REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE BLIDA

INSTITUT D'ELECTRONIQUE

OPTION : CONTROLE

MEMOIRE DE MAGISTER

Etudié par

LAGRAA NACEREDDINE

THEME

Identification des systèmes non-linéaires par les réseaux de neurones d'ordre élevé
Application à une ruche apicole

Devant le jury

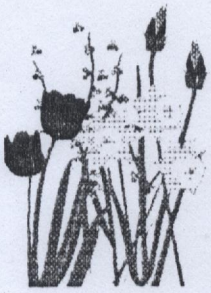
H.SALHI	MAITRE DE CONFERENCE	USTB	PRESIDENT
M.ATTARI	MAITRE DE CONFERENCE	USTHB	EXAMINATEUR
B. KAZED	CHARGE DE COURS	USTB	EXAMINATEUR
F.BOUDJEMA	MAITRE DE CONFERENCE	ENPA	INVITE
A.GUESSOUM	PROFESSEUR	USTB	RAPPORTEUR
H. BOUGHRIRA	CHARGE DE COURS	USTB	RAPPORTEUR

1999/2000

الإهـــداء

- إلى اللذين آملآ إذ آمنآ و سميا إذ سعيا والدي العزيزين.
- إلى إخوتي و كل أفراد عائلتي.
- إلى أحبتي و أصدقائي.
- إلى نادية و جمال.

نصرالدين



Remerciements

J'exprime ma profonde reconnaissance à mes promoteurs M^{me} H.Boughrira et M^{r} A.Guessoum qui en dirigeant ce travail, m'ont profiter de leurs connaissances et de leurs précieux conseils.

Je tiens à remercier aussi le comité de lecture composé de Mrs H.Salhi et B.Kazed qui se sont donnés la peine de faire le parcours des corrections nécessaires pour une thèse aussi parfaite que possible.

J'exprime ma connaissance et je témoigne de ma gratitude à mes professeurs Mrs R.Djellouli et M.R.Ouared de l'E.N.P.A. et tout l'ensemble des enseignants de l'institut de l'électronique qui ont contribué à ma formation.

Mes sincères remerciements sont aussi adressés à toutes les personnes qui n'ont cessé de m'apporter aide et soutien lors de l'élaboration de ce travail en particulier : M.Oubbati, M.Bentireche, D.Bentireche et à tout mes ammis.

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

ملخـــــــص

إن استعمال الشبكات العصبونية على إختلاف تشكيلاتها في عدة ميادين مختلفة أصبح شيئا إعتياديا.
الهدف ممن هذا البحث هو دراسة فئة خاصة من الشبكات العصبونية تعرف باسم الشبكات ذات الدرجات العليا، لأجل ذلك اخترنا ثلاثة أنواع و هي:
الشبكات العصبونية الطبقية و الشبكات العصبونية الوحدوية و الشبكات العصبونية التراجعية و هذا لدراستها و مقارنة خصائصها مع خصائص نظائرهم من شبكات الدرجات الأولى.
استعملنا هذه التركيبات الجديدة لمطابقة الأنظمة الديناميكية غيرالخطية ثم لمطابقة نظام طبيعي متمثل في خلية نحل.

ABSTRACT

Several neural networks have been developed and applied to various engineering problems. This thesis studies learning properties of one class of neural networks, known as high-order neural networks. In this case, the multilayer network, modular networks and current networks are studied and developed. Their learning properties are compared by simulations with these of nets of first-order.
Finally, these new architectures of nets applies to the identification of dynamical systems, then to identificatiion of a real system which is the bee hive.

RESUME

Le but de ce travail est d'étudier les caractéristiques d'apprentissage des nouveaux réseaux de neurones appelés « les réseaux de neurones d'ordre élevé ». pour cela les caractéristiques des réseaux d'ordre élevé multicouches, modulaires et récurrents sont comparés par des simulations avec celles des réseaux d'ordre un.
Ces nouvelles architectures des réseaux sont utilisés pour l'identification des systèmes dynamiques et ensuite à l'identification d'un système réel, qui est dans notre cas la ruche apicole

SOMMAIRE

CHAPITRE II : LES RESEAUX DE NEURONES D'ORDRE ELEVE

CHAPITRE III: IDENTIFICATION DES SYSTEMES NON-LINEAIRES PAR LES RESEAUX DE NEURONES D'ORDRE ELEVE

CHAPITRE IV : IDENTIFICATION D'UNE RUCHE APICOLE PAR LES RESEAUX DE NEURONES D'ORDRE ELEVE

CONCLUSION GENERALE

INTRODUCTION GENERALE

Un réseau de neurones d'ordre un est une approche inspirée du système nerveux humain, les problèmes que peut traiter efficacement ce réseau sont aujourd'hui mieux définis : vision, reconnaissance de la parole et des formes, robotique... .

Toutes ces tâches qui paraissent si simples à l'esprit humain sont en fait très complexes pour les traiter par un ordinateur.

Le développement des réseaux de neurones est le souci de tous les chercheurs. C'est une tâche pluridisciplinaire qui demande des connaissances dans différents domaines :

- *Des mathématiciens et physiciens apportent leurs outils pour modéliser le système nerveux.*
- *Des chercheurs issus de milieu biologique, neurophysiologique ou médical fournissent leurs connaissances du système nerveux aux ingénieurs, informaticiens, électroniciens,....*

L'étude sur les réseaux de neurones réunit aussi des chimistes, des psychologues et spécialistes des neurosciences.

Actuellement, la plupart des recherches sont effectuée sur des logiciels de simulations de réseaux de neurones ce qui permet d'expérimenter un grand nombre de lois de fonctionnement de ces réseaux sans nécessiter de gros investissements matériels. Ces simulations demandent des ordinateurs classiques dopés par des coprocesseurs pour accélérer les temps de calculs.

Dans ce cadre, les réseaux de neurones d'ordre élevé se présentent comme une approche évolué du cerveau humain, en essayant d'améliorer les réseaux d'ordre un pour qu'il soient capable d'apprendre assez rapidement avec le minimum d'exemples tout à fait comme le cerveau humain.

L'utilisation des réseaux de neurones dans le domaine de l'identification des systèmes non-linéaires est aujourd'hui très répandue vu :

- *La puissance des réseaux de neurones d'analyser des non-linéarités dures.*
- *Le nombre minimal d'informations sur le système nécessaire par le réseau pour faire l'apprentissage.*

Les réseaux de neurones multicouches et récurrents d'ordre un et deux et le réseau modulaire d'ordre un sont utilisés dans plusieurs articles ***[14],[15],[18]****..., pour l'identification des systèmes non-linéaires et les résultats obtenus sont très satisfaisants.*

Les problèmes qui restent communs pour tout ces travaux sont :

- *le temps d'apprentissage des réseaux de neurones qui est très grand.*
- *La détermination du taux d'apprentissage.*
- *La détermination pour un problème donné, de la taille exacte des réseaux ,pour un réseau multicouches il est nécessaire, de déterminer le nombre de couches et le nombre de neurones dans chaque couche. pour un réseau modulaire nous devons déterminer le nombre de module, et suivant l'ordre de l'entrée et l'ordre de sortie, le nombre de neurones est fixé. pour un réseau récurrent. il est indispensable de déterminer le nombre de neurones.*

Dans ce travail, nous sommes intéressés par l'identification des systèmes non-linéaires par les réseaux de neurones multicouches, modulaires et récurrents d'ordre un et deux (jusqu'à l'ordre trois pour les réseaux multicouches). Les résultats obtenus seront utilisés pour identifier un système réel qui est dans notre cas, une ruche apicole. Pour cela trois types des réseaux seront étudiés :

- *Les réseaux multicouches : chaque réseau de ce type contient une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées.*
- *Les réseaux modulaires : chaque réseau modulaire est constitué de k modules (réseau experts) et un réseau de coordination pour gérer les sorties des modules et la sortie du réseau.*
- *Les rèseaux récurrents : chaque réseau récurrent contient une seule couche mais avec retour de sortie, qui offre la possibilité de résoudre le problème du comportement dynamique des systèmes physiques.*

Dans le cadre de l'identification, quatre modèles sont introduits pour la représentation des systèmes non-linéaires monovariables avec la possibilité de généraliser aux systèmes multivariables

- *Modèle I : Dans ce cas, la sortie du système inconnu et non-linéaire est supposée linéairement dépendante de ses valeurs précédentes et non-linéairement des anciennes valeurs de l'entrée. Pour ce modèle, il est possible de discuter la stabilité en régime libre.*
- *Modèle II : Dans ce modèle, la sortie du système non-linéaire est supposé linéairement dépendante de l'entrée et de ses valeurs précédentes et non linéairement de ses valeurs précédentes.*
- *Modèle III : Dans ce cas, la sortie du système inconnu et non-linéairement dépendante à la fois de ses valeurs anciennes et des valeurs passées de l'entrée.*
- *Modèle IV : Ce modèle est le plus général puisqu'il englobe les précédents modèles et la sortie à n'importe quel instant est une fonction non linéaire des valeurs passées de l'entrée et de la sortie.*

Ce travail comporte quatre chapitres :

Le premier chapitre contient des notions de base sur les réseaux de neurones artificiels et la présentation des deux classes des réseaux de neurones (les réseaux statiques et les réseaux dynamiques) et une étude détaillée des réseaux statiques multicouches et modulaires et le réseau dynamique récurrent.

Le deuxième chapitre concerne les réseaux d'ordre élevé contient une étude sur les réseaux multicouches et récurrent d'ordre élevé en essayant de présenter les différences entre ces réseaux et les réseaux d'ordre un (la définition et les algorithmes d'apprentissage pour chaque type de réseau).

Dans ce chapitre, nous concevons un réseau modulaire d'ordre élevé qui peut être la base pour d'autres représentations plus, et nous développons pour ce réseau un algorithme d'apprentissage qui repose sur le même principe que l'algorithme du réseau d'ordre un.

Le troisième chapitre contient deux parties : La première permet d'aborder des notions sur les modèles, les systèmes et l'identification des systèmes par les réseaux de neurones. La deuxième partie présente les résultats des simulations des systèmes monovariables et multivariables.

Le quatrième chapitre montre les résultats d'identification d'un système physique qui est la ruche apicole. Après avoir discuter les résultats obtenus dans les travaux précédents ***[29],[30]****.*

La conclusion générale présente les résultats obtenus, les problèmes rencontrés et les perspectives de ce travail dans le futur.

Chapitre 01

LES RESEUAX DE NEURONES D'ORDRE UN

I.1 INTRODUCTION :

Les études récentes sur le cerveau ont montré que les cellules nerveuses ''neurones'' sont des cellules relativement simples, et les performances dont se montre capable le système nerveux étant dues au comportement d'un très grand nombre de ces neurones connectés entre eux. Les chercheurs relèvent le défi de construire des circuits électroniques imitant la capacité exceptionnelle dont font preuve les neurones., les neurones artificiels sont des éléments de base, qui vont être assemblés entre eux de manière à construire un système de grande taille capable de résoudre de nombreuses tâches. Un tel système est dit ***réseau de neurones****. Un réseau de neurones fonctionne sans programme, n'exécute pas d'instructions, ne manipule pas de nombres, et ne contient pas de mémoire pour y stocker des données. La destruction d'une partie de ces circuits n'empêche pas le réseau de fonctionner.*

Durant la phase de recherche et de développement, les chercheurs ne font pas appel à des véritables neurones électroniques, mais les simulent par des programmes sur ordinateur conventionnel.

Notre but est d'implémenter des algorithmes représentant les réseaux de neurones d'ordre élevé, il s'avère nécessaire de définir et d'étudier, au préalable, les réseaux d'ordre un.

Dans ce chapitre des notions de base sur les réseaux de neurones seront présentées, puis les réseaux de neurones d'ordre un statiques (multicouches et modulaires) et dynamiques (récurrents) seront développés.

I.2 GENERALITEES :

I.2.1 LE NEURONE ARTIFICIEL :

L'idée de la machine neuronale apparaît pour la première fois en 1943, proposée par **MC CULLOCH** et **PITTS.** L'unité de base du système est le neurone formel ou l'automate, caractérisé par son état binaire (actif ou inactif), et par les connexions qui le relient à d'autres unités, dont les états peuvent évoluer au cours du temps en fonction de leurs interactions mutuelles. **MC CULLOCH** et **PITTS** établissent alors qu'il est possible de représenter toute fonction calculable à l'aide de réseau de neurones binaires, et construisent une machine capable de reconnaître des formes déduites les unes des autres par des transformations simples. Le réseau ainsi proposé est simple : chaque neurone possède des connexions les reliant aux autres neurones. Les synapses qui effectuent les connexions entre neurones peuvent être excitatrices ou inhibitrices. Périodiquement, le neurone calcule son degré d'activation. Si celui-ci dépasse un certain seuil, le neurone se met à l'état actif sinon il se met à l'état inactif figure (1.1).

Le neurone formel peut être donc représenté comme suit :

- Les entrées $[x_1, x_2 \dots x_n]$
- Le seuil θ_i
- La fonction d'activation :

$$f(x) = \begin{cases} -1 & si \quad x \le 0 \\ +1 & si \quad x > 0 \end{cases}$$

- La sortie du neurone est :

$$y_i = \begin{cases} -1 & si \quad u_i(w,x) \le 0 \\ +1 & si \quad u_i(w,x) > 0 \end{cases}$$

- L'état du neurone :

$$u_i(w,x) = \sum_{j=1}^{n} w_{i,j} x_j - \theta_i$$

w_{ij} sont les poids synaptiques

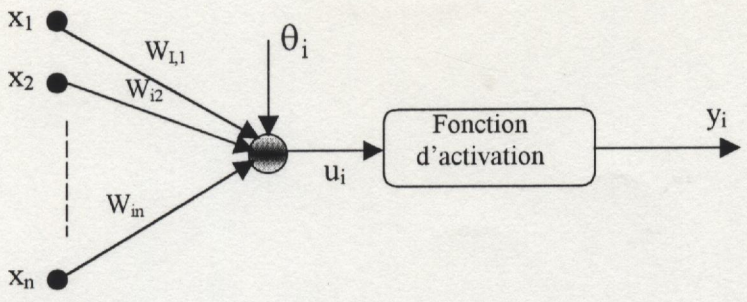


Figure (1.1) : *Modèle de base d'un neurone artificiel*

Un neurone artificiel est donc caractérisé par :

- Sa fonction d'activation.
- Sa fonction de base ou son état.

I.2.1.1. LA FONCTION DE BASE :

Les connexions d'un réseau de neurones sont représentées analytiquement par la fonction $u_i(w,x)$ avec :

w : étant la matrice des poids synaptiques.

x : étant le vecteur des entrées.

$u_i(w,x)$ qui décrit l'état du réseau possède généralement deux formes :

1. La fonction **L.B.F** (Linear Basis Function) qui représente une combinaison linéaire des entrées.

$$u(w,x)=\sum_{j=1}^{n} w_{j,i}x_j$$

2. La fonction **N.L.B.F** (No-Linear Basis Function) qui représente une combinaison non-linéaire des entrées et peut prendre plusieurs formes ; La plus utilisée est la fonction **R.B.F** (Radial basis function) :

$$u_i(w,x)=\left(\sum_{j=1}^{n}\left(x_j-w_{j,i}\right)^2\right)^{1/2}$$

I.2.1.2. LA FONCTION D'ACTIVATION :

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

La fonction sigmoïde :

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

La fonction gaussienne :

$$f(u_i) = ce^{\frac{u_i^2}{\sigma^2}}$$

I.2.2. LES RESEAUX DE NEURONES ARTIFICIELS :

Un réseau de neurones artificiels est un ensemble de neurones interconnectés entre eux. L'architecture du réseau est entièrement spécifiée par :

- Le nombre de cellules(une cellule étant un neurone d'entrée, caché ou de sortie).
- La nature des cellules (La fonction d'activation qui est généralement la même pour toutes les cellules).
- Le graphe d'interconnexion des cellules.
- Les relations entre le réseau et l'extérieur.

La relation entre le réseau et l'extérieur permet de définir dans le réseau trois types de cellules :

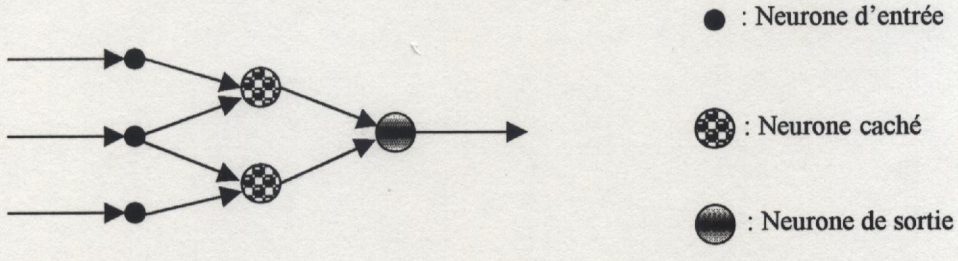


Figure (1.2) : *types de cellules dans un réseau de neurones.*

a) Les cellules d'entrées : sont des cellules qui disposent d'une connexion en entrée reliant le réseau à l'extérieur et permettent d'agir sur son fonctionnement.

b) Les cellules de sortie :possèdent une connexion vers l'extérieur du réseau. Elles indiquent la réponse du réseau.

c) Les cellules cachées : ont des entrées qui ne proviennent que des cellules du réseau et ont une sortie non consultable.

Ces paramètres sont en général fixés à priori, certains peuvent cependant être modifiés au cours de la vie du réseau (exp. : nombre de cellules).

I.2.3. TRAITEMENT DE L'INFORMATION PAR LES RESAUX DE NEURONES :

Dans le traitement de l'information par les réseaux de neurones, il existe deux phases :

La phase d'apprentissage et la phase de reconnaissance.

I.2.3.1.LA PHASE D'APPRENTISSAGE :

C'est une phase de développement du réseau pendant laquelle il détermine les poids des connexions afin de réaliser la tache souhaitée. La puissance de l'apprentissage relie le temps d'exécution d'une tâche à la durée pendant laquelle on s'y est entraîné. Il énonce que la performance est égale à une puissance de la durée de l'entraînement il existe trois grandes classes d'apprentissage.

a) APPRENTISSAGE SUPERVISE :

L'apprentissage supervisé figure (1.3) est basé sur des exemples d'apprentissage de type entrées/sorties (x_i, y_i) le problème consiste à construire un réseau de neurones tel que :

$$R_N(x_i) = y_i \quad \forall i$$

R_N : équation de sortie d'un réseau de neurones.

Pour apprendre le réseau doit savoir qu'il a commis une erreur et doit connaître la réponse qu'il aurait dû donner. La règle d'apprentissage est locale dans le sens que chaque cellule de sortie apprend sans avoir besoin de connaître la réponse des autres cellules.

La cellule ne modifie l'intensité de ses synapses (apprend) que lorsqu'elle se trompe.

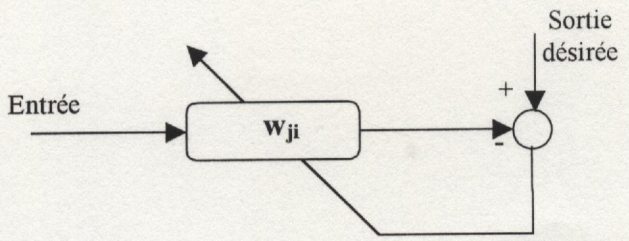


Figure (1.3) *:apprentissage supervisé*

b) APPRENTISSAGE SEMI SUPERVISE :

Il est basé sur des exemples d'apprentissage de la forme (entrée), la seule information disponible en sortie du réseau de neurones est un signal d'échec ou un signal d'erreur.

c) APPRENTISSAGE NON SUPERVISE :

Il est basé sur des exemples d 'apprentissage de type (entrées) figure (1.4). Le réseau organise ses entrées de telle sorte à optimiser un critère de performance donné.

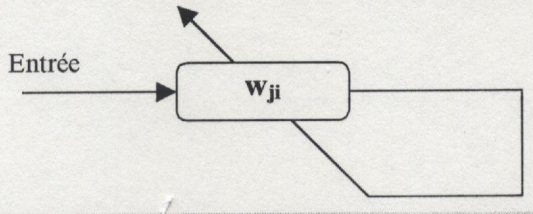


Figure (1.4) *:apprentissage non supervisé*

I.2.3.2. LA PHASE DE RECONNAISSANCE:

C'est la phase utilisation du réseau pour laquelle il est destiné, et c'est à l'aide d'un ensemble de test que les performances du réseau seront testées.

I.2.4. ARCHITECTURE DES RESEAUX DE NEURONES :

Les réseaux de neurones sont divisés en deux grandes classes :

- Réseaux de neurones statiques.
- Réseaux de neurones dynamiques.

I.2.4.1 LES RESEAUX DE NEURONES STATIQUES :

Dans le cas des réseaux statiques, la sortie actuelle d'un neurone n'a aucune influence sur les sorties futures des autres neurones ; La sortie actuelle d'un neurone n'est injectée ni directement ni indirectement à son entrée.

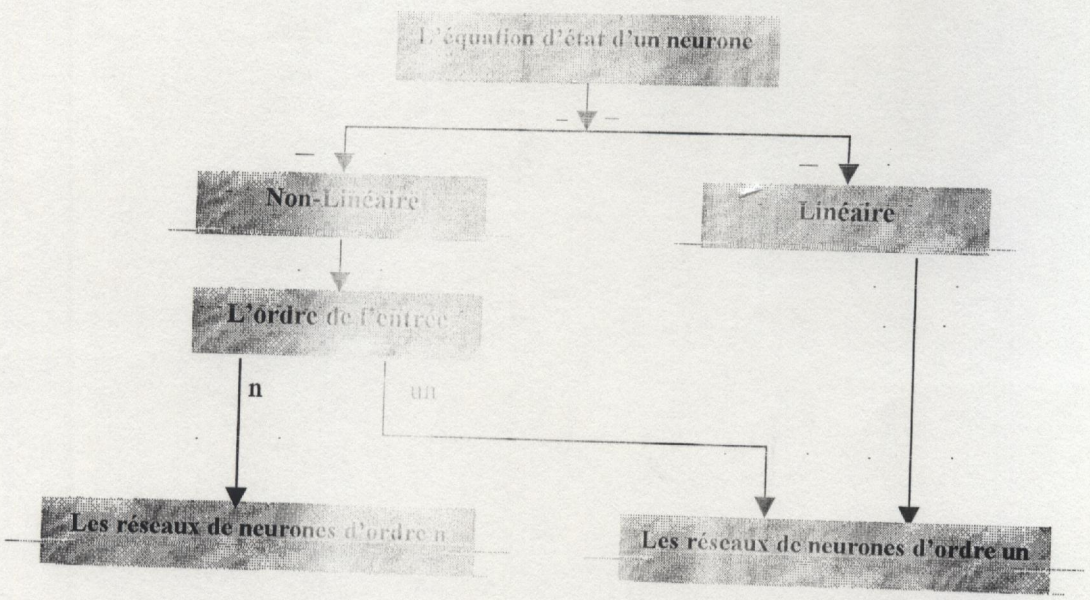
I.2.4.2 LES RESEAUX DE NEURONES DYNAMIQUES:

Dans les réseaux dynamiques, appelés aussi réseaux récurrents, l'influence entre les neurones s'exerce dans les deux sens.

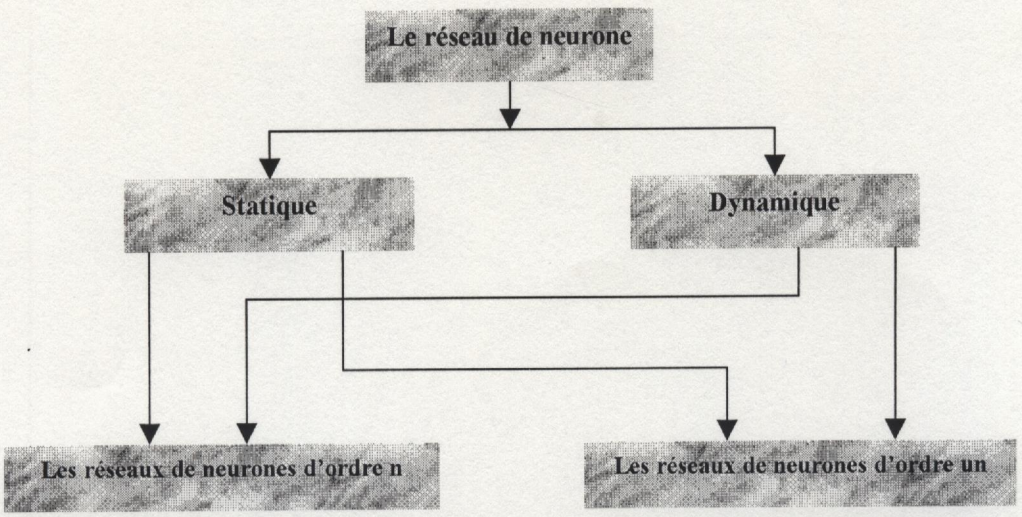
L'état global du réseau dépend de ses états précédents c'est à dire que chaque neurone reçoit une partie, ou la totalité, des sorties précédentes des autres neurones.

La classification des réseaux en réseaux dynamiques et statiques n'est pas unique, les réseaux de neurones peuvent être aussi divisés en classes suivant l'ordre de l'entrée, ils existent, les réseaux de neurones dynamiques ou statiques d'ordre un, deux ...etc figure (1.5) et (1.6).

Les réseaux dont l'ordre de l'entrée est supérieur ou égale à 2 sont appelés les réseaux de neurones d'ordre élevé.



Figure(1.5) : *Classification des réseaux de neurones suivant l'ordre de l'entrée*



Figure(1.6) : *Classification des réseaux de neurones suivant le type de réseau*

I.3.DEFINITION DES RESEAUX D'ORDRE UN :

Un réseau de neurones d'ordre un est défini comme étant un réseau dont les entrées sont d'ordre un, c'est à dire que dans les équations des sorties ou des états des neurones les multiplications entre les entrées, ou les sorties des neurones n'existent pas.

I.4.LES RESEAUX DE NEURONES STATIQUES D'ORDRE UN:

I.4.1.LES RESEAUX MULTICOUCHES MLP 'MULTILAYER PERCEPTRON'

Un réseau multicouche contient une couche d'entrées, une couche de sorties et une ou plusieurs couche(s) cachée(s) figure (1.7). Chaque couche contient un nombre défini de neurones, et chaque neurone de la couche l est connecté à tous les neurones de la couche l+1.

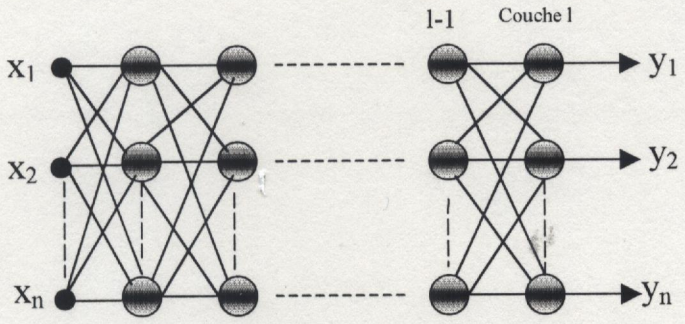


Figure (1.7) : *Réseau de neurone multicouche*

Pour les réseaux multicouches d'ordre n on respecte les notations suivantes :

$U_{l,j}$	La sortie du neurone j de la couche l
$U_{0,i}$	La i^{eme} entrée
$d_j(x_p)$	La sortie désirée de la j^{eme} sortie
N_l	Nombre de neurones de la couche l
L	Nombre de couches
P	Nombre de données d'apprentissage
$W_{l,j,i}$	Le poids entre le i^{eme} neurone de la couche l-1 et le j^{eme} neurone de la couche l
$W_{l,j,i,k}$	Le poids entre le produit du i^{eme} et k^{eme} neurone de la couche l-1 et le j^{eme} neurone de la couche l
$W_{l,j,j1,j2,\ldots,jn}$	Le poids entre le produit des $j_1, j_2 \ldots j_n$ neurones de la couche l-1 et le j^{eme} neurone de la couche l

I.4.1.1.L'algorithme d'apprentissage des réseaux multicouches MLP :

La sortie d'un neurone de la couche l est donnée par :

$$u_{l,j} = f\left(\sum_{i=0}^{N_{l-1}} w_{l,j,i} u_{l-1,i}\right) \tag{1.1}$$

où $f(\bullet)$ est une fonction sigmoidale dont la dérivée est telle que :

$$f'(\alpha) = \frac{df(\alpha)}{d\alpha} = f(\alpha)(1 - f(\alpha)) \tag{1.2}$$

L'algorithme d'apprentissage utilisé pour les réseaux MLP est basé sur la méthode du gradient dont le but est de trouver les poids synaptiques qui minimisent une fonction de critère donnée.

La fonction de critère à minimiser dans notre cas est :

$$J(w) = \sum_{p=1}^{p} j_p(w) \tag{1.3}$$

P est le nombre de données d'apprentissage,

et

$$j_p(w) = \frac{1}{2}\sum_{q=1}^{N_l}\left(u_{l,q}(x_p) - d_q(x_p)\right)^2 \tag{1.4}$$

Les poids seront ajustés de la façon suivante :

$$\begin{aligned} w_{l,j,i}(k+1) &= w_{l,j,i}(k) - \mu \left.\frac{\partial j(w)}{\partial w_{l,j,i}}\right|_{w(k)} \\ &= w_{l,j,i}(k) - \mu \sum_{p=1}^{p} \left.\frac{\partial j_p(w)}{\partial w_{l,j,i}}\right|_{w(k)} \end{aligned} \tag{1.5}$$

où μ est le taux d'apprentissage.

Pour construire l'algorithme, il faut développer l'expression de la dérivée partielle de j_p en respectant chaque poids dans le réseau.

Alors

$$\frac{\partial j_p(w)}{\partial w_{l,j,i}} = \frac{\partial j_p(w)}{\partial u_{l,j}} \frac{\partial u_{l,j}(w)}{\partial w_{l,j,i}} \tag{1.6}$$

$$\begin{aligned} \frac{\partial u_{l,j}}{\partial w_{l,j,i}} &= \frac{\partial}{\partial w_{l,j,i}} \left[f\left(\sum_{m=0}^{N_{l-1}} w_{l,j,m} u_{l-1,m} \right) \right] \\ &= f'\left(\sum_{m=0}^{N_{l-1}} w_{l,j,m} u_{l-1,m} \right) \frac{\partial}{\partial w_{l,j,i}} \left(\sum_{m=0}^{N_{l-1}} w_{l,j,m} u_{l-1,m} \right) \\ &= f'\left(\sum_{m=0}^{N_{l-1}} w_{l,j,m} u_{l-1,m} \right) u_{l-1,i} \end{aligned} \tag{1.7}$$

En remplaçant la valeur de f ' dans l'équation (1.2) on trouve

$$\frac{\partial u_{l,j}(w)}{\partial w_{l,j,i}} = u_{l,j}\left(1 - u_{l,j}\right) u_{l-1,i} \tag{1.8}$$

Donc l'équation (1.6) sera :

$$\frac{\partial j_p(w)}{\partial w_{l,j,i}} = \frac{\partial j_p(w)}{\partial u_{l,j}} u_{l,j}\left(1 - u_{l,j}\right) u_{l-1,i} \tag{1.9}$$

Le terme $\dfrac{\partial j_p(w)}{\partial u_{l,j}}$ représente la sensibilité de $j_p(w)$ par rapport à la sortie de neurone $u_{l,j}$

Le terme $u_{l,j}$ exerce son influence sur j_p sur tous les neurones des couches suivantes.

$\frac{\partial j_p(w)}{\partial u_{l,j}}$ est une fonction de sensibilité du terme $j_p(w)$ à la sortie du neurone $u_{l,j}$ et peut être représentée en fonction des sensibilités par rrapport aux sorties des neurones de la couche suivante.

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} \frac{\partial u_{l+1,m}}{\partial u_{l,j}} \tag{1.10}$$

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

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

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

On continue de la même façon jusqu'à la couche de sortie ; on trouve pour la couche de sortie :

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = u_{l,j}(x_p) - d_j(x_p) \tag{1.11}$$

Résumé de l'algorithme d'apprentissage du MLP :

1. Initialisation des poids synaptiques avec des petites valeurs aléatoires entre 0 et 1.
2. Présentation des données d'apprentissage (entrées et sorties désirées)
3. Calcul de la sortie de neurone *equation* (1.1).
4. Calcul du gradient :

   Equation (1.12) pour la couche de sortie.

   Equation (1.10) ou (1.11) pour la (les) couche(s) cachée(s).
5. Ajustement des poids é*quation* (1.5).
6. Répétition des étapes (2) jusqu'à (5) pour toutes les données et ce, jusqu'à convergence de l'algorithme.

I.4.1.2. RECOMMANDATIONS POUR L'UTILISATION DE L'ALGORITHME DU M.L.P :

a)* Initialisation des poids synaptiques *:

Une étude faite par **Y.K. Kim et J.B. Ra [10]** a montré que l'apprentissage peut être plus rapide si on respecte le critère suivant pour initialiser les poids synaptiques:

$$\sqrt{\frac{\mu}{N}} < |w_i(0)| \quad pour \ \ i = 0 \ldots N$$

N : Nombre de neurones dans une seule couche.

μ: le taux d'apprentissage.

b) Choix du taux d'apprentissage :

Plusieurs méthodes existent pour déterminer un taux d'apprentissage convenable d'un réseau bien spécifique, cependant il reste difficile de déterminer le meilleur taux d'apprentissage. La méthode la plus simple consiste à ajouter un terme de la forme $\mu\left(w_{ji}(k) - w_{ji}(k-1)\right)$ à chaque ajustement de poids avec $0 < \mu < 1$. Une autre méthode consiste à diminuer le taux d'apprentissage si l'algorithme diverge de façon que le taux reste toujours entre 0 et 1, mais le temps d'apprentissage sera très grand dans ce cas, à cause des tests effectués.

c) La taille du réseau :

est une tache difficile car l'efficacité des méthodes n'est pas bonne, Une des méthodes pour la détermination de la taille d'un réseau consiste à démarrer avec un réseau de petit taille puis à ajouter des neurones jusqu'obtention de bonnes performances ; Une autre méthode consiste à démarrer avec un réseau large puis éliminer les neurones.

d) Le test d'arrêt :

Il y a des méthodes pour l'arrêt de l'algorithme. La première consiste à tester l'amplitude du gradient, car par définition, il sera nul lorsqu'on a un minimum. La seconde consiste à comparer un critère de performance avec un certain seuil ; et lorsqu'il sera inférieur on arrête l'apprentissage. La troisième est la méthode de "cross validation" qui repose sur le principe de test des performances de généralisation lors de l'apprentissage. Mais cette méthode demande plus de calcul par rapport aux autres.

I.4.2 LE RESEAU DE NEURONES MODULAIRE :

Le réseau modulaire figure (1.8) est constitué de k modules supervisés nommés modules experts fonctionnant de façon indépendante et d'une unité intégrée nommée ''réseau de coordination'' «Gating Network» dont le rôle est de gérer les sorties des réseaux experts ainsi que leurs tâches.

Le principe est donc de décomposer une tache en deux ou plusieurs sous-taches simples, puis chaque module fait l'apprentissage d'une sous-tache.

Le réseau de coordination gère les réseaux experts pour avoir la sortie finale du réseau.

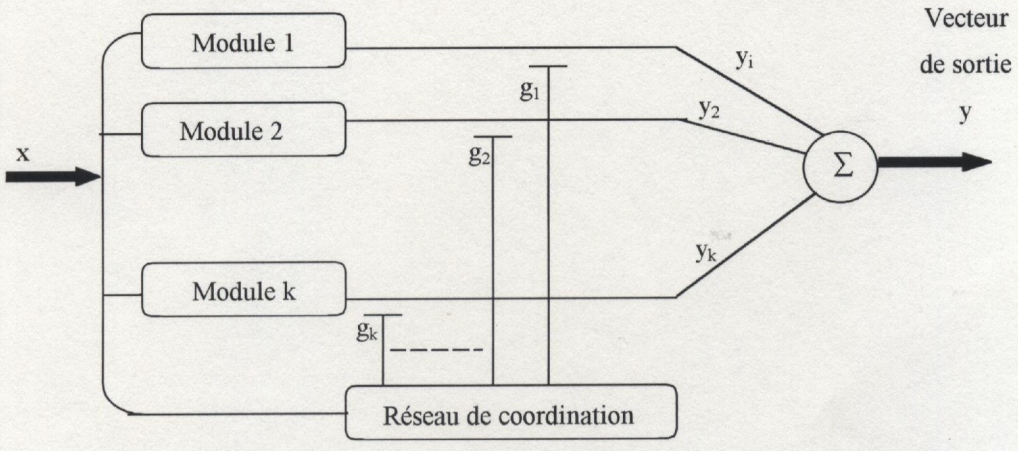


Figure (1.8) : *Réseau modulaire*

I.4.2.1.L'algorithme d'apprentissage du réseau modulaire :

Pour le réseau modulaire on respecte les notations suivantes :

X	Le vecteur des entrées d'une dimension p
d	Le vecteur des sorties du i^{eme} réseau expert d'une dimension qx1
g_I	L'activation de la i^{eme} sortie de neurone du réseau de coordination
Y	Le vecteur des sorties du réseau modulaire

Dans un réseau modulaire l'apprentissage des réseaux experts et le réseau de coordination se font simultanément ; La sortie du réseau est donnée par :

$$y = \sum_{i=1}^{k} g_i y_i \qquad (1.12)$$

L'objectif de l'algorithme d'apprentissage est de déterminer une distribution de probabilité de l'ensemble {x, d} en respectant les étapes suivantes :

- Choisir un vecteur d'entrées x.
- Choisir un module sous forme de probabilité p(i/x) d'avoir le module i connaissant le vecteur d'entrée x.
- Générer une sortie désirée d'un module :

$$d = F_i(x) + \varepsilon_i \qquad i = 1,2,\cdots\cdots k \qquad (1.13)$$

$F_i(x)$ est une fonction déterministe pour une entrée x.

ε_i est un vecteur aléatoire, pour des raisons de simplifications, on suppose que ε_i suit une loi normale avec une moyenne nulle et une matrice de covariance identité.

On suppose aussi que :

$$\varepsilon_1 = \varepsilon_2 = \cdots = \varepsilon_k \qquad (1.14)$$

Soit Λ_i la matrice de covariance de ε_i alors

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

La sortie du module est représentée par la moyenne conditionnelle de la réponse désirée connaissant l'entrée et le module.

donc :

$$y_i = \mu_i \qquad (1.15)$$

$$\mu_i = E\left(d/_{x,i}\right) = F_i(x) \qquad (1.16)$$

E est l'espérance mathématique.

La distribution gaussienne multivariable de la sortie désirée connaissant l'entrée x et le i^{eme} réseau expert est :

$$\begin{aligned} f\left(d/_{x,i}\right) &= \frac{1}{(2\pi\, det\, \Lambda_i)^{q/2}} exp\left(-\frac{1}{2}(d-y_i)^T \Lambda_i^{-1}(d-y_i)\right) \\ &= \frac{1}{(2\pi)^{q/2}} exp\left(-\frac{1}{2}(d-y_i)^T(d-y_i)\right) \qquad (1.17) \\ &= \frac{1}{(2\pi)^{q/2}} exp\left(-\frac{1}{2}\|d-y_i\|^2\right) \qquad i = 1,2,\cdots k \end{aligned}$$

En utilisant les équations (1.17) et (1.12), la sortie désirée d sachant que l'entrée x est :

$$f\left(d/x\right)=\sum_{i=1}^{k} g_i f\left(d/x,i\right) = \sum_{i=1}^{k} g_i \exp\left(-\frac{1}{2}\|d-y_i\|^2\right) \qquad (1.18)$$

à cette étape le problème consiste à déterminer les poids synaptiques *w* d'un module pour maximiser la densité de probabilité *f*(d/x).

On considère *f*(d/x) comme une fonction de vraisemblance avec les poids *w*, les activations jouent le rôle des paramètres inconnus à identifier.

$$w=[w_1,w_2,\cdots w_n]^T \quad et \quad g=[g_1,g_2,\cdots g_n]^T$$

Dans notre cas, il préférable d'utiliser le logarithme naturel de *f*(d/x) au lieu de *f*(d/x).

On définit :

$$L(w,g)=ln\, f\left(d/x\right) \qquad (1.19)$$

avec les équations (1.8) (1.7) on trouve :

$$L(w,g)=ln\left(g_i \exp\left(-\frac{1}{2}\|d-y_i\|^2\right)\right) \qquad (1.20)$$

Le terme $ln(2\pi)^{q/2}$ est constant négligé.

Les sorties du réseau de coordination sont considérées comme étant des probabilités conditionnelles pour le choix du module suivant l'ensemble des entrées sorties données.

Les activations du réseau de coordination doivent satisfaire les conditions suivantes :

$$0 \le g_i \le 1 \qquad (1.21)$$

$$\sum_{i=1}^{k} g_i = 1 \qquad (1.22)$$

suivant ces deux conditions on définit g_i comme suit :

$$g_i = \frac{exp(u_i)}{\sum_{i=1}^{k} exp(u_i)} \qquad (1.23)$$

où u_i est la somme pondérée des entrées du i^{eme} neurone du réseau de coordination.

Nous définissons une probabilité postérieure associée à la sortie du i^eme réseau expert comme suit **[32]**:

$$h_i = \frac{g_i \, exp\left(-\frac{1}{2}\|d - y_i\|^2\right)}{\sum_{j=1}^{k} g_j \, exp\left(-\frac{1}{2}\|d - y_j\|^2\right)} \qquad i = 1,2,\cdots k \qquad (1.24)$$

En notant que h_i doit satisfaire (1.21) (1.22).

D'autre part on peut écrire :

$$0 \le h_i \le 1$$

$$\sum_{i=1}^{k} h_i = 1$$

Il faut noter qu'il est nécessaire, d'étudier les modifications des poids pour les différents réseaux experts ainsi que le réseau de coordination.

a) Apprentissage des réseaux experts :

Un réseau expert peut être représenté par la figure suivante :

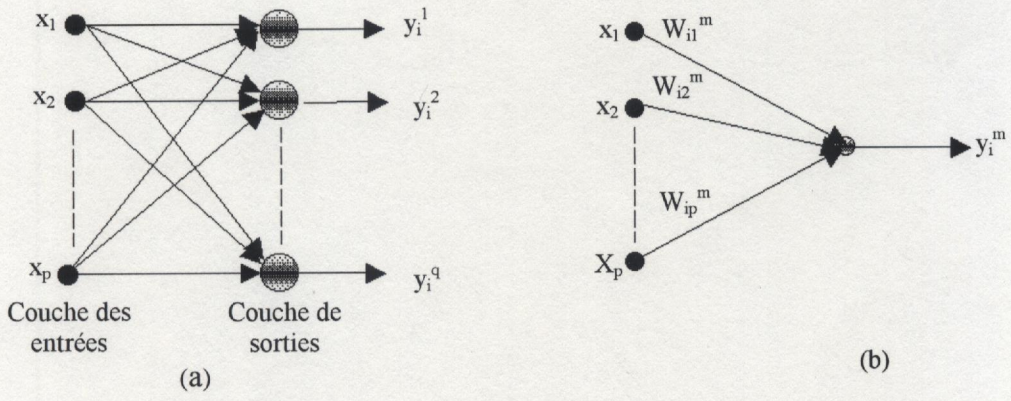


Figure (1.9) :

a) la couche des neurones d'un réseau expert.

b) graphe d'influence d'un neurone dans un réseau expert.

La sortie du neurone m est donnée par :

$$y_i^m = x^T w_i^m \tag{1.25}$$

Pour maximiser la fonction de vraisemblance, les poids seront ajustés par la méthode du gradient de la façon suivante :

$$w_i^m(k+1) = w_i^m(k) + \Delta w_i^m(k) \tag{1.26}$$

où

$$\Delta w_i^m(k) = \mu \frac{\partial L}{\partial w_i^m} \qquad \begin{cases} i = 1,2,\cdots\cdots k \\ m = 1,2,\cdots\cdots q \end{cases} \tag{1.27}$$

μ : est le taux d'apprentissage

concernant le terme $\frac{\partial L}{\partial w_i^m}$ on a :

$$\frac{\partial L}{\partial w_i^m} = \frac{\partial L}{\partial y_i^m}\frac{\partial y_i^m}{\partial w_i^m} \tag{1.28}$$

par simple calcul nous obtenons

$$\begin{aligned} \frac{\partial L}{\partial y_i^m} &= h_i\left(d^m - y_i^m\right) \\ &= h_i e_i^m \end{aligned} \tag{1.29}$$

de l'équation (1.25) on trouve :

$$\frac{\partial y_i^m}{\partial w_i^m} = x \tag{1.30}$$

donc

$$w_i^m(k+1) = w_i^m(k) + \mu h_i e_i^m(k)x \tag{1.31}$$

b) Apprentissage du réseau de coordination :

Le réseau de coordination peut être représenté par la figure (1.10), et c'est un réseau qui contient une seule couche avec un nombre de neurones égal au nombre de réseaux experts. La fonction d'activation est différente de celle des réseaux experts.

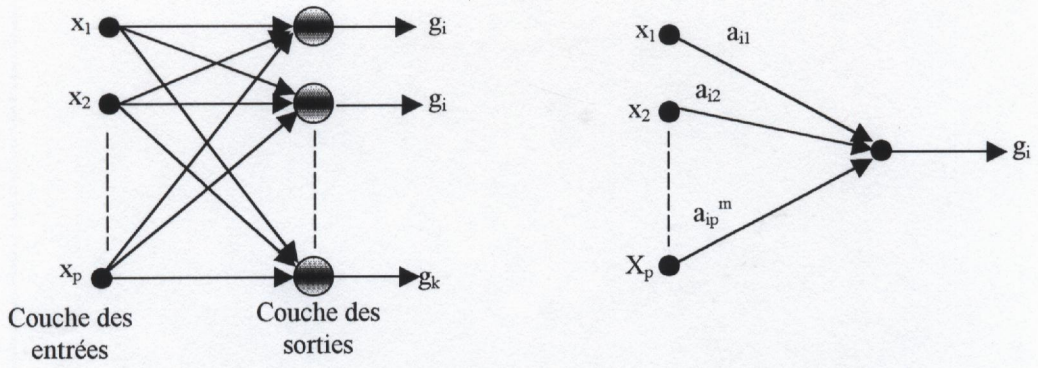


Figure (1.10) :

a) La couche des neurones du réseau de coordination

b) Graphe d'influence d'un neurone dans le réseau de coordination.

D'après les équations (1.20) et (1.23)on trouve :

$$L = ln\left(\sum_{i=1}^{k} exp(u_i)exp\left(-\frac{1}{2}\|d - y_i\|^2\right)\right) - ln\sum_{j=1}^{k} exp(u_j) \tag{1.32}$$

de la figure (2.5) on a :

$$u_i = x^T a_i \tag{1.33}$$

où $a_i = [a_1, a_2, \cdots, a_n]$ est le vecteur des poids du i^{eme} neurone du réseau de coordination ; l'adaptation de ces poids se fait de façon récursive comme suit:

$$a_i(k+1) = a_i(k) + \Delta a_i(k) \tag{1.34}$$

$\Delta a_i(k)$ est défini par :

$$\Delta a_i = \mu \frac{\partial L}{\partial a_i} \tag{1.35}$$

où

$$\frac{\partial l}{\partial a_i} = \frac{\partial l}{\partial u_i}\frac{\partial u_i}{\partial a_i} \tag{1.36}$$

de la même façon que pour les réseaux experts nous trouvons :

$$\frac{\partial l}{\partial u_i} = h_i - g_i \tag{1.37}$$

$$\frac{\partial u_i}{\partial a_i} = x \tag{1.38}$$

$$a_i(k+1) = a_i(k) + \mu(h_i(k) - g_i(k))x \tag{1.39}$$

Résumé de l'algorithme d'apprentissage du réseau modulaire :

1. Initialiser les poids des réseaux experts et du réseau de coordination.
2. Calculer pour i=1,2,.....k et m=1,2,.....q

$$u_i = x^T a_i$$

$$g_i = \frac{\exp(u_i)}{\sum_{i=1}^{k} \exp(u_i)}$$

$$y_i^m = x^T w_i^m$$

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

$$e_i^m = d^m - y_i^m$$

$$w_i^m(k+1) = w_i^m(k) + \mu h_i e_i^m(k)x$$

$$a_i(k+1) = a_i(k) + \mu(h_i(k) - g_i(k))x$$

3. Répéter l'étape 2 pour toutes les données.
4. Répéter les étapes 2 et 3 jusqu'à la convergence de l'algorithme.

I.4.2.2.Avantages du réseau modulaire :

a) La vitesse d'apprentissage:

Décomposer une fonction en deux ou plusieurs sous-fonctions simples rend la tache d'apprentissage plus rapide. Le réseau modulaire, avec son l'architecture interne, peut découvrir cette décomposition **[32]**

Supposons que nous voulons approximer la fonction de la figure (1.11) définit comme suit :

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

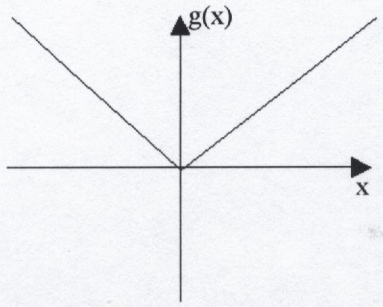


Figure (1.11) : *La fonction g(x)*

Avec un réseau M.LP à la limite on aura besoin d'un réseau avec une seule couche cachée. Par contre avec un réseau modulaire on aura besoin seulement de deux neurones et une unité intégrée (neurones du réseau de coordination) le premier neurone (du réseau expert) apprend $g(x) = x$; pour $x \geq 0$ et l'autre apprend $g(x) = -x$; pour $x < 0$.

Le réseau de coordination gère la sortie du réseau et de décider le neurone correspondant à la valeur de x.

b) La représentation des données :

La représentation des données sera plus simple, puisque la tache complexe sera décomposée en deux ou plusieurs sous-taches simples. Cette propriété est étudiée par **Rueckl .al (1989) [32].**

c) Intégration hardware

Le nombre faible des neurones utilisés dans un réseau modulaire rend l'intégration hardware plus simple**[32]**.

I.5 LES RESEAUX DE NEURONES DYNAMIQUES D'ORDRE UN :

I.5.1 LES RESEAUX RECURRENTS :

un réseau récurrent ou un réseau avec retour de sorties figure (1.12) est constitué d'une seule couche de neurones, chaque sortie de neurone est reliée avec un temps de retard aux entrées de tous les neurones.

certains neurones sont considérés comme étant des sorties du réseau et les autres neurones sont considérés comme des neurones cachés.

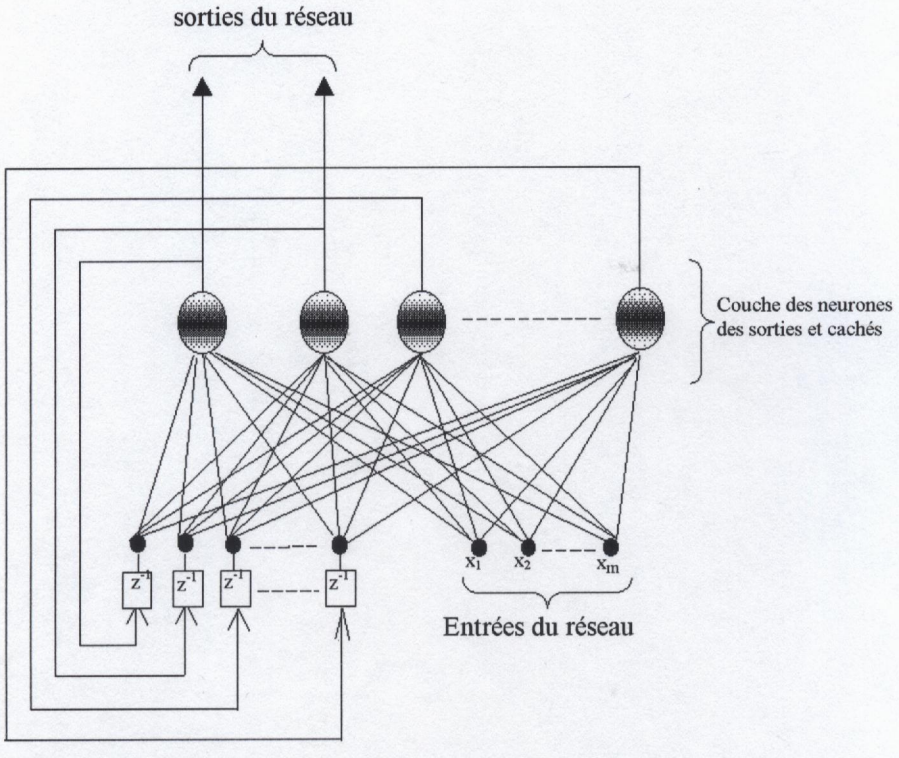


Figure (1.12) : *Réseau récurrent (avec retour de sorties)*

La figure (1.12) montre un réseau récurrent avec :

m entrées, n neurones, 2 sorties.

Avant de développer l'algorithme nous supposons que nous avons un réseau avec m entrées extérieures et n neurones, et nous respectons les notations suivantes :

y(t)	Sorties des neurones à l'instant t
x(t)	Entrées extérieures du réseau à l'instant t
z(t)	x(t) et y(t) concatenés
U	L'ensemble des indices k pour lesquels z_k est une sortie d'un neurone
I	L'ensemble des indices k pour lesquels z_k est une entrée extérieure
T(t)	L'ensemble des indices $k \in U$ pour lesquels ils existent des sorties désirées

Donc nous avons :

$$z_k = \begin{cases} x_k(t) & si\ k \in I \\ y_k(t) & si\ k \in U \end{cases} \tag{1.40}$$

w est la matrice des poids synaptiques de dimension n x (m+n) telle que : il y a un seul poids entre chaque paire de neurones et entre chaque entrée extérieure et un neurone.

Pour que les neurones puissent avoir un seuil, on introduit parmi les m entrées extérieures une entrée égale à 1.

L'entrée du neurone k au moment t est :

$$s_k(t) = \sum_{l \in U \cup I} w_{kl} z_l(t) \qquad pour \quad k \in U \tag{1.41}$$

la sortie du neurone au moment t+1 est :

$$y_k(t+1) = f_k(s_k(t)) \tag{1.42}$$

f_k est la fonction d'activation.

Il faut noter que l'entrée extérieure à l'instant t n'influe pas sur la sortie avant l'instant t+1.

I.5.1.1 L'algorithme d'apprentissage d'un réseau récurrent :

a) L'algorithme de base *:*

L'algorithme que l'on développe est celui de l'apprentissage supervisé temporaire, basé sur le principe qu'à des instants spécifiques certaines sorties des neurones peuvent être égales aux sorties désirées.

La différence entre ces neurones et les sorties désirées est définie par :

$$e_k(t) = \begin{cases} d_k(t) - y_k(t) & si\ k \in T(t) \\ 0 & ailleurs \end{cases} \tag{1.43}$$

l'erreur globale du réseau à l'instant t est :

$$J(t) = \tfrac{1}{2} \sum_{k \in U} [e_k(t)]^2 \tag{1.44}$$

si le réseau effectue l'apprentissage de l'instant t_0 jusqu'à l'instant t_1, la fonction à minimiser sera :

$$J_{total}(t_0, t_1) = \sum_{t=t_0+1}^{t_1} J(t) \tag{1.45}$$

les poids synaptiques seront ajustés de la façon suivante :

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{1.46}$$

$\Delta w_{ij}(t)$ est la somme des variations $\nabla_w j(t)$ de t_0 à t_1.

Donc $$\Delta w_{ij} = \sum_{t=t_0+1}^{t_1} \Delta w_{ij}(t) \tag{1.47}$$

où $$\Delta w_{ij}(t) = -\mu \frac{\partial J(t)}{\partial w_{ij}} \tag{1.48}$$

μ : est le taux d'apprentissage.

De l'équation (1.44) et (1.43) on trouve :

$$\begin{aligned} \frac{\partial J(t)}{\partial w_{ij}} &= \sum_{k \in T(t)} e_k(t) \frac{\partial e_k(t)}{\partial w_{ij}} \\ &= -\sum_{k \in T(t)} e_k(t) \frac{\partial y_k(t)}{\partial w_{ij}} \end{aligned} \tag{1.49}$$

Nous utilisons les équations (1.41) et (1.42) pour déterminer $\frac{\partial y_k(t)}{\partial w_{ij}}$:

$$\begin{aligned} \frac{\partial y_k(t+1)}{\partial w_{ij}} &= \frac{\partial y_k(t+1)}{\partial s_k(t)} \frac{\partial s_k(t)}{\partial w_{ij}} \\ &= f_k'(s_k(t)) \frac{\partial s_k(t)}{\partial w_{ij}} \end{aligned} \tag{1.50}$$

$$\begin{aligned} \frac{\partial s_k(t)}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \left[\sum_{l \in U \cup I} w_{kl} z_l \right] \\ &= \sum_{l \in U \cup I} \left[w_{kl} \frac{\partial z_l(t)}{\partial w_{ij}} + \frac{\partial w_{kl}}{\partial w_{ij}} z_l \right] \end{aligned} \tag{1.51}$$

En notant que $$\frac{\partial w_{kl}}{\partial w_{ij}} = \begin{cases} 1 & si\ j = l\ et\ i = k \\ 0 & ailleurs \end{cases} \tag{1.52}$$

l'équation (1.51) sera :

$$\frac{\partial s_k(t)}{\partial w_{ij}} = \sum_{l \in U \cup I} \left[w_{kl} \frac{\partial z_l(t)}{\partial w_{ij}} + \delta_{ik} z_j \right] \tag{1.53}$$

δ_{ik} est le coefficient de *kronecker*

$$\delta_{ik} = \begin{cases} 1 & si\ k = j \\ 0 & ailleurs \end{cases} \tag{1.54}$$

$$\frac{\partial z_l(t)}{\partial w_{ij}} = \begin{cases} 0 & si\ k \in I \\ \dfrac{\partial y_k(t)}{\partial w_{ij}} & si\ k \in U \end{cases} \tag{1.55}$$

d'après les équations (1.55), (1.53) et (1.50) on trouve :

$$\frac{\partial y_k(t+1)}{\partial w_{ij}} = f_k'(s_k(t))\left[\sum_{l\in U} w_{kl}\frac{\partial y_l(t)}{\partial w_{ij}} + \delta_{ik} z_j(t)\right] \qquad (1.56)$$

Pour tout $k \in U, i \in U, j \in U \cup I$.

L'état initial du réseau au moment t_0 n'ayant aucune relation avec les poids synaptiques on a :

$$\frac{\partial y_k(t_0)}{\partial w_{ij}} = 0 \qquad (1.57)$$

Nous pouvons définir le système dynamique avec un ensemble des variables p_{ij}^k

Pour tout $k \in U, i \in U, j \in U \cup I$.

$$p_{ij}^k(t) = \frac{\partial y_k(t)}{\partial w_{ij}} \qquad (1.58)$$

$$p_{ij}^k(t+1) = f_k'(s_k(t))\left[\sum_{l\in U} w_{kl} p_{ij}^k(t) + \delta_{ik} z_j(t)\right] \qquad (1.59)$$

L'algorithme consiste alors à calculer pour chaque pas d'apprentissage t de t_0 à t_1 p_{ij}^k ;

Ensuite en utilisant $e_k(t)$ on peut calculer $\Delta w_{ij}(t)$:

$$\Delta w_{ij}(t) = \mu \sum_{k\in U} e_k(t) p_{ij}^k(t) \qquad (1.60)$$

b) L'algorithme d'apprentissage en temps réel :

L'algorithme d'apprentissage de base est développé en supposant que les poids restent fixes le long de la trajectoire (de t_0 à t_1).

Si les poids sont ajustés au fur et à mesures durant l'exécution du réseau on dit que l'apprentissage est fait en temps réel.

L'avantage majeur de cet algorithme est qu'il n'est pas nécessaire de déterminer les bornes de la trajectoire pour faire l'apprentissage, ce qui rend l'algorithme plus facile. Dans ce cas on incrémente simplement chaque poids w_{ij} de Δw_{ij} (l'équation (1.60)) ; mais l'inconvénient potentiel de l'apprentissage en temps réel est la divergence de l'algorithme. Pour résoudre ce problème, il faut choisir pour cet algorithme un taux d'apprentissage très petit.

c) Apprentissage en temps réel forcé :

Cette technique importante, utilisée fréquemment dans l'apprentissage supervisé temporaire, **[20]** consiste à remplacer la sortie d'un neurone y_k par sa sortie désirée correspondante.

La dynamique du réseau dans le cas de l'apprentissage forcé est définie par les équations (1.61), (1.41) et (1.42) avec la différence suivante :

$$z_k(t) = \begin{cases} x_k(t) & si \ \ k \in I \\ d_k(t) & si \ \ k \in T(t) \\ y_k(t) & si \ \ k \in U - T(t) \end{cases} \qquad (1.61)$$

dans ce cas on aura :

$$\frac{\partial y_k(t+1)}{\partial w_{ij}} = f_k'(s_k(t)) \left[\sum_{l \in U-T(t)} w_{kl} \frac{\partial y_l(t)}{\partial w_{ij}} + \delta_{ik} z_j(t) \right] \qquad (1.62)$$

$$p_{ij}^k(t+1) = f_k'(s_k(t)) \left[\sum_{l \in U-T(t)} w_{kl} p_{ij}^k(t) + \delta_{ik} z_j(t) \right] \qquad (1.63)$$

les p_{ij}^l sont remis à zéro pour $l \in T(t)$ après chaque calcul de $\Delta w_{ij}(t)$.

Remarque :

L'algorithme que nous avons adopté est celui de l'apprentissage en temps réel

Résumé de l'algorithme d'apprentissage en temps réel :

1. Initialiser les poids synaptiques.
2. Calculer les sorties en utilisant les équations (1.40), (1.41) et (1.42).
3. Calculer les valeurs p_{ij}^k pour $k \in U, i \in U, j \in U \cup I$.
4. Calculer les variations $\Delta w_{ij}(t)$ en utilisant l'équation (1.60).
5. Ajuster les poids en utilisant (1.46).
6. Répéter les étapes 2 jusqu'à 5 et jusqu'à la convergence de l'algorithme.

Le réseau récurrent présenté ci-dessus est appelé un réseau récurrent complet (Full Reccurent Neural Neworks), car tous les connections possibles entre les neurones et les entrées existent. Dans certains cas, comme pour les réseaux d'ordre élevé, le nombre de connections sera réduit. Une étude est faite sur les réseaux de neurones d'ordre élevé dans le chapitre suivant, et montre en détail le cas des réseaux récurrents avec un nombre réduit d'interconnections.

Chapitre 02

LES RESEAUX DE NEURONES D'ORDRE ELEVE

II.1 INTRODUCTION :

Les études sur les réseaux de neurones ne cessent d'évoluer chaque jour et le progrès des algorithmes d'apprentissage (rapidité, erreur, taille ...)est le souci de tous les chercheurs. Dans ce contexte, les réseaux de neurones d'ordre élevé sont apparus dans le but de réduire la taille des réseaux et d'améliorer la vitesse d'apprentissage....

Notre but est d'implémenter des algorithmes d'apprentissage pour les réseaux de neurones d'ordre élevé.

Dans le présent chapitre, nous commençons par définir les réseaux d'ordre élevé, puis nous développons le réseau multicouche (représentation et algorithmes). ensuite nous concevons un réseau modulaire d'ordre élevé (la partie de conception représente notre contribution dans ce travail) ;nous terminons ce chapitre par présenter le réseau récurrent d'ordre élevé avec un nombre réduit d'interconnections et le réseau récurrent complet.

II.2 LES RESEAUX MULTICOUCHES MLP D'ORDRE ELEVE :

Les réseaux de neurones multicouches d'ordre élevé (figure (2.1)) sont caractérisés par la présence de termes non-linéaires dans les équations qui définissent la dynamique du réseau.

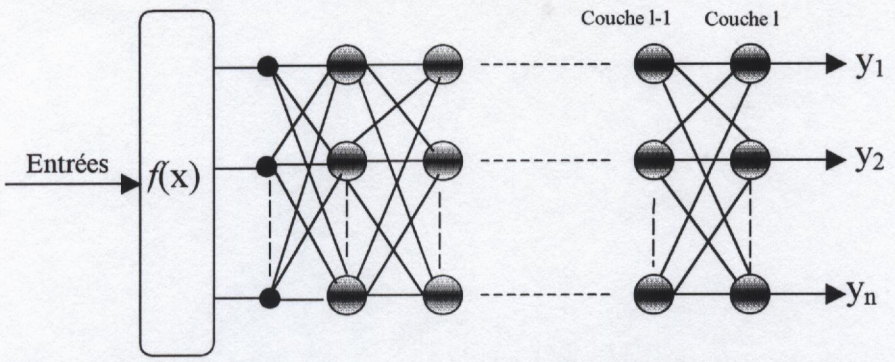


Figure (2.1) : *Réseau de neurones multicouches d'ordre élevé*

f(x) peut être :

- Une fonction linéaire :

$$f(x) = x^T \qquad x^T = [x_1, x_2, \cdots x_n]$$

- Une fonction non-linéaire :

$$f(x) = x_{i1} + x_{i1}x_{i2} + x_{i1}x_{i2}\cdots x_{ir} \quad pour\ i1,i2\cdots ir = 1\cdots p$$

II.2.1 LES RESEAUX D'ORDRE 2 :

Dans ce réseau l'équation de sortie d'un neurone n'est plus linéaire et a la forme suivante :

$$u_{l,j} = f\left(\sum_{i=0}^{N_{l-1}} w_{l,j,i}u_{l-1,i} + \sum_{i=0}^{N_{l-1}}\sum_{k=i}^{N_{l-1}} w_{l,j,i,k}u_{l-1,i}u_{l-1,k}\right) \qquad (2.1)$$

Le terme $\sum_{i=0}^{N_{l-1}}\sum_{k=i}^{N_{l-1}} w_{l,j,i,k}u_{l-1,i}u_{l-1,k}$ dans l'équation (2.1) représente la non-linéarité

L'algorithme d'apprentissage se base sur le principe du gradient et les poids seront ajustés de la façon suivante :

$$w_{l,j,i}(n+1) = w_{l,j,i}(n) - \mu \left.\frac{\partial j(w)}{\partial w_{l,j,i}}\right|_{w(n)} \qquad (2.2)$$

$$w_{l,j,i,k}(n+1) = w_{l,j,i,k}(n) - \mu \left.\frac{\partial j(w)}{\partial w_{l,j,i,k}}\right|_{w(n)} \qquad (2.3)$$

L'algorithme d'apprentissage, pour ce cas contient les mêmes étapes que l'algorithme d'apprentissage du réseau linéaire.

Donc on a :

$$\frac{\partial j_p(w)}{\partial w_{l,j,i}} = \frac{\partial j_p(w)}{\partial u_{l,j}} \frac{\partial u_{l,j}(w)}{\partial w_{l,j,i}} \tag{2.4}$$

$$\frac{\partial j_p(w)}{\partial w_{l,j,i}} = \frac{\partial j_p(w)}{\partial u_{l,j}} u_{l,j}\left(1-u_{l,j}\right)u_{l-1,i} \tag{2.5}$$

où

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} \frac{\partial u_{l+1,m}}{\partial u_{l,j}} \tag{2.6}$$

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

$$= \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} u_{l+1,m}\left(1-u_{l+1,m}\right) \frac{\partial}{\partial u_{l,j}}$$

$$\left[\sum_{q=0}^{N_l} w_{l+1,m,q} u_{l,q} + \sum_{q=0}^{N_l} \sum_{r=q}^{N_l} w_{l+1,m,q,r} u_{l,q} u_{l,r} \right]$$

$$= \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} u_{l+1,m}\left(1-u_{l+1,m}\right)$$

$$\bullet \left[w_{l+1,m,j} + \sum_{q=0}^{N_l} w_{l+1,m,q,j} u_{l,q} + \sum_{r=j}^{N_l} w_{l+1,m,j,r} u_{l,r} \right]$$

pour la couche de sortie on trouve :

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = u_{l,j}(x_p) - d_j(x_p) \tag{2.7}$$

pour développer un algorithme d'apprentissage on doit déterminer le terme $\frac{\partial j(w)}{\partial w_{l,j,i,k}}$

dans l'équation (2.3)

donc

$$\frac{\partial j_p(w)}{\partial w_{l,j,i,k}} = \frac{\partial j_p(w)}{\partial u_{l,j}} \frac{\partial u_{l,j}(w)}{\partial w_{l,j,i,k}} \tag{2.8}$$

$\frac{\partial j_p(w)}{\partial u_{l,j}}$ est donné par l'équation (2.6) ou (2.7) et le terme $\frac{\partial u_{l,j}}{\partial w_{l,j,i,k}}$ est calculé comme suit

$$\frac{\partial u_{l,j}}{\partial w_{l,j,i,k}} = \frac{\partial}{\partial w_{l,j,i,k}} \left[f\left(\sum_{i=0}^{N_{l-1}} w_{l,j,i} u_{l-1,i} \sum_{i=0}^{N_{l-1}} \sum_{k=i}^{N_{l-1}} w_{l,j,i,k} u_{l-1,k} u_{l-1,i} \right) \right]$$

$$\frac{\partial u_{l,j}}{\partial w_{l,j,i,k}} = u_{l,j}\left(1-u_{l,j}\right)u_{l-1,i}u_{l-1,k} \tag{2.9}$$

Résume de l'algorithme d'apprentissage pour MLP d'ordre 2

1. Initialisation des poids synaptiques avec des petites valeurs aléatoires.
2. Présentation des données d'apprentissage (entrées et sorties désirées)
3. Calcul de la sortie de neurone *équation* (2.1).
4. Calcul du gradient
5. Ajustement des poids comme suit :

$$w_{l,j,i} = w_{l,j,i} - \mu g_{l,j,i}$$

$$w_{l,j,i,k} = w_{l,j,i,k} - \mu g_{l,j,i,k}$$

pour la couche de sortie

$$g_{l,j,i} = (u_{l,j} - d_j)u_{l,j}(1-u_{l,j})u_{l-1,i}$$
$$g_{l,j,i,k} = (u_{l,j} - d_j)u_{l,j}(1-u_{l,j})u_{l-1,i}u_{l-1,k}$$

pour la (les) couche (s) cachée (s)

$$g_{l,j,i} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p}{\partial u_{l+1,m}} [S] u_{l+1,m}(1-u_{l+1,m})u_{l,j}(1-u_{l,j})u_{l-1,i}$$

$$S = \left[w_{l+1,m,j} + \sum_{q=0}^{N_l} w_{l+1,m,q,j} u_{l,q} + \sum_{r=j}^{N_l} w_{l+1,m,j,r} u_{l,r} \right]$$

$$g_{l,j,i,k} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p}{\partial u_{l+1,m}} [S] u_{l+1,m}(1-u_{l+1,m})u_{l,j}(1-u_{l,j})u_{l-1,i}u_{l-1,k}$$

6. Répétition des étapes (2) jusqu'à (5) pour toutes les données et ce, jusqu'à convergence de l'algorithme.

II.2.2 LES RESEAUX D'ORDRE n :

La sortie d'un neurone pour un réseau d'ordre n est :

$$u_{l,j} = f(t) = f\Big(\sum_{i=0}^{N_{l-1}} w_{l,j_1,j_2} u_{l-1,j_2} + \sum_{j_2=0}^{N_{l-1}} \sum_{j_3=j_2}^{N_{l-1}} w_{l,j_1,j_2,j_3} u_{l-1,j_2} u_{l-1,j_3} + \cdots + \sum_{j_2=0}^{N_{l-1}} \cdots \sum_{j_n=j_{n-1}}^{N_{l-1}} w_{l,j_1,\cdots,j_n} u_{l-1,j_2} \cdots u_{l-1,j_n} \Big) \tag{2.10}$$

de la même façon que le MLP d'ordre 2 on doit déterminer seulement les termes suivants :

$$w_{l,j_1,\cdots,j_l}(n+1) = w_{l,j_1,\cdots,j_n}(n) - \mu \left. \frac{\partial j(w)}{\partial w_{l,j_1,\cdots,j_l}} \right|_{w(n)} \qquad l = 2 \cdots n \tag{2.11}$$

$$\frac{\partial j_p(w)}{\partial u_{l,j}} \tag{2.12}$$

$$\frac{\partial u_{l,j}}{\partial w_{l,j_1,\cdots,j_n}} \tag{2.13}$$

$$\frac{\partial u_{l,j}}{\partial w_{l,j_1,\cdots,j_n}} = u_{l,j}\left(1 - u_{l,j}\right) u_{l-1,j_2} \cdots u_{l-1,j_n} \tag{2.14}$$

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} \frac{\partial u_{l+1,m}}{\partial u_{l,j}} \tag{2.15}$$

$$\frac{\partial u_{l+1,m}}{\partial u_{l,j}} = \frac{\partial}{\partial u_{l,j}} f(t) \tag{2.16}$$

$$\frac{\partial j_p(w)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial j_p(w)}{\partial u_{l+1,m}} u_{l+1,m}\left(1 - u_{l+1,m}\right) * \left[w_{l+1,j_1,j_2} + \sum_{q=0}^{N_l} w_{l+1,j_1,j_2,j_3} u_{l,j_2} + \sum_{r=j}^{N_l} w_{l+1,j_1,j_2,j_3} u_{l,j_3} \right] \cdots + \left[\sum_{j_3=j_2}^{N_l} \cdots \sum_{j_n=j_{n-1}}^{N_l} w_{l+1,j_1,\cdots,j_n} u_{l,j_2} \cdots u_{l,j_n} \right] \cdots + \left[\sum_{j_2=0}^{N_l} \cdots \sum_{j_{n-1}=j_{n-2}}^{N_l} w_{l+1,j_1,\cdots,j_n} u_{l,j_2} \cdots u_{l,j_{n-1}} \right]$$

II.2.3 LE RESEAU D'ORDRE n AVEC UN NOMBRE REDUIT D'INTERCONNEXIONS :

En général un neurone d'ordre n est défini comme suit :

$$y_i = f\left(w_i^0 + \sum_j^N w_{ij}^{(1)} x_j + \sum_j^N \sum_k^N w_{ijk}^{(2)} x_j x_k + \cdots \right) \tag{2.17}$$

où $w_{ij}^{(1)}$ représente les poids d'ordre 1 et $w_{ijk}^{(2)}$ représente les poids d'ordre 2.

f est une fonction d'activation.

Pour un réseau d'ordre 2 l'équation (2.17) s'écrit :

$$y_i = f\left(w_i^0 + \sum_j^N w_{ij}^{(1)} x_j + \sum_j^N \sum_k^N w_{ijk}^{(2)} x_j x_k \right) \tag{2.18}$$

Dans un réseau d'ordre 2 le nombre total d'interconnexions pour un neurone est n(n+1),un nombre relativement grand. **Yang C. Guest [4]** ont montré qu'on peut réduire le nombre d'interconnexions à n+2 pour rendre un tel réseau plus rapide avec les mêmes performances, en utilisant au lieu de l'équation (2.18) l'équation suivante :

$$y_i = f\left(w_{iN+2}\left(\sum_{j=1}^N w_{ij}^2\right)\left(\sum_{j=1}^N x_j^2\right) - w_{iN+2}\left(\sum_{j=1}^N w_{ij} x_j\right)^2 + \left(\sum_{j=1}^N w_{ij} x_j + w_{N+1}\right) \right) \tag{2.19}$$

Remarque :

La représentation que nous avons adopté est la représentation du réseau avec le nombre complet d'interconnexions

II.3 RESEAU MODULAIRE D'ORDRE n :

Le réseau modulaire d'ordre élevé que nous concevons, en basant sur la définition des réseaux d'ordre élevé et la configuration d'un réseau modulaire est caractérisé par la présence des termes d'ordre 2 ou plus au niveau de l'entrée et l'influence de ces termes s'applique au réseau de coordination, comme il est applique aux réseaux experts (figure (2.2)).

Cette partie originale représente notre contribution dans ce travail.

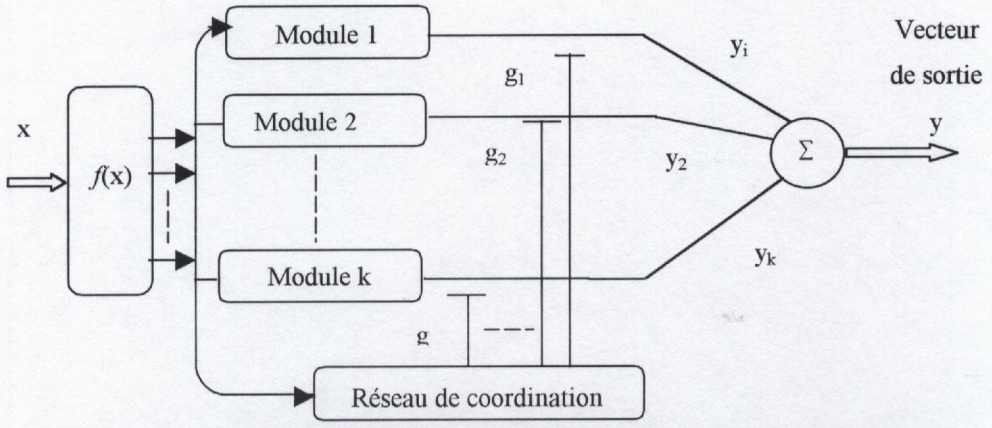


Figure (2.2) : *Réseau modulaire d'ordre élevé*

II.3.1 L'ALGORITHME D'APPRENTISSAGE D'UN RESEAU MODULAIRE D'ORDRE n :

a) Apprentissage du réseau expert :

L'algorithme d'apprentissage qu'on doit développer dans cas repose sur le même principe de l' algorithme d'apprentissage du réseau linéaire et contient les mêmes étapes.

Dans ce cas, la sortie du neurone m est donnée par :

$$y_i^m = w_i^m f(x) \tag{2.20}$$

tel que f(x) peut être :

- Une fonction linéaire :

$$f(x) = x^T \qquad x = [x_1, x_1, \ldots\ldots, x_p] \tag{2.21}$$

- Une fonction quadratique :

$$f(x) = x_k x_j \qquad pour \quad k, j = 1 \ldots\ldots p \tag{2.22}$$

- Une fonction polynomiale d'ordre r :

$$f(x) = x_{i1} x_{i2} \ldots\ldots x_{ir} \qquad pour\ i1, i2, \ldots ir = 1 \ldots p \tag{2.23}$$

Remarque :

Nous limitons notre étude à un réseau d'ordre deux

Dans le cas d'un réseau d'ordre deux on a :

$$f(x) = x_k x_j \qquad pour \quad k, j = 1 \ldots\ldots p \tag{2.24}$$

Notre but est de maximiser la fonction de vraisemblance donnée par l'équation (2.9), les poids synaptiques seront ajustés selon l'équation suivante :

$$w^m_{i,j,k}(n+1) = w^m_{i,j,k}(n) + \Delta w^m_{i,j,k}(n) \tag{2.25}$$

où

$$\Delta w^m_{i,j,k}(n) = \mu \frac{\partial L(w,g)}{\partial w^m_{i,j,k}(n)} \tag{2.26}$$

en utilisant la règle de Chain on peut écrire :

$$\frac{\partial L(w,g)}{\partial w^m_{i,j,k}} = \frac{\partial L(w,g)}{\partial y^m_i} \frac{\partial y^m_i}{\partial w^m_{i,j,k}} \tag{2.27}$$

$\frac{\partial L(w,g)}{\partial y^m_i}$ est donné par l'équation (1.22).

D'après les équations (2.24), (2.20) on trouve :

$$\frac{\partial y^m_i}{\partial w^m_{i,j,k}} = x_k x_j \tag{2.28}$$

donc l'équation (2.25) sera :

$$w^m_{i,j,k}(n+1) = w^m_{i,j,k}(n) + \mu h_1 e^m_i(n) x_k x_j \tag{2.29}$$

b) Apprentissage du réseau de coordination :

La sortie d'un neurone dans le réseau de coordination est donnée par :

$$u_i = a_i f(x) \tag{2.30}$$

Pour un réseau d'ordre deux $f(x)$ est donnée par l'équation (2.24).

L'adaptation des poids se fait de façon récursive selon l'équation suivante :

$$a^i_{j,k}(n+1) = a^i_{j,k}(n) + \Delta a^i_{j,k}(n) \tag{2.31}$$

$$\Delta a^i_{j,k}(n) = \mu \frac{\partial L(w,g)}{\partial a^i_{j,k}} \tag{2.32}$$

on a :

$$\frac{\partial L(w,g)}{\partial a^i_{j,k}} = \frac{\partial L(w,g)}{\partial u_i}\frac{\partial u_i}{\partial a^i_{j,k}} \tag{2.33}$$

$\frac{\partial L(w,g)}{\partial u_i}$ est donné par l'équation (1.37), et

d'après les équations (2.31), (2.24) on trouve :

$$\frac{\partial u_i}{\partial a^i_{j,k}} = x_k x_j \tag{2.34}$$

donc l'équation (2.29) sera :

$$w^i_{j,k}(n+1) = a^i_{j,k}(n) + \mu(h_i - g_i)x_k x_j \tag{2.35}$$

Résumé de l'algorithme d'apprentissage du réseau modulaire :

1. Initialiser les poids des réseaux experts et du réseau de coordination.
2. Calculer pour i=1,2,.....k et m=1,2,.....q

$$u_i = a_i f(x)$$

$$g_i = \frac{\exp(u_i)}{\sum_{i=1}^{k}\exp(u_i)}$$

$$y_i^m = w_i^m f(x)$$

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

$$e_i^m = d^m - y_i^m$$

$$w_i^m(k+1) = w_i^m(k) + \mu h_i e_i^m(k) f(x)$$

$$a_i(k+1) = a_i(k) + \mu(h_i(k) - g_i(k)) f(x)$$

3. Répéter l'étape 2 pour toutes les données.
4. Répéter les étapes 2 et 3 jusqu'à convergence de l'algorithme.

II.4 LES RESEAUX RECCURENT D'ORDRE DEUX :

II.4.1 PREMIERE REPRESENTATION MATHEMATIQUE :

Un réseau récurrent d'ordre deux [3] figure (2.3) contient n états de neurones avec retour de sorties et m entrées et N^2 x M poids synaptiques

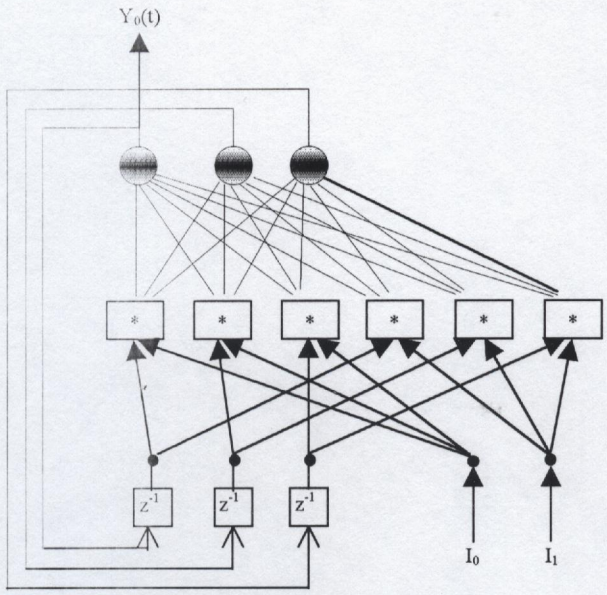


Figure (2.3) : *Réseau récurrent d'ordre deux.*

La sortie d'un neurone est définit par :

$$y_i(t+1) = f(v_i) \tag{2.36}$$

avec

$$v_i = \sum_{j=1}^{n}\sum_{k=1}^{m} w_{i,j,k}\, y_j(t)\, x_k(t) \tag{2.37}$$

f : est la fonction d'activation (sigmoïde).

L'algorithme d'apprentissage en temps réel :

L'algorithme est le même que pour le réseau récurrent d'ordre un, mais les variations des poids sont données par :

$$\Delta w_{lmn} = -\mu \frac{\partial J}{\partial w_{lmn}} \tag{2.38}$$

$$\begin{aligned} \frac{\partial J}{\partial w_{lmn}} &= \sum_{i\in T} e_i(t) \frac{\partial e_i}{\partial w_{lmn}} \\ &= -\sum_{i\in T} e_i(t) \frac{\partial y_i}{\partial w_{lmn}} \end{aligned} \tag{2.39}$$

pour développer l'équation (2.39) on doit déterminer le terme $\frac{\partial y_i}{\partial w_{lmn}}$:

$$\begin{aligned}\frac{\partial y_i}{\partial w_{lmn}} &= \frac{\partial y_i}{\partial v_i}\frac{\partial v_i}{\partial w_{lmn}} \\ &= f'(v_i)\frac{\partial v_i}{\partial w_{lmn}}\end{aligned} \tag{2.40}$$

$$\begin{aligned}\frac{\partial v_i}{\partial w_{lmn}} &= \frac{\partial}{\partial w_{lmn}}\left(\sum_{j=1}^{n}\sum_{k=1}^{m} w_{i,j,k} y_j(t) x_k(t)\right) \\ &= \sum_{j=1}^{n}\sum_{k=1}^{m}\left(\frac{\partial w_{i,j,k}}{\partial w_{lmn}} y_j(t) x_k(t) + w_{i,j,k} x_k(t)\frac{\partial y_j(t)}{\partial w_{lmn}}\right) \\ &= \left(\delta_{il} y_m(t) x_n(t) + \sum_{j=1}^{n}\sum_{k=1}^{m} w_{i,j,k} x_k(t)\frac{\partial y_j(t)}{\partial w_{lmn}}\right)\end{aligned} \tag{2.41}$$

$$\frac{\partial y_j(t+1)}{\partial w_{lmn}} = f'(v_i)\left[\delta_{il} y_m(t) x_n(t) + \sum_{j=1}^{n}\sum_{k=1}^{m} w_{i,j,k} x_k(t)\frac{\partial y_j(t)}{\partial w_{lmn}}\right]$$

$$p^j_{lmn}(t+1) = y_j(t)\left(1 - y_j(t)\right)\left[\delta_{il} y_m(t) x_n(t) + \sum_{j=1}^{n}\sum_{k=1}^{m} w_{i,j,k} x_k(t) p^j_{lmn}(t)\right] \tag{2.42}$$

avec

$$p^j_{lmn}(0) = 0 \tag{2.43}$$

alors l'équation (2.38) sera :

$$\Delta w_{ij}(t) = \mu \sum_{k \in U} e_k(t) p^k_{ij}(t) \tag{2.44}$$

Résumé de l'algorithme d'apprentissage du réseau récurrent d'ordre deux en temps réel :

1. Initialiser des poids synaptiques.
2. Calculer les sorties en utilisant les équations (2.36), (2.37).
3. Calculer les valeurs p^k_{ij} $k \in U, i \in U, j \in U \cup I$.
4. Calculer les variations $\Delta w_{ij}(t)$ en utilisant l'équation (2.44).
5. Ajuster les poids en utilisant (2.33).
6. Répéter les étapes 2 à 5 jusqu'à la convergence de l'algorithme.

II.4.2 DEUXIEME REPRESENTATION MATHEMATIQUE :

Les réseaux récurrents sont caractérisés par la connectivité dans les deux sens entre les unités figure (2.3).
L'état de chaque unité (ou neurone) est donné par l'équation différentielle suivante :

$$\dot{x}_i = -a_i x_i + b_i \sum_j w_{i,j} y_j \tag{2.45}$$

où
x_i :est l'état du i^{eme} neurone.
a_i ,b_i : sont des constantes
$w_{i,j}$: Le poids reliant l'entrée j avec le neurone i.
y_j est l'entrée précédente du neurone j tel que :

$$y_j = f(x_j) \tag{2.46}$$

où f est la fonction d'activation.
Considérant maintenant un réseau récurrent d'ordre élevé (RHONN) constitué de n neurones et m entrées, l'état de chaque neurone est donné par :

$$\dot{x}_i = -a_i x_i + b_i \left[\sum_{k=1}^{L} w_{ik} \prod_{j \in I_k} y_j^{d_j(k)} \right] \tag{2.47}$$

avec $\{I_1, I_2, \cdots, I_L\}$ est un ensemble de L sous-ensembles (non-ordonnée) de $\{1,2,\cdots,m+n\}$, $d_j(k)$ est un entier positif.

$y = [y_1, y_2, \cdots y_{m+n}]^T$ est le vecteur des entrées de chaque de chaque neurone défini comme suit :

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ y_{n+1} \\ \vdots \\ y_{m+n} \end{bmatrix} = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \\ u_1 \\ \vdots \\ u_n \end{bmatrix} \tag{2.48}$$

$u = [u_1, u_2, \cdots, u_m]^T$ est le vecteur des entrées extérieures du réseau.
Nous définissons un vecteur z de dimension L comme suit :

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_L \end{bmatrix} = \begin{bmatrix} \prod_{j \in I_1} y_j^{d_j(1)} \\ \prod_{j \in I_2} y_j^{d_j(2)} \\ \vdots \\ \prod_{j \in I_L} y_j^{d_j(L)} \end{bmatrix} \tag{2.49}$$

donc l'équation (2.47) sera écrite sous la forme suivante :

$$\dot{x}_i = -a_i x_i + b_i \left[\sum_{k=1}^{L} w_{ik} z_k \right] \tag{2.50}$$

pour faciliter l'écriture de l'équation (2.50) on définit un nouveau vecteur de paramètres ajustables :

$\theta_i = b_i[w_{i1}; w_{i2}, \cdots, w_{iL}]$, θ_i représente les poids ajustables du réseau.

Alors l'équation (3.47) devient :

$$\dot{x}_i = -a_i x_i + \theta^T z \tag{2.51}$$

II.4.2.1 LES ALGORITHMES D'APPRENTISSAGE :

Remarque :

Il est démontré qu'on peut approximer n'importe quel système non-linéaire avec un degré de précision quelconque par un RHONN si un nombre suffisant de connections est alloué à ce réseau [15].

a) APPRENTISSAGE PAR L'ALGORITHME D'ERREUR REGRESSIF FILTRE D'UN RHONN " Filtered regressor RHONN ":

Lemme **[15]** :

un système décrit par :

$$\dot{\chi}_i = -a_i \chi_i + \theta^{*T} z(\chi, u) \qquad \chi_i(0) = \chi_i^0 \tag{2.52}$$

peut être écrit sous la forme suivante :

$$\dot{\zeta}_i = -a_i \zeta_i + z \qquad \zeta_i(0) = 0 \tag{2.53}$$

$$\chi_i = \theta_i^{*T} \zeta_i + e^{-a_i t} \chi_i^0 \tag{2.54}$$

De (2.53) on a :

$$\zeta_i(t) = \int_0^t e^{-a_i(t-\tau)} z(\chi(\tau), u(\tau)) d\tau \tag{2.55}$$

alors :

$$\theta_i^{*T} \zeta_i + e^{-a_i t} \chi_i^0 = e^{-a_i t} \chi_i^0 + \int_0^t e^{-a_i(t-\tau)} z(\chi(\tau), u(\tau)) d\tau \tag{2.56}$$

En utilisant (2.56) on trouve que terme droit de l'équation (2.54) est égale à χ_i et

D'après le lemme (1) Le système dynamique décrit par l'équation (2.52) peut être écrit par :

$$\chi_i = \theta_i^{*T} \zeta_i + \varepsilon_i \qquad i = 1,2,\ldots,n \tag{2.57}$$

ζ_i est donné par l'équation (2.53), ce n'est qu'une autre forme du vecteur z (version filtrée).

Le terme $\varepsilon_i = e^{-a_i t} \chi_i^0$ apparaît quand l'état initial n'est pas nul.

L'erreur d'état entre le système et le modèle est :

$$e_i = x_i - \chi_i \tag{2.58}$$

$$e_i = \Phi_i^T \zeta_i - \varepsilon_i \tag{2.59}$$

où $\Phi_i = \theta_i - \theta_i^*$ est la matrice de l'erreur estimée des poids.

Notre but maintenant est de trouver des lois adaptatives pour ajuster les poids synaptiques. On peut atteindre ce but par l'utilisation des techniques d'optimisation pour minimiser la fonction de coût quadratique suivante :

$$J(\theta_1, \cdots, \theta_n) = \frac{1}{2} \sum_{i=1}^{n} e_i^2 = \frac{1}{2} \sum_{i=1}^{n} \left[(\theta_i - \theta_i^*)^T \zeta_i - \varepsilon_i \right]^2 \tag{2.60}$$

Par l'utilisation de la méthode du gradient on trouve :

$$\dot{\theta}_i = -\Gamma_i \zeta_i e_i \qquad i = 1,2,\ldots,n \tag{2.61}$$

où Γ_i est la matrice des taux d'apprentissage, Γ_i une matrice définie positive .

par l'utilisation de la technique du moindre carrée on trouve :

$$\begin{cases} \dot{\theta}_i = P_i \zeta_i e_i \\ \dot{P}_i = -P_i \zeta_i \zeta_i^T P_i \qquad i = 1,2,\ldots n \end{cases} \tag{2.62}$$

P(0) est une matrice symétrique définie positive.

La stabilité et la convergence des équations (2.61) (2.62) ont été démontrées dans **[15]**.

APPRENTISSAGE D'ERREUR FILTREE D'UN RHONN " Filtered error RHONN " :

Le système est toujours :

$$\dot{\chi}_i = -a_i \chi_i + \theta_i^{*T} z \tag{3.63}$$

Le model adopté pour identifier ce système est le suivant :

$$\dot{x}_i = -a_i x_i + \theta^T z \qquad i = 1,\ldots,n \tag{3.64}$$

l'erreur d'état $e_i = x_i - \chi_i$ dans ce cas est :

$$\dot{e}_i = -a_i e_i + \Phi_i^T z \qquad i = 1\ldots n \tag{3.65}$$

les poids seront ajustés suivant la loi :

$$\dot{\theta}_i = -\Gamma_i z e_i \tag{3.66}$$

Γ_i est la matrice des taux d'apprentissage , Γ_i est matrice définie positive.

Remarque :

Après la présentation théorique de quelques réseaux d'ordre élevé, une comparaison entre les réseaux d'ordre un et élevé concernant la vitesse d'apprentissage et la réduction de la taille doit être faite par des simulations.

Chapitre 03

IDENTIFICATION DES SYSTEMES NON-LINEAIRES PAR LES RESEAUX DE NEURONES D'ORDRE ELEVE

III.1.INTRODUCTION :

Dans toutes les disciplines de l'ingénieur et Sous l'influence du développement de l'automatisation, la compréhension et l'amélioration de tout fonctionnement passent nécessairement par la modélisation qui tient une place prépondérante dans tous les domaines. De ce fait de nombreuses études relatives à la détermination de modèles mathématiques des processus complexes ont été développées.

Pour trouver un modèle de représentation d'un processus industriel on est amené à approcher sa dynamique par un modèle linéaire et stationnaire dans un domaine plus ou moins restreint autour de son point de fonctionnement ; Cependant dans une large plage de fonctionnement, un tel modèle étant généralement non linéaire, on est obligé d'utiliser d'autres méthodes, plus puissantes.

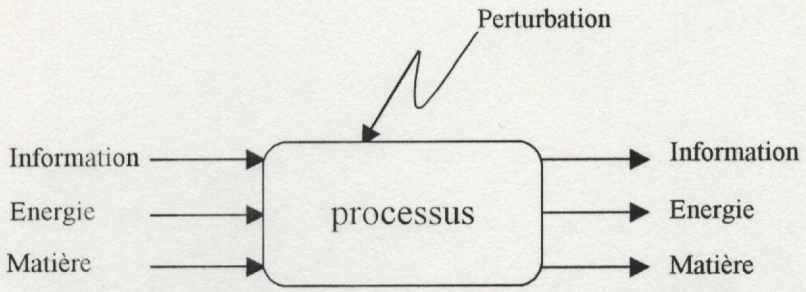
L'utilisation des réseaux de neurones dans les domaines du traitement d'images et de reconnaissance des caractères a donné des résultats très satisfaisants. Dans le domaine de l'identification, les réseaux de neurones ne sont utilisés largement qu'après la publication du chercheur ***Kumpati.S.Narendra [18]*** *1990 où il montre que les réseaux de neurones permettent d'identifier et de contrôler des systèmes non linéaires. Les chercheurs* ***P.Ioannou et M.Christodoulou [15]*** *ont montré que les réseaux d'ordre élevé peuvent être aussi utilisés pour l'identification et le contrôle des systèmes non linéaires.*

Le présent chapitre est divisé en deux parties :

Dans la première partie, on va citer les notions de processus et de modèle, ensuite on expose le principe d'identification par réseaux de neurones d'ordre élevé. Dans la deuxième partie, on présente des simulations concernant l'identification des systèmes SISO et MIMO.

III.2.NOTION DE PROCESSUS :

Un processus est un système dynamique, c'est à dire un système évolutif pour lequel le temps joue un rôle fondamental.
Dans le cas général figure(3.1) un processus est un système traversé par des flux d'information, d'énergie et de matière tout en étant soumis à des perturbations ayant l'une des trois formes précitées.



Figure(3.1) : *Le cas général d'un processus*

Du point de vue d'un observateur, un processus correspond à un système physique envisagé dans le cadre de l'évolution des échanges réalisés avec son environnement. Il faut noter que divers variables peuvent être mises en évidence sur un processus :

- *Des entrées* de commande, qui permettent d'agir sur l'évolution du processus.
- *Des entrées* de perturbations, en général non contrôlable par l'utilisateur et qui agissent également le processus.
- *Des sorties*, variables mesurables ou au moins détectables, qui caractérisent l'action du processus sur son environnement.
- *Des variables d'état,* variables internes au système, dont l'action sur l'environnement n'est pas nécessairement directement perceptible mais aussi dont l'évolution régit celle du processus.

L'étude et la commande d'un processus s'effectuent à partir d'un modèle de ce processus, il existe plusieurs types de modèles, principalement les modèles de connaissance d'une part, et les modèles de représentation d'autre part ; il est important de noter que dans tous les cas, le système existe indépendamment de tout modèle que l'on peut lui attribuer, et que le modèle n'est le plus souvent qu'une simplification et une caractérisation de la réalité.

III.3.NOTION DE MODELE :

III.3.1.MODELES DE CONNAISSANCES :

Un modèle de connaissances est un modèle dont les caractéristiques et les équations ont été établies en faisant appel à des modèles plus généraux mettant en œuvre les lois de la physique de la chimie, de la biologie,... ;Les paramètres d'une telle méthode ont alors une interprétation physique directe : température, pression, courant, force,.... ;ils sont beaucoup plus riches de signification que les modèles de représentation définis ci-dessous et contiennent toutes les informations utiles sur le processus étudié. Par contre ils sont en général difficile à déterminer et de mise en œuvre complexe.

III.3.2.MODELES DE REPRESENTATION :

Ce modèle ne permet pas, le plus souvent, d'interprétation physique des phénomènes étudiés. Il est constitué d'un ensemble de relations mathématiques qui vont relier dans un domaine d'évolution donné les différentes variables du processus, les paramètres d'un tel modèle peuvent n'avoir aucun sens physique particulier connu.
Les modèles utilisés dans les divers cas envisagés peuvent être très différents :

- Modèle à temps continu linéaire
- Modèle à temps continu non-linéaire.
- Modèle à temps discontinu ou discret linéaire.
- Modèle à temps discontinu non-linéaire.

III.4.IDENTIFICATION DES SYSTEMES :

L'identification constitue une phase importante dans la définition du modèle c'est par elle que le choix de la classe de modèles à adopter puis les valeurs des paramètres qui le caractérisent vont se préciser.
Le plus souvent l'identification s'effectue en optimisant un critère de qualité qui caractérise l'écart entre le comportement du processus (repère par un ensemble de mesures), et celui de son modèle (étudié par simulation) pour un ensemble de sollicitations données. Des méthodes d'identification sont précisées dans la suite correspondant le plus souvent à l'un des schémas des figures (3.2) et (3.3).

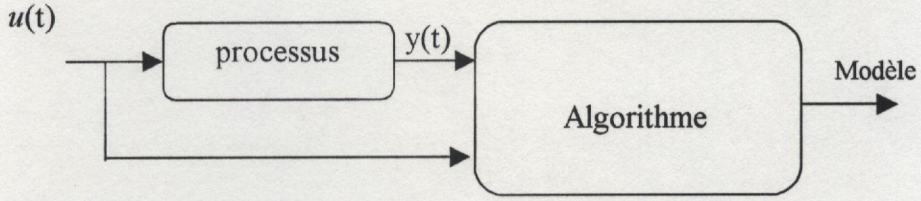


Figure (3.2) : *identification à partir du comportement entrées-sorties.*

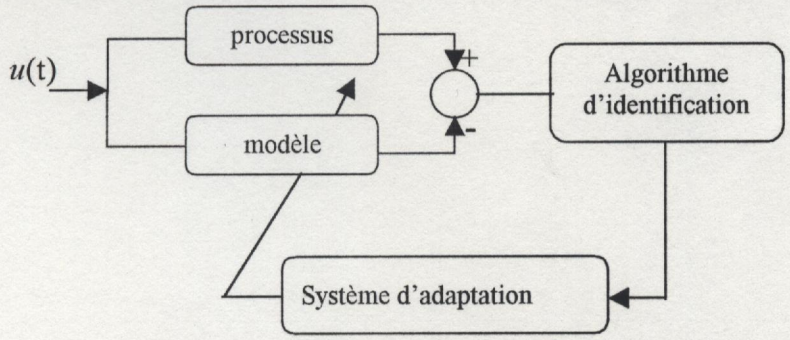


Figure (3.3) : *identification de type paramétrique.*

Dans la représentation (I.S.O) (Input. State. Output), Les systèmes dynamiques sont souvent représentés par des équations différentielles ou aux différences :
Dans le cas d'une représentation par les équations différentielles le système peut être écrit de la façon suivante :

$$\begin{aligned} \dot{x}(t) &= f(x(t),u(t),t) \\ \dot{y}(t) &= h(x(t),u(t),t) \end{aligned} \qquad (3.1)$$

où

$x(t) = [x_1(t), x_2(t), \cdots x_n(t)]^T$ représente l'état du système.

$y(t) = [y_1(t), y_2(t), \cdots y_m(t)]^T$ représente la sortie du système.

$u(t) = [u_1(t), u_2(t), \cdots u_p(t)]^T$ représente la commande du système.

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

$$\begin{aligned} x(k+1) &= \Phi(x(k),u(k),k) \\ y(k+1) &= \Psi(x(k),u(k),k) \end{aligned} \qquad (3.2)$$

Dans le cas où les fonctions Ψ et Φ sont inconnues, le but de l'identification consiste à les déterminer.

III.5.IDENTIFICATION DES SYSTEMES NON-LINEAIRES PAR LES RESEAUX DE NEURONES D'ORDRE ELEVE :

L'identification par réseau de neurones a fait l'objet de plusieurs études après 1990 où le chercheur **kumpati.S** a montré que les réseaux de neurones peuvent être utilisés dans le domaine de l'identification et du contrôle. Des études récentes faites sur les réseaux de neurones d'ordre élevé [15] ont montré que ces types de réseaux sont capables d'identifier n'importe quel système non linéaire si un nombre suffisant de neurones est attribué au réseau.
Dans l'identification par réseaux de neurones on remplace la fonction classique par un modèle neuronal pour un système qui a comme équation :

$$y(k) = f(y(k-1), y(k-2), \ldots, y(k-n), u(k-1), u(k-2), \ldots, u(k-m)) \qquad (3.3)$$

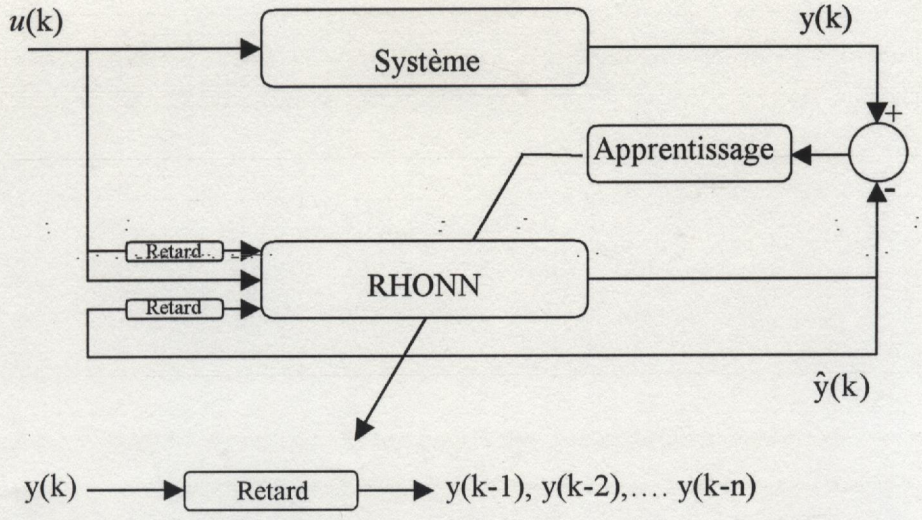
Le principe d'identification consiste à trouver le réseau de neurones qui reproduit la fonction non linéaire $f(*)$.
Durant l'identification l'apprentissage supervisé est réalisé ; où les poids sont ajustés de manière récursive, et pour des résultats plus satisfaisants, l'entrée doit être un signal riche en fréquence.
Les deux structures de l'identification classique parallèle, et série-parallèle peuvent être utilisées la première correspond aux réseaux récurrents et la deuxième aux réseaux non récurrents [18].

III.5.1.IDENTIFICATION PAR MODELE NEURONAL RECURRENT :

Le modèle neuronal utilisé pour l'identification du système décrit par (3.3) a la forme suivante :

$$y(k) = N(\hat{y}(k-1), \hat{y}(k-2), \ldots, \hat{y}(k-n), u(k-1), u(k-2), \ldots, u(k-m)) \qquad (3.4)$$

N est un réseau de neurones.
La caractéristique principale de l'identification par le modèle parallèle est qu'on utilise les sorties du modèle au lieu des sorties du système pour l'entraînement du réseau figure(3.4).
L'inconvénient majeur de cette méthode est le problème de la stabilité durant la tâche d'identification c'est à dire que rien ne garantit la convergence des poids vers des valeurs constantes.



HONN : signifié réseaux de neurones d'ordre élevé (High Order Neural Network).

Figure (3.4) : *identification par modèle parallèle.*

III.5.2.IDENTIFICATION PAR MODELE NEURONAL NON RECURRENT :

Le modèle neuronal utilisé pour l'identification du système dans ce cas est :

$$y(k) = N\left(\hat{y}(k-1), \hat{y}(k-2), \ldots, \hat{y}(k-n), u(k-1), u(k-2), \ldots, u(k-m)\right) \quad (3.5)$$

Dans l'identification par le modèle série-parallèle, les sorties du système sont utilisées directement pour l'entraînement du réseau figure(3.5).

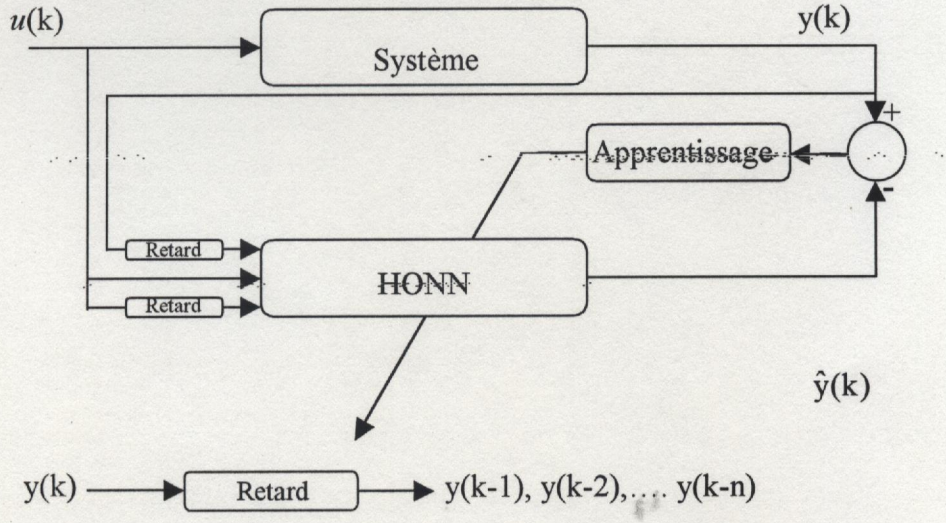


Figure (3.5) : *identification par modèle série-parallèle.*

III.6 IDENTIFICATION DES SYSTEMES SISO :

Remarque :
Pour chaque type de réseau et dans chaque simulation, on a effectué plusieurs essais sur la taille du réseau, le taux d'apprentissage et le nombre de données nécessaire pour faire l'apprentissage.
Le micro-ordinateur utilisé dans les simulations, possède un µp pentium 166 MHZ et une RAM de 40Mo . Le langage utilisé est le Borland Pascal

Simulation 1 :
Le système SISO à identifier est représenté par l'équation aux différences suivante :

$$y(k+1) = f(y(k)) + u(k) \qquad (3.6)$$

$f(y(k))$ est choisie comme suit :

$$f(y(k)) = \frac{y(k)}{1+y^2(k)} \qquad (3.7)$$

le modèle neuronal d'identification a la forme :

$$y(k+1) = N(y(k)) + u(k) \qquad (3.8)$$

N : représente un réseau de neurones.

L'apprentissage est réalisé par des réseaux récurrents d'ordre 1 et 2 et des réseaux multicouches d'ordre 1,2 et 3.Le tableau comparatif (3.1) résume les caractéristiques et le nombre d'itérations pour chaque simulation.
Concernant L'état du système et d'après la (figure (3.6)), le système libre a y(0)=0 comme point d'équilibre. Le système est aussi stable en présence d'une entrée bornée (figure (3.7)) en prenant comme condition initiale y(0)=0.

Le réseau multicouche d'ordre 1 appartenant à la classe N^4_{1931} avec un taux d'apprentissage $\mu = 0.5$ réalise l'apprentissage du réseau au bout de 2800 itérations, mais un réseau MLP d'ordre 2 appartenant à la classe N^3_{151} avec un taux d'apprentissage $\mu = 0.25$ nécessite 5800 itérations pour réaliser l'apprentissage. Avec le même taux d'apprentissage et le nombre de données utilisés pour le réseau d'ordre 2, le réseau MLP d'ordre 3 appartenant à la classe N^3_{141} converge moins rapidement qu'un réseau d'ordre1 ou 2 (figure (3.8)).
Le réseau récurrent d'ordre 1 de 5 neurones converge moins rapidement qu'un réseau d'ordre 2 de 3 neurones pour les mêmes caractéristiques (figure (3.9)).

La superposition des données de sorties du système et du modèle est donnée par la (figure (3.10)), et il est mieux pour ce système de travailler avec un réseau récurrent d'ordre 2 vu le nombre d'itérations réduit nécessaire pour la convergence.

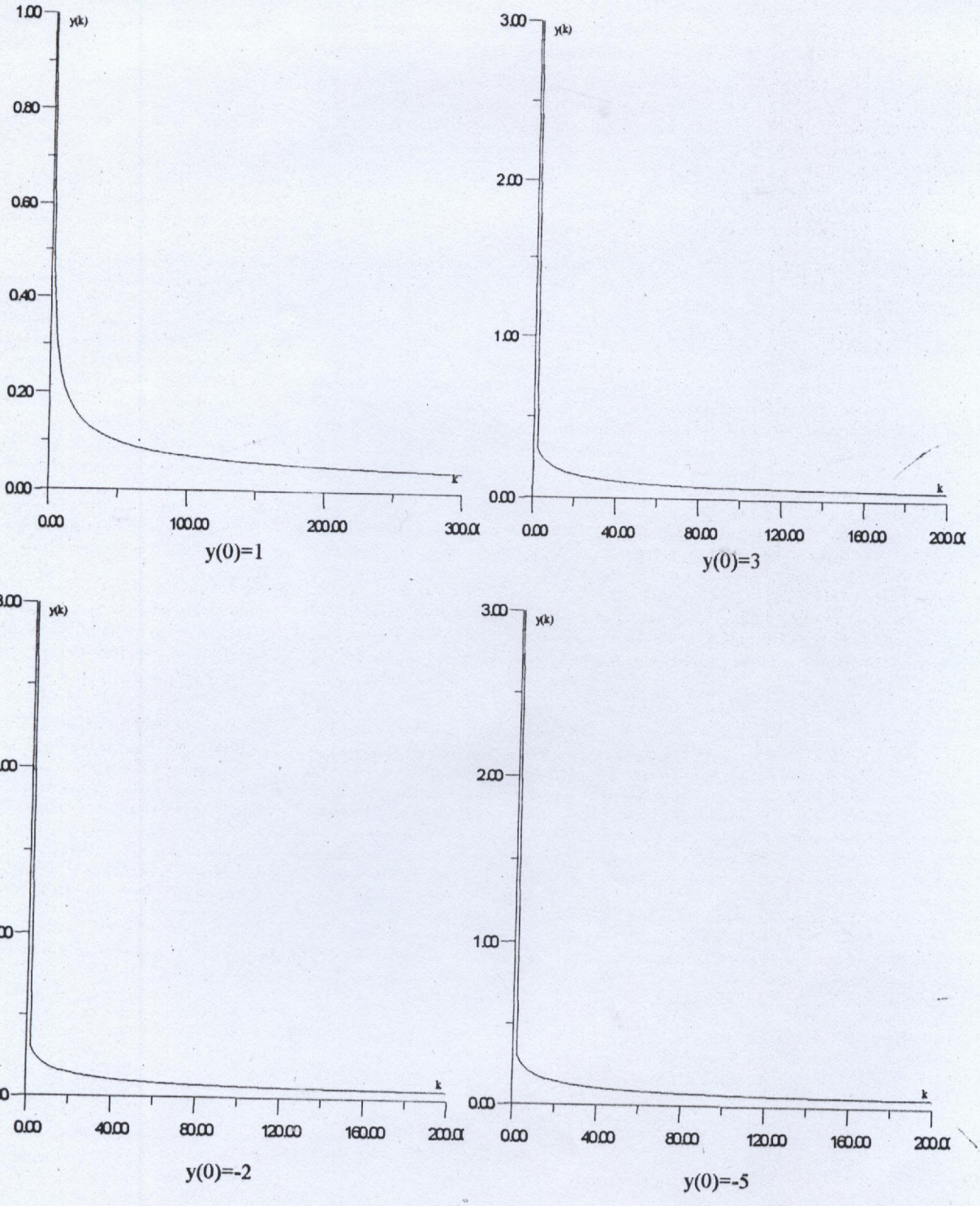


Figure (3.6) : *test de stabilité du système pour une entrée u(0)=0*

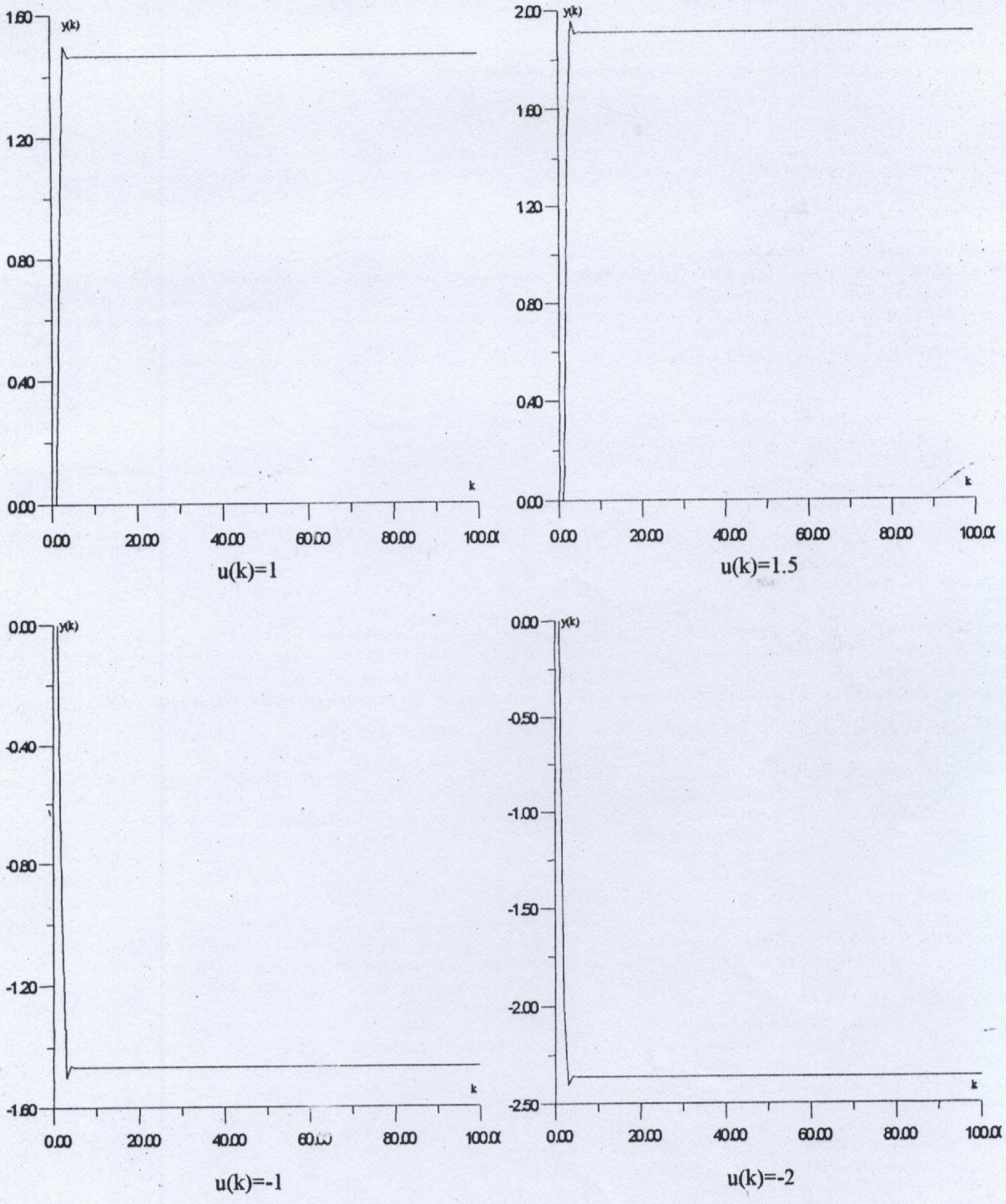
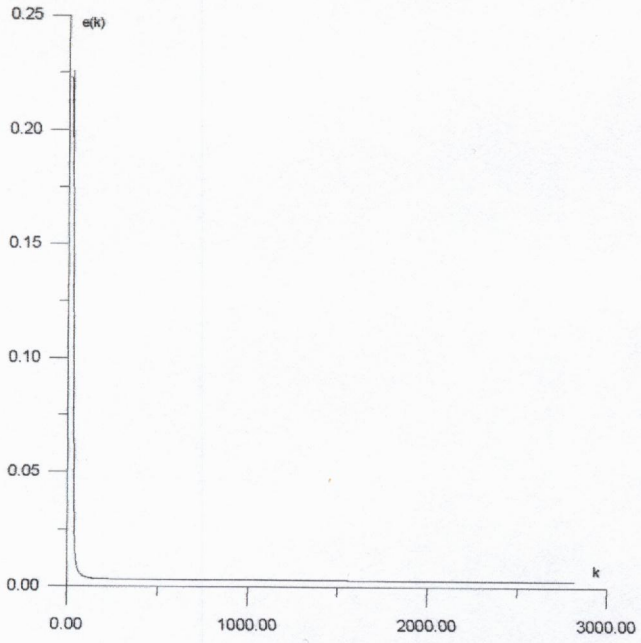
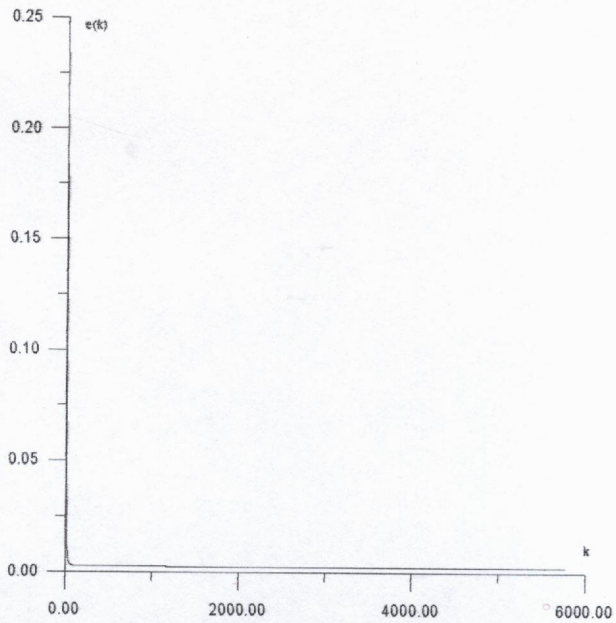


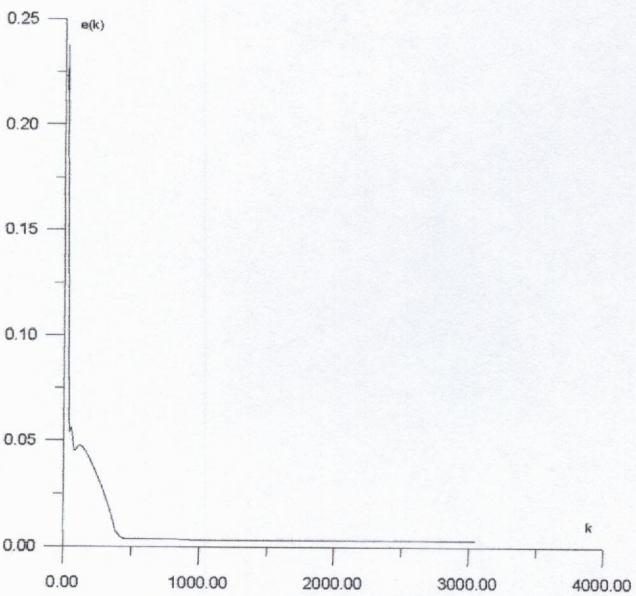
Figure (3.7) : *test de stabilité du système en présence d'une entrée bornée.*



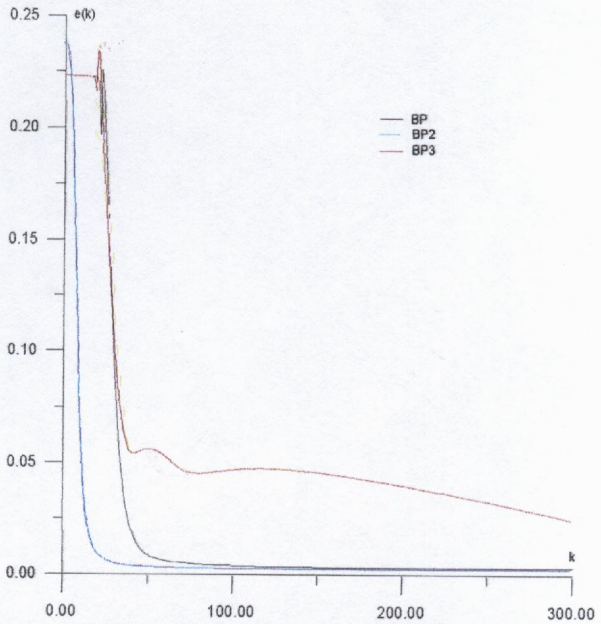
a) Erreur d'apprentissage pour un réseau M.L.P



b) Erreur d'apprentissage pour un réseau M.L.P D'ordre 2

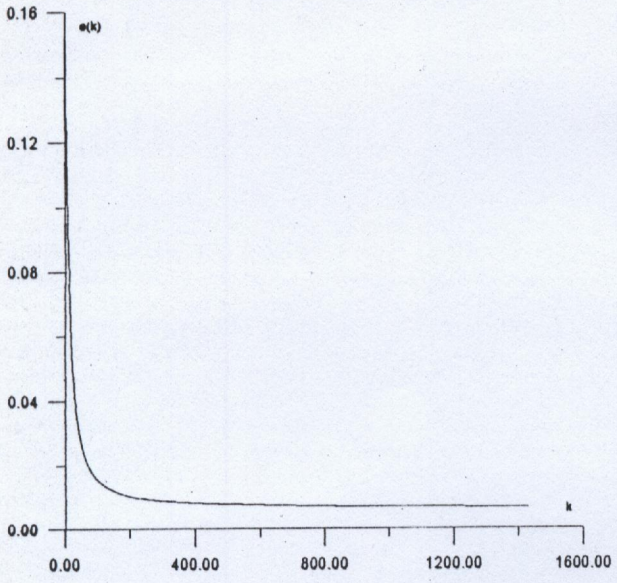


c) Erreur d'apprentissage pour un réseau M.L.P D'ordre 3

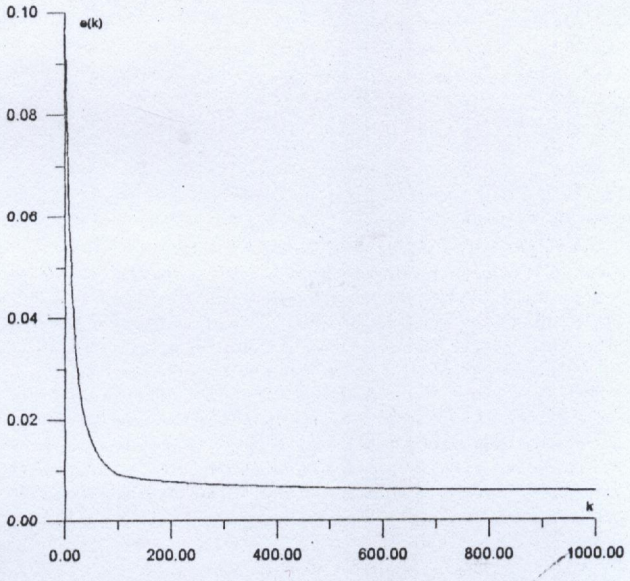


d) Comparaison de la vitesse d'apprentissage entre Les trois types du réseau M.L.P

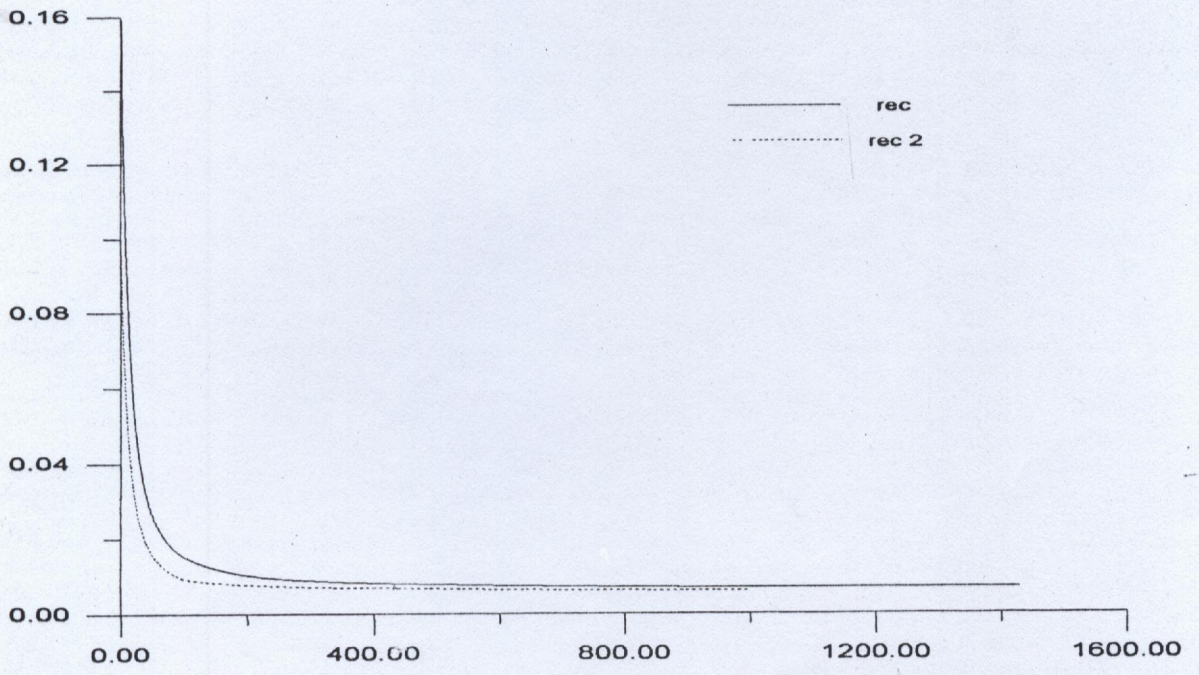
Figure (3.8) : *vitesse d'apprentissage des réseaux multicouches.*



a) Erreur d'apprentissage pour un réseau récurrent



b) Erreur d'apprentissage pour un réseau récurrent D'ordre 2



c) comparaison entre la vitesse d'apprentissage des deux types des réseaux récurrents.

Figure (3.9) *: vitesse d'apprentissage des réseaux récurrents.*

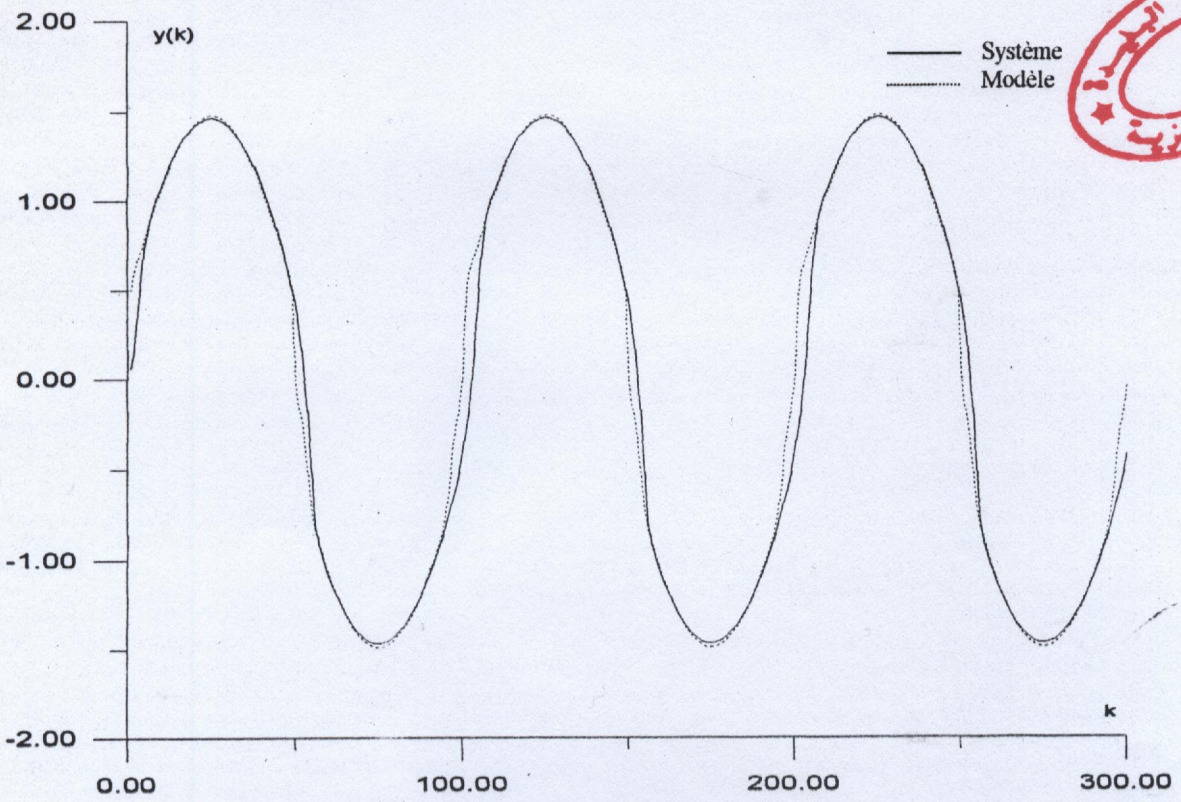


Figure (3.10) : *sortie du système et du modèle pour une entrée :* $u(k) = sin(2\pi k/100)$.

Type du Réseau	Taille du réseau	Pas d'apprentissage	Nombre de données	Nombre d'itérations
MLP	N_{1931}	0.25	100	2800
MLP(2)	N_{151}	0.25	100	5800
MLP(3)	N_{141}	0.25	100	3100
Récurrent	5 neurones	0.001	100	1500
Récurrent (2)	3 neurones	0.001	100	1000

Tableau (3.1) : *caractéristiques de la première simulations.*

Simulation2 :

Le système SISO à identifier est représenté par l'équation (3.6):
Dans ce cas, f est choisie comme suit :

$$f(y(k)) = \sin(\cos(y(k))) \qquad (3.9)$$

le modèle neuronal d'identification a la même forme :

$$y(k+1) = N(y(k)) + u(k) \qquad (3.10)$$

D'après la figure (3.11) l'état d'équilibre du système libre est la séquence "0.69" et d'après la figure (3.12) le système est borné avec quelques entrées bornées pour une condition initiale y(0)=0

L'adaptation des poids est faite par les réseaux multicouches d'ordre 1,2 et 3, les réseaux récurrents d'ordre 1 et 2 et les réseaux modulaires d'ordre 1 et 2.
Le tableau (3.2) résume les caractéristiques et le nombre d'itérations nécessaires pour la convergence de l'algorithme d'apprentissage pour chaque type de réseau.

Pour les réseaux multicouches, la figure (3.13) montre que l'adaptation des poids synaptiques du réseau MLP d'ordre 1 appartenant à la classe N^4_{1821} est réalisé au bout de 10000 itérations, cependant pour les mêmes performances et pour un réseau d'ordre 2 de la classe N^3_{151} et un réseau d'ordre 3 de la classe N^3_{141} l'adaptation a requis respectivement 5600 itérations et 5200 itérations.

Pour les réseaux modulaires, la figure (3.14) montre que l'apprentissage d'un réseau modulaire d'ordre 2 de 3 modules (7 neurones) s'est fait rapidement par rapport à un réseau modulaire d'ordre 1 de 5 modules (11 neurones).

L'adaptation des poids synaptiques pour un réseau récurrent d'ordre 1 de 5 neurones en utilisant l'algorithme d'apprentissage en temps réel nécessite plus de 6000 itérations.

L'adaptation avec un réseau récurrent d'ordre 2 s'est faite plus rapidement par apport à un réseau d'ordre 1 mais l'algorithme diverge par la suite (au paragraphe I.8.2 (b) il est vu que rien ne garantit la convergence d'un réseau récurrent utilisant l'algorithme d'apprentissage en temps réel).
Dans notre cas et suivant les résultats obtenus, il est préférable de travailler avec un réseau modulaire d'ordre 2 ou un réseau récurrent d'ordre 2 (dans le dernier cas, l'apprentissage doit être arrêter à l'itération 160).

La figure(3.17) montre la superposition de la sortie du système et celle du modèle neuronal.
Dans l'annexe, on trouve les valeurs des poids synaptiques initiales et finales pour cette simulations.

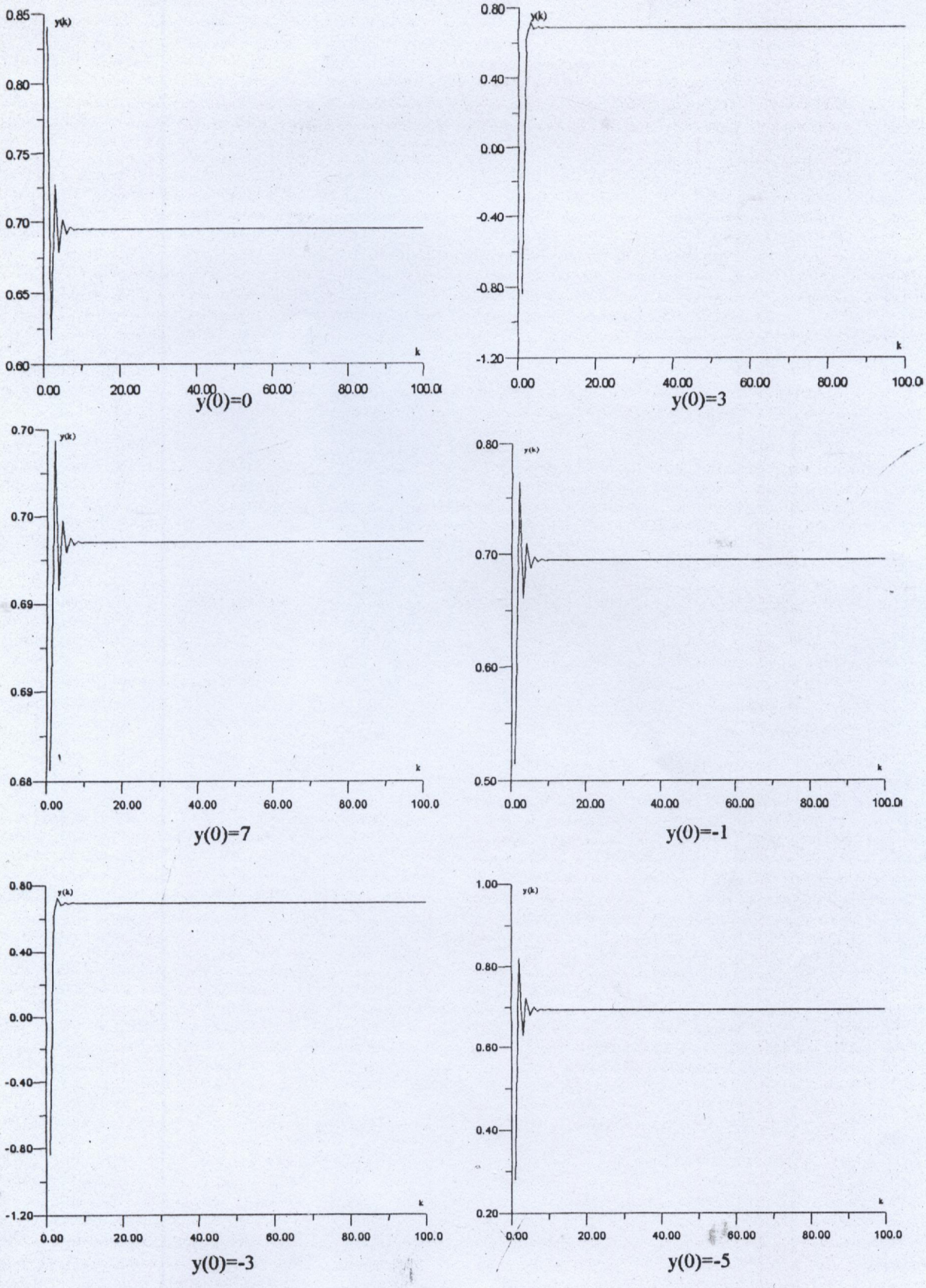


Figure (3.11) : *test de stabilité du système pour une entrée u(k)=0.*

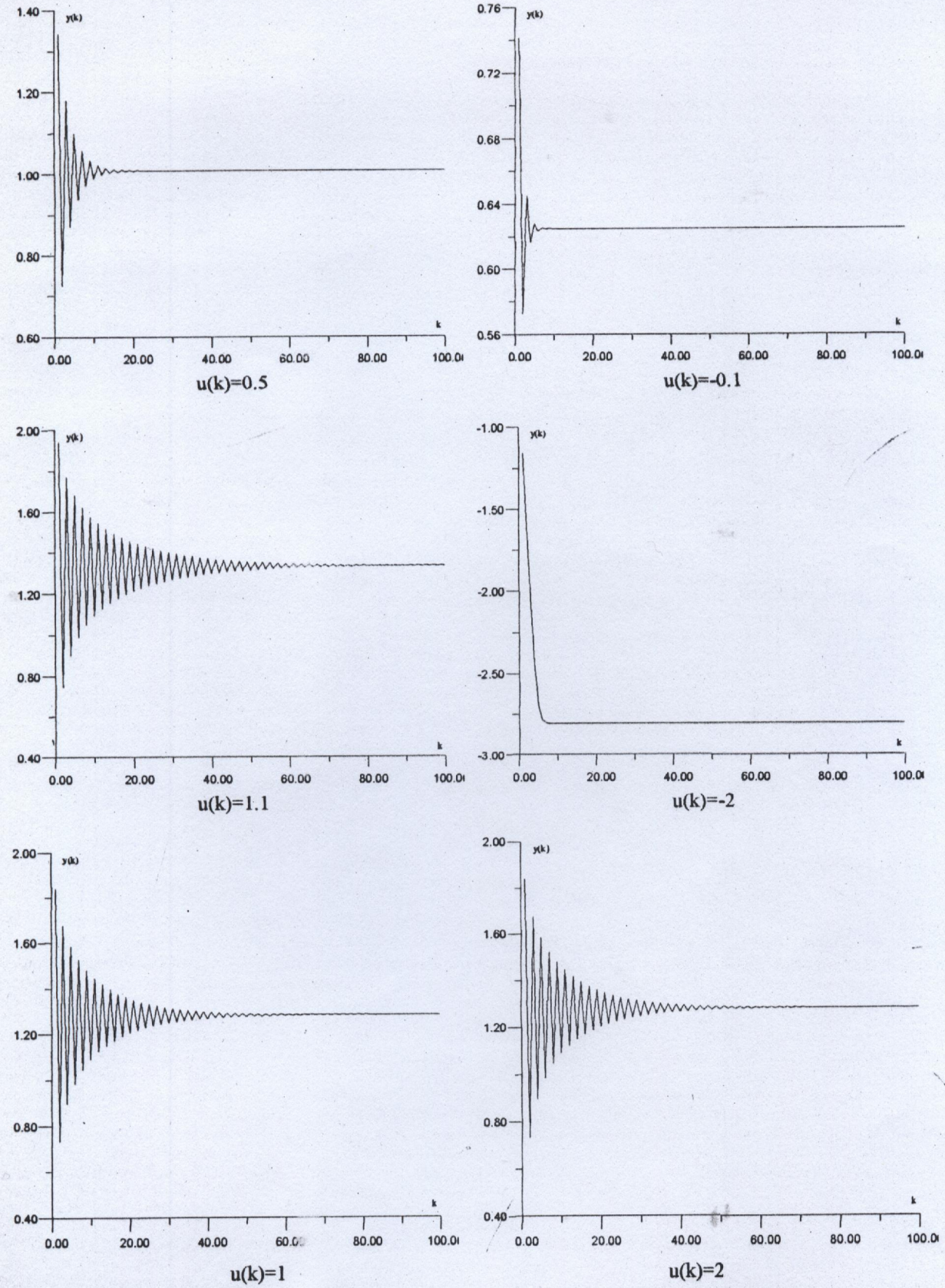
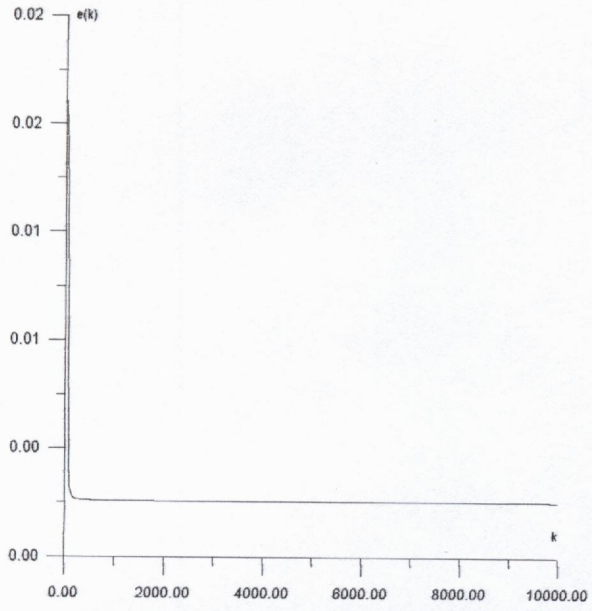
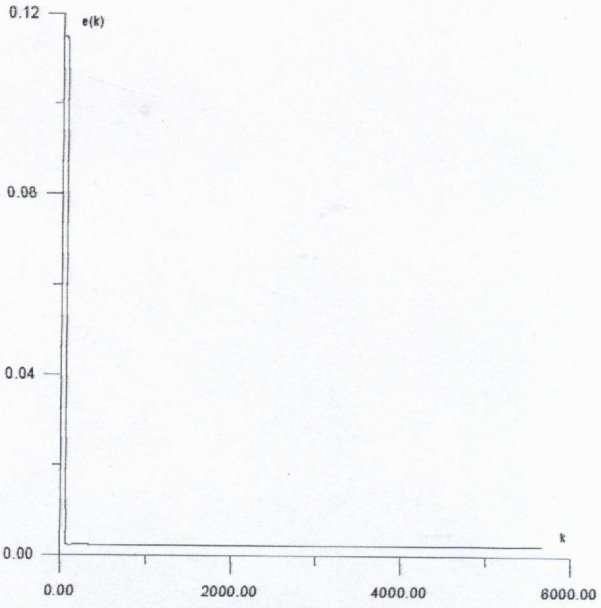


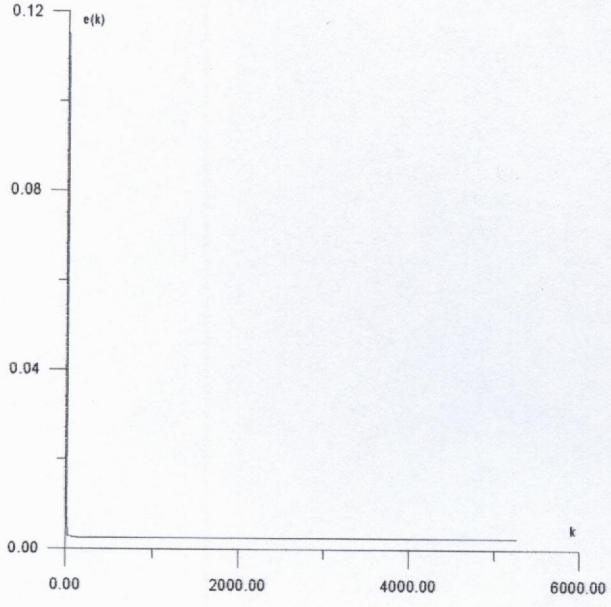
Figure (3.12) : *test de stabilité du système en présence d'une entrée bornée.*



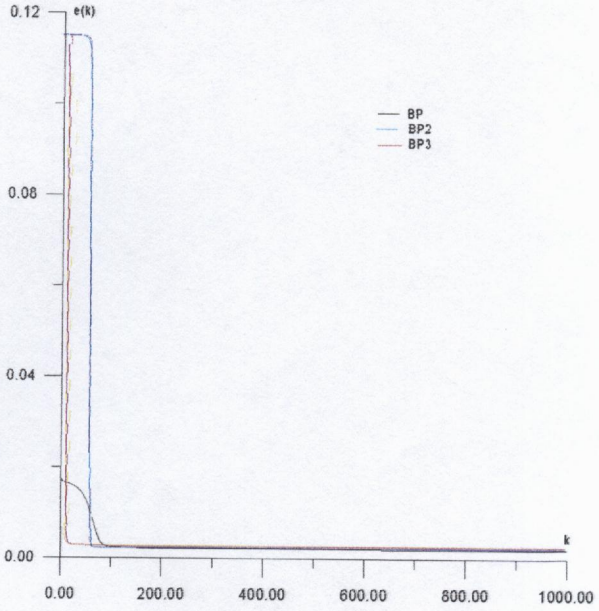
a) Erreur d'apprentissage pour un réseau M.L.P



b) Erreur d'apprentissage pour un réseau M.L.P
D'ordre 2

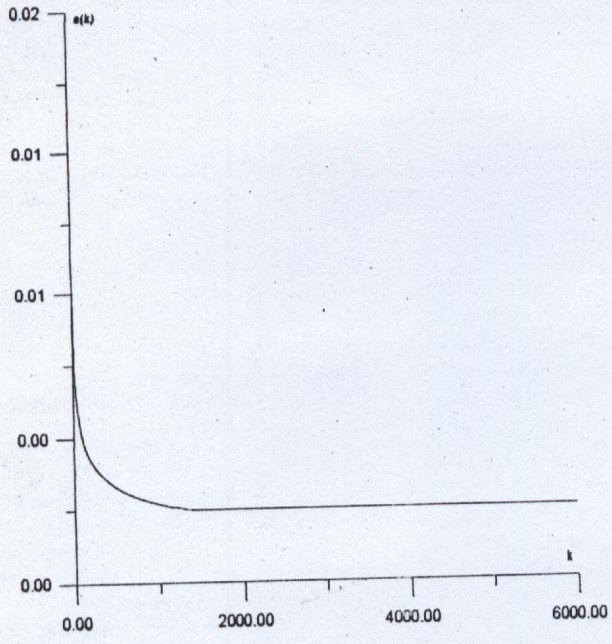


c) Erreur d'apprentissage pour un réseau M.L.P D'ordre 3

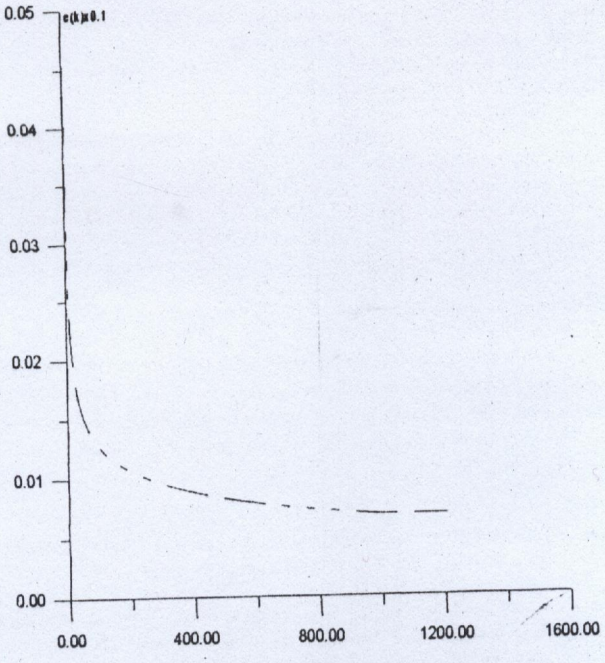


d) Comparaison de la vitesse d'apprentissage entre Les trois types du réseau M.L.P

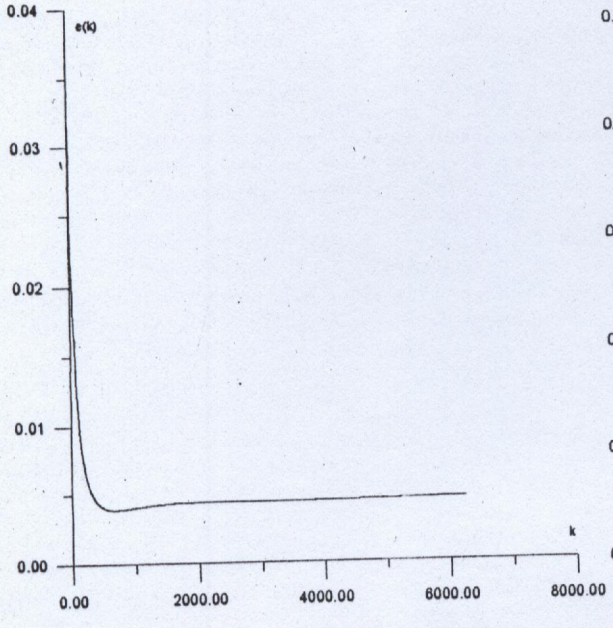
Figure (3.13) *: vitesse d'apprentissage des réseaux multicouches*



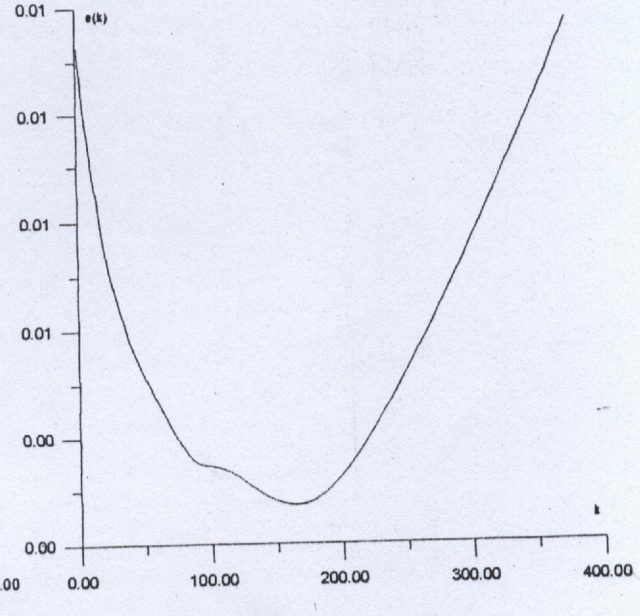
a) Erreur d'apprentissage pour un réseau modulaire



b) Erreur d'apprentissage pour un réseau modulaire D'ordre 2

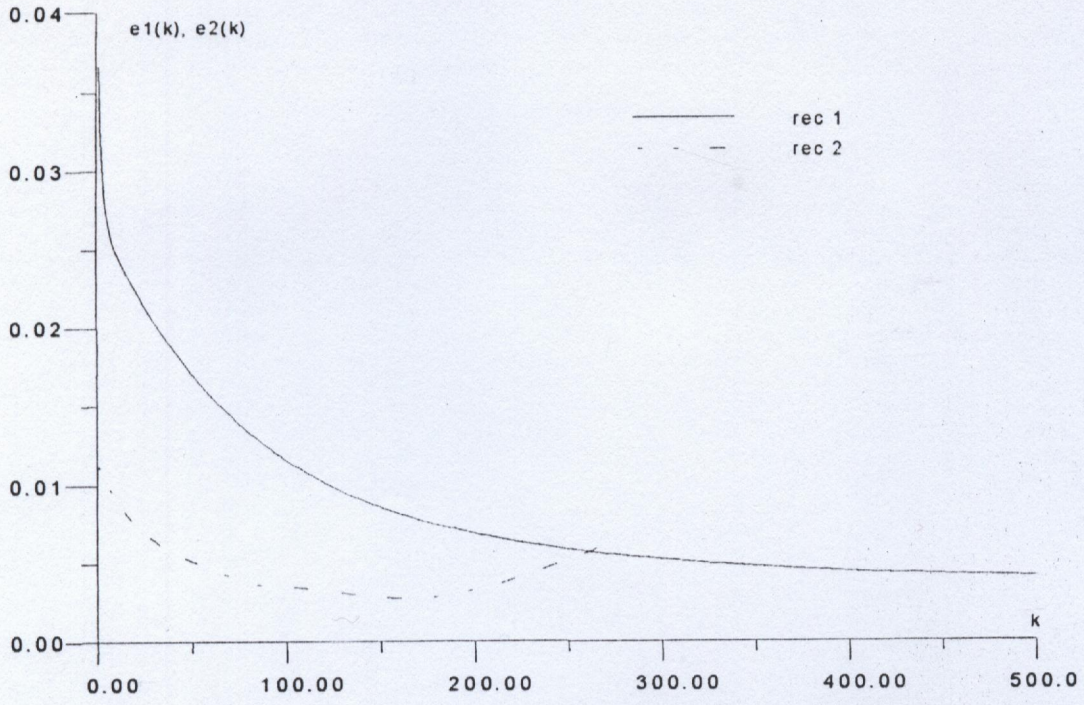


c) Erreur d'apprentissage pour un réseau récurrent

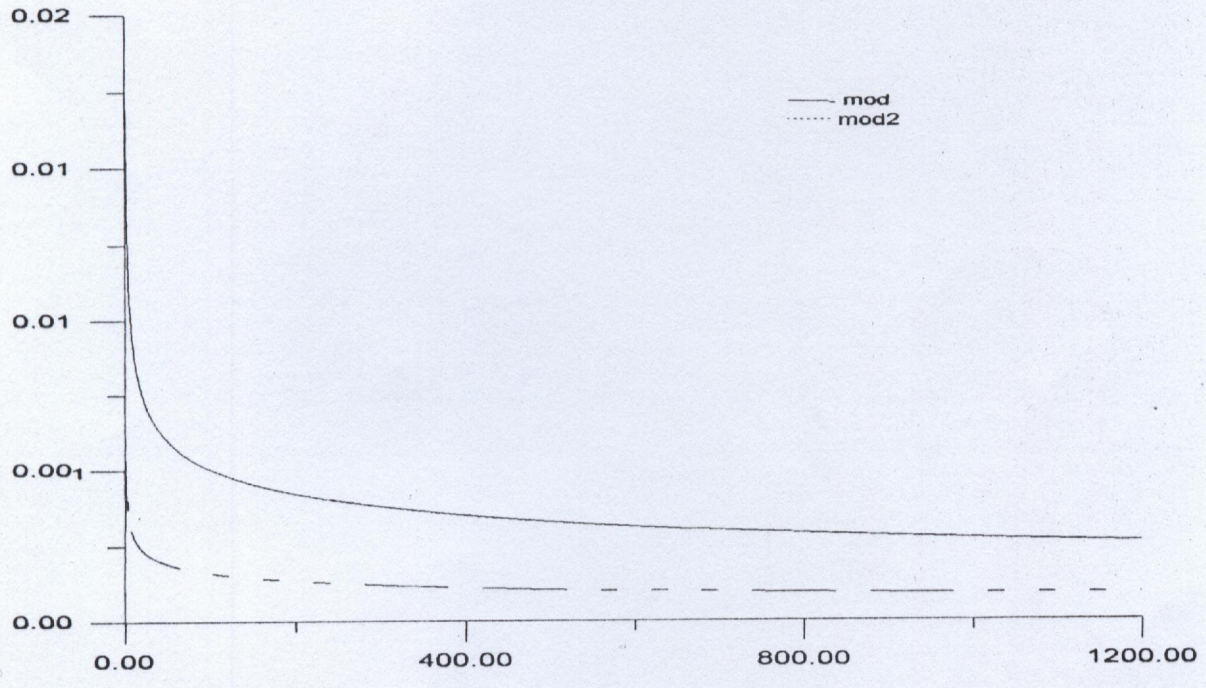


d) Erreur d'apprentissage pour un réseau récurrent d'ordre 2

Figure (3.14) *: vitesse d'apprentissage des réseaux modulaires et récurrents.*



Figure(3.15) : *Comparaison entre la vitesse d'apprentissage des deux types du réseau récurrent*



Figure(3.16) : *Comparaison entre la vitesse d'apprentissage des deux types du réseau modulaire*

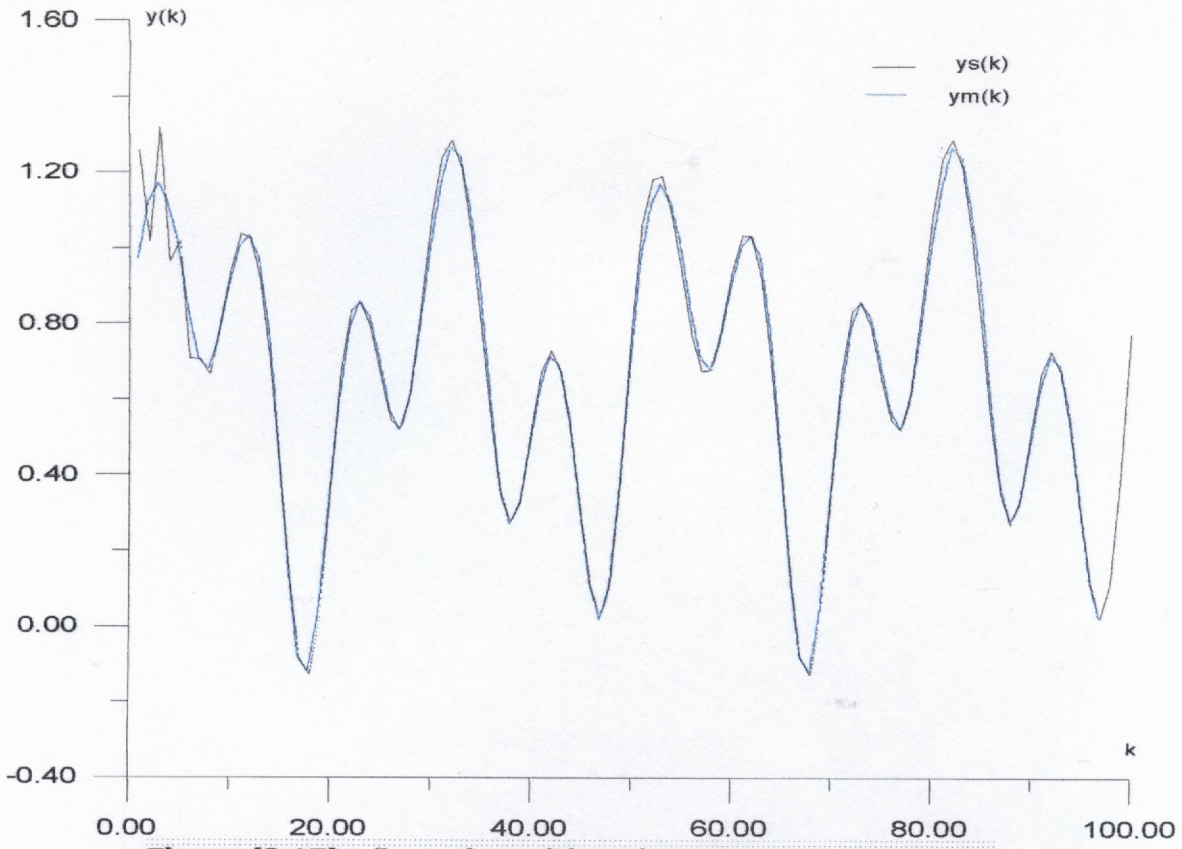


Figure (3.17) : *Sortie du modèle et du système pour une entrée :*

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

Type du Réseau	Taille du réseau	Pas d'apprent	Nbre de données	Nbre d' itérations	Temps d'apprentissage
MLP	N_{1821}	0.25	100	10000	1min 4sec 86ct
MLP(2)	N_{151}	0.25	100	5600	4sec et 12 ct
MLP(3)	N_{141}	0.25	100	5200	4sec et 67ct
Récurrent	5 neurones	0.001	100	6000	3min 18sc 83ct
Récurrent (2)	3 neurones	0.001	100	1200	1min 1sec 25ct
Modulaire	5 modules	0.001	100	6200	4 min 48sc 13ct
Modulaire (2)	3 modules	0.001	100	350	1min 2sec 59ct

Tableau (3.2) : *Caractéristiques de la deuxième simulation.*

III.7 IDENTIFICATION DES SYSTEMES MULTIVARIABLES :

Le système MIMO à identifier est représenté par l'équation suivante :

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} f(y_1(k)), y_2(k) \\ g(y_1(k), y_2(k)) \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \qquad (3.11)$$

où

$$f(y_1(k), y_2(k)) = \frac{y_1(k)}{1+y_2^2(k)}$$
$$g(y_1(k), y_2(k)) = \frac{y_1(k)y_2(k)}{1+y_2^2(k)} \qquad (3.12)$$

Le modèle neuronal utilisé pour l'identification a la forme suivante :

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} N_1(y_1(k)), y_2(k) \\ N_2(y_1(k), y_2(k)) \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \qquad (3.13)$$

le système libre y_1 est stable suivant les conditions initiales et y_2 a la séquence ''0'' comme état d'équilibre figure (3.18) et (3.19), et d'après les figures (3.20) et (3.21) le système est borné en présence d'une entrée bornée.

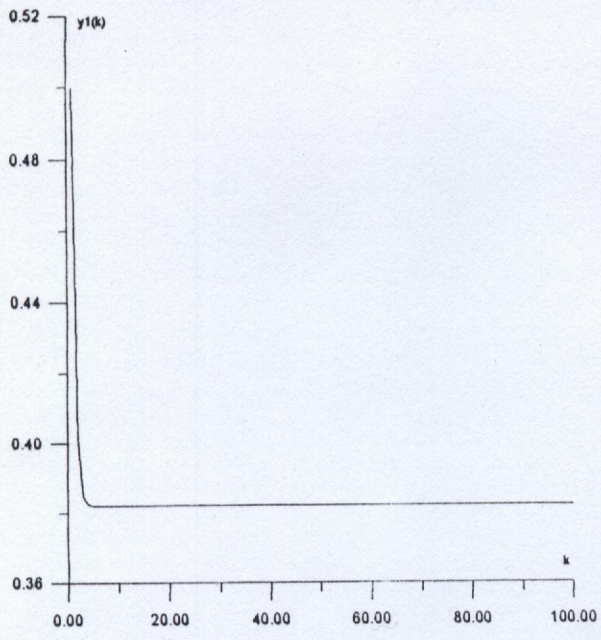
L'adaptation des poids synaptiques est réalisée par les réseaux MLP d'ordre 1,2 et 3, les réseaux récurrents d'ordre1 et 2 et les réseaux modulaires d'ordre 1 et 2.

Les tableau (3.3) et (3.4) résument les caractéristiques et les nombres des itérations pour chaque simulation.

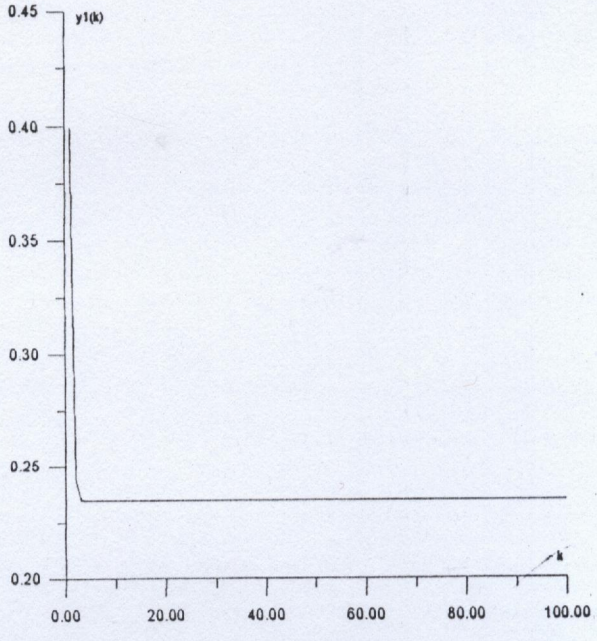
L'adaptation des réseaux modulaires est réalisée au bout de 2000 itérations pour y_1 et 3000 itérations pour y_2 avec les deux types de réseaux mais en utilisant un réseau de taille réduite pour les réseaux d'ordre 2.

Pour les réseaux récurrents, l'apprentissage est assuré par un réseau d'ordre 1 de 7 neurones et un réseau d'ordre 2 de 4 neurones ; la convergence est réalisé après environ 2700 à 3000 itérations pour y_1 et y_2 .

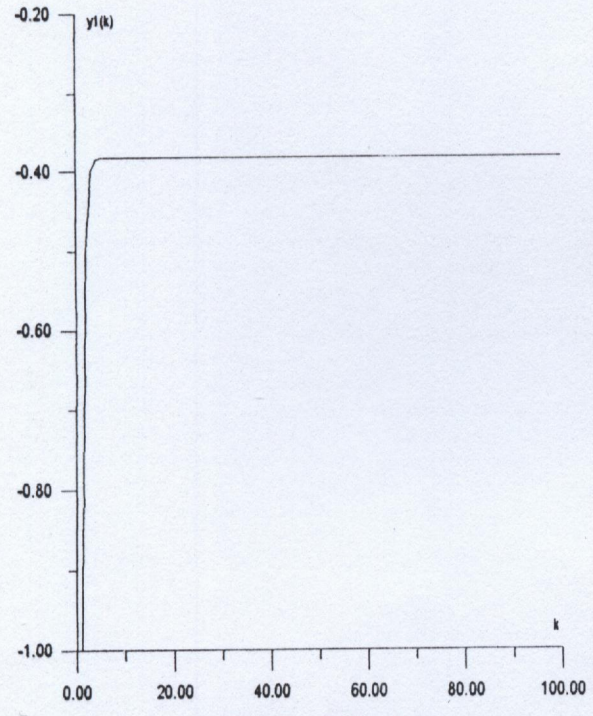
Les réseaux multicouches utilisés ont montré très clairement la rapidité des réseaux d'ordre 2 avec une taille réduite et un taux d'apprentissage plus grand par apport aux réseaux d'ordre 1.



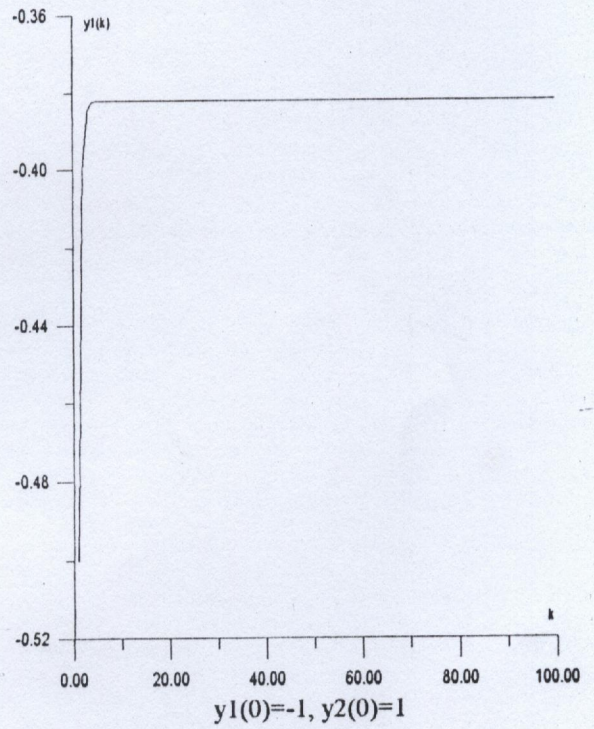
y1(0)=1,y2(0)=1



y1(0)=0,y2(0)=0

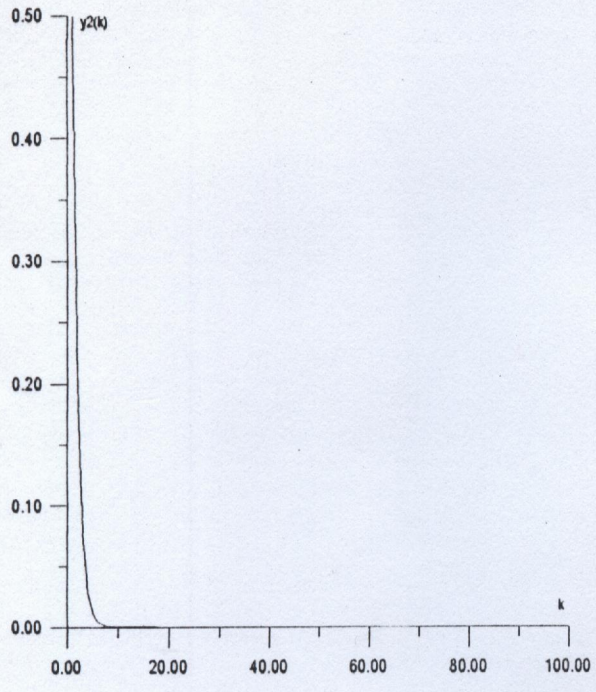


y1(0)=-2, y2(0)=-1

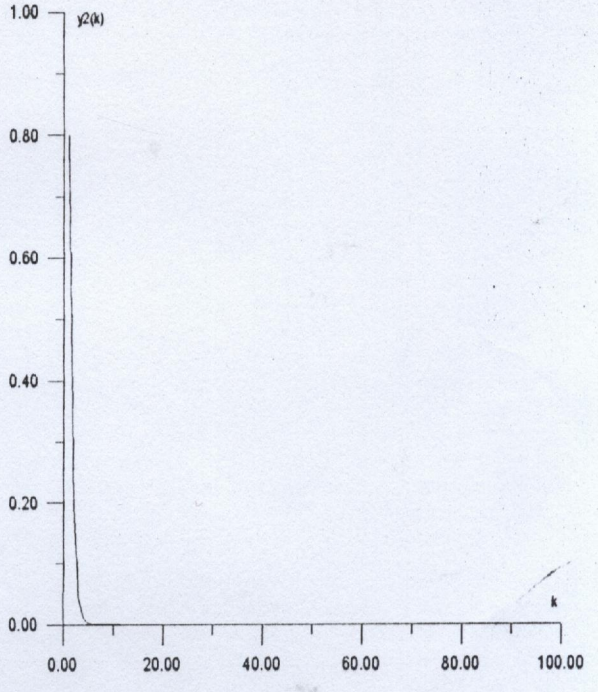


y1(0)=-1, y2(0)=1

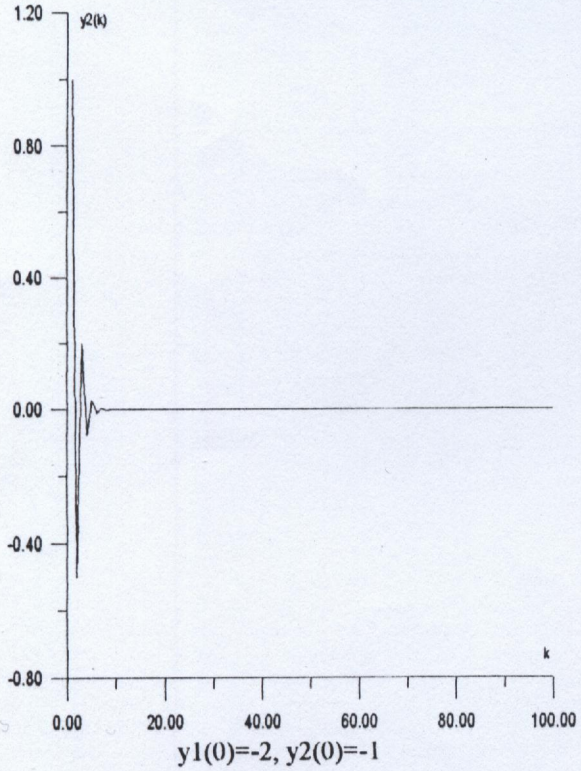
Figure (3.18) : *test de stabilité du système ($y_1(k)$) pour des entrées nuls.*



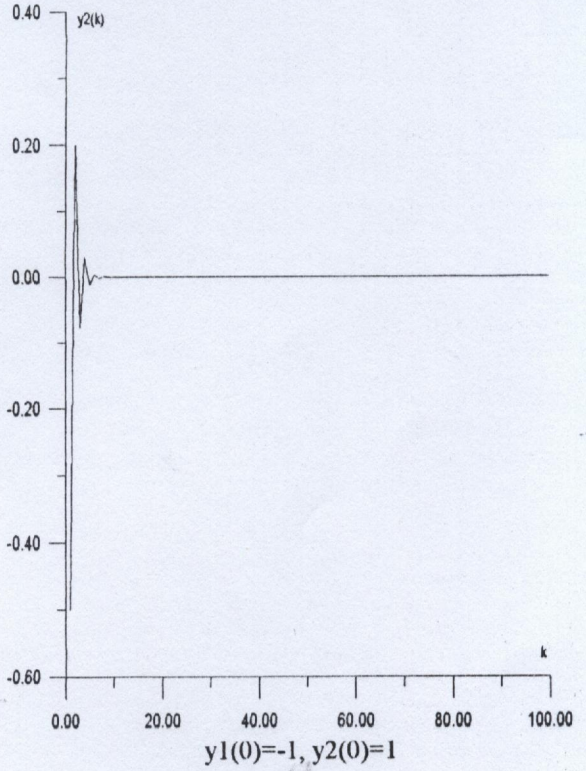
y1(0)=1,y2(0)=1



y1(0)=2,y2(0)=2

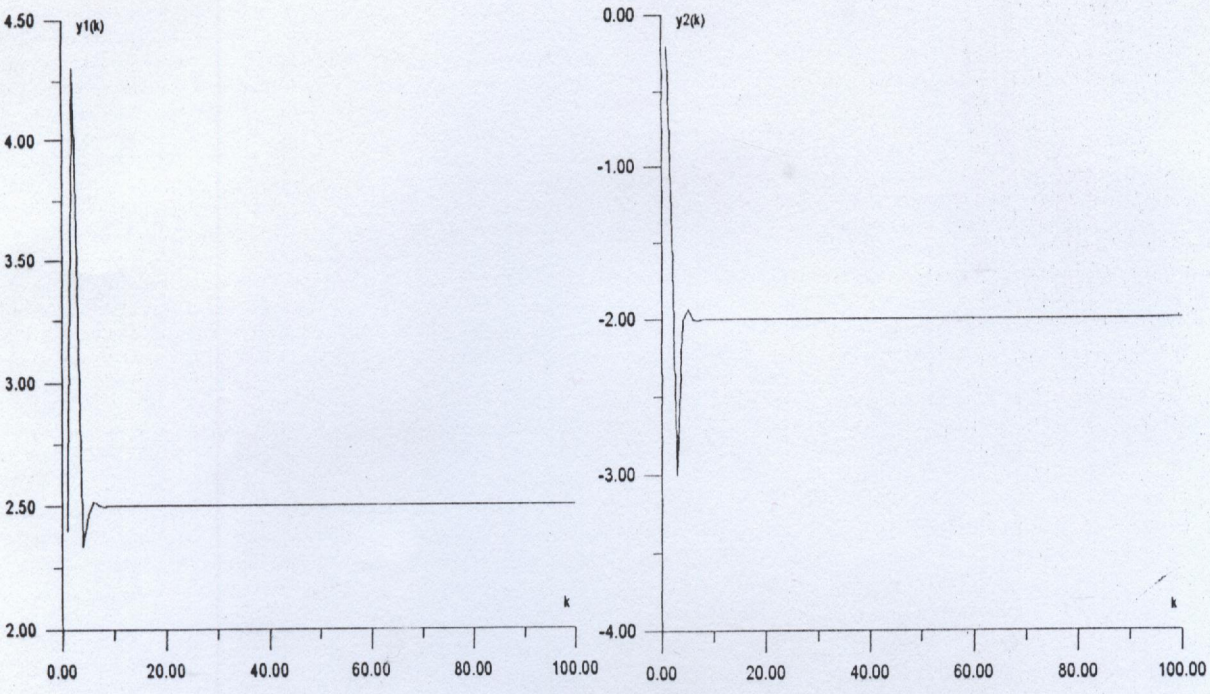


y1(0)=-2, y2(0)=-1

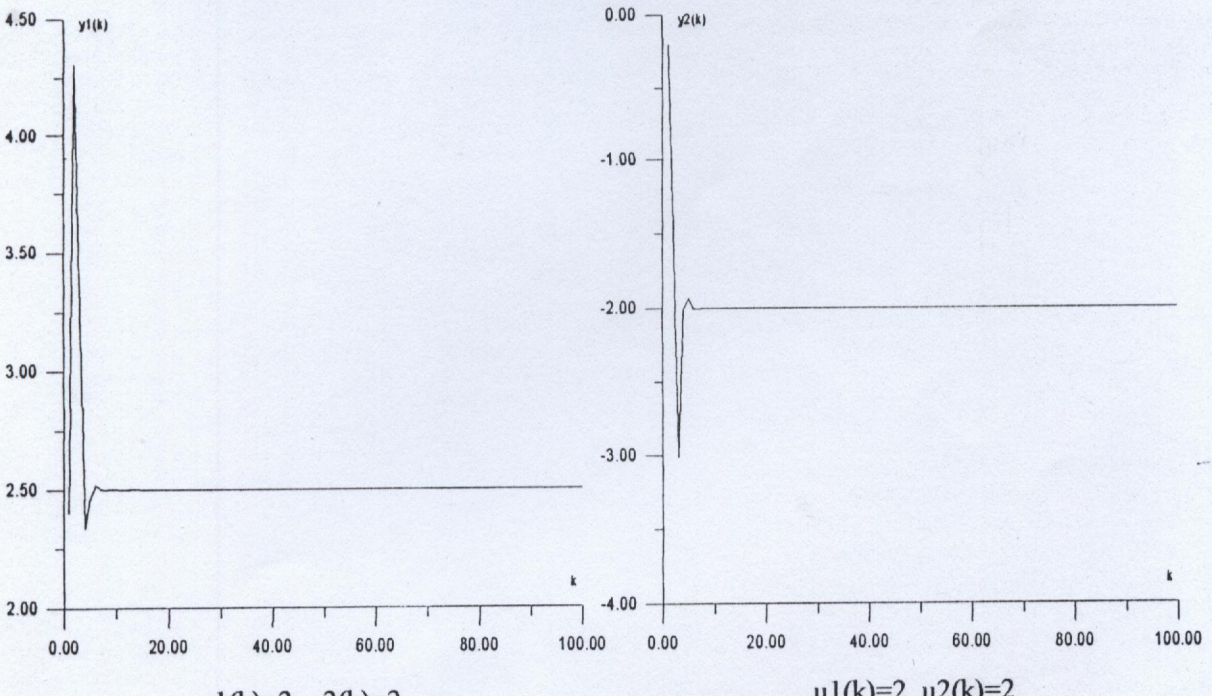


y1(0)=-1, y2(0)=1

Figure (3.19) : *test de stabilité du système ($y_2(k)$) pour des entrées nuls.*

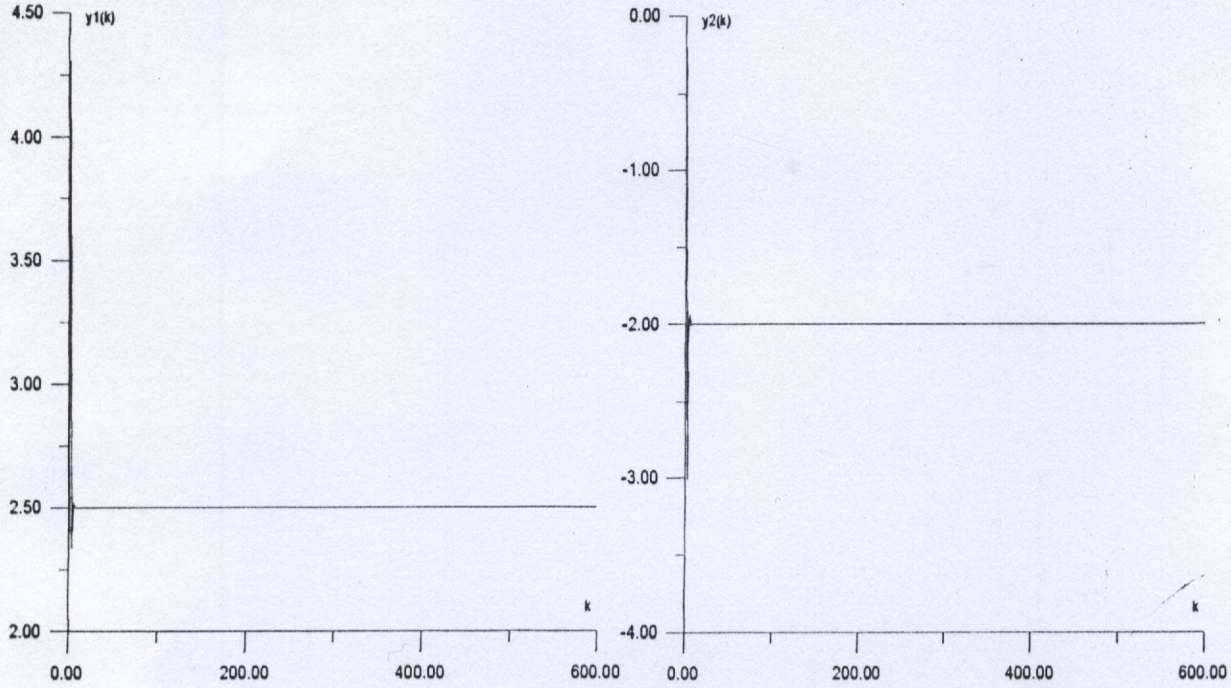


u1(k)=1, u2(k)=1 u1(k)=1, u2(k)=1



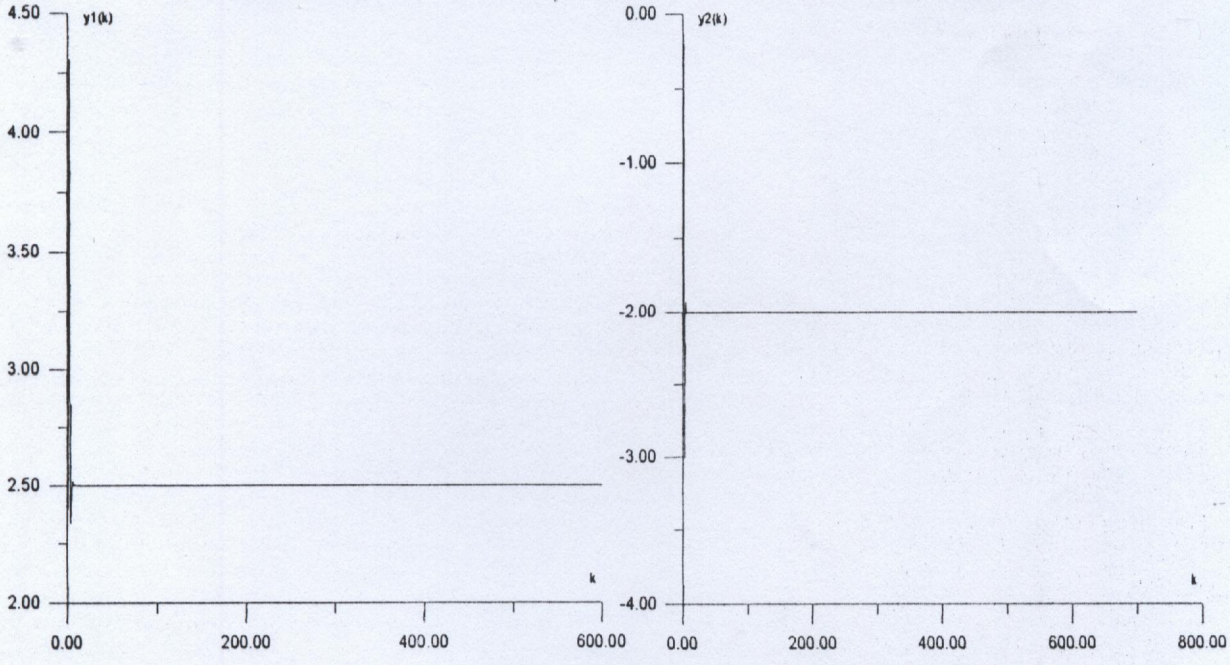
u1(k)=2, u2(k)=2 u1(k)=2, u2(k)=2

Figure (3.20) : *test de stabilité du système en présence des entrées bornées.*



u1(k)=-1, u2(k)=-2

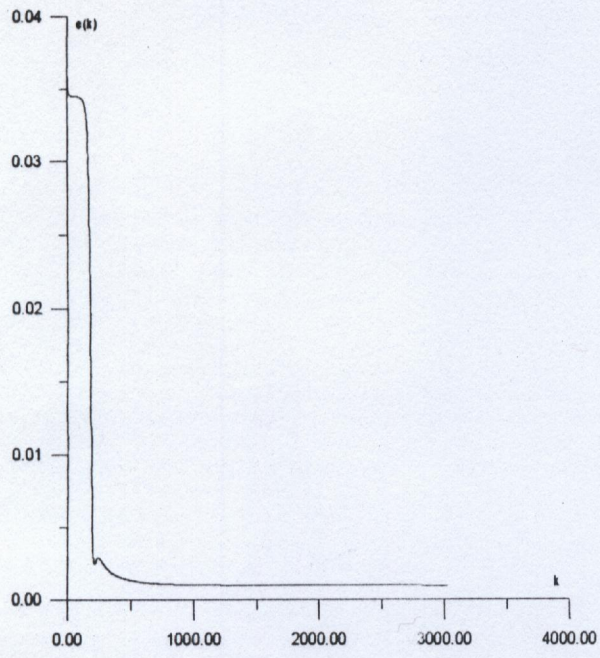
u1(k)=-1, u2(k)=-2



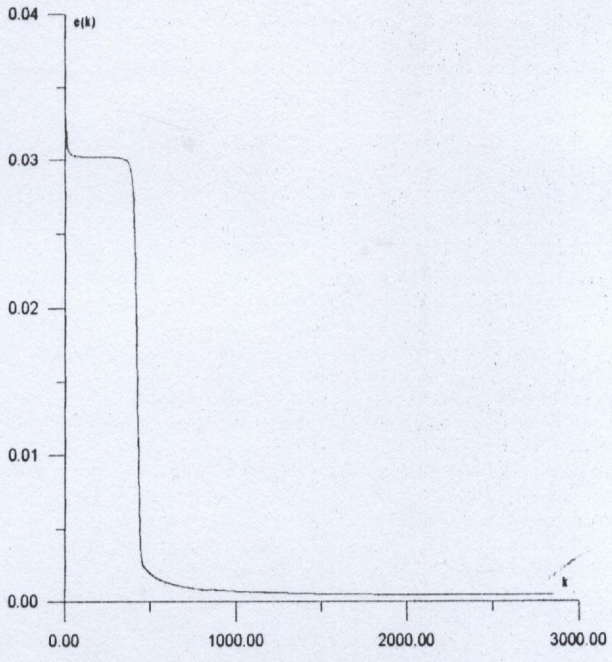
u1(k)=-1, u2(k)=2

u1(k)=-1, u2(k)=2

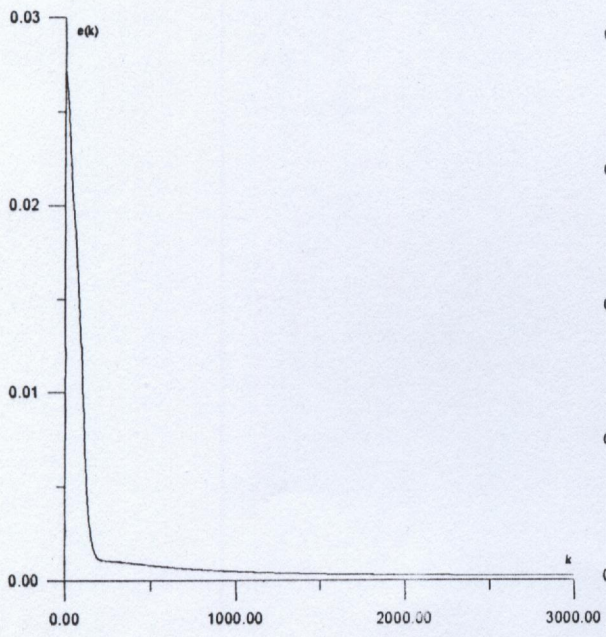
Figure (3.21) *: test de stabilité du système en présence des entrées bornées non nuls.*



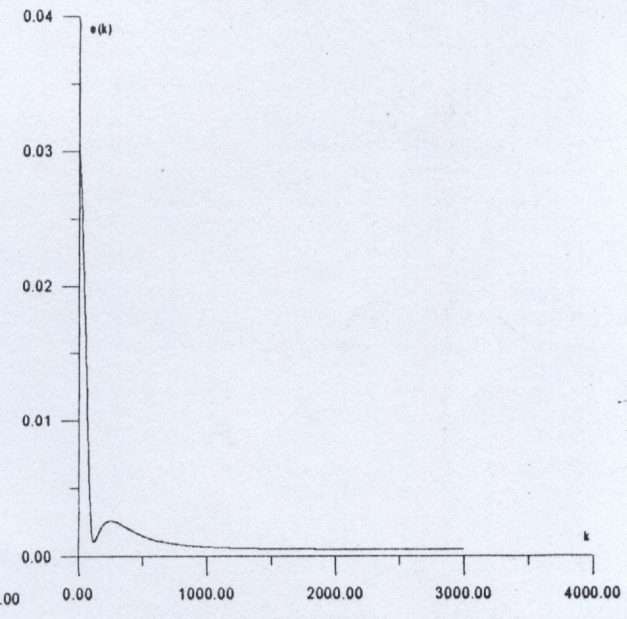
a) Erreur d'apprentissage pour un réseau modulaire



b) Erreur d'apprentissage pour un réseau modulaire d'ordre 2

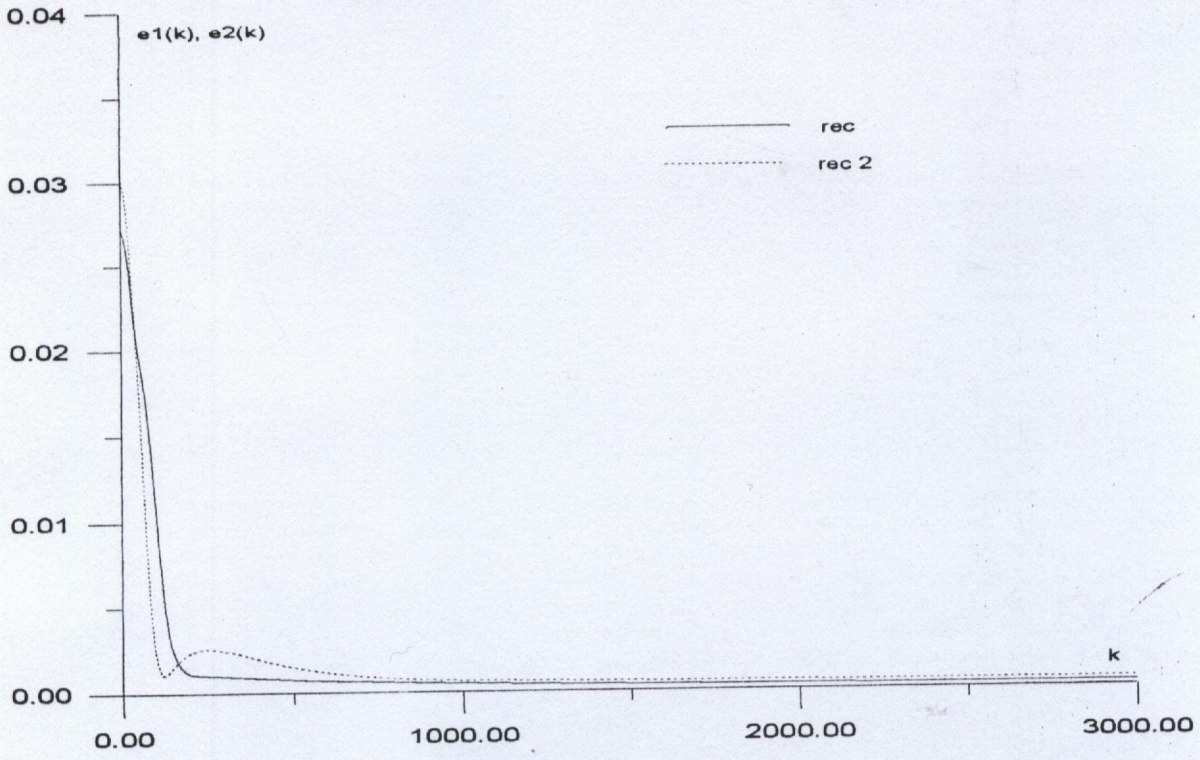


c) Erreur d'apprentissage pour un réseau récurrent

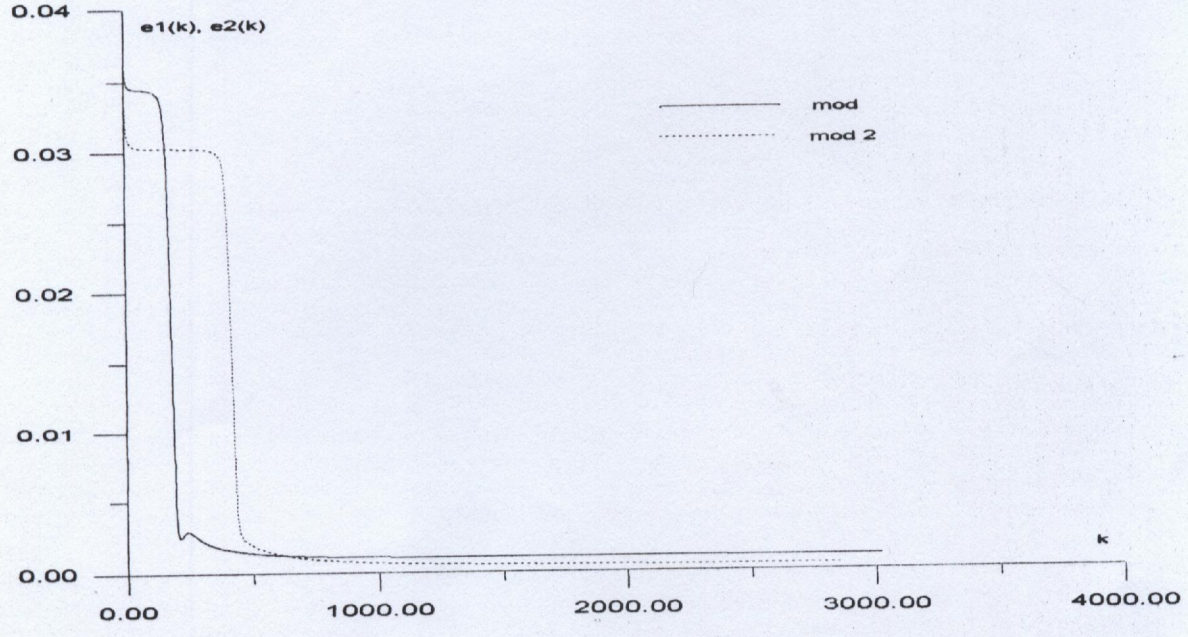


d) Erreur d'apprentissage pour un réseau récurrent d'ordre 2

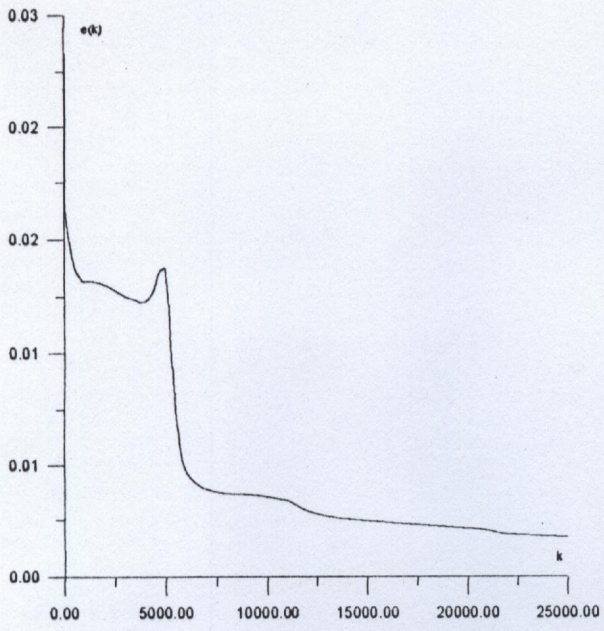
Figure (3.22) *: vitesse d'apprentissage des réseaux modulaires et récurrents pour* $y_1(k)$



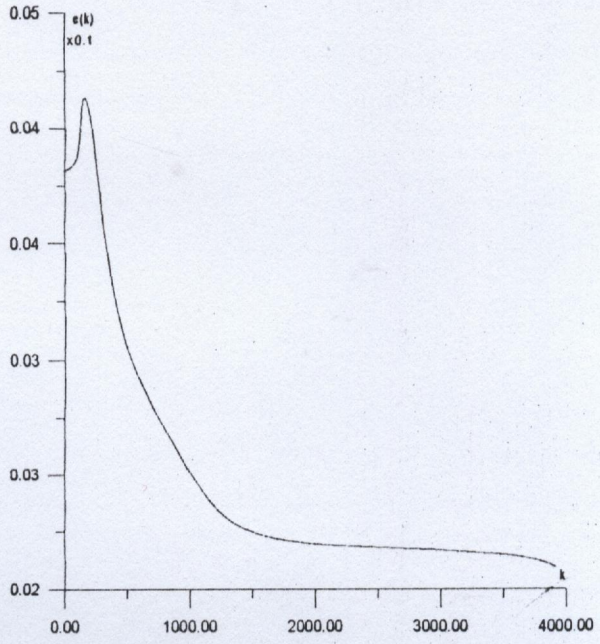
Figure(3.23) : *Comparaison entre la vitesse d'apprentissage des deux types du réseau récurrent*



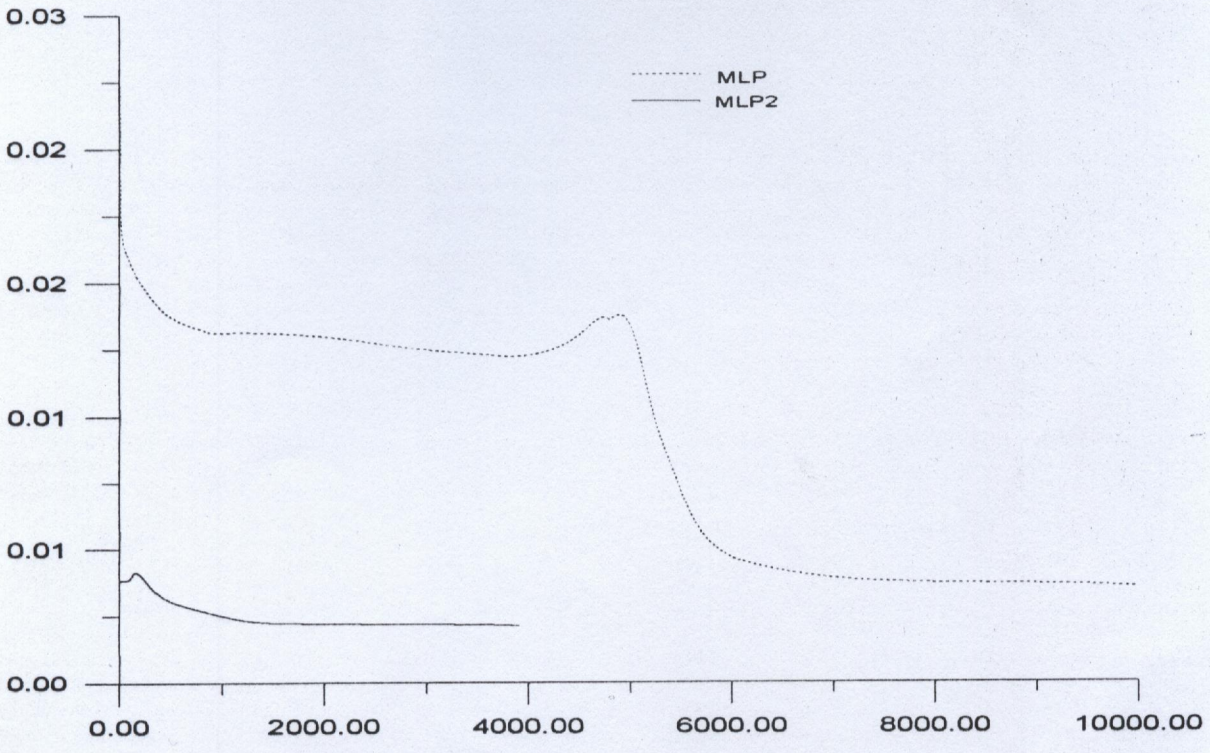
Figure(3.24) : *Comparaison entre la vitesse d'apprentissage des deux types du réseau modulaire.*



a)Erreur d'apprentissage pour un réseau M.L.P

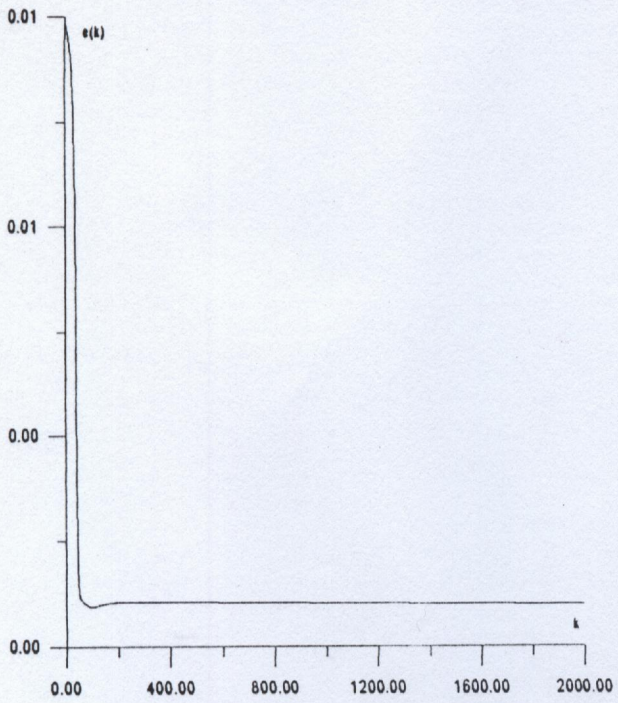


b) Erreur d'apprentissage pour un réseau M.L.P d'ordre 2

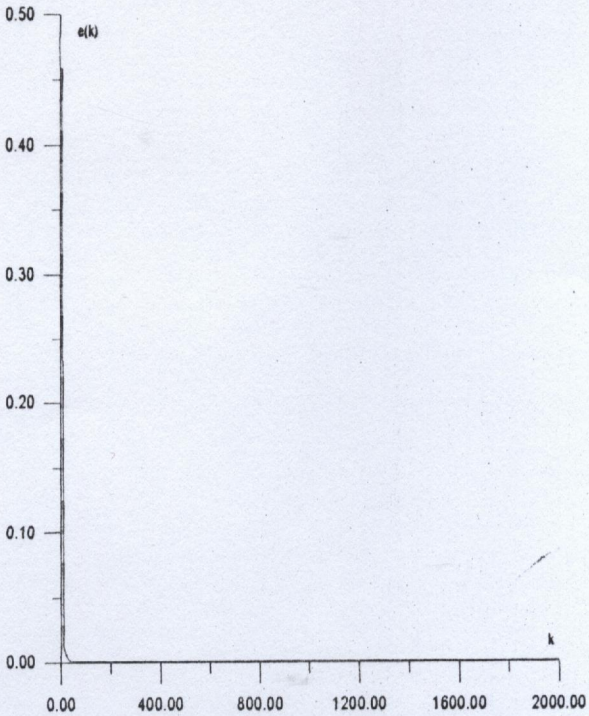


c) Comparaison entre la vitesse d'apprentissage des réseaux multicouches

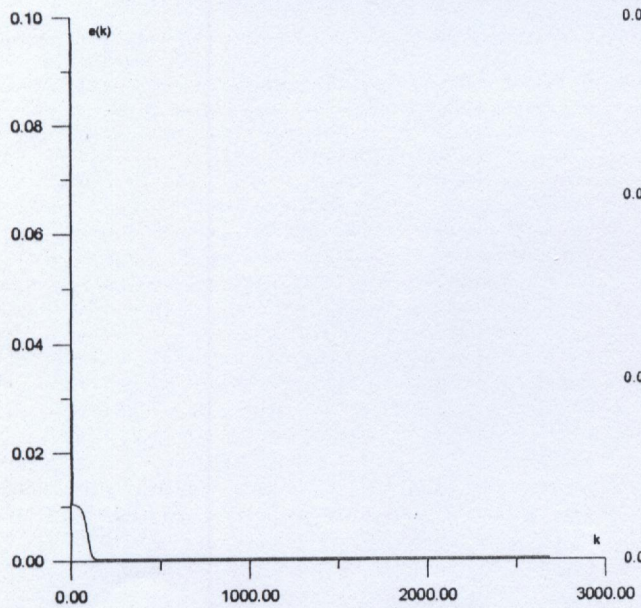
Figure(3.25) : *la vitesse d'apprentissage des réseaux multicouches pour la fonction $y_1(k)$.*



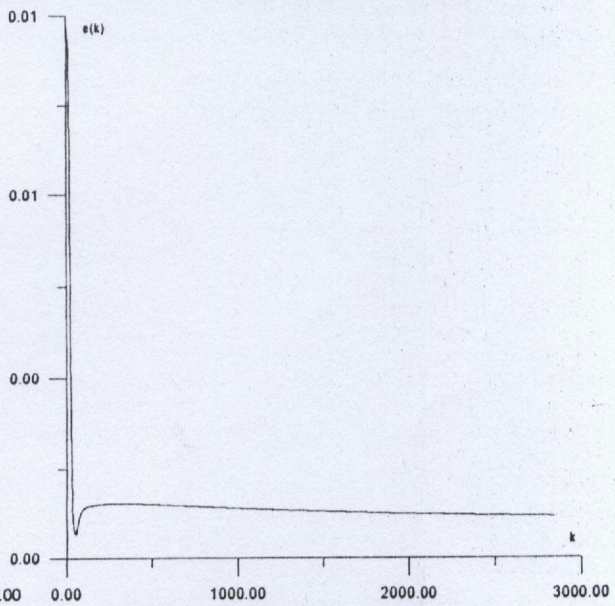
a) Erreur d'apprentissage pour un réseau modulaire



b) Erreur d'apprentissage pour un réseau modulaire d'ordre 2

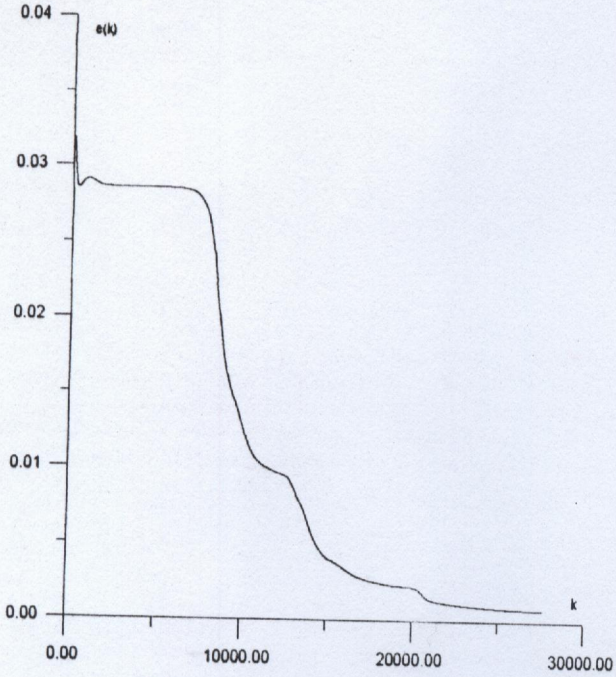


c)Erreur d'apprentissage pour un réseau récurrent

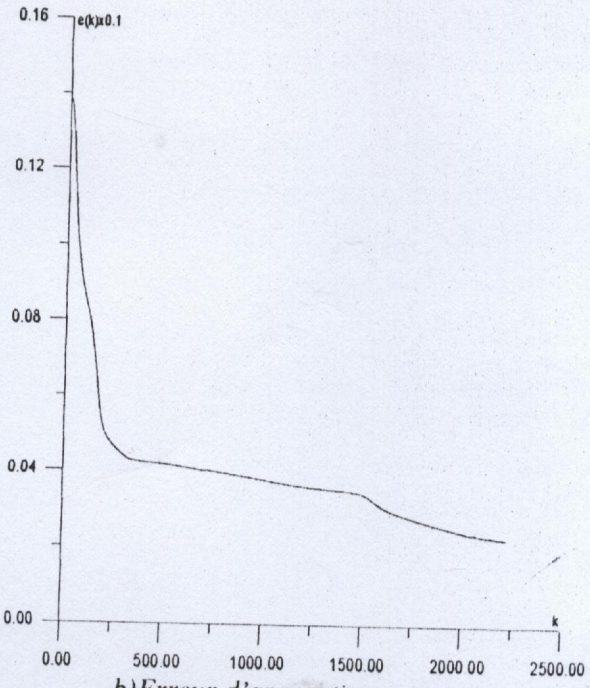


d) Erreur d'apprentissage pour un réseau récurrent d'ordre 2

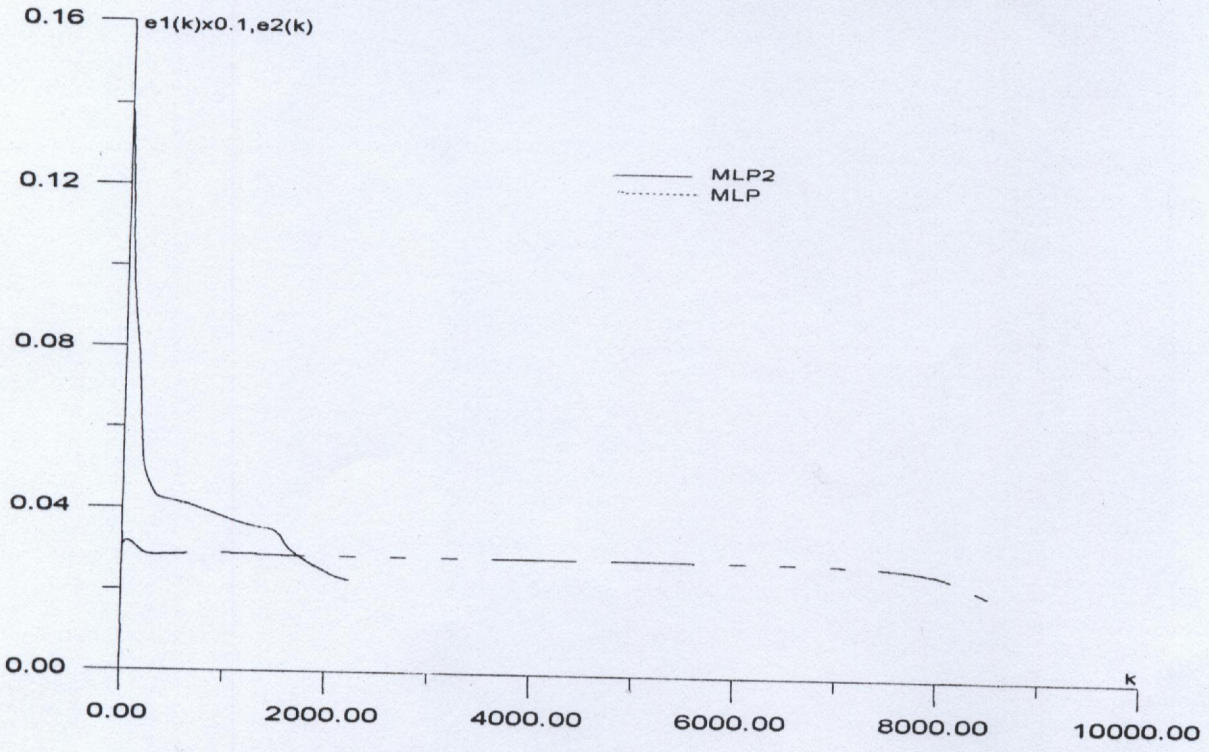
Figure (3.26) : *la vitesse d'apprentissage des réseaux modulaires et récurrents pour $y_2(k)$.*



a)Erreur d'apprentissage pour un réseau M.L.P

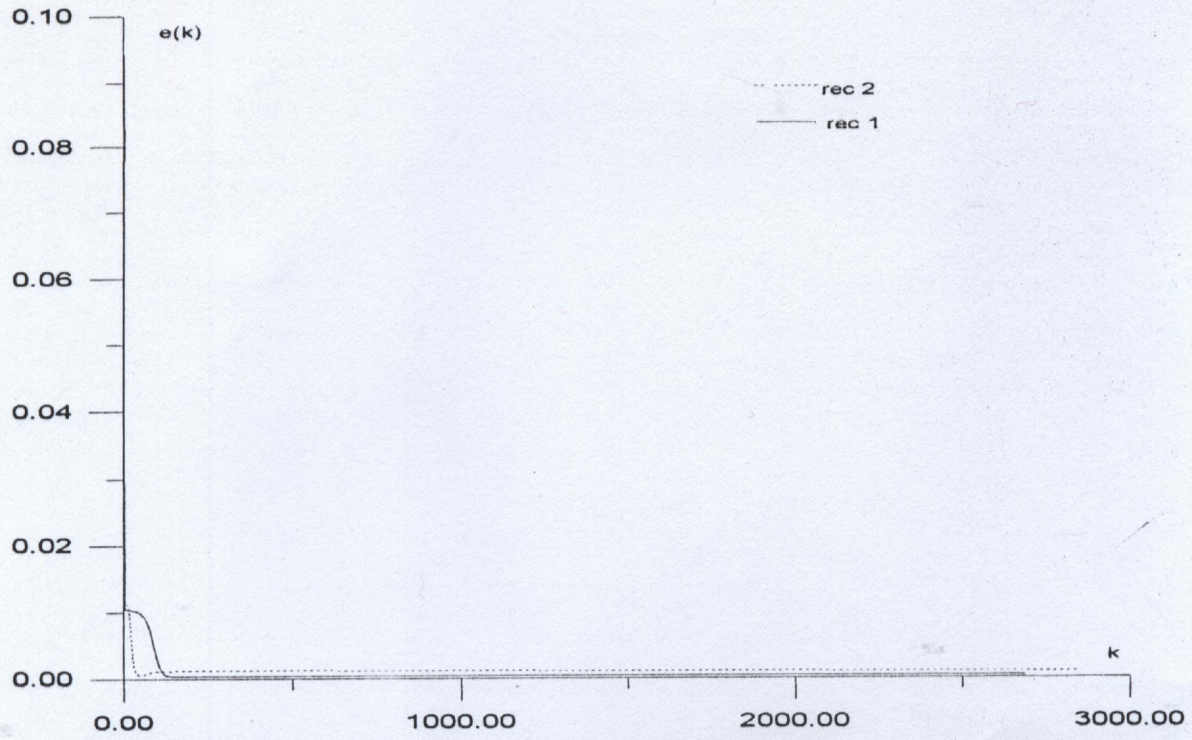


b)Erreur d'apprentissage pour un réseau M.L.P d'ordre 2

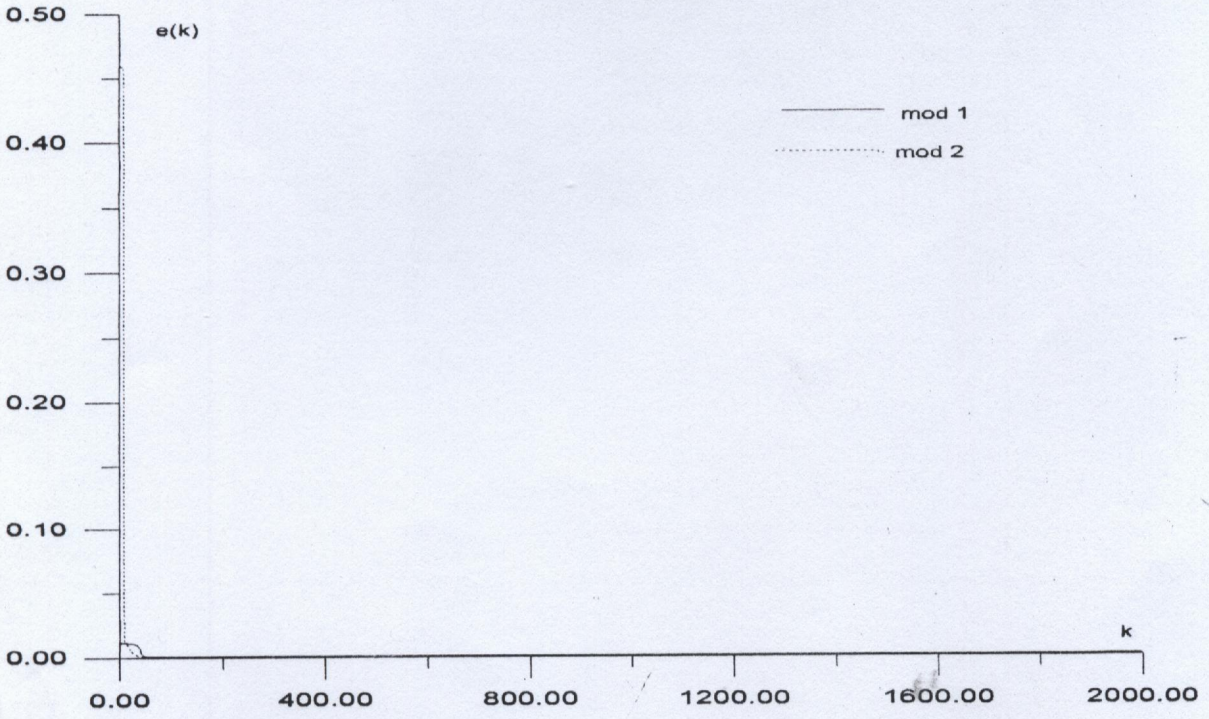


c)Comparaison entre la vitesse d'apprentissage des réseaux multicouches

Figure(3.27) : *la vitesse d'apprentissage des réseaux multicouches pour $y_2(k)$.*



Figure(3.28) : *Comparaison entre la vitesse d'apprentissage des deux types du réseau récurrent*



Figure(3.29) : *Comparaison entre la vitesse d'apprentissage des deux types du réseau modulaire.*

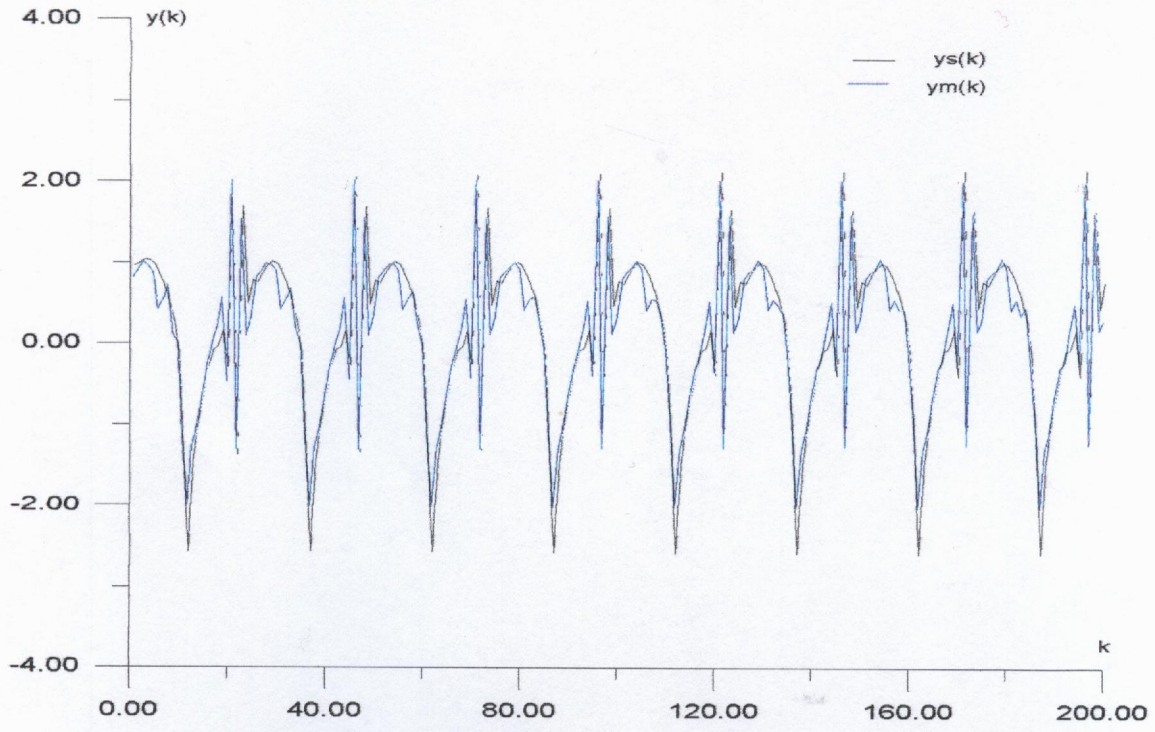


Figure (3.30) : *sortie du système et du modèle de $y_1(k)$ pour le vecteur des entrées*

$$u = \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} = \begin{bmatrix} sin(2\pi k/25) \\ cos(2\pi k/25) \end{bmatrix}$$

Type du Réseau	Taille du réseau	Pas d'apprentissage	Nombre de données	Nombre d'itérations
MLP	$N_{2\,10\,10\,1}$	0.1	100	25000
MLP(2)	$N_{2\,5\,5\,1}$	0.25	100	5600
Récurrent	7 neurones	0.001	100	3000
Récurrent (2)	4 neurones	0.001	100	3000
Modulaire	6 modules	0.001	100	3000
Modulaire (2)	3 modules	0.001	100	2000

Tableau(3.3) : *caractéristiques de la simulation de $y_1(k)$.*

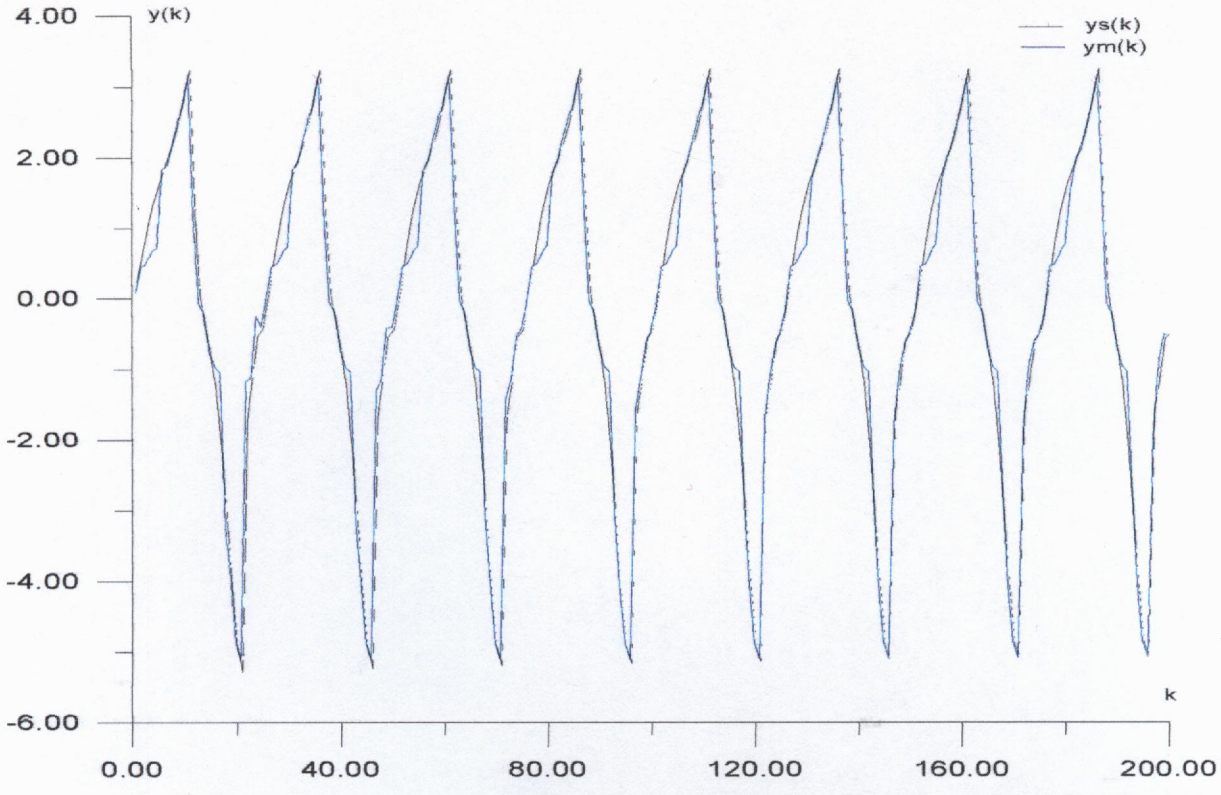


Figure (3.30) : *sortie du système et du modèle de $y_2(k)$ pour le vecteur des entrées*

$$u = \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} = \begin{bmatrix} sin(2\pi k/25) \\ cos(2\pi k/25) \end{bmatrix}$$

Type du Réseau	Taille du réseau	Pas d'apprentissage	Nombre de données	Nombre d'itérations
MLP	$N_{2\ 10\ 10\ 1}$	0.1	100	27000
MLP(2)	$N_{2\ 5\ 5\ 1}$	0.25	100	3000
Récurrent	7 neurones	0.001	100	2700
Récurrent (2)	4 neurones	0.001	100	2900
Modulaire	5 modules	0.001	100	2000
Modulaire (2)	3 modules	0.001	100	2000

Tableau(3.4) : *caractéristiques de la simulation de $y_2(k)$.*

V.3 CONCLUSION :

Les réseaux de neurones d'ordre élevé modulaires, récurrents ou multicouches, avec un choix adéquat de la taille du réseau et le taux d'apprentissage peuvent réaliser l'apprentissage plus rapidement que les réseaux de neurones d'ordre 1, mais le problème majeur est la détermination de la taille exacte pour une fonction non-linéaire quelconque. Des études récentes sont faites sur le réseau multicouche pour déterminer pour une fonction quelconque la taille de réseau (le nombre de couches et le nombre de neurones).

Les simulations ont montré aussi qu'avec ces réseaux, les systèmes MIMO tout à fait comme les systèmes SISO peuvent être identifiés.

Chapitre 04

IDENTIFICATION D'UNE APICOLE PAR LES RESEAUX DE NEURONES D'ORDRE ELEVE

IV.1 INTRODUCTION :

Les abeilles sociales amasseuses de miel existent depuis 10 à 20 millions d'années, bien avant l'apparition de l'homme. Les recherches archéologiques ont prouvé que, l'abeille était domestiquée 2400 avant J-C, par les égyptiens, pour l'exploitation des produits qu'elles élaborent (miel, cire,...). Les premiers égyptiens élevaient des abeilles, avec des ruches traditionnelles en terre cuite et faisaient déjà le commerce du miel et de la cire le long de la côte.

Depuis l'utilisation de la ruche en terre cuite, la recherche apicole s'est fortement développée sous les angles scientifiques et appliqués, principalement par les biologistes, les apiculteurs, et dernièrement par les mathématiciens, en essayant de résoudre les problèmes et d'augmenter la production.

Le manque en provisions de la ruche, dû à la diminution des entrées en pollen et en nectar dans les régions à miellée courte et espacées est le problème le plus connu par les apiculteurs. Dans ce cas, les apiculteurs se remettent à leurs expériences pour faire un nourrissement artificiel contenant des produits nutritifs (le candi ou les sirops de saccharoses) afin d'assurer la survie de la colonie, et car il reste difficile d'établir des normes de quantités et de types de nourriture à attribuer aux colonies ; parmi les problèmes engendrés par ce nourrissement on cite :

- *L'apparition des maladies du couvain (les loques européennes et américaines ou l'épuisement des provisions de pollen).*
- *La diminution de la taille de la population par l'essaimage naturel.*

Le contrôle d'un système complexe de la nature tel que la ruche apicole, sur une base scientifique est donc, très nécessaire. Les travaux de modélisation et de contrôle faits dans [28],[29],[30] ont conduit à de bons résultats .

L'objectif de ce travail est l'identification, afin de permettre à d'autres études de contrôler la productivité et le contrôle en approvisionnements.

Dans ce chapitre, nous commençons par présenter la ruche apicole, ensuite nous discutons les résultats obtenus dans [28], [29], [30] puis nous entamons l'identification de la ruche avec les réseaux de neurones récurrents et multicouches d'ordre 1 et 2 .avant de conclure

IV.2 DESCRIPTION GENERALE DE LA RUCHE APICOLE :

Dans le monde, ils existent plusieurs centaines de modèles de ruches à cadres mobiles plus ou moins répandus et plus ou moins brevetés.

Les ruches les plus répandues et les plus connues sont :

- La ruche *LONGSTROTH.*
- La ruche *DADANT-BLATTE.*
- La ruche *PERFECCION.*
- La ruche *PROKOPOVITCH.*

La caractéristique principale commune de ces ruches est qu'elles sont toutes à cadre mobile, ce qui facilite de façon importante l'intervention de l'homme.

La ruche est composée :

- Du corps de la ruche pour élevage de couvain, stockage des réserves de miel et de pollen.
- D'une ou plusieurs hausses pour la récolte du miel avec un nourrisseur au-dessus.
- D'un plateau couvre-cadre.

IV.3 DESCRIPTION DE L'AUTOMATICIEN :

Dans une ruche apicole, la température, le taux d'humidité, et le cycle biologique sont minutieusement contrôlés, ce qui rend la ruche, l'un des systèmes les plus automatisés dans la nature.

Dans notre étude, la ruche apicole est considérée comme une boite noire, placée dans un environnement spécifié, et contenant une colonie d'abeilles.

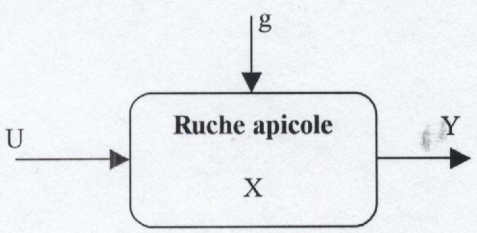


Figure (4.1) : *présentation de la ruche apicole comme système.*

U est le vecteur de commande.

Le système peut être contrôlé par le nourrisseur, en ajoutant une quantité de sirop (sucre dilué auquel sont ajoutées, des matières protoniques : pollen, farine de seigle ou de mais...etc), du candi (sucre cristallisé mélangé à du miel et de l'eau).

X est le vecteur de variables d'état :

L'état du système est défini par le nombre d'abeille, la quantité de miel, la quantité de pollen(sous forme de pain d'abeilles) et la quantité de cire (sous forme de cadres bâtis).

Y est le vecteur des sorties :

qui est constitué de miel naturel, de propolis, de pollen, de cire, de venin, de gelée royale,...etc.

g : est le vecteur des perturbations du système :

contenant le nectar et le pollen récolté par des beilles butineuses. Ce vecteur dépend de la population d'abeilles, de la saison, et de nombreux autres paramètres.

IV.4 MODELISATION D'UNE RUCHE APICOLE :

Pour obtenir un modèle simple et utile, des hypothèses simplificatrices ont été prises :

- Jeune reine sélectionnée.
- Traitement systématique de toutes les maladies.
- Abondance d'eau et de matières premières ''nectar et pollen'' au voisinage de la ruche.
- L'eau utilisée et la gelée royale produite par les abeilles ne sont pas modélisées.
- Climat tempéré.
- Existence de miellés multiples.
- Récolte de miel non entreprise.
- Modélisation du miel, de la cire, et du pollen uniquement.

Les mesures des quatre variables d'états x_i et les deux entrées g_1 et g_2 sont effectuées durant l'année 1992 à savoir :

- $x_1(k)$: Le nombre d'abeilles.
- $x_2(k)$: La quantité de miel.
- $x_3(k)$: La quantité de pollen.
- $x_4(k)$: La quantité de cire.
- $g_1(k)$: Entrée en nectar.
- $g_2(k)$: Entrée en pollen.

La procédure suivie pour avoir les données du tableau (4.1) est très approximative en supposant que :

Le nombre d'abeilles est estimé en considérant que chaque cadre contient 3000 abeilles.

en connaissant le nombre des cadre par ruche :

Le nombre d'abeilles = Nombre de cadre X 3000.

La quantité de miel est estimée de la façon suivante :

Quantité de miel = Nombre de cadres sans couvains X 02 kg

Quantité de miel = Nombre de cadres avec couvains X 01 kg

La quantité de pollen est estimée de la manière suivante :

Quantité de pollen = Nombre de cadres contenant du pain d'abeilles X 0.4 kg

Remarque :

Les mesures ont été prises tout les 10 jours durant une année, pour ne pas refroidir le couvain subitement et ainsi compromettre le développement de la colonie d'abeilles.

T	$x_1(t)$	$x_2(t)$	$x_3(t)$	$x_4(t)$	$g_1(t)$	$g_2(t)$
0	10	10	0.25	1	0.01	0.01
10	10	10	0.25	1	0.05	0.02
20	10	10	0.25	1	0.10	0.04
30	13	10	0.25	1	0.15	0.06
40	20	10	0.5	1	0.25	0.09
50	26	10	0.75	1	0.35	0.14
60	36	10	1.25	1	0.50	0.20
70	46	13	2.25	1.25	0.70	0.30
80	56	17	3.25	1.50	1.05	0.30
90	73	20	5	2.25	1.50	0.30
100	100	25	5	3	1.50	0.30
110	100	32	5	3	1.50	0.30
120	100	50	5	3	1.50	0.30
130	100	50	5	3	1.05	0.20
140	100	40	4	3	0.65	0.12
150	93	30	3	3	0.20	0.02
160	75	26.5	2.75	3	0.10	0.01
170	60	25	2.60	3	0.40	0.06
180	50	25	2.50	3	0.85	0.15
190	55	33.5	2.80	3.08	1.55	0.28
200	67	47.5	3.20	3.15	2.00	0.19
210	80	60	3.50	3.25	1.15	0.05
220	80	60	3.50	3.25	0.13	0.02
230	70	53.5	2.85	3.25	0.07	0.02
240	59	46	2.25	3.25	0.05	0.01
250	47	37	1.85	3.25	0.04	0.01
260	42	28.5	1.50	3.25	0.04	0.01
270	37	21.5	1.12	3.25	0.03	0.01
280	31	15	0.80	3.25	0.03	0.01
290	26	11	0.55	3.25	0.03	0.01
300	20	7	0.45	3.25	0.03	0.01
310	18.3	7.5	0.40	3.25	0.03	0.01
320	16.7	6.5	0.38	3.25	0.03	0.01
330	15	5.5	0.35	3.25	0.03	0.01
340	13.3	5.3	0.30	3.25	0.03	0.01
350	11.7	5	0.28	3.25	0.03	0.01
360	10	5	0.25	3.25	0.03	0.01
jours	Milliers	kg	kg	kg	Kg/jour	Kg/jour

***Tableau 4.1** : Les données expérimentales d'une ruche apicole de l'année 1992*

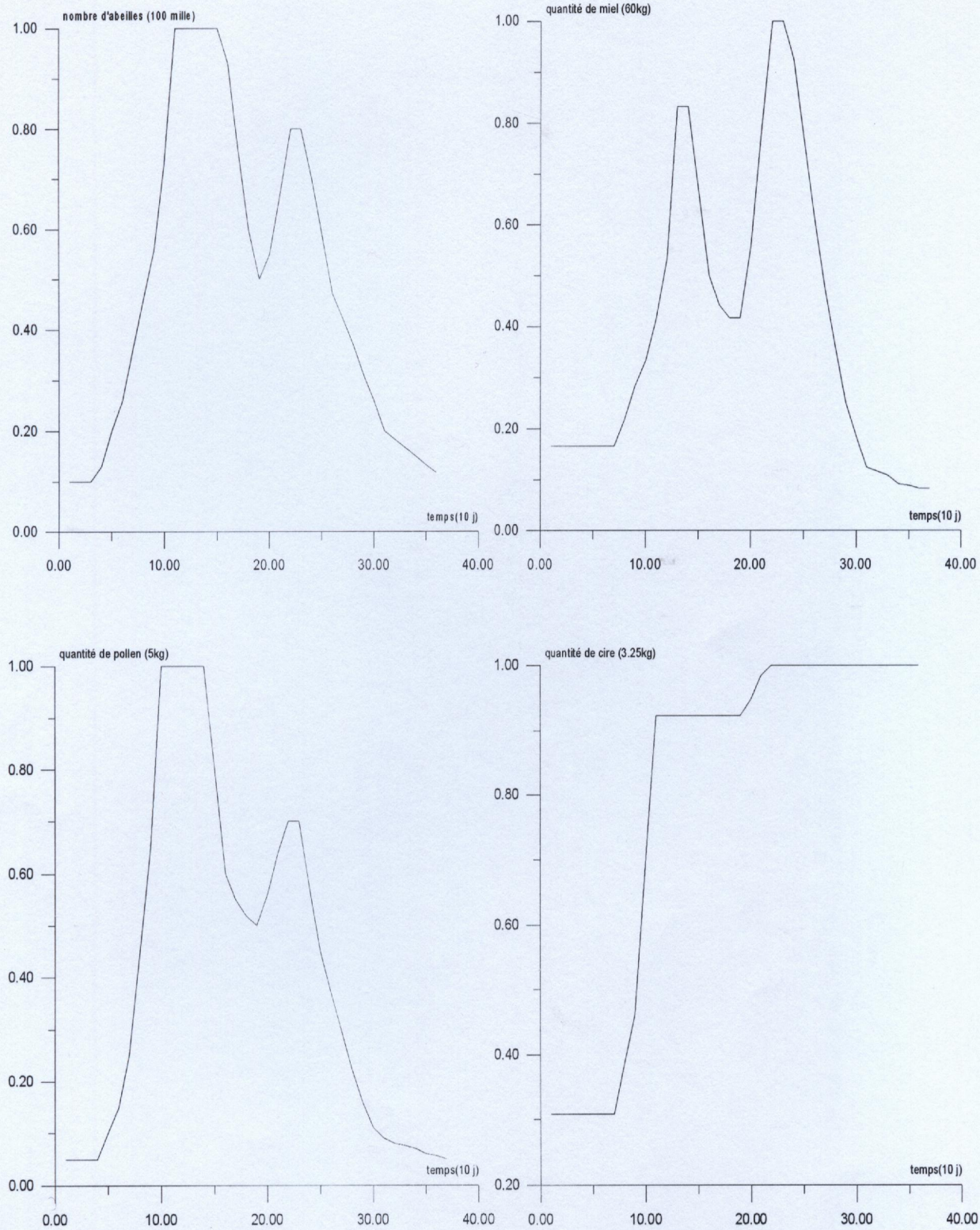


Figure (4.2) : *les variations des données expérimentales*

IV.5 IDENTIFICATION D'UNE RUCHE APICOLE PAR ADOPTION DES MODELES PARAMETRIQUES

L'identification de la ruche apicole est réalisée dans **[29]**, **[30]** en utilisant le modèle linéaire, le modèle général bilinéaire, et le modèle *Lotka-Volterra.* Les annexes **I**, **II**, **III** montrent la superposition des résultats obtenus par les trois modèles et les données expérimentales.

IV.5.1 IDENTIFICATION PAR ADOPTION DU MODELE BILINEAIRE :

Dans le cas continu, le modèle général bilinéaire a la forme suivante :

$$\begin{aligned}
\dot{x}_1(t) &= \sum_{j=1}^{4} x_j(t)\left[a_{1j} + \sum_{L=j}^{4} b_{jL}x_L(t)\right] \\
\dot{x}_2(t) &= \sum_{j=1}^{4} x_j(t)\left[a_{2j} + \sum_{L=j}^{4} c_{jL}x_L(t)\right] + \beta_1 g_1 \\
\dot{x}_3(t) &= \sum_{j=1}^{4} x_j(t)\left[a_{3j} + \sum_{L=j}^{4} d_{jL}x_L(t)\right] + \beta_2 g_2 \\
\dot{x}_4(t) &= \sum_{j=1}^{4} x_j(t)\left[a_{4j} + \sum_{L=j}^{4} e_{jL}x_L(t)\right]
\end{aligned} \qquad (4.1)$$

Il est à noter, que la méthode des moindres carrés est utilisée pour estimer les paramètres du modèle général bilinéaire ; La détermination de ces paramètres est équivalente à la résolution d'un système linéaire.

IV.5.2 IDENTIFICATION PAR ADOPTION DU MODELE LOTKA-VOLTERRA :

Ce modèle est un cas particulier du modèle bilinéaire, car quelques coefficients sont supposés nuls. Le modèle est représenté par un système d'équations différentielles bilinéaire, et se formule de la manière suivante:

$$\dot{x}_1(t) = \sum_{j=1}^{4} x_j(t)\left[a_{1j} + \sum_{L=j}^{4} b_{jL}x_L(t)\right] \qquad (4.2)$$

avec $b_{il}=0$, $i=1,2,3,4$

IV .5.3 IDENTIFICATION PAR ADOPTION DU MODELE LINEAIRE :

C'est le modèle bilinéaire mais avec la supposition que tous les coefficients bilinéaires sont nuls dans ce cas, le modèle est représenté par :

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

IV.6 DISCUSSIONS DES RESULTATS OBTENUS :

Les travaux effectués dans **[29]** et **[30]** présentent quelques inconvénients qui peuvent affecter la précisions des résultats obtenus.

- Les paramètres des modèles mathématiques utilisés pour identifier la ruche apicole sont estimés par la méthode des moindres carrés qui ne donne un estimateur non biaisé que dans le cas de l'affectation de la sortie du système par un bruit blanc. En pratique le bruit blanc n'existe pas (énergie infinie), donc les résultas de cette méthode ne sont pas toujours bonnes. L'imprécision des modèles mathématiques utilisés est due aussi à l'insuffisance des données utilisées durant l'identification (36 données).
- En pratique, des variations internes et externes-[22]
- .
- -*-10 peuvent affecter le système ; par conséquent les paramètres de ces modèles peuvent changer dans le temps d'où le risque d'utiliser des paramètres différents par les lois de commande.
- La linéarisation des modèles mathématiques fournit des conditions initiales ou finales imprécises.

IV.7 IDENTIFICATION D'E/S D'UNE RUCHE APICOLE PAR LES RESEAUX DE NEURONES D'ORDRE ELEVE :

Le système de la ruche apicole à identifier est représenté par les équations discrètes suivantes :

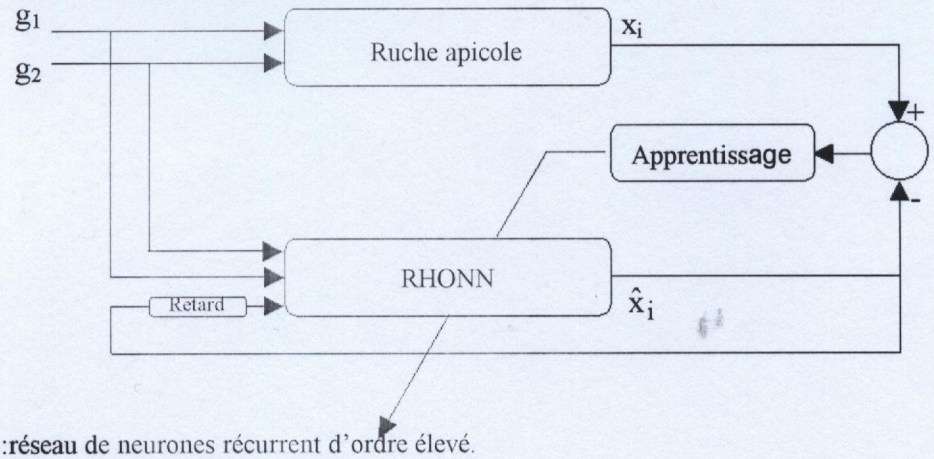
$$\begin{aligned} x_1(k+1) &= f_1(X(k)) \\ x_2(k+1) &= f_2(X(k)) + g_1(k) \\ x_3(k+1) &= f_3(X(k)) + g_2(k) \\ x_4(k+1) &= f_4(X(k)) \end{aligned}$$

le modèle d'identification par les réseaux de neurones est le suivant :

$$\begin{aligned} \hat{x}_1(k+1) &= N_1(X(k)) \\ \hat{x}_2(k+1) &= N_2(X(k)) + g_1(k) \\ \hat{x}_3(k+1) &= N_3(X(k)) + g_2(k) \\ \hat{x}_4(k+1) &= N_4(X(k)) \end{aligned}$$

N_i est un réseau de neurones récurrent ou à multicouches.

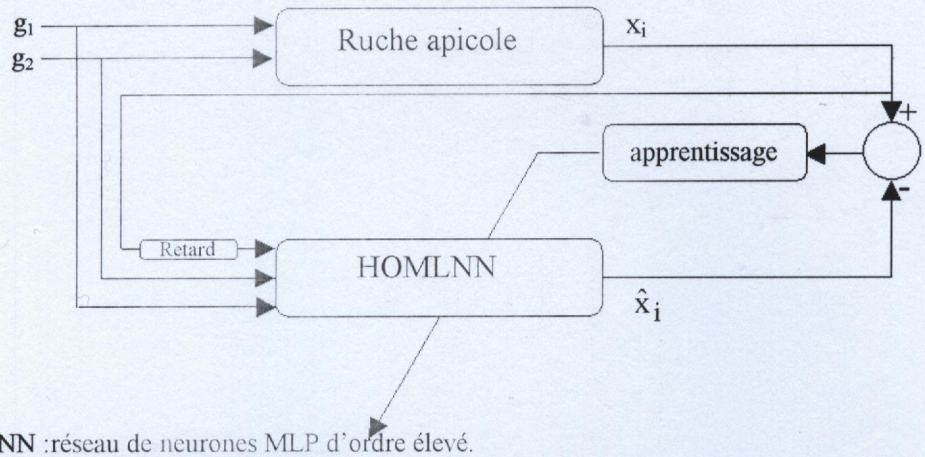
Le schéma d'identification par un réseau récurrent est :



HORNN :réseau de neurones récurrent d'ordre élevé.

Figure (4.3) : *identification de la ruche apicole par HORNN*

Le schéma d'identification par les réseaux MLP est le suivant :



HOMLNN :réseau de neurones MLP d'ordre élevé.

Figure (4.4) : *identification de la ruche apicole par HOMLNN*

Pour identifier la ruche apicole nous utilisons les réseaux de neurones multicouches d'ordre 1 et 2 et les réseaux récurrents d'ordre 1 et 2. La superposition des données expérimentales utilisées et les sorties des réseaux de neurones obtenues après l'apprentissage est illustrée par la figure (4.5).
En comparant ces résultats avec ceux des annexes I, II, III, on remarque que l'écart entre les sorties du système et du modèle est négligeable si on utilise pour identifier la ruche apicole les réseaux de neurones d'ordre élevé.
Les figures (4.6) jusqu'à (4.13) montrent aussi que l'apprentissage est fait plus rapidement par les réseaux de neurones d'ordre deux ; on remarque cependant que la vitesse d'apprentissage reste lente vu que l'on utilise peu de données pour l'apprentissage.
On remarque l'existence des points de divergence continus ou discrets dans chaque graphes d'apprentissage malgré le taux d'apprentissage utilisé (0.001 pour les réseaux récurrents et 0.1 pour les réseaux multicouches) qui est très petit. Il faut donc mieux choisir la taille du réseau et le taux d'apprentissage.
Les tableaux comparatifs (4.2) jusqu'à (4.5) donnent les caractéristiques de chaque simulation avec le nombre d'itérations nécessaire pour faire l'apprentissage par les réseaux .

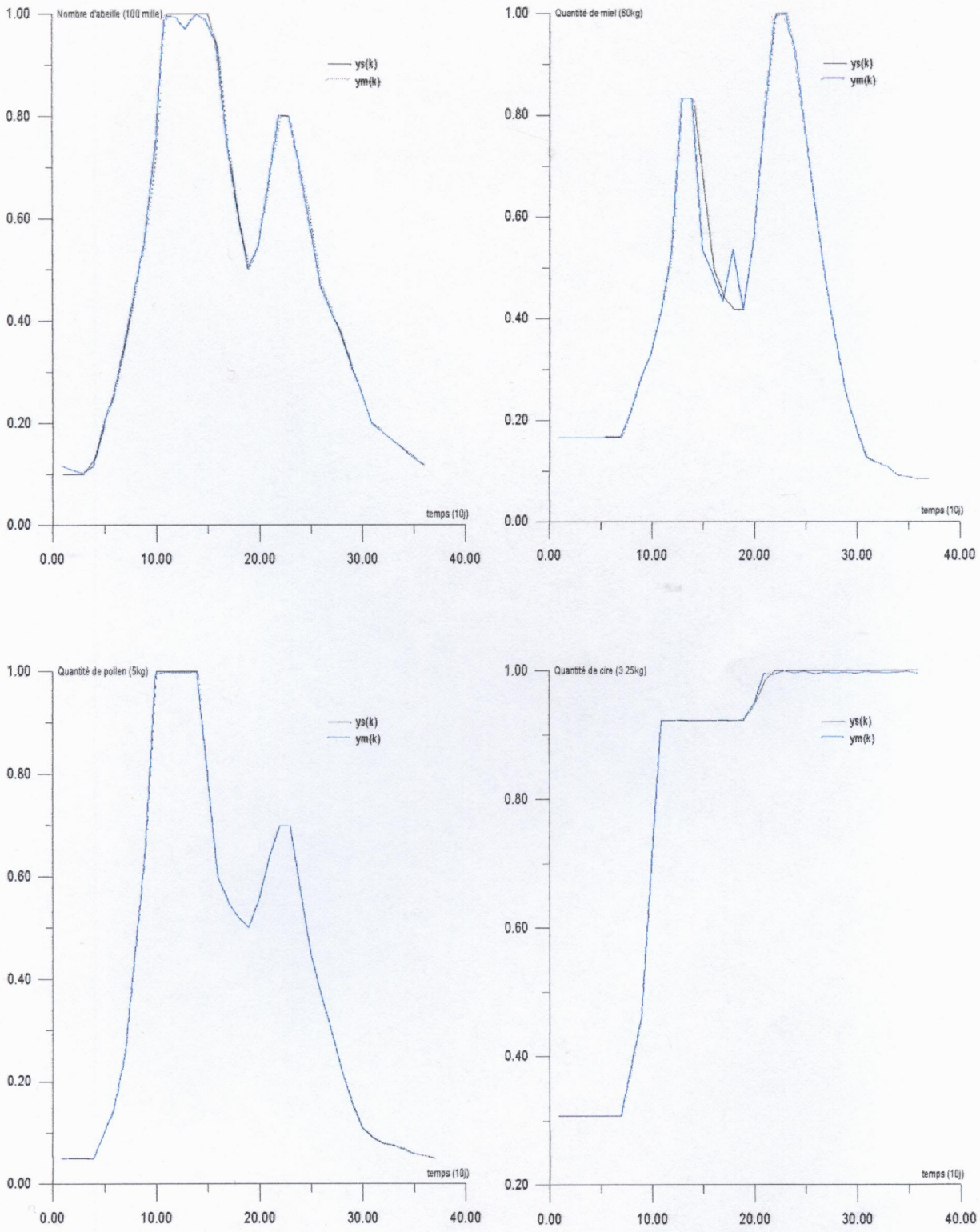
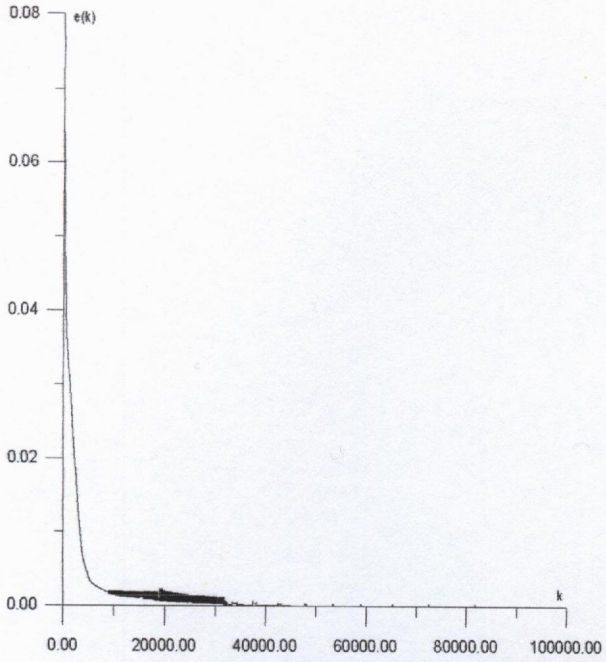
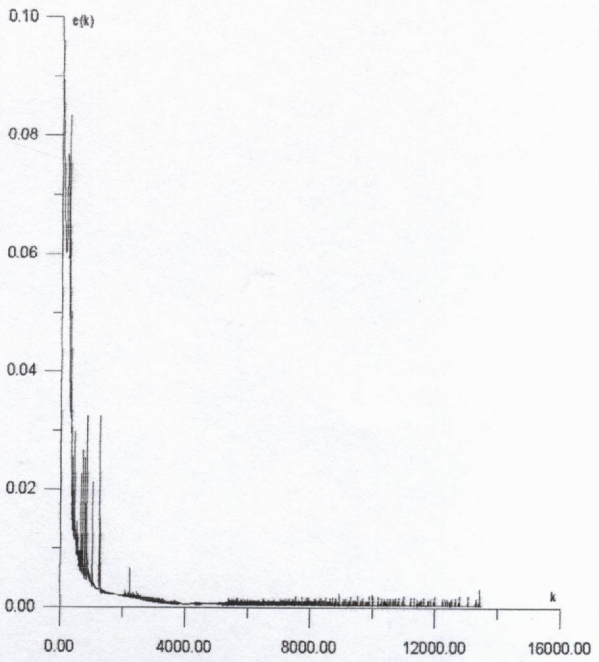


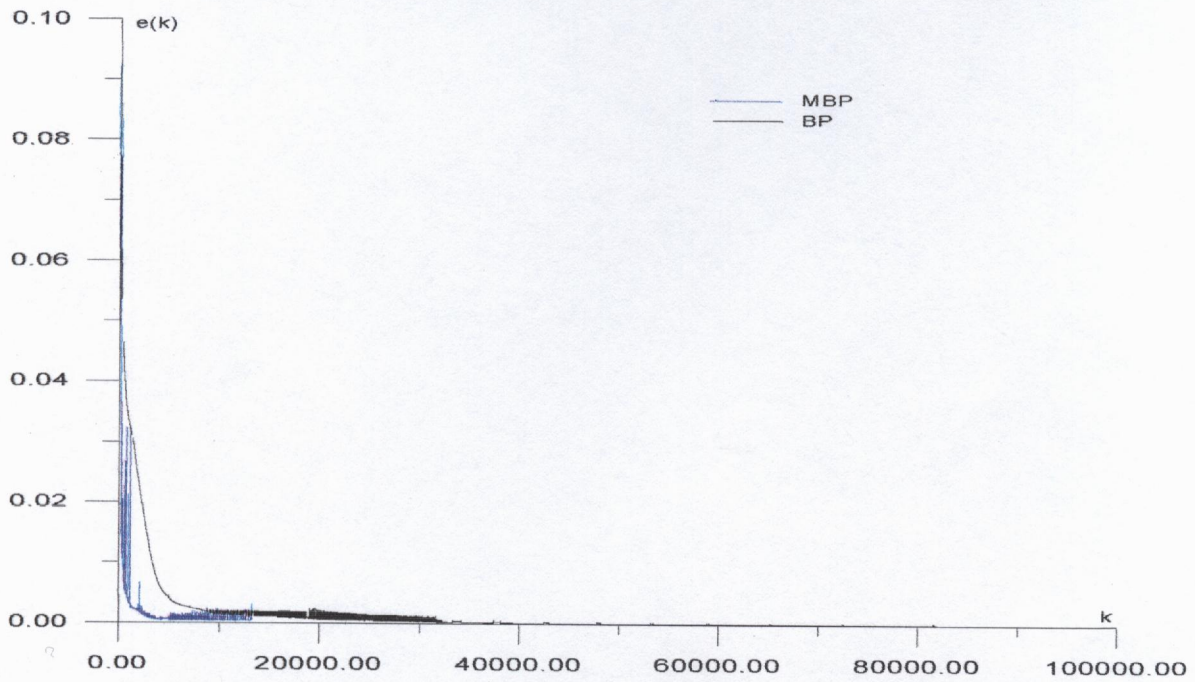
Figure (4.5) *:superposition des résultats des modèles neuronaux et les données expérimentales*



a) Erreur d'apprentissage du réseau MLP utilisé pour identifier x_1

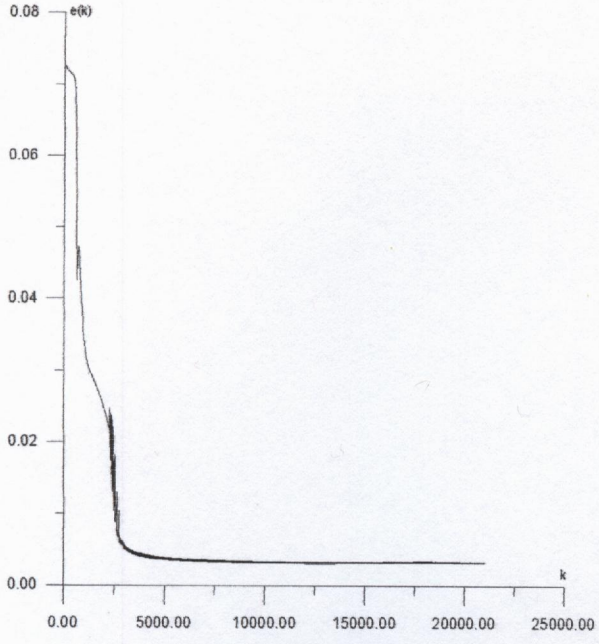


b) Erreur d'apprentissage du réseau MLP d'ordre 2 utilisé pour identifier x_1

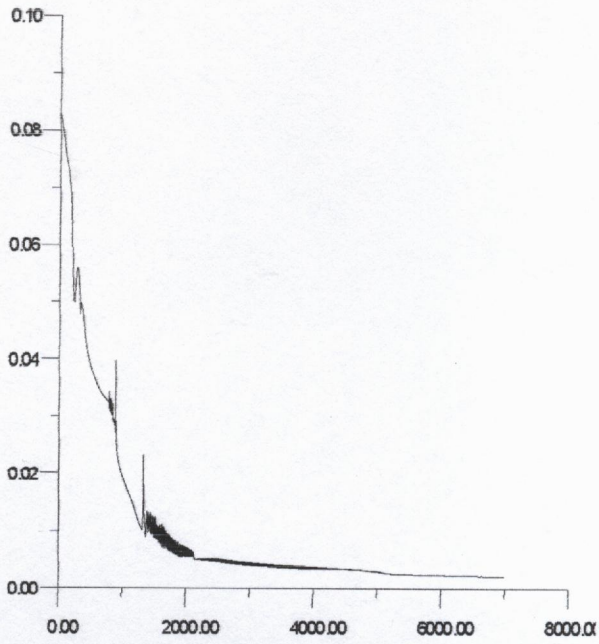


c) comparaison de la vitesse d'apprentissage des réseaux MLP

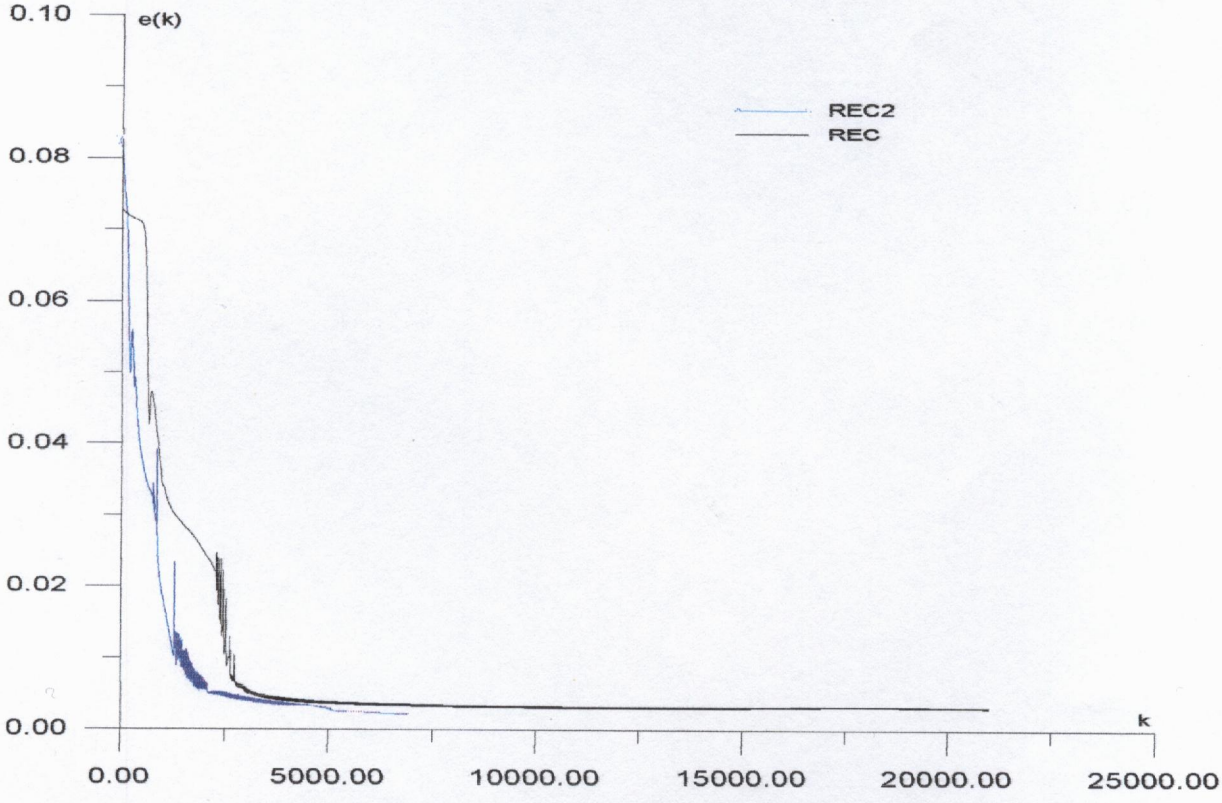
Figure (4.6) : *apprentissage de x_1 par des réseaux multicouches*



a) Erreur d'apprentissage du réseau récurrent utilisé pour identifier x_1

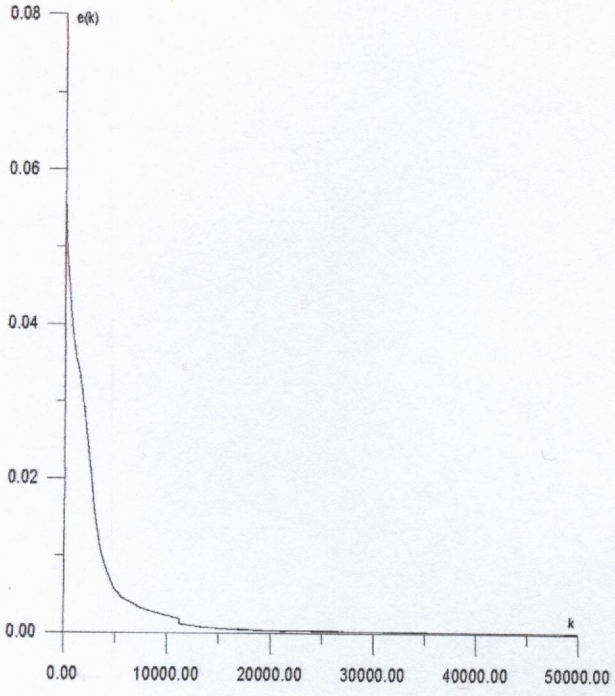


b) Erreur d'apprentissage du réseau récurrent d'ordre 2 utilisé pour identifier x_1

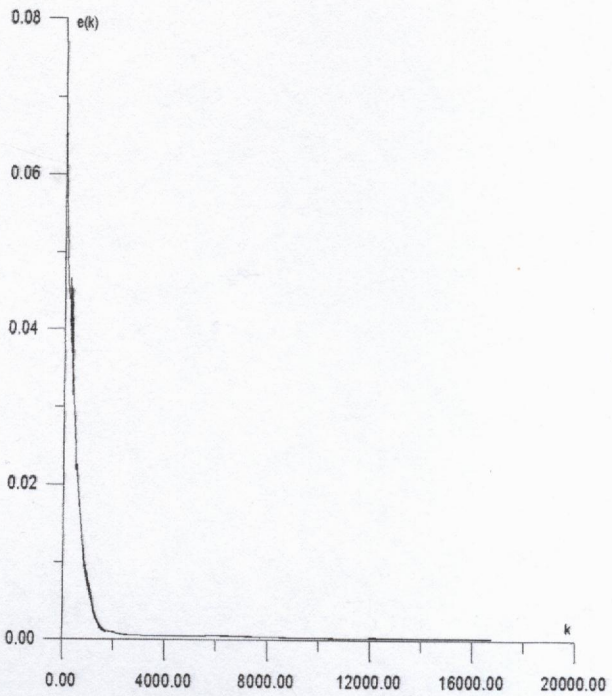


c) comparaison de la vitesse d'apprentissage des réseaux récurrents

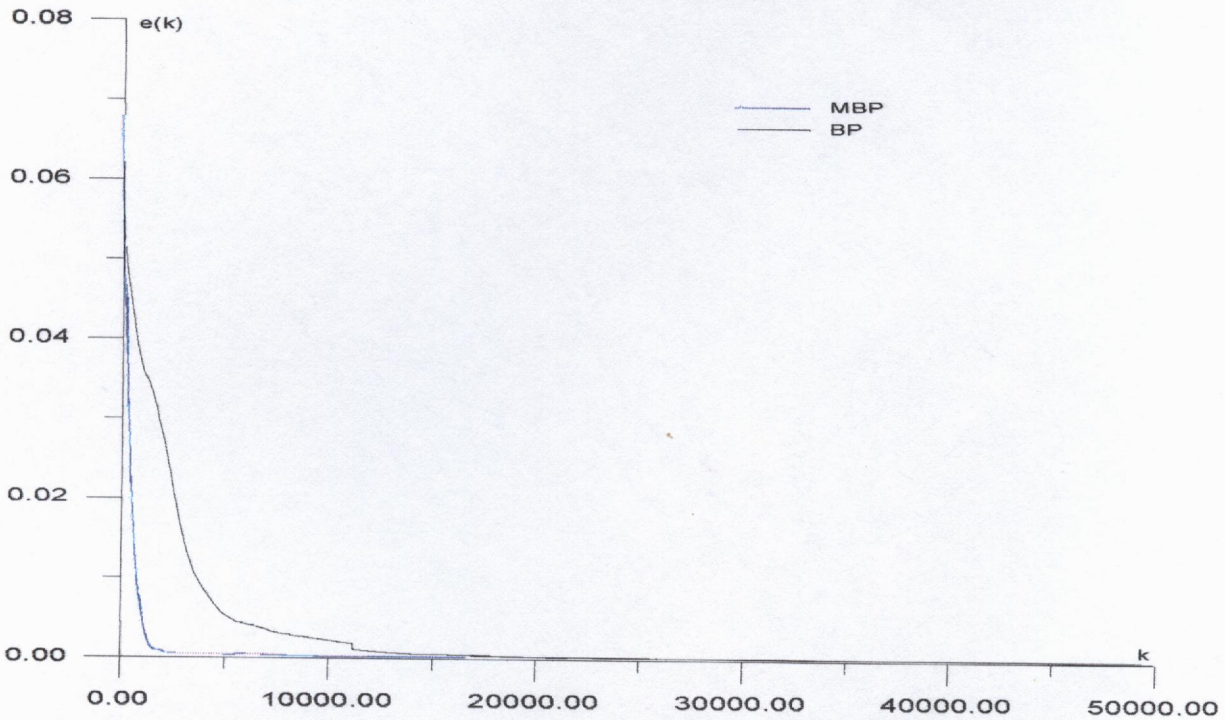
Figure (4.7) : *apprentissage de x_1 par des réseaux récurrents*



a) Erreur d'apprentissage du réseau MLP utilisé pour identifier x_2

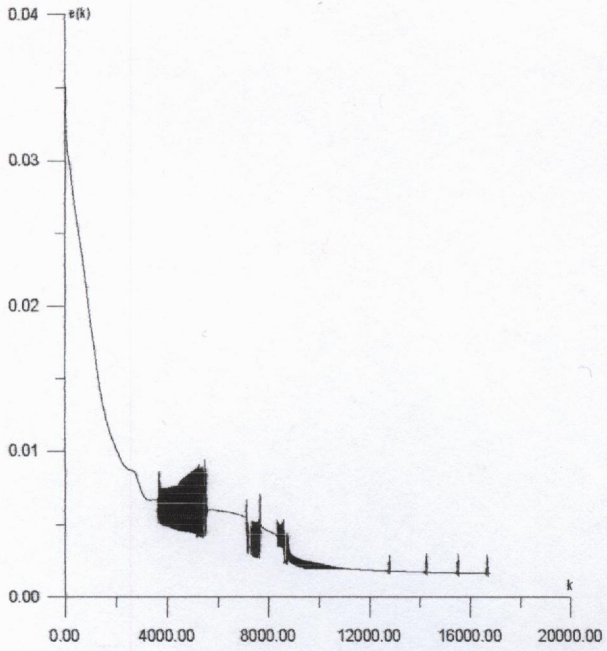


b) Erreur d'apprentissage du réseau MLP d'ordre 2 utilisé pour identifier x_2

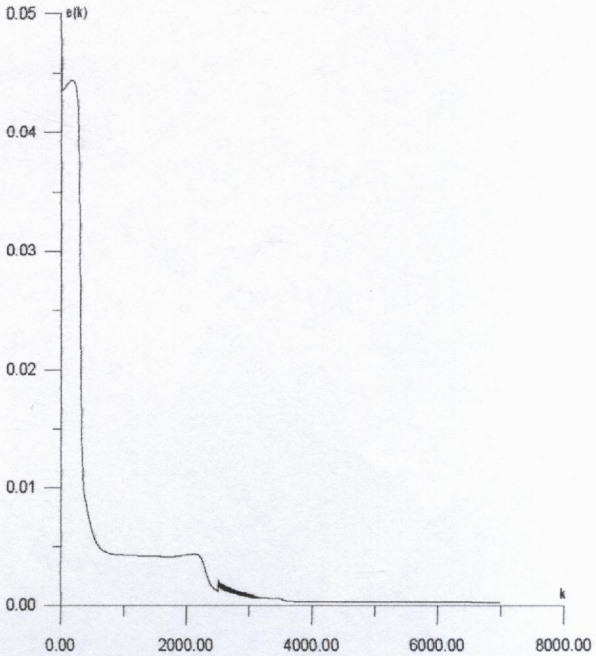


c) comparaison de la vitesse d'apprentissage des réseaux MLP

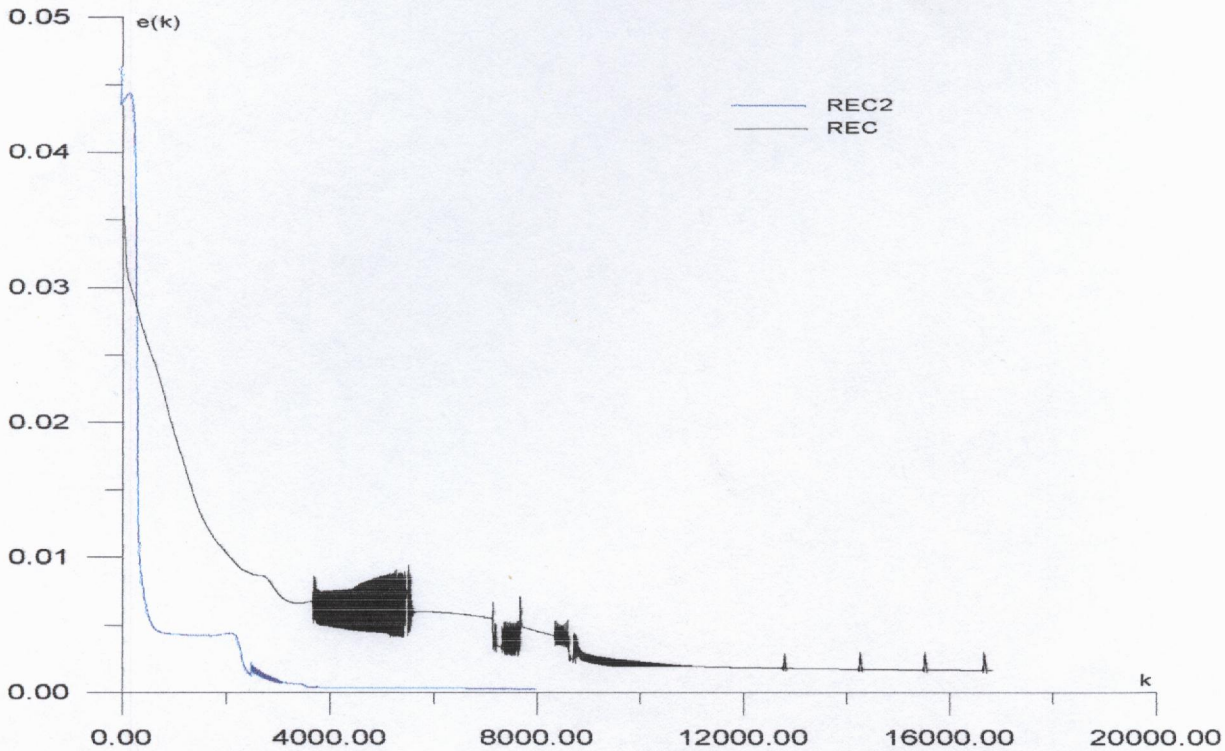
Figure (4.8) : *apprentissage de x_2 par des réseaux multicouches*



a) Erreur d'apprentissage du réseau récurrent utilisé pour identifier x_2

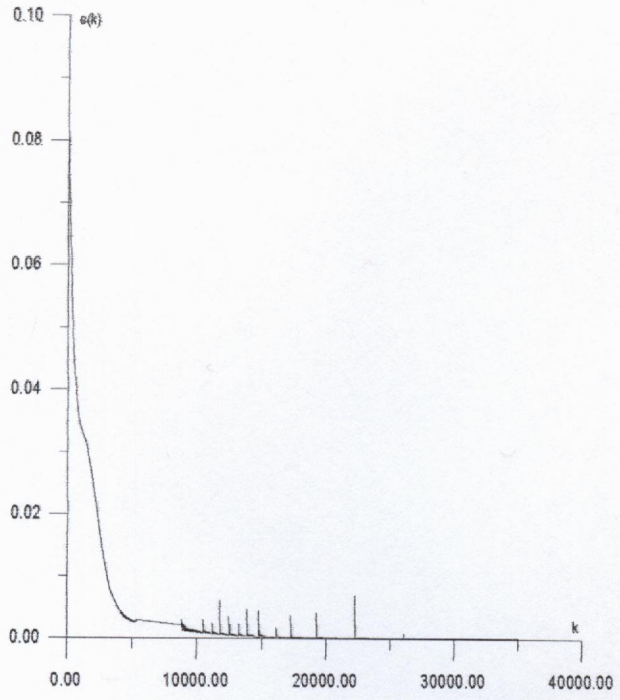


b) Erreur d'apprentissage du réseau récurrent d'ordre 2 utilisé pour identifier x_2

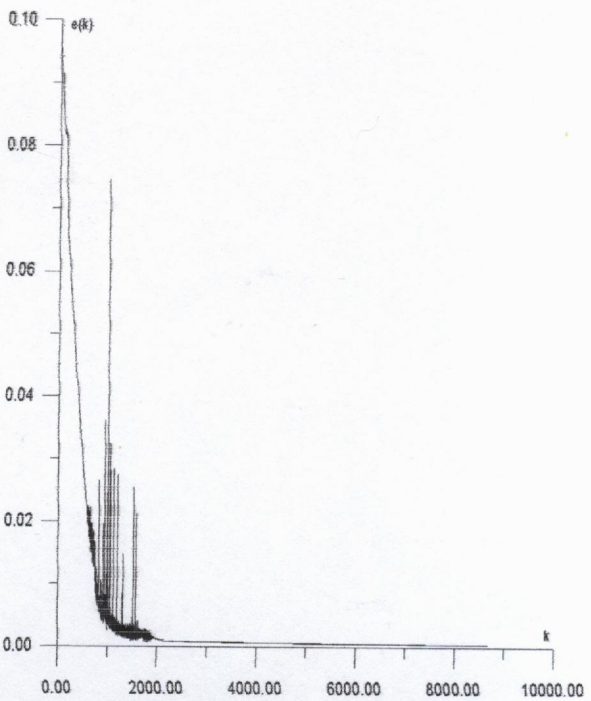


c) comparaison de la vitesse d'apprentissage des réseaux récurrents

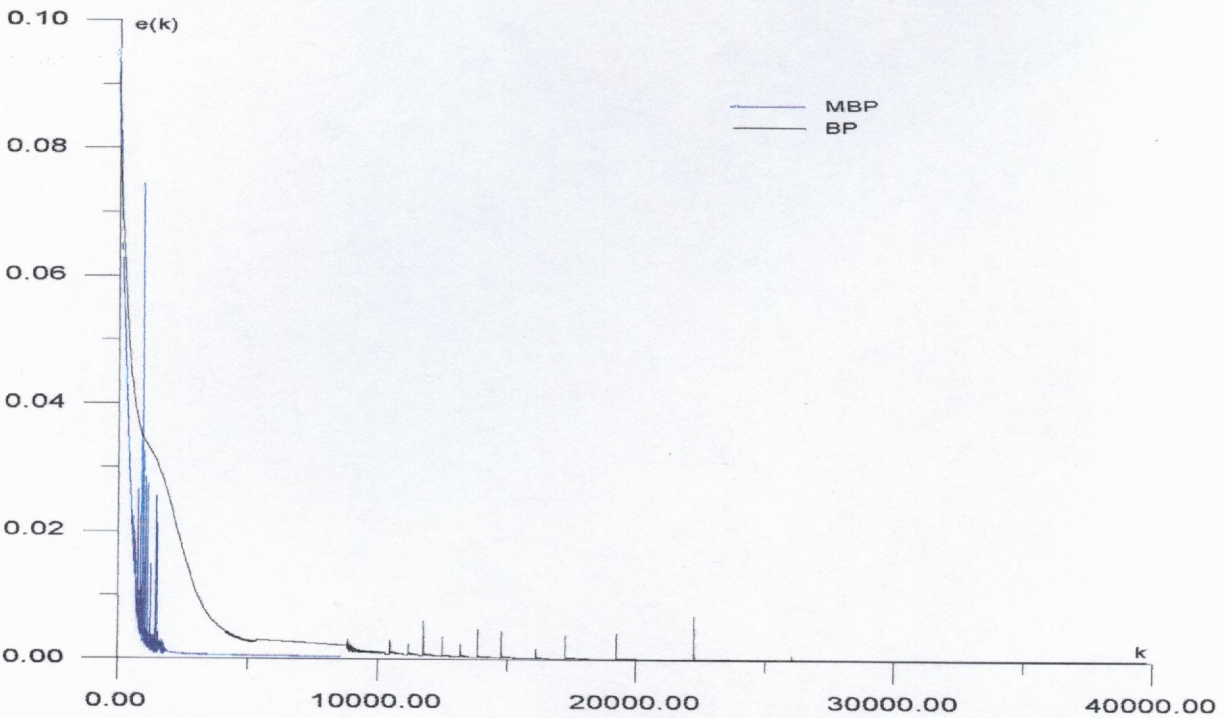
***Figure (4.9)** : apprentissage de x_2 par des réseaux récurrents*



a) Erreur d'apprentissage du réseau MLP utilisé pour identifier x_3

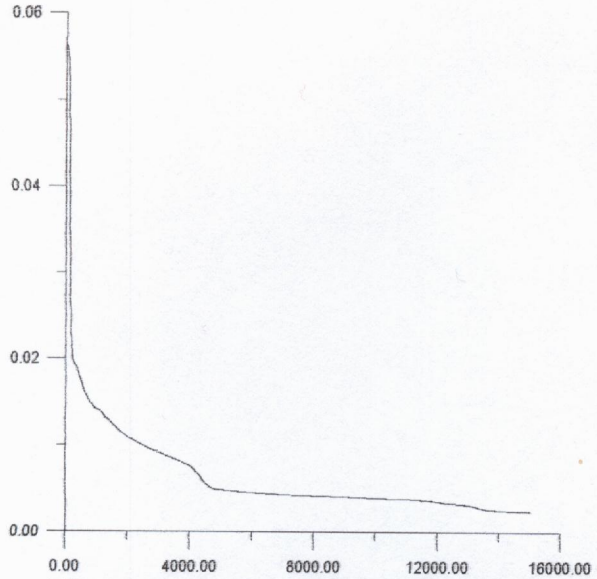


b) Erreur d'apprentissage du réseau MLP d'ordre 2 utilisé pour identifier x_3

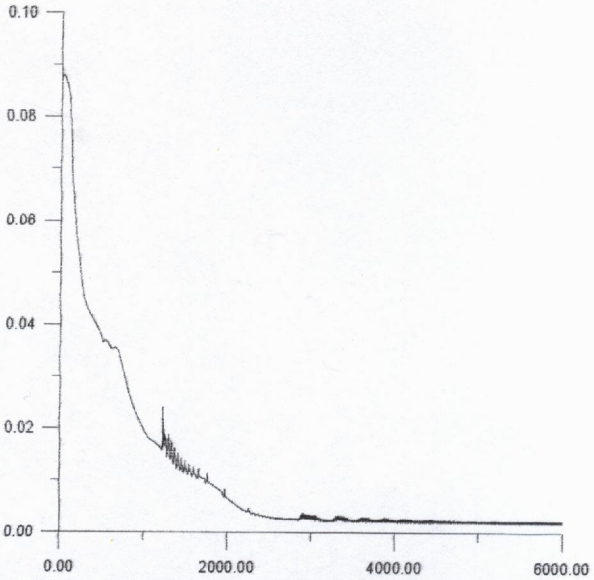


c) comparaison de la vitesse d'apprentissage des réseaux MLP

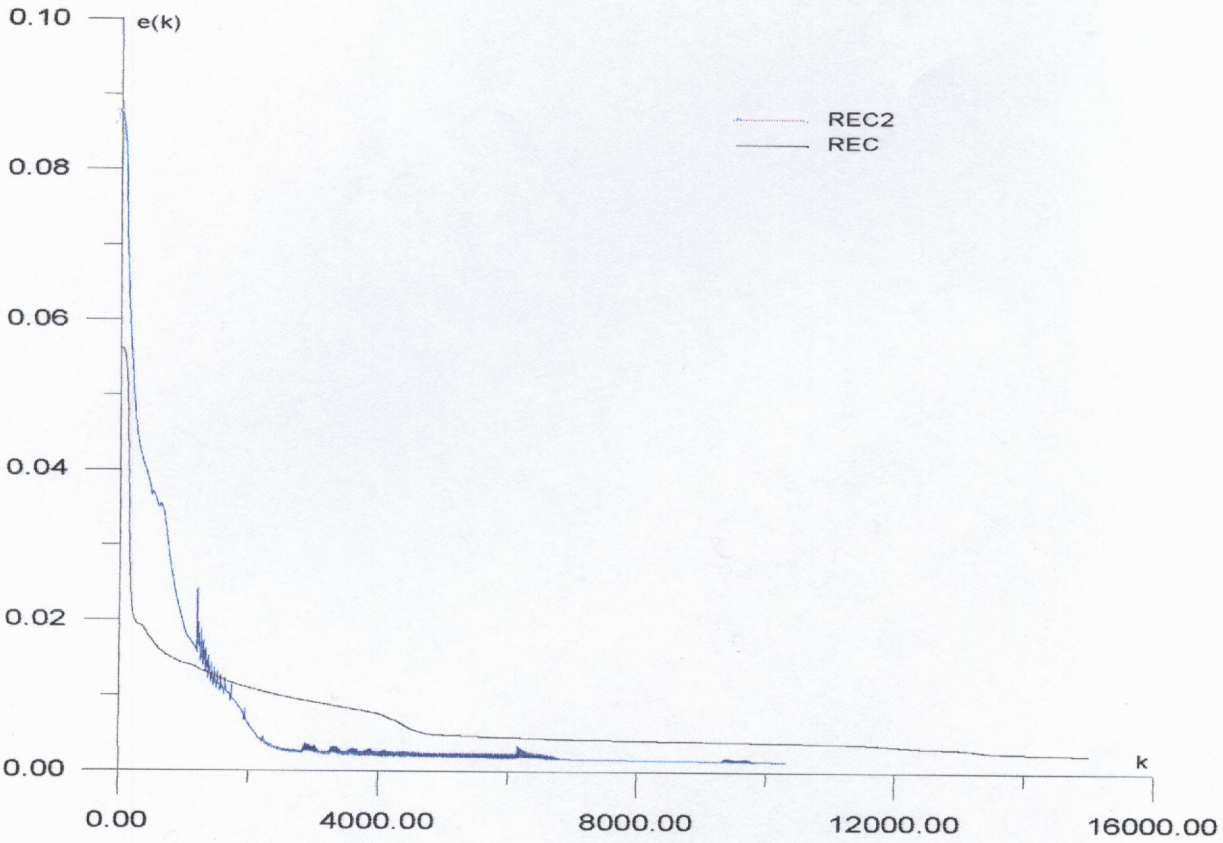
Figure (4.10) : *apprentissage de x_3 par des réseaux multicouches*



a) Erreur d'apprentissage du réseau récurrent utilisé pour identifier x_3

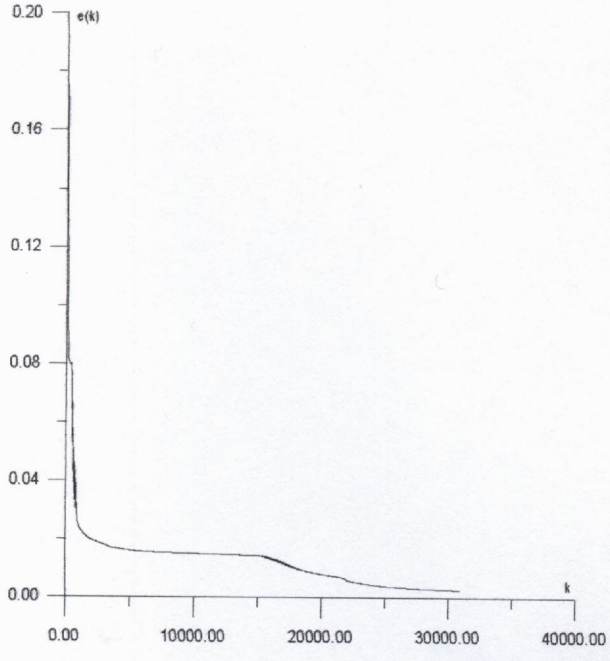


b) Erreur d'apprentissage du réseau récurrent d'ordre 2 utilisé pour identifier x_3

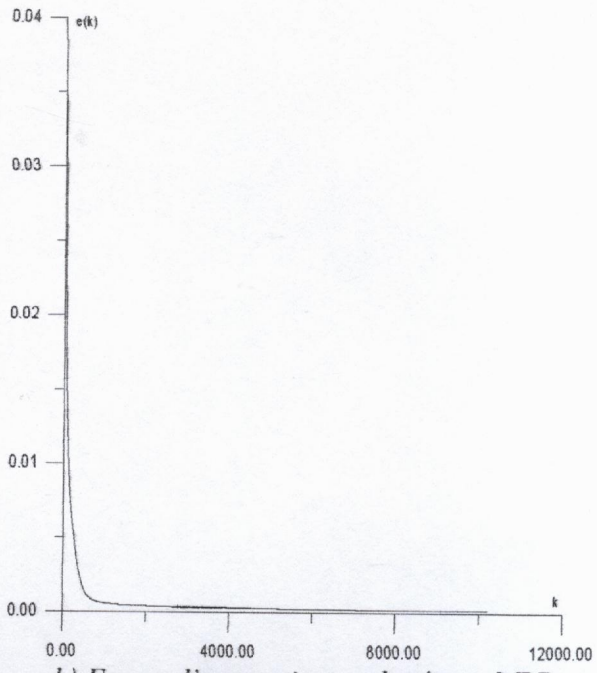


c) comparaison de la vitesse d'apprentissage des réseaux récurrents

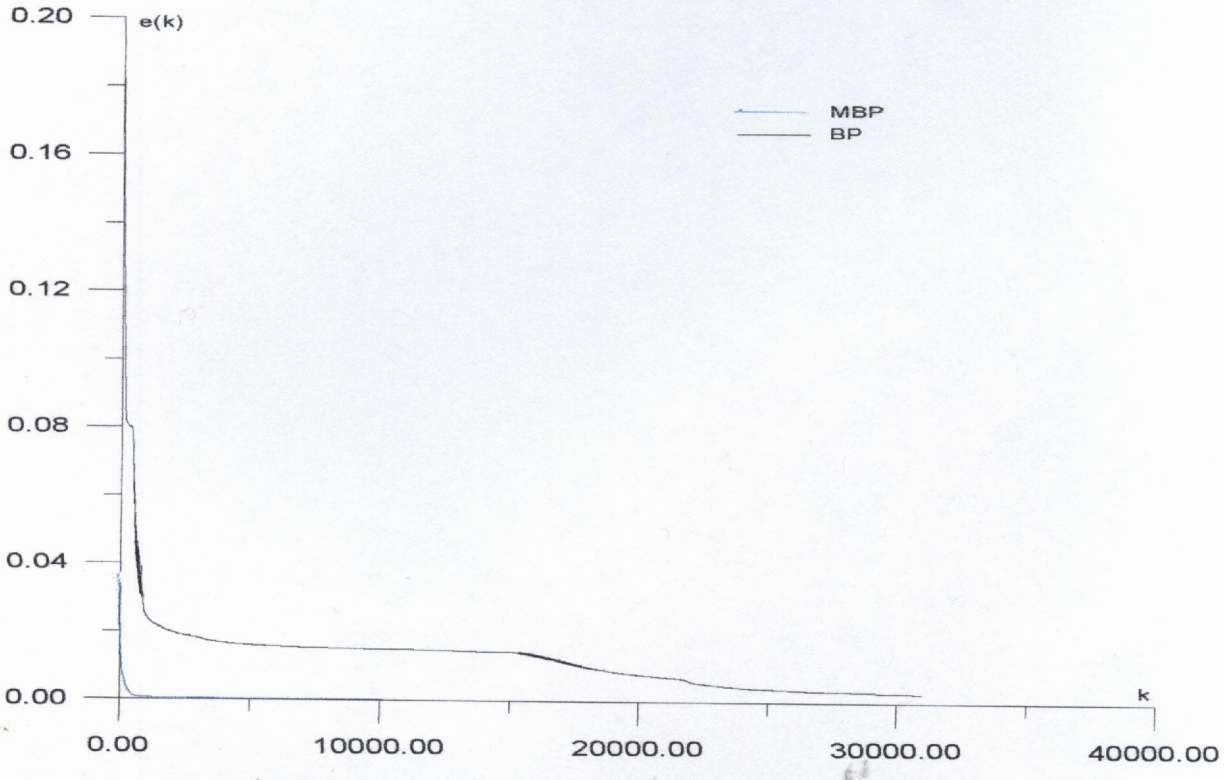
Figure (4.11) : *apprentissage de x_3 par des réseaux récurrents*



a) Erreur d'apprentissage du réseau MLP utilisé pour identifier x_4

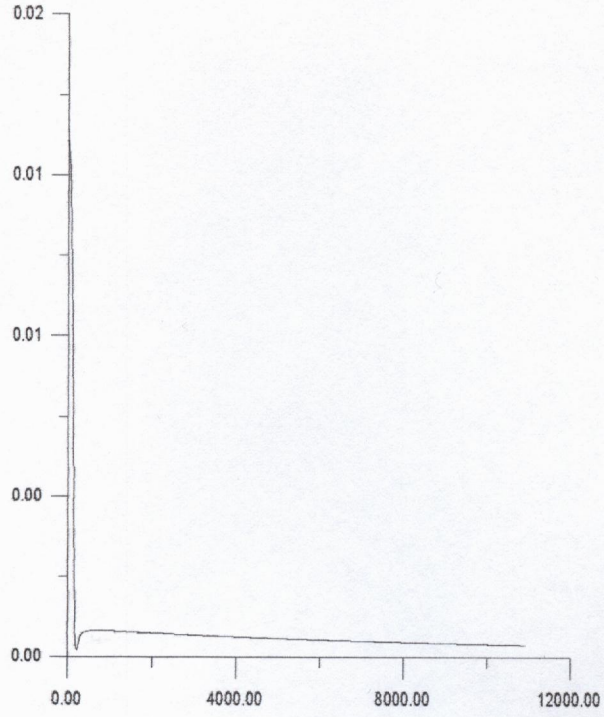


b) Erreur d'apprentissage du réseau MLP d'ordre 2 utilisé pour identifier x_4

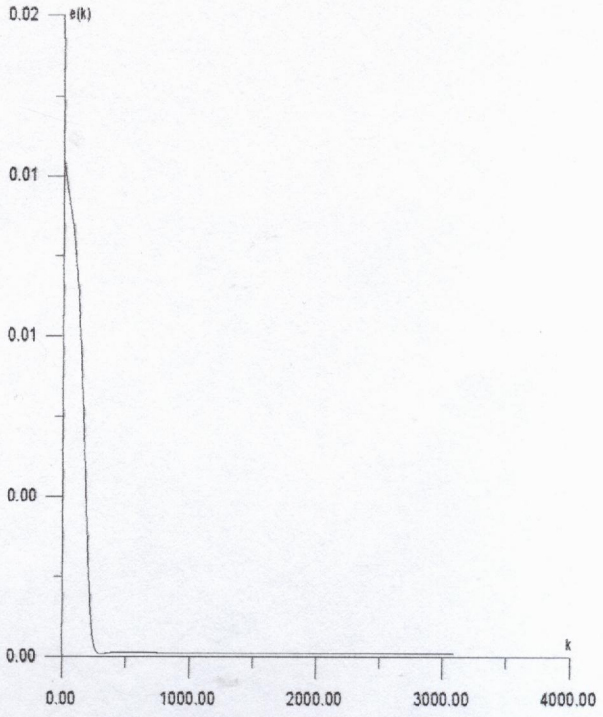


c) comparaison de la vitesse d'apprentissage des réseaux MLP

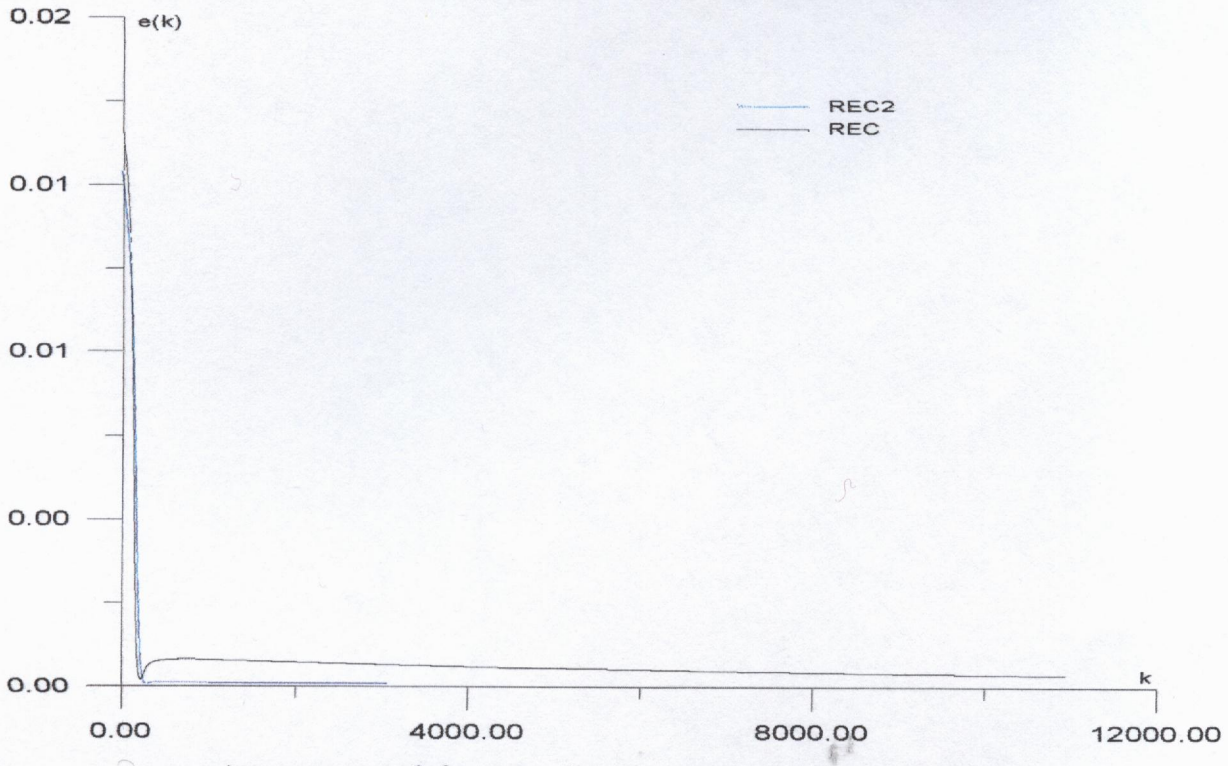
Figure (4.12) : *apprentissage de x_4 par des réseaux multicouches*



a) Erreur d'apprentissage du réseau récurrent utilisé pour identifier x_4



b) Erreur d'apprentissage du réseau récurrent d'ordre 2 utilisé pour identifier x_4



c) comparaison de la vitesse d'apprentissage des réseaux récurrents

Figure (4.13) : *apprentissage de x_4 par des réseaux récurrents*

Type du Réseau	Taille du réseau	Pas d'apprentissage	Nombre de données	Nombre des itérations
MLP	$N_{4\ 10\ 10\ 1}$	0.1	36	82000
MLP(2)	$N_{4\ 5\ 1}$	0.1	36	13500
Récurrent	6 neurones	0.001	36	21000
Récurrent (2)	3 neurones	0.001	36	7000

Tableau (4.2) : *caractéristiques de simulation pour identifier* x_1

Type du Réseau	Taille du réseau	Pas d'apprentissage	Nombre de données	Nombre des itérations
MLP	$N_{4\ 10\ 10\ 1}$	0.1	36	50000
MLP(2)	$N_{4\ 5\ 1}$	0.1	36	17000
Récurrent	6 neurones	0.001	36	16900
Récurrent (2)	3 neurones	0.001	36	7000

Tableau (4.3) : *caractéristiques de simulation pour identifier* x_2

Type du Réseau	Taille du réseau	Pas d'apprentissage	Nombre de données	Nombre des itérations
MLP	$N_{4\ 10\ 10\ 1}$	0.1	36	40000
MLP(2)	$N_{4\ 5\ 1}$	0.1	36	9000
Récurrent	6 neurones	0.001	36	15000
Récurrent (2)	3 neurones	0.001	36	6000

Tableau (4.4) : *caractéristiques de simulation pour identifier* x_3

Type du Réseau	Taille du réseau	Pas d'apprentissage	Nombre de données	Nombre des itérations
MLP	$N_{4\ 10\ 10\ 1}$	0.1	36	31000
MLP(2)	$N_{4\ 5\ 1}$	0.1	36	10300
Récurrent	6 neurones	0.001	36	11000
Récurrent (2)	3 neurones	0.001	36	3100

Tableau (4.5) *: caractéristiques de simulation pour identifier* x_4

IV.8 CONCLUSION :

L'identification de la ruche apicole par les réseaux de neurones d'ordre élevé a donné des résultats très satisfaisants, ou l'écart entre les données empiriques et les résultats des simulations est négligeable.

Il est à noter que l'identification d'entrées/sorties sera meilleure ; si on utilisait les données expérimentales provenant de capteurs électroniques (le nombre de données sera : grand et exact)

CONCLUSION GENERALE

L'objectif de ce travail est l'identification des systèmes non-linéaires par les *réseaux de neurones d'ordre élevé et de comparer leurs performances à celles des réseaux d'ordre un (vitesse, taille,...) pour les simulations qui ont été* faites.

Le réseau multicouches est le réseau le plus répandu ; il est souvent utilisé dans l'identification et le contrôle, et peut approximer n'importe quelle fonction non linéaire ; le *temps d'apprentissage de ce réseau est cependant très long. Les simulations ont montré qu'un réseau d'ordre deux d'une taille réduite est plus rapide(nombre d'itérations) qu'un réseau* d'ordre un.

Le réseau récurrent est un réseau dynamique qui est souvent utilisé pour identifier les systèmes physiques avec un comportement dynamique. L'apprentissage par les réseaux d'ordre deux nécessite un nombre d'itérations plus petit par rapport aux réseaux d'ordre un. Mais l'inconvénient des réseaux récurrents est la divergence rapide de ces algorithmes d'apprentissage en temps réel.

Pour les réseaux modulaires, les simulations ont montré qu'on peut utiliser ces réseaux pour identifier des systèmes non-linéaires. Les réseaux de neurones modulaires d'ordre deux nécessitent une amélioration concernant la réduction de nombre de poids dans le réseau pour un apprentissage plus rapide.

L'application des réseaux de neurones d'ordre élevé pour identifier la ruche apicole a donné des résultats meilleurs par rapport à l'identification par adoption des modèles paramétriques [29] et [30], malgré la difficulté de déterminer la taille exacte des réseaux et la divergence, dans plusieurs cas, des algorithmes d'apprentissage.

Dans le futur, plusieurs axes de recherche peuvent être poursuivis, notamment :

- L'amélioration des réseaux d'ordre élevé (Réduction de nombre des poids...)
- Le contrôle des systèmes non-linéaires par les réseaux de neurones d'ordre élevé (application à la ruche apicole)
- Comparaison entre le contrôle des systèmes non-linéaires par les réseaux de neurones d'ordre un et d'ordre élevé.
- L'identification et le contrôle par d'autres types des réseaux de neurones d'ordre élevé (par exemple: the piece-wise neural networks).

BIBLIOGRAPHIE

[1] Emile Fiesler
Handbook of neural computation (CH: Neural Network topologies)
1997 IOP publishing LTd and oxford university release 97/1.

[2] E..B.kosmatopolous, M.M. Polycarpou, M. A. christodoulou et P.A. Ioannou
High-order neural network structures for identification of dynamical systems.
IEEE transactions on neural networks, vol.6 No.2, pp 422-431. Mars 1995.

[3] C.L.Giles, D.Chen, C.B. Miller, H.H. Chen, G.Z.Sun et Y.C.Lee.
Second-order recurrent neural networks for grammatical inference.
.IEEE, CH 7803-0164. pp 273-281. jan.1991.

[4] H. yang et C.C.Guest
High-order neural networks with reduced numbers of interconnection weights.
IEEE IJCNN, Sandiego, California, V.III. pp 281-286, Juillet 1990.

[5] *L.Min, S.zhangkang et L.Juchang*
The second-order neural, networks for radar ship target recognition.
IEEE, CH 3158, *pp 270-274, Mars 1992.*

[6] Hon Koung Kwan
High-order feedbackward neural networks
IEEE Chine international conference on circuits and systems. pp 49-51 juin 1991

[7] D.R.hush et B.G.Horne
Progress in supervised neural networks
IEEE signal processing magazine, pp 8-39, jan. 1993

[8] Pablo.A. Estevez et Yoichi. Okabe
Training the piecewise linear high-order neural network through error back-propagation
IEEE conférence on neural networks, pp 711-716, *Mars 1991*

[9] A.Karakasoglu, S.I. Sunbramania et M.K. Sundershan
Identification and decentralized adaptive control using dynamical neural networks with application to robotic manipulators
IEEE Transactions on Neural Networks, vol.4, No.6, pp 919-930 Nov 1993.

[10] Y.K.Kim et J.B.RA
Weight value initialization for improving training speed in the back-propagation networks
IEEE. CH 3065. pp 2396-2401, 1991

[11] *T.Yamadi et T. Yabuta*
Dynamic system identification using neural networks
IEEE Trans on systems, man, and cybernetics, vol. 23,No.1, PP.204-210 Jan. / Feb. 1993.

[12] Raymond . L. Watrous et Gary M. Kuhn
Induction of finite-state languages using second-order recurrent networks
Neural computation 4, 406-414 MIT 1992.

[13] Tai, H-M ; Jong, T.-I
neural networks with higher-order non linearity
Electreonics letters vol 24, pp 1226-1231, sept 1988

[14] A.Delgado, C.Kambhampati, et K.Warwick
Identification of nonlinear systems with a dynamic recurrent neural networks
Artificial neural networks, conference publication No.409. pp 318-322 IEE 95.

[15] E..B.kosmatopolous, M. A. christodoulou et P.A. Ioannou
Identification of non-linear systems using new dynamic neural network structures.
IEEE, 31st conference on decision and control, pp 20-25 Dec. 1992.

[16] Kumpati.s, Narendra, K.parthasarathy
Gradient methods for the optimization of dynamical systems containing neural networks
IEEE Transactions on Neural Networks. Vol. 2 No. 2. pp 252-262. Mars 1991.

[17] C.L.Giles, et T.Maxwell
Learning, invariance, and generalization in high-order neural networks
Applied optics/ vol.26, No. 23 pp 4972-4978 Dec 1987

[18] Kumpati.s, Narendra Fellow et Kannan Parthasarathy
Identification and control of dynamical systems using neural networks
IEEE Trans. on Neural Networks, vol. 1, pp 4-27 Mars 1990.

[19] M.Lee, S.Y.Lee, et C.H.Park
Neural controller of non linear dynamic systems using higher-order neural networks
Electronics Letters, Jan 1992 vol. 28 No. 3. pp. 276-277

[20] R.J.Williams et David Zipser.
A learning algorithm for continually running fully recurrent neural networks
Neural computation 1, 270-280 (1989) M.I.T

[21] Y.C. Lee. et Gary Doolen.
Machine learning using a higher order correlation network
Physica 22D (1986). pp 276-306.Amsterdam Holland.

[22] Hong. K. Kwan
high-order feedbackward neural networks
IEEE on circuits and systems, vol 2, pp 1007-1018 juin 1991

[23] Jeffries. C
high-order neural networks
IEEE on neural networks, vol 2, pp 646-670, juin 1990.

[24] R.L.Watrous, et G.M. Kuhn
Induction of finite state languages using second-order recurrent networks
Neural computation 4,406-414. M.I.T 1992

[25] G.Z.sun, H.H.Chen, Y.C. Lee, et L.Giles
Turing equivalence of neural networks with second order connection weights
IEEE CH 7803, pp.357-360. Jan. 1991

[26] A.Dembo, O.Farotimi, et T. Kailath
High-order absolutely stable neural networks.
IEEE Transactions on circuits and systems, 38, No.1, PP.57-65 Jan.1991

[27] C. L. Giles, C. B. Miller, D. Chen, H. H. Chen, G. Z. Sun, Y. C. Lee
Learning and extracting finite state automata with second-order recurrent neural networks.
Neural computation 4, 393-405 1992 M.I.T

[28] M. Oubbati
Identification et contrôle des systèmes dynamiques par les réseaux de neurones application à une ruche apicole
Thèse de magister-Institut d'électronique BLIDA 1996

[29] I. Rizoug, A. Djellal
Commande d'une ruche apicole par la logique floue
Thèse de projet de fin d'étude-Institut d'électronique BLIDA 1995

[30] B.Dra-El-Mizan
Commande d'une ruche apicole
Thèse de projet de fin d'étude-Institut de mathémathique BLIDA 1994

[31] S. Y. Kung
Digital neural networks
PTR Prentice Hall.Englewood Cliffs, New Jersey 07632, 1993

[32] S. Haykin
Neural networks A Comprehensive Foundation
Copyright by Macmillan college publishing company, INC, 1994

LES POIDS SYNAPTIQUES DU RESEAU MULTICOUCHE D'ORDRE UN

Poids	Poids initiaux	Poids finaux	Poids	Poids initiaux	Poids finaux
W(1,1,0)	5.0476E-01	4.9912E-02	W(2,2,1)	-5.2252E-01	-1.3667E+00
W(1,2,0)	2.3962E-01	-8.0679E-01	W(2,1,2)	-5.7303E-01	-1.1728E+00
W(1,3,0)	2.3329E-02	4.1117E-01	W(2,2,2)	1.4488E-01	-7.5845E-01
W(1,4,0)	-2.9263E-01	-7.9555E-01	W(2,1,3)	2.9190E-01	1.9684E+00
W(1,5,0)	-4.5019E-01	-5.0131E-01	W(2,2,3)	9.4850E-01	2.0018E+00
W(1,6,0)	-6.6111E-01	-7.3564E-01	W(2,1,4)	-3.1425E-01	-1.0282E+00
W(1,7,0)	2.2406E-02	-2.6732E-01	W(2,2,4)	-6.4708E-01	-1.5650E+00
W(1,8,0)	-1.8049E-01	-1.9232E+00	W(2,1,5)	7.1468E-01	7.0180E-01
W(1,1,1)	8.8524E-01	1.5809E+00	W(2,2,5)	-4.6047E-01	-6.1964E-01
W(1,2,1)	3.1693E-01	1.6357E+00	W(2,1,6)	3.3454E-01	5.0089E-01
W(1,3,1)	1.3513E-01	-2.1999E+00	W(2,2,6)	1.4090E-01	9.6518E-02
W(1,4,1)	3.4070E-01	1.9229E+00	W(2,1,7)	1.4760E-01	1.7270E-01
W(1,5,1)	7.4350E-01	1.8121E-01	W(2,2,7)	-1.4984E-01	-5.3061E-01
W(1,6,1)	4.7650E-01	-1.0076E-01	W(2,1,8)	-4.2547E-01	-2.7236E+00
W(1,7,1)	4.5329E-01	6.1971E-01	W(2,2,8)	-9.4668E-02	-9.8671E-01
W(1,8,1)	9.9557E-01	3.2388E+00	W(3,1,0)	4.0119E-01	-1.5304E+00
W(2,1,0)	-4.9221E-02	1.9636E+00	W(3,1,1)	4.4875E-01	2.0629E+00
W(2,2,0)	-6.6092E-01	-3.9477E-01	W(3,1,2)	-2.9938E-01	1.8251E+00
W(2,1,1)	4.1245E-01	1.3931E-01			

LES POIDS SYNAPTIQUES DU RESEAU MULTICOUCHE D'ORDRE DEUX

Poids	Initiaux	Finaux	Poids	Initiaux	Finaux
W(1,1,0)	5.0476E-01	-3.5891E-01	P(2,1,0,0)	7.1468E-01	-6.1470E-01
P(1,1,0,0)	2.3962E-01	-2.5442E-01	P(2,1,0,1)	-4.6047E-01	-2.0928E-01
P(1,1,0,1)	2.3329E-02	-1.9094E-01	P(2,1,0,2)	3.3454E-01	-3.5842E-02
W(1,2,0)	-2.9263E-01	6.1325E-02	P(2,1,0,3)	1.4090E-01	-6.0989E-01
P(1,2,0,0)	-4.5019E-01	-9.7482E-01	P(2,1,0,4)	1.4760E-01	-8.3961E-01
P(1,2,0,1)	-6.6111E-01	4.1715E-01	P(2,1,0,5)	-1.4984E-01	-8.9224E-02
W(1,3,0)	2.2406E-02	5.7593E-02	W(2,1,1)	-4.2547E-01	7.4992E-01
P(1,3,0,0)	-1.8049E-01	-1.0568E+00	P(2,1,1,1)	-9.4668E-02	-2.1573E-01
P(1,3,0,1)	8.8524E-01	5.7634E-01	P(2,1,1,2)	4.0119E-01	-1.1868E-01
W(1,4,0)	3.1693E-01	-2.6592E-01	P(2,1,1,3)	4.4875E-01	2.1809E-01
P(1,4,0,0)	1.3513E-01	-3.9719E-01	P(2,1,1,4)	-2.9938E-01	1.9127E-01
P(1,4,0,1)	3.4070E-01	-1.1625E+00	P(2,1,1,5)	-2.3856E-01	5.1808E-02
W(1,5,0)	7.4350E-01	-1.2078E-01	W(2,1,2)	1.2976E-01	-6.2445E-01
P(1,5,0,0)	4.7650E-01	-2.1630E-01	P(2,1,2,2)	-6.2757E-01	-2.1467E-01
P(1,5,0,1)	4.5329E-01	-6.0571E-01	P(2,1,2,3)	6.0148E-01	3.5188E-01
W(1,1,1)	9.9557E-01	-1.1046E+00	P(2,1,2,4)	4.3194E-02	8.3916E-01
P(1,1,1,1)	-4.9221E-02	-1.8401E-02	P(2,1,2,5)	3.3135E-01	-2.3214E-01
W(1,2,1)	-6.6092E-01	2.3255E-01	W(2,1,3)	-3.7297E-01	4.8389E-01
P(1,2,1,1)	4.1246E-01	7.5059E-01	P(2,1,3,3)	-7.4043E-01	-4.2849E-01
W(1,3,1)	-5.2252E-01	-7.3347E-01	P(2,1,3,4)	-1.9665E-01	5.1717E-01
P(1,3,1,1)	-5.7303E-01	-9.6541E-01	P(2,1,3,5)	-1.5223E-01	6.7619E-01
W(1,4,1)	1.4488E-01	2.1308E-01	W(2,1,4)	6.8716E-01	7.2533E-01
P(1,4,1,1)	2.9190E-01	-3.9421E-01	P(2,1,4,4)	-7.4898E-01	-7.4808E-01
W(1,5,1)	9.4850E-01	2.5135E-01	P(2,1,4,5)	3.1338E-01	8.6046E-01
P(1,5,1,1)	-3.1426E-01	7.7582E-02	W(2,1,5)	8.6454E-01	-1.2185E-01
W(2,1,0)	-6.4708E-01	1.3662E-01	P(2,1,5,5)	-3.7908E-03	-3.6820E-01

LES POIDS SYNAPTIQUES DU RESEAU MULTICOUCHE D'ORDRE TROIS

Poids	Initiaux	Finaux	Poids	Initiaux	Finaux
w1(1,1,0)	5.0476E-01	-1.1924E+00	w3(2,1,0,1,3)	-1.5223E-01	9.3737E-01
w2(1,1,0,0)	2.3962E-01	-4.0169E-01	w3(2,1,0,1,4)	6.8716E-01	-3.9497E-01
w3(1,1,0,0,0)	2.3329E-02	2.3329E-02	w2(2,1,0,2)	-7.4898E-01	1.7602E-01
w3(1,1,0,0,1)	-2.9263E-01	4.5329E-01	w3(2,1,0,2,2)	3.1338E-01	9.1331E-01
w2(1,1,0,1)	-4.5019E-01	-1.2590E+00	w3(2,1,0,2,3)	8.6454E-01	6.2237E-01
w3(1,1,0,1,1)	-6.6111E-01	1.3125E-01	w3(2,1,0,2,4)	-3.7908E-03	2.6217E-01
w1(1,2,0)	2.2406E-02	-9.9285E-01	w2(2,1,0,3)	2.4575E-01	3.1963E-01
w2(1,2,0,0)	-1.8049E-01	-5.8160E-01	w3(2,1,0,3,3)	5.6352E-01	2.7592E+00
w3(1,2,0,0,0)	8.8524E-01	-6.6111E-01	w3(2,1,0,3,4)	4.1098E-01	-1.9649E-01
w3(1,2,0,0,1)	3.1693E-01	-6.6092E-01	w2(2,1,0,4)	4.6626E-01	-2.1019E-01
w2(1,2,0,1)	1.3513E-01	6.4304E-02	w3(2,1,0,4,4)	-4.9253E-01	-2.5009E+00
w3(1,2,0,1,1)	3.4070E-01	-3.4365E-01	w1(2,1,1)	7.6276E-01	-3.6258E-01
w1(1,3,0)	7.4350E-01	-4.6084E-01	w2(2,1,1,1)	-1.3986E-01	1.0594E+00
w2(1,3,0,0)	4.7650E-01	-1.1309E+00	w3(2,1,1,1,1)	-2.1811E-02	-4.2513E-01
w3(1,3,0,0,0)	4.5329E-01	8.8524E-01	w3(2,1,1,1,2)	-6.0643E-01	8.4271E-01
w3(1,3,0,0,1)	9.9557E-01	-5.7303E-01	w3(2,1,1,1,3)	-1.6115E-01	-1.4775E-01
w2(1,3,0,1)	-4.9221E-02	-6.6751E-01	w3(2,1,1,1,4)	4.2879E-01	-1.6691E-01
w3(1,3,0,1,1)	-6.6092E-01	-8.1927E-01	w2(2,1,1,2)	8.8159E-01	9.1679E-01
w1(1,4,0)	4.1245E-01	-9.9962E-01	w3(2,1,1,2,2)	8.5522E-01	-2.8375E-03
w2(1,4,0,0)	-5.2252E-01	-6.1305E-01	w3(2,1,1,2,3)	9.3822E-01	3.6371E-01
w3(1,4,0,0,0)	-5.7303E-01	3.4070E-01	w3(2,1,1,2,4)	6.7750E-01	2.0702E-01
w3(1,4,0,0,1)	1.4488E-01	9.4850E-01	w2(2,1,1,3)	-5.0516E-01	-1.3644E-01
w2(1,4,0,1)	2.9190E-01	1.1353E+00	w3(2,1,1,3,3)	9.3737E-01	-3.6524E-01
w3(1,4,0,1,1)	9.4850E-01	4.7211E-01	w3(2,1,1,3,4)	9.2616E-01	6.4113E-01
w1(1,1,1)	-3.1425E-01	-9.1134E-01	w2(2,1,1,4)	1.0565E-02	-9.5915E-01
w2(1,1,1,1)	-6.4708E-01	-8.2200E-01	w3(2,1,1,4,4)	-3.9497E-01	-6.6172E-01
w3(1,1,1,1,1)	7.1468E-01	1.1383E-01	w1(2,1,2)	2.7867E-01	4.6882E-01
w1(1,2,1)	-4.6047E-01	-8.9781E-01	w2(2,1,2,2)	-1.1336E-01	1.0089E+00
w2(1,2,1,1)	3.3454E-01	1.1692E-01	w3(2,1,2,2,2)	1.7602E-01	9.2461E-01
w3(1,2,1,1,1)	1.4090E-01	1.0675E-01	w3(2,1,2,2,3)	-3.1973E-02	-7.8872E-01
w1(1,3,1)	1.4760E-01	-1.8631E-01	w3(2,1,2,2,4)	6.1155E-01	4.9441E-02
w2(1,3,1,1)	-1.4984E-01	-7.6677E-01	w2(2,1,2,3)	6.2237E-01	5.0106E-01
w3(1,3,1,1,1)	-4.2547E-01	-7.4407E-01	w3(2,1,2,3,3)	7.6567E-01	-4.3094E-01
w1(1,4,1)	-9.4668E-02	5.3753E-01	w3(2,1,2,3,4)	-1.9155E-01	-9.0752E-01
w2(1,4,1,1)	4.0119E-01	-1.2886E-01	w2(2,1,2,4)	2.6217E-01	-7.4908E-02
w3(1,4,1,1,1)	4.4875E-01	-2.1281E-01	w3(2,1,2,4,4)	1.5487E-01	-2.2653E-01
w1(2,1,0)	-2.9938E-01	3.7194E-02	w1(2,1,3)	7.1507E-01	-2.1367E-01
w2(2,1,0,0)	-2.3856E-01	-2.7798E-01	w2(2,1,3,3)	3.1963E-01	2.3496E+00
w3(2,1,0,0,0)	1.2976E-01	8.6454E-01	w3(2,1,3,3,3)	-3.4952E-01	7.3222E-01
w3(2,1,0,0,1)	-6.2757E-01	5.6352E-01	w3(2,1,3,3,4)	-7.7564E-01	3.4005E-01
w3(2,1,0,0,2)	6.0148E-01	-4.9253E-01	w2(2,1,3,4)	-1.9649E-01	6.1143E-01
w3(2,1,0,0,3)	4.3194E-02	-2.1811E-02	w3(2,1,3,4,4)	1.6371E-01	3.8717E-01
w3(2,1,0,0,4)	3.3135E-01	4.2879E-01	w1(2,1,4)	7.9011E-01	-7.3746E-01
w2(2,1,0,1)	-3.7297E-01	4.1242E-01	w2(2,1,4,4)	-2.1019E-01	-3.0636E+00
w3(2,1,0,1,1)	-7.4043E-01	9.3822E-01	w3(2,1,4,4,4)	1.5402E-01	-1.0534E+00
w3(2,1,0,1,2)	-1.9665E-01	9.3737E-01			

LES POIDS SYNAPTIQUES DES RESEAUX MODULAIRES

Réseau modulaire d'ordre un			Réseau modulaire d'ordre deux		
Poids	Initiaux	Finaux	Poids	Initiaux	Finaux
W(1,1,1)	5.0476E-01	3.9783E+00	W(1,1,1,1)	5.0476E-01	9.5196E+00
W(2,1,1)	2.3962E-01	3.9822E+00	W(2,1,1,1)	2.3962E-01	9.5174E+00
W(3,1,1)	2.3329E-02	3.9801E+00	W(3,1,1,1)	2.3329E-02	9.5148E+00
W(4,1,1)	-2.9263E-01	3.9855E+00	A(1,1,1)	-2.9263E-01	-2.4460E-01
W(5,1,1)	-4.5019E-01	3.9821E+00	A(2,1,1)	-4.5019E-01	-4.5202E-01
A(1,1)	-6.6111E-01	-5.9931E-01	A(3,1,1)	-6.6111E-01	-7.0732E-01
A(2,1)	2.2406E-02	6.4060E-02			
A(3,1)	-1.8049E-01	-1.8246E-01			
A(4,1)	8.8524E-01	8.6000E-01			
A(5,1)	3.1693E-01	2.4068E-01			

LES POIDS SYNAPTIQUES DES RESEAUX RECURRENT D'ORDRE UN

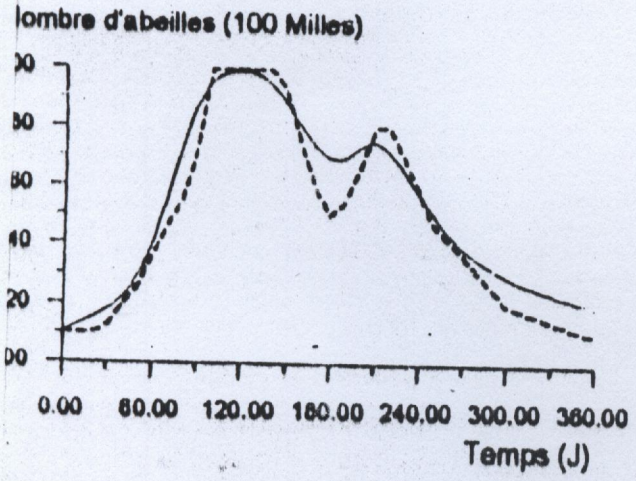
Poids	Initiaux	Finaux	Poids	Initiaux	Finaux
W(1,0)	5.0476E-01	1.2435E-01	W(4,3)	4.1245E-01	9.7713E-01
W(2,0)	2.3962E-01	-2.2912E-01	W(5,3)	-5.2252E-01	-9.7939E-02
W(3,0)	2.3329E-02	-3.6611E-01	W(1,4)	-5.7303E-01	1.6671E-01
W(4,0)	-2.9263E-01	-6.5056E-01	W(2,4)	1.4488E-01	7.2346E-01
W(5,0)	-4.5019E-01	-8.8306E-01	W(3,4)	2.9190E-01	7.4110E-01
W(1,1)	-6.6111E-01	-7.8440E-01	W(4,4)	9.4850E-01	1.2951E+00
W(2,1)	2.2406E-02	-1.7551E-01	W(5,4)	-3.1425E-01	-1.8830E-01
W(3,1)	-1.8049E-01	-2.9851E-01	W(1,5)	-6.4708E-01	-2.3473E-01
W(4,1)	8.8524E-01	7.0535E-01	W(2,5)	7.1468E-01	1.0895E+00
W(5,1)	3.1693E-01	3.1474E-02	W(3,5)	-4.6047E-01	5.4463E-02
W(1,2)	1.3513E-01	2.3898E-01	W(4,5)	3.3454E-01	8.0949E-01
W(2,2)	3.4070E-01	3.3277E-01	W(5,5)	1.4090E-01	5.4849E-01
W(3,2)	7.4350E-01	7.4512E-01	W(1,6)	1.4760E-01	-8.7747E-01
W(4,2)	4.7650E-01	3.7422E-01	W(2,6)	-1.4984E-01	-1.3816E+00
W(5,2)	4.5329E-01	2.1575E-01	W(3,6)	-4.2547E-01	-2.4211E+00
W(1,3)	9.9557E-01	1.5564E+00	W(4,6)	-9.4668E-02	-2.0870E+00
W(2,3)	-4.9221E-02	4.5017E-01	W(5,6)	4.0119E-01	-1.4479E+00
W(3,3)	-6.6092E-01	9.0489E-04			

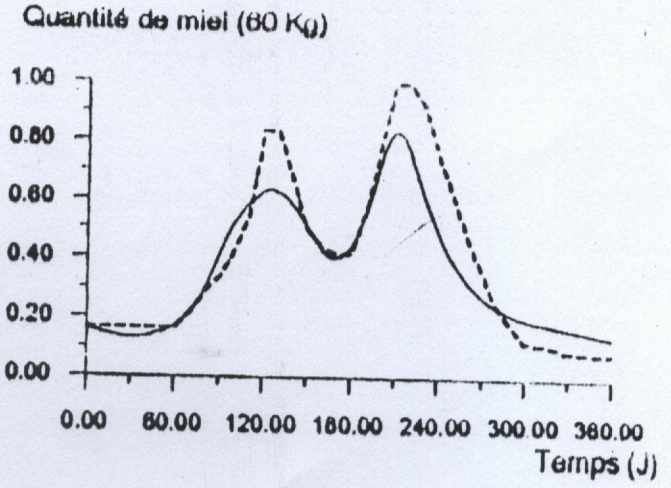
LES POIDS SYNAPTIQUES DES RESEAUX RECURRENTS D'ORDRE DEUX

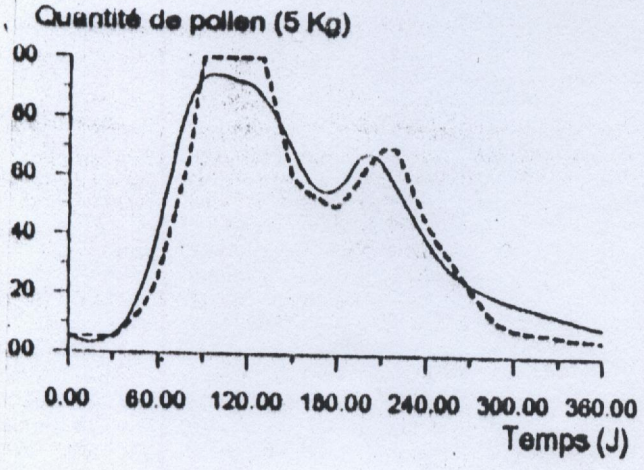
Poids	Initiaux	Finaux	Poids	Initiaux	Finaux
W(1,1,0)	5.0476E-01	-3.1119E-01	W(2,2,1)	3.1693E-01	1.6896E+00
W(1,1,1)	2.3962E-01	5.6073E-01	W(2,3,0)	1.3513E-01	-3.2304E-01
W(1,2,0)	2.3329E-02	-1.1286E+00	W(2,3,1)	3.4070E-01	5.3224E-01
W(1,2,1)	-2.9263E-01	1.8895E-01	W(3,1,0)	7.4350E-01	4.0588E+00
W(1,3,0)	-4.5019E-01	-4.8552E-01	W(3,1,1)	4.7650E-01	-9.0945E-01
W(1,3,1)	-6.6111E-01	-6.4634E-01	W(3,2,0)	4.5329E-01	4.7043E+00
W(2,1,0)	2.2406E-02	-2.4478E+00	W(3,2,1)	9.9557E-01	-7.8157E+00
W(2,1,1)	-1.8049E-01	8.5218E-01	W(3,3,0)	-4.9221E-02	5.8627E+00
W(2,2,0)	8.8524E-01	-2.2984E+00	W(3,3,1)	-6.6092E-01	-2.1567E-01

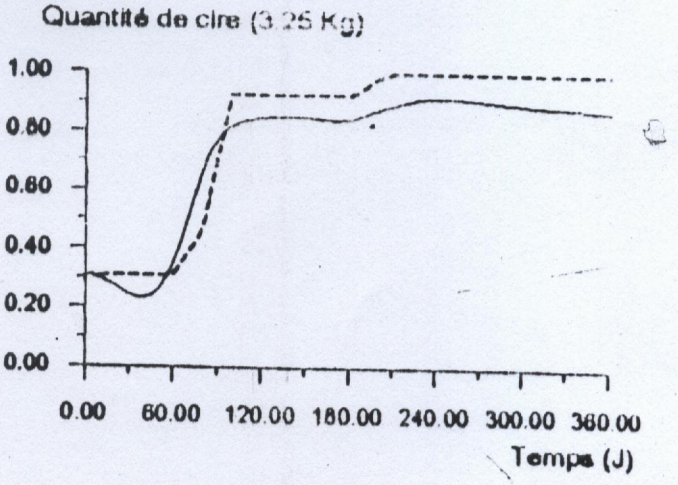
ANNEXE I

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







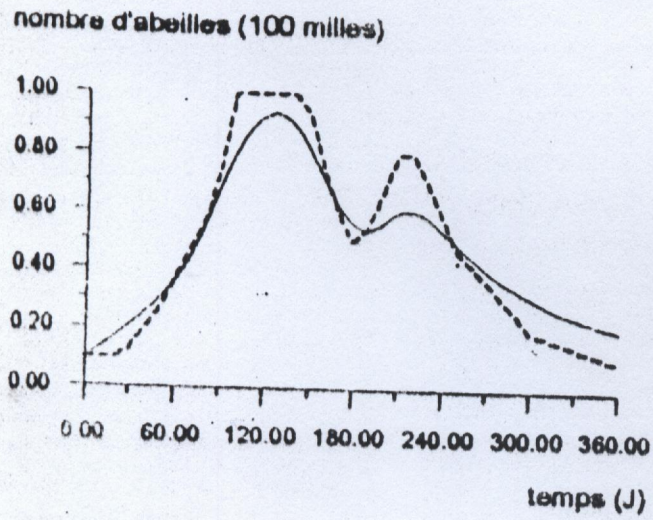


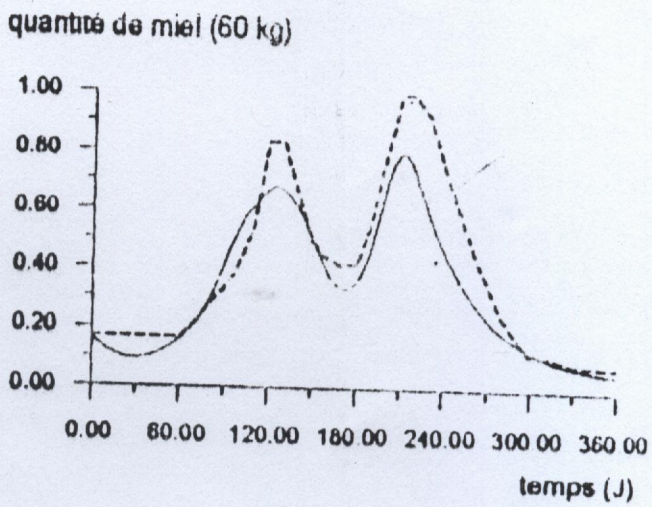
——— Modèle

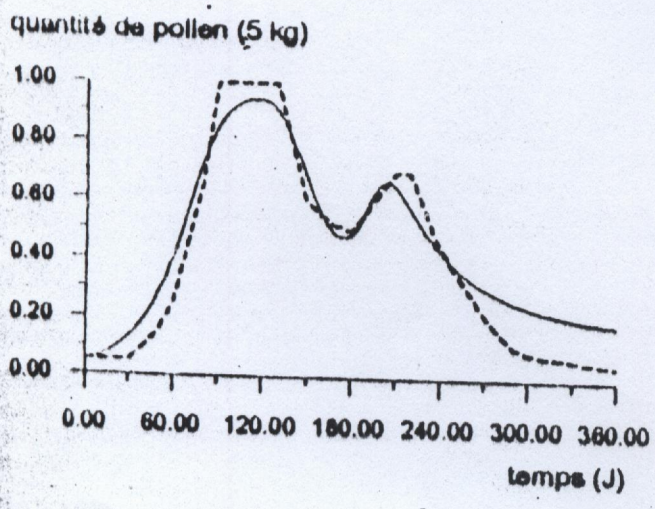
..... Les données expérimentales

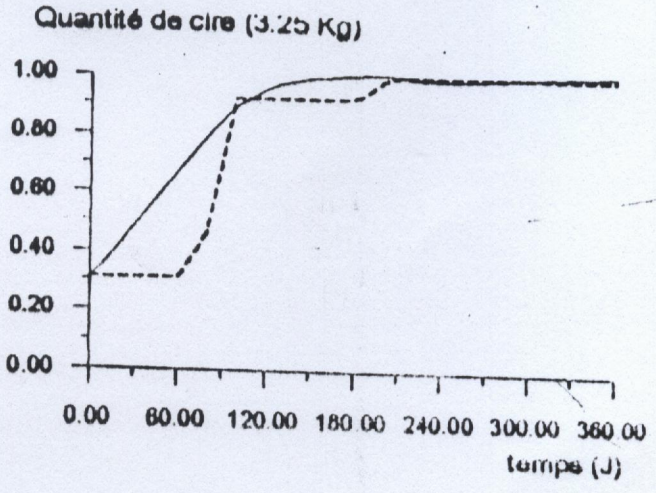
ANNEXE II

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









——— Modèle

..... Les données expérimentales

ANNEXE III

Superposition des résultats du modèle linéaire et les données expérimentales.