Chapitre II

Commande et Identification par les Réseaux de Neurone Artificiels

II.1. Introduction

L'un des défis de l'homme aujourd'hui est de copier la nature et de reproduire des modes de raisonnement et de comportement qui lui sont propre. Les réseaux de neurones, sont nés de cette envie, ils constituent une famille de fonctions non linéaires paramétrées, utilisées dans de nombreux domaines (physique, chimie, biologie, finance, etc.), notamment pour la modélisation de processus et la synthèse de lois de commandes, Ce cours décrit une technique intelligente nouvellement introduite dans le monde de l'automatique. Il s'agit principalement des réseaux de neurones artificiels et les différentes structures qui leurs sont associées ainsi que nous abordons par la suite l'identification et le contrôle de processus par les réseaux de neurones pour la synthèse de lois de commandes.

II.2. Généralités

L'origine des réseaux de neurones vient de l'essai de modélisation mathématique du cerveau humain. Les premiers travaux datent de 1943 et sont l'œuvre de MM. Mac Culloch et Pitts. Ils supposent que l'impulsion nerveuse est le résultat d'un calcul simple effectué par chaque neurone et que la pensée née grâce à l'effet collectif d'un réseau de neurones interconnectés (figure I.1). Ils ont connu des débuts prometteurs vers la fin des années 50, mais le manque d'approfondissement de la théorie a gelé ces travaux jusqu'aux années 80.

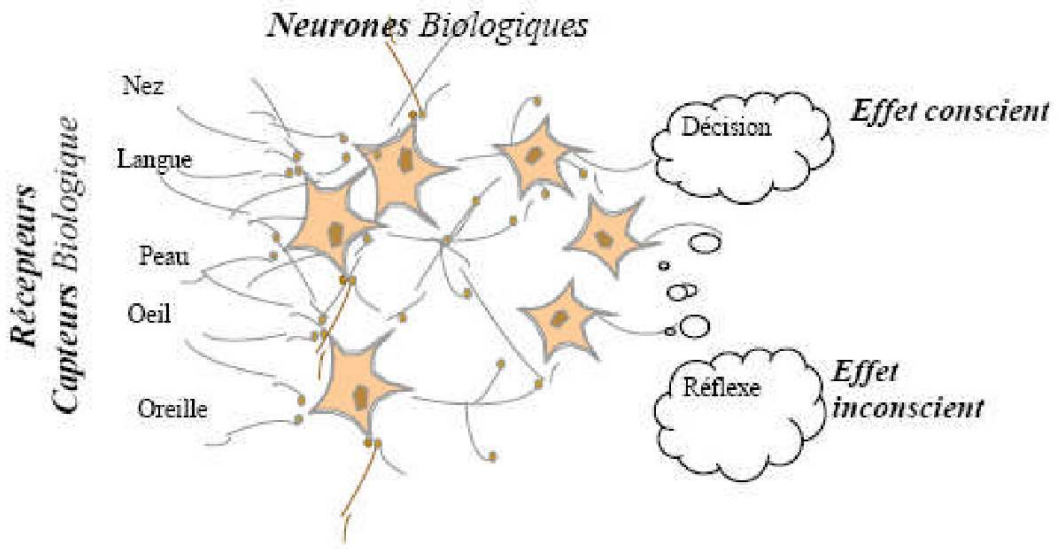


Figure II.1 Structure d'un réseau de neurones biologiques.

Les réseaux de neurones forment une famille de fonctions non linéaires, permettant de construire, par apprentissage, une très large classe de modèles et de contrôleurs. Un réseau de neurones est un système d'opérateurs non linéaires interconnectés, recevant des signaux de l'extérieur par ses entrées, et délivrant des signaux de sortie, qui sont en fait les activités de certains neurones.

II.3. Neurone biologique

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone (unique) pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de

quelques dizaines d'Angstroms (10- 9 m) entre l'axone du neurone afférent et les dendrites (on dit une dendrite) du neurone efférent. La jonction entre deux neurones est appelée la synapse (figure I.2).

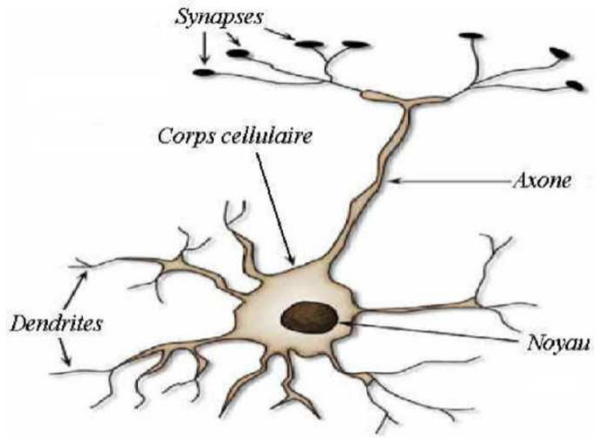


Figure II.2 Schéma simplifié d'un neurone biologique.

Les réseaux de neurones artificiels (Artificial Neural Networks – ANN) constituent une approche fondamentalement nouvelle dans le traitement de l'information. Ce sont des systèmes parallèles, adaptatifs et distribués dont le fonctionnement imite celui des réseaux de neurones biologiques tout en reproduisant leurs caractéristiques de base :

1. La connaissance est acquise par le réseau à travers un processus d'apprentissage ;
2. Les connexions entre neurones, appelées poids synaptiques, sont utilisées pour le stockage de la connaissance.

Du point de vue structurel, un réseau de neurones est d'un certain nombre d'unités de traitement simples appelées neurones formels ou artificiels. Ces derniers sont connectés entre eux de façon à produire la réponse correspondant aux entrées reçues par le réseau. Plusieurs modèles de neurones artificiels ont été développés, s'inspirant du principe de fonctionnement du neurone biologique qui assure essentiellement les fonctions suivantes :

- Réception des signaux provenant des neurones voisins ;
- Intégration de ces signaux ;
- Génération d'une réponse ;
- Transmission de celle-ci à d'autres neurones.

II.4. Eléments de base

II.4.1. Structure de base

Le premier modèle du neurone formel date des années quarante. Il été présenté par Mc Culloch et Pitts. S'inspirant de leurs travaux sur les neurones biologiques ils ont proposés le modèle suivant:

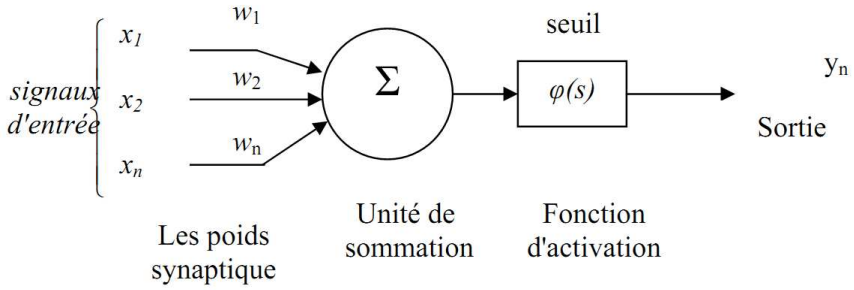


Figure II.3 le neurone formel

Un neurone formel est une fonction algébrique non linéaire et bornée, dont la valeur dépend des paramètres appelés poids synaptiques ou poids des connexions. D'une façon plus générale, un neurone formel est un élément de traitement (opérateur mathématique) possédant n entées (qui sont les neurones externes ou les sorties des autres neurones), et une seule sortie. Ce modèle est décrit mathématiquement par les équations suivantes:

$$s = \sum_{i=1}^{n} w_i x_i$$

$$y_n = \varphi(s)$$

Où x_i, w_i, φ et y_n sont respectivement, les entrées, les poids synaptiques, la fonction d'activation et la sortie du neurone.

II.4.2 Fonction d'activation

Les fonctions d'activations représentent généralement certaines formes de non linéarité. L'une des formes de non linéarité la plus simple, et qui est appropriée aux réseaux discret, est la fonction signe, figure II.3.a. Une autre variante de ce type des non linéarités est la fonction de Heaviside, figure II.3.b. Pour la majorité des algorithmes d'apprentissage il est nécessaire d'utiliser des fonctions sigmoïdes différentiables, telles que la fonction sigmoïde unipolaire, figure II.3.c et la fonction sigmoïde bipolaire, figure II.3.d. La classe, la plus utilisée des fonctions d'activation, dans le domaine de la modélisation et de la commande des systèmes non linéaires est la fonction sigmoïde bipolaire.

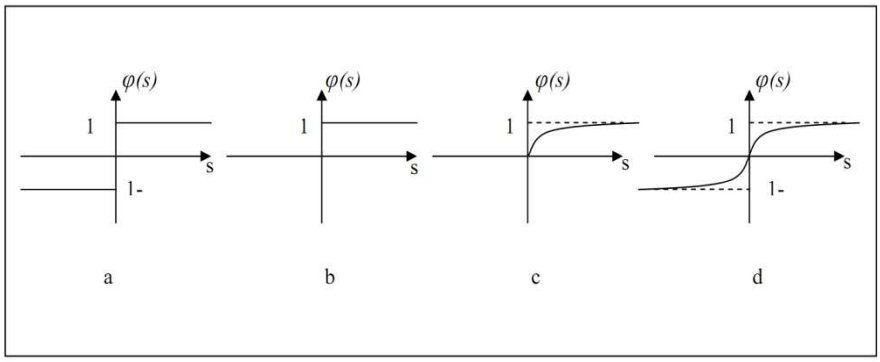


Figure II.4 Différentes fonctions d'activation

Donc un réseau de neurone est un ensemble d'éléments de traitement de l'information, avec une topologie spécifique d'interconnexions (architecteur du réseau) entre ces éléments et une loi d'apprentissage pour adapter les poids de connexions (poids synaptiques), il est caractérisé par un parallélisme à gain très fin et à forte connectivité. Nous entendons par là que dans un réseau de neurones donné, l'information est traitée par grand nombre de processeurs élémentaires très simples, chacun étant relié à d'autres processus. Ce processus très simple est un neurone formel désigné ainsi son fonctionnement s'inspire d'une modélisation des cellules biologiques. Ils sont dotés de deux propriétés importantes qui sont à l'origine de leur intérêt pratique des domaines très divers:

- Capacité d'adaptation ou d'apprentissage qui permet au réseau de tenir compte des nouvelles contraintes ou de nouvelles données du monde extérieur,

- Capacité de généralisation qui est son aptitude de donner une réponse satisfaisante à une entrée qui ne fait pas partie des exemples à partir desquels il apprit.

II.5. L'apprentissage des réseaux de neurones

L'information que peut acquérir un réseau de neurones est représentée dans les poids des connexions entre les neurones. L'apprentissage consiste donc à ajuster ces poids de telle façon que le réseau présente certains comportements désirés. En d'autres termes, l'apprentissage des réseaux de neurones consiste à ajuster les poids synaptiques de telle manière que les sorties du réseau soient aussi proches que possible des sorties désirées. Il existe deux types d'apprentissage:

• L'apprentissage supervisé: pour lequel on dispose de la sortie désirée et qui consiste à ajuster les poids synaptiques de telle sorte à minimiser l'écart entre la sortie désirée et la sortie du réseau,

• L'apprentissage non supervisé: pour lequel le réseau de neurones organise lui-même les entrées qui lui sont présentées de façon à optimiser un critère de performances donné,

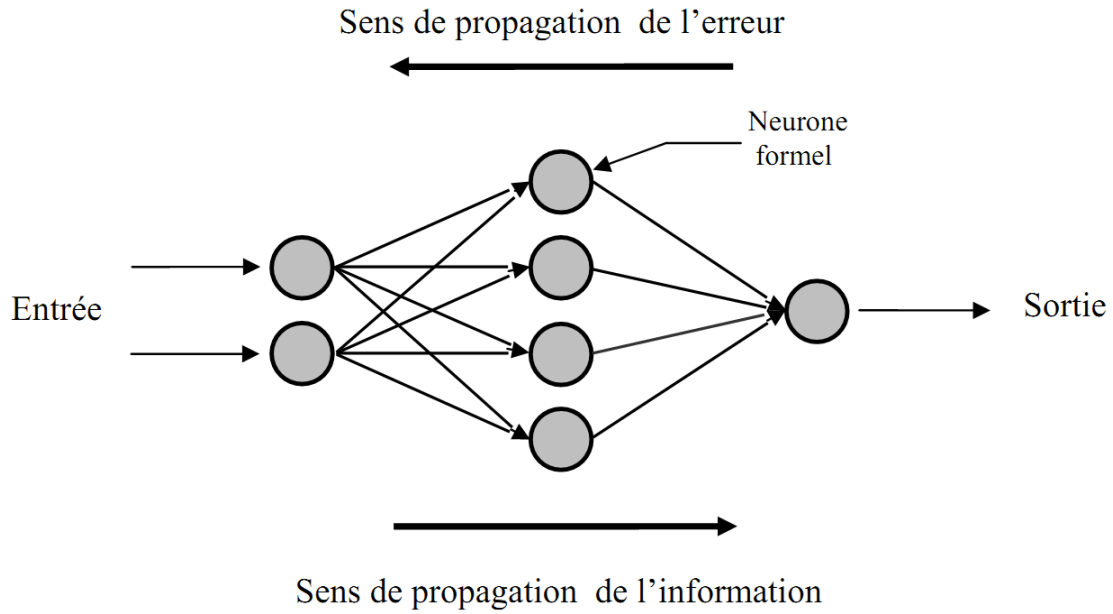


Figure II.5 Un réseau multicouche comportant 2 neurones d'entrée, 4 neurones cachés et un neurone de sortie

II.5 Algorithme d'apprentissage

L'apprentissage des réseaux de neurones, consiste à adapter les poids synaptiques de telle manière que l'erreur entre la sortie du réseau et la sortie désirée soit aussi petite que possible. La plupart des algorithmes d'apprentissage des réseaux de neurones est basée sur les méthodes du gradient: ils cherchent à minimiser un critère de la forme suivante:

$$J = \frac{1}{2}\sum_{q=1}^{m}\left(y_{rq}\left(X_p\right) - y_q^d\left(X_p\right)\right)^2$$

Où y_{rq} et y_q^d sont la sortie du réseau et la sortie désirée pour le vecteur d'entrée Xp. Cette optimisation se fait d'une manière itérative, en modifiant les poids en fonction du gradient de la fonction de coût selon la loi d'adaptation suivante:

$$w_{ji}^l(k+1) = w_{ji}^l(k) - \eta\frac{\partial J}{\partial w_{ji}^l}$$

Où w_{ji}^{l} est le poids de la connexion entre le $j^{ème}$ neurone de la couche *l* et le $i^{ème}$ neurone de la couche *l-1*

$i=1,\ldots,n+1$ la $i^{ème}$ composante du vecteur d'entrée,
$j=1,\ldots,m+1$ la $i^{ème}$ composante du vecteur de sortie,
$l=1,\ldots,L$ l'ordre d'une couche dans le réseau de neurone,

η est une constante positive ($\eta>0$) appelée taux d'apprentissage.
Le gradient est calculé par une méthode spécifique aux réseaux de neurones, dites méthode de rétro propagation. Dans cette méthode il est possible de calculer le gradient de la fonction coût en retropropageant l'erreur commise en sortie vers les couches cachées.

II.6 Identification et commande des systèmes par les réseaux de neurones

Par leur capacité d'approximation universelle, les réseaux de neurones sont bien adaptés pour l'identification et commande des systèmes non linéaires. En effet dans ce cas la fonction commande est une fonction non linéaire, l'objectif est alors d'approximer cette fonction par les RNA (réseaux de neurones artificiels). Cette approximation est réalisée par apprentissage des poids du réseau, l'apprentissage peut se faire hors ligne ou en ligne :

- Dans le cas de hors ligne, l'apprentissage est basé sur un ensemble de données définissant la fonction commande,
- Dans le cas d'en ligne, la mise à jour des poids est essentiellement adaptative.

II.6.1. Identification des processus par réseaux de neurones

Le principe de l'identification par réseau neuronaux consiste à substituer aux modèles paramétriques classiques des modèles neuronaux, c'est-à-dire proposer un modèle établissant une relation entre son entrée et sa sortie et à déterminer, à partir du couple des signaux d'entrée-sortie, le comportement du modèle. Deux raisons importantes nous motivent:

- Prédire le comportement d'un système pour différentes conditions de fonctionnement ;
- Elaborer une loi de commande à appliquer au processus, pour qu'il réalise l'objectif assigné.

Nous citerons deux techniques d'identification à base de réseaux de neurones multicouches : la méthode d'identification directe et la méthode d'identification inverse.

II.6.1.1. Identification directe

La figure II.6 montre le schéma général d'identification directe d'un processus. Sur cette figure, le réseau de neurones identificateur RNI est utilisé en parallèle avec un processus de type boite noire. La sortie du processus, *y*, est comparée avec la sortie du réseau de neurones, $\hat{y}$, puis l'erreur $y-\hat{y}$ est utilisée afin d'affiner les paramètres du système neuronal.

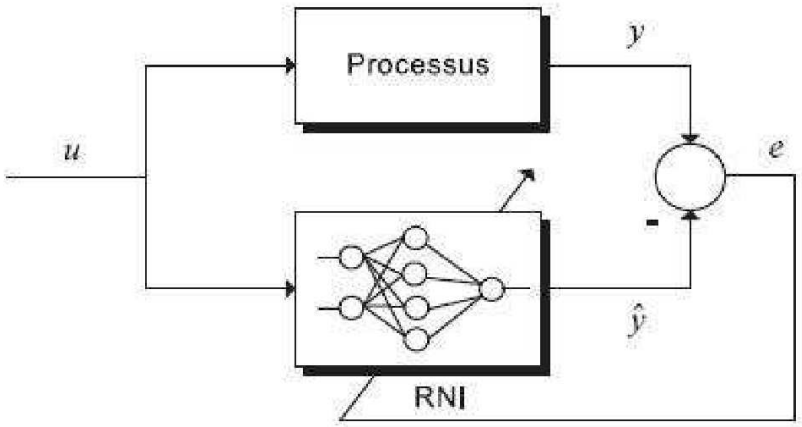


Figure II.6 Schéma d'identification directe d'un processus par réseau de neurones.

Considérons un système non linéaire avec une entrée *u(k)* et une sortie *y(k)*

y(k) peut dépendre de *u(k)* seulement, ou de *u(k)* et les états précédents de y et/ou *u* c'est à dire

y(k)=F(u(k))

Ou

y(k)=F(u(k-1),…,u(k-m),y(k-1),y(k-2),…,y(k-n))

Le principe de l'identification directe par réseau de neurones consiste à construire le réseau (en générale MLP) qui approxime la fonction F, cette approximation se fait par apprentissage hors ligne.
Etant donné la fonction F, il faut construire un RNA pour réaliser l'approximation, on génère une base de données de *N* échantillon *y(k-1), y(k-2),….,y(k-n), u(k-1),…,u(k-m)* (les entrés) et on obtient *y(k)* (la sortie) à partir de F. L'objectif de l'apprentissage est de déterminer la fonction F à partir de ces données. A l'instant t, on donne au RNA :
y(k-1), y(k-2),….,y(k-n), u(k-1),…,u(k-m) et on obtient à la sortie $\hat{y}(k)$. La mise à jour des poids du réseau est basée sur la minimisation de l'erreur entre *y(k)* et $\hat{y}(k)$.

$$J = \frac{1}{2}\sum_{i=1}^{m} y_i - \hat{y}_i$$

II.6.1.2. Identification inverse

Dans cette méthode, l'entrée du processus est comparée avec la sortie de l'identificateur neuronal RNI est la sortie du processus est injectée comme entrée du réseau de neurones (figure II.7). Après un apprentissage hors-ligne du modèle inverse, le RNI peut être configuré afin d'assurer un contrôle directe du processus.

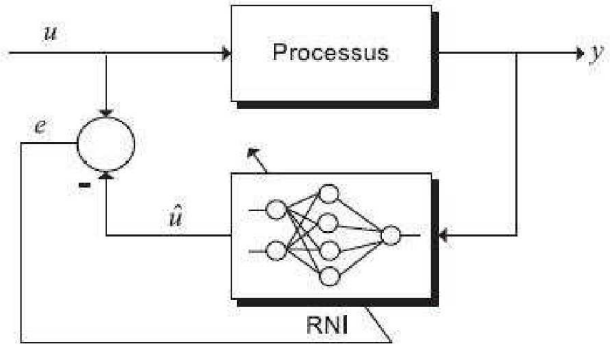


Figure II.7 Schéma d'identification inverse d'un processus avec un réseau de neurones.

I.6.2. Commande des processus par réseaux de neurones

La littérature scientifique fait mention de différentes architectures de commande. Les plus simples se basent sur l'apprentissage d'un contrôleur conventionnel déjà existant, d'autres opèrent un apprentissage hors-ligne du modèle inverse du processus ou d'un modèle de référence et enfin, d'autres travaillent complètement en ligne.

I.6.2.1. Apprentissage d'un contrôleur conventionnel

Un réseau de neurones peut reproduire le comportement d'un contrôleur conventionnel déjà existant (PI, PID, RST, ...) grâce à ses facultés d'apprentissage et d'approximation. Il suffit de le soumettre à un apprentissage hors ligne pendant une phase d'identification directe en considérant que le contrôleur est lui-même un processus. La figure II.7 montre le principe de l'identification directe d'un contrôleur conventionnel.

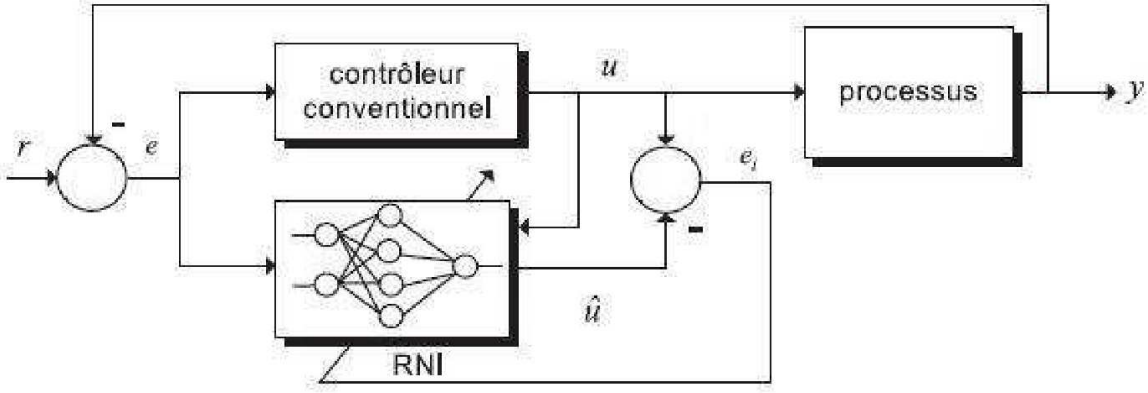


Figure II.7 Schéma d'identification directe d'un contrôleur conventionnel avec un RNI.

Le but de cette architecture n'est pas de perfectionner les performances du contrôleur conventionnel déjà existant, mais de s'affranchir des contraintes d'implémentations matérielles que peuvent nécessiter certains régulateurs. La méthode de régulation de type RST par exemple est reconnue pour ses bonnes performances en commande mais elle pose de sérieux problèmes en intégration numérique.

I.6.2.2. Commande inverse avec apprentissage en ligne

Le principe de cette commande repose sur une d'identification par modèle inverse. La figure II.8 représente le schéma de commande inverse avec un (RNC). Cette architecture reprend le même principe que celui de l'identification inverse montrée dans la figure II.8. En effet, L'entrée de référence *r* est comparée à la sortie y du processus pour former l'erreur de poursuite, $e = r - y$ qui sert

à modifier les paramètres du réseau en ligne. Après avoir appris le modèle inverse, le neuro-contrôleur délivre la sortie *u* du RNC qui est la commande injectée en entrée du processus, l'erreur est alors nulle et la sortie *y* est égale à la référence *r*. Ce principe est identique au RNI de la figure II.6 ou lorsque l'apprentissage du modèle inverse est accompli, la sortie du RNI est égale à l'entrée du processus. L'avantage de la commande inverse avec un RNC est le suivi en temps réel de l'évolution du processus, car l'apprentissage est réalisé en ligne.

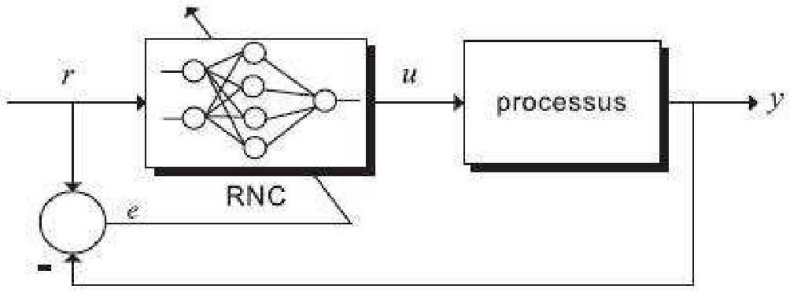


Figure II.8 Schéma de commande inverse avec un RNC.

Comme les performances de cet organe de commande dépendent étroitement de la fidélité du modèle inverse, la stabilité et le niveau de performance ne sera pas garanti dans le cas ou le modèle inverse n'existe pas ou s'il est difficile à trouver.

II.7. Avantages et Inconvénients des réseaux de neurones

II.7.1. Avantages des réseaux de neurones

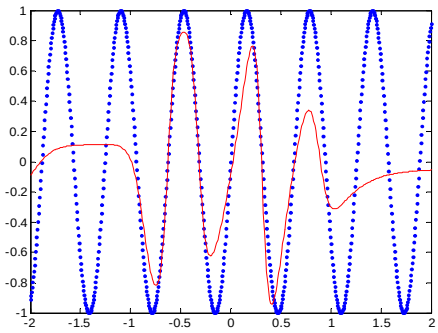
• Capacité de représenter n'importe quelle fonction, linéaire ou pas, simple ou complexe ;
• Faculté d'apprentissage à partir d'exemples représentatifs, par" rétropropagation des erreurs". L'apprentissage (ou construction du modèle) est automatique ;
• Résistance au bruit ou au manque de fiabilité des données ;
• Simple à manier, beaucoup moins de travail personnel à fournir que dans l'analyse statistique classique. Aucune compétence en matis, informatique statistique requise ;
• Comportement moins mauvais en cas de faible quantité de données ;
• Pour l'utilisateur novice, l'idée d'apprentissage est plus simple à comprendre que les complexités des statistiques multivariables.

II.7.2. Inconvénients des réseaux de neurones

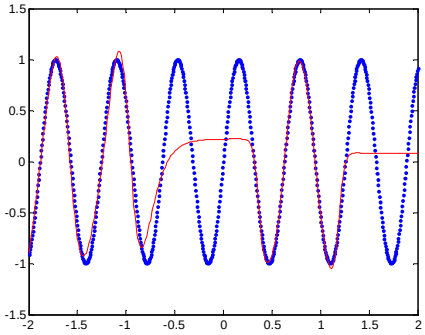
• L'absence de méthode systématique permettant de définir la meilleure topologie du réseau et le nombre de neurones à placer dans la (ou les) couche(s) cachée(s) ;
• Le choix des valeurs initiales des poids du réseau et le réglage du pas d'apprentissage, qui jouent un rôle important dans la vitesse de convergence ;
• Le problème du sur-apprentissage (apprentissage au détriment de la généralisation) ;
• La connaissance acquise par un réseau de neurone est codée par les valeurs des poids synaptiques, les réseaux de neurones sont donc des boites noires où les connaissances sont inintelligibles pour l'utilisateur.

Exemple 1 :

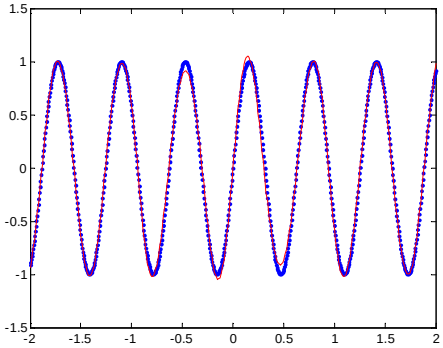
Le but de cet exemple est de mettre en place un réseau de neurones qui contient une couche cachée de 10 neurones et une couche de sortie de 1 neurone, on utilisant la rétropropagation du gradient avec une base d'apprentissage qui contient 70 point, pour élaborer un programme d'identification de la fonction $f(x)=\sin(x) \quad -2 \le x \le 2$.



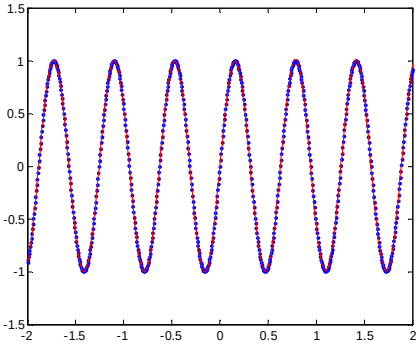
Apprentissage avec 5 itérations



Apprentissage avec 50 itérations



Apprentissage avec 100 itérations



Apprentissage avec 1000 itérations

Exemple 2 :

Soit le système discret donné par la fonction de transfert suivante :

$$H(z)=z^{-1}\frac{b_1+b_2 z^{-1}}{1-a_1 z^{-1}}$$

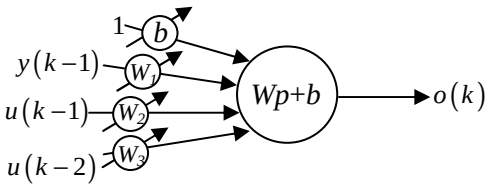
Avec

$b_1=1$, $b_2=-0.8$, $a_1=-0.5$

et l'équation de récurrence liant la sortie $y(k)$ à l'entrée $u(k)$

$$y(k)=a_1\, y(k-1)+b_1\, u(k-1)+b_2\, u(k-2)$$

Le but de cet exercice est l'identification des paramètres (b_1, b_2, a_1) on utilisant une base d'apprentissage qui contient 500 points d'entrés sortie ($u(k)$,$y(k)$) et un résaux de neurone exprimé par la figure suivante :



Les entrés du réseau $p=\left[\, y(k-1) \quad u(k-1) \quad u(k-2) \,\right]$

L'apprentissage du réseau consiste à modifier, a chaque pas les poids et les biais afin de minimiser la somme des carrés des erreurs en sortie.

A chaque pas d'apprentissage, l'erreur en sortie est calculée comme la différence entre la cible recherchée $y(k)$ et la sortie $o(k)$ du réseau.

La quantité à minimiser, a chaque pas d'apprentissage *k* est :

$$J=\frac{1}{2}e^T(k)e(k)=\frac{1}{2}\left(y(k)-o(k)\right)^T\left(y(k)-o(k)\right)$$

L'adaptation des poids se faisant dans le sens inverse du gradient, la matrice de poids de l'étape future (*t*+1) est :

$$W(t+1)=W(t)-\eta\frac{\partial J}{\partial W}$$

$$\frac{\partial J}{\partial W}=\frac{\partial J}{\partial o(k)}\frac{\partial o(k)}{\partial W}$$

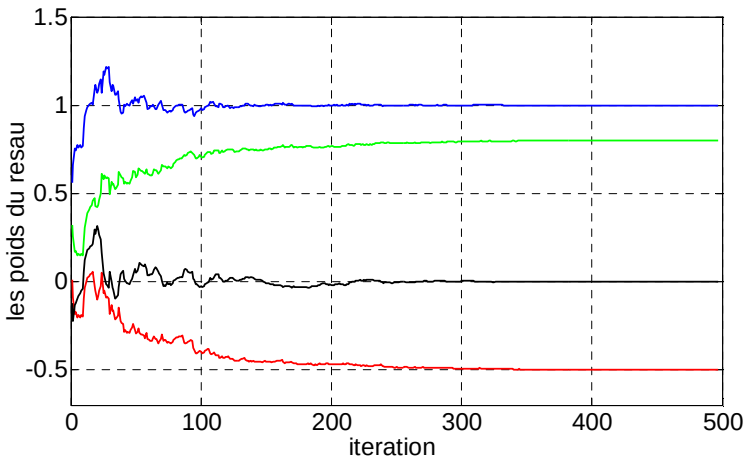
$$\frac{\partial J}{\partial o(k)}=o(k)-y(k)$$

$\frac{\partial o(k)}{\partial W}=\frac{\partial\left(W\ p(k)+b\right)}{\partial W}=p^T(k)$ Avec $p(k)$ désigne les entrés du réseau

$W(t+1)=W(t)+\eta\left(y(k)-o(k)\right)p^T(k)$ Avec η désigne les entrés du réseau

De même, on obtient l'expression de la modification du biais :

$$b(t+1)=b(t)+\eta\left(y(k)-o(k)\right)$$



On remarque que les poids du réseau convergent vers les paramètres (b_1, b_2, a_1) du système