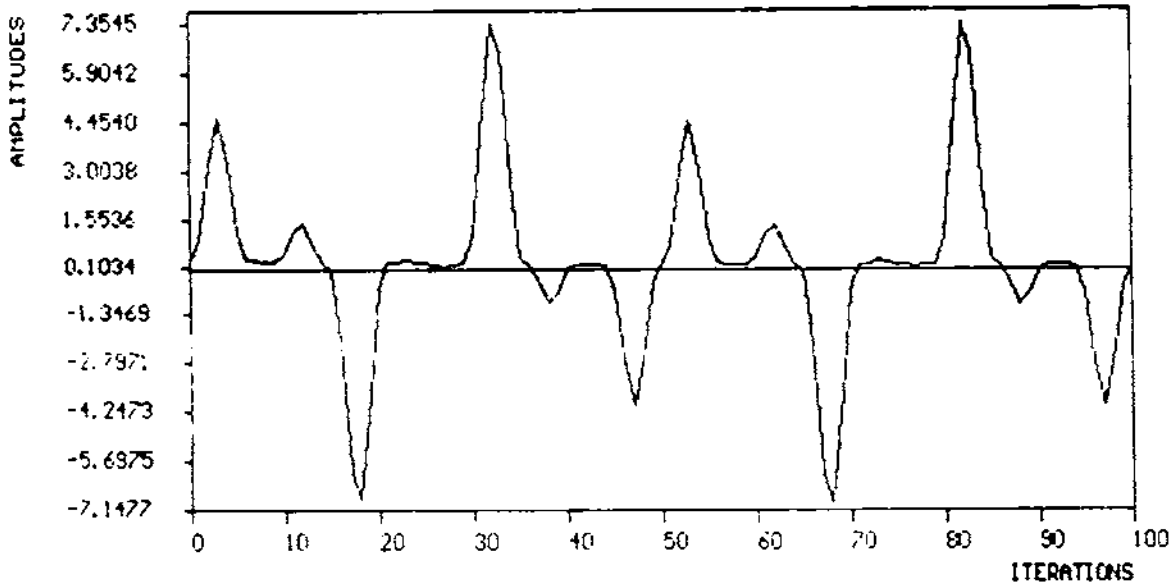


fig.(3-13): Sorties du système et du modèle superposées pour l'exemple 3-6.

Les formes de la réponse du système et celle du modèle pour une entrée de test donnée par l'équation (3-11) sont représentées dans la fig.(3-13). On remarque que l'écart entre le système et le modèle est négligeable, la plage de variation de cet écart est indiqué sur la fig.(3-14).

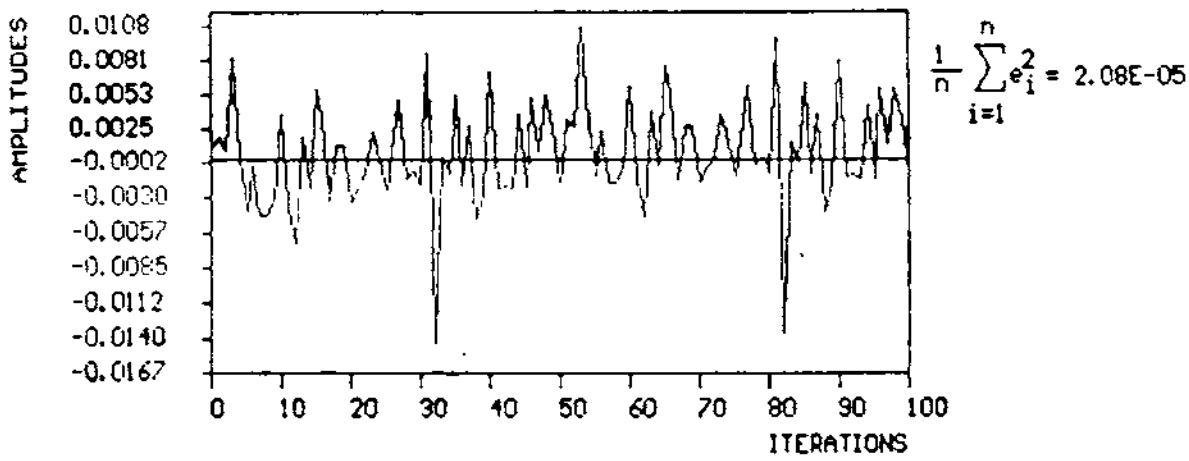


fig.(3-14): Erreur d'identification pour l'exemple 3-6.

Exemple 3-7 :

Il s'agit de l'identification de la dynamique directe d'un système du deuxième ordre décrit par l'expression (3-12).

Le modèle d'identification utilisé est donné par l'équation suivante:

$$y_m(k+1) = R[y_s(k), y_s(k-1)] + u(k)$$

où R est un réseau RBF défini par:
2 neurones sur la couche d'entrée, 30 neurones sur la couche cachée et un neurone sur la couche de sortie. Ce réseau est entrainé pendant 1500 itérations avec une entrée aléatoire uniformément répartie dans l'intervalle [-1 1].

La réponse du modèle obtenu et celle du système, à une entrée de test donnée par l'équation (3-13), sont représentées dans la fig.(3-15). L'erreur d'identification représentée dans la fig.(3-16) est très faible et corréspond à une validation satisfaisante de ce modèle.

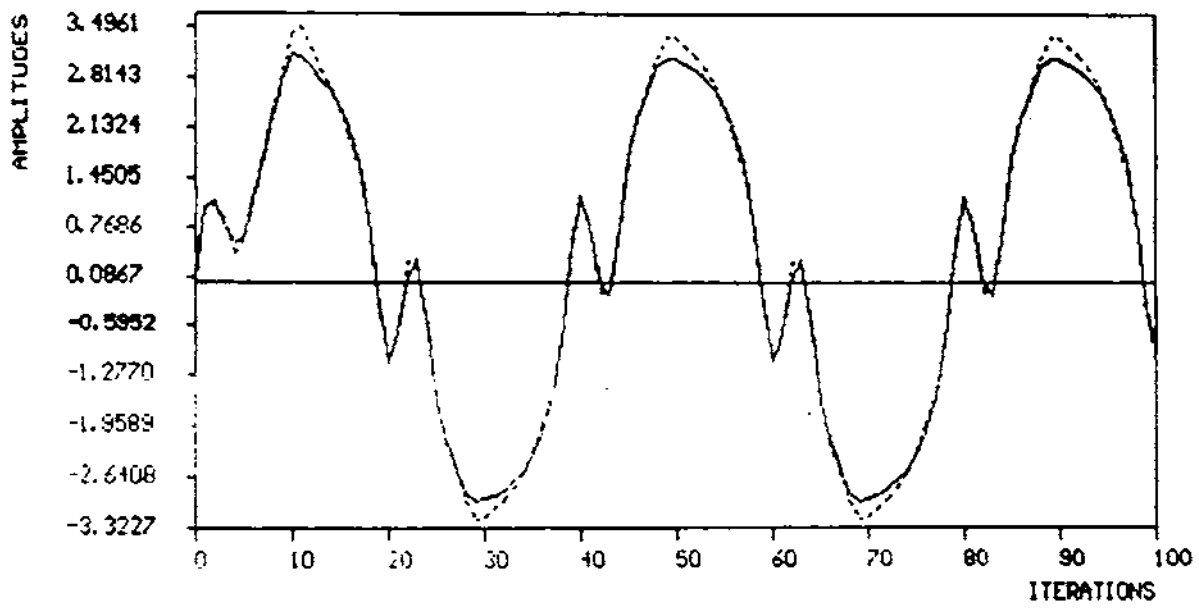


fig.(3-15): sortie du système (en traits pleins) et celle du modèle (en pointillés) pour l'exemple 3-7.

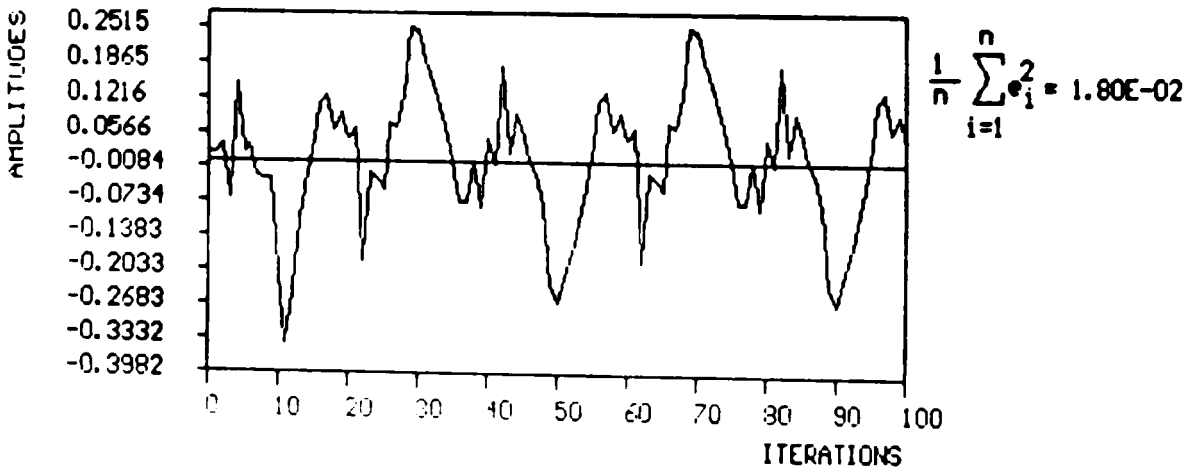


fig.(3-16): **Erreur d'identification pour l'exemple 3-7.**

Exemple 3-8 :

Dans ce cas, on considère l'identification d'un système du troisième ordre dont la structure est complètement inconnue. Le comportement dynamique de ce système est simulé par l'équation aux différence (3-14).

En vue d'identifier la dynamique directe de ce système nous avons procédé à l'entraînement d'un réseau RBF comportant 40 neurones sur sa couche cachée et dont le vecteur d'entrée est donné par: $X(k) = [y_s(k), y_s(k-1), y_s(k-2); u(k), u(k-1)]^T$.

L'entrée d'entraînement utilisée est aléatoire et uniformément répartie dans l'intervalle [-1 1].

Après 5000 itérations d'entraînement, l'entrée de test donnée par l'équation (3-15) est utilisée pour tester la validité du modèle obtenu. Les réponses du système et modèle à cette entrée sont représentées dans la fig.(3-17). On constate qu'il y a une légère différence entre les deux réponses, cette différence est représentée dans la fig.(3-18).

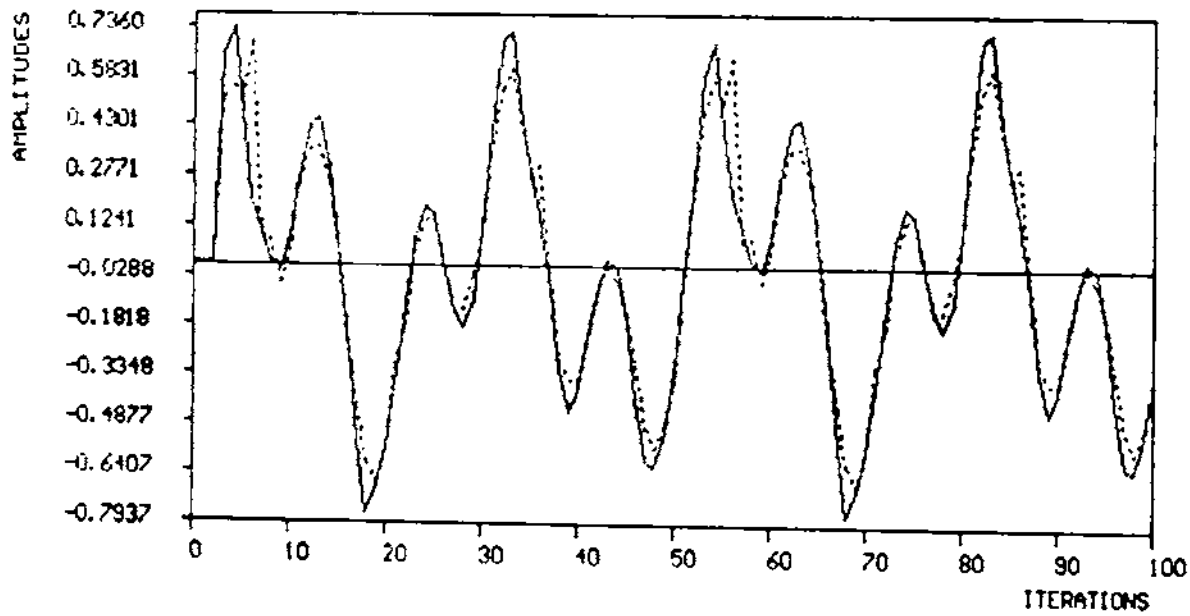


fig.(3-17): Sortie du système (en traits pleins) et celle du modèle (en pointillés) pour l'exemple 3-8.

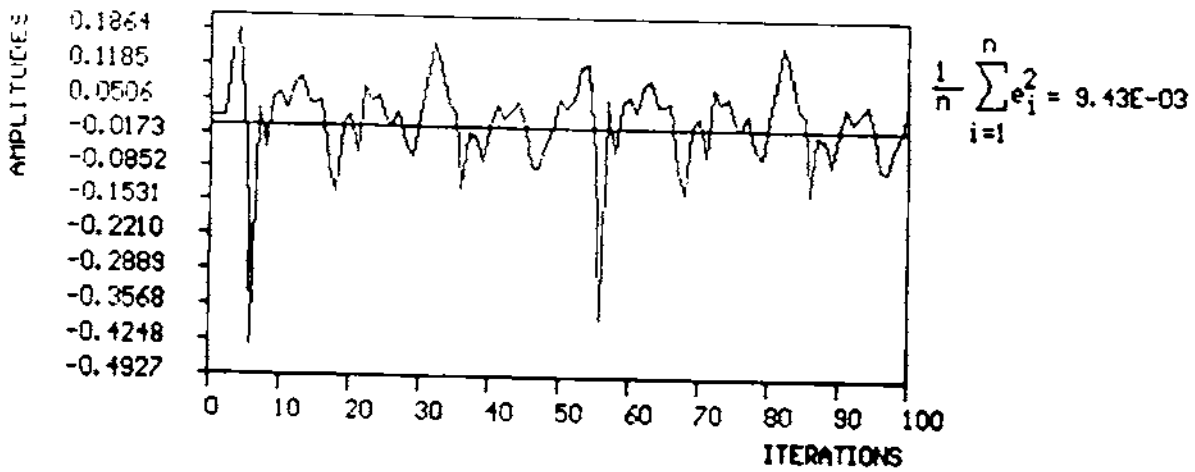


fig (3-18): Erreur d'identification pour l'exemple 3-8.

3-5 REJECTION DES PERTURBATIONS:

L'objectif est de construire un modèle pour le système à étudier de telle façon à minimiser l'effet des perturbations, supposées additives en entrée, sur la sortie. Les différents types de perturbations sont considérés comme étant la réponse libre d'un systeme dynamique linéaire ou non.

3-5-1 POSITION DU PROBLEME:

Un système mono-variable en absence des perturbations externes est représenté par les équations d'état suivantes:

$$x_s(k+1) = f_1[x_s(k), u(k)]$$
$$y(k) = h_1[x_s(k)] \qquad (3\text{-}16)$$

Où $x_s(k) \in R^n$ est le vecteur d'état du système, $u(k) \in R$ son entrée et $y(k) \in R$ sa sortie.

En supposant que le système est observable, une représentation entrée-sortie (auto-regressive) de ce dernier est donnée par l'équation aux différences suivante:

$$y(k+1) = F_1[Y(k), U(k)] \qquad (3\text{-}17)$$

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

Une perturbation externe peut être décrite par les équations d'état suivantes:

$$x_p(k+1) = f_2[x_p(k)]$$
$$v(k) = h_2[x_p(k)] \qquad (3\text{-}18)$$

Où $x_p(k) \in R^p$ est le vecteur d'état du système générateur de perturbations et $v(k)$ sa réponse libre.

On suppose que le système générateur de perturbations admet aussi une représentation auto-regressive telle que:

$$v(k+1) = F_2[V(k)] \qquad (3\text{-}19)$$

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

Si $v(k)$ est aussi considérée comme étant une entrée externe au système décrit par l'équation (3-16), le système global (système en présence des perturbations) sera représenté par:

$$x_s(k+1) = f_3[x_s(k), u(k), v(k)]$$
$$x_p(k+1) = f_2[x_p(k)]$$
$$v(k) = h_2[x_p(k)] \qquad (3\text{-}20)$$
$$y(k) = h_1[x_s(k)]$$

ou d'une manière équivalente, sous forme auto-regressive par:

$$x_g(k+1) = f_4[x_g(k), u(k)]$$
$$y(k) = h_1[x_s(k) = h_4[x_g(k)]] \qquad (3\text{-}21)$$

Où $x_g(k+1) = [x_n^T, x_p^T] \in R^{n+p}$ est le vecteur d'état du système global.

On remarque que, malgré que le système global possède la même entrée et la même sortie que le système en absence des perturbations, la dimension de l'espace d'état a été augmentée afin de tenir en compte le système générateur de perturbations. A chaque instant, les seules grandeurs mesurables sont l'entrée et la sortie du système global.

Le modèle non linéaire, tel qu'il est donné par les équations (3-20) et (3-21), présente des difficultés mathématiques dans son analyse théorique, par conséquent plusieurs modèles de représentation particuliers sont utilisés pour développer des méthodes permettant de résoudre le problème de rejection des perturbations, ensuite la possibilité de généralisation est envisagée.

3-5-2 CAS PARTICULIER:

Les différents modèles particuliers (modèles I - III) donnés dans le paragraphe 3-2-1 peuvent être utilisés pour étudier le problème de rejection des perturbations, cependant on se limite au modèle particulier donné par:

$$\begin{aligned} y(k+1) &= f[y(k),\ldots,y(k-n+1)] + \sum_{j=0}^{m-1} \beta_j [u(k-j) + v(k-j)] \\ v(k+1) &= \sum_{i=0}^{p-1} \gamma_i v(k-i) \end{aligned} \qquad (3\text{-}22)$$

Où la fonction $f(.)$ et les coefficients β_j sont inconnus et doivent être estimés.

soient $\begin{aligned} D(z) &= \beta_0 + \beta_1 z^{-1} + \ldots + \beta_{m-1} z^{-(m-1)} \\ R(z) &= \gamma_0 + \gamma_1 z^{-1} + \ldots + \gamma_{p-1} z^{-(p-1)} \end{aligned}$, l'expression (3-22)

devient alors:

$$\begin{aligned} y(k+1) &= f[Y(k)] + D(z)\,[u(k) + v(k)] \\ v(k+1) &= R(z)\,v(k) \end{aligned} \qquad (3\text{-}23)$$

Dans ce cas, le système global admet une représentation entrée-sortie. En effet en éliminant $v(k)$ de (3-23), on aura:

$$D(z)\,v(k-i) = y(k-i+1) - f[Y(k-i)] - D(z)\,u(k-i) \qquad (3\text{-}24)$$

l'expression (3-23) peut être écrite de la façon suivante:

$$y(k+1) = f[Y(k)] + D(z)\,u(k) + \sum_{i=1}^{p} \gamma_{i-1} D(z)\,v(k-i) \qquad (3\text{-}25)$$

donc, la représentation entrée-sortie du système global est donnée par:

$$y(k+1) = \bar{f}[\bar{Y}(k)] + \bar{D}(z)\,u(k) \qquad (3\text{-}26)$$

Où:

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

et

$$\bar{D}(z) = D(z)\,[1 - z^{-1}R(z)] = \sum_{i=0}^{n+p-1} \beta_j z^{-i}$$

Un réseau MLP ou RBF peut être utilisé pour approximer la non linéarité $\bar{f}$. Les coefficients β_j sont estimés en utilisant une loi d'adaptation linéaire [38].

On remarque que, la représentation obtenue (Eq.3-26) reste linéaire par rapport à l'entrée u(k). Cependant le nombre des valeurs passées de l'entrée et de la sortie nécessaires pour cette représentation a augmenté par p valeurs en comparaison avec la représentation entrée-sortie en absence des perturbations.

3-5-3 CAS GENERAL :

Dans ce cas, on considère le problème général pour lequel la commande apparaît sous forme non linéaire dans la dynamique du système, et la perturbation est considérée comme étant la réponse libre d'un système non linéaire. Le problème d'existence, dans ce cas, d'une représentation entrée-sortie pour le système en présence des perturbations est difficile à résoudre. Si le système en présence d'une perturbation externe est représenté par:

$$\begin{aligned} y(k+1) &= f[Y(k), U(k)+V(k)] \\ v(k+1) &= g[V(k)] \end{aligned} \qquad (3\text{-}27)$$

Où:

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

et $V(k)=[v(k),...,v(k-p+1)]^T$. Le problème est de déterminer si le système global admet une représentation entrée-sortie de la forme:

$$y(k+1)=\bar{f}[\bar{Y}(k),\bar{U}(k)] \qquad (3\text{-}28)$$

où:

$\bar{Y}(k)=[y(k),..,y(k-(n+s)+1)]^T$, $\bar{U}(k)=[u(k),...,u(k-(m+s)+1)]^T$ et s est un entier positif fini.

Si la rejection des perturbations est possible pour le système obtenu par linéarisation de (3-27) et (3-28) autour d'un point d'équilibre de l'espace d'état augmenté, on admet que la rejection est aussi possible pour le problème non linéaire au voisinage de ce point d'équilibre. plusieurs travaux sont aujourd'hui en progrès pour justifier ceci [38].

EXEMPLE 3-9:

Dans cet exemple, on considère l'identification d'un système du troisième ordre en présence d'une perturbation externe. Le comportement dynamique de ce système est décrit par l'équation aux différences suivante:

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

La perturbation v(k) est considérée comme étant la réponse libre d'un système non linéaire du deuxième ordre, elle est donnée par l'expression suivante:

$$v(k+1)=\gamma_1 v(k)+\gamma_2 v(k-1)+\gamma_3(v^2(k-1)-1)(v(k)-v(k-1))$$

avec $v(1) = 0.1736$, $v(0) = 0$, $\gamma_1 = 0.02$, $\gamma_2 = 0.0154$, $\gamma_3 = 1.0154$.

Le modèle neuronal utilisé, pour identifier la dynamique directe du système considéré, est un réseau MLP comportant: 40 neurones sur la première couche cachée et 25 neurones sur la deuxième. Le vecteur d'entrée de ce réseau est donnée par:

$$x(k)=[y(k),y(k-1),\ldots,y(k-s-2);u(k),u(k-1),\ldots,u(k-s-1)]^T$$

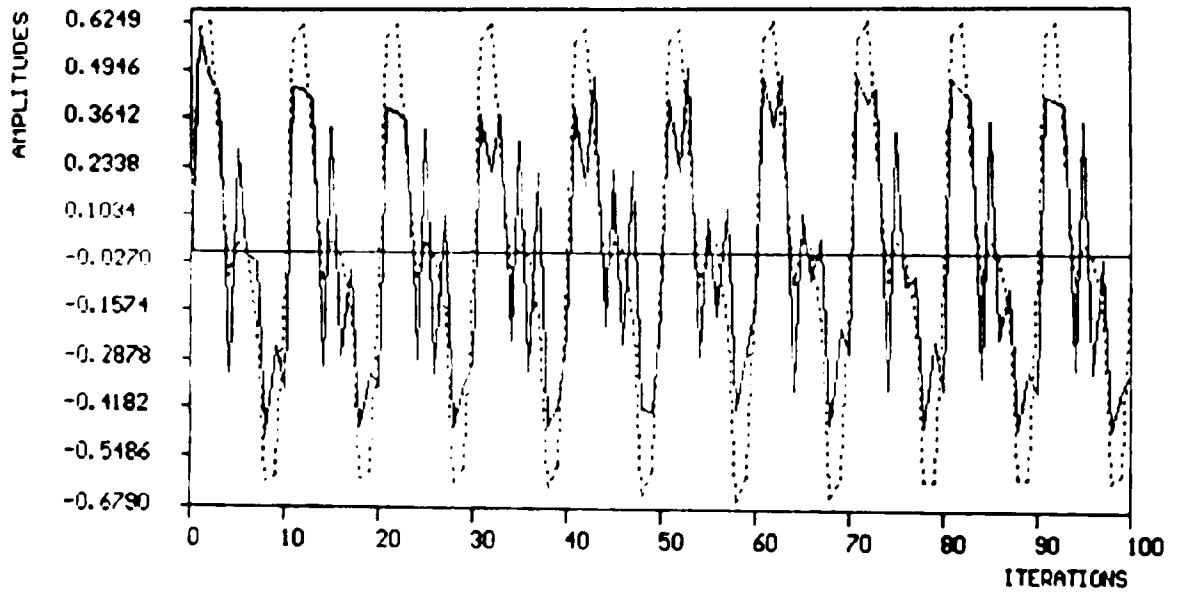
Dans une première étape, nous avons procédé à l'identification de ce système sans augmenter la dimension de l'espace d'état (s = 0), ensuite cette dimension a été augmentée par s = 2 et enfin par s = 4.

Un nombre total de 80000 de mesures d'entrées-sortie est utilisé pour entraîner ce modèle, avec une commande aléatoire uniformément répartie dans l'intervalle [-2 2] et un pas d'apprentissage $\mu = 0.05$. Ensuite une entrée de test de la forme:

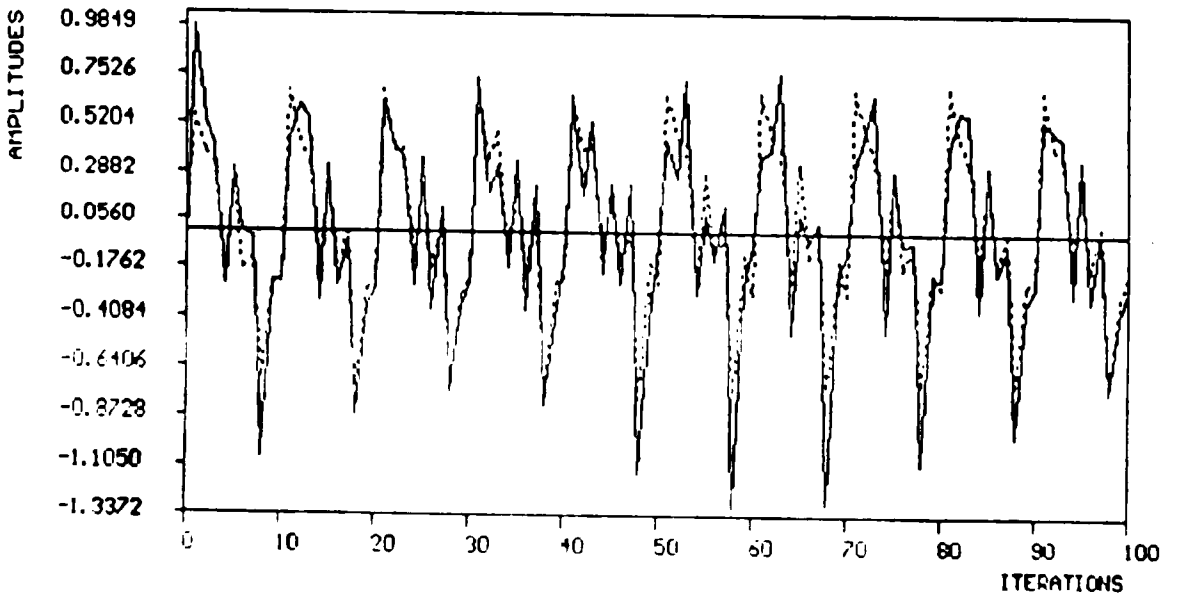
$$u(k)=\cos(2\pi/10)+0.75\sin(2\pi/5)$$

est utilisée pour comparer la sortie du modèle avec celle du système.

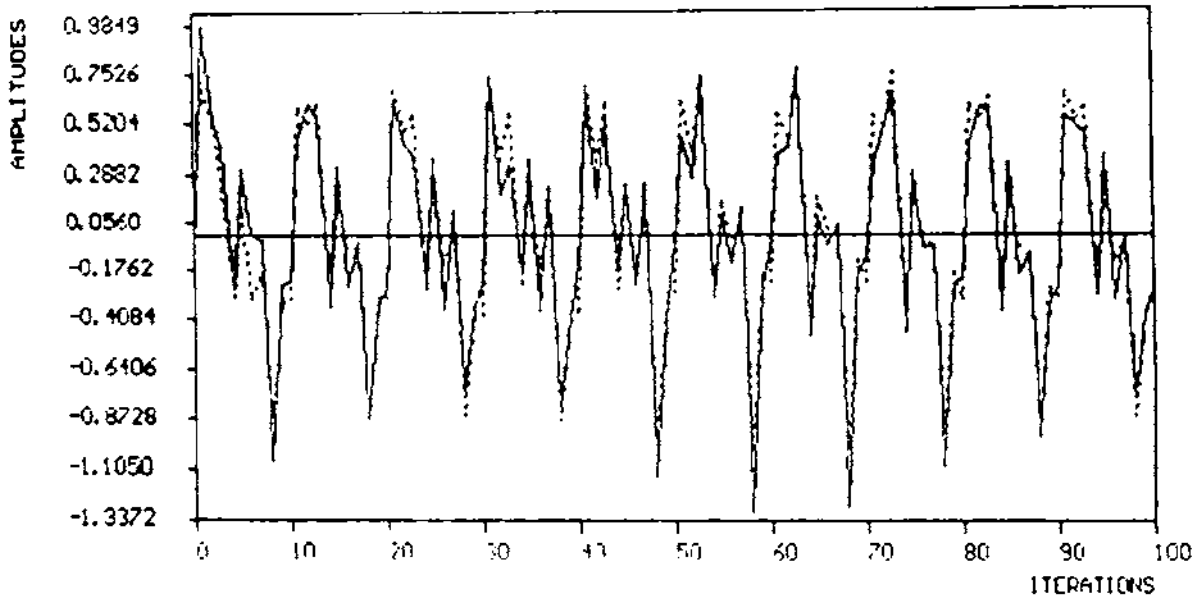
On constate que l'effet de la perturbation sur la sortie du modèle est important pour s = 0 (fig.(3-19 a)), il est moins important pour s = 2 (fig.(3-19 b)) et il est faible pour s = 4 (fig.(3-19 c)).



(a) avec s=0

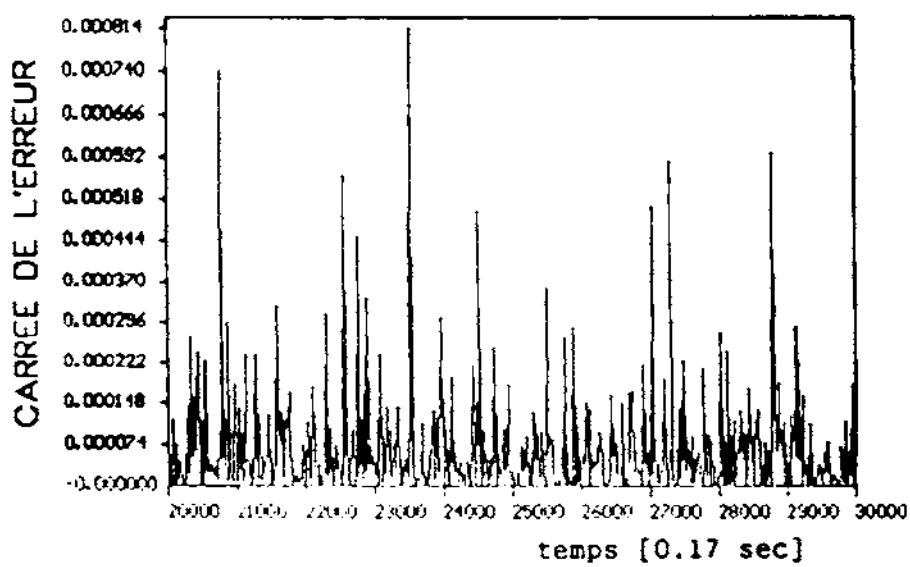


(b) avec s=2

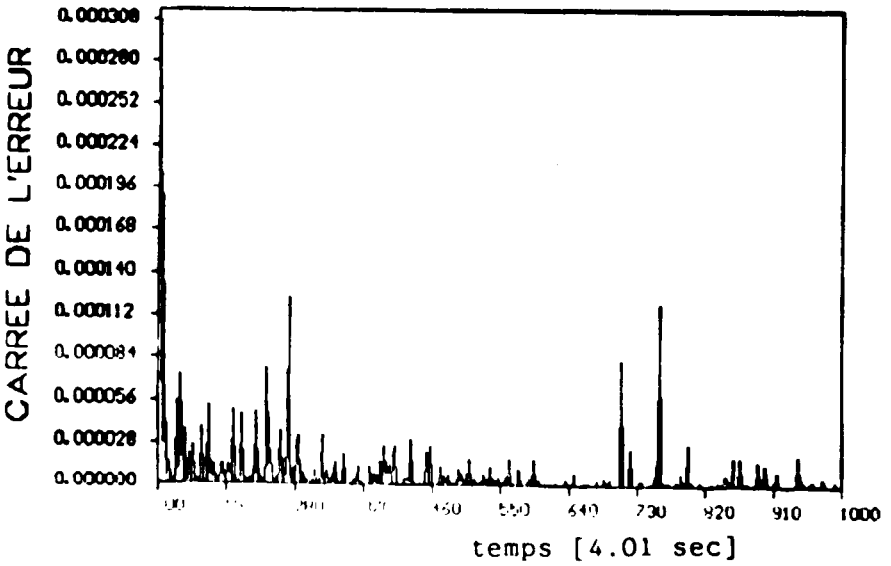


(c) avec s=4.

fig.(3-19): Sortie du système (en traits pleins) et celle du modèle (en traits pointillés) pour l'exemple (3-9).

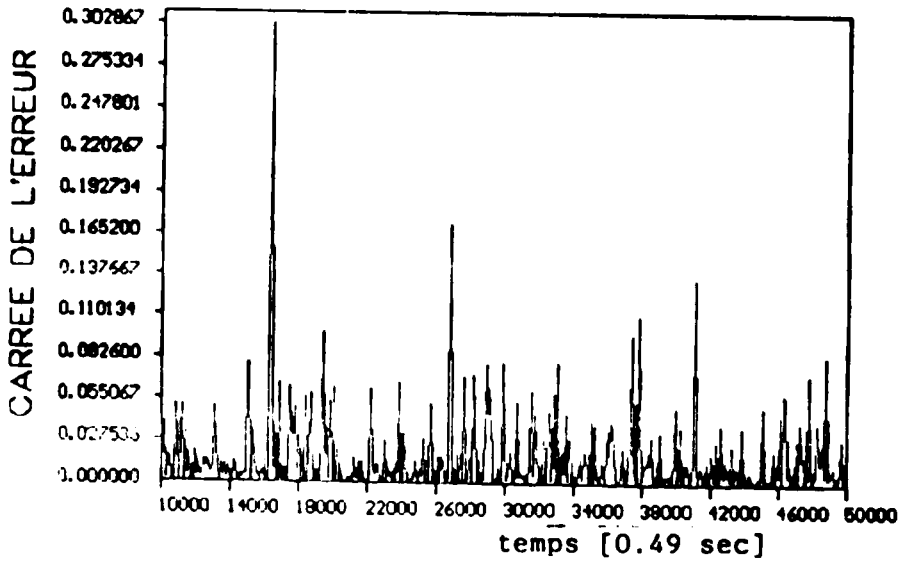


(a) avec un réseau MLP.

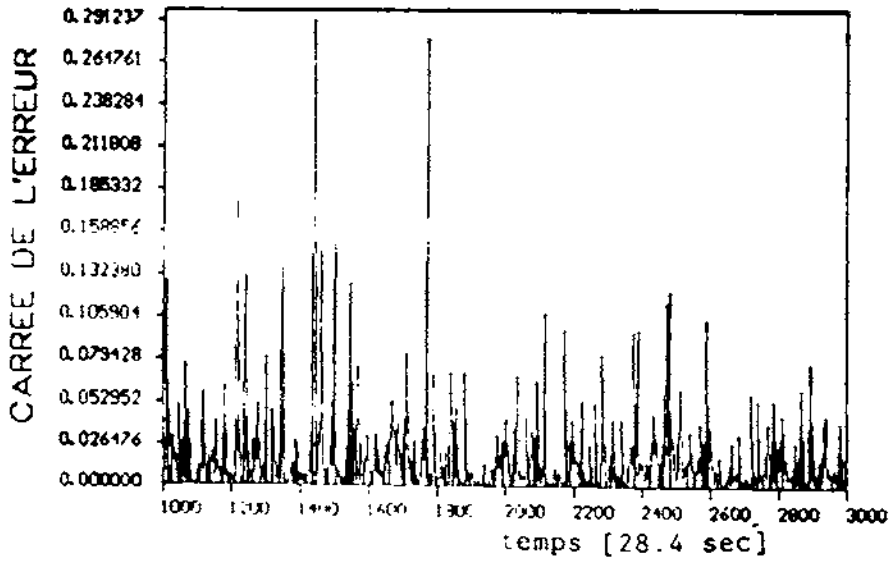


(b) avec un réseau RBF.

Fig.3-20: Erreur d'identification pendant la phase d'apprentissage pour le système donné par l'Eq.(3-8).



(a) avec un réseau MLP.



(b) avec un **réseau RBF.**

Fig.3-21: Erreur d'identification pendant la phase d'apprentissage pour le système donné par l'Eq.(3-14).

3-6 CONCLUSION :

La capacité des réseaux de neurones dans les problèmes d'approximation des fonctions non linéaires et leurs possibilités d'apprentissage et de généralisation, nous ont conduit à étudier le problème d'identification des systèmes non linéaires, pour les quels les techniques classiques se heurtent encore aujourd'hui à des difficultés d'ordre pratique et théorique, en utilisant les techniques neuronales. En effet nous venons de voir dans ce chapitre, que les réseaux de neurones peuvent être utilisés comme des blocs constructeurs dans l'identification des systèmes non linéaires inconnues avec un minimum d'information a priori, tel que l'ordre du système.

Deux architectures des réseaux de neurones ont été utilisées pour identifier la dynamique directe d'un système non linéaire. L'efficacité de ces architectures a été démontré par des exemples de simulation et des résultats satisfaisants ont été obtenus.

Les résultats de simulation montrent que les réseaux MLP

sont des approximateurs universels pour les systèmes non linéaires inconnus. Cependant, plusieurs difficultés peuvent se présenter pendant la phase d'apprentissage, ceci revient au fait que les réseaux MLP sont non linéaires par rapport aux paramètres. Par conséquent l'utilisation d'un algorithme d'optimisation non lineaire (la rétropropagation) est nécessaire pour l'entraînement.

Les réseaux RBF présentent une réponse linéaire par rapport aux paramètres. Ceci permet de surmonter les difficultés rencontrées dans les réseaux MLP, en utilisant des techniques d'optimisation linéaires pour leur entraînement. Les résultats de simulation montrent que pour des systèmes d'ordre relativement faible, l'entraînement des réseaux RBF est beaucoup plus rapide que celui des réseaux MLP fig.(3-20).

Cependant, pour des systèmes d'ordre relativement élevé, le nombre de fonctions de base (RBF) nécessaires croit d'une façon exponentielle. Par conséquent, l'apprentissage devient très lent (fig.(3-21)). Ceci est dû principalement à l'augmentation de la dimension de la matrice du gain d'adaptation dans l'algorithmes des moindres carrées récursifs.

Le problème de rejection des perturbations d'entrée a été aussi étudié et des justifications théoriques ont été données pour effectuer les modifications nécessaires dans le modèle d'identification.

CHAPITRE IV

Identification avec les réseaux récurrents

IDENTIFICATION AVEC LES RESEAUX RECURRENTS

4-1 INTRODUCTION :

Du fait que la plupart des systèmes physiques sont non linéaires et dynamiques, l'utilisation des réseaux récurrents pour l'identification et le contrôle de ces systèmes s'avère intéressante, et peut offrir plus d'avantages que l'utilisation des réseaux statiques. Cependant l'obtention des règles d'apprentissage pour les réseaux récurrents est plus complexe à cause de leur caractère dynamique.

En général, les algorithmes d'apprentissage dans les réseaux récurrents sont basés sur le calcul du gradient d'un critère de performance, par rapport aux poids du réseau [40]. Parmi ces algorithmes, on distingue: l'algorithme de la rétropropagation à travers le temps [17,39], l'algorithme de la rétropropagation dynamique [15] et l'apprentissage récursif en temps réel [8]. Ces algorithmes nécessitent un temps de calcul et un espace mémoire importants.

Récemment, le problème d'apprentissage dans les réseaux récurrents a connu de nombreuses recherches, et plusieurs algorithmes ont été proposés [41-45]. Dans ce chapitre, on présente l'algorithme de la rétropropagation à travers le temps et on considère l'identification des systèmes non linéaires avec une structure neuronale récurrente. Ensuite, on introduit une solution pour le problème de rejection des perturbations additives en sortie.

4-2 LA RETROPROPAGATION A TRAVERS LE TEMPS :

La rétropropagation à travers le temps est une généralisation de la rétropropagation ordinaire dans les réseaux statiques. Cette méthode a été proposée par Werbos [17], et utilisée par plusieurs chercheurs dans de nombreuses applications, telles que le contrôle, et le traitement de la

parole. L'idée de cet algorithme repose sur la notion des dérivées partielles ordonnées.

4-2-1 THEOREME DES DERIVEES PARTIELLES ORDONNEES [17,46] :

Pour définir la dérivée partielle ordonnée, on doit d'abord introduire la notion d'un ensemble d'équations ordonné.

soit $\{ Z_1, Z_2, \ldots, Z_i, \ldots, Z_n \}$ un ensemble de n variables dont les valeurs sont données par un ensemble de n équations. Ce dernier est dit ensemble ordonné si chaque variable Z_i est une fonction des variables $\{ Z_1, Z_2, \ldots, Z_{i-1} \}$ uniquement. Ou d'une manière équivalente:

$$Z_i = f_i(Z_1, Z_2, \ldots, Z_{i-1}) \qquad (4\text{-}1)$$

La dérivée ordonnée est une dérivée partielle dont les termes constants et variants sont déterminés par un ensemble d'équations ordonné. Mathématiquement, cette dérivée est définie de la façon suivante:
pour j=i+1:

$$\frac{\partial^* Z_j}{\partial Z_i} = \frac{\partial Z_j}{\partial Z_i} \qquad (4\text{-}2)$$

Si J > i+1, cette dérivée est donnée par:

$$\frac{\partial^* Z_j}{\partial Z_i} = \frac{\partial Z_j}{\partial Z_i} + \sum_{k=i+1}^{j-1} \frac{\partial^* Z_j}{\partial Z_k} \frac{\partial Z_k}{\partial Z_i} \qquad (4\text{-}3)$$

ou

$$\frac{\partial^* Z_j}{\partial Z_i} = \frac{\partial Z_j}{\partial Z_i} + \sum_{k=i+1}^{j-1} \frac{\partial Z_j}{\partial Z_k} \frac{\partial^* Z_k}{\partial Z_i} \qquad (4\text{-}4)$$

4-2-2 ALGORITHME :

Soit un réseau récurrent à m entrées, n sorties, N-m neurones cachés, et décrit par les équations suivantes:

$$x_i = u_i \qquad 1 \le i \le m \qquad (4\text{-}5)$$

$$net_i(t) = \sum_{j=1}^{i-1} W_{ij}^0 x_j(t) + \sum_{p=1}^{r} \sum_{\substack{j=1 \\ m<i\le N+n}}^{N+n} W_{ij}^p x_j(t-p) \qquad (4\text{-}6)$$

$$x_i(t) = S(net_i(t)) \qquad m < i \le N+n \qquad (4\text{-}7)$$

$$y_i(t) = x_{i+N}(t) \qquad 1 \le i \le n \qquad (4\text{-}8)$$

(les composantes du vecteur de sortie).

avec N ≥ m, S(.) est la fonction d'activation choisie et r est l'ordre de récurrence.

Le critère à minimiser est défini par:

$$E = \sum_{t=1}^{T} E(t) = \sum_{t=1}^{T} \sum_{i=1}^{n} \frac{1}{2} (y_i(t) - y_i^d(t))^2 \qquad (4\text{-}9)$$

Où T est l'horizon d'apprentissage.

En appliquant le théorème des dérivées partielles ordonnées, les dérivées partielles du critère (4-9) par rapport à chaque poids sont données par les expressions suivantes:

$$\frac{\partial^+ E}{\partial W_{ij}^p} = \sum_{t=1}^{T} \frac{\partial^+ E(t)}{\partial x_i(t)} \frac{\partial x_i(t)}{\partial W_{ij}^p} \qquad 0 \le p \le r \qquad (4\text{-}10)$$

A partir de (4-6) et (4-7) on obtient:

$$\frac{\partial x_i(t)}{\partial W_{ij}^p} = S'(net_i(t))\, x_j(t-p) \qquad (4\text{-}11)$$

l'expression (4-10) devient donc:

$$\frac{\partial^+ E}{\partial W_{ij}^p} = \sum_{t=1}^{T} \frac{\partial^+ E(t)}{\partial x_i(t)} S'(net_i(t))\, x_j(t-p) \qquad (4\text{-}12)$$

Le théorème des dérivées partielles ordonnées permet d'écrire:

$$\frac{\partial^{+}E(t)}{\partial x_i(t)} = \frac{\partial E(t)}{\partial x_i(t)} + \sum_{j=i+1}^{N+n} \frac{\partial^{+}E(t)}{\partial x_j(t)} \frac{\partial x_j(t)}{\partial x_i(t)} + \sum_{p=1}^{r} \sum_{j=m+1}^{N+n} \frac{\partial^{+}E(t)}{\partial x_j(t+p)} \frac{\partial x_j(t+p)}{\partial x_i(t)} \qquad (4\text{-}13)$$

avec $\frac{\partial E(t)}{\partial x_i(t)} = y_i(t) - y_i^d(t)$ pour les neurones de sortie et

$\frac{\partial E(t)}{\partial x_i(t)} = 0$ pour les neurones cachés.

A un instant donné t, il n'est pas possible d'évaluer l'expression donné par (4-13), car elle dépend des dérivées de l'erreur par rapport aux sorties futures et des dérivées de ces sorties par rapport aux sorties actuelles. Pour résoudre ce problème, on calcule les sorties et les erreurs correspondantes pour t allant de 1 à T. En annulant tous les termes qui sont en fonction des sorties pour t > T, on calcule ensuite les dérivées de l'erreur par rapport aux sorties à l'instant T, puis on effectue un retour de t=T jusqu'à t=1. D'où le nom donc, de la rétropropagation à travers le temps.

On doit noter que cet algorithme présente les inconvénients suivants:
- espace mémoire important
- trop consommateur du temps de calcul
- n'est pas général.

4-3 RESEAU RECURRENT A RETOUR D'ETAT :

A partir de l'expérience acquise dans l'identification et le contrôle des systèmes non linéaires avec les réseaux statiques, les études ont été étendues aux réseaux dynamiques. En effet, plusieurs structures de ces réseaux ont été proposées, et utilisées dans de nombreuses applications.

Dans ce paragraphe, on présente une structure des réseaux récurrents et on donne l'algorithme d'apprentissage correspondant.

4-3-1 STRUCTURE DU RESEAU :

Ce réseau est composé d'une seule couche cachée, dont les neurones sont complètement interconnectés entre eux, et d'une couche de sortie constituée par des neurones non interconnectés. Les neurones cachés réalisent une transformation non linéaire de la forme: $X(k)=f(U(k),X(k-1))$, où $U(k) \in R^n$ et $X(k) \in R^s$, n et m sont les dimensions du vecteur d'entrée U(k) et du vecteur d'état du réseau X(k) respectivement. Par contre, les neurones de la couche de sortie réalisent une transformation linéaire de la forme: $Y(k)=g(X(k))$, où $Y(k) \in R^p$, p est la dimension du vecteur de sortie. La structure du réseau est illustrée par la fig.(4-1).

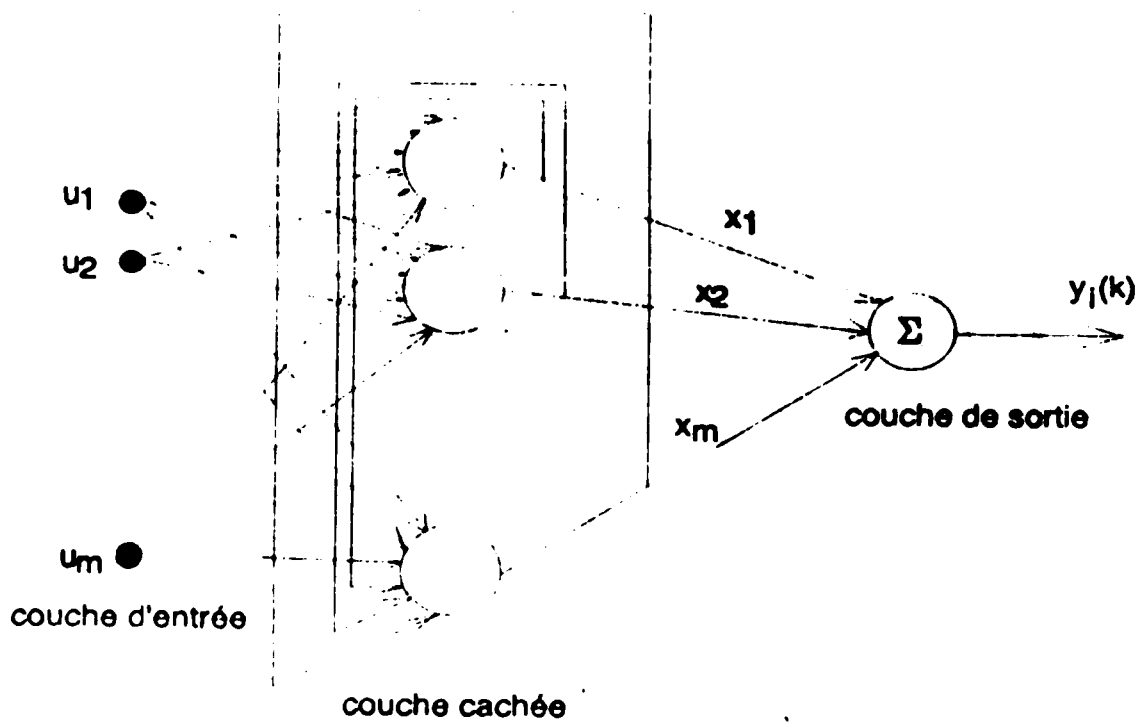


fig.(4-1): Structure du réseau récurrent.

Les états sont données par les équations suivantes:

$$z_i(k) = \sum_{j=1}^{n} b_{ij} u_j(k) + \sum_{j=1}^{m} a_{ij} x_j(k-1) \qquad 1 \le i \le m \qquad (4\text{-}14)$$

$$x_i(k) = f(z_i(k)) \qquad 1 \le i \le m \qquad (4\text{-}15)$$

Où f(.) est la fonction d'activation donnée par l'expression (1-5), a_{ij} est le poids de la connexion entre le i^{ème} et j^{ème} neurone de la couche cachée et b_{ij} celui de la connexion entre le i^{ème} neurone de la couche cachée et le j^{ème} neurone de la couche d'entrée.

L'équation de sortie, pour le cas d'un réseau mono-sortie (sans perte de généralisation dans le cas multi-sorties), est donnée par:

$$y(k) = \sum_{j=1}^{m} c_j x_j(k) \qquad (4\text{-}16)$$

avec C_j est le poids de la connexion entre le j^{ème} neurone caché et le neurone de sortie.

4-3-2 ALGORITHME D'APPRENTISSAGE :

Il s'agit d'un apprentissage supervisé qui permet d'ajuster les poids des connexions, en minimisant la fonction coût suivante:

$$E(k) = \frac{1}{2} (y_d(k) - y(k))^2 \qquad (4\text{-}17)$$

avec $y_d(k)$ est la sortie désirée.

L'approche utilisée pour minimiser (4-17) est basée sur la méthode du gradient, dont la formule itérative est donnée par l'équation suivante:

$$W(k+1) = W(k) - \alpha \frac{\partial E}{\partial W(k)}$$

où α est le pas d'apprentissage et W(.) le poids à ajuster.

A) POUR LA COUCHE DE SORTIE :

Le calcul des dérivées partielles du critère (4-17), par rapport à chaque poids $c_j(k)$, s'établit comme suit:

$$\frac{\partial E(k)}{\partial c_j(k)} = \frac{\partial[\frac{1}{2}(y_d(k)-y(k))^2]}{\partial y(k)} \frac{\partial y(k)}{\partial c_j(k)}$$

$$-\frac{\partial E(k)}{\partial c_j(k)} = -e(k)\,x_j(k)$$

avec $e(k) = (y_d(k) - y(k))$.

Les poids c_j sont ajustés en utilisant la loi d'apprentissage suivante:

$$c_j(k+1) = c_j(k) + \alpha_c e(k)\,x_j(k) \qquad \textbf{(4-18)}$$

La valeur de α_c qui correspond à la descente la plus profonde du gradient est donnée par:

$$\alpha_c = \min\{\frac{1}{2}[y_d(k) - \sum_{j=1}^{m} c_j(k)\,x_j(k) - \alpha e(k) \sum_{j=1}^{m} x_j^2(k)]^2\}$$

$$\Rightarrow \quad \alpha_c = \frac{1}{\sum_{j=1}^{m} x_j^2(k)}$$

L'expression (4-18) devient, donc:

$$c_j(k+1) = c_j(k) + e(k)\frac{x_j(k)}{\sum_{j=1}^{m} x_j^2(k)} \qquad 1 \le j \le m \qquad \textbf{(4-19)}$$

B) POUR LA COUCHE CACHEE :

Les dérivées partielles de E(k), par rapport à chaque poids a_{ij}, sont calculées de la façon suivante:

$$\frac{\partial E(k)}{\partial a_{ij}(k)} = -e(k)\frac{\partial y(k)}{\partial a_{ij}(k)}$$

$$\frac{\partial y(k)}{\partial a_{ij}(k)} = c_i(k)\,f'(z_i(k))\frac{\partial zi(k)}{a_{ij}(k)}$$

$$= c_i(k)\,f'(z_i(k))\,x_j(k-1)$$

avec $f'(z_i(k)) = 0.5(1-x_i^2(k))$

$$\rightarrow \frac{\partial E(k)}{\partial a_{ij}(k)} = -\frac{1}{2} e(k)\, c_i(k)\, (1-x_i^2(k))\, x_j(k-1)$$

les poids a_{ij} sont ajustés selon la loi d'adaptation suivante:

$$a_{ij}(k+1) = a_{ij}(k) + \frac{\alpha_a}{2} e(k)\, c_i(k)\, (1-x_i^2(k))\, x_j(k-1) \qquad (4\text{-}20)$$
$$1 \le j \le m \qquad\qquad 1 \le i \le m$$

avec α_a est le pas d'apprentissage pour les poids a_{ij}.

De la même manière, la loi d'adaptation pour les poids b_{ij} est donnée par:

$$b_{ij}(k+1) = b_{ij}(k) + \frac{\alpha_b}{2} e(k)\, c_i(k)\, (1-x_i^2(k))\, u_j(k) \qquad (4\text{-}21)$$
$$1 \le i \le m \qquad\qquad 1 \le j \le n$$

4-3-3 EXEMPLES DE SIMULATION :

Dans tous les exemples RN désigne la structure neuronale décrite dans le paragraphe 4-3.

EXEMPLE 4-1 :

Dans cet exemple, on considère l'identification récursive d'un système non linéaire du premier ordre, dont le comportement dynamique est décrit par l'équation suivante:

$$y_s(k+1) = f[y_s(k)] + u(k)$$

où f[.] est donnée par l'équation (3-8). Cette fonction est supposée inconnue.

Le modèle d'identification utilisé possède la structure suivante:

$$y_m(k+1) = RN[y_s(k)] + u(k)$$

avec RN comportant 15 neurones sur la couche cachée.

Pour entraîner ce modèle, nous avons utilisé l'algorithme décrit dans (4-3-2), avec $\alpha_a = \alpha_b = 0.25$, et une commande aléatoire uniformément répartie dans l'intervalle [-2 2].

La réponse du système et celle du modèle obtenu après 20000 itérations d'entraînement à l'entrée teste donnée par l'expression (3-9) sont représentées, par deux courbes superposées, dans la fig.(4-2). On remarque que le modèle obtenu présente le même comportement que le système considéré, avec une erreur très faible comme le montre la fig.(4-3). On note que, pour les mêmes entrées d'entraînement et de test, on a obtenu un résultat plus précis et avec un temps d'apprentissage plus petit (fig.(4-4)) que dans le cas de l'exemple (3-1) où on a utilisé un réseau MLP statique.

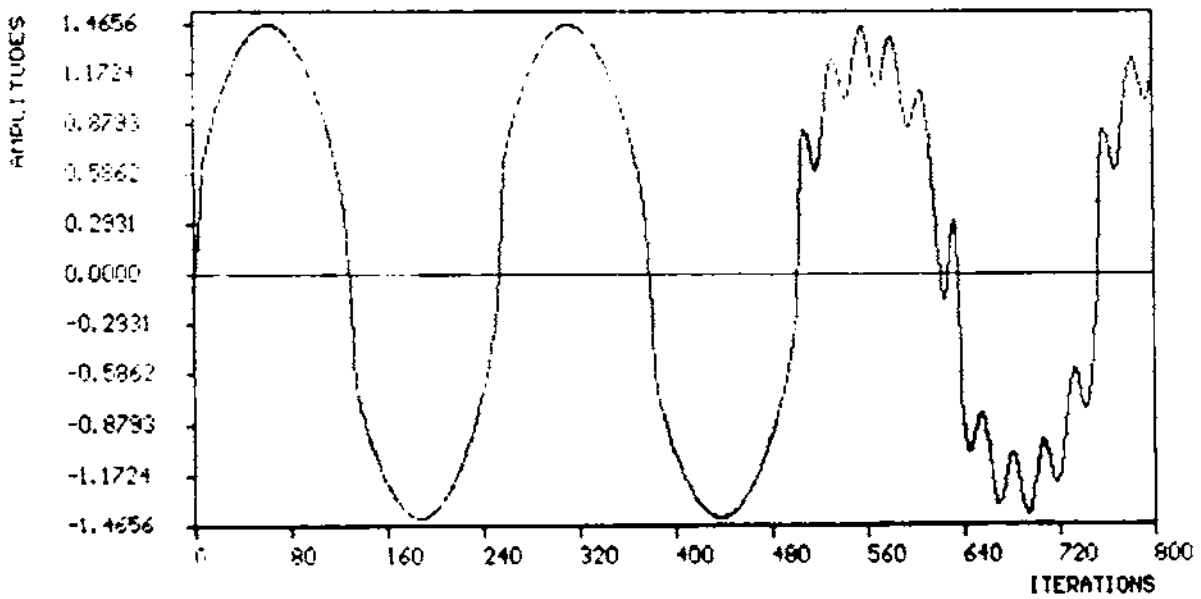


fig.(4-2): Sorties du système et celle du modèle superposées pour l'exemple (4-1).

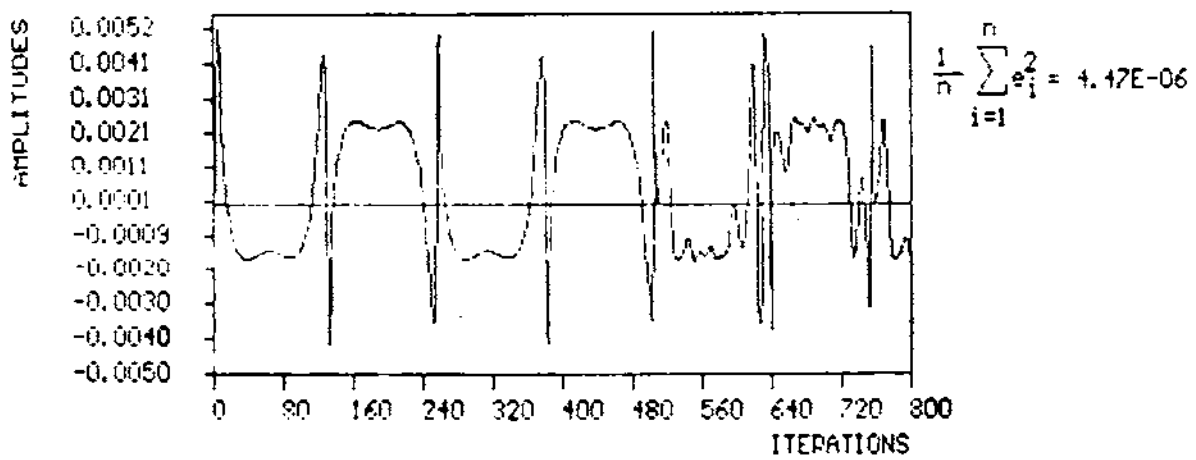


fig.(4-3): Erreur d'identification pour l'exemple 4-1.

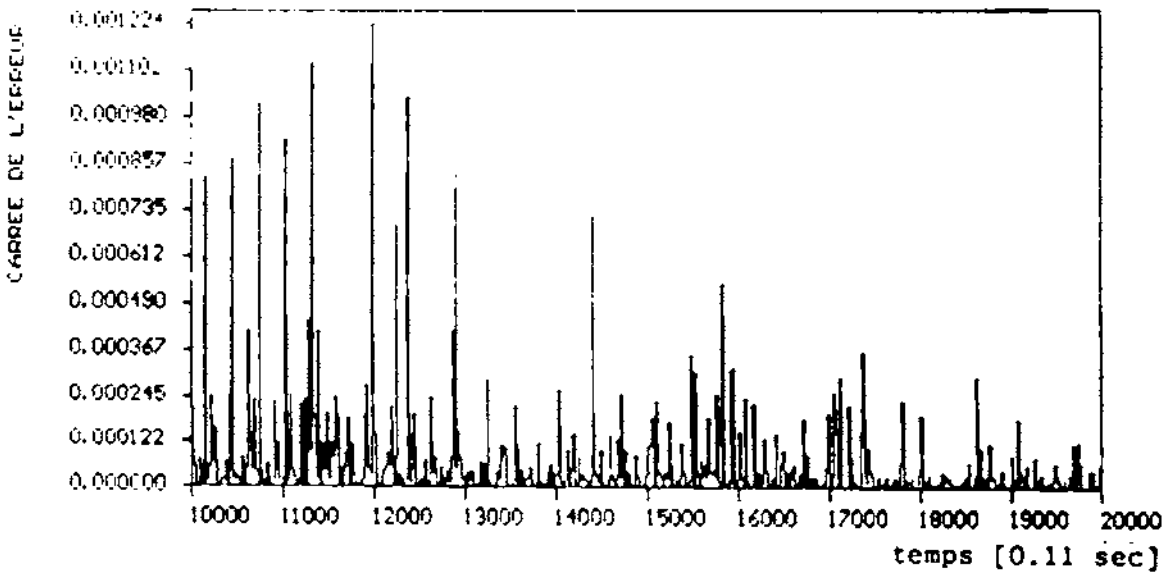


fig.(4-4): variation de l'erreur d'identification pendant la phase d'apprentissage pour l'exemple 4-1.

EXEMPLE 4-2:

Dans cet exemple, le système à identifier est décrit par l'équation aux différences du troisième ordre donnée par (3-11).

Un réseau récurrent à retour d'état avec 5 neurones sur la couche d'entrée et 35 sur la couche cachée est utilisé pour une identification série-parallèle de ce système. L'algorithme décrit dans (4-3-2) (avec α_a = 0.5, α_b = 0.6) et une commande aléatoire uniformément répartie dans l'intervalle [-1 1] ont été utilisées pour l'entraînement récursif de ce modèle.

La sortie du modèle obtenu après 20000 itérations d'entraînement, et celle du système pour l'entrée donnée par (3-15) sont représentées dans la fig.(4-5). On constate que l'écart entres les deux sorties est faible (fig.(4-6)), et que la durée d'apprentissage (fig.(4-7)) est plus courte que celle dans l'exemple (3-4) où on a utilisé un réseau MLP statique.

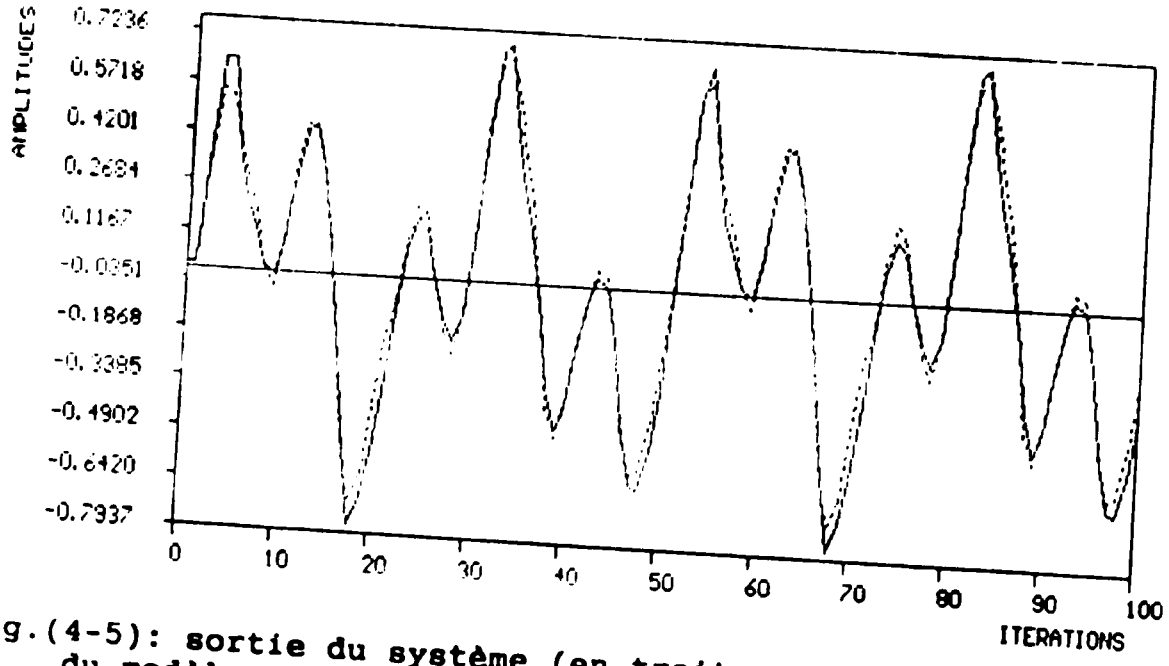


fig.(4-5): sortie du système (en traits pleins) et celle du modèle (en traits pointillés) pour l'exemple 4-2.

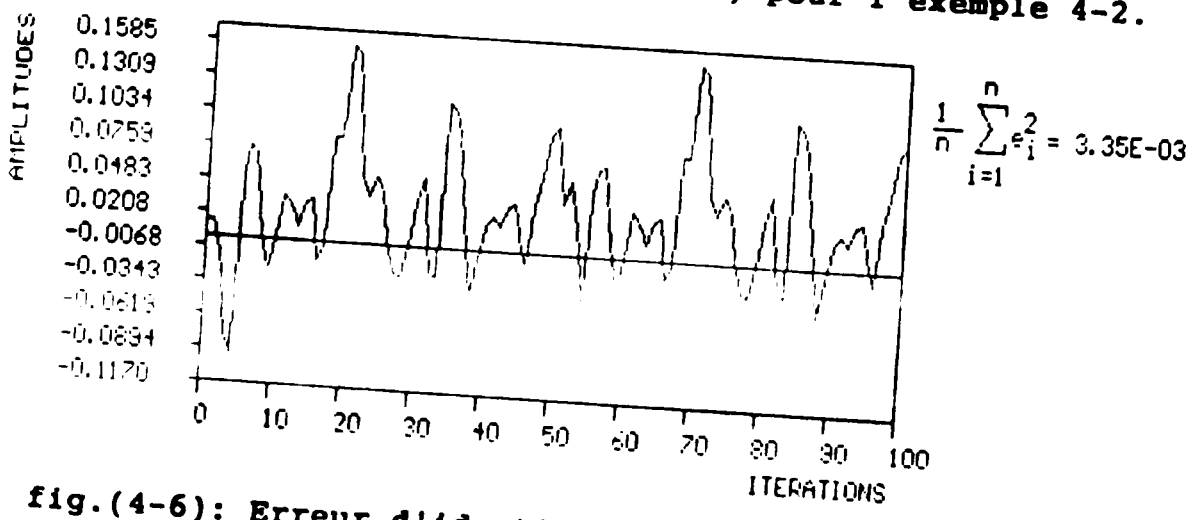


fig.(4-6): Erreur d'identification pour l'exemple 4-2.

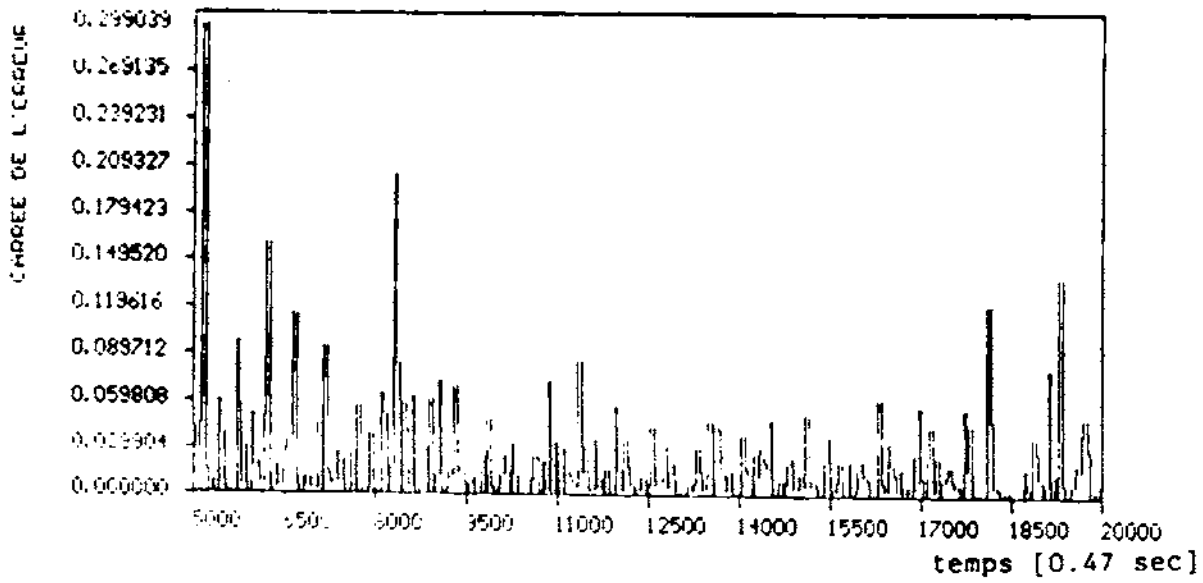


fig.(4-7): variation de l'erreur d'identification pendant la phase d'apprentissage pour l'exemple 4-2.

EXEMPLE 4-3 :

Dans tous les exemples précédents, nous n'avons considéré que les systèmes monovariables. Cet exemple concerne l'identification récursive d'un système multivariables, à deux entrées de commande et deux sorties. La dynamique directe de ce système est décrite par les équations suivantes:

$$y_1(k+1) = f_1[y_1(k), y_2(k)] + u_1(k)$$

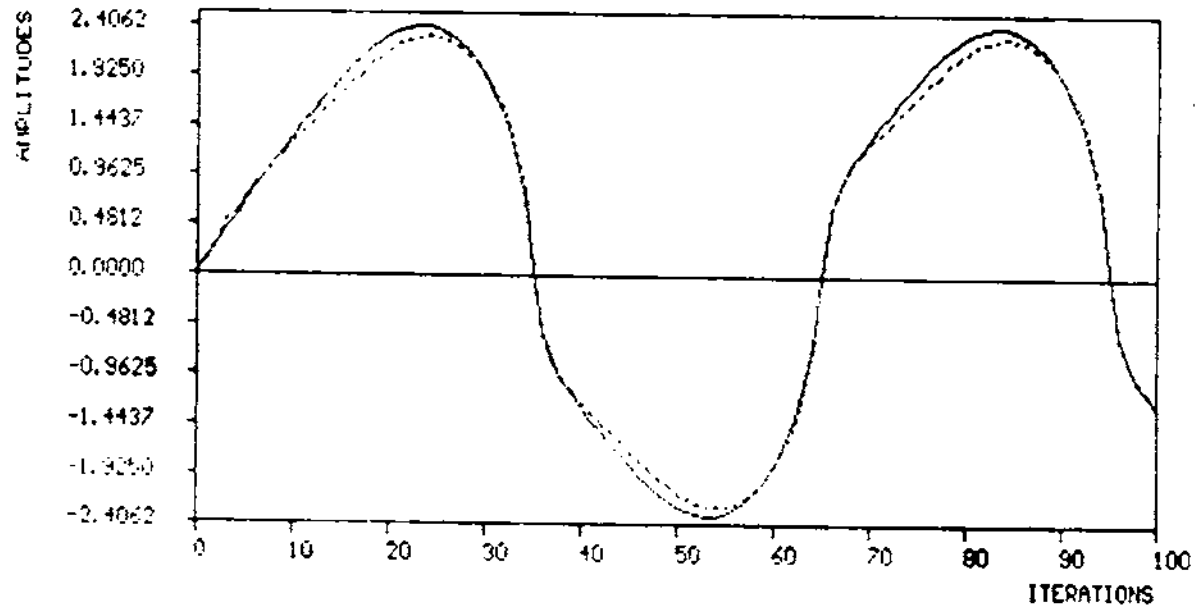
$$y_2(k+1) = f_2[y_1(k), y_2(k)] + u_2(k)$$

avec $$f_1[y_1(k), y_2(k)] = \frac{y_1(k)}{1+y_2^2(k)}$$

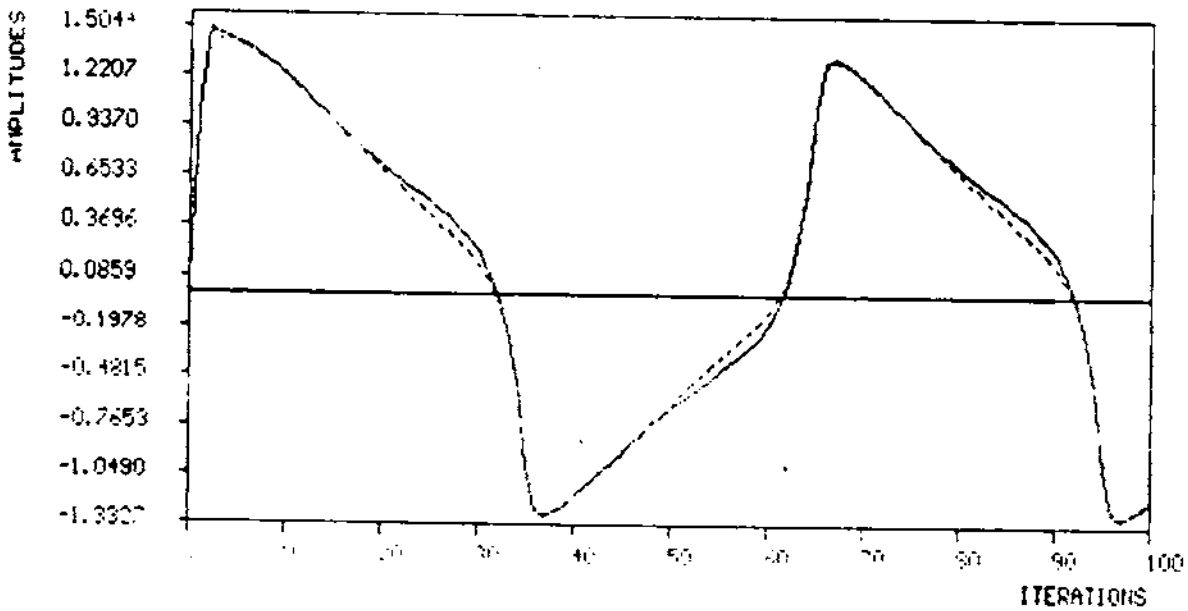
$$f_2[y_1(k), y_2(k)] = \frac{0.5y_1(k)+y_2(k)}{1+y_2^2(k)}$$

Deux réseaux récurrents ont été utilisés pour approximer les fonctions f_1 et f_2. Chacun de ces réseaux comporte deux neurones sur la couche d'entrée et 25 sur la couche cachée. Des commandes aléatoires uniformément réparties dans l'intervalle [-2 2] sont appliquées au système et au modèle, et les poids des connexions pour chaque réseau sont ajustés en utilisant l'algorithme décrit dans (4-3-2).

Les sorties du modèle et du système, pour les entrées $u_1(k)=\sin(2\pi k/60)$, $u_2(k)=\cos(2\pi k/60)$ sont représentées dans la fig.(4-8). On remarque que l'erreur de prédiction est assez faible.



(a): $y_1(k)$ (en traits pleins) et $y_{m1}(k)$ (en traits pointillés)



(b): $y_2(k)$ (en traits pleins) et $y_{m2}(k)$ (en traits pointillés)

fig.(4-8): Les performances du modèle d'identification pour l'exemple 4-3.

EXEMPLE 4-4 :

Dans tous les exemples précédents nous avons supposé que le bruit de mesure est négligeable. Cette fois-ci, on étudie l'effet de ce bruit sur les performances du modèle neuronale d'identification. La fig.(4-9) présente une structure générale adaptée à l'identification d'un système non linéaire en présence d'un bruit de mesure. On suppose qu'il s'agit d'un bruit additif à valeur moyenne nulle.

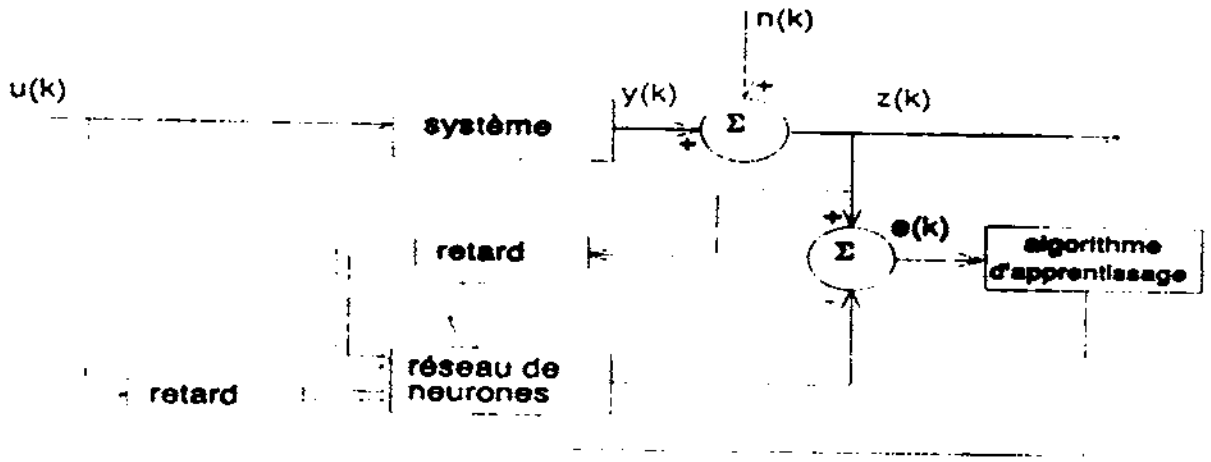


fig.(4-9): Structure d'identification d'un système non linéaire en présence d'un bruit de mesure.

Pour faire cette étude, on considère l'identification du système non lineaire:

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

La structure du réseau récurrent à retour d'état, utilisée pour l'identification de ce système, comporte 2 neurones sur la couche d'entrée et 20 sur la couche cachée. La procédure d'apprentissage est répétée pour plusieurs valeurs de l'amplitude maximale (Am) du bruit.

La réponse du modèle obtenu, dans le cas où Am = 0 (bruit négligé), et celle du système à l'entrée donnée par la fig.(4-10) sont représentées dans la fig.(4-11). L'écart entre les deux sorties est assez faible et la valeur moyenne du carrée de l'erreur, dans ce cas, est de 1.89E-3.

Le tableau ci-dessous représente les variations de la moyenne du carré de l'écart (noté Er) entre la sortie du système et celle du modèle d'une part, et les variations de la moyenne du carré de l'écart (noté Em) entre la sortie mesurée et celle du modèle d'autre part, pour les valeurs 0, 0.1, 0.2 et 0.5 de Am et l'entrée donnée par la fig.(4-10).

Am	0	0.1	0.2	0.5
Em	1.89E-3	5.24E-3	2.38E-2	1.37E-1
Er	1.89E-3	2.59E-3	6.61E-3	4.68E-2

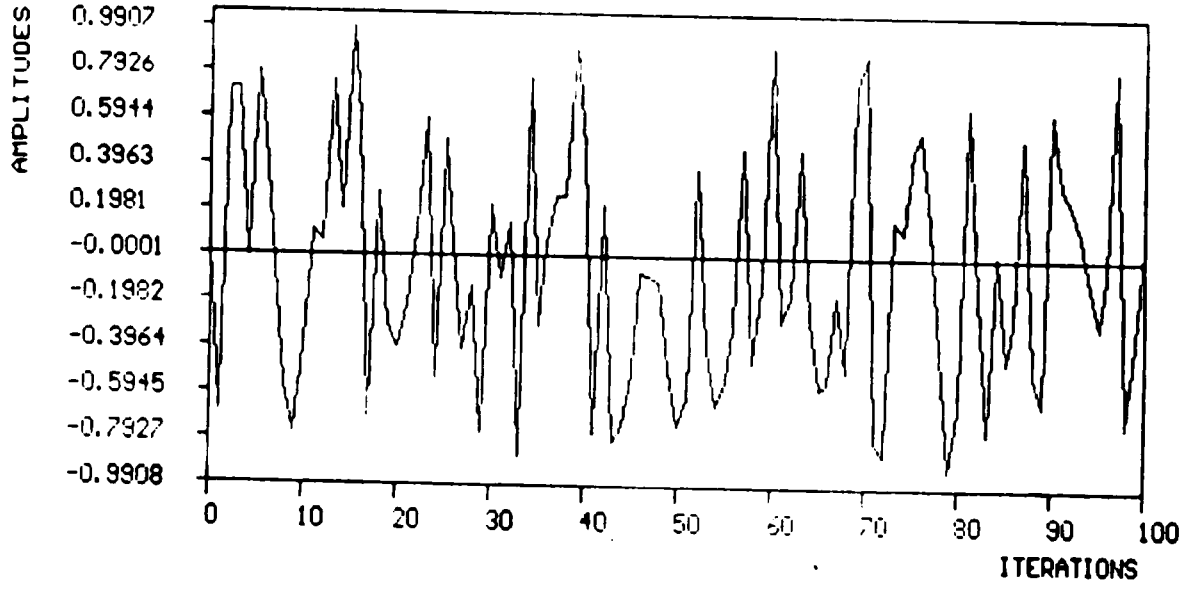


Fig.(4-10): L'entrée de teste utilisée dans l'exemple 4-4.

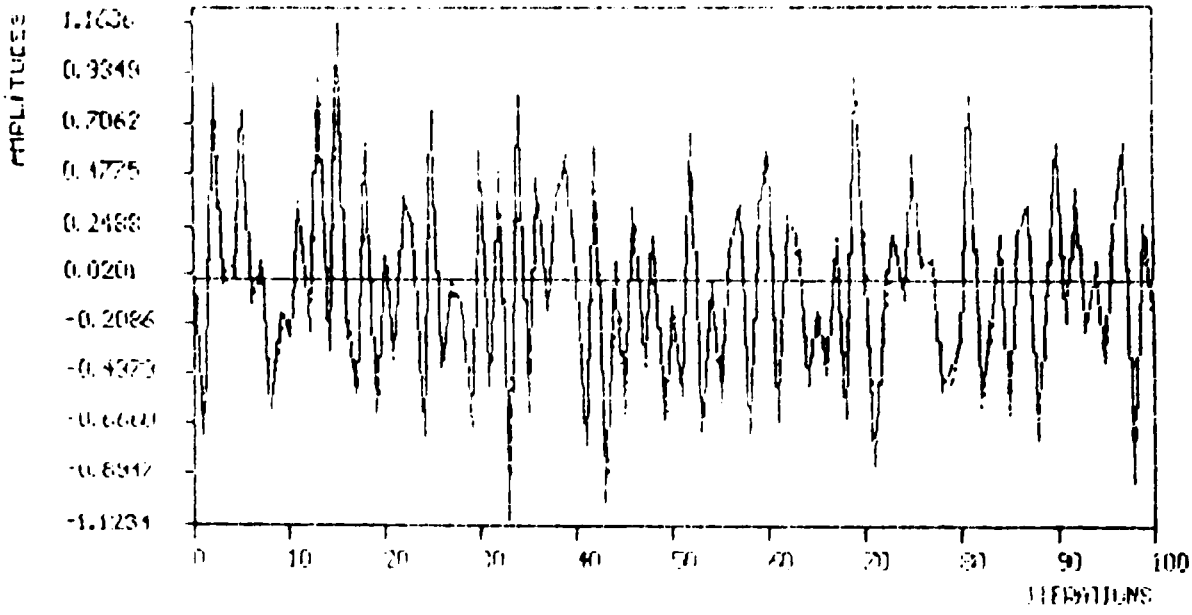


fig.(I 4-11): La sortie du système (en traits pleins) et celle du modèle (en traits pointillés) pour l'exemple 4-4.

On constate quela valeur de Em et celle de Er croit avec l'augmentation de l'amplitude Am du bruit, et que la valeur de Em croit plus rapidement que celle de Er; le modèle a tendance à identifier la sortie réelle du système au lieu de celle mesurée.

4-4 REJECTION DES PERTURBATIONS :

Le problème de réjection des perturbations additives en entrée a été envisagé par Mukhopadhyay et al [38], et présenté dans le chapitre précédent. Dans ce paragraphe, on reprend l'étude de ce problème, mais cette fois-ci les différents types de perturbations sont supposées additives en sortie. On distingue deux cas différents:

4-4-1 CAS D'UNE PERTURBATION LINEAIRE :

Soit le système monovariable en présence d'une perturbation externe, décrit par l'équation suivante:

$$y(k+1) = f[y(k), y(k-1), \ldots, y(k-n+1); u(k), u(k-1), \ldots, u(k-m+1)] + v(k) \qquad (4\text{-}22)$$

où la perturbation v(k) est donnée par une équation aux différences linéaire de la forme:

$$v(k+1) = \sum_{i=0}^{p-1} \gamma_i v(k-i) \qquad (4\text{-}23)$$

A partir de (4-22) et (4-23) on obtient:

$$v(k) = \sum_{i=0}^{p-1} \gamma_i [y(k-i) - f[y(k-i-1), \ldots, y(k-i-n); u(k-i-1), \ldots, u(k-i-m)]] \qquad (4\text{-}24)$$

L'équation (4-24) nous permet d'écrire (4-22) de la manière suivante:

$$y(k+1) = f[y(k), \ldots, y(k-n+1); u(k), \ldots, u(k-m+1)] + \sum_{i=0}^{p-1} \gamma_i y(k-i) - \sum_{i=0}^{p-1} \gamma_i f[y(k-i-1), \ldots, y(k-i-n); u(k-i-1), \ldots, u(k-i-m)]$$

ou d'une manière équivalente, sous la forme:

$$y(k+1) = F[Y(k), U(k)] \qquad (4\text{-}25)$$

avec $Y(k) = [y(k), y(k-1), \ldots, y(k-n-p+1)]^T$,

$U(k) = [u(k), u(k-1), \ldots, u(k-m-p+1)]$

4-4-2 CAS D'UNE PERTURBATION NON LINEAIRE :

Dans ce cas, la perturbation est considérée comme étant la réponse libre d'un système non linéaire. Si un système non linéaire en présence de cette perturbation, est décrit par les équations suivantes:

$$\begin{aligned} y(k+1) &= f[Y(k), U(k)] + v(k) \\ v(k+1) &= g[V(k)] \end{aligned} \qquad (4\text{-}26)$$

où:

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

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

Alors, l'équation de v(k) peut s'écrire de la façon suivante:

$$v(k) = y(k+1) - f[Y(k), U(k)]$$

et

$$v(k+1) = g[W(k)] \qquad (4\text{-}27)$$

où

$$W(k) = [y(k+1) - f(Y(k), U(k)), y(k) - f(Y(k-1), U(k-1)), \ldots, y(k-p+2) - f(Y(k-p+1), U(k-p+1))]^T \qquad (4\text{-}28)$$

en utilisant les équations (4-27) et (4-28), l'expression (4-26) devient:

$$y(k+1) = f[Y(k), U(k)] + g[W(k-1)] \qquad (4\text{-}29)$$

il est clair que l'équation (4-29) dépend des (n+p) et (m+p) valeurs passées de la sortie y(k) et la commande u(k), respectivement. On peut l'écrire donc, de la manière suivante:

$$y(k+1) = \bar{f}[\bar{Y}(k), \bar{U}(k)] \qquad (4\text{-}30)$$

avec

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

et

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

on conclut que, la solution du problème de rejection des perturbations additives en sortie consiste (dans les deux cas) à utiliser un modèle d'identification d'ordre n+p.

4-4-3 SIMULATION :

EXEMPLE 4-5 :

Cet exemple concerne le cas où la perturbation externe est décrite par un modèle linéaire. On considère l'identification récursive du système non linéaire du premier ordre:

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

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

$$\begin{aligned} v_1(k+1) &= \cos(\phi)\, v_1(k)+\sin(\phi)\, v_2(k) \\ v_2(k+1) &= -\sin(\phi)\, v_1(k)+\cos(\phi)\, v_2(k) \\ v(k) &= 0.5 v_1(k) \end{aligned}$$

$$avec \quad \phi=\frac{\pi}{6}$$

Dans un premier cas, nous avons utilisé le modèle d'identification:

$$y_m(k+1)=RN[y(k),u(k)]$$

RN comportant 40 neurones sur la couche cachée. Une commande aléatoire uniformément répartie dans l'intervalle [-1 1] est utilisée pour entraîner ce modèle.

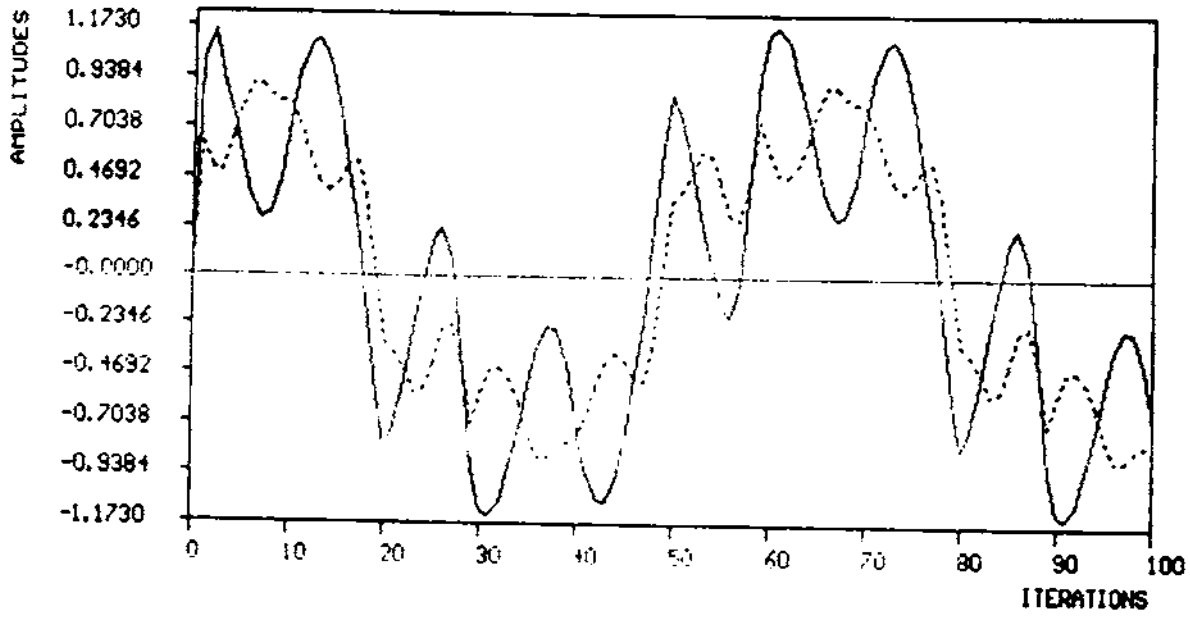
La sortie du modèle obtenu et celle du système pour l'entrée donnée par $u(k)=0.25\sin(2\pi k/60)+0.5\cos(2\pi k/60)$, sont représentées dans la fig.(4-12-a). On constate que l'écart entre les deux sorties est important.

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

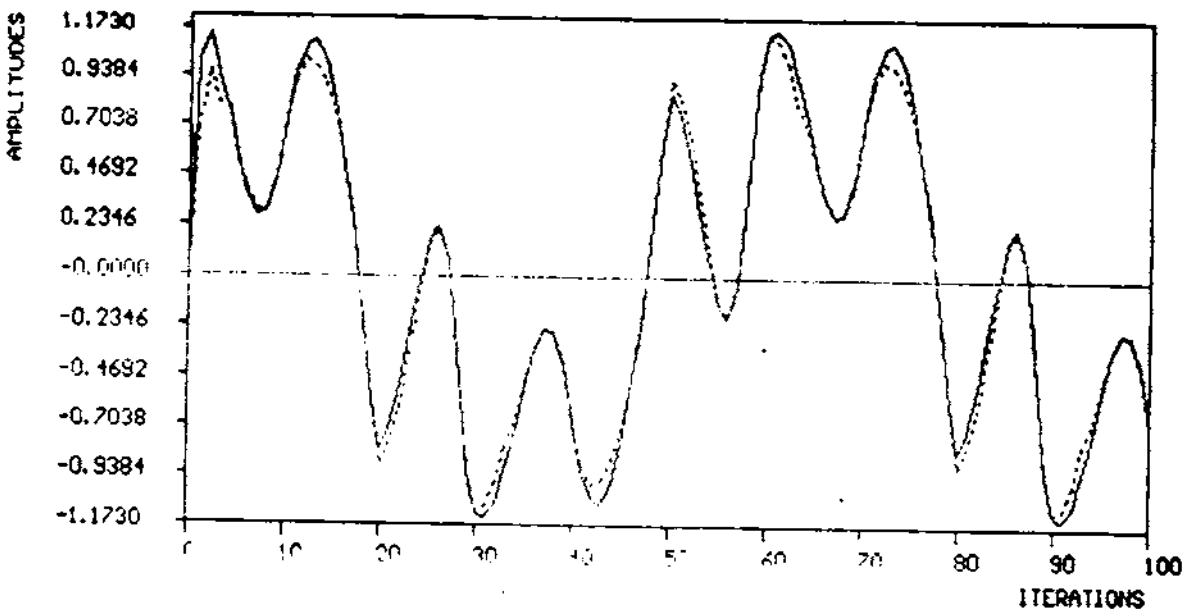
$$y_m(k+1)=RN[y(k),y(k-1),y(k-2);u(k),u(k-1),u(k-2)]$$

avec RN est la même structure neuronale que celle utilisée dans le premier cas.

La fig.(4-12-b) représente la réponse de ce modèle et celle du système à l'entrée donnée dans le cas précédent. On remarque que l'écart entre les deux réponses est négligeable et que la perturbation v(k) ne présente, dans ce cas, aucun effet sur les performances du modèle obtenu.



(a) sans réjection.



(b) avec réjection.

fig.(4-12): les sorties des modéles obtenus (en taits pointillés) et celle du système (en traits pleins) pour l'exemple 4-5.

EXEMPLE 4-6 :

Dans cet exemple, il s'agit d'identifier la dynamique directe du système non linéaire donné dans l'exemple précédent. Cette fois, la perturbation v(k) est décrite par le modèle non linéaire:

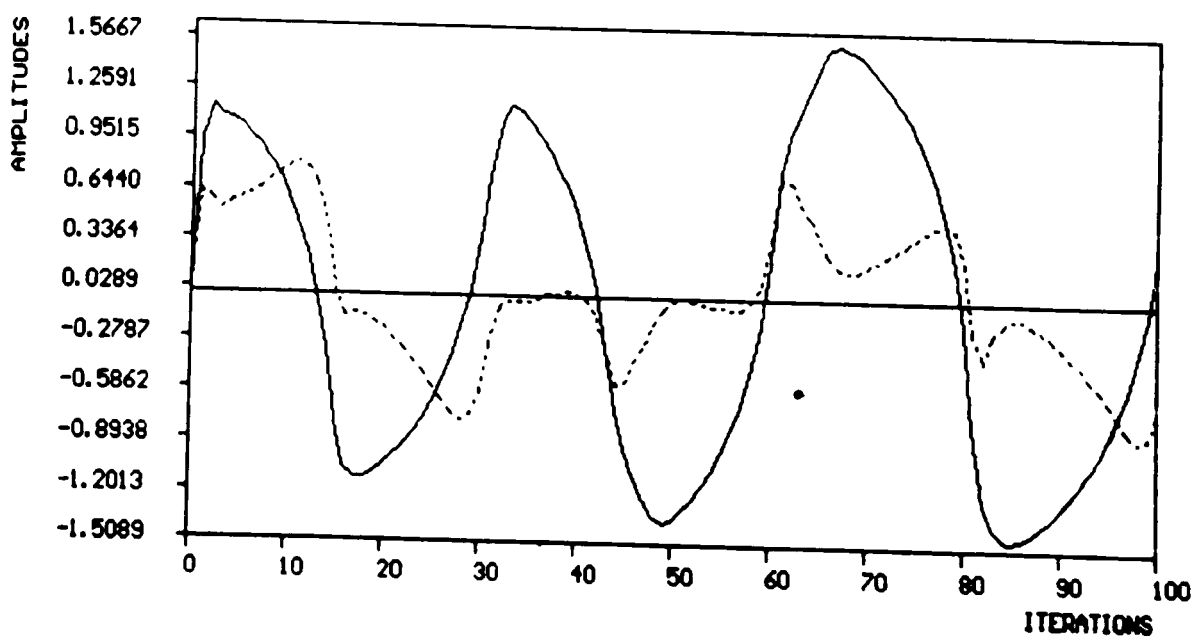
$$v_1(k+1) = v_1(k) + 0.2\,v_2(k)$$
$$v_2(k+1) = -0.2\,v_1(k) + v_2(k) - 0.1\,(v_1^2(k) - 1)\,v_2(k)$$
$$v(k) = 0.4\,v_1(k)$$

Le modèle d'identification utilisée est donné par:

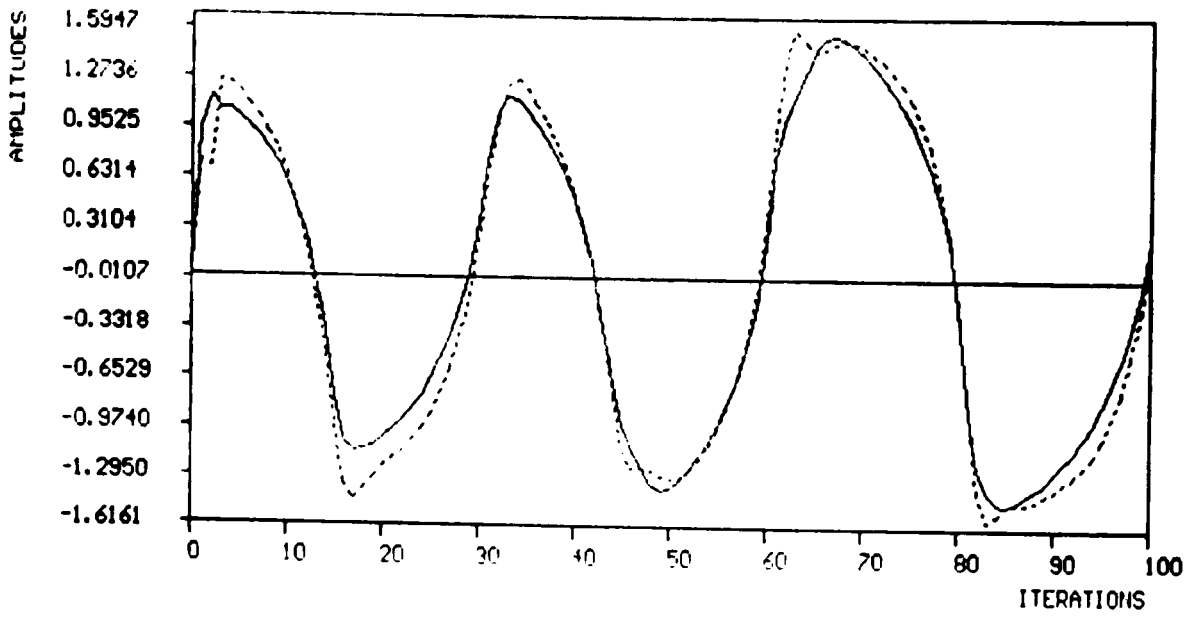
$$y_m(k+1) = RN[y(k), \ldots, y(k-s); u(k), \ldots, u(k-s)]$$

avec la structure neuronale RN comporte 40 neurones sur la couche cachée. En utilisant une commande aléatoire uniformément répartie dans l'intervalle [-1 1] et selon que p = 0 (sans de rejection) ou p = 2 (avec rejection), les deux modèles correspondants ont été entraînés. Ensuite l'entrée donnée dans l'exemple précédent a été utilisée pour comparer les sorties des modèles obtenus dans les deux cas (p = 0 et p = 2) avec celle du système considéré.

La fig.(4-13-a) montre que l'écart entre la sortie du système et celle du modèle obtenu pour p = 0 est important. Cet écart est faible pour p = 2 comme le montre la fig.(4-13-b).



(a): pour p=0.



(b) : pour p = 2.

fig.(4-13): les sorties des modéles obtenus (en traits pointillés) et celle du systéme (en traits pleins) pour l'exemple 4-6.

4-6 CONCLUSION :

Nous venons de voir dans ce chapitre, l'application des réseaux récurrents à l'identification des systèmes non linéaires. Une nouvelle structure des réseaux récurrents a été introduite et son efficacité dans les problèmes d'identification a été démontrée par des exemples de simulation. En effet cette structure présente les mêmes performances que les réseaux MLP statiques, mais avec un temps d'apprentissage plus court.

Nous avons envisagé le problème de réjection des perturbations additives en sortie. Des justifications théoriques pour les modifications nécessaires dans le modèle d'identification ont été données, ensuite nous avons vérifié par simulation la validité de ces modifications.

CONCLUSION ET PERSPECTIVES

CONCLUSIONS ET PERSPECTIVES

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

Après avoir donné un aperçu général sur les réseaux de neurones, présenté les réseaux statiques et étudié en détail les structures neuronales MLP et RBF, ainsi que leurs algorithmes d'apprentissage, nous avons envisagé l'application de ces structures à l'identification des systèmes non linéaires.

Ayant présenté les structures d'identification en utilisant les réseaux de neurones, on a considéré ensuite le problème d'identification des systèmes non linéaires avec les structures statiques MLP et RBF. L'efficacité de ces deux structures a été démontrée par des exemples de simulation et des résultats satisfaisants ont été obtenus. En effet, ces exemples ont montrés que ces réseaux sont des approximateurs universels pour les fonctions non linéaires inconnues.

L'apprentissage dans les réseaux MLP, repose sur les méthodes d'optimisation non linéaires (les méthodes du gradient). Cependant, la surface d'erreur pour ces architectures est souvent très complexe et présente des caractéristiques peu satisfaisantes pour effectuer une descente de gradient (minima locaux, plateau où les pentes sont très faibles,...). Ceci est l'inconvénient majeur des reseaux MLP.

L'utilisation d'un réseau RBF, pour lequel l'apprentissage est basé sur les méthodes d'optimisation linéaire dans les problèmes d'identification, fournit une autre alternative plus efficace. Malheureusement, le nombre de neurones cachés augmente, dans ce cas, avec le degré de complexité du problème traité d'une façon considérable: c'est l'inconvénient des réseaux à une seule couche cachée, par conséquent l'apprentissage devient très lent.

[19] M. R. Azimi-Sadjadi, S. Sheedvash and F. O. Trujillo, "Recursive dynamic node creation in multilayer neural networks", IEEE Trans. Neural Networks, vol. 4, no. 2, pp 242-256, Mar. 1993.
[20] H. A. Malki and A. Moghaddamjoo, "Using the Karhunen-Loeve transformation in the backpropagation training algorithm", IEEE Trans. Neural Networks, vol. 2, no. 1, Jan. 1991.
[21] J. A. Leonard, M. A. Kramer and L. H. Ungar, "Using radial basis functions to approximate a function and its error bounds", IEEE Trans. Neural Networks, vol. 3, no. 4, pp 624-627, Jul. 1992.
[22] S. Chen, C. N. Cowan and P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", IEEE Trans. Neural Networks, vol. 2, no. 2, pp 302-309, Mar. 1991
[23] J. P. Tou, A. [illegible], "Pattern Recognition Principles", Addison-wesly 1974.
[24] C. Foulard, S. Gentil et J. P. Sandraz, "Commande et régulation par calculateur numérique", Eyrolles, Paris, 1987.
[25] L. Personnaz, G. Dreyfus et I. Guyon, "Les machines neuronales", La recherche, vol. 19, no. 204, pp 1362-1371, Nov. 1988.
[26] K. S. Narendra and K. Parthasarathy,"Identification and Control of dynamical systems using neural networks", IEEE Trans. Neural Networks, vol. 1, no. 1, pp 4-27, Mar. 1990.
[27] S. R. Chu, R. Shoureshi and N. Tenorio, "Neural Network for system identification", IEEE Control Systems Magazine, vol. 10, no. 3, pp 31-35, Apr 1990.
[28] S. A. Billings, "Identification of nonlinear systems-a survey", IEE Proc., vol. 127 Pt.D, no. 6, pp 272-285, Nov. 1980.
[29] S. A. Billings and I. J. Leontaritis, "Identification of nonlinear systems using parameter estimation techninques", Research Report, no. 117, Sep. 1982, University of Sheffield, UK.
[30] P. Eykhoff, "System identification", Wiley, New York, 1974.
[31] [illegible] systems", Wiley, New York, 1980.
[32] K. J. Hunt, D. Sbarbaro, R. Zbikowski and P. J. Gawthrop, "Neural Networks for control systems-A survey", Automatica, vol.28, no. 6, pp 1083-1112, Dec. 1992.
[33] S. Chen and S. A. Billings, "Neural Networks for nonlinear system modeling and identification", Int. J. Control, vol. 56, no. 2, pp 319-346, Aug. 1992.
[34] M. Boumehraz,"Identification et contrôle avec réseaux de neurones", Thèse de magister présentée à l'institut d'Electronique, Université de Setif, 1993.
[35] S. Chen, S. A. Billings, C. F. N. Cowan and P. M. Grant, "Practical identification of NARMAX models using radial basis functions", Int. J. Control, vol. 52, no. 6, pp 1327-

1350, 1990.
[36] S. Chen, S. A. Billings, C. F. N. Cowan and P. M. Grant, "Non-linear sytems identification using radial basis functions", Int. J Systems SCI., vol. 21, no. 12, pp 2513-2530, 1990.
[37] R. M. Sanner and J. E. Slotine, "Gaussien networks for direct adaptive control", IEEE Trans. Neural Networks, vol.3, no. 6, pp 837-863, Nov. 1992.
[38] S. Mukhopadhyay and K. S. Narendra, "Disturbance Rejection in nonlinear systems using neural networks", IEEE Trans. Neural Networks, vol. 4, no. 1, pp 63-72, Jan. 1993.
[39] S. Z. Qin, H. T. Su and T. J. MacAvoy,"Comparaison of four neural net learning methods for dynamic system identification", IEEE Trans. Neural Networks, vol. 3 no. 1, pp 122-130, Jan. 1993.
[40] D. E. Rumelhart and J. L. McClelland, "Parallel Distributed Processings" vol. 1, pp 318-362 Cambridge, MA: MIT Press, 1986.
[41] O. Nerrand, P. Roussel-Ragot, D. Urbani, L. Personnaz, and G. Dreyfus, "Training Recurrent Neural Networks: Why and How? An Illustration in Dynamical Process Modeling", IEEE Trans. Neural Networks, vol. 5, no. 2, pp 178-184, Mar. 1994.
[42] O. Olurotimi, "Recurrent neural network training with feedforward complexity", IEEE Trans. Neural Networks, vol. 5, no. 2, pp 185-197, Mar. 1994.
[43] B. Srinivasan, U. R. Prasad and N. J. Rao, "Back propagation through adjoints for the identification of nonlinear dynamic systems using recurrent neural models", IEEE Trans. Neural Networks, vol. 5, no. 2, pp 213-228, Mar. 1994.
[44] J. T. Connor, R. D. Martin and L. E. Atlas,"Recurrent Neural Networks and Robust Time series Prediction", IEEE Trans. Neural Networks, vol. 5, no. 2, pp 240-253, Mar. 1994.
[45] A. G. Parlos, K. T. Chong and A. F. Atiya, "Application of the recurrent multilayer Perceptron in modeling complex process dynamics", IEEE Trans. Neural Networks, vol. 5, no. 2, pp 255-266, Mar. 1994.
[46] S. W. Piché, " Steepest descent algorithms for neural network controllers and filters", IEEE Trans. Neural Networks, vol. 5, no. 2, pp198-212, Mar. 1994.